DISSERTATION

SIRE SELECTION FOR PROFIT IN POSTWEANING BEEF PRODUCTION

Submitted by

Matthew Alan Cleveland

Department of Animal Sciences

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2006

UMI Number: 3226116

UMI Microform 3226116

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

COLORADO STATE UNIVERSITY

January 3, 2006

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY MATTHEW ALAN CLEVELAND ENTITLED SIRE SELECTION FOR PROFIT IN POSTWEANING BEEF PRODUCTION BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Garrick

Dr. Dorian Garrick

Dr. J. Daryl Tatum

Wendy J Umberger

Dr. Wendy Umberger

R Mark Enns

Advisor – Dr. R. Mark Enns

Department Head

ABSTRACT OF DISSERTATION

SIRE SELECTION FOR PROFIT IN POSTWEANING BEEF PRODUCTION

The objectives of this research were to: 1) develop a model of postweaning beef production that uses growth and carcass expected progeny differences (EPD) from random regression models to predict the profitability of alternative sires; 2) evaluate expected profit outcomes and the impact on profit of not accounting for growth and carcass genetic differences when feeding progeny of alternative sires to a constant endpoint; 3) evaluate and compare profit and profit variability accounting for growth and carcass genetic differences, and the impact on expected profit of not accounting for these differences, when feeding progeny of alternative sires to an optimum endpoint; and 4) determine the impact on profit at alternative finish endpoints of single-trait selection for desirable changes in growth and carcass traits and evaluate the importance of accounting for selection when identifying finish endpoints.

An object-oriented model was developed as the basis for a postweaning decision support system (DSS) to predict the daily growth and profitability of feedlot animals, based on genetic differences in growth and carcass traits, for selection of sires considering alternative finish endpoints. Simulated growth was a function of components describing nutrition, environment, management decisions, market conditions and genetics. Model parameters included average postweaning performance as designated by the user to identify the appropriate growth curve, which was then adjusted by an animal's genetic potential to obtain a prediction of growth, carcass traits, and profit. The DSS contained both deterministic and stochastic processes for flexibility in implementation. Validation indicated that the DSS appropriately accounted for differences in genetic potential, but there exists room for improvement in the prediction of marbling score throughout the feeding period.

The last three project objectives were addressed by simulating performance for 16 sire genotypes in the DSS. These genotypes consisted of all possible combinations of high (H) and low (L) levels for weight (consisting of available weights - birth, weaning and yearling weight), backfat thickness, marbling score and longissimus muscle area relative to average EPD in the Red Angus sire summary. Weight EPD for a specified day on feed were calculated using a random regression approach and variance components for all traits obtained from the Red Angus analysis. Average postweaning performance was determined from literature values and other model parameters were chosen to reflect a plains/high plains commercial feedlot. Median profit and distributions of profit values (n = 1,000) were calculated for each sire genotype at a 1.1 cm constant backfat thickness (FT) and the optimum finish (OPT) endpoint (point of profit maximum). Selection for improvements in growth and carcass traits was also simulated for one sire genotype at three finish endpoints.

Simulation results indicated a wide range of median profit and profit values different than would be expected considering breed or biological type averages only, suggesting that certain sire genotypes may be less profitable than expected at the FT endpoint. The results also showed that some genotypes conferred a greater probability of negative profit values compared to average. Profit increased at the OPT endpoint for all genotypes with some reduction in negative profit potential, but some sires continued to have large negative impacts on profitability. Profit also increased with selection, but not in a predictable manner and increases were not maintained with continued selection.

This research illustrates the difficulty in selecting sires for maximizing profit in the postweaning phase of beef production based on EPD alone. Accounting for genetic differences concurrently with finish endpoints is necessary to correctly predict expected outcomes allowing selection for sires with the most profitable combination of traits for a given production and marketing system.

Matthew Alan Cleveland
Department of Animal Sciences
Colorado State University
Fort Collins, CO 80523
Spring 2006

ACKNOWLEDGEMENTS

There are a number of people I need to thank for their assistance and support throughout the course of this degree, as well as my entire graduate career. All that follows would not have been possible without them.

I would first like to thank my committee for their help and guidance during the completion of this project: my advisor, Dr. Mark Enns, Dr. Dorian Garrick, Dr. Wendy Umberger and Dr. Daryl Tatum. I appreciate their belief in this project and their excitement about the research.

I wish to also acknowledge all of the graduate students that I have worked with throughout this long journey, as it is them that make completion of the journey possible. I think I learned more from my fellow students than from any other source. I would like to single out Scott and Norman for their continued help throughout my career in helping me understand what was really going on, but I have enjoyed working with each of the Breeding student in one way or another. Room 202 was definitely an interesting place to work.

Last, but certainly not least, I wouldn't be anywhere near where I am today without the support and guidance of my family. I can't thank them enough for their constant presence in my life. I'm finally done.

DEDICATION

This dissertation is dedicated to my beautiful wife, Susan and wonderful little girl, Maisie. They are my greatest joys and without them this would mean nothing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND OBJECTIVES

Countless marketing opportunities exist for producers of beef cattle. Value-based pricing systems have increased in popularity in an attempt to pass price signals to those making selection decisions to increase the producer's share of the consumer dollar. As beef producers utilize these types of systems they are faced with the challenge of efficiently growing cattle that meet the complex needs of the customer while maintaining their own profitability. An important contribution to that efficiency, in part, will be a focus on developing new selection and management tools for improving profitability in the postweaning phase of production.

The adoption of new marketing strategies has increased the focus on genetic improvement of traits important in the postweaning production phase. The industry has responded to interest in new traits by developing growth and carcass expected progeny differences (EPD) and indexes of postweaning production to aid in sire selection decisions. Producers have access to increasing amounts of information but are not provided the necessary tools to evaluate selection effects on profitability. Breeders have little context for using genetic predictions intelligently (Bourdon, 1998). Complicating the issue further is the lack of EPD for traits that directly affect revenue and/or costs of production. Additionally, even if appropriate EPD are available selection that does not account for differences in finish endpoints may have little predictable impact on profit.

To account for the differences in finish endpoints recent work has focused on developing EPD using random regression models (Kuehn, 2000; Jubileu, 2003) for traits that are economically relevant to postweaning production (Golden et al., 2000). The shift towards value-based marketing systems has increased the incentive to collect feedlot and carcass performance data and will make it feasible to produce genetic predictions using random regression models (e.g., days to finish, value at finish). The advantage of using random regression EPD is that genetic predictions can be made for any age or number of days on feed, essentially tailoring genetic improvement to individual production systems. The nature of random regression, however, negates the possibility of providing a useful EPD to producers in the traditional format (e.g., paper sire summaries) due to the large number of possible production-specific feeding endpoints. Additionally, supplying yet another EPD to an already information-overloaded industry is not likely to improve profitability or realized genetic improvement.

The EPD from random regression models can account for costs or revenue at different finish endpoints, but alone provide little context for making selection decisions when the objective is profit maximization. A decision-making framework is needed that can rank sires for profit in a given production and marketing system and can provide easily-understood phenotypic outcomes for alternative selection and management decisions in the postweaning phase. Such a framework could be used to select sires for profit and identify optimum finish endpoints, while concurrently selecting for optimum trait levels that maximize profit in a given system.

Researchers have previously developed models that incorporate specific production systems to evaluate cattle performance at alternative finish endpoints and to identify differences in optimum finish endpoints as aids to selection (Amer et al., 1994; Basarab et al., 1999; Koontz et al., 2000). Differences, in most cases, were assessed relative to breeds or biological types. There is a dearth of research that evaluates differences in performance due to differences in individual animal genetic potential for traits important to profitability in the postweaning phase. Re-ranking for profit between biological types has been demonstrated (Amer et al., 1994), but there is likely a risk of reducing profit when ignoring genetic differences within a breed or biological type, considering this re-ranking at alternative finish endpoints.

Objectives

The project objectives identified to address the issues presented above involving sire selection for profit in the postweaning phase were:

1. Develop a model of postweaning beef production that uses growth and carcass expected progeny differences from random regression models to predict the profitability of alternative sires for use in a web-based decision support system.
2. Evaluate expected profit outcomes and the impact on profit of not accounting for growth and carcass genetic differences when feeding progeny of alternative sires to a constant endpoint typical of current commercial production, using simulation.
3. Evaluate and compare profit and profit variability accounting for growth and carcass genetic differences, and the impact on profit of not accounting for these differences, when feeding progeny of alternative sires to the endpoint at which

profit is maximized versus feeding to a constant endpoint typical of commercial production, using simulation.

4. Determine the impact on profit at alternative finish endpoints of single-trait selection for desirable changes in growth and carcass traits and evaluate the importance of accounting for selection when identifying finish endpoints, using simulation.

Literature Cited

Amer, P. R., R. A. Kemp, J. G. Buchanan-Smith, G. C. Fox, and C. Smith. 1994. A bioeconomic model for comparing beef cattle genotypes at their optimal economic slaughter end point. J. Anim. Sci. 72:38-50.

Basarab, J. A., J. R. Brethour, D. R. ZoBell, and B. Graham. 1999. Sorting feeder cattle with a system that integrates ultrasound backfat and marbling estimates with a model that maximizes feedlot profitability in value-based marketing. Can. J. Anim. Sci. 79:327-335.

Bourdon, R. M. 1998. Shortcomings of current genetic evaluation systems. J. Anim. Sci. 76:2308-2323.

Golden, B. L., D. J. Garrick, S. Newman, and R. M. Enns. 2000. Economically relevant traits, a framework for the next generation of EPDs. Proc. Beef Improvement Federation, Wichita, KS, pp 2-13.

Jubileu, J.S. 2003. The use of random regression models to predict days to finish in beef cattle. Thesis. Colorado State University, Fort Collins.

Koontz, S. R., D. L. Hoag, J. L. Walker, and J. R. Brethour. 2000. Returns to sorting and market timing of animals within pens of fed cattle. Western Agricultural Economics Association Annual Meetings, Vancouver, BC.

Kuehn, L. A. 2000. Parameterization of random regression models for beef cattle data sets. Thesis. Colorado State University, Fort Collins.

CHAPTER II

GROWTH AND COMPOSITION OF CATTLE IN THE POSTWEANING PHASE: REVIEW OF LITERATURE

Introduction

Growth and development of livestock species and corresponding changes in carcass composition have been widely studied in the past five decades. An understanding of the ways in which livestock, and especially beef cattle, grow from birth to maturity is integral to making good management decisions in all phases of livestock production. Beef cattle growth and changes in carcass composition are determined by numerous endogenous and exogenous factors and knowledge of their impact on development is a necessary step in the process of manipulating or predicting cattle growth to improve profitability and to produce a superior product. The following reviews literature concerning the growth of beef cattle from birth to maturity with a particular emphasis on the phase of production from weaning through normal harvest age. Factors affecting growth and development of feedlot cattle are highlighted.

Cattle growth and development

Growth and development can be generally defined as “directed biosynthesis”. Development specifically refers to the coordination of various significant processes to yield an adult organism, while growth indicates the production of new biochemical units.

All growth is not necessarily development, but it is the aspect of development concerned with an increase in a living substance (Brody, 1945). The production of new cells during growth includes cell multiplication (hyperplasia) and cell enlargement (hypertrophy). Also contributing to growth is the incorporation of materials from the environment (Brody, 1945; Owens et al., 1993). Growth occurs in the pre-natal animal as hyperplasia and as hypertrophy in the post-natal animal for the majority of tissues, with the exception of adipose tissue, organs and a few others that continue to exhibit cell division throughout life (Marple, 2003).

There are many complex processes that regulate growth and development of tissues. The main concerns of this review are the "principles" that govern these processes for tissues of economic importance to the livestock industry and how tissues change under normal production circumstances. The most general operational definition of growth is a progressive increase in weight (Brody, 1945). This definition is somewhat satisfying as the notion of increases in animal weight or mass is intuitively related to increases in cell number and size, but a more specific description of growth is needed. There are two aspects that generally comprise and can define the growth process. The first is an increase in the overall mass of an animal that can be described by an individual growth curve. The second are the changes in composition that result from differential growth of component parts of the body (Fowler, 1968; Mukhoty and Berg, 1971).

In the remainder of the discussion, growth will be used generally to mean a change in body weight coupled with changes in composition; however the specifics of each change will be addressed individually.

The growth curve. In general, animal species grow in a predictable manner. Brody (1945) showed that lifetime growth can be represented by a sigmoid curve, also called a size-age curve as it describes changes to some measurement of size with changes in age. Figure 2.1 shows a generalized growth curve and includes points that divide the curve in two distinct segments: a self-accelerating phase occurs prior to puberty; an inflection point (b); and a post-pubertal self-inhibiting phase. The shape of the curve is determined by opposing forces that act in each of these phases (Brody, 1945).

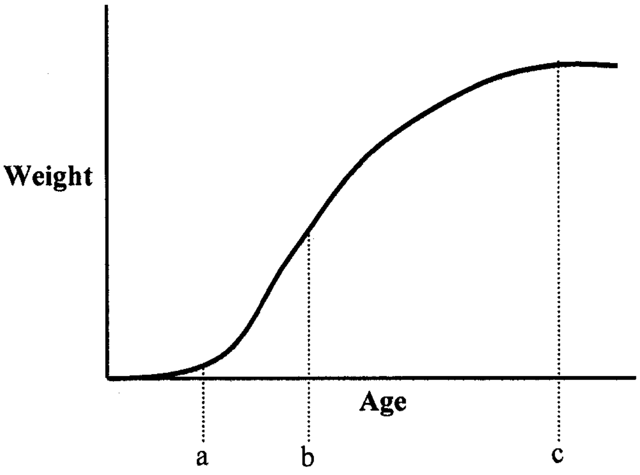

Figure 2.1. Generalized growth curve showing the sigmoid shape. Points represent a) birth, b) the inflection point (puberty), and c) maturity. Adapted from Owens et al., 1993.

The most rapid growth, proportionally, occurs after birth (a) until the animal's body weight is twice its birth weight. Growth continues rapidly to puberty (self-accelerating phase), at which point the increase in growth velocity ceases, but the decrease in growth rate has not yet begun. The point of inflection in livestock generally occurs when 30% of mature weight is reached (Brody, 1945), beginning around six months of age in cattle. After the inflection point an animal continues to increase in size, but at a decreasing rate (self-inhibiting phase). Maturity (c) is reached when an animal essentially ceases to increase in size (Marple, 2003).

The sigmoid shape of the growth curve is considered a generalization of the average growth of a population. The curve for an individual animal will not be nearly as smooth and for livestock marketed at normal weights and/or composition, measured growth from early life will be nearly linear. Brody (1945) suggested that the growth curve be separated into segments for analysis due to the distinct phases of interest in livestock production (e.g., birth to weaning, weaning to harvest).

A number of growth functions have been developed to quantify the growth process in beef cattle represented by the sigmoid curve. The intent of these functions is generally to predict weight or size values at specific points along the growth curve, such as at weaning, yearling, maturity, etc. A number of functions model beef cattle growth well, but each has advantages and disadvantages when considering stage of growth, nature of the data or model complexity. The details of individual models are beyond the scope of this review, but Brown et al. (1976) compared five nonlinear growth models for describing weight-age relationships in beef cattle. Table 2.1 lists each of these models and the relative appropriateness of each, in terms of goodness of fit.

Other researchers have agreed with the conclusions of Brown et al. listed in Table 2.1. Goonewardene et al. (1980/1981) using beef cattle data and Perotto et al. (1992) using dairy cattle data found poor fits in early stages using a logistic model. The Von Bertalanffy function was found to overestimate early weights (Goonewardene et al., 1980/1981) as was the Gompertz function (Perotto et al., 1992). All three studies agreed that the Richards function best fit the data over all ages, but was especially useful in early stages of growth, a characteristic lacking in the other models. In all cases the Brody function was considered to be computationally less complex and had only a slightly worse fit than the Richards model, but was less appropriate prior to weaning. The fit of any function varies throughout the growth period and is an important consideration when selecting an appropriate model.

Table 2.1. Comparison of the appropriateness of five functions used to describe multiple phases of growth in beef cattle[a].

Model	Advantage	Disadvantage
Von Bertalanffy	Fit reasonably well over all ages	Overestimated weights prior to six months of age
Brody	Fit observed data well Computationally simple	Under- or overestimated weights before six months of age
Gompertz		Overestimated early weights
Logistic		Overestimated early weights Underestimated mature weights
Richards	Flexible and slightly more accurate due to additional parameters Provided a generally unbiased fit at all ages Especially useful prior to ten months of age	Computationally more complex

[a]Results from Brown et al., 1976

There has long been the desire by livestock producers to alter the growth curve for improved biological, and hence, economic efficiency of production. In a review of growth curve analysis Fitzhugh Jr. (1976) enumerated several reasons for altering the shape of the growth curve. The first was to reconcile the genetic antagonism between the desire for rapid, efficient growth and the need to maintain smaller parents for reduced maintenance. Second was the need to improve efficiency though increased maturation rates. Third was reducing dystocia by reducing birth weights, while achieving increased post-weaning gain. The final reasons for altering the growth curve were to decrease the age at puberty as a perceived reproductive improvement and reduce the amount of carcass fatness at preferred market weights. Any alteration of the growth curve, however, will have implications not just for weight and size, but for the relative growth of body tissues. Any decisions regarding changes in growth should be weighed against the costs and benefits of changing body composition.

Relative growth of tissues. Animal growth is an allometric process; that is growth of all body tissues does not proceed in the same manner or at the same rate but are relative to overall changes in mass. The three major tissues that comprise the body: bone, muscle and fat, change proportionally as the weight of the animal increases. It is well known that these tissues develop preferentially at different stages in the maturing process of an individual beginning with bone, then muscle and finally fat (Fowler, 1968). The relative proportions of these tissues in the animal carcass will determine the economic value of that carcass at harvest. Berg and Butterfield (1966) generally defined the point of optimum tissue proportions in slaughter cattle as "...when the proportion of muscle is at a maximum, bone is at a minimum and fat is at an optimum which is determined by local consumer preferences". Clearly, the ability to predict carcass composition at a particular stage of growth is important in improving beef production in the postweaning phase.

The following section will review the general growth of bone, muscle and fat in beef cattle from birth to maturity. The various factors that may affect this growth are addressed in a later section.

Bone. The functions of bone in the mammal include: a supportive framework for the body - including points of attachment for muscles; a store for minerals; and protection of internal organs (Farquharson, 2003). Development of bone begins in the antenatal animal and at birth bone is a larger proportion of the carcass in cattle than at any other time (Berg and Butterfield, 1968). In the postnatal animal, bone growth has been shown to follow a specific growth gradient in the body, anterior to posterior in the axial skeleton and distal to proximal in the limbs (Shahin and Berg, 1985b), supporting Hammond's (1932) theory. The cranium matures earliest in the axial skeleton, while the metacarpus and metatarsals mature earliest in the fore- and hind-limbs.

To more satisfactorily describe the growth of a tissue in allometric terms Berg and Butterfield (1966) calculated growth coefficients (*b*) for bone, muscle and fat (Figure 2.2). The growth patterns of each tissue can be classified using these coefficients where: *b*>1 indicates faster growth; *b*=1 indicates average growth; and *b*<1 indicates slower growth, relative to other tissues or tissue components. Bone growth is continuous with age, but at a lower proportional rate than other tissues. Bone growth appears to be generally monophasic (Berg and Butterfield, 1968; 1976) (Figure 2.3) and proceeds at a low impetus; i.e., its increase is

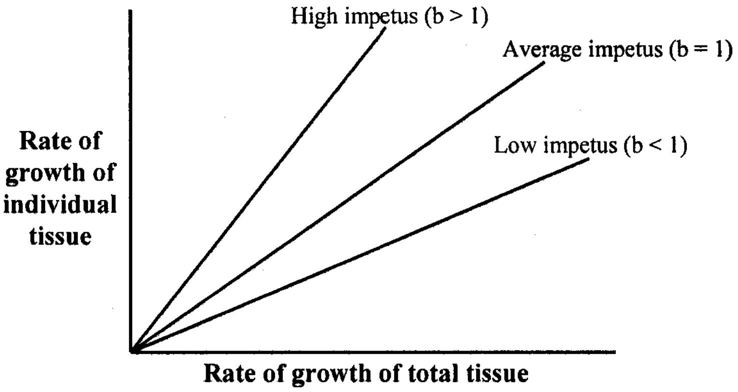

Figure 2.2. Monophasic growth patterns of body tissues, where *b* is the growth coefficient for a particular tissue or specific component of a tissue. Adapted from Butterfield, 1975b.

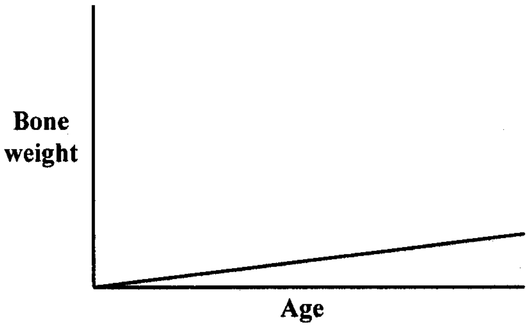

Figure 2.3. A general illustration of bone growth in cattle from birth to a typical harvest age.

proportionally less than other tissues. Monophasic growth can refer to a specific point in the growth of a tissue in which a growth coefficient is estimated, or in the case of bone, cumulative growth to maturity is considered one phase. Bone growth, then, is low impetus throughout the growth process and has a growth coefficient of less than one, indicating that the growth of all tissues is occurring faster than bone growth.

Muscle. Muscle is the most prevalent tissue in an animal's body and is essential, biologically, for animal mobility and as an amino acid store. Economically, muscle is the principal part of the carcass desired by consumers (Allen and Goll, 2003). The process of muscle development in cattle can be divided in four phases (Berg and Butterfield, 1976). The first phase is antenatal, where muscle growth is predominantly determined by genetic influences. The second phase is immediately postnatal and reflects the major influence of function on muscle growth. The third phase encompasses growth from prepubertal ages to adolescence and is a phase of minimal differential muscle development. Finally, the maturing phase is again influenced by genetics and the production of hormones which are modified by function, rather than the development of muscle being dependent on function. The greatest change in the relative weight of individual muscles or muscle groups occurs in the early phases (Butterfield, 1965; Berg and Butterfield, 1966).

Similar to the bone growth gradient discussed previously, growth of muscle occurs differentially for major muscle groups of the body. Muscles of the distal thoracic and pelvic limb and muscles of the proximal thoracic limb are considered early-developing. Abdominal and proximal pelvic limb muscles are late-developing (Butterfield, 1965).

Overall muscle growth is essentially monophasic, but occurs later in the development process than growth of bone (Figure 2.4). From birth to maturity muscle has a higher growth impetus than bone, i.e., the increase in muscle is proportionately larger than the increase in bone (Berg and Butterfield, 1968). Average growth of all muscle groups is generally considered to be of average impetus (or slightly above average) relative to overall tissue growth (Berg and Butterfield, 1966; Mukhoty and Berg, 1971) (Figure 2.3).

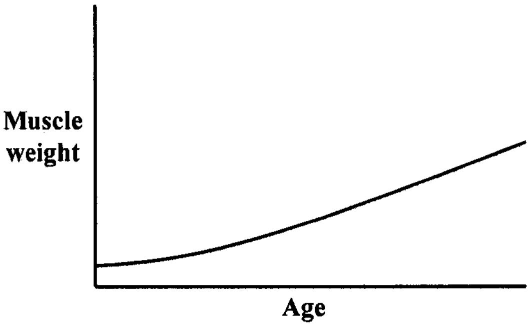

Figure 2.4. A general illustration of muscle growth in cattle from birth to a typical harvest age.

Though the muscle growth impetus is average, some individual muscles and muscle groups develop in a diphasic manner and follow differential growth patterns (Berg and Butterfield, 1968; Butterfield, 1975b). Butterfield and Berg (1966a) classified individual muscle growth in three categories. In this first category, muscles grow in a continuous, monophasic manner and may be identified as high, average or low growth impetus. The second category contains muscles that show a diphasic growth pattern, with a change in growth impetus sometime during development. Muscles in the third category show variable growth patterns and are not defined. Butterfield and Berg found 26 muscles conforming to the first category, 30 to the second and 13 to the third, over five growth phases from birth to more than 1,400 days of age in cattle.

The most common diphasic growth patterns for muscle in cattle are low/average, high/average and average/high (Figure 2.5). The first word (low, average or high) indicates the relative growth impetus in early life, while the second represents later growth. Most muscles finish their differential growth early in postnatal life (before birth weight has doubled) and then grow at an average impetus (Berg and Butterfield, 1968). The *longissimus dorsi*, an important component of the USDA yield grade formula, has been classified as high/average in regards to growth. It has a high impetus for growth in early life (0 to 84 days), but average impetus thereafter (Butterfield and Berg, 1966b).

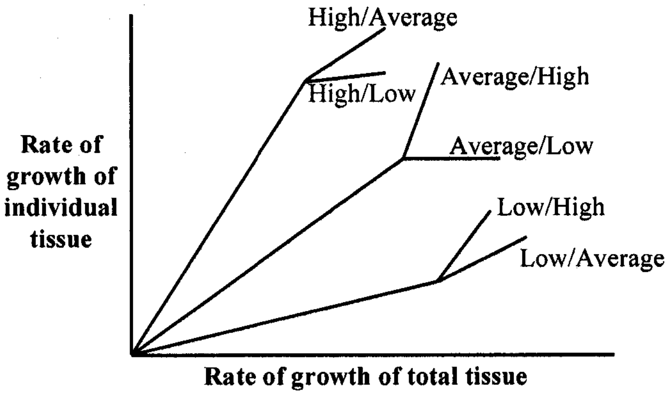

Figure 2.5. Diphasic growth patterns of individual muscles. From Butterfield, 1975b.

Muscle growth, relative to growth impetus, generally increases from the distal to proximal limbs and in the anterior direction on the dorsal line. Ventral growth moves centripetally from the anterior and posterior directions, with the greatest impetus in the abdominal muscle group (Butterfield and Berg, 1966b; Jones et al., 1980c). Jones et al. examined growth for the muscle groups as wholesale cuts and found the common growth coefficient (the sum muscle weight for wholesale cuts compared to the carcass side muscle weight) for the higher-priced cuts (round, sirloin, loin and rib) to be less than one. Individually, only the rib had a high growth impetus. They concluded that increased muscle development leads to a decline in the proportion of muscle in the high-priced cuts.

To achieve optimum tissue proportion (Berg and Butterfield, 1966) muscle growth impetus should be increased to develop the maximal amount of muscle given a specific body composition. Even more important is increasing the growth coefficient for the high-priced cuts, though Butterfield (1975b) argues that "muscle weight distribution is an unimportant feature of animal composition". There are many factors and interactions to consider, however, that make these decisions complex.

Fat. Unlike bone and muscle, fat does not seem to have an effect on function during growth and therefore its accumulation and distribution are not as orderly as other tissues (Jones et al., 1980a). Fat is a late-developing tissue and in late stages grows at a rapid rate relative to both bone and muscle. Similar to previous discussion, fat also follows a general growth gradient in its accumulation. Callow (1962) showed that fat deposition (as a percentage of total fat) proceeds proximally from the distal limbs during growth.

The deposition of total fat is diphasic as development proceeds in a linear manner from early postnatal life, but increases rapidly as the animal matures (Berg and Butterfield, 1968) (Figure 2.6). Owens et al. (1995) reported a quadratic increase in fat for feedlot cattle with empty body weights between 120 and 520 kg, compared to linear increases in muscle and bone. Overall fat accumulation has a high growth impetus (Figure 2.3) relative to other tissues, i.e., the relative rate of fat growth is higher. Published fat growth

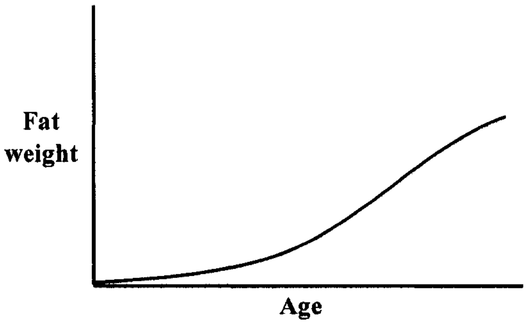

Figure 2.6. A general illustration of fat growth in cattle from birth to a typical harvest age.

coefficients have been one and a half to two times the coefficients for muscle and bone for the entire growth phase (Berg and Butterfield, 1966; Mukhoty and Berg, 1971). An increasing proportion of growth in post-pubertal animals is due to deposition of fat (Marple, 2003).

Overall, fat is distributed at a rapid rate but partitioned differentially among major depots: subcutaneous; intermuscular; intramuscular; and internal. In early stages of development most fat is contained in the intermuscular and internal depots. The proportion of intramuscular fat is only slightly smaller. As the animal grows, the proportion of subcutaneous fat increases such that internal and intramuscular fat become a much smaller percentage of total fat (Johnson et al., 1972) Several reported fat growth coefficients support subcutaneous fat as a high growth impetus, late-developing tissue. Growth coefficients for subcutaneous fat exceed one, while coefficients for the other depots are generally less than one (Berg et al., 1979; Jones et al., 1980a) indicating slower growth relative to total fat growth. However, Jones et al. (1980a) showed differential coefficients for subcutaneous and intermuscular fat among wholesale cuts.

Intramuscular fat, an important component of carcass value, becomes a smaller proportion of total carcass fat as an animal fattens. The difference between the subcutaneous and intramuscular fat coefficients also becomes larger during growth (Johnson et al., 1972). Johnson et al. (1973) found that the distribution of fat among different muscles did not change with an increase in total carcass fat. These results indicate the difficulty in obtaining sufficient amounts of intramuscular fat without increasing over-all body fat, especially in the subcutaneous depot, another important carcass value component.

Carcass fat is the most variable of the tissues and is the most responsive to manipulation. To obtain the optimum amount of fat dictated by the market, the growth impetus for subcutaneous fat would ideally be reduced to a level that would provide the optimum amount of intramuscular fat, at optimum fat levels carcass-wide, without adversely affecting other economically important carcass components.

Relationship between bone, muscle and fat. Tissue growth does not occur independently and it is important to understand the relationship between bone, muscle and fat in order to evaluate the growth process. There are a number of ways in which the tissues interact and numerous ways to describe growth using a combination of tissue measures. Knowledge of these processes is key to understanding and predicting changes in carcass composition.

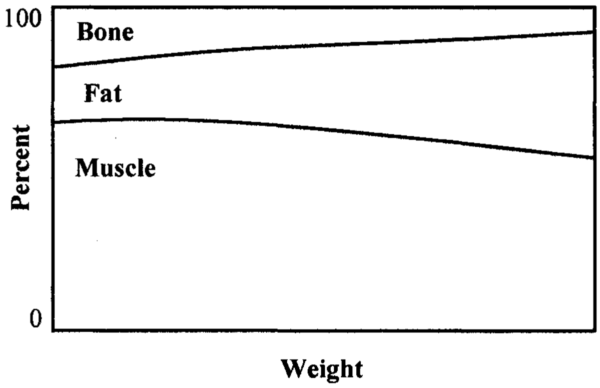

Figure 2.7. Proportion of bone, muscle and fat in the carcass of growing cattle.

The proportion of tissues in an animal carcass changes during growth (Figure 2.7) as does their relative importance. In early growth, muscle has a major influence on carcass composition and, later, the major influence is fat (Berg and Butterfield, 1976). With regard to weight, this influence can generally be seen as a decrease in bone and muscle percentage and an increase in fat percentage with increases in weight (Berg and Butterfield, 1968). In a comprehensive dissection study, Callow (1948) reported specific proportions of each tissue during growth and fattening as an average for all meat animals. His results indicated a steady decline in the percent of muscular tissue, from 62% in the early fattening period to 49% in later periods, in conjunction with an increase in carcass fat from 20 to 40%. A similar decrease in percent bone was also reported. Callow concluded that the level of fatness in a carcass is the primary determinant of the percentages of muscle and bone.

Long (1976) presented ranges for the proportion of each tissue in slaughter cattle, indicating that many combinations of each proportion are possible (Table 2.2). Callow (1948) categorized carcasses with more than 18% fat as cattle in the fattening stage of growth, so Long's ranges seem to span only the fattening period. The ranges generally agree with the results of Callow, with the exception of percent bone, which is larger than Callow's estimate for fat cattle. The relative amounts of tissue become important in evaluating cattle as Berg and Butterfield (1966) suggested that "carcass composition is best assessed by use of two measures: muscle:bone ratio and percentage fat tissue in the carcass".

Table 2.2. Range of composition of major tissues in slaughter cattle carcasses[a].

	Percent		
	High	Low	Range
Bone	16	12	4
Muscle	67	37	30
Fat	45	20	25

[a] From Long, 1976

Research has not shown a consistent association between muscular development and bone growth (Shahin and Berg, 1987), but the muscle:bone ratio is an important indicator of carcass value. Butterfield (1975a) suggested that the ideal carcass has a high muscle:bone ratio with the smallest possible amount of fat. A high muscle:bone ratio indicates a larger amount of saleable meat at the same carcass fat, and is desirable. Bone is used as a baseline of sorts for comparing muscle and fat due to the small amount of variation in bone percentage (Table 2.2). In early growth, the muscle:bone ratio is rising because muscle is growing faster than bone (Callow, 1948). In later growth stages increases in fat levels preclude a relative increase in muscularity (muscle:bone ratio) (Berg and Butterfield, 1966).

Callow (1948) compared the fat:muscle ratio during growth to evaluate the relationship between muscle development and fat deposition. His results have been confirmed with recent studies showing a rapid increase in the fat:muscle ratio with increases in carcass fat deposition. The larger ratio is due to the high fat growth impetus relative to muscle growth in later stages. Callow considered the point at which fat was growing faster than muscle to be the beginning of the fattening phase. Conversely, Johnson et al. (1973) reported on the relationship between fat partitioning and muscle growth. They found no changes in intramuscular fat partitioning with animal growth indicating little relationship between the fat:muscle ratio and fat partitioning.

The carcass tissue changes that occur during growth are affected by growth of individual tissues and other biological processes. Tissue development proceeds in a general fashion, as described in previous sections, but changes in individual animals are affected by numerous factors. An understanding of these factors is essential to predicting carcass composition on an individual basis.

Factors affecting cattle growth

Measures of postweaning growth in cattle are important for economically efficient production in growing and finishing phases. The main production concerns are the amount of weight gain that can be achieved, the efficiency of that gain and changes in carcass composition during that gain to the optimum harvest point. The focus of this section is on the traditionally quantifiable growth of slaughter cattle exhibited as changes in weight (or mass) and the efficiency of that growth. The growth curve was earlier presented as a representation of changes in animal weight (or size) over time. The following discusses a number of factors that can impact that growth curve for individual animals. Knowledge of these factors allows manipulation of growth for improved production efficiency under differing management situations.

Breed/Biological type. Breeds of cattle are generally categorized as *Bos Taurus* or *Bos Indicus*. The former category includes Continental-type (CON) and British-type (BRI) cattle, and the latter includes Brahman-influenced (BRM) cattle, considering only the beef breeds. Essentially there are three types of cattle, and associated crosses, which are of interest for comparison. Difference between types, and not necessarily individual breeds, will be of primary concern for this discussion as a number of studies have characterized cattle types in terms of daily gains, feed efficiency and days on feed to an endpoint.

Average daily gain (ADG), or rate of gain, has been used as a predictor of an animal's ability to reach a certain weight in a given amount of time. The faster an animal grows, the sooner it will be ready for harvest (assuming carcass composition is acceptable). There tends to be a difference in size (and presumably maturity) of each of the breed types during the normal growing and finishing period which results in animals that are at different points on the growth curve (Berg and Butterfield, 1976). Animals in pre-inflection growth will naturally gain more weight over a feeding period than those animals in post-inflection growth (Brody, 1945).

Urick et al. (1991) reported sire breed effects for ADG in BRI and CON steers, with performance adjusted to a constant age. The CON-sired steers exceeded the ADG of the BRI-sired steers, with the exception of Tarentaise, which was intermediate to both and not different than either. Similarly, Smith et al. (1976) found differences in ADG between BRI-cross and BRI x CON steers fed for a constant number of days. The BRI x CON steers were generally heavier at the end of the feeding period and had ADG values at least 10% larger then the BRI-cross cattle. Comparable results were reported by Smith and Rahnefeld (1988) for BRI x CON calves fed for a constant number of days.

No differences in ADG were detected by Wyatt et al. (2002) between BRI, BRI x BRM and CON x BRM sire breeds for steers fed to a constant fat thickness endpoint, though breed did significantly affect final weights. Block et al. (2001) found that CON-cross steers actually had lower ADG than their BRI counterparts. This study also calculated ADG based on feeding to a constant fat thickness endpoint. The apparent discrepancy between earlier and later studies stems from the feeding endpoint or endpoint adjustment used. Continental and BRI cattle fed for the same number of days, or to the same age, are likely to be compositionally different. When CON cattle are fed to an endpoint that takes a longer number of days (such as to a constant backfat thickness) they enter the phase of growth where weight gain begins to increase at a decreasing rate. Gain that occurs on the flattening part of the growth curve causes a decreasing ADG (Berg and Butterfield, 1976).

A relative growth rate (RGR) has instead been used to report postweaning growth as a percentage increase in body weight (Brody, 1945) to adjust for breed-type size differences and differences in maturity. Brody indicates, though, that RGR may not be a useful indicator except in the case of a short time span between weight measurements. When viewing RGR as a daily percent change in body weight over the feeding period there are few differences between breeds and breed types (Smith et al., 1976; Urick et al., 1991). Relative growth rate may not be a suitable statistic for evaluating breed growth differences in the postweaning phase.

The efficiency of feedlot gains are described by feed efficiency which, over a feeding period, is the amount of feed required for growth to a specific endpoint. Efficiency is usually expressed as the amount of feed (or the cumulative metabolizable energy) per unit of gain. The literature suggests as discussed in detail below that a breed effect does exist for feed efficiency but, like ADG, is dependent on the harvest endpoint selected. When performance is adjusted to a constant age, the larger CON cattle tend to be more efficient than smaller BRI cattle at converting feed to weight gain (Smith et al., 1976; Urick et al., 1991; Amer et al., 1992). When fed to a constant weight endpoint, CON cattle are again more efficient, but at a constant backfat thickness, BRI cattle tend to be more efficient (Smith et al., 1976; Urick et al., 1991). The difference can be explained by the greater amount of time needed for the larger CON cattle to reach a set backfat thickness compared to the BRI cattle. In the same manner that ADG is reduced, an increasing time on feed to reach the endpoint will reduce feed efficiency (Robison, 1976).

Associated with ADG and feed efficiency is the amount of time needed for cattle to reach a target compositional endpoint. Differences in days to finish are specifically related to attributes of a particular breed type, a concept that will be discussed in detail in subsequent sections. In terms of a target endpoint, cattle are generally fed as groups to a specific backfat thickness or a perceived level of fatness in commercial settings. Under normal backgrounding conditions, BRI cattle tend to need fewer days on feed to reach a given level of fatness than BRM influenced cattle, which in turn need fewer days than CON cattle (Block et al., 2001; Wyatt et al., 2002). Similarly, BRI cattle tend to need fewer days than CON cattle to reach a constant marbling score endpoint (Wheeler et al., 2004; 2005). A number of interactions may exist, however, that make the evaluation of breed type differences for days to finish quite complex.

Environment. There are a number of environmental factors that can impact postweaning growth of animals including, but not limited to: ambient temperature; humidity; light; and hygiene (Eisen, 1976). Studies in beef cattle have generally evaluated season (or some measure of climatic conditions) and type of housing or shelter and their effect on feedlot performance. Season (generally defined as winter or summer) has a significant impact on the feedlot growth traits ADG, dry matter intake (DMI) and feed:gain (feed efficiency). Mader (2003) reported increased ADG and decreased DMI and feed:gain for steers fed in the winter versus the summer. Kreikemeier and Mader (2004) reported a similar decrease in DMI for heifers fed in the winter versus the summer. In both studies the decrease in DMI was attributed to variation in temperature, especially in the winter, and not just sub-freezing temperatures alone.

Conversely, Pusillo et al. (1991) found a decrease in ADG and an increase in feed:gain during the winter months. There was some interaction between each trait and three types of housing, but cattle performance in the summer was still more efficient in this study. They did find that shelter improved ADG and feed:gain in each season. The apparent contradiction between studies might be explained by differences in temperature that constituted winter and summer in each of the studies, but mean temperatures were not reported by all researchers. Pusillo et al. (1991) cited a number of studies with contradictory evidence for a decrease in performance in both warm and cold seasons.

Frame size. As a general rule, small-framed cattle are considered early maturing as they reach mature weight at a younger age than cattle with larger frames. This early maturity allows small-framed cattle to progress along the growth curve and reach desired compositional endpoints earlier than larger cattle. Large-framed cattle tend to have greater feed intake (Owens et al., 1993), reach target endpoints later and at heavier weights, but they often grow more rapidly and efficiently (depending on the endpoint) than smaller cattle (Smith et al., 1976; Urick et al., 1991; Amer et al., 1992). In the case of an extended period of time on feed, Cianzio et al. (1982) found that larger-framed steers tended to have a lower ADG than smaller-framed steers, though the difference was not significant.

A measure of absolute growth rate (AGR) can be used to estimate the amount of gain at a specific point in time (Brody, 1945) to better compare growth in the feeding period. Absolute growth rate is a measure of the actual change in weight over time, as opposed to RGR which measures the change in weight over time, relative to animal weight. Tatum et al. (1986a) reported a frame size effect on AGR considering small-, medium- and large-framed steers fed for 140 days. The large-framed steers grew at a faster rate than the other frame classes indicating that these cattle gained at an increasing rate. Though smaller-framed cattle grow more slowly, Dolezal et al. (1993) found that they needed fewer days on feed to reach a constant backfat thickness endpoint. There is an apparent interaction for growth performance between frame size and choice of harvest endpoint.

Hormonal control. Specific hormones, both endogenous and exogenous, have been shown to affect growth (Owens et al., 1993). The U.S. beef industry has used hormone implants for a number of years to improve the efficiency of growth, particularly in the postweaning phase of production. Numerous studies have shown the benefits of various implants on ADG and/or feed efficiency (Simms et al., 1988; Foutz et al., 1997; Rumsey et al., 1999; Hermesmeyer et al., 2000; Guiroy et al., 2002; Bruns et al., 2005). In general, implants increase ADG and feed efficiency, ultimately increasing final weight at harvest. The timing of the implant may be important as Foutz et al. (1997) found implants to be most effective during the middle of the finishing period and Simms et al. (1988) reported the need to re-implant for finishing periods longer than 120 days. Hormonal implants have also been shown to increase the mature body weight of steers, thus increasing the weight required for an animal to reach a desired compositional endpoint (Guiroy et al., 2002).

Management. Postweaning performance is also greatly influenced by decisions on management of cattle from weaning to harvest. Days on feed is management decision that is most easily controlled, at least in terms of finishing performance, and has easily interpretable results. Van Koevering et al. (1993) found increasing ADG and feed intake over a 147-day feeding period using British x Continental steers and reported an optimum number of days on feed for feed efficiency. Generally, increasing the number of days on feed will eventually lead to reduced ADG and feed efficiency. Manipulation of days to finish can be used to manage cattle to specific compositional endpoints but may have little bearing on optimum growth performance.

Numerous researchers have shown that there are weaning and feeding management strategies that influence postweaning growth and the number of days needed to reach a desired compositional endpoint. The control of this time is impacted by ADG, DMI and feed efficiency. Early weaning has been shown to be effective at improving growth during the growing and finishing period. Fluharty et al. (2000) reported increases in ADG for early-weaned steers in the early growing period and heavier weights at the beginning of finishing. Over the entire finishing period early-weaned steers had a higher mean ADG and an increased feed conversion ratio, with fewer days on feed needed to finish.

Backgrounding is often used to increase animal weight without the increase in fatness that accompanies finishing diets. Backgrounding is essentially a growing period. Several studies have evaluated the impact of this management decision on subsequent growth performance in the feedlot. Generally, as the age of a steer increases the intake and ADG are expected to increase, coupled with decreases in efficiency (Short et al., 1999). Block et al. (2001) found shorter days to finish, larger daily gains and decreased feed efficiency for cattle from longer backgrounding programs which correspondingly produced older animals at time of placement on feed. Wertz et al. (2001) found similar results for early-weaned heifers. Sainz and Vernazza Paganini (2004) found no significant differences but noted that ADG tended to increase with increases in the length of the growing period.

Many of the differences in growth performance (including weaning and backgrounding strategies) are attributable to compensatory gain. Restriction of nutrient intake in growing or finishing animals can lead to greater gains when feed is no longer restricted, compared to cattle that were never restricted (Owens et al., 1993). To take advantage of compensatory gains, programmed intake systems have used a step-wise feeding process (Loerch and Fluharty, 1998). Programmed intake reduces overall ADG in the feedlot, but allows steers to continue to increase in ADG as they approach market weight, with an associated increase in feed efficiency, as opposed to a reduction in rate of gain for full-fed steers (Hironaka et al., 1979; Sainz et al., 1995; Knoblich et al., 1997; Rossi et al., 2001).

Nutrition. Cattle performance will differ depending on the level of nutrition provided throughout the feeding period. Nutritional restriction impacts normal growth (Eisen, 1976) in any organism. Much of the research in slaughter cattle has involved investigating the impact of alternative planes of nutrition on body composition or the possibility of substituting more readily available feedstuffs. In an *ad libitum* feeding situation a lower energy diet will reduce gain and feed efficiency, increasing days to a target finish endpoint, as compared to a higher energy diet (Dubeski et al., 1997; Fluharty et al., 2000). Specifically, a decrease in the ratio of grain to roughage reduces ADG, feed conversion and increases days to finish (Pryor et al., 1972; Price et al., 1980; Mandell et al., 1997b). The manipulation of nutritional levels throughout the growing and finishing period, is an effective tool for achieving desired performance and compositional levels.

Sex. In a typical harvest setting steers and heifers are the only sexes of interest and will be the focus of this discussion. Typically there is little difference in growth attributable to sex, especially in the early growing period (Berg and Butterfield, 1976) with the potential for some small differences later in the feeding period as heifers tend to gain more slowly and mature earlier than steers. Gettys et al. (1987) evaluated growth difference between sexes and reported a very small difference between steers and heifers for gain, but no difference in feed conversion (efficiency). Differences in days to finish are likely to exist as a result of differences in fat deposition, but taken as a whole, sex alone does not seem to be a major factor affecting growth in growing and finishing phases.

Sire genotype. The growth potential of any animal is determined by genes for growth and the multitude of factors previously discussed several of which are under genetic control. The impact of sire genotype, as determined by large-scale genetic evaluation, on growing and finishing cattle for growth traits has been evaluated in many studies. The desire is to predict an animal's performance based on its sire's genetic merit. Some studies of cattle indicate that no significant differences exist for ADG, feed intake or feed efficiency of steers in growing and finishing periods due strictly to high-growth versus low-growth potential sires, though high-growth sired steers tend to be larger (Grings, et al., 1996; Short et al., 1999). Growth potential of sires in these studies was determined by traditional genetic evaluation, but some QTL research has identified certain chromosomal segments that may impart improvements in growth characteristics (particularly ADG) when a particular genotype is present (Beever et al., 1990; Casas et al., 2001). The inconsistency in results between the studies using breeding values and those using QTL might be due to the definition of growth potential used to assign genetic values in the breeding value studies versus the definition used to identify QTL.

Factors affecting carcass composition

When cattle are valued on a carcass basis changes in composition of the carcass as the animal grows become economically important. In such a marketing system the prediction of traits that directly impact value, such as weight, dressing percent, yield grade and marbling score, has generated considerable interest. In order to predict composition at harvest, knowledge is needed about the relative growth of major tissues (bone, muscle and fat), interactions among the tissues, and their impact on carcass characteristics. The growth of each tissue was previously discussed, but there are a number of factors that can affect tissue changes and, in turn, affect the resulting carcass composition. The following section examines these factors with particular interest in carcass component changes from weaning to harvest.

Breed/Biological type. There are numerous research studies that have examined breed or biological type differences in the relative growth and distribution of carcass tissues. Callow (1961) suggested that these differences were due to the selection history of various cattle breeds. Cattle selected for beef versus dairy-type traits have differential carcass composition at the same physiological point in life. The mechanism for breed effects on carcass composition is not well understood within cattle types (Owens et al., 1993); the largest effect appears not to be strictly related to breed but to mature size. The observable breed type differences are of most interest for this discussion, while the reasons for these differences are less important.

Relative tissue growth and partitioning. In an earlier discussion it was mentioned that the growth of bone was least variable as compared to muscle and fat, especially in the context of relative growth and distribution of bone in carcasses of normal growth BRI, CON, dairy and various crossbred cattle (Mukhoty and Berg, 1971; Berg et al., 1978b; Jones et al., 1978; Patterson et al., 1985). In these studies no differences in bone growth coefficients due to breed were reported. Similarly, bone distribution was found to be similar in these breeds, though some small differences were reported but thought to be commercially insignificant. Callow (1961) and Mukhoty and Berg (1971) reported differences in mean bone weight were attributed to differences in the size (weight) of the carcass at the slaughter endpoint.

Differences in bone weight and distribution have been seen in comparisons of double-muscled and "normal" growth BRI and BRI-cross cattle. At equal side weights (half-carcass weight) double-muscled cattle were shown to have lower bone weights than British-cross cattle (Shahin and Berg, 1985c) and at equal bone weight were shown to a have a bone distribution that is shifted towards the thoracic region (Shahin and Berg, 1985b). In the case of double-muscled cattle, increases in bone are thought to occur in areas that are more involved in weight support (Shahin and Berg, 1987). In general, breed seems to have little effect on relative bone growth.

Callow (1961) showed that muscle tissue in a carcass comprises about one-third of live weight. Breed affects that proportion slightly and also the percentage of muscle tissue relative to other carcass tissues. Callow found that Friesians had the highest percent of muscle in carcass weight (60.0%), followed by Herefords (57.1%) and Shorthorns (55.1%). Breed differences in muscle weights have also been shown by a number of researchers (Butterfield, 1962; Jones et al., 1981; Patterson et al., 1985). The differences in relative muscle weight are seemingly affected by fat deposition. Increases in muscle proportion or weight are dependent on fattening, with little variability in relative bone weight.

Distribution of individual muscles compared to total muscle tissue does not differ between breeds (Berg et al., 1978d; Jones et al., 1980c), but some breed differences in the overall rate of muscle growth have been identified (Toelle et al., 1986; Korver et al., 1987). The ratio of low- to high-value muscles is a consideration for assessing the value of a carcass. Butterfield (1965) found no differences in this ratio between British and Brahman cattle, and was subsequently supported by Mukhoty and Berg (1973) studying Hereford and Shorthorn cattle. Shahin and Berg (1985d) suggested, however, that the ratio of "expensive" muscles is greater in cattle with a higher degree of maturity compared to others at a constant muscle weight and thus breed differences may exist. Kempster et al. (1976) suggested that breed differences in muscle distribution are small and can probably be ignored unless small differences exist for other traits of interest and then muscle distribution comparisons may be needed.

Muscle:bone ratio has been suggested as an appropriate comparison for breed differences in carcass composition (Berg and Butterfield, 1966). A larger muscle:bone ratio is desirable in terms of carcass value. Berg and Butterfield (1966, 1968) found breed differences in muscle:bone ratios, with BRI cattle having larger ratios than either BRM or dairy-type cattle. Similarly, Shahin and Berg (1985c) found differences in muscle:bone ratio between double-muscled and BRI and BRI x CON cattle. Jones et al. (1978), studying BRI, CON and dairy crossbred cattle, reported no breed differences, but suggested that muscle:bone ratio is most variable in early growth and that differences may disappear in later growth.

There exists a large amount of evidence that breed differences exist for the amount, deposition and partitioning of fat tissue in the carcass. Callow (1962) reported that the percentage of fat was highest in Shorthorn as compared to Hereford and Friesian cattle. Charles and Johnson (1976) further reported percentage of fat for BRI, CON and dairy cattle at a constant carcass weight with Angus cattle having the highest fat percentage, followed by Hereford and Charolais. These differences are likely as much a function of differential maturity levels as they are of breed due to the constant endpoint considered. Several researchers have shown there to be no differences in the rate of fat deposition, but rather differences in the onset of fattening (Berg et al., 1978a; Patterson et al., 1985; Shahin and Berg, 1985a).

Breed or biological type does have an influence on the partitioning of fat into the major fat stores in the carcass. Callow (1961) reported that dairy-type breeds stored more fat internally, while beef-type breeds partitioned more fat in the subcutaneous store. Wood (1982) further suggested that there is an association between fat partitioning and a breed's capacity for milk production - higher production indicating more internal fat. A number of studies support these findings of differences between internal and subcutaneous fat partitioning (Berg et al., 1978c; Jones et al., 1980ab; Jones, 1985; Toelle et al., 1986), but Callow (1961) found no breed differences for partitioning of fat in the intramuscular depot. Kempster et al. (1976) and Berg et al., (1978c) suggest that there may be partitioning differences, but they are small and probably not commercially important (especially when comparing beef breeds) when cattle are compared at equal levels of maturity, i.e., the same point on the growth curve. Korver (1987) concluded, however, that equal body composition can not be assumed at the same level of maturity for different breeds.

Carcass characteristics. The small differences in carcass composition and comparisons at equal maturity discussed above do not necessarily translate to carcass differences under normal production systems. Cattle are often harvested after a predetermined time on feed or at a target weight or level of finish. These different systems highlight the breed differences that exist for carcass traits. Table 2.3 indicates the presence of breed (or breed of sire) effects for carcass traits from a number of literature sources. Performance for all traits listed has been shown to be affected by breed.

Carcass trait comparisons between breeds can be difficult at a single endpoint due to breed by endpoint interactions that may exist (Mandell et al., 1997b; Block et al., 2001). Comparing breeds or breed-types at different endpoints, however, can highlight the breed differences in some carcass traits. Wheeler et al. (2005) evaluated carcass characteristics of cattle with BRI and CON sire breed types (Table 2.3), adjusted to several endpoints. Steers sired by the BRI breeds had greater levels of backfat, smaller longissimus muscle areas, higher yield grades, a greater percentage of carcasses above USDA Yield Grade four, higher marbling scores and a greater percent of carcasses grading USDA Choice when observations were adjusted to a constant mean carcass

Table 2.3. Presence of breed (or breed of sire) effects reported in the literature for important carcass traits.

	Carcass Traits[a]									
Breed Types	LWT	CWT	DP	FT	LMA	MS	%CH	YG	%RP	Source
Angus Charolais Hereford Jersey Limousin Simmental South Devon	**	**		**	**			**	**	Koch et al., 1976[bce]
Angus Brangus		*		NS	NS	NS				Beaver et al., 1989[g]
Angus Pinzgauer Red Poll Simmental Tarentaise	**	**		**	**	**				Urick et al., 1991[g]
Angus Brahman Charolais Hereford		*		*	*	**				DeRouen et al., 1992b[d]
Angus Braunvieh Charolais Gelbvieh Hereford Limousin Pinzgauer Red Poll Simmental MARC-Composite				**	**	**			**	Gregory et al., 1994[h]
Charolais Limousin				NS	**	NS				Mandell et al., 1997b[dh]

(Continued)

Table 2.3. *Continued.* Presence of breed (or breed of sire) effects reported in the literature for important carcass traits.

	Carcass Traits[a]									
Breed Types	L W T	C W T	DP	FT	L M A	MS	% C H	Y G	% RP	Source
Angus Simmental		**		**		**			**	Hassen et al., 1998[b]
Angus Charolais Hereford	*	*		*	*	*				Block et al., 2001[d]
Angus Friesian Hereford Norwegian Red Swedish Red and White	*	*	*	*	NS	*	*	*	*	Wheeler et al., 2004[bcdef]
Angus Charolais Gelbvieh Hereford Limousin Red Angus Simmental	*	*	*	*	*	*	*	*	*	Wheeler et al., 2005[bcdef]

[a] LWT = live weight at slaughter; CWT = carcass weight; DP = dressing percent; LMA = longissimus muscle area; FT = 12th rib backfat thickness; MS = marbling score; %CH = percent USDA Choice; YG = USDA yield grade; %RP = percent retail product

[b] Endpoint evaluated = age

[c] Endpoint evaluated = carcass weight

[d] Endpoint evaluated = backfat thickness

[e] Endpoint evaluated = marbling score

[f] Endpoint evaluated = fat trim

[g] Endpoint evaluated = days on feed

[h] Endpoint evaluated = live weight

* $P < 0.05$

** $P < 0.01$

NS = No significant effect reported

weight (363 kg). At a constant mean backfat thickness (1.1 cm) steers sired by CON cattle had larger carcass weights, dressing percents and longissimus muscle areas, but yield grades, marbling scores and percent USDA Choice were similar between BRI and CON breed types. At a constant marbling score (Small^{35}) the same trends were apparent for carcass weight, dressing percent and longissimus muscle area, but CON sired-steers had a higher %KPH. No other breed-type trends were discernable.

In studies comparing BRI (Angus) and BRM or BRI x BRM crossed cattle small differences in carcass characteristics were reported for animals fed a constant number of days. Beaver et al. (1989) reported differences in carcass weight only, with Angus steers having larger weights than Brangus steers when fed for 143 days. Lunt et al. (1985) evaluated Angus and Brahman steers at five lengths of time on feed finding few significant differences in carcass traits at any time point. Angus steers tended to have a larger amount of KPH fat, higher marbling scores and higher quality grades. Differences depended, however, on the number of days on feed.

Environment. There is evidence that environmental factors can impact beef cattle carcass characteristics, considering consistent management scenarios. Traits related to fat deposition and distribution seem to be most influenced by the effects of ambient temperature (season) and access to shelter. Kreikemeier and Mader (2004) found seasonal effects on carcass characteristics for heifers in the feedlot, particularly for traits relating to fat levels. They found greater backfat thickness in summer, but greater marbling scores and longissimus muscle areas in winter. Consequently, quality grades were higher and yield grades were lower in the winter than in the summer. Pusillo et al. (1991), similarly, found lower quality grades in the summer, but reported no season effects for yield grade. Mader (2003) also reported a small significant effect due to housing, with decreased backfat thickness and marbling scores for animals provided shelter.

Frame size. Frame size is commonly used to indicate the live weight at which cattle will reach a given level of fatness and, consequently, differences in frame size are perceived to equate to differences in the relative composition of the carcass. Tatum et al. (1986a) found differences in the proportions of bone, muscle and fat of feeder cattle with different frame sizes. An increase in frame size was also associated with an increase in the percent of muscle and bone and a decrease in the percent of fat at a constant weight. At a constant backfat thickness, however, no differences were detected. Dolezal et al. (1993) found similar results when a constant fat endpoint was used. No differences in muscle:bone ratio were reported in either study.

Differences in percent fat are not expected at a given level of overall fatness (or maturity), regardless of frame size. Differences in the rate of fat deposition are also not dependent on frame size, but an effect on fat partitioning does exist (Cianzio et al., 1982; Tatum et al., 1986b). Large-framed steers tend to partition less fat in the subcutaneous depot, while depositing more fat internally (Dolezal et al., 1993). Large-framed cattle tend to deposit intramuscular fat at lower levels of overall fat (Tatum et al., 1986b). Belk et al. (1991) suggested, however, that frame size differences in fat distribution are small and not commercially important.

Carcass trait differences due to frame size have been reported. At similar days on feed large-framed steers tend to have greater carcass weights, smaller amounts of backfat, lower marbling scores and lower quality grades (Camfield et al., 1997). May et al.

(1992) showed that large-framed cattle tend to have lower yield grades and larger longissimus muscle areas. These differences are assumed to be due to differences in the onset of fattening between maturity groups. At similar fatness endpoints the differences in fat-related traits would be somewhat reduced, though weight and dressing percent would be affected (Dolezal et al., 1993).

Hormonal control. There are a number of substances, including hormones, which are thought to control deposition of fat and lean tissue in animals (Wood, 1982). Hormonal implants are used to increase growth and feed efficiency under most postweaning production systems. These implants can affect the composition of the carcass at harvest. A number of studies have evaluated the effect of differing implant strategies on carcass characteristics (Simms et al., 1988; Gerken et al., 1995; Foutz et al., 1997; Roeber et al., 2000; Kreikemeier and Mader, 2004; Bruns et al., 2005). In all studies growth-promoting implants were associated with increased carcass weights at similar compositional endpoints. In general, most other carcass traits were not largely affected by the application of implants (of those that currently have value to producers). There is some evidence that marbling score (and quality grade) may be reduced under some implanting strategies (Simms et al., 1988; Roeber et al., 2000), but Bruns et al. (2005) found that the timing of the implant can have a positive affect on marbling deposition. Roeber et al. (2000) showed a reduction in tenderness due to implants that will need to be considered in systems that place value on product tenderness.

Management. The management factors that seem to impact the composition of major carcass tissues involve the feeding program and length of time cattle are on feed. Butterfield (1966) and Berg and Butterfield (1968) presented differences in the relative growth of tissues between animals fed for uninterrupted growth and those under conditions of semi-starvation and recovery. They found that subcutaneous fat was most affected by semi-starvation and that muscle depletion occurred in the reverse order of muscle development. Upon recovery, the cattle were able to regain muscle at a rapid rate, but would need a longer period of time to attain body fat levels equal to animals under normal growth conditions.

The length of time cattle are on feed will affect the relative tissue composition at slaughter, especially considering differences in stage of maturity. Animals fed to gain steadily will begin to deposit relatively more fat as they approach maturity and the longer the cattle are on feed the fatter they will become (Long, 1976). Cattle with rapid rates of gain will accumulate lean in early growth, but will begin to deposit more fat in fewer days on feed and will have the fattest carcasses at harvest (Butterfield, 1976). Length of time on feed is best matched with maturity type to determine the optimum composition of carcass tissues.

There are a number of management strategies that have been used to improve the efficiency of growth and in turn yield optimal carcass characteristics. The number of days needed for cattle to reach an optimum carcass endpoint is critical to profitability in the postweaning phase. The effect of days on feed on carcass composition has been extensively studied using serial slaughter data (Smith et al., 1981; May et al., 1992; Duckett et al., 1993; Van Koevering et al., 1993; Van Koevering et al., 1995; Camfield et al., 1997; Mandell et al., 1997a). Generally, carcass weight, adjusted backfat thickness, longissimus muscle area and yield grade increase linearly with days on feed under typical feedlot programs. Marbling score, quality grade and kidney, pelvic and heart fat tend to increase in a quadratic manner with increased time. Studies involving small-framed crossbred cattle found that after 112 to 130 days on feed marbling deposition ceases (Duckett et al., 1993; Van Koevering et al., 1993). Evaluating large-framed crossbred cattle Mandell et al. (1997a) found increasing deposition of marbling through 147 days. In all cases subcutaneous fat continued to increase.

Manipulation of intake and rate of gain also has implications for carcass performance. Programmed intake feeding systems have been suggested as a way to reduce the quantity of feed required postweaning and may be used to alter carcass composition (Sainz et al, 1995). The effects on the carcass are dependent on the timing and length of feed restriction, but in general, carcass weight and backfat thickness are reduced in steers with restricted intake (Sainz et al., 1995; Knoblich et al., 1997; Rossi et al., 2001). Other carcass traits, especially marbling score, did not seem to be affected by feeding regimen. In those studies the length of time on feed was likely sufficient for maximal marbling deposition. Similarly, Purchas et al. (2002) showed that cattle with faster rates of gain had increased subcutaneous fat with subsequent improvements in intramuscular fat development.

Forage conditions often necessitate the early weaning of calves which in turn may impact on carcass characteristics. Sainz and Vernazza Paganini (2004) indicated that backgrounded calves may need more time on feed to reach acceptable levels of finish and Fluharty et al. (2000) and Wertz et al. (2001) reported that early-weaned calves achieve acceptable accumulation of intramuscular fat, but not excessive amounts of subcutaneous fat after lengthy amounts of time on feed (>200 d). Early-weaning or backgrounding programs should be matched with maturity type for an optimal carcass produced in an acceptable amount of time.

Nutrition. Factors that can influence the relative composition of carcass tissues include the postweaning plane of nutrition (Long, 1976). Callow (1961) and Patterson et al. (1985) reported no differences in tissue proportions due to dietary level. Other studies, however, have shown differences in proportion of fat, fat deposition and relative amounts of bone, muscle and fat attributable to differences in nutrition (Butterfield et al., 1966; Mukhoty et al., 1970; Pryor and Laws, 1972; Batra et al., 1973; Jones, 1985). Conversely, nutritional level does not seem to affect the distribution of individual tissues, particularly muscle weight and muscle:bone ratio (Butterfield et al., 1971; Mukhoty and Berg, 1973; Kempster et al., 1976).

There are number of interactions with nutritional level (diet) that make it difficult to discuss effects of nutrition on carcass traits. Generally, cattle can be fed to meet any carcass endpoint by increasing/decreasing days on feed. Price et al. (1980) and Mandell et al. (1997a) found few differences in carcass components due to diet when animals were fed to similar endpoints. Furthermore, Mandell et al. did not find the expected increase in quality grade due to grain versus forage feeding. Wertz et al. (2001) also found few differences in carcass traits due to grain versus forage feeding systems.

Sex. There are few differences between sexes (steers and heifers) for the relative growth of tissues or the carcass composition at harvest when fed to similar compositional endpoints. Mukhoty and Berg (1971) found no differences in growth coefficients for bone, muscle and fat between steers and heifers. Further, no differences in the distribution of muscle or fat have been reported (Mukhoty and Berg, 1973; Berg et al., 1979). Any differences are due to the early onset of fattening in heifers and the more rapid fat deposition relative to muscle as compared to steers (Long, 1976).

Factor interactions. There are a number of interactions that exist among the factors affecting postweaning growth and composition of cattle. Mandell et al. (1997b) described interactions between breed and feeding endpoint, nutrition and feeding endpoint and nutrition and breed. Similarly, Block et al. (2001) reported breed by management, breed by endpoint and management by endpoint interactions. Many of these interactions are important considerations for efficient production, but they are quite varied and too numerous to individually discuss here.

Selection for postweaning growth and carcass composition

Some form of selection has long been used to improve production efficiency for anticipated improvement of profitability given particular marketing alternatives. Traditionally, U.S. cattle producers have focused selection on traits important to the marketing of weaned calves (e.g., cow productivity and longevity traits and calf weaning weight). With the increasing popularity of alliances, branded beef programs and other grid marketing systems there has been more interest in genetic improvement of feedlot and carcass traits. In order to make intelligent selection decisions it is important to understand the possible ramifications due to selection for the traits of interest, as well changes in any correlated traits.

Genetic parameter estimates for growth and carcass composition. Table 2.4 lists published heritability estimates (and associated standard errors when available) for several traits that are indicative of postweaning growth. In general, the cited studies indicate that sufficient additive genetic variation exists to make genetic progress in the growth traits through selection, possible. Heritability estimates for weaning weight (0.09 to 0.39), feed conversion ratio (0.06 to 0.46) and relative growth rate (0.18 to 0.35) were generally in the low-moderate to moderate range. Estimates for postweaning gain (0.36 to 0.49), yearling weight (0.16to 0.48) and average daily gain (0.19 to 0.64) were mostly high-moderate to high in value.

Table 2.4. Published heritability estimates for postweaning growth traits in beef cattle^{a}.

Growth traits^{b}						
WWT	PWG	YWT	ADG	RGR	FCR	Source
				0.25±0.12		Smith and Cundiff, 1976
0.18±0.01	0.49±0.04	0.37±0.03				Buchanan et al., 1982
			0.57^{c}			Koch et al., 1982
0.12±0.12	0.44±0.17					Lamb et al., 1990
0.39±0.02	0.36±0.02	0.40±0.02				Winder et al., 1990
0.21	0.56					Kriese et al., 1991
			0.38±0.16			MacNeil et al., 1991
0.09^{c}		0.16^{c}	0.19^{c}	0.18^{c}		Johnston et al., 1992
0.18						Woodward et al., 1992
			0.22±0.10	0.35±0.12		Gilbert et al., 1993
0.17±0.09	0.40±0.12	0.35±0.11		0.15±0.09	0.36±0.15	Koots et al., 1994a
0.34±0.09	0.36±0.09					Gregory et al., 1995
0.20±0.009		0.28±0.02				Meyer, 1995
0.23±0.008		0.31±0.01				Meyer, 1995
0.20±0.07			0.47±0.14			Shepard et al., 1996
0.33		0.48				Tosh et al., 1999
						Kaps et al., 2000
					0.32±0.06	Arthur et al., 2001a
			0.34±0.04	0.33±0.05	0.46±0.04	Arthur et al., 2001b
			0.64			Riley et al., 2002
0.14±0.02						Splan et al., 2002
0.19	0.39	0.48				Koch et al., 2004
			0.23±0.06		0.06±0.04	Robinson and Oddy, 2004

$^{a}h^{2}$ ± SE

^{b}WWT = weaning weight; PWG = postweaning gain; YWT = yearling weight; ADG = average daily gain; RGR = relative growth rate (percent change in weight postweaning); FCR = feed conversion ratio (kg feed/kg gain)

^{c}Standard errors < 0.10

Selection for the improvement of one growth trait is likely to have an effect on other growth traits when they are genetically correlated. Table 2.5 contains phenotypic and genetic correlations between each of the postweaning growth traits from a number of studies. The correlations between weight and gain traits were generally positive and moderate to high indicating that selection for the increase of one weight trait, for example, would yield genetic progress in other weight and gain traits. Especially high, positive genetic correlations were observed between weaning weight and yearling weight (0.47 to 0.78), postweaning gain and yearling weight (0.44 to 0.93), yearling weight and average daily gain (0.83) and average daily gain and relative growth rate (0.71 to 0.81).

The estimates in Table 2.5 also indicate a strong negative relationship between gain and feed conversion. Negative correlations between feed conversion ratio and all other growth traits (-0.21 to -0.90) were observed. A favorable genetic relationship appears to exist between the amount of gain during postweaning growth and the efficiency of that gain, for which a smaller value is more desirable.

Table 2.6 lists published heritability estimates (and associated standard errors) for beef cattle carcass traits that may be important to profitability when cattle are valued on a carcass basis. There is sufficient additive genetic variation for genetic improvement in carcass performance as heritability estimates for all traits were in the moderate to high range. The traits with the largest estimates were longissimus muscle area (0.22 to 0.97), marbling score (0.12 to 0.88) and quality grade (0.47 to 0.62). These are not surprising as the distribution of muscle and of fat in the intramuscular depot occur in early growth (Berg and Butterfield, 1966; Johnson et al., 1973) and would not be expected to be affected to a great extent by management for cattle at similar maturity levels.

Table 2.5. Published estimates of phenotypic (P) and genetic (G) correlations (± SE) between postweaning growth traits in beef cattle.

	Growth traits[a]					
	PWG	YWT	ADG	RGR	FCR	Source
WWT						
P				-0.48		Smith and Cundiff, 1976
G				0.08±0.45		
P						Buchanan et al., 1982
G	0.46±0.18	0.74±0.11				
P	0.10	0.78		-0.45		Winder et al., 1990
G	0.18	0.63		-0.38		
P		0.69	0.18	-0.36		Johnston et al., 1992
G		0.47	-0.10	-0.62		
P	0.12	0.68		-0.31		Koots et al., 1994b
G	0.39	0.78		-0.10		
P						Arthur et al., 2001a
G					-0.21±0.20	
PWG						
P						Buchanan et al., 1982
G		0.93±0.04				
P		0.74				Winder et al., 1990
G		0.75				
P		0.74			-0.64	Koots et al., 1994b
G		0.44			-0.76	
P					-0.08±0.03	Robinson and Oddy, 2004
G					-0.86±0.10	
YWT						
P				0.24		Winder et al., 1990
G				0.25		
P			0.75	0.21		Johnston et al., 1992
G			0.83	0.39		
P				0.36	-0.46	Koots et al., 1994b
G				0.35	-0.60	
FDG						
P				0.77		Smith and Cundiff, 1976
G				0.78±0.15		
P					-0.48	MacNeil et al., 1991
G					-0.43±0.28	
P				0.79		Johnston et al., 1992
G				0.81		
P				0.67		Gilbert et al., 1993
G				0.71±0.15		
P					-0.74	Arthur et al., 2001a
G					-0.62±0.06	
P				0.68	-0.54	Arthur et al., 2001b
G				0.71±0.09	-0.46±0.08	
RGR						
P					-0.72	Koots et al., 194b
G					-0.79	
P					-0.64	Arthur et al., 2001b
G					-0.90±0.04	

[a] WWT = weaning weight; PWG = postweaning gain; YWT = yearling weight; ADG = average daily gain; RGR = relative growth rate (percent change in weight postweaning); FCR = feed conversion ratio (kg feed/kg gain)

Table 2.6. Published heritability estimates for carcass traits in beef cattle[a].

End-point	Carcass traits[b]								Source
	CWT	DP	FT	LMA	MS	QG	YG	%RP	
	0.39		**0.43**	**0.73**		**0.62**	**0.36**	**0.40**	Cundiff et al., 1964
	±0.24[c]		±0.33	±0.29		±0.27	±0.31	±0.24	
			0.43	**0.40**	**0.73**	**0.74**			Brackelsberg et al., 1971
Days	**0.68**		**0.68**	**0.28**	**0.34**				Koch, 1978
	±0.25		±0.25	±0.24	±0.25				
			0.41[d]	**0.56**[d]	**0.40**[d]			**0.63**[d]	Koch et al., 1982
	0.31		**0.24**	**0.28**	**0.33**		**0.24**		Lamb et al., 1990
	±0.15		±0.14	±0.15	±0.15		±0.14		
Weight	**0.24**		**0.49**	**0.46**	**0.35**				Arnold et al., 1991
QG	**0.09**[d]			**0.38**[d]	**0.26**[d]				Jonhston et al., 1992
FT	**0.26**		**0.14**	**0.48**	**0.28**				Gilbert et al., 1993
	±0.16		±0.14	±0.20	±0.17				
Age	**0.31**		**0.26**	**0.32**	**0.26**				Wilson et al., 1993
	±0.04		±0.04	±0.04	±0.04				
Age	**0.45**	**0.35**	**0.43**	**0.43**	**0.37**				Koots et al., 1994a
	±0.12	±0.13	±0.13	±0.14	±0.12				
FT	**0.35**		**0.43**	**0.40**	**0.66**				Koots et al., 1994a
	±0.22		±0.22	±0.22	±0.22				
Weight	**0.30**	**0.51**	**0.44**	**0.48**	**0.37**				Koots et al., 1994a
	±0.13	±0.11	±0.13	±0.23	±0.09				
	0.23	**0.19**	**0.25**	**0.22**	**0.48**				Gregory et al., 1995
	±0.08	±0.08	±0.08	±0.08	±0.09				
Age	**0.59**		**0.27**	**0.39**					Moser et al., 1998
	±0.06		±0.05	±0.06					
	0.60	**0.49**	**0.46**	**0.97**	**0.88**		**0.54**		Pariacote et al., 1998
	±0.19	±0.19	±0.19	±0.21	±0.21		±0.19		
	0.50		**0.66**	**0.61**	**0.71**			**0.66**	Splan et al., 1998
	±0.06		±0.07	±0.06	±0.07			±0.07	
Weight /FT	**0.38**		**0.46**	**0.54**	**0.55**				Crews and Kemp, 2001
	±0.16		±0.18	±0.19	±0.19				
Age	**0.32**		**0.10**	**0.26**	**0.12**			**0.09**	Shanks et al., 2001
Weight			**0.14**	**0.22**	**0.12**			**0.12**	Shanks et al., 2001
MS	**0.30**		**0.10**	**0.28**				**0.09**	Shanks et al., 2001
FT	**0.33**			**0.29**	**0.13**			**0.17**	Shanks et al., 2001
Age	**0.48**		**0.35**	**0.45**	**0.42**				Kemp et al., 2002
FT	**0.55**	**0.77**	**0.63**	**0.44**	**0.44**	**0.47**			Riley et al., 2002

[a] h^2±SE

[b] CWT = carcass weight; DP = dressing percent; FT = 12th rib backfat thickness; LMA = longissimus muscle area; MS = marbling score; QG = USDA quality grade; YG = USDA yield grade; %RP = percent retail product

[c] CWT per day of age

[d] SE < 0.10

The feeding endpoint (where known) is listed with the heritability estimates in Table 2.6. Koots et al. (1994a) found no consistent differences between unadjusted heritability estimates (unknown endpoints) and those adjusted for age, weight or backfat thickness. Shanks et al. (2001) also reported little effect of adjusting heritability estimates to different endpoints.

Genetic and phenotypic correlations between carcass traits are listed in Table 2.7. In general, there were high, positive correlations between carcass weight and dressing percent (0.47 to 0.61), backfat thickness (0.13 to 0.95) and longissimus muscle area (0.02 to 0.66). There was also generally high positive genetic relationships between backfat thickness and marbling score (-0.13 to 0.62), between marbling score and quality grade (0.73 to 1.00) and between longissimus muscle area and percent retail product (0.18 to 0.53). The genetic correlation between marbling score and yield grade was positive and moderate (0.26 to 0.45), but this relationship is usually unfavorable.

There were few consistent negative correlations (Table 2.7) except for the weight and fat traits with percent retail product. The genetic correlation between backfat thickness and percent retail product was negative and ranged from low to high (-0.07 to -0.74). There were also some low to moderate negative genetic correlations between longissimus muscle area and marbling score (-0.12 to 0.44) and yield grade (-0.26 to -0.61). The latter being a favorable relationship. The correlation estimates highlight the well-known and previously discussed antagonism between traits relating to animal fatness and those important to muscle yield.

The relationships between growth and carcass traits are important in gauging the anticipated effect of selection. Table 2.8 contains the phenotypic and genetic correlations

Table 2.7. Published estimates of phenotypic (P) and genetic (G) correlations (± SE) between carcass traits in beef cattle.

	Carcass traits[a]							
	DP	FT	LMA	MS	QG	YG	%RP	Source
CWT								
P		0.31	0.46		0.16		-0.26	Cundiff et al., 1964
G		0.15	0.66		0.47		0.02	
P		0.42	0.37	0.18				Koch, 1978
G		0.95±0.21	0.02±0.46	-0.33±0.39				
P		0.38	0.58	0.28				Lamb et al., 1990
G		0.14±0.37	0.68±0.19	0.64±0.20				
P								Arnold et al., 1991
G		0.36	0.09	0.33				
P			0.51	-0.03				Johnston et al., 1992
G			0.45	-0.31				
P		0.24	0.43	0.08				Wilson et al., 1993
G		0.38	0.47	-0.06				
P	0.42	0.39	0.45	0.15				Koots et al., 1994b
G	0.61	0.38	0.47	0.10				
P		0.28	0.40	0.13			-0.24	Gregory et al., 1995
G		0.13±0.25	0.66±0.20	0.31±0.20			-0.12±0.22	
P	0.41±0.03	0.19±0.04	0.47±0.03	0.09±0.04		0.20±0.04		Pariacote et al., 1998
G								
P	0.24		0.44	0.17	0.16	0.48		Riley et al., 2002
G	0.47		0.52	0.39	0.37	0.56		
DP								
P		0.17	0.08	0.06				Koots et al., 1994b
G		0.31	0.34	0.16				
P		0.19±0.04	0.31±0.04	0.04±0.04		0.07±0.04		Pariacote et al., 1998
G		-0.16±0.31	0.79±0.16	0.08±0.24		-0.56±0.29		
P		0.23	0.16	0.16	0.13	0.24		Riley et al., 2002
G		0.42	0.02	0.35	0.26	0.48		
FT								
P			-0.01		0.05		-0.14	Cundiff et al., 1964
G			0		0.07		-0.07	
P				0.42	0.34			Brackelsberg et al., 1971
G				0.62	0.48			
P			-0.08	0.25				Koch, 1978
G			0.03±0.44	0.73±0.38				
P			-0.15	0.24			-0.74	Koch et al., 1982
G			-0.44	0.16			-0.74	
P			-0.08	0.12				Wilson et al., 1993
G			-0.06	-0.13				
P			-0.09	0.22				Koots et al., 1994b
G			-0.08	0.36				
P			-0.16±0.04	0.20±0.04		0.78±0.02		Pariacote et al., 1998
G			-0.31±0.23	0.26±0.24		0.67±0.15		
P			0.10	0.30	0.31	0.81		Riley et al., 2002
G			0.02	0.56	0.58	0.93		

(Continued)

Table 2.7. *Continued.* Published estimates of phenotypic (P) and genetic (G) correlations (± SE) between carcass traits in beef cattle.

	Carcass traits[a]							
	DP	FT	LMA	MS	QG	YG	%RP	Source
LMA								
P					0.05		0.62	Cundiff et al., 1964
G					0.32		0.18	
P				-0.07	-0.06			Brackelsberg et al., 1971
G				-0.12	-0.09			
P				0.03			0.60	Koch et al., 1982
G				-0.14			0.53	
P				-0.08±0.05		-0.61±0.03		Pariacote et al., 1998
G				-0.17±0.19		-0.85±0.10		
P				0.12	0.10	-0.30		Riley et al., 2002
G				0.44	0.32	-0.26		
MS								
P					0.96			Brackelsberg et al., 1971
G					0.73			
P							-0.07	Koch et al., 1982
G							-0.37	
P							-0.43	Gregory et al., 1995
G							-0.60±0.20	
P						0.22±0.04		Pariacote et al., 1998
G						0.26±0.22		
P					0.96	0.26		Riley et al., 2002
G					1.00	0.45		
QG								
P						0.27		Riley et al., 2002
G						0.48		

[a] CWT = carcass weight; DP = dressing percent; FT = 12th rib backfat thickness; LMA = longissimus muscle area; MS = marbling score; QG = USDA quality grade; YG = USDA yield grade; %RP = percent retail product

between all of the postweaning and carcass traits previously discussed. The genetic correlations between carcass weight and the postweaning weight and gain traits were positive and moderately-high to high. The correlations between backfat thickness and the same postweaning traits were not consistent and ranged from large negative to large positive values. In general, though, increases in phenotypic performance and genetic potential for postweaning gain, feedlot daily gain and relative growth rate would yield increased performance and genetic progress in backfat thickness.

Correlations involving marbling score were also not consistent, but there was a general negative relationship with weaning weight, yearling weight and average daily gain and a positive relationship with postweaning gain and relative growth rate. It is

Table 2.8. Published estimates of phenotypic (P) and genetic (G) correlations (± SE) between postweaning growth and carcass traits in beef cattle.

Carcass Traits[a]	Growth Traits[b]						
	WWT	PWG	YWT	ADG	RGR	FCR	Source
CWT							
P					0.29		Smith and Cundiff, 1976
G					0.72±0.27		
P	0.59		0.94	0.74			Koch, 1978
G	0.48±0.25		0.96±0.03	0.78±0.11			
P		0.64					Lamb et al., 1990
G		0.94±0.04					
P							Arnold et al., 1991
G	0.13		-0.03	0.00	0.10		
P	0.55		0.66	0.44			Johnston et al., 1992
G	0.03		0.32	0.32			
P	0.57	0.60	0.81		0.29		Koots et al., 1994b
G	0.84	0.77	0.91		0.72		
P	0.58						Gregory et al., 1995
G	0.42±0.18						
P				0.87			Riley et al., 2002
G				0.84			
DP							
P	0.20	0.03	0.18			0.06	Koots et al., 1994b
G	0.08	0.07	0.19			0.21	
P				-0.03			Riley et al., 2002
G				-0.01			
FT							
P					0.08		Smith and Cundiff, 1976
G					0.85±0.34		
P	0.12		0.33	0.32			Koch, 1978
G	0.59±0.34		0.86±0.24	0.62±0.21			
P				0.17			Koch et al., 1982
G				0.05			
P	0.20	0.30					Lamb et al., 1990
G	0.49±0.41	0.05±0.34					
P							Arnold et al., 1991
G	-0.28		-0.13	0 .17	0.19		
P	0.16	0.31			0.06	0.15	Koots et al., 1994b
G	0.04	0.32			0.85	-0.24	
P	0.16						Gregory et al., 1995
G	0.15±0.22						
P			0.19				Tosh et al., 1999
G			-0.13				
P				0.39			Riley et al., 2002
G				0.49			
LMA							
P					0.17		Smith and Cundiff, 1976
G					0.46±0.34		
P	0.23		0.35	0.27			Koch, 1978
G	0.16±0.50		0.01±0.46	-0.07±0.38			
P				0.32			Koch et al., 1982
G				0.34			

(Continued)

Table 2.8. *Continued.* Published estimates of phenotypic (P) and genetic (G) correlations (± SE) between postweaning growth and carcass traits in beef cattle.

Carcass Traits[a]	Growth Traits[b]						
	WWT	PWG	YWT	ADG	RGR	FCR	Source
LMA							
P	0.39	0.34					Lamb et al., 1990
G	0.43±0.42	0.48±0.25					
P							Arnold et al., 1991
G	0.33		-0.06	-0.18	-0.13		
P	0.33		0.31	0.13			Johnston et al., 1992
G	-0.27		-0.11	-0.07			
P	0.22	0.25	0.35			0.17	Koots et al., 1994b
G	0.40	0.24	0.29			0.46	
P	0.24						Gregory et al., 1995
G	0.49±0.21						
P				0.39			Riley et al., 2002
G				0.58			
MS							
P	-0.05		0.13	0.20			Koch, 1978
G	-0.02±0.47		-0.57±0.41	-0.62±0.35			
P				0.07			Koch et al., 1982
G				0.15			
P	0.15	0.24					Lamb et al., 1990
G	0.71±0.24	0.48±0.23					
P							Arnold et al., 1991
G	-0.01		0.20	0.54	0.62		
P	-0.05		-0.09	-0.08	-0.04		Johnston et al., 1992
G	-0.55		-0.51	-0.16	0.09		
P	0.02						Woodward et al., 1992
G	0.16						
P	-0.04	0.15	0.14		0.09		Koots et al., 1994b
G	-0.17	0.08	-0.37		1.04		
P	0.01						Gregory et al., 1995
G	0.12±0.17						
P				0.15			Riley et al., 2002
G				0.28			
QG							
P				0.14			Riley et al., 2002
G				0.32			
YG							
P				0.42			Riley et al., 2002
G				0.41			
%RP							
P					-0.10		Smith and Cundiff, 1976
G					-0.54±0.47		
P				-0.15			Koch et al., 1982
G				-0.13			
P	-0.12						Gregory et al., 1995
G	-0.09±0.18						

[a] CWT = carcass weight; DP = dressing percent; FT = 12th rib backfat thickness; LMA = longissimus muscle area; MS = marbling score; QG = USDA quality grade; YG = USDA yield grade; %RP = percent retail product

[b] WWT = weaning weight; PWG = postweaning gain; YWT = yearling weight; ADG = average daily gain; RGR = relative growth rate (percent change in weight postweaning); FCR = feed conversion ratio (kg feed/kg gain)

interesting to note, however, that the one correlation estimate between quality grade and average daily gain is positive, contradicting the marbling score relationship with average daily gain. Few other relationships were large enough or consistent enough to be of interest.

Response to selection for growth and carcass composition. Often it is desirable to predict the rate of genetic change to evaluate alternative selection strategies. The amount of genetic change possible per year for a given trait, assuming phenotypic selection for that trait, can be calculated using (Bourdon, 1997):

$$\Delta G/t = \frac{h^2 i \sigma_P}{L}$$

where t is time, h^2 is the heritability, i is the selection intensity (the selection differential in standard deviation units), σ_P is the phenotypic standard deviation and L is the generation interval. With access to genetic predictions, the heritability can be replaced with the accuracy of selection and phenotypic variation with genetic variation.

Similarly, knowledge of genetic relationships between traits can be used to predict the correlated response in one trait, given selection for another. Correlated response per year in trait Y given phenotypic selection for trait X can be calculated using (Bourdon, 1997):

$$\Delta G_{Y/X} \Big/ t = \frac{r_{g_X g_Y} h_X h_Y i_X \sigma_{P_Y}}{L}$$

where t is time, $r_{g_X g_Y}$ is the genetic correlation between trait X and trait Y, h_X is the square root of heritability for trait X, h_Y is the square root of heritability for trait Y, i_X is the selection intensity for trait X, σ_{P_Y} is the phenotypic variation for trait Y and L is the generation interval. A number of studies have predicted genetic change based on estimated parameters for a particular population (Smith and Cundiff, 1976; Koch, 1978; Koch et al., 1982; Arnold et al., 1991).

Researchers have used these concepts to evaluate the effect on growth and carcass traits of single trait selection on growth and/or carcass traits. Selection for increased weight or growth has historically been used to increase the number of pounds of product available for sale. The correlations in Tables 2.5 and 2.8 would suggest that such selection will likely impact profitability in some way by affecting animal growth and carcass performance. Irgang et al. (1985a) found selection for postweaning gain to be effective in increasing postweaning gain and weaning weight which was supported by Burrow et al. (1991) who found increased weaning weight through selection for high growth rate. Irgang et al. (1985b) also found an increase in feed intake accompanying selection for growth. Bishop et al. (1991) reported increased gains from selection for feed conversion and Seifert (1975) concluded that selecting for weights at older ages was most effective for improving growth rate.

Selection for growth traits appears to have little effect on carcass composition. Perry and Arthur (2000) found no differences in the body composition of steers from lines divergently selected for yearling weight. Similarly, Bishop et al. (1991) reported no effect of selection for feed conversion on yield or quality grades while Burrows et al. (1991) reported a decrease in backfat thickness from selection for high growth rates, but no effects for any other carcass characteristics.

Vieselmeyer et al. (1996) evaluated the effect of selection for marbling score on growth and carcass traits. Selecting for two levels of marbling score, they found differences only in marbling score, and, thus, the percent of animals that grade USDA

Choice. However, the study results indicate that selection for high marbling scores can produce animals that need fewer days on feed to reach acceptable carcass grades and can reach higher levels of marbling without increasing subcutaneous fat levels.

Selection for days to finish

Selection for traits that account for both growth and composition in beef cattle, namely days to finish, may be advantageous for improving profitability. Days to finish (the number of days on feed required for cattle to reach a specified endpoint) is economically relevant in postweaning beef production (Golden et al., 2000), particularly in an integrated system with well-defined costs of production. The number of days cattle are on feed can impact performance for both growth and carcass traits as animals move further along the growth curve. Van Koevering et al. (1993) found that increasing the number of days on feed increased marbling scores, but only to a point; they reported no benefits of feeding steers longer than 133 days. Similarly, Hermesmeyer et al. (2000) showed that increased feeding time improved quality grade and carcass weight, but reduced feed efficiency.

While relatively little research has addressed days to finish in beef cattle, but it has long been of interest in swine production as selection for reduced days to finish weight is practiced to improve economic efficiency (Faust et al., 1992). Several studies have shown that the adequate genetic variation in days to finish makes genetic progress possible. Published heritability estimates for days to finish in swine production are listed in Table 2.9. Heritability estimates ranged from 0.11 to 0.69, depending on the breed and definition of the trait.

Johnston et al. (1992) estimated heritability and correlations with carcass traits for days to a constant backfat endpoint in Charolais cattle. They found days to finish in beef cattle to be moderately heritable at 0.24 (with a standard error less than 0.10). They reported a negative correlation between days to finish and postweaning weight and growth traits. Conversely, days to finish was positively correlated with carcass weight, longissimus muscle area and marbling score. The phenotypic correlation estimates followed the same trend. Newcom et al. (2005) reported similar correlations between days on feed and carcass traits in Duroc pigs.

Table 2.9. Published heritability estimates for days to finish weight in swine[a].

Source	Breed	Trait	$h^2 \pm SE$
Keele et al., 1988	Hampshire	Days to 100 kg	0.11 ± 0.05
	Duroc		0.25 ± 0.01
	Yorkshire		0.22 ± 0.04
Kaplon et al., 1991	Large Polish White	Days to 100 kg	0.26
Li and Kennedy, 1994	Yorkshire	Days to 100 kg	0.31
	Landrace		0.30
	Duroc		0.26
	Hampshire		0.32
Newcom et al., 2005	Duroc	Days to 113.5 kg	0.69 ± 0.12

[a]Adapted from Kuehn, 2000

Recent work has focused on using random regression models for genetic predictions of days to finish. Kuehn (2000) studied the viability of using random regression models that include only an intercept and linear term to produce a genetic evaluation of days to finish weight and days to finish backfat in beef cattle. He found that an average of at least 2.5 observations per animal would be needed to obtain accurate variance components for each trait, but determined that it is possible to predict breeding values for the amount of time required for cattle to reach a specific compositional endpoint. Following Kuehn, Jubileu (2003) compared genetic evaluation for days to finish using random regression techniques to traditional univariate and multivariate models. Jubileu reported that he could successfully calculate breeding values for days to finish weight using the random regression model. The advantage of this approach is that a weight EPD may be calculated at any age (or number of days on feed) or an age EPD derived for any custom weight simply using the following (Kuehn, 2000):

$$\text{EPD (age or weight)} = b_0 + b_1 * \text{(desired endpoint)}$$

where b_0 is the intercept breeding value (or EPD) and b_1 is the linear coefficient breeding value (or EPD) for each individual sire.

The likely interaction between days to finish and feeding endpoint makes selection decisions regarding days to finish a complex matter. Selection to reduce days to finish may cause unintended changes to carcass composition and, thus, profitability.

Literature Cited

Allen, R. E. and D. E. Goll. 2003. Cellular and developmental biology of skeletal muscle as related to muscle growth. In: C. G. Scanes (Ed.) Biology of growth of domestic animals. Iowa State Press, Ames, Iowa.

Amer, P.R., R.A. Kemp, and C. Smith. 1992. Genetic differences among the predominant beef cattle breeds in Canada: An analysis of published results. Can. J. Anim. Sci. 72:759-771.

Arnold, J. W., J. K. Bertrand, L. L. Benyshek, and C. Ludwig. 1991. Estimates of genetic parameters for live animal ultrasound, actual carcass data and growth traits in beef cattle. J. Anim. Sci. 69:985-992.

Arthur, P. F., J. A. Archer, D. J. Johnston, R. M. Herd, E. C. Richardson, and P. F. Parnell. 2001a. Genetic and phenotypic variance and covariance components for feed intake, feed efficiency and other postweaning traits in Angus cattle. J. Anim. Sci. 79:2805-2811.

Arthur, P. F., G. Renand, and D. Krauss. 2001b. Genetic and phenotypic relationships among different measures of growth and feed efficiency in young Charolais bulls. Livest. Prod. Sci. 68:131-139.

Batra, T. R., W. R. Usborne, D. G. Grieve, and E. B. Burnside. 1973. Genotype-environment interaction in calf production. II. Live measurements and carcass traits. J. Anim. Sci. 36:471-475.

Beaver, E. E., J. E. Williams, S. J. Miller, D. L. Hancock, S. M. Hannah, and D. L. O'Conner. 1989. Influences of breed and diet on growth, nutrient digestibility, body composition and plasma hormones of Brangus and Angus steers. J. Anim. Sci 67:2415-2425.

Beever, J. E., P. D. George, R. L. Fernando, C. J. Stoemont, and H. A. Lewin. 1990. Associations between genetic markers and growth and carcass traits in a paternal half-sib family of Angus cattle. J. Anim. Sci. 68:337-344.

Belk, K. E., J. D. Tatum, and F. L. Williams Jr. 1991. Deposition and distribution of carcass fat for steers differing in frame size and muscle thickness. J. Anim. Sci. 69:609-616.

Berg, R. T., B. B. Andersen, and T. Liboriussen. 1978a. Growth of bovine tissues 1. Genetic influences on growth patterns of muscle, fat and bone in young bulls. Anim. Prod. 26:245-258.

Berg, R. T., B. B. Andersen, and T. Liboriussen. 1978b. Growth of bovine tissues 2. Genetic influences on muscle growth and distribution in young bulls. Anim. Prod. 27:51-61.

Berg, R. T., B. B. Andersen, and T. Liboriussen. 1978c. Growth of bovine tissues 3. Genetic influences on patterns of fat growth and distribution in young bulls. Anim. Prod. 27:63-69.

Berg, R. T., B. B. Andersen, and T. Liboriussen. 1978d. Growth of bovine tissues 4. Genetic influences on patterns of bone growth and distribution in young bulls. Anim. Prod. 27:71-77.

Berg, R. T. and R. M. Butterfield. 1966. Muscle:bone ratio and fat percentage as measures of beef carcass composition. Anim. Prod. 8:1-11.

Berg, R. T. and R. M. Butterfield. 1968. Growth of bovine muscle, fat and bone. J. Anim. Sci. 27:611-619.

Berg, R. T. and R. M. Butterfield. 1976. New concepts of cattle growth. John Wiley and Sons, New York.

Berg, R. T., S. D. M. Jones, M. A. Price, F. Fukuhara, R. M. Butterfield, and T. Hardin. 1979. Patterns of carcass fat deposition in heifers, steers and bulls. Can. J. Anim. Sci. 59:359-366.

Bishop, M. D., M. E. Davis, W. R. Harvey, G. R. Wilson, and B. D. VanStavern. 1991. Divergent selection for postweaning feed conversion in Angus beef cattle: I. Mean comparisons. J. Anim. Sci. 69:4348-4359.

Block, H. C., J. J. McKinnon, A. F. Mustafa, and D. A. Christensen. 2001. Manipulation of cattle growth to target carcass quality. J. Anim. Sci. 79:133-140.

Bourdon, R. M. 1997. Understanding Animal Breeding. Prentice-Hall, Inc., Upper Saddle River, NJ.

Brackelsberg, P. O., E. A. Kline, R. L. Willham, and L. N. Hazel. 1971. Genetic parameters for selected beef carcass traits. J. Anim. Sci. 33:13-17.

Brody, S. 1945. Bioenergetics and growth. Reinhold Publishing Corporation, New York.

Brown, J. E., H. A. Fitzhugh Jr., and T. C. Cartwright. 1976. A comparison of nonlinear models for describing weight-age relationships in cattle. J. Anim. Sci. 42:810-818.

Bruns, K. W., R. H. Pritchard, and D. L. Boggs. 2005. The effect of stage of growth and implant exposure on performance and carcass composition in steers. J. Anim. Sci. 83:108-116.

Buchanan, D. S., M. K. Nielsen, R. M. Koch, and L. V. Cundiff. 1982. Selection for growth and muscling score in beef cattle. II. Genetic parameters and predicted response. J. Anim. Sci. 55:526-532.

Burrow, H. M., B. Gulbransen, S. K. Johnson, G. P. Davis, W. R. Shorthose, and R. F. Elliott. 1991. Consequences of selection for growth and heat resistance on growth, feed conversion efficiency, commercial carcass traits and meat quality of Zebu crossbred cattle. Aust. J. Agric. Res. 42:1373-1383.

Butterfield, R. M. 1962. Prediction of muscle content of steer carcasses. Nature 195:193-195.

Butterfield, R. M. 1965. Practical implications of anatomical research in beef cattle. Proc. New Zealand Soc. Anim. Prod. 25:152-163.

Butterfield, R. M. 1966. The effect of nutritional stress and recovery on the body composition of cattle. Res. Vet. Sci. 7:168-179.

Butterfield, R. M. 1975a. Fat: Its role in survival and carcass composition. Beef Cattle Science Handbook 12:136-139.

Butterfield, R. M. 1975b. Muscle growth patterns in cattle. Beef Cattle Science Handbook 12:130-135.

Butterfield, R. M. 1976. Beef carcass composition - Our knowledge to 1976. Beef Cattle Science Handbook 13:139-144.

Butterfield, R. M. and R. T. Berg. 1966a. A classification of bovine muscles, based on their relative growth patterns. Res. Vet. Sci. 7:326-332.

Butterfield, R. M. and R. T. Berg. 1966b. Relative growth patterns of commercially important muscle groups of cattle. Res. Vet. Sci. 7:389-393.

Butterfield, R. M., E. R. Johnson, and J. W. Pryor. 1971. A study of growth in calves I. Carcass tissues. J. Agric. Sci., Camb. 76:453-458.

Butterfield, R. M., W. J. Pryor, and R. T. Berg. 1966. A study of carcase growth in calves. Res. Vet. Sci. 7:417-423.

Callow, E. H. 1948. Comparative studies of meat II. The changes in the carcass during growth and fattening, and their relation to the chemical composition of the fatty and muscular tissues. J. Agric. Sci. 38:174-198.

Callow, E. H. 1961. Comparative studies of meat VII. A comparison between Hereford, Dairy Shorthorn and Friesian steers on four levels of nutrition. J. Agric. Sci. 56:265-282.

Callow, E. H. 1962. Comparative studies of meat VIII. The percentage of fat in the fatty and muscular tissues of steers and the iodine number of the extracted fat, as affected by breed and level of nutrition. J. Agric. Sci. 58:295-307.

Camfield, P. K., A. H. Brown Jr., P. K. Lewis, L. Y. Rakes, and Z. B. Johnson. 1997. Effects of frame size and time-on-feed on carcass characteristics, sensory attributes and fatty acid profiles of steers. J. Anim. Sci. 75:1837-1844.

Casas, E., R. T. Stone, J. W. Keele, S. D. Shackelford, S. M. Kappes, and Koohmaraie. 2001. A comprehensive search for quantitative trait loci affecting growth and carcass composition of cattle segregating alternative forms of the myostatin gene. J. Anim. Sci. 79:854-860.

Charles, D. D. and E. R. Johnson. 1976. Breed differences in amount and distribution of bovine carcass dissectible fat. J. Anim. Sci. 42:332-341.

Cianzio, D. S., D. G. Topel, G. B. Whitehurst, D. C. Beitz, and H. L. Self. 1982. Adipose tissue growth in cattle representing two frame sizes: Distribution among depots. J. Anim. Sci. 55:305-312.

Crews Jr., D. H. and R. A. Kemp. 2001. Genetic parameters for ultrasound and carcass measures of yield and quality among replacement and slaughter beef cattle. J. Anim. Sci. 79:3008-3020.

Cundiff, L. V., D. Chambers, D. F. Stephens, and R. L. Willham. 1964. Genetic analysis of some growth and carcass traits in beef cattle. J. Anim. Sci. 23:1133-1138.

DeRouen, S. M., D. E. Franke, T. D. Bidner, and D. C. Blouin. 1992a. Direct and maternal genetic effects for carcass traits in beef cattle. J. Anim. Sci. 70:3677-3685.

DeRouen, S. M., D. E. Franke, T. D. Bidner, and D. C. Blouin. 1992b. Two-, three-, and four-breed rotational crossbreeding of beef cattle: Carcass traits. J. Anim. Sci. 70:3665-3676.

Dolezal, H. G., J. D. Tatum, and F. L. Williams. 1993. Effects of feeder cattle frame size, muscle thickness, and age class on days fed, weight, and carcass composition. J. Anim. Sci. 71:2975-2985.

Dubeski, P. L., J. L. Aalhus, S. D. M. Jones, A. K. W. Tong, and W. M. Robertson. 1997. Fatttening heifers to heavy weights to enhance marbling: Efficiency of gain. Can. J. Anim. Sci. 77:625-633.

Duckett, S. K., D. G. Wagner, L. D. Yates, H. G. Dolezal, and S. G. May. 1993. Effects of time on feed on beef nutrient composition. J. Anim. Sci. 71:2079-2088.

Eisen, E. J. 1976. Results of growth curve analyses in mice and rats. J. Anim. Sci. 42:1008-1023.

Farquharson, C. 2003. Bone growth. In: C. G. Scanes (Ed.) Biology of growth of domestic animals. Iowa State Press, Ames, Iowa.

Faust, M. A., M. W. Tess, and O. W. Robison. 1992. A bioeconomic simulation model for a hierarchical swine breeding structure. J. Anim. Sci. 70:1760-1774.

Fitzhugh Jr., H. A. 1976. Analysis of growth curves and strategies for altering their shape. J. Anim. Sci. 42:1036-1051.

Fluharty, F. L., S. C. Loerch, T. B. Turner, S. J. Moeller, and G. D. Lowe. 2000. Effects of weaning age and diet on growth and carcass characteristics in steers. J. Anim. Sci. 78:1759-1767.

Foutz, C. P., H. G. Dolezal, T. L. Gardner, D. R. Gill, J. L. Hensley, and J. B. Morgan. 1997. Anabolic implant effects on steer performance, carcass traits, subprimal yields, and longissimus muscle properties. J. Anim. Sci. 75:1256-1265.

Fowler, V. R. 1968. Body development and some problems of its evaluation. Growth and Development of Mammals. pp. 195-211.

Gerken, C. L., J. D. Tatum, J. B. Morgan, and G. C. Smith. 1995. Use of genetically identical (clone) steers to determine the effects of estrogenic and androgenic implants on beef quality and palatability characteristics. J. Anim. Sci. 73:3317-3324.

Gettys, T. W., D. M. Hendricks, P. M. Burrows, and B. D. Schanbacher. 1987. Partition of food intake between maintenance and gain among bovine sex phenotypes. Anim. Prod. 44:209-217.

Gilbert, R. P., D. R. C. Baily, and N. H. Shannon. 1993. Body dimensions and carcass measurements of cattle selected for postweaning gain fed two different diets. J. Anim. Sci. 71:1688-1698.

Golden, B. L., D. J. Garrick, S. Newman, and R. M. Enns. 2000. Economically relevant traits, a framework for the next generation of EPDs. Proc. Beef Improvement Federation, Wichita, KS, pp 2-13.

Goonewardene, L. A., R. T. Berg, and R. T. Hardin. 1980-1981. Relative merits of growth curves as applied to beef cattle. J. Nat. Agric. Soc. Ceylon 17/18:73-84.

Gregory, K. E., L. V. Cundiff, and R. M. Koch. 1995. Genetic and phenotypic (co)variances for growth and carcass traits of purebred and composite populations of beef cattle. J. Anim. Sci. 73:1920-1926.

Gregory, K. E., L. V. Cundiff, R. M. Koch, M. E. Dikeman, and M. Koohmaraie. 1994. Breed effects, retained heterosis, and estimates of genetic and phenotypic parameters for carcass and meat traits of beef cattle. 72:1174-1183.

Grings, E. E., R. E. Short, M. D. MacNeil, M. R. Haferkamp, and D. C. Adams. 1996. Efficiency of production in cattle of two growth potentials on northern great plains rangelands during spring-summer grazing. J. Anim. Sci. 74:2317-2326.

Guiroy, P. J., L. O. Tedeschi, D. G. Fox, and J. P. Hutcheson. 2002. The effects of implant strategy on finished body weight of beef cattle. J. Anim. Sci. 80:1791-1800.

Hammond, J. 1932. Growth and development of mutton qualities in sheep. Oliver and Boyd, Edinburgh.

Hassen, A., D. E. Wilson, and G. H. Rouse. 1998. Evaluation of carcass, live, and real-time ultrasound measures in feedlot cattle: I. Assessment of sex and breed effects. J. Anim. Sci. 76:273-282.

Hermesmeyer, G. N., L. L. Berger, T. G. Nash, and R. T. Brandt Jr. 2000. Effects of energy intake, implantation, and subcutaneous fat end point on feedlot steer performance and carcass composition. J. Anim. Sci. 78:825-831.

Hironaka, R., B. H. Sonntag, and G. C. Kozub. 1979. Effects of feeding programs and diet energy on rate of gain, efficiency of digestible energy utilization, and carcass grades of steers. Can. J. Anim. Sci. 59:385-394.

Irgang, R., E. U. Dillard, M. W. Tess, and O. W. Robison. 1985a. Selection for weaning weight and postweaning gain. II. Response to selection. J. Anim. Sci. 60:1142-1155.

Irgang, R., E. U. Dillard, M. W. Tess, and O. W. Robison. 1985b. Selection for weaning weight and postweaning gain in Hereford cattle. III. Correlated responses to selection in milk yield, preweaning and postweaning traits. J. Anim. Sci. 60:1156-1164.

Johnson, E. R., R. M. Butterfield, and W. J. Pryor. 1972. Studies of fat distribution in the bovine carcass I. The partition of fatty tissues between depots. Aust. J. Agic. Res. 23:381-388.

Johnson, E. R., W. J. Pryor, and R. M. Butterfield. 1973. Studies of fat distribution in the bovine carcass II. Relationship of intramuscular fat to the quantitative analysis of the skeletal musculature. Aust. J. Agric. Res. 24:287-296.

Johnston, D. J., L. L. Benyshek, J. K. Bertrand, M. H. Johnson, and G. M. Weiss. 1992. Estimates of genetic parameters for growth and carcass traits in Charolais cattle. Can. J. Anim. Sci. 72:493-499.

Jones, S. D. M. 1985. Carcass tissue yield and distribution in three biological types of cattle fed grain or forage-based diets. Can. J. Anim. Sci. 65:363-374.

Jones, S. D. M., M. A. Price, and R. T. Berg. 1978. Effects of breed and sex on the relative growth and distribution of bone in cattle. Can. J. Anim. Sci. 58:157-165.

Jones, S. D. M., M. A. Price, and R. T. Berg. 1980a. Fattening patterns in cattle 1. Fat partition among the depots. Can. J. Anim. Sci. 60:843-850.

Jones, S. D. M., M. A. Price, and R. T. Berg. 1980b. Fattening patterns in cattle 2. Fat distribution among the wholesale cuts. Can. J. Anim. Sci. 60:851-856.

Jones, S. D. M., M. A. Price, and R. T. Berg. 1980c. The growth and distribution of muscle in bulls and heifers of two breeds. Can. J. Anim. Sci. 60:669-675.

Jones, S. D. M., M. A. Price, and R. T. Berg. 1981. The influence of breed on the efficiency of growth and muscle deposition in bulls and heifers. Can. J. Anim. Sci. 61:237-242.

Jubileu, J. S. 2003. The use of random regression models to predict days to finish in beef cattle. Master's thesis. Colorado State University, Fort Collins.

Kaplon, M. J., M. F. Rothschild, P. J. Berger, and M. Healey. 1991. Population parameter estimates for performance and reproductive traits in Polish Large White nucleus herds. J. Anim. Sci. 69:91-98.

Kaps, M., W. O. Herring, and W. R. Lamberson. 2000. Genetic and environmental parameters for traits derived from the Brody growth curve and their relationships with weaning weight in Angus cattle. J. Anim. Sci. 78:1436-1442.

Keele, J. W., R. K. Johnson, L. D. Young, and T. E. Socha. 1988. Comparison of methods of predicting breeding values of swine. J. Anim. Sci. 66:3040-3048.

Kemp, D. J., W. O. Herring, and C. J. Kaiser. 2002. Genetic and environmental parameters for steer ultrasound and carcass traits. J. Anim. Sci. 80:1489-1496.

Kempster, A. J., A. Cuthbertson, and R. J. Smith. 1976. Variation in lean distribution among steer carcasses of different breeds and crosses. J. Agric. Sci., Camb. 87:533-542.

Knoblich, H. V., F. L. Fluharty, and S. C. Loerch. 1997. Effects of programmed gain strategies on performance and carcass characteristics of steers. J. Anim. Sci. 75:3094-3102.

Koch, R. M. 1978. Selection in beef cattle. III. Correlated response of carcass traits to selection for weaning weight, yearling weight and muscling score in cattle. J. Anim. Sci. 47:142-150.

Koch, R. M., L. V. Cundiff, and K. E. Gregory. 1982. Heritabilities and genetic, environmental and phenotypic correlations of carcass traits in a population of diverse biological types and their implications in selection programs. J. Anim. Sci. 55:1319-1329.

Koch, R. M., L. V. Cundiff, K. E. Gregory, and L. D. Van Vleck. 2004. Genetic response to selection for weaning weight or yearling weight or yearling weight and muscle score in Hereford cattle: Efficiency of gain, growth, and carcass characteristics. J. Anim. Sci. 82:668-682.

Koch, R. M., M. E. Dikeman, D. M. Allen, M. May, J. D. Crouse, and D. R. Campion. 1976. Characterization of biological types of cattle III. Carcass composition, quality and palatability. J. Anim. Sci. 43:48-62.

Koots, K. R., J. P. Gibson, C. Smith, and J. W. Wilton. 1994a. Analyses of published genetic parameter estimates for beef production traits. 1. Heritability. Anim. Breed. Abstr. 62:309-338.

Koots, K. R., J. P. Gibson, and J. W. Wilton. 1994b. Analyses of published genetic parameter estimates for beef production traits. 2. Phenotypic and genetic correlations. Anim. Breed. Abstr. 62:825-853.

Korver, S., M. W. Tess, T. Johnson, and B. B. Andersen. 1987. Size-scaled lean and fat growth patterns of serially slaughtered beef animals. J. Anim. Sci. 64:1292-1301.

Kreikemeier, W. M. and T. L. Mader. 2004. Effects of growth-promoting agents and season on yearling feedlot heifer performance. J. Anim. Sci. 82:2481-2488.

Kriese, L. A., J. K. Bertrand, and L. L. Benyshek. 1991. Genetic and environmental growth trait parameter estimates for Brahman and Brahman-derivative cattle. J. Anim. Sci. 69:2362-2370.

Kuehn, L. A. 2000. Parameterization of random regression models for beef cattle data sets. Master's thesis. Colorado State University, Fort Collins.

Lamb, M. A., O. W. Robison, and M. W. Tess. 1990. Genetic parameters for carcass traits in Hereford bulls. J. Anim. Sci. 68:64-69.

Li, X. and B. W. Kennedy. 1994. Genetic parameters for growth rate and backfat in Canadian Yorkshire, Landrace, Duroc, and Hampshire pigs. J. Anim. Sci. 72:1450-1454.

Loerch, S. C. and F. L. Fluharty. 1998. Effects of programming intake on performance and carcass characteristics of feedlot cattle. J. Anim. Sci. 76:371-377.

Long, R. A. 1976. The importance of composition of gain. Beef Cattle Science Handbook 13:145-150.

Lunt, D. K., G. C. M. C. E. Smith, J. W. Savell, and Z. L. Carpenter. 1985. Carcass characteristics and composition of Brahman, Angus and Brahman X Angus steers fed for different times-on-feed. Meat Science 14:137-152.

MacNeil, M. D., D. R. C. Bailey, J. J. Urick, R. P. Gilbert, and W. L. Reynolds. 1991. Heritabilities and genetic correlations for postweaning growth and feed intake of beef bulls and steers. J. Anim. Sci. 69:3183-3189.

Mader, T. L. 2003. Environmental stress in confined beef cattle. J. Anim. Sci. 81:E110-E119.

Mandell, I. B., E. A. Gullett, J. G. Buchanan-Smith, and C. P. Campbell. 1997a. Effects of diet and slaughter endpoint on carcass composition and beef quality in Charolais cross steers. Can. J. Anim. Sci. 77:403-414.

Mandell, I. B., E. A. Gullett, J. W. Wilton, O. B. Allen, and V. R. Osborne. 1997b. Effects of diet, breed and slaughter endpoint on growth, performance, carcass composition and beef quality traits in Limousin and Charolais steers. Can. J. Anim. Sci. 77:23-32.

Marple, D. 2003. Fundamental concepts of growth. In: C. G. Scanes (Ed.) Biology of growth of domestic animals. pp. 9-19. Iowa State Press, Ames, Iowa.

May, S. G., H. G. Dolezal, D. R. Gill, F. K. Ray, and D. S. Buchanan. 1992. Effects of days fed, carcass grade traits, and subcutaneous fat removal on postmortem muscle characteristics and beef palatability. J. Anim. Sci. 70:444-453.

Meyer, K. 1995. Estimates of genetic parameters and breeding values for New Zealand and Australian Angus cattle. Aust. J. Agric. Res. 46:1219-1229.

Moser, D. W., J. K. Bertrand, I. Misztal, L. A. Kriese, and L. L. Benyshek. 1998. Genetic parameter estimates for carcass and yearling ultrasound measurements in Brangus cattle. J. Anim. Sci. 76:2542-2548.

Mukhoty, H. and R. T. Berg. 1971. Influence of breed and sex on the allometric growth patterns of major bovine tissues. Anim. Prod. 13:219-227.

Mukhoty, H. and R. T. Berg. 1973. Influence of breed and sex on muscle weight distribution of cattle. J. Agric. Sci. Camb. 81:317-326.

Mukhoty, H., R. T. Berg, and C. M. Grieve. 1970. Proportions of major bovine tissues as influenced by rations based on barley or oats. Can. J. Anim. Sci. 50:253-258.

Newcom, D. W., T. J. Baas, C. R. Schwab, and K. J. Stadler. 2005. Genetic and phenotypic relationships between individual subcutaneous backfat layers and percentage of longissimus intramuscular fat in Duroc swine. J. Anim. Sci. 83:316-323.

Owens, F. N., P. Dubeski, and C. F. Hanson. 1993. Factors that alter the growth and development of ruminants. J. Anim. Sci. 71:3138-3150.

Owens, F. N., D. R. Gill, D. S. Secrist, and S. W. Coleman. 1995. Review of some aspects of growth and development of feedlot cattle. J. Anim. Sci. 73:3152-3172.

Pariacote, F., L. D. Van Vleck, and R. E. Hunsley. 1998. Genetic and phenotypic parameters for carcass traits of American Shorthorn beef cattle. J. Anim. Sci. 76:2584-2588.

Patterson, D. L., M. A. Price, and R. T. Berg. 1985. Patterns of muscle, bone, and fat accretion in three biological types of feedlot bulls fed three dietary energy levels. Can. J. Anim. Sci. 65:351-361.

Perotto, D., R. I. Cue, and A. J. Lee. 1992. Comparison of nonlinear functions for describing the growth curve of three genotypes of dairy cattle. Can. J. Anim. Sci. 72:773-782.

Perry, D. and P. F. Arthur. 2000. Correlated responses in body composition and fat partitioning to divergent selection for yearling growth rate in Angus cattle. Anim. Prod. 62:143-153.

Price, M. A., S. D. M. Jones, G. W. Mathison, and R. T. Berg. 1980. The effects of increasing dietary roughage level and slaughter weight on the feedlot performance and carcass characteristics of bulls and steers. Can. J. Anim. Sci. 60:345-358.

Pryor, W. J. and L. Laws. 1972. Feeding wheat to cattle 1. The effect of grain to roughage ratio, gain processing and sodium bicarbonate supplementation on productivity and health in steers. Aust. Vet. J. 48:500-503.

Purchas, R. W., D. L. Burnham, and S. T. Morris. 2002. Effects of growth potential and growth path on tenderness of beef longissimus muscle from bulls and steers. J. Anim. Sci. 80:3211-3221.

Pusillo, G. M., M. P. Hoffman, and H. L. Self. 1991. Effects of placing cattle on feed at two-month intervals and housing on feedlot performance and carcass grades. J. Anim. Sci. 69:443-450.

Riley, D. G., C. C. Chase Jr., A. C. Hammond, R. L. West, D. D. Johnson, T. A. Olson, and S. W. Coleman. 2002. Estimated genetic parameters for carcass traits of Brahman cattle. J. Anim. Sci. 80:955-962.

Robinson, D. L. and V. H. Oddy. 2004. Genetic parameters for feed efficiency, fatness, muscle area and feeding behavior of feedlot finished beef cattle. Anim. Prod. 90:255-270.

Robison, O. W. 1976. Growth patterns in swine. J. Anim. Sci. 42:1024-1035.

Roeber, D. L., R. C. Cannell, K. E. Belk, R. K. Miller, J. D. Tatum, and G. C. Smith. 2000. Implant strategies during feeding: Impact on carcass grades and consumer acceptability. J. Anim. Sci. 78:1867-1874.

Rossi, J. E., S. C. Loerch, S. J. Moeller, and J. P. Schoonmaker. 2001. Effects of programmed growth rate and days fed on performance and carcass characteristics of feedlot steers. J. Anim. Sci. 79:1394-1401.

Rumsey, T. S., A. C. Hammond, and T. H. Elsasser. 1999. Responses to an estrogenic growth promoter in beef steers fed varying nutritional regimens. J. Anim. Sci. 77:2865-2872.

Sainz, R. D., F. De la Torre, and J. W. Oltjen. 1995. Compensatory growth and carcass quality in growth-restricted and refed beef steers. J. Anim. Sci. 73:2971-2979.

Sainz, R. D. and R. F. Vernazza Paganini. 2004. Effects of different grazing and feeding periods on performance and carcass traits of beef steers. J. Anim. Sci. 82:292-297.

Seifert, G. W. 1975. Effectiveness of selection for growth rate in Zebu x British crossbred cattle. II. Post-weaning growth and genetic estimates. Aust. J. Agric. Res. 26:1093-1108.

Shahin, K. A. and R. T. Berg. 1985a. Fat growth and partitioning among the depots in double muscled and normal cattle. Can. J. Anim. Sci. 65:295-306.

Shahin, K. A. and R. T. Berg. 1985b. Growth and distribution of bone in double muscled and normal cattle. Can. J. Anim. Sci. 65:319-332.

Shahin, K. A. and R. T. Berg. 1985c. Growth and distribution of muscle in double muscled and normal cattle. Can. J. Anim. Sci. 65:307-318.

Shahin, K. A. and R. T. Berg. 1985d. Growth patterns of muscle, fat and bone, and carcass composition of double muscled and normal cattle. Can. J. Anim. Sci. 65:279-294.

Shahin, K. A. and R. T. Berg. 1987. Influence of bone growth on muscle growth and bone-muscle relationships in double-muscled and normal cattle. Anim. Prod. 44:219-225.

Shanks, B. C., M. W. Tess, D. D. Kress, and B. E. Cunningham. 2001. Genetic evaluation of carcass traits in Simmental-sired cattle at different slaughter end points. J. Anim. Sci. 79:595-604.

Shepard, H. H., R. D. Green, B. L. Golden, K. E. Hamlin, T. L. Perkins, and J. B. Diles. 1996. Genetic parameter estimates of live animal ultrasonic measures of retail yield indicators in yearling breeding cattle. J. Anim. Sci. 74:761-768.

Short, R. E., E. E. Grings, M. D. MacNeil, R. K. Heitschmidt, C. B. Williams, and G. L. Bennett. 1999. Effects of sire growth potential, growing-finishing strategy, and time on feed performance, composition and efficiency of steers. J. Anim. Sci. 77:2406-2417.

Simms, D. D., T. B. Goehring, R. T. Brandt Jr., G. L. Kuhl, J. J. Higgins, S. B. Laudert, and R. W. Lee. 1988. Effect of sequential implanting with zeranol on steer lifetime performance. J. Anim. Sci. 66:2736-2741.

Smith, G. M. and L. V. Cundiff. 1976. Genetic analysis of relative growth rate in crossbreed and straightbred Hereford, Angus and Shorthorn steers. J. Anim. Sci. 43:1171-1175.

Smith, E. G. and G. W. Rahnefeld. 1988. Economic value of feeder calves between breed crosses. Can. J. Anim. Sci. 68:1041-1050.

Smith, G. C., H. G. Dolezal, J. W. Savell, and Z. L. Carpenter. 1981. Time-on-feed effects on carcass and palatability characteristics of steers and heifers. Beef Cattle Science Handbook 18:237-242.

Smith, G. M., D. B. Laster, L. V. Cundiff, and K. E. Gregory. 1976. Characterization of biological types of cattle II. Postweaning growth and feed efficiency of steers. J. Anim. Sci. 43:37-47.

Splan, R. K., L. V. Cundiff, M. E. Dikeman, and L. D. Van Vleck. 2002. Estimates of parameters between direct and maternal genetic effects for weaning weight and diirect genetic effects for carcass traits in crossbred cattle. J. Anim. Sci. 80:3107-3111.

Splan, R. K., L. V. Cundiff, and L. D. Van Vleck. 1998. Genetic parameters for sex-specific traits in beef cattle. J. Anim. Sci. 76:2272-2278.

Tatum, J. D., H. G. Dolezal, F. L. Williams Jr., R. A. Bowling, and R. E. Taylor. 1986a. Effects of feeder-cattle frame size and muscle thickness on subsequent growth and carcass development. II. Absolute growth and associated changes in carcass composition. J. Anim. Sci. 62:121-131.

Tatum, J. D., F. L. Williams Jr., and R. A. Bowling. 1986b. Effects of feeder-cattle frame size and muscle thickness on subsequent growth and carcass development. III. Partitioning of separable carcass fat. J. Anim. Sci. 62:132-138.

Toelle, V. D., M. W. Tess, T. Johnson, and B. B. Andersen. 1986. Lean and fat growth patterns of serially slaughtered beef bulls fed different energy levels. J. Anim. Sci. 63:1347-1360.

Tosh, J. J., R. A. Kemp, and D. R. Ward. 1999. Estimates of direst and maternal genetic parameters for weight traits and backfat thickness in a multibreed population of beef cattle. Can. J. Anim. Sci. 79:433-439.

Urick, J. J., M. D. MacNeil, and W. L. Reynolds. 1991. Biological type effects on postweaning growth, feed efficiency and carcass characteristics of steers. J. Anim. Sci. 69:490-497.

Van Koevering, M. T., D. R. Gill, F. N. Owens, H. G. Dolezal, and C. A. Strasia. 1993. Effect of slaughter date on performance and carcass quality of feedlot steers. Oklahoma Agric. Exp. Sta. Rpt. pp191-196.

Van Koevering, M. T., D. R. Gill, F. N. Owens, H. G. Dolezal, and C. A. Strasia. 1995. Effect of time on feed on performance of feedlot steers, carcass characteristics, and tenderness and composition of longissimus muscles. J. Anim. Sci. 73:21-28.

Vieselmeyer, B. A., R. J. Rasby, B. L. Gwartney, C. R. Calkins, R. A. Stock, and J. A. Gosey. 1996. Use of expected progeny differences for marbling in beef: I. Production Traits. J. Anim. Sci. 74:1009-1013.

Wertz, E., L. L. Berger, P. M. Walker, D. B. Faulkner, F. K. McKeith, and S. Rodriguez-Zas. 2001. Early weaning and postweaning nutritional management affect feedlot performance of Angus X Simmental heifers and the relationship of 12th rib fat and marbling score to feed efficiency. J. Anim. Sci. 79:1660-1669.

Wheeler, T. L., L. V. Cundiff, S. D. Shackelford, and M. Koohmaraie. 2004. Characterization of biological types of cattle (Cycle VI): Carcass, yield, and longissimus palatability traits. J. Anim. Sci. 82, 1177-1189.

Wheeler, T. L., L. V. Cundiff, S. D. Shackelford, and M. Koohmaraie. 2005. Characterization of biological types of cattle (Cycle VII): Carcass, yield, and longissimus palatability traits. J. Anim. Sci. 83:196-207.

Wilson, D. E., R. L. Willham, S. L. Northcutt, and G. H. Rouse. 1993. Genetic parameters for carcass traits estimated from Angus field records. J. Anim. Sci. 71:2365-2370.

Winder, J. A., J. S. Brinks, R. M. Bourdon, and B. L. Golden. 1990. Genetic analysis of absolute growth measurements, relative growth rate and restricted selection indices in Red Angus cattle. J. Anim. Sci. 68:330-336.

Wood, J. D. 1982. Factors controlling fat deposition in meat animals. Proc. New Zealand Soc. Anim. Prod. 42:113-116.

Woodward, B. W., E. J. Pollak, and R. L. Quaas. 1992. Parameter estimation for carcass traits including growth information of Simmental beef cattle using Restricted Maximum Likelihood with a Multiple-Trait Model. J. Anim. Sci. 70:1098-1109.

Wyatt, W. E., T. D. Bidner, P. E. Humes, D. E. Franke, and D. C. Blouin. 2002. Cow-calf and feedlot performance of Brahman-derivative breeds. J. Anim. Sci. 80:3037-3045.

CHAPTER III

PREDICTION OF POSTWEANING PERFORMANCE OF CATTLE: REVIEW OF LITERATURE

Introduction

In recent years there has been an increase in the number of beef cattle marketed under programs that value individual carcass composition. Under such systems, the costs of feeding cattle to a particular harvest endpoint, and the carcass performance at that endpoint are important to producers. Ideally, animals are marketed at an optimum finish point that maximizes profitability. Predictions of postweaning growth, compositional changes during growth and the resulting composition at harvest are required to identify these endpoints. Tools and techniques that have been used to predict postweaning cattle performance are reviewed in this chapter, with particular emphasis on the finishing phase of beef production.

Prediction of carcass composition using live animal evaluation

Beef cattle have long been subjectively evaluated for carcass composition using physical observations of perceived finish. A subjective appraisal is only appropriate if it is an accurate indicator of composition (Busch et al., 1969). It is useful to quantify accuracy of subjective assessments. Bass et al. (1981) found that visual conformation systems only accounted for a small amount of variation in carcass traits, as compared to objective systems where physical measurements of carcass conformation and composition were recorded.

A number of researchers have evaluated objective techniques for predicting carcass traits using various live animal measurements or scores and their feasibility has often been determined by the correlation between live and carcass traits. Brackelsberg et al. (1967) found moderate to high positive correlations between a needle probe of backfat over the *longissimus dorsi* at the 12th rib and carcass backfat and percentage fat trim. Busch et al. (1969) evaluated body measurements, conformation scores and estimates of carcass traits to predict edible portion. They found that slaughter weight controlled most of the variation in edible portion. In a review of total carcass density, Jones et al. (1978) concluded that various measures of carcass density have the potential to predict carcass fatness since fat is less dense than muscle. These are only a small sample of prediction methods that have been investigated but they demonstrate the interest in predicting economically important carcass traits. Much of the current research, however, has focused on using real-time ultrasound techniques to predict carcass composition and assess compositional changes throughout the finishing period.

Prediction using real-time ultrasound. Real-time ultrasound (RTU) has been investigated in recent decades for application in the growing and finishing phases of livestock production. For beef cattle production, RTU has been suggested for the prediction of lean composition at harvest, days on feed to a target compositional endpoint and carcass chemical composition. Several points throughout the feeding period including feedlot arrival, (re)implant time or measures made near the end of feeding, may be useful for producers in predicting composition (Houghton and Turlington, 1992).

RTU measurements can be effective if they are made on traits in the live animal related to economically important carcass traits (Wilson, 1992) and if that relationship is well defined.

Numerous researchers have evaluated RTU measurements in live animals that are thought to determine the yield, and therefore value of the carcass. Backfat thickness and longissimus muscle area are the traits most commonly measured. Real-time ultrasound measurements of backfat thickness have been shown to be highly phenotypically correlated (generally > 0.8) with carcass backfat thickness (Perkins, et al., 1992; Smith et al., 1992; Griffin et al., 1999; May et al., 2000; Greiner et al., 2003). Perkins et al. (1992) indicated a less precise measurement at higher levels of fat, but Brethour (1992) concluded that RTU measurements can be more precise than actual carcass measures.

Correlations between RTU measurements of longissimus muscle area and carcass longissimus muscle area are not as large as those found for backfat thickness. Correlations between live and carcass measures were generally moderate (Perkins, et al., 1992; Smith et al., 1992; Griffin et al., 1999; May et al., 2000; Greiner et al., 2003) indicating that accurate prediction of longissimus muscle area in the live animal may be problematic. Berg and Butterfield (1976) indicate, however, that the longissimus muscle area is lowly correlated with actual carcass muscling and we may do well to look at other predictors of carcass yield.

Measures of the longissimus muscle area using RTU are poor predictors of retail product yield (Hamlin et al., 1995), while fat traits (primarily backfat thickness) are better predictors of yield. Backfat thickness has been shown to be strongly negatively correlated with various measures of carcass yield (Hassen et al., 1999; May et al., 2000;

Wolcott et al., 2001; Greiner et al., 2003). Walcott et al. (2001) found that there was a consistent negative relationship between backfat thickness and retail yield regardless of finish endpoint and Hassen et al. (1999) showed that RTU measurements almost 100 days prior to harvest were nearly as accurate as measurements made just prior to harvest.

It is also of interest to predict marbling score in live animals as carcass quality grade (determined by marbling score) is economically important. Brethour (1990) evaluated ultrasound speckle as a predictor of carcass marbling score and found that speckle scores were highly repeatable and strongly correlated with carcass marbling. He also found that marbling score could be predicted as much as 148 days before harvest. Herring et al. (1998) evaluated RTU systems for predicting marbling finding that marbling score could be precisely predicted using any of four systems (ultrasound units), though two of the RTU systems were more precise at lower levels of intramuscular fat.

The logical extension of RTU research is the development of equations that use RTU measurements to predict finish or carcass performance and allow the sorting of animals into groups with similar time to finish (Houghton and Turlington, 1992). Basarab et al. (1999) described such a situation in which the Kansas State University (KSU) sorting system is used to predict future yield (based on RTU backfat thickness), future quality grade (based on RTU marbling score), and the resulting days to finish prediction for clustering of like cattle groups. Brethour (2000) introduced equations to predict accumulation of marbling during the feeding period and Wall et al., (2004) reported equations to predict quality and yield grade at extended periods of time before harvest. A number of other prediction equations have also been derived from empirical data to aid in management decisions concerning carcass composition, such as equations for backfat thickness, marbling score and longissimus muscle area (Hamlin et al., 1995; Hassen et al., 1999; Nash et al., 2000; Wolcott et al., 2001; Greiner et al., 2003).

Models of postweaning growth and composition

The objective of any model is to provide an abstract representation of reality (Bolte, 1998). Computer models of agricultural systems fall under this definition and are often expressed in mathematical and logical terms, but, like all models, are limited as they can never fully represent reality (Mayer, 2002). The development of systems models of agricultural production, for both scientific purposes and as an aid in decision-making, has been given increased emphasis with advances in computer technology (Gaunt et al., 1997; Sørensen, 1998). Mayer (2002) described a range of uses for valid systems models including: as a substitute for manipulations and experiments that are not possible in the real world, to quantify complex interactions that exist in the real system, identification of optimum management strategies, to evaluate long-term effects of management options, for experimental control over environmental conditions, exploration of hypothetical situations or insight into the relative importance of system variables. Development of a valid model, however, is a complex process.

Modeling agricultural systems. An important aspect of systems-oriented research is the consideration of interactions and effects of innovations that previously were overlooked (Sørensen, 1998). Systems models are often used for resource allocation, production management enhancement and understanding biological processes (Bolte, 1998). The model-building process should proceed in a structured manner and should not be merely based on the development of a set of equations to describe a certain set of data (Gaunt et al., 1997), but should include the activities described by Gaunt et al. (1997) and listed in Table 3.1. Bolte (1998) provided a more simplified view of the modeling process, suggesting that the steps to construct a model include: identification of important features of a system, development of an abstract conceptual framework describing the system, formalization of the conceptual framework (including development of system components) and implementation and testing of the model. The type of model to implement will generally depend on the desired use and outcome.

Table 3.1. Stages in the modeling process[a].

Stage	Requirement
Purpose established	Specification of: Process Spatial scale Data requirements
Model selection	Model functionality to match data availability
Model compounding	Compatibility of scale and complexity in models
Evaluation and testing	Error and uncertainty analysis Quantitative analysis
Application	Decision support/shells for operational model

[a]From Gaunt et al., 1997

Models can be classified in a number of ways. Sørensen (1998) distinguishes between mathematical programming models and simulation models, where the former includes linear and dynamic models and incorporates some form of optimization. Mathematical models find the combination of inputs that yields an optimum output and are often the basis for decision support tools. Simulation models are often used for research, contain no optimization routine and are generally used for exploring outcomes of various experimental situations. Quoting France and Thornly (1984), Sørensen (1998) further classifies simulation models as static versus dynamic, empirical versus mechanistic and deterministic versus stochastic. The last classification is of most interest for modeling agricultural systems.

Kinghorn (1993) defines the deterministic and stochastic approaches in relation to the evaluation of livestock breeding program design. A deterministic model uses mathematical equations to predict a result, which is fully dependent on the input parameters. A stochastic model is based on probability distributions where results for each run of the model will be different, unless the same random seed is used. Most agricultural models are of a deterministic nature (especially in livestock production), however Anderson (1976) argued that deterministic approaches disregard "the intrinsic uncertainties in embedded generalized relationships" and only work correctly under restrictive assumptions. Knap et al. (2003) further warn that a reduction in the number of model parameters (as in deterministic modeling) may lead to a number of assumptions that are often of a subjective nature. Sandland and McGilchrist (1979) suggested that growth, in particular, is a stochastic process and a model should include environmental randomness, rather than adding some residual effect onto an expected value as an afterthought. In terms of simplicity, though, deterministic models are often necessary as they are computationally fast and are conducive to decision support software (Kinghorn, 1993). A compromise may be to include stochastic processes for certain components of a deterministic system (e.g. survival).

Whatever the model classification, the orientation of model development can be important. Many agricultural models have been developed using procedural languages, such as FORTRAN (Pan et al., 2000), but an object-oriented approach may be more appropriate when dealing with complex natural systems. Object-oriented programming

(OOP) is a scheme that seeks to represent real-world systems as objects in the program and simplifies the translation of concepts to code (Van Evert and Campbell, 1994; Sequeira et al., 1997). More simply, OOP uses objects in the program to mirror objects in the real world, one-to-one (Bolte, 1998). Elements for developing a biological model using OOP are listed in Table 3.2.

Table 3.2. Elements of system design for a biological model using OOP[a].

Components	Identify natural components, i.e. the smallest unit (given a pre-determined level of interest) for which we can observe a response to driving factors (e.g. environment).
Attributes	Determine what attributes associated with each component should be included in the model.
Behavior	Evaluate (model) the response to other system components and to the driving factors associated with those natural components. This determines what operations can be performed on each component.
Kind-of (inheritance) hierarchy	Determine the hierarchy of system components, i.e. the taxonomy of objects.
Part-of links	Determine the structural linkage between system components.

[a]From Sequeira et al., 1997

Object-oriented design approaches are particularly appropriate for decision support systems due to the complexities in managing biological, physical, environmental and economic interactions (Bolte, 1998). Concepts of OOP provide for a needed decision support framework that includes: a standardized public interface that defines object access, mechanisms for synchronizing flow control between system components, capability for components to communicate with other components in a nonspecific manner and standardized methodologies for collecting and transferring information between system components. More generally, OOP allows for needed representation in the system (i.e. objects as real-world entities) and re-use of code (Bolte, 1998). Models developed in the Java programming language, in particular, have been shown to fully support and simplify all elements of the object model (Saleh, 1999).

Models to predict postweaning growth. Predicting growth and development of beef cattle in terms of daily gains, feed efficiency, days to finish, etc., or on a more fundamental level in terms of cellular processes, is of interest to researchers and producers alike. Researchers have predicted growth as controlled by chronological age, sex, mature size, energy intake, hormonal status, relative turnover of tissues and cell number (DNA) and activity (Owens et al., 1995). A wide variety of models have been developed to aid in describing the growth process and, for this discussion, generally fall into two categories: models to predict some form of energy requirements or feed intake during growth, and models to predict changes in weight or body composition as a result of growth at a certain level of energy. Often models of one type incorporate elements of the other. Models that reflect the entire system will be addressed in a subsequent section.

The determination of energy requirements for growth and maintenance or of daily feed intake is important for developing ration formulations, predicting feed efficiency, calculating feed costs and selecting for least-cost gains. Fox et al. (1988) described a model for determining the requirements of growing cattle adjusted for animal and environmental variations. This model was further applied in the Cornell Net Carbohydrate and Protein System (CNCPS) (Fox et al., 1995) to predict feed requirements for cattle on forage. Similarly, the Georgia Feedlot model (Ely et al., 1978) predicts energy requirements for feedlot growth based on the California Net Energy System and was designed to formulate total rations for beef cattle.

The prediction of animal intake is a function of animal feed requirements and may also be influenced by feed quality and quantity (Pittroff and Kothmann, 2001). Daily feed intake estimates are important components of efficiency of gain and have been modeled in a number of ways. Song and Dinkel (1978) proposed a mathematical model to predict postweaning feed intake based on physiological demand for energy as affected by maturity, breed and diet. Loewer et al. (1983b) proposed alterations to the California Net Energy System to calculate daily feed intake based on feed availability, nutrient composition, potential skeletal size and heat loss potential. Fox et al. (2002) described the determination of feed intake for animals fed in groups used in the Cornell Value Discovery System (CVDS), based on equations presented by Guiroy et al. (2001). The CVDS contains a sub-model to calculate dry matter intake and daily gain for each day on feed. Output from these models could be used in research or production as a basis for predicting future gains.

The knowledge of cattle weight and composition changes during growing and finishing aids in management and marketing decisions. Various methods have been used to predict these changes. Loewer et al. (1983a) modeled body composition (fat, protein, water and various minerals) to determine physiological age and to predict changes in empty body weight. Each component was assumed to follow a sigmoidal curve and changes were affected by the amount of available energy. Fox and Black (1984) described a model to predict body composition of cattle as a means to predict net nutrient requirements and performance. The model accounted for variations in frame size, breed type, environment, growth stimulants, previous nutritional treatment and the use of feed additives as factors that influence consumption. Modeling the growing period, Korver et al. (1988) predicted lean and fat gain of beef cattle, dependent on a prediction of empty body weight, as influenced by breed and feeding level. Oltjen et al. (1986) modeled empty body weight and composition changes at the cellular level adjusting for effects of body condition and nutrition level.

The weight and composition of the empty body are primary drivers in models for predicting growth as a more precise index of the energy and nutrient content of the body (Owens et al., 1995). Williams et al. (1992b) developed a computer model to estimate empty body weight (EBW) in terms of unfasted body weight and characteristics of the diet, as opposed to calculating EBW as simply a percentage of body weight (Owens et al., 1995), which may not correctly account for factors unrelated to body weight. Keele et al. (1992) further described a model to predict differences in empty body composition of cattle with similar genotypes and EBW due to the plane of nutrition. The model is useful for evaluating compositional changes not due to EBW, particularly in situations affected by compensatory gains, and accurately predicts the effects of nutrition on fatness (Williams et al., 1992a). Williams and Jenkins (1998) proposed a further-refined model to predict the composition of EBW changes for cattle at all ages, and found that they could accurately predict empty body composition at slaughter.

A number of models have incorporated predictions of intake and weight and composition changes over time. Keener (1979) modeled the partitioning of energy utilization in cattle to predict feed intake, based on feed availability, hunger drive, animal size, feed palatability and rates of protein and fat deposition. The model was not intended for estimating growth when nutrition is restricted, however. Rayburn and Fox (1990) refined the model of Fox et al. (1988) to predict the growth performance of Holstein steers. They were able to predict intake and gain under different feeding systems and environments. Rotz et al. (2005) described a recent model to simulate feed intake and growth performance for any class of cattle in the production system. The model was designed as one component of a systems model to simulate whole-herd production.

Models to predict carcass composition. Predicting carcass composition is especially useful in the finishing phase and aids in identifying the individual endpoints that are optimum for harvest, in terms of profitability. The predicted carcass components can be compared to a packer pricing scenario to determine revenue as one part of a profit function. Many of the components are estimated using equations or models derived from empirical data, especially in cases where the models are not part of a larger systems model (which are discussed in a subsequent section). Such predictions may not be useful in situations unlike experimental conditions on which the model was based where sources of variation may differ, but model adjustments can improve usability.

The longissimus muscle area is an important component of the yield grade equation, and thus economically important, but its prediction in the live animal can be difficult (as discussed in an earlier section). Field and Schoonover (1967) developed an equation to estimate longissimus muscle area using weight observations in bulls. They found their equation to be useful for predicting average, rather than individual, performance. Perry and Fox (1997) presented an equation to predict longissimus muscle area from ultrasound measures and shrunk body weight, but warned that an adjustment to the equation would be necessary for different ultrasound equipment and operators. Crews, Jr. (2001) showed that longissimus muscle area could be estimated from ultrasound measures of longissimus muscle width and depth in individual animals, and is seemingly an improvement over previous methods.

A number of models have been developed to predict multiple carcass components, including those directly related to carcass value and those that are the basis for value traits (e.g., muscle, fat, etc.). Song and Dinkel (1978) developed a mathematical model to predict physical and chemical compositions of the carcass. Model equations included live weight, mature weight, degree of maturity, diet and biological type to predict the weight of total separable muscle, fat and bone and to estimate the caloric value of live weight prior to slaughter. Johnson (1994) predicted the composition of the carcass, in terms of percent bone, fat and muscle using measurements of anal fold thickness, backfat thickness and empty body weight for steers and heifers. Also presented were equations to predict the weight of each component tissue.

Perry and Fox (1997) described a system of equations to predict carcass fat and yield by live measurements, including ultrasound measurements of backfat thickness, longissimus muscle area, and marbling and estimates of empty body weight and fatness. The system was designed to determine the point of optimum profitability, in terms of carcass revenue and costs of gain. Herring et al (1994) presented a set of equations to predict the percentage and weight of retail product and trimmable fat in the carcass. Each was developed using empirical evidence as the basis for predictions that were based on subjective and objective measurements on the live animal. The Perry and Fox system was designed, in part, to identify optimum finish endpoints, while Herring et al. were more interested in specifically predicting cutability at defined finish endpoints.

Models of postweaning systems. The development of models involves the use of mathematical representations of the underlying biology, economics and management practices of the system (Bourdon, 1988). Bioeconomic models have been developed to predict optimal genotypes for all manner of production systems and can generally be viewed in two categories: as simulation models for research purposes or models designed for decision support. Often the systems modeled include the cow/calf, stocker/grower, finish and harvest segments of the production chain. Such models will be reviewed below with particular emphasis on the postweaning phase.

An early model of a beef production system described by Wilton et al. (1974) used a linear programming technique to determine the optimum combination of resources, given some constraint, to maximize or minimize an objective function, such as profitability. The model included cropping, feeding and breeding activities that could be limited by finances, land, buildings, machinery and labor, and economic considerations relating to traits such as calving ease, rates of gain and mature weight. The inputs form a major part of the model and include market prices, crop yields, animal feed requirements, type of production system and simulation length, labor requirements and various measures of animal performance. The model output included the optimal combination of resources, given pre-determined constraints (such as cow size), and the associated costs and revenue for cow/calf and/or feedlot enterprises. A limitation of such a model is the requirement to input levels of animal performance that may be difficult to obtain, such as feed intake.

The Texas A&M (TAMU) Cattle Production Systems Model (Sanders and Cartwright, 1979a,b) was developed to simulate levels of beef cattle herd performance based on available feed resources and animal potential. The deterministic model was designed to simulate the production of cattle varying widely in genotype and to consider any number of management and environmental conditions. The whole herd was simulated with specific subroutines for growth (lean tissue growth rate, body weight gain or loss and milk production levels), fertility and survivability. Each component interacted fully with all other components for the simulation of any production system. Though postweaning growing and finishing were not specifically considered, included in the simulated animal classes were steers, whose disposition was determined by the user. Growth rates and feed intake were calculated for steers (and other classes) using a monthly time step, allowing performance based on genetic potential to be simulated.

Other researchers have revised the TAMU Model, to include feedlot and harvest performance of cattle (Notter et al., 1979a,b,c; Kahn and Spedding, 1983; Bourdon and Brinks, 1987; Meek and Kilpatrick, 1991). Kahn and Spedding (1983) adapted the model for use in small herds representative of developing countries. They altered the model structure to allow for the simulation of individual animal performance, the use of variable time steps and additional management options. Further they treated randomly occurring discrete events (conception, mortality and sex) in a stochastic, rather than deterministic, manner. Bourdon and Brinks (1987) also altered the TAMU Model to reflect a typical Northern plains production system. They refined the calculation of genetic potential and of intake, and added a subroutine to account for the effects of cold weather on feed digestibility and maintenance requirements.

Meek and Kilpatrick (1991) added a linear programming, least-cost ration approach to the TAMU Model for use in research and decision support in Northern Ireland. This model allows for economically optimum growth and reduces production to two time periods, winter and summer. An economic model was developed to determine optimum daily gain, or the rate of gain that maximized marginal profit by calculating the least-cost diet for each rate of daily live weight gain. The model was designed to be versatile and adaptable to a number of production situations, but Meek and Kilpatrick (1991) suggest that errors exist in equations to simulate nutrient intake.

Koots and Gibson (1998) described a deterministic model for a Canadian integrated beef production system used to derive economic values for a number of traits when the objective was to maximize profit. Sixteen traits were determined to affect costs and returns and for terminal animals included: residual postweaning growth rate, residual feed intake, residual slaughter weight at constant backfat thickness, dressing percentage at constant backfat thickness, marbling and lean meat yield, where the residual traits indicated the difference between actual and predicted values. Performance for each trait could be simulated and a set of profit functions used to determine the profit per year based on herd potential. Pang et al. (1999) also presented a model of Canadian beef production in the Alberta Beef Production Simulation System. They modeled an integrated production system using four major components: herd inventory, nutrient requirements, forage production and economics. The economic sub-model calculated net return for the system, including the sale of weaned, backgrounded and finished calves. The system was modeled in a deterministic manner, with the exception of weather variables, which were generated stochastically due to their unpredictable nature.

A systems model to predict energy and protein requirements for maintenance and gain to predict herd replacements for optimum growth, animal performance, cost of gain and carcass value was generally described by Fox et al. (1985). The model contained a number of sub-models for diet formulation and performance prediction and accounted for a large number of physical and environmental factors believed to impact maintenance requirements, body composition and composition of gain in the cow herd and calves in the postweaning phase. The model can be used to identify optimum feeding systems, management practices and compositional endpoints for cattle in the feedlot.

A systems model on a much larger scale was developed by Miller et al. (1980) to describe the beef production system from conception to harvest for the entire United States. The dynamic model divided the country into five regions to allow the evaluation of various management strategies on the weaned calf supplies and proportion of cattle that grade USDA Choice. The authors assumed Choice grade production was optimum and thus the model was designed to identify the feeding options that would maximize the number of cattle grading Choice. The model can be used for evaluating the effect of feed constraints on regional production levels.

A number of models exist strictly for production in the postweaning phase, generally evaluating growth and time on feed, carcass composition at harvest and associated economic values. Such models can be useful when only postweaning growth and composition is considered in the decision-making activity, or when an integrated system is of interest but other appropriate/available models are designed for the cow/calf segment only.

A deterministic linear programming model was described by Brennan and Hoffman (1989) for use in simulating production of a mid-western farm-feedlot. The model included a function for maximizing net returns as affected by type of ration, feeding management system and marketing alternatives. Growth and intake were modeled based on values obtained from other feeding experiments.

Amer et al. (1994) developed a bioeconomic model to describe the growth of cattle in the feedlot to their optimal finish endpoint. The model incorporated expected feed requirements, labor, postweaning carcass growth, postweaning fat deposition and value at slaughter. Total costs and returns were calculated per day based on animal performance as the animal matured. The animal reached the optimal endpoint when the present value of profits were maximized. Equations were used to adjust cattle performance based on genotype so that genotypes could be compared, and future slaughter criteria determined at each optimal endpoint. Similarly, Williams and Bennett (1995) presented a bioeconomic model to determine the slaughter endpoint that maximized profit. The model simulated daily changes in carcass value of groups of feeder cattle grown under any postweaning production system. To determine carcass value, predictions of empty body weight, yield grade, backfat thickness and marbling were incorporated. Daily costs were also calculated to predict daily profit. Like Amer et al. (1994) the model can be used to compare different genotypes at their optimum endpoint.

Tedeschi et al. (2004) presented a mechanistic, dynamic model for predicting growth rate, accumulated weight, days to finish, carcass weight and carcass composition of cattle in the feedlot. The model accounts for numerous factors: animal; environmental;

and management, in the prediction of daily weight and composition changes for individual animals. The model is also designed to assign feed intake to individual animals in group feeding situations, based on data collected at harvest. Using predictions of costs associated with feeding, and predictions of revenue determined by the carcass composition and market prices, profitability may be calculated each day so animals are harvested at the most profitable point in growth.

Literature Cited

Amer, P. R., R. A. Kemp, J. G. Buchanan-Smith, G. C. Fox, and C. Smith. 1994. A bioeconomic model for comparing beef cattle genotypes at their optimal economic slaughter end point. J. Anim. Sci. 72:38-50.

Anderson, J. R. 1976. Essential probabilities in modelling. Agric. Syst. 1:219-231.

Basarab, J. A., J. R. Brethour, D. R. ZoBell, and B. Graham. 1999. Sorting feeder cattle with a system that integrates ultrasound backfat and marbling estimates with a model that maximizes feedlot profitability in value-based marketing. Can. J. Anim. Sci. 79:327-335.

Bass, J. J., D. L. Johnson, F. Colomer-Rocher, and G. Binks. 1981. Prediction of carcass composition from carcass conformation in cattle. J. Agric. Sci., Camb. 97:37-44.

Berg, R. T. and R. M. Butterfield. 1976. New concepts of cattle growth. John Wiley and Sons, New York.

Bolte, J. 1998. Object-oriented programming for decision systems. In: R. M. Peart and R. B. Curry (Eds.) Agricultural Systems Modeling and Simulation. pp. 629-650. Marcel Dexter, Inc., New York.

Bourdon, R. M. 1988. Bovine Nirvana - From the perspective of a modeler and purebred breeder. J. Anim. Sci. 66:1892-1898.

Bourdon, R. M. and J. S. Brinks. 1987. Simulated efficiency of range beef production I. Growth and milk production. J. Anim. Sci. 65:943-955.

Brackelsberg, P. O., R. L. Willham, and L. E. Walters. 1967. Probing beef cattle to predict carcass fatness. J. Anim. Sci. 26:713-717.

Brennan, R. W. and M. P. Hoffman. 1989. Computer simulation of a cattle feedlot production system. J. Anim. Sci. 67:1116-1127.

Brethour, J. R. 1990. Relationship of ultrasound speckle to marbling score in cattle. J. Anim. Sci. 68:2603-2613.

Brethour, J. R. 1992. The repeatability and accuracy of ultrasound in measuring backfat of cattle. J. Anim. Sci. 70:1039-1044.

Brethour, J. R. 2000. Using serial ultrasound measures to generate models of marbling and backfat thickness changes in feedlot cattle. J. Anim. Sci. 78:2055-2061.

Busch, D. A., C. A. Dinkel, and J. A. Minyard. 1969. Body measurements, subjective scores and estimates of certain carcass traits as predictors of edible portion in beef cattle. J. Anim. Sci. 29:557-566.

Crews, Jr. D. H. 2001. Alternative ultrasound predictors of beef carcass longissimus muscle area. Prof. Anim. Sci. 17:303-308.

Ely, L. O., M. E. McCullough, and J. R. Allison. 1978. The development, evaluation and use of the Georgia feedlot model. Res. Bull. Ga. Exp. Stn. 209.

Field, R. A. and C. O. Schoonover. 1967. Equations for comparing *longissimus dorsi* areas in bulls of different weights. J. Anim. Sci. 26:709-712.

Fox, D. G., M. C. Barry, R. E. Pitt, D. K. Roseler, and W. C. Stone. 1995. Application of the Cornell net carbohydrate and protein model for cattle consuming forages. J. Anim. Sci. 73:267-277.

Fox, D. G. and J. R. Black. 1984. A system for predicting body composition and performance of growing cattle. J. Anim. Sci. 58:725-739.

Fox, D. G., C. J. Sniffen, and J. D. O'Connor. 1988. Adjusting nutrient requirements of beef cattle for animal and environmental variations. J. Anim. Sci. 66:1475-1495.

Fox, D. G., C. J. Sniffen, P. J. VanSoest, J. R. Russell, P. D. George, and P. H. Robinson. 1985. Using the microcomputer to predict the effect of animal, feed and environmental variations on the performance of cattle. Computer applications in feeding and management of animals, Madison, WI, pp 121-139.

Fox, D. G., L. O. Tedeschi, and M. J. Baker. 2002. Determining post-weaning feed efficiency in beef cattle. Proc. Beef Improvement Federation., Omaha, NE, pp 44-66.

France, J. and J. H. M. Thornly. 1984. Mathematical models in agriculture. Butterworth, London.

Gaunt, J. L., J. Riley, A. Stein, and F. W. T. Penning de Vries. 1997. Requirements for effective modelling strategies. Agric. Syst. 54:153-168.

Greiner, S. P., G. H. Rouse, D. E. Wilson, L. V. Cundiff, and T. L. Wheeler. 2003. The relationship between ultrasound measurements and carcass fat thickness and longissimus muscle area in beef cattle. J. Anim. Sci. 81:676-682.

Griffin, D. B., Savell, H. A. Recio, R. P. Garrett, and H. R. Cross. 1999. Predicting carcass composition of beef cattle using ultrasound technology. J. Anim. Sci. 77:889-892.

Guiroy, P. J., D. G. Fox, L. O. Tedeschi, and M. J. C. M. D. Baker. 2001. Predicting individual feed requirements of cattle fed in groups. J. Anim. Sci. 79:1983-1995.

Hamlin, K. E., R. D. Green, L. V. Cundiff, T. L. Wheeler, and M. E. Dikeman. 1995. Real-time ultrasonic measurement of fat thickness and longissimus muscle area: II. Relationship between real-time ultrasound measures and carcass retail yield. J. Anim. Sci. 73:1725-1734.

Hassen, A., D. E. Wilson, and G. H. Rouse. 1999. Evaluation of Carcass, Live, and Real-Time Ultrasound Measures in Feedlot Cattle: II. Effects of Different Age End Points on the Accuracy of Predicting the Percentage of Retail Product, Retail Product Weight, and Hot Carcass Weight. J. Anim. Sci. 77:283-290.

Herring, W. O., L. A. Kriese, J. K. Bertrand, and J. Crouch. 1998. Comparison of four real-time ultrasound systems that predict intramuscular fat in beef cattle. J. Anim. Sci. 76:364-370.

Houghton, P. L. and L. M. Turlington. 1992. Application of ultrasound for feeding and finishing: A review. J. Anim. Sci. 70:930-941.

Johnson, E. R. 1994. Comparison of the prediction of carcass components using fat thickness measurements in heifers and steers. Aust. J. Exp. Agric. 34:435-438.

Jones, S. D. M., M. A. Price, and R. T. Berg. 1978. A review of carcass density, its measurement and relationship with bovine carcass fatness. J. Anim. Sci. 46:1151-1157.

Kahn, H. E. and C. R. W. Spedding. 1983. A dynamic model for the simulation of cattle herd production systems: Part 1 - General description and the effects of simulation techniques on model results. Agric. Syst. 12:101-111.

Keele, J. W., C. B. Williams, and G. L. Bennett. 1992. A computer model to predict the effects of level of nutrition on composition of empty body gain in beef cattle: I. Theory and development. J. Anim. Sci. 70:841-857.

Keener, H. M. 1979. Simulation of energy utilisation of bovine animals. Agric. Syst. 4:79-100.

Kinghorn, B. P. 1993. Deterministic versus stochastic approaches to evaluating the design of livestock breeding programs. Design of Livestock Breeding Programs, Armidale, AGBU, University of New England, pp 111-125.

Knap, P. W., R. Roehe, K. Kolstad, C. Pomar, and P. Luiting. 2003. Characterization of pig genotypes for growth modeling. J. Anim. Sci. 81:E1878-E195.

Koots, K. R. and J. P. Gibson. 1998. Economic values for beef production traits from a herd level bioeconomic model. Can. J. Anim. Sci. 78:29-45.

Korver, S., M. W. Tess, and T. Johnson. 1988. A model of growth and growth composition for beef bulls of different breeds. Agric. Syst. 27:279-294.

Loewer, O. J., E. M. Smith, N. Gay, and R. Fehr. 1983a. Incorporation of environment and feed quality into a net energy model for beef cattle. Agric. Syst. 11:67-94.

Loewer, O. J., E. M. Smith, K. L. Taul, L. W. Turner, and N. Gay. 1983b. A body comparison model for predicting beef animal growth. Agric. Syst. 10:245-256.

May, S. G., W. L. Mies, J. W. Edwards, J. J. Harris, J. B. Morgan, R. P. Garrett, F. L. Williams, J. W. Wise, H. R. Cross, and J. W. Savell. 2000. Using live estimates and ultrasound measurements to predict beef carcass cutability. J. Anim. Sci. 78:1255-1261.

Mayer, D. G. 2002. Evolutionary algorithms and agricultural systems. Kluwer Academic Publishers, Boston, MA.

Meek, R. M. and D. J. Kilpatrick. 1991. An economic nutrition model of beef production. Agric. Syst. 36:105-116.

Miller, W. C., G. M. Ward, T. P. Yorks, D. L. Rossiter, and J. J. Combs. 1980. A mathematical model of the United States beef production system. Agric. Syst. 5:295-307.

Nash, S. A., S. N. Harrison, J. H. Packham, R. P. Panting, and S. K. Duckett. 2000. Case study: Monitoring changes in carcass quality across time-on-feed using real-time ultrasound to optimize marketing endpoints. Prof. Anim. Sci. 16:202-205.

Notter, D. R., J. O. Sander, G. E. Dickerson, G. M. Smith, and T. C. Cartwright. 1979a. Simulated efficiency of beef production for a midwestern cow-calf feedlot management system. I. Milk Production. J. Anim. Sci. 49:70-82.

Notter, D. R., J. O. Sander, G. E. Dickerson, G. M. Smith, and T. C. Cartwright. 1979b. Simulated efficiency of beef production for a midwestern cow-calf feedlot management system. II. Mature Body Size. J. Anim. Sci. 49:83-91.

Notter, D. R., J. O. Sander, G. E. Dickerson, G. M. Smith, and T. C. Cartwright. 1979c. Simulated efficiency of beef production for a midwestern cow-calf feedlot management system. III. Crossbreeding Systems. J. Anim. Sci. 49:92-102.

Oltjen, J. W., A. C. Bywater, and R. L. Baldwin. 1986. Development of a dynamic model of beef cattle growth and composition. J. Anim. Sci. 62:86-97.

Owens, F. N., D. R. Gill, D. S. Secrist, and S. W. Coleman. 1995. Review of some aspects of growth and development of feedlot cattle. J. Anim. Sci. 73:3152-3172.

Pan, X., J. D. Hesketh, and M. G. Huck. 2000. OWSimu: an object-oriented and web-based simulator for plant growth. Agric. Syst. 63:33-47.

Pang, H., M. Makarechian, J. A. Basarab, and R. T. Berg. 1999. Structure of a dynamic simulation model for beef cattle production systems. Can. J. Anim. Sci. 79:409-417.

Perkins, T. L., R. D. Green, and K. E. Hamlin. 1992. Evaluation of ultrasonic estimates of carcass fat thickness and longissimus muscle area in beef cattle. J. Anim. Sci. 70:1002-1010.

Perry, T. C. and D. G. Fox. 1997. Predicting carcass composition and individual feed requirement in live cattle widely varying in body size. J. Anim. Sci. 75:300-307.

Pittroff, W. and M. M. Kothmann. 2001. Quantitative prediction of feed intake in ruminants II. Conceptual and mathematical analysis of models for cattle. Livest. Prod. Sci. 71:151-169.

Rayburn, E. B. and D. G. Fox. 1990. Predicting growth and performance of Holstein steers. J. Anim. Sci. 68:788-798.

Rotz, C. A., D. R. Buckmaster, and J. W. Comerford. 2005. A beef herd model for simulating feed intake, animal performance, and manure excretion in farm systems. J. Anim. Sci. 83:231-242.

Saleh, K. 1999. Object oriented model in Java: elements and application. Information and Software Technology 41:235-241.

Sanders, J. O. and T. C. Cartwright. 1979a. A general cattle production systems model. I. Structure of the model. Agric. Syst. 4:217-227.

Sanders, J. O. and T. C. Cartwright. 1979b. A general cattle production systems model Part 2 - Procedures used for simulating animal performance. Agric. Syst. 4:289-309.

Sandland, R. L. and C. A. McGilchrist. 1979. Stochastic growth curve analysis. Biometrics 35:255-271.

Sequeira, R. A., R. L. Olson, and J. M. McKinnon . 1997. Implementing generic, object-oriented models in biology. Ecological Modelling 94:17-31.

Smith, M. T., J. W. Oltjen, H. G. Dolezal, D. R. Gill, and B. D. Behrens. 1992. Evaluation of ultrasound for prediction of carcass fat thickness and longissimus muscle area in feedlot steers. J. Anim. Sci. 70:29-37.

Song, H. and C. A. Dinkel. 1978. Mathematical model of postweaning growth, feed intake and carcass composition of beef cattle I. Empirical predictive model of voluntary feed intake from weaning to slaughter. J. Anim. Sci. 47:56-69.

Sorensen, J. T. 1998. Modeling and simulation in applied livestock production science. In: R. M. Peart and R. B. Curry (Eds.) Agricultural Systems Modeling and Simulation. pp. 475-494. Marcel Dexter, Inc., New York.

Tedeschi, L. O., D. G. Fox, and P. J. Guiroy. 2004. A decision support system to improve individual cattle management. 1. A mechanistic, dynamic model for animal growth. Agric. Syst. 79:171-204.

Van Evert, F. K. and G.S. Campbell. 1994. CropSyst: a collection of object-oriented simulation models of agricultural systems. Agron. J. 86:325-331.

Wall, P. B., G. H. Rouse, R. G. Wilson, Jr. R. G. Tait, and W. D. Busby. 2004. Use of ultrasound to predict body composition changes in steers at 100 and 65 days before slaughter. J. Anim. Sci. 82:1621-1629.

Williams, C. B. and G. L. Bennett. 1995. Application of a computer model to predict optimum slaughter end points for different biological types of feeder cattle. J. Anim. Sci. 73:2903-2915.

Williams, C. B. and T. G. Jenkins. 1998. A computer model to predict composition of empty body weight changes in cattle at all stages of maturity. J. Anim. Sci. 76:980-987.

Williams, C. B., J. W. Keele, and G. L. Bennett. 1992a. A computer model to predict the effects of level of nutrition on composition of empty body gain in beef cattle: II. Evaluation of the model. J. Anim. Sci. 70:858-866.

Williams, C. B., J. W. Keele, and D. R. Waldo. 1992b. A computer model to predict empty body weight in cattle from diet and animal characteristics. J. Anim. Sci. 70:3215-3222.

Wilson, D. E. 1992. Application of ultrasound for genetic improvement. J. Anim. Sci. 70:973-983.

Wilton, J. W., C. A. Morris, E. A. Jenson, A. O. Leigh, and W. C. Pfeiffer. 1974. A linear programming model for beef cattle production. Can. J. Anim. Sci. 54:693-707.

Wolcott, M. L., J. M. Thompson, and D. Perry. 2001. The prediction of retail beef yield from real time ultrasound measurements on live animals at three stages though growout and finishing. Aust. J. Exp. Agric. 41:1005-1011

CHAPTER IV

DEVELOPMENT AND VALIDATION OF A POSTWEANING DECISION SUPPORT SYSTEM

Introduction

An object-oriented model was developed as a basis for a decision support system (DSS) for sire selection in postweaning beef production. The DSS is designed to allow users to compare alternative sires for the profitability of progeny in the feedlot at selected compositional endpoints and to identify optimum finish endpoints based on a sire's genetic potential for weight and composition each day on feed. This chapter describes the equations used by the model, and the development and validation of the postweaning DSS.

Model purpose

The purpose of the model is to predict profitability of steers and heifers in the feedlot whether fed to an average endpoint or to an individualized endpoint accounting for genetic differences. To determine profitability, daily feed intake, daily growth and daily changes in carcass composition for steers and heifers, accounting for genetic differences in weight and composition each day, were needed to compute production costs and revenue. The continuous growth model (CGM) described by Tedeschi et al.

(2004) was used as the basis for simulating average animal daily growth, accounting for animal, environment, management and nutritional factors.

The subsequent sections discuss the implementation of, and alterations to the CGM to simulate animal performance and describe adjustments to model output for growth and composition genetic potential each day to compute profitability. Table 4.1 lists the variables used in the model.

Model equations

Determining animal starting weight. The growth model depends on user-input of the current live body weight (LBW) of the animal for prediction of future weight. The LBW is converted to a shrunk weight (SBW) to reduce bias due to rumen content weight by:

$$\mathrm{SBW} = \mathrm{LBW} \times (1 - \mathrm{SHRINK}) \qquad [1]$$

where SHRINK is the percent of LBW that is attributable to the content of the rumen and gastro-intestinal tract, usually assumed to be 3-4%.

An estimate of body weight at a known body composition is also needed to compute an equivalent shrunk body weight (EqSBW) allowing energy requirements developed in research animals to be applied to the current animal. A shrunk body weight at 28% empty body fat (AFSBW), equivalent to a quality grade of low Choice, is used to calculate EqSBW by:

$$\mathrm{EqSBW} = \frac{\mathrm{SRW} \times \mathrm{SBW}}{\mathrm{AFSBW}} \qquad [2]$$

where SRW, the standard reference weight, is the shrunk body weight of animals used to develop the energy requirement equations and is assumed to be 478 kg (Tedeschi et al.,

Table 4.1. Description of variables used in the postweaning model.

Variable	Units	Description
adjBP	\$	Carcass base price adjusted for carcass characteristics
adjEBF	%	Percent empty body fat adjusted for the backfat thickness EPD
adjExpLMA	cm^2	Longissimus muscle area expected at low Choice, adjusted for the longissimus muscle area EPD
adjFAT	kg	Empty body fat adjusted for the adjusted percent empty body fat
adjFT	cm	Backfat thickness adjusted for backfat thickness EPD on a specified day
adjLMA	cm^2	Longissimus muscle area adjusted for longissimus muscle area EPD on a specified day
adjMS		Marbling score adjusted for marbling score EPD on a specified day
adjSBW	kg	Shrunk weight adjusted for live body weight EPD on a specified day
AFSBW	kg	Shrunk body weight at 28% empty body fat
AIC	\$	Animal interest cost
BCS		Body condition score
BP	\$	Carcass base price
CWP	%	Dressing percent
CWT	kg	Carcass weight
CWT_L	kg	Lower cutoff for normal carcass weight range
CWT_u	kg	Upper cutoff for normal carcass weight range
dDMI	kg	Daily dry matter intake
DEV_{CH}	\$	Deviation from base price for Choice quality grade
DEV_{CWT}	\$	Deviation from base price for carcasses outside a specified carcass weight range
DEV_{PR}	\$	Deviation from base price for Prime quality grade
DEV_{QG}	\$	Deviation from base price for quality grade
DEV_{SE}	\$	Deviation from base price for Select quality grade
$DEV_{underSE}$	\$	Deviation form base price for any quality grade under Select
$DEV_{upperCH}$	\$	Deviation from base price for upper 2/3 of the Choice quality grade
DEV_{YG}	\$	Deviation from base price for yield grade
DEV_{yg1}	\$	Deviation from base price for yield grade 1
DEV_{yg2}	\$	Deviation from base price for yield grade 2
DEV_{yg3}	\$	Deviation from base price for yield grade 3
DEV_{yg4}	\$	Deviation from base price for yield grade 4
DEV_{yg5}	\$	Deviation from base price for yield grade 5
DFC	\$	Daily feed cost
DFI	kg	Daily feed intake
DM	%	Percent dry matter of feed
DMI	kg	Dry matter intake
DOF	d	Days on feed

(Continued)

Table 4.1. *Continued.* Description of variables used in the postweaning model.

Variable	Units	Description
EBF	%	Percent empty body fat
EqSBW	kg	Equivalent shrunk body weight
expCWP	%	Expected carcass dressing percent at low Choice
expDMI	kg	Expected total dry matter intake at low Choice
expDOF	d	Expected number of days on feed to reach low Choice
expFT	cm	Expected backfat thickness at low Choice
expLMA	cm^2	Expected longissimus muscle area at low Choice
expOutWT	kg	Expected live weight at low Choice
FAT	kg	Empty body fat
FC	$	Fixed costs
FCkg	$	Cost of feed (per kg)
FCP	$	Feeder calf price (per kg)
FCV	$	Feeder calf value
FIC	$	Feed interest cost
FT	cm	Carcass backfat thickness
FT_{EPD}	cm	Random regression backfat thickness EPD for a specified day
FTr		Accuracy of FT EPD
HCC		Hair coat code (1 = dry and clean; 2 = some mud on lower body; 3 = wet and matted; 4 = covered with wet snow or mud)
HC		Hide depth code (1 = thin; 2 = average; 3 = thick)
HD	cm	Hair depth
HRS	hr	Number of hours of sunlight
IMP		Implant decision
inWT	kg	Live weight at the beginning of the feeding period
$inWT_f$	kg	Average weight of female calves entering feedlot
$inWTs_f$	kg	Standard deviation of average weight of females entering feedlot
$inWT_m$	kg	Average weight of male calves entering feedlot
$inWTs_m$	kg	Standard deviation of average weight of male calves entering feedlot
ION		Use of inonphores decision
LBW	kg	Live body weight
LMA	cm^2	Carcass longissimus muscle area
LMA_{EPD}	cm^2	Longissimus muscle area EPD
LMAr		Accuracy of LMA EPD
MD	cm	Mud depth in the feedlot
ME	Mcal/kg	Metabolizable energy of diet
MS		Carcass marbling score
MS_{EPD}		Random regression marbling score EPD for a specified day
MSr		Accuracy of MS EPD
NE_g	Mcal/kg	Net energy of diet for growth
NE_m	Mcal/kg	Net energy of diet for maintenance

(Continued)

Table 4.1. *Continued.* Description of variables used in the postweaning model.

Variable	Units	Description
OLA	\$	Operating loan amount
outWT	kg	Live weight at end of feeding period
PC		Possible change
PEH	m^3	Pen area allotted per head
QG		USDA Quality Grade
RATE	%	Annual interest percentage rate
R_f	%	Proportion of females kept as replacements
RH	%	Percent relative humidity
SBW	kg	Shrunk body weight
SHRINK	%	Factor to convert live weight to shrunk weight
SRW	kg	Standard reference weight
SWG	kg	Shrunk weight gain
tDMI	kg	Total cumulative dry matter intake
TC	\$	Total costs
T_c	°C	Average temperature
TFC	\$	Total fixed costs
WS	km/hr	Wind speed
WT_{EPD}	kg	Random regression weight EPD for a specific day
WTr		Accuracy of WT EPD
YC	\$	Daily yardage cost
YG		USDA Yield Grade

2004). The AFSBW selected will depend on the animal's genotype as different animal types will reach the same level of fatness at different body weights. The estimation of AFSBW is important so that the weight of the current animal is equivalent to the standard reference animal and predictions based on energy requirements are appropriate. The AFSBW may be estimated from literature sources where similar postweaning management strategies were employed or based on known mean animal performance in a commercial setting.

Predicting feed intake. A prediction of feed intake for the individual animal is needed for computing costs of production. The CGM model predicts daily dry matter intake (DMI) accounting for animal age and weight, the environment (average temperature, relative humidity, wind speed, hours of sunlight and depth of mud) and the dietary content of net energy for maintenance. Daily feed intake (DFI), in kilograms, is calculated from DMI by:

$$\mathrm{DFI} = \frac{\mathrm{DMI}}{\mathrm{DM}} \qquad [3]$$

where DM is the percent dry matter of the feed.

Predicting weight and weight gain. A prediction of daily weight gain is used to determine the new SBW for each day on feed and subsequent carcass weight (CWT) at harvest. The CGM predicts the amount of shrunk weight gain (SWG) accounting for animal, environmental and nutritional factors. The new SBW is calculated for average animals by:

$$\mathrm{SBW}_{\mathrm{new}} = \mathrm{SBW} + \mathrm{SWG} \qquad [4].$$

To account for the current animal's genetic potential for body weight at a particular day on feed, the SBW is adjusted by a sire or dam expected progeny difference (EPD) weight at the corresponding age. The weight EPD results from a random regression analysis that most generally includes an intercept (b_o) and a linear coefficient (b_l) component of the form:

$$\mathrm{WT}_{\mathrm{EPD}} = b_0 + b_1 d \qquad [5]$$

where $\mathrm{WT}_{\mathrm{EPD}}$ is the live body weight EPD for a specific day (d), that represents the day on feed or the age of the animal in days, depending on how the regression is calculated. Equation 5 assumes that the EPD is based on a linear function, which may not be realistic, but higher-order terms are easily included.

The SBW is adjusted by the sire and dam (if known) genetic potential for live weight on a particular day by:

$$\mathrm{adjSBW} = (\mathrm{LBW} + \mathrm{WT}_{\mathrm{EPD\text{-}S}} + \mathrm{WT}_{\mathrm{EPD\text{-}D}}) \times (1 - \mathrm{SHRINK}) \qquad [6]$$

where $\mathrm{WT}_{\mathrm{EPD\text{-}S}}$ and $\mathrm{WT}_{\mathrm{EPD\text{-}D}}$ are live weight EPD for the sire and dam, respectively. The adjusted live weight is multiplied by the percent shrink constant to yield the adjusted SBW. Eq.4 becomes:

$$\mathrm{SBW}_{\mathrm{new}} = \mathrm{adjSBW} + \mathrm{SWG} \qquad [7]$$

to yield the new SBW that can be used to calculate CWT. Carcass weight is a function of shrunk weight and dressing percent and is calculated by:

$$\mathrm{CWT} = \mathrm{CWP} \times \mathrm{adjSBW} \qquad [8]$$

where CWP is the carcass dressing percent calculated based on animal weight, adjusted for the expected shrunk body weight at a low Choice quality grade. The use of adjSBW in Eq. 8 adjusts CWT to account for differences in the weight of the current animal, compared to the average at a specified day on feed, due strictly to additive genetics.

Predicting carcass composition. A prediction of percent empty body fat (EBF) to estimate average carcass backfat thickness (FT) and marbling score (MS) is needed for the calculation of quality grade (QG) and yield grade (YG). The CGM calculates EBF from total carcass fat, which is predicted as a function of fat and weight in the daily gain. The EBF is calculated by:

$$\mathrm{EBF} = \frac{\mathrm{FAT}}{\mathrm{SBW} \times 0.891} \qquad [9]$$

where FAT is the amount of empty body fat (kg) and the denominator is empty body weight (kg), assumed to be 89.1% of SBW (CGM). To determine FT, a linear regression equation was developed ($r^2 = 0.97$) using mean 12th rib fat thickness measurements at different levels of EBF (n = 763) published by Guiroy et al. (2001). The equation to predict average FT was:

$$FT = -1.293 + (8.536 \times EBF) \qquad [10].$$

The data ranged from 25 to 32% EBF, so predictions using EBF estimates outside this range may be problematic.

Similar to LBW, the estimate for FT is adjusted to account for the animal's genetic potential for FT on a specific day. The FT EPD is calculated similar to Eq. 5, but containing a quadratic term, by:

$$FT_{EPD} = b_o + b_1 d + b_2 d^2 \qquad [11]$$

where FT_{EPD} is the backfat thickness EPD on a specified day (d). Several studies have shown that phenotypic FT changes as a function of days are not linear (Hamlin et al., 1995; Brethour, 2000; Nash et al., 2000) and thus the most general form is assumed to be quadratic. As with Eq. 5 higher order terms can be easily included. The FT prediction is adjusted by the sire and dam genetic potential by:

$$adjFT = FT + FT_{EPD-S} + FT_{EPD-D} \qquad [12]$$

where adjFT is the new FT prediction and FT_{EPD-S} and FT_{EPD-D} are the random regression FT EPD for the sire and dam, respectively.

Marbling score was also determined from EBF using a linear regression equation ($r^2 = 0.98$) developed from mean marbling score observations at different levels of EBF published by Perry and Fox (1997). The equation to predict average MS was:

$$MS = 0.944 + (17.120 \times EBF) \qquad [13]$$

The data ranged from 19 to 31% EBF, so predictions of MS using EBF estimates outside this range may be problematic.

The estimate of MS is adjusted to account for the animal's genetic potential for MS on a specific day. The MS EPD is then calculated by:

$$MS_{EPD} = b_o + b_1 d + b_2 d^2 \qquad [14]$$

where MS_{EPD} is the marbling score EPD on a specified day (d). Studies have shown that MS changes with respect to day are also not linear (Brethour, 2000; Nash et al., 2000). The most general function is assumed to be quadratic, but could be easily expanded. The MS prediction is adjusted by the sire and dam genetic potential with:

$$adjMS = MS + MS_{EPD-S} + MS_{EPD-D} \qquad [15]$$

where adjMS is the new MS prediction and MS_{EPD-S} and MS_{EPD-D} are the random regression MS EPD for the sire and dam, respectively.

The longissimus muscle area (LMA) was predicted to determine carcass yield (in conjunction with FT). LMA was calculated using a quadratic function based on body weight described by Hamlin et al. (1995). The equation to predict average LMA was:

$$LMA = -12.05 + 0.30 \times SBW - 0.00022 \times SBW^2 \qquad [16].$$

LMA was then adjusted for an animal's genetic potential using the routinely reported LMA EPD (LMA_{EPD}). To account for genetic differences each day on feed, however, the expected average LMA at a low Choice finish (expLMA) was calculated by:

$$expLMA = -12.05 + (0.30 \times AFSBW) - (0.00022 \times AFSBW^2) \qquad [17]$$

and adjusted by the LMA_{EPD}:

$$adjExpLMA = expLMA + LMA_{EPD-S} + LMA_{EPD-D} \qquad [18]$$

where adjExpLMA is the expected LMA at finish accounting for sire and dam EPD for LMA. Equation 18 assumes that the standard age used to adjust the LMA_{EPD} and the age at harvest are similar. The adjusted LMA for a specific day is then calculated using a quadratic function based on days on feed adapted from Hamlin et al. (1995):

$$\mathrm{adjLMA} = \mathrm{adjExpLMA} + (0.47 \times \mathrm{d}) - (0.000040 \times \mathrm{d}^2) \qquad [19]$$

where adjLMA is the LMA adjusted for sire and dam EPD and the day on feed and d is the expected number of days on feed to reach low Choice subtracted from the current number of days on feed. Each of the equations to calculate trait EPD (Eqs. 5, 11, 14, 18) assume no correlation between traits.

Yield grade was determined from the adjusted FT and LMA predictions for the calculation of revenue. Yield grade was calculated using the USDA equation (AMS, 2006):

$$\mathrm{YG} = 2.5 + (2.5 \times (\mathrm{adjFT}/2.54)) + (0.2 \times \mathrm{KPH}) + (0.0038 \times ((\mathrm{CWP} \times \mathrm{adjSBW}/100) \times 2.2)) - (0.32 \times (\mathrm{adjREA}/6.45)) \qquad [20]$$

where KPH is the percent kidney, pelvic and heart fat and was assumed to be 3%.

Quality grade was determined from adjMS (where a small MS = 5.0). Marbling score indicates QG and thus a MS observation translates directly to a QG observation. This model assumes all animals are in the A-maturity category and are free from defects that may reduce the QG assessed.

Calculating production costs. Daily production costs are a function of animal value and feedlot maintenance and are divided into fixed and variable components. The fixed component is comprised of two parts: fixed costs uniquely associated with each animal and fixed costs that are assumed static throughout the herd. The value of the steer or heifer at the beginning of the feeding period is calculated based on current market price by:

$$\mathrm{FCV} = \mathrm{inWT} \times \mathrm{FCP} \qquad [21]$$

where FCV is feeder calf value, inWT is the average weight of the steer or heifer at the start of feeding and FCP is the price of a feeder steer or heifer (per kg) of a given weight range that includes the average weight. In a retained ownership situation, FCV represents a fixed opportunity cost equal to the amount of revenue that could have been returned to the producer if the calf had been sold, rather then kept and fed.

The static fixed costs (FC) are combined into one value and include any per animal costs to the producer that are equivalent across animals. These may include transport of animals to the feedlot and to the harvest facility, animal processing, medical treatment, death loss, insurance, taxes, etc. The total fixed costs (TFC) therefore for an individual animal are:

$$\mathrm{TFC} = \mathrm{FCV} + \mathrm{FC} \qquad [22]$$

where TFC is calculated one time, regardless of the number of days on feed.

The variable costs incurred by an individual animal are dependent mainly on animal performance and length of time on feed. These daily costs include: feed, yardage, interest on the animal and interest on the feed and yardage. Feed costs are determined by DFI (Eq. 3) and the cost of feed by:

$$\mathrm{DFC} = \mathrm{FCkg} \times \mathrm{DFI} \qquad [23]$$

where DFC is the daily cost of feed consumed by the individual animal and FCkg is the cost of feed per kilogram on an as-fed basis. Yardage costs (YC) are a daily per animal charge for maintaining cattle in the feedlot.

The daily animal interest cost, for a retained ownership situation, represents the cost of not having sold the steer or heifer before feeding, and thus is an opportunity cost for the animal. The daily animal interest cost (AIC) is calculated by:

$$\mathrm{AIC} = \frac{\mathrm{RATE}}{365} \times \mathrm{FCV} \qquad [24]$$

where RATE is the annual percentage rate. The feed interest cost is similar to the animal interest, but assumes an operating loan has been procured to purchase feed and pay yardage costs. Feed interest costs (FIC) accrue for each day cattle are on feed and are calculated by:

$$\mathrm{FIC} = \frac{\mathrm{RATE}}{365} \times \mathrm{OLA} \qquad [25]$$

where OLA is the expected per head amount of the operating loan that is needed to cover feeding expenses for the whole feeding period. The OLA can be estimated from:

$$\mathrm{OLA} = (\mathrm{DFC} + \mathrm{YC}) \times \mathrm{DOF} \qquad [26]$$

where the components of DFC and the DOF can be predicted based on previous performance.

Costs are calculated on a daily basis for each individual animal and accumulate from the beginning of the feeding period to harvest. The total costs (TC) at harvest used to calculate profit are:

$$\mathrm{TC} = ((\mathrm{DFC} + \mathrm{YC} + \mathrm{DIC} + \mathrm{FIC}) \times \mathrm{DOF}) + \mathrm{TFC} \qquad [27]$$

where DOF is the number of days from initiation of feeding to the harvest endpoint.

Calculating revenue. Total revenue is determined by the weight, quality and yield characteristics of an individual carcass. Each carcass is assigned a base price (BP) per 45.45 kg of CWT, dependent on current market prices for dressed cattle, the percent of carcasses expected to grade Choice and the Choice/Select price spread. The BP is adjusted for carcass performance based on the selected grid. The adjusted BP is calculated by:

$$\text{adjBP} = \text{BP} + \text{DEV}_{\text{CWT}} + \text{DEV}_{\text{QG}} + \text{DEV}_{\text{YG}} \qquad [28]$$

where adjBP is the BP adjusted for carcass characteristics, DEV_{CWT} is the deviation assigned to carcass weights that are below or above a specified normal weight range ($\text{DEV}_{\text{CWT}} = 0$ for carcass weights in the normal range), DEV_{QG} is the deviation for QG and DEV_{YG} is the deviation for YG. Each deviation is a user-input value that includes a premium or discount (positive or negative deviation) for each quality and yield grade level. Revenue is calculated from the adjBP by:

$$\text{Revenue} = \left(\frac{\text{CWT}}{45.45}\right) \times \text{adjBP} \qquad [29].$$

Model design

The model is composed of ten classes that describe attributes of system components and contain methods to perform various functions. Each class exists independently of the other classes but interaction between classes occurs within the larger scope of the model. In the DSS, objects of the classes are created to pass needed parameter values and perform necessary calculations. Table 4.2 briefly describes the function of each class which are discussed in more detail.

Carcass grid. The CarcassGrid class stores information pertaining to the grid used to determine the value of a carcass at slaughter. A number of variables are required for each object of this class, each being a dollar value relating to quantitative carcass measurements, with the exception of measures identifying the acceptable carcass weight range. The grid is based on a BP and several values for deviations from the BP are required. The first is an out weight deviation (DEV_{CWT}) for carcasses that are below

(CWT_L) or above (CWT_u) a designated CWT range. This deviation is usually in the form of a carcass discount per 45.45 kg of CWT.

The deviation from the BP for QG in the CarcassGrid class is assigned in five categories: Prime (DEV_{PR}), upper 2/3 Choice ($DEV_{upperCH}$), Choice (DEV_{CH}), Select (DEV_{SE}) and any QG under Select ($DEV_{underSE}$). Similarly, deviations for YG also fall into five categories: YG 1 (DEV_{yg1}), YG 2 (DEV_{yg2}), YG 3 (DEV_{yg3}), YG 4 (DEV_{yg4})

Table 4.2. Description of classes in the postweaning model.

Class	Description
CarcassGrid	Describes the specified grid used to determine the value of a carcass
Economics	Describes variable and fixed daily costs associated with postweaning production and has methods to calculate total costs and revenue
Environment	Describes factors external to the animal that may affect cattle growth and composition (e.g. temperature, relative humidity, etc.)
FeedRation	Describes the nutritive content of the feed ration used in the growing or finishing phase of production
Growth	Predicts the weight and composition change in a steer or heifer for one day on feed, given animal, sire and dam, nutrition, environment and management effects
Management	Describes feedlot characteristics and management decisions that may impact growth of cattle
Parent	Describes the genetic potential of a sire or dam for random regression growth and carcass traits, prediction accuracy and the variation associated with each trait, and has methods to stochastically generate EPD accounting for possible change
PossibleChange	Methods to determine the possible change for EPD
SamplingAdjustment	Describes variables randomly sampled from a normal distribution with specified mean and variance used to simulate progeny phenotype to account for genetic and residual effects
TerminalAnimal	Describes a weaned heifer or steer placed in the feedlot in terms of growth and carcass traits

and YG 5 (DEV_{yg5}). The CarcassGrid class returns each of these fourteen parameters for use in calculating revenue.

Economics. The Economics class describes costs associated with feeding cattle to harvest and has methods to calculate costs and revenue based on animal performance. Each Economics object includes the previously defined variables: FCkg, FCP, OLA, YC, RATE and FC. The Economics class has a method to update the daily variable costs and methods to return the cumulative total costs for all days on feed and the total revenue based on carcass composition and the specified grid.

Environment. The Environment class describes factors external to the animal that affect growth and changes in composition while animals are on feed. The variables required for each Environment object consist of weather-related measurements. The variables include: current average Centigrade temperature (T_c), average percent relative humidity (RH), average wind speed (km/h; WS), average number of hours of sunlight (HRS) and depth of mud in the feedlot (cm; MD). The Environment class returns each variable for use in the Growth class for prediction of dry matter intake (DMI).

Feed ration. The feed ration class describes the energy content of the feed during postweaning growth. Variables required for each FeedRation object include: the dietary content of metabolizable enery (ME), the dietary content of net energy for maintenance (NE_m) and for growth (NE_g) and the percent dry matter of the feed (DM). The FeedRation class returns each parameter for use in the Growth class to determine intake and gain per day.

Management. The Management class contains feedlot factors and management decisions that are accounted for in the model. Each Management object includes: implant decision (IMP), the area (m^3) allotted to each animal in the feedlot (PEH) and the use of ionophores (ION). IMP may be used to adjust the final expected weight at a given level of fat, but is not currently implemented. PEH and ION are used as adjustment factors for daily growth. The Management class returns each variable for use in other classes.

Parent. The Parent class describes the sire or dam of an animal, or pen of animals, in the feedlot in terms of genetic potential for growth and composition. The genetic potentials are used to adjust performance for weight, backfat thickness, marbling score and longissimus muscle area each day on feed. Random regression components for each of the first three EPD are required for each Parent object with the number dependent on the order of the regression function. EPD accuracies for each (WTr, FTr, MSr, LMAr) are also required. In the case of the random regression EPD, each component is returned for use in calculating an EPD for a specified number of days on feed, or age (Eq. 5, 11, 14).

The Parent class also has methods to stochastically determine random regression component EPD for WT, FT and MS from a distribution of possible true values around the animal's EPD based on possible change (PC) to account for the error associated with predicting genetic merit. A new EPD is generated from the original EPD by:

$$\mathrm{EPD}_{\mathrm{new}} = \mathrm{EPD}_{\mathrm{original}} + (\mathrm{RandG} * \mathrm{PC}) \quad [30]$$

where RandG is a pseudo-random number from a Gaussian distribution used to randomly deviate PC and PC is a standard error obtained from the PossibleChange class. The PC is calculated in that class by:

$$\mathrm{PC} = t * (1 - \mathrm{acc}) * \frac{\sqrt{\sigma_a^2}}{2} \quad [31]$$

where t is the table value for the desired significance level, acc is the BIF accuracy of the EPD and σ_a^2 is the additive genetic variance of the trait. New and unique random regression component EPD can be generated each time an object of the Parent class is created.

Terminal animal. The TerminalAnimal class describes characteristics of an animal for a single day postweaning. The class has methods to set and return weight and carcass composition traits and observations concerning age and days on feed. Each TerminalAnimal object must include inputs: average weight of male calves entering feedlot ($inWT_m$), standard deviation of $inWT_m$ ($inWTs_m$), average weight of female calves entering feedlot ($inWT_f$), standard deviation of $inWT_f$ ($inWTs_f$), and the proportion of females kept as replacements (R_f) from the herd. The sex of the animal being simulated can be generated for each object of the TerminalAnimal class.

Sex is stochastically generated assuming an equal proportion of males and females available, adjusting for R_f. The probability of a selected animal being male is calculated as:

$$\text{pMale} = 0.5 \times (1 + R_f) \qquad [32]$$

and the sex is then determined by a function including:

$$\begin{aligned} &\text{if Rand} < \text{pMale; sex} = \text{male} \\ &\text{if Rand} >= \text{pMale; sex} = \text{female} \end{aligned} \qquad [33]$$

where Rand is a pseudo-randomly generated number with a value from 0.0 to 1.0. When no female replacements are kept the probability of an animal being male is 0.5.

The TerminalAnimal class includes methods to increment the animal's age in days (from the age at start of feeding) and number of days on feed. Methods are available to set or update weight and carcass characteristics including: LBW, AFSBW, CWT, FT, MS, LMA, QG and YG. Other animal characteristics may also be set, including: hair coat code (HCC), hair depth (HD), hide code (HC) and body condition score (BCS). These variables are also returned by the TerminalAnimal class for use in growth and economic calculations.

Growth. The Growth class describes the one-day growth and composition of animals in the postweaning phase. Each Growth object requires a TerminalAnimal object for use in setting the initial SBW and FAT. The Growth class has a method to assign weight and composition changes to the current animal requiring a number of objects as parameters including: TerminalAnimal, FeedRation, Environment, Management, Parent (sire), Parent (dam) and expected average animal composition at finish. The variables needed for growth and composition calculations are accessed through these objects. The Growth class also has a method to set the amount of shrink to adjust LBW and a method to return the DMI for use in cost calculations.

Model implementation in the DSS

The model classes described previously in this chapter were implemented in the DSS to predict postweaning performance and profitability of progeny from alternative sires. The DSS was developed for flexibility providing a deterministic option for simulating profitability of average progeny fed to a user-defined carcass endpoint, or a stochastic option that predicts ranges of optimum endpoints and profit for simulated progeny performance to incorporate a measure of risk in decision-making. Each option simulates animal composition and profit on a daily basis using user-supplied inputs.

Before profitability is simulated with either option, the DSS ensures that simulated performance matches that expected by the user, using a method to correct model parameters. The following sections discuss this method and each option in more detail.

Correction of simulated performance. The DSS includes a method (runBase) to compare the simulated performance of an average animal to user-expected performance, iterating until differences are negligible. This method is designed to model the current performance of the user's cattle in the feedlot by ensuring that the growth curve simulated by the model for the average animal matches the growth curve of average animals in the user's herd, without requiring the user to input breed, environmental and management parameters that may be difficult to estimate, allowing for the ready comparison of animals of different biological types and in different environments. The runBase method provides a foundation for running the other model options ensuring that differences between animals are due to differences in genetic potential (and residual effects) and are not due to differences in the procedures and assumptions in the simulation of average growth and carcass composition.

The runBase method accepts user inputs of inWT, feed energy (ME, NEg and NEm) and expected days on feed (expDOF), based on past performance, to simulate the average LBW at the end of feeding (outWT), carcass composition and total DMI (tDMI) for expDOF. The simulated performance is then compared with the average weight expected at finish (expOutWT), expected carcass backfat thickness (expFT), expected longissimus muscle area (expLMA), expected carcass dressing percent (expCWP) and total expected DMI (expDMI) at a low Choice quality grade endpoint. Differences are assumed to be due to differences in energy requirements simulated by the model and those actually observed in a specific production circumstance.

Differences in LBW at finish (outWT versus expOutWT) are attributed to differences in maintenance requirements. A value for outWT that exceeds expOutWT indicates that the simulated animal reaches the expected finish weight in fewer days than would normally be observed. The energy required for maintenance (MR) is too low, allowing too much energy to be available for growth. The requirements for maintenance are then numerically increased. Conversely, if the expOutWT exceeds the value for outWT then the simulated maintenance requirements were too high, not allowing enough energy for growth, and are reduced.

The amount of FT at finish was used to represent EBF for comparison of carcass composition given the linear relationship in the model. Differences in FT (FT versus expFT) were attributed to differences in retained energy (RE) as the CGM assumes "...the most important variable determining body and gain composition is the RE per unit of gain..." (Tedeschi et al., 2004). The RE directly influences the amount of body fat in the gain, and thus the current amount of empty body fat (kg) used to calculate EBF. A value for FT that exceeds the expFT in the expected number of days indicates that too much energy was retained by the simulation and left over for the deposition of fat. The RE is then reduced. An expFT that exceeds FT indicates that not enough energy was retained to deposit a sufficient amount of fat, and RE is increased.

Simulated LMA and CWP are compared to expected values due to their impact on carcass composition and in the calculation of revenue at harvest. The function used to predict LMA (Eq. 16) was originally developed based on specific biological types and is thus adjusted based on the current biological type implied by the expected LMA at low Choice. Similarly, CWP is adjusted in the simulated animal to match expCWP assuring the average animal CWT is as expected.

Total DMI is compared because daily DMI will impact weight and composition changes and an accurate representation of tDMI will allow for accurate feed cost calculations. Differences in DMI at finish (tDMI versus expDMI) were accounted for by adjusting simulated daily DMI (dDMI) in the appropriate direction. In cases where tDMI exceeds expDMI, the simulated animal is consuming more than expected and dDMI is reduced. A value of expDMI that is greater than tDMI indicates that simulated dDMI should be increased.

The runBase method simultaneously adjusts MR, RE, LMA, CWP and dDMI in the correct direction by the percentage difference between the simulated and expected values (except RE which is adjusted by half the distance due to the large difference between values). The parameters are adjusted until all are within a significant number of decimal places of the expected value ($>10^8$). In cases where the input parameters can reasonably be expected to yield the expected performance, convergence of simulated and expected outputs occurs in a short time (< 2 seconds) and in relatively few rounds (< 20). The adjustments are stored in the input parameter file for use in future deterministic or stochastic simulations. To obtain the correct growth curve the runBase method must be run for each production circumstance to obtain reliable results from future simulations.

User-defined endpoint option. The deterministic option of the DSS is designed to allow the user to compare the profitability of using alternative sires at user-defined finish endpoints (weight, backfat thickness or marbling score). This option does not account for the risks associated with animals that have low levels of accuracy for EPD nor the variation in performance due to Mendelian sampling or residual effects, but it gives the user more control over the outcome and may prove to be the most suitable for web-based implementation. Figure 4.1 is a flow diagram of model classes and methods to simulate profit, days on feed and carcass composition at a constant endpoint.

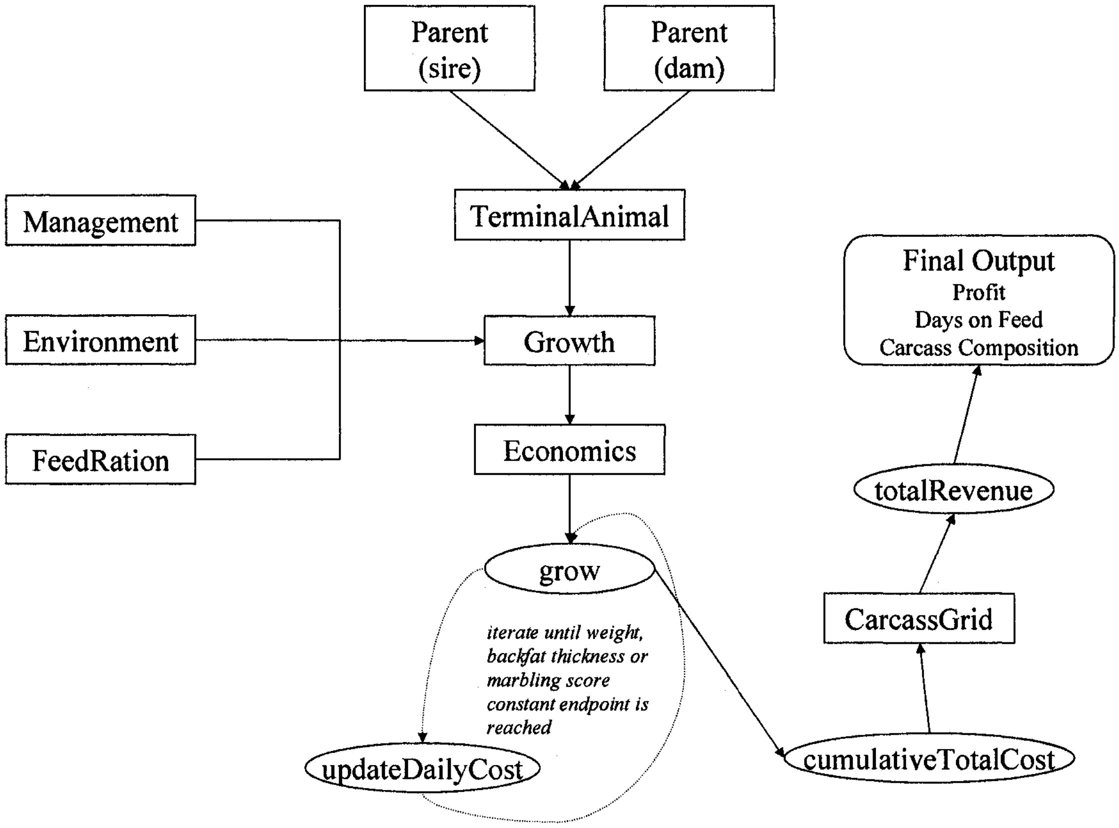

Figure 4.1. A flow diagram of model class and method use in the deterministic option of the postweaning decision support system. Class objects are denoted by ▭ and methods by ⬭.

The deterministic model simulates the performance of a single animal, adjusted by sire and dam EPD for weight, backfat thickness, marbling score and longissimus muscle area each day on feed. Attributes of the TerminalAnimal, Management, Environment and FeedRation classes are used by the *grow* method to simulate a single day of animal growth and composition. Growth begins from the average weight of male calves at the beginning of the feeding period. The *grow* method is iterated, with costs updated each round, until the value of the selected endpoint is reached. The number of days on feed are also updated each round. Following the iterations, total costs and revenue are computed to yield profit. The final output represents the average expected performance of a particular sire and dam's progeny under specific feeding conditions which can be used to evaluate differences in alternative sires.

Optimum endpoint option. The stochastic option of the DSS is designed to simulate growth to the endpoint that maximizes profitability and produces summary information on profit, days on feed and carcass composition at that endpoint. This option can account for expected changes in genetic potential for weight and composition as a function of sire and dam EPD accuracy and the variation in performance that exists due to Mendelian sampling and residual effects. Outputs can be in the form of distributions that provide the user with a measure of variability in the predicted outcome, a measure of the risk associated with alternative selection decisions. Figure 4.2 is a flow diagram of the model classes and methods used to simulate growth and composition.

The simulation begins with the creation of a sire and dam with random regression EPD for weight, backfat thickness and marbling score and EPD for longissimus muscle

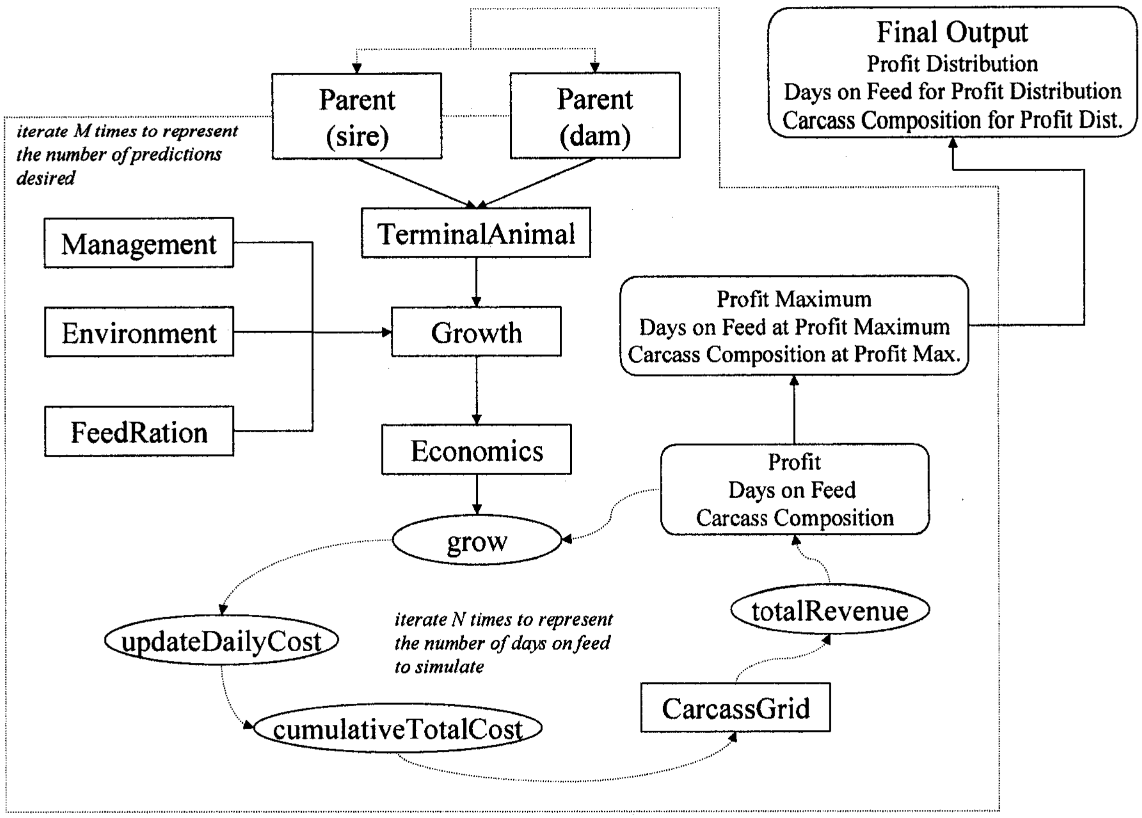

Figure 4.2. A flow diagram of model class and method use in the stochastic option of the postweaning decision support system. Class objects are denoted by ▭ and methods by ⬭.

area. Accuracy values for each EPD may be used to pseudo-randomly select EPD variables from a distribution of possible values. The sex of the animal on feed may also be stochastically determined based on the probability of sex. Attributes of the TerminalAnimal, Management, Environment and FeedRation classes are used by the *grow* method to simulate a single day of animal growth and composition for 1 to *n* animals per pen.

The phenotype of an individual animal (i) for a growth or composition trait each day on feed may be specified by:

$$y_i = \mu + \frac{1}{2}u_s + \frac{1}{2}u_d + \theta_s + \theta_d + e_i \qquad [34]$$

where μ is the mean performance of the specified population, u_s and u_d are the breeding values of the sire and dam of animal (i), θ_s and θ_d are the Mendelian sampling terms for the sire and dam of animal (i) and e_i is the residual term for animal (i).

The variance of progeny breeding values is defined as:

$$\mathrm{var}[u] = \frac{1}{4}\sigma^2_{g_s} + \frac{1}{4}\sigma^2_{g_d} + \frac{1}{2}\sigma^2_{g_{sd}} \qquad [35]$$

where σ^2_g is the additive genetic variance. One-quarter of the additive variance is accounted for by one-half of the sire's estimated breeding value ($\hat{u}_s$). Often the dam's genetic contribution is unknown and so the remaining genetic value (a_i) for animal (i) is assumed to be from a Normal distribution where:

$$a_i \sim N(0, \frac{3}{4}\sigma^2_g) \qquad [36]$$

and, thus a_i is:

$$a_i = \mathrm{RND} \times \sqrt{\frac{3}{4}\sigma^2_g} \qquad [37]$$

where RND is a random normal deviate. The residual term is also assumed to be from a Normal distribution:

$$e_i \sim N(0, \sigma^2_e) \qquad [38]$$

where σ^2_e is the residual variance, and thus e_i is:

$$e_i = \mathrm{RND} \times \sqrt{\sigma^2_e} \qquad [39].$$

The phenotype for animal (i) is then simulated as:

$$y_i = \mu + \frac{1}{2}\hat{u}_s + a_i + e_i \qquad [40].$$

The *grow* method is iterated N times for each animal in the pen, where N is the number of days on feed that is thought to be large enough that the global optimum profit can be attained. The value of N should then be larger than the expected number of days to reach traditional finish endpoints. During each iteration of the *grow* method costs are updated, total cumulative costs and revenue calculated, and profit determined based on weight and carcass composition for that number of days on feed. The number of days on feed are also updated each round. Profit is stored at each round and upon completion of the iterations maximum profit is determined. Days on feed and carcass composition at maximum profit are also stored. The entire process is repeated and iterated M times with new values for sire and dam random regression EPD components and progeny sex and starting weight at each round. Following M iterations, N values of maximum profit are stored, including days on feed and carcass composition at each profit maximum. A distribution of profits is then obtained with the associated carcass composition.

Validation of postweaning DSS

A validation of the growth and composition portion of the model was performed to ensure that differences in genetic potential for weight, backfat thickness, marbling score and longissimus muscle area each day on feed were appropriately represented in simulation output of weight and carcass composition at harvest. Empirical validation, or the comparison of model predictions with observations from the real-world, is an important step in model development, especially where model predictions will be used instead of actual measurements on the real system (Mitchell, 1997). The following describes the validation data and procedures that were used to simulate a real production system and validate model results.

Data. The dataset used for model validation included eleven Charolais sire groups and is summarized in Table 4.3. Monthly weight observations were available for 130 Charolais-cross calves from weaning to harvest (D.H. Crews, personal communication). Daily intake was recorded starting 177 days prior to harvest and constitutes the feeding period for simulation with inWT representing the weight at the beginning of this period. Not all animals had intake observations for each of the 177 days, so the average intake over this period for an individual was used to estimate the intake for days where no observation was recorded. Dry matter intake for the entire feeding period was then determined by summation of daily intake, accounting for an assumed percent dry matter of the feed of 90%.

Table 4.3. Mean progeny performance by sire for the model validation data[a].

Sire	No. Prog.	inWT, kg	Age[b], d	DMI[c], kg	LBW[d], kg	HCW, kg	CWP, %	FT, cm	MS	LMA, cm^2	YG
1	14	329.29	272.86	1940.58	609.93	349.51	59.71	0.85	4.71	95.07	2.14
2	9	302.78	261.11	1470.17	519.22	302.42	60.66	0.70	4.90	86.44	2.03
3	13	361.54	307.00	1771.32	590.69	346.75	61.15	1.03	5.54	92.38	2.43
4	14	344.71	284.86	1673.91	590.64	342.37	60.37	0.86	4.79	95.14	2.09
5	15	343.47	301.40	1696.76	583.67	329.58	58.82	1.06	4.53	85.93	2.63
6	10	356.00	287.80	1662.40	604.10	356.91	61.63	0.81	5.80	96.70	2.09
7	9	304.33	282.44	1722.02	530.44	309.49	60.70	0.60	5.67	87.44	1.94
8	15	335.80	272.67	1801.96	625.53	367.79	61.27	0.70	4.80	104.27	1.69
9	11	364.82	289.91	1793.81	649.82	368.18	59.00	1.10	5.91	90.64	2.76
10	11	352.73	296.82	1701.51	621.91	355.08	59.42	0.99	5.36	92.45	2.46
11	9	363.22	260.56	1797.72	613.44	350.71	59.51	1.24	5.00	89.78	2.80

[a] inWT = weight at start of feeding period; Age = age in days at start of feeding period; DMI = total dry matter intake; LBW = live weight at slaughter; HCW = hot carcass weight; CWP = carcass dressing percent; FT = backfat thickness; MS = marbling score (small = 5.00); LMA = longissimus muscle area; YG = USDA yield grade

[b] Age at start of feeding period

[c] Days on feed = 177

[d] Average age at harvest was 450 days

All animals were harvested on the same day and weight and carcass composition recorded. Carcass dressing percent (CWP) was calculated from the live weight at harvest (LBW) and hot carcass weight (HCW) assuming the amount of shrink associated with LBW was 4%. USDA yield grade was also calculated from the available data using Eq. 20 and assuming kidney, pelvic and heart fat was 3%.

Variance components for model input. Variance components (VC) for growth and carcass traits were needed to parameterize the model. Stochastic simulations utilize the additive genetic and residual variances for random regression weight (WT), backfat thickness (FT) and marbling score (MS) traits, and longissimus muscle area (LMA), for simulation of animal phenotype. Variance components for random regression on WT are easily calculated from published estimates for weight traits, but equivalent information is not generally available for FT and MS. The traditionally-reported VC were used for FT and MS with modifications to the model that will be described in a subsequent section. Genetic parameters for growth and carcass traits in Charolais cattle were reported by Crews et al. (2004) and are listed in Table 4.4 as adapted for model input.

The additive genetic (G_0) and residual (R) (co)variance matrices calculated using a multivariate analysis were transformed into random regression (co)variance matrices using covariance functions described by Kirkpatrick et al. (1990):

$$G_{rr} = Z_i^{-1} G_0 Z_i^{-T} \qquad [41]$$

and

$$R_{rr} = Z_i^{-1} R Z_i^{-T} \qquad [42]$$

where G_{rr} and R_{rr} are *n x n* additive genetic and residual (co)variance matrices for the random regression components, respectively and Z_i is an *n x n* matrix of actual or

Table 4.4. Estimates of heritability (h^2), additive genetic variance (σ_g^2) and residual variance (σ_e^2) for growth and carcass traits for model input.

Trait[a]	h^2	σ_g^2	σ_e^2
BW	0.53	11.02	9.60
WW[b]	0.23	188.80	500.20
YW[c]	0.33	538.16	1163.69
FT	0.38	0.032	0.05
MS	0.34	0.11	0.23
LMA	0.43	35.43	47.40

[a] BW = birth weight; WW = weaning weight; YW = yearling weight; FT = backfat thickness; MS = marbling score; LMA = longissimus muscle area

[b] Includes the direct component only

[c] Parameters calculated from weaning weight and postweaning gain

standardized ages, such as:

$$Z_i = \begin{bmatrix} 1 & age_1 & age_1^2 \\ 1 & age_2 & age_2^2 \\ 1 & age_3 & age_3^2 \end{bmatrix}$$

where, in this case, G_0 and R are third order matrices. The G_0 and R matrices for the validation included birth weight (BW), weaning weight (WW) and yearling weight (YW) and were:

$$G_0 = \begin{array}{c} \begin{bmatrix} 11.02 & 15.05 & 28.61 \\ 15.05 & 188.80 & 264.28 \\ 28.61 & 264.28 & 538.16 \end{bmatrix} \begin{array}{l} \text{BW} \\ \text{WW} \\ \text{YW} \end{array} \end{array} \quad R = \begin{array}{c} \begin{bmatrix} 9.60 & 14.55 & 19.73 \\ 14.55 & 500.20 & 444.14 \\ 19.73 & 444.14 & 1163.69 \end{bmatrix} \begin{array}{l} \text{BW} \\ \text{WW} \\ \text{YW} \end{array} \end{array}$$

The Z_i matrix, in which ages were not standardized or scaled as the difference between ages was small enough that scaling was thought to be unnecessary, was:

$$Z_i = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 205 & 42025 \\ 1 & 365 & 133225 \end{bmatrix}$$

yielding additive genetic and residual (co)variance matrices for the random regression intercept (b_0) and linear (b_1) and quadratic (b_2) coefficients:

$$G_{rr} = \left[\begin{array}{ccc} 11.06 & -0.026 & 0.000020 \\ -0.026 & 0.0091 & -0.000018 \\ 0.0020 & -0.000018 & 0.00000006 \end{array}\right]\begin{array}{c} b_0 \\ b_1 \\ b_2 \end{array} \quad R_{rr} = \left[\begin{array}{ccc} 9.60 & -0.022 & 0.00013 \\ -0.022 & 0.041 & -0.00011 \\ 0.00013 & -0.00011 & 0.00000034 \end{array}\right]\begin{array}{c} b_0 \\ b_1 \\ b_2 \end{array}$$

The additive genetic and residual variances can be obtained for any age from G_{rr} and R_{rr} by substituting a new age in Z_i such as (to obtain variances at 300 d):

$$Z_i = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 205 & 42025 \\ 1 & 300 & 90000 \end{bmatrix}$$

by:

$$G_0 = Z_i G_{rr} Z_i^T \qquad [43]$$

and

$$R = Z_i R_{rr} Z_i^T \qquad [44]$$

where the new variance is represented by the 1 x 3 matrix G_{300}. The new variances can then be used in Eqs. 37 and 39 to simulate an animal phenotype for a particular day on feed.

Genetic predictions for model input. The goal of validation in this case is to evaluate the ability of the postweaning model to appropriately predict differences in progeny performance due to differences in genetic potential. To that end, EPD for random regression on WT, FT, MS and for LMA were calculated on a within-herd basis. The small numbers of progeny per sire make the within-herd EPD more likely to be representative of differences in performance in this dataset than the EPD produced by a national cattle evaluation that may or may not include the performance data used in this study. The EPD are listed in Table 4.5, with one sire's progeny (sire 1 from Table 4.3) set as having the mean performance and the basis for calculation of the EPD.

The EPD were calculated using an index for a single source of information described by Bourdon (1997) as:

$$I = bx \qquad [45]$$

where I is the prediction desired, EPD in this case, x is the mean performance deviated from the overall mean and b is the regression coefficient for the amount of information and, in the case of progeny records, is calculated by:

$$b = \frac{ph^2}{4 + (p-1)h^2} \qquad [46]$$

where p is the number of progeny records.

Table 4.5. Within-herd EPD for growth and carcass traits.

	Within-herd EPD[a]					
Sire	BW[b]	WW	YW	FT	MS	LMA
1	0.00	0.00	0.00	0.00	0.00	0.00
2	1.27	9.56	-27.01	-0.07	0.09	-4.49
3	-1.05	12.16	4.05	0.11	0.45	-1.64
4	1.00	14.08	0.08	0.01	0.05	0.04
5	0.05	2.33	-2.31	0.13	-0.10	-5.88
6	-1.36	18.26	6.20	-0.02	0.52	0.89
7	-3.18	0.74	-23.57	-0.12	0.44	-3.97
8	1.77	4.59	4.46	-0.09	0.05	5.92
9	0.36	19.86	24.91	0.13	0.61	-2.53
10	-0.59	13.08	13.10	0.08	0.33	-1.49
11	0.86	19.18	10.40	0.19	0.13	-2.75

[a] BW = birth weight; WW = 205-adjusted weaning weight; YW = yearling weight; FT = backfat thickness; MS = marbling score (small = 5.00); LMA = longissimus muscle area

[b] The dataset contained no BW observation; the BW EPD was obtained from the Canadian Charolais national cattle evaluation

The dataset did not contain a BW observation on progeny and the BW EPD from the Canadian Charolais national cattle evaluation was substituted. This substitution was not expected to have a large effect due to the high heritability of BW ($h^2 = 0.53$). Each progeny WW observation was adjusted to 205 days following the Beef Improvement Federation guidelines (BIF) (BIF, 2002), assuming a BW of 36.36 kg and no age of dam adjustment (age of dam was unknown). Yearling weight observations were determined using the weight recorded after all animals had attained at least 365 days of age adjusted to 365 days following the BIF guidelines for YW.

The three weight EPD were transformed into components for the random regression on WT using the relationship between random effects predicted from the multivariate model and those predicted by the random regression model. This relationship is defined as:

$$Z_m \hat{u}_m = Z_i \hat{u}_r \qquad [47]$$

where Z_m is the incidence matrix representing the random effects in a multivariate analysis, u_m is a vector of random effects for a multivariate analysis and $\hat{u}_r$ is a vector of solutions for the random regression analysis. In cases where the models are equivalent Eq. 47 holds true and can be rearranged to solve for the estimated random regression random effects (breeding values) as:

$$\hat{u}_r = Z_i^{-1} Z_m \hat{u}_m \qquad [48].$$

Using Eq. 48 the components of the random regression WT EPD for each sire were calculated as:

$$\begin{bmatrix}\hat{u}_{b_0}\\ \hat{u}_{b_1}\\ \hat{u}_{b_2}\end{bmatrix} = \begin{bmatrix}1.01 & -0.011 & 0.0035\\ -0.0077 & 0.011 & -0.0035\\ 0.000014 & -0.000031 & 0.000017\end{bmatrix} \times \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}BV_{BW}\\ BV_{WW}\\ BV_{YW}\end{bmatrix}$$

yielding estimated breeding values (and subsequently EPD) for the intercept ($\hat{u}_{b_0}$) and linear ($\hat{u}_{b_1}$) and quadratic ($\hat{u}_{b_2}$) coefficients for the random regression on WT. A quadratic, rather than linear, regression was used to ensure a proper fit since the addition of the quadratic coefficient does not excessively complicate the model. The EPD used for model simulation are listed in Table 4.6, where the EPD for weight on a specific day can be calculated by adding a quadratic term to Eq. 5.

Table 4.6. Within-herd random regression and traditional EPD for growth and carcass traits for model input.

	Within-herd EPD[a]					
Sire	WTb_0	WTb_1	WTb_2	FT	MS	LMA
1	0.00	0.00	0.00000	0.00	0.00	0.00
2	1.08	0.19	-0.00074	-0.07	0.09	-4.49
3	-1.18	0.13	-0.00032	0.11	0.45	-1.64
4	0.85	0.15	-0.00042	0.01	0.05	0.04
5	0.01	0.03	-0.00011	0.13	-0.10	-5.88
6	-1.56	0.19	-0.00047	-0.02	0.52	0.89
7	-3.30	0.12	-0.00047	-0.12	0.44	-3.97
8	1.75	0.02	-0.00004	-0.09	0.05	5.92
9	0.23	0.13	-0.00018	0.13	0.61	-2.53
10	-0.70	0.10	-0.00018	0.08	0.33	-1.49
11	0.69	0.17	-0.00040	0.19	0.13	-2.75

[a] WTb_0 = random regression intercept; WTb_1 = random regression linear coefficient; WTb_2 = random regression quadratic coefficient; FT = backfat thickness; MS = marbling score (small = 5.00); LMA = longissimus muscle area

Issues with predicting WT. The quadratic random regression function described above is usually designed to calculate a weight EPD within the age range of the data used to develop the regression. In this case, the average slaughter age of progeny was 450 days, while the upper age for the random regression was 365 days. It is more accurate to make predictions based on interpolation rather than extrapolation, but the model incorporated in the postweaning DSS was developed to use information that is routinely collected and available to most breed associations. A cubic term at 450 days could have been calculated using this dataset, but that is not yet the case in many commercial settings and thus was not included as an input variable. The average harvest age (450 d) was not much larger than yearling age and it was hoped that any errors associated with extrapolating from the regression function would be small as the animal is expected to have a static pattern of growth between yearling and harvest. In fact, the phenotypic variance observed for simulated sire weight (4100.02 kg^2) was only slightly larger than that observed for the data used for validation (3698.00 kg^2).

Issues with predicting FT and MS. The model is designed to use random regression FT and MS components to determine an EPD for FT and MS for a particular day on feed (Eqs. 11 and 14). These components are not yet available, and their availability is unknown at this time, so the model was altered to use the traditional FT and MS EPD routinely reported by cattle breed associations. The EPD are generally adjusted to some pre-determined age and in this case that age was assumed to be near the average age at harvest for this dataset.

Backfat thickness was modeled each day on feed accounting for the FT EPD using an exponential function adapted from Brethour (2000):

$$\mathrm{adjFT} = \mathrm{adjExpFT} \times e^{0.0117 \times \mathrm{day}} \qquad [49]$$

where adjExpFT is the phenotypic FT expected at harvest (low Choice) accounting for the FT EPD and genetic and residual sampling and *day* is the difference between the current day on feed and the expected number of days to harvest. The value of 0.0117 was developed specifically for Continental cross cattle but was not adjusted for this simulation (Brethour, 2000).

A number of functions are available to model marbling score over time (Brethour, 2000; Bruns et al., 2004; 2005), but none were identified that could sufficiently predict MS using the current dataset. Marbling score was therefore modeled in a fairly elementary manner by simply adjusting the daily MS by the MS EPD, in addition to the genetic and residual sampling terms. Increases in MS do not proceed in a linear manner (Brethour, 2000) and MS EPD are likely different at the beginning and end of the feeding period, but with typical harvest ages near the age used to adjust the MS EPD, differences between true and predicted MS should be no larger than would normally be expected. The inclusion of random regression MS EPD will no doubt improve the predictive ability of the model.

Model parameters. The model validation was performed to compare the prediction of growth and carcass performance accounting for genetic differences in growth and carcass traits and therefore the economic components of the model were not parameterized and the model was altered to simulate feeding for a specified number of days. Various animal, management, environmental and feed parameters are needed to allow the model to function, but because individual animal growth curves are forced to expected values, these parameters, in this validation case, serve only to reduce the number of rounds needed for the growth curve to converge. In situations where differences in management or environment are of interest, then the parameters become much more important. Additional model parameters are listed in Table 4.7.

Table 4.7. Parameters used for model validation.

Parameter[a]	Value
BCS	5
HC	2
HCC	2
HD	1
HRS, h	12.2
MD, cm	1.0
ME, Mcal·kg^{-1}	3.5
NE_g, Mcal·kg^{-1}	1.4
NE_m, Mcal·kg^{-1}	2.0
PEH, m^3	5.0
RH, %	40.0
T_c, °C	10.3
WS, km·h^{-1}	6.7

[a] BCS = body condition score; HC = hide code; HCC = hair coat code; HD = hair depth; HRS = hours sunlight; MD = mud depth; ME = metabolizable energy; NE_g = net energy for growth; NE_m = net energy for maintenance; PEH = pen area per head; RH = relative humidity; T_c = average temperature; WS = wind speed

Simulation procedures. The runBase method was used to calculate the adjustment factors needed to establish the growth curve for the average animal. The stochastic method (runStoch) of the DSS was then used to simulate 1000 progeny per sire, with the starting weight and expected performance equal to the progeny mean performance of sire 1 (Table 4.3). Variation due to the accuracy of sire EPD was not simulated. Mean progeny performance after 177 days on feed was determined for each sire for growth and carcass traits affected by sire selection including outWT, CWT, FT, MS, LMA, QG, and YG.

Statistical analysis. Analysis was performed on the mean progeny performance calculated from the observed data (*Obs*) and simulated output from the model (*Sim*) for each sire (n = 11). To evaluate model performance the *Obs* values (Y) were regressed on *Sim* values (X) as described by Mayer and Butler (1993) and the coefficient of determination (r^2) calculated to assess fit. The significance of the r^2 for each trait regression was calculated using a permutation test (n = 10,000 rounds) for the hypothesis that the r^2 was obtained merely by chance using a procedure described by Good (2000). The *Obs* values naturally fulfill the role of the dependent variable, while the *Sim* values, though generated stochastically, are means of a large number of rounds and have a small standard error, making them suitable for the role of independent variable. Several authors have questioned the application of this type of regression for model validation beyond initial stages of development (Harrison, 1990; Mitchell, 1997; Analla, 1998) so the regression of *Sim* minus *Obs* on *Obs* was also performed as it does not depend on the assumptions of the normal regression model that may be violated when validating a model (Mitchell, 1997). This regression is essentially a residual plot that was used to visually assess the presence of outliers, bias and homogeneity of variance. The regression can further be used to evaluate the adequacy of model outputs by establishing the confidence intervals for the residuals when the desired precision of measurements is known.

Mean bias for model variables was computed as the subtraction of the mean *Sim* from the mean *Obs*, divided by the mean *Sim* (Tedeschi et al., 2004). Each mean bias was compared to zero (indicating no bias) using a two-tailed t-test. Overall re-ranking between *Obs* and *Sim* means was evaluated for each sire. Sires were ranked (highest to lowest numerical value) for each trait and a mean ranking for all traits calculated for *Obs* and *Sim*. The difference between these two mean rankings was then computed for each sire to evaluate the magnitude of the re-ranking that occurred.

Results. Figures 4.3 through 4.8 show the *Obs* versus *Sim* trait means plotted with a line representing an intercept of zero and slope of one. Ideally, all points would fall on the line, indicating a perfect relationship between the model output and actual observations. Models can not be perfect however, and some lack of precision is expected (Mitchell, 1997). Assessment of the scatter of the points around the line provides a good subjective measure for the validity of model outputs for each trait.

Figures 4.3 and 4.4 depict outWT and CWT and show good agreement between *Obs* and *Sim* values over the range of values represented by that data, with no apparent bias at most weights. There may have been some over-prediction at lower weights, but it is difficult to evaluate with few data points, an issue that exists for all of the plots. Figure 4.5, showing FT, again shows acceptable agreement with some possible over-prediction. Figure 4.6 depicts a problem with the model prediction of MS. Marbling score appears to be over-predicted in most cases and seems to be only mildly related to observed values, especially in the mid-range of values. There is no particular shape to the scatter to suggest that the relationship is not linear and the small range of the data (4.5 to 5.9; high Select to low Choice) may indicate that, in practice, the problem is not too serious. Figure 4.7 depicts acceptable agreement between *Obs* and *Sim* LMA means with some possible over-prediction for the range of that data. Figure 4.8 shows a close relationship between *Obs* and *Sim* means.

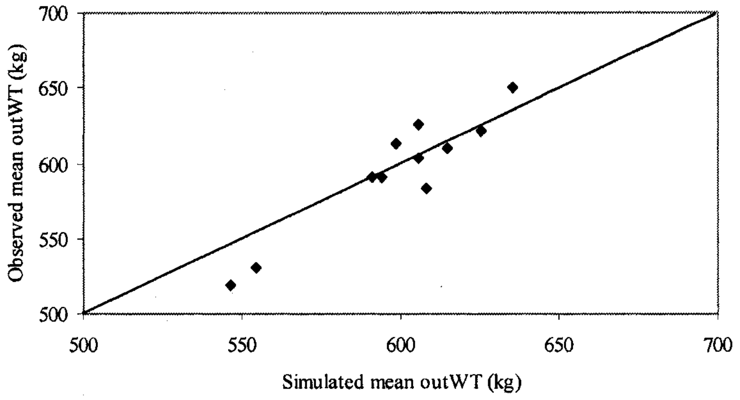

Figure 4.3. Observed versus simulated mean weight at harvest (outWT). Line represents y = x.

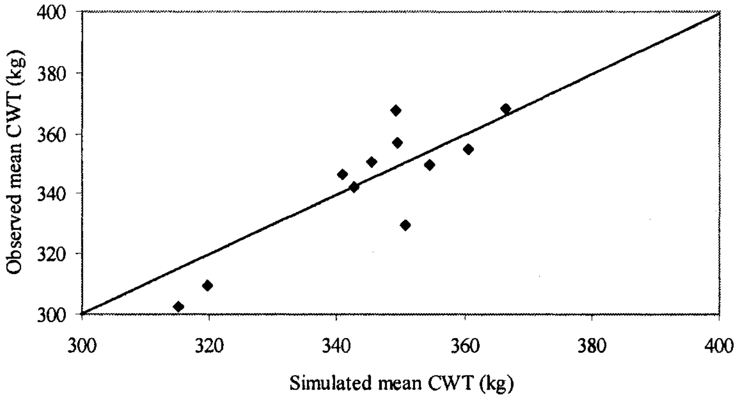

Figure 4.4. Observed versus simulated mean carcass weight (CWT). Line represents y = x.

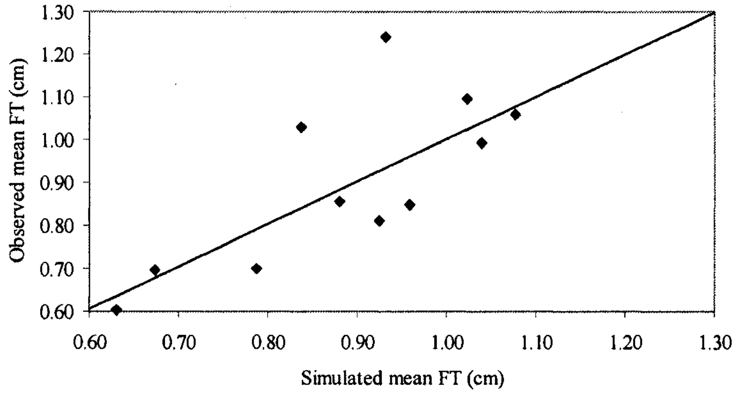

Figure 4.5. Observed versus simulated mean backfat thickness (FT). Line represents y = x.

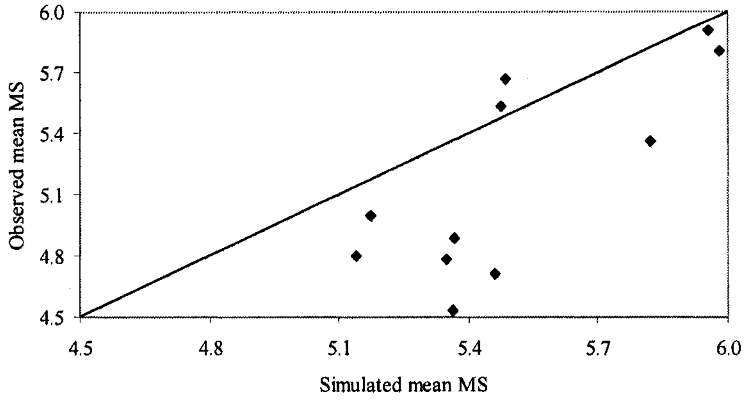

Figure 4.6. Observed versus simulated mean marbling score (MS). Line represent y = x.

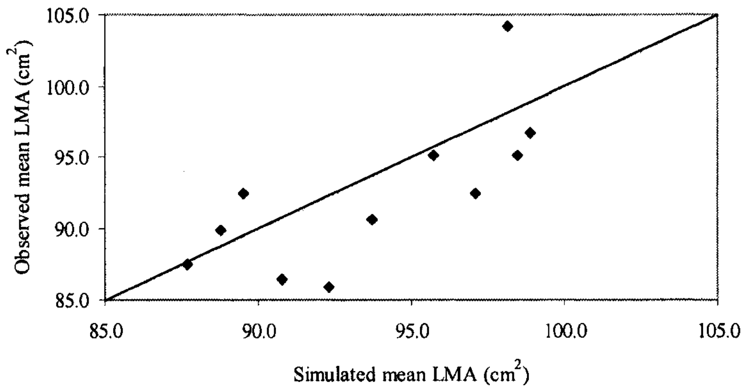

Figure 4.7. Observed versus simulated mean longissimus muscle area (LMA). Line represent y = x.

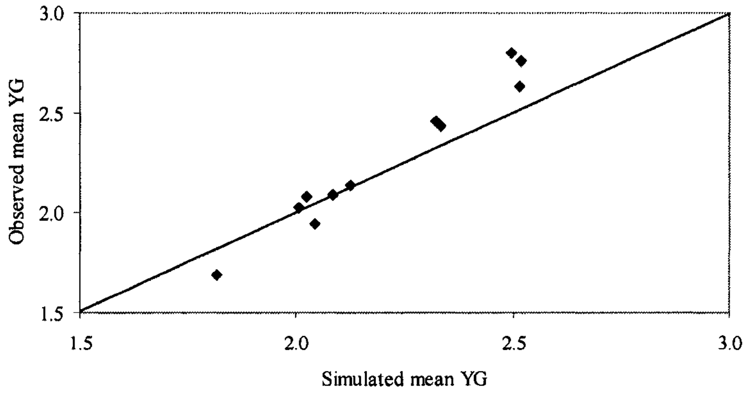

Figure 4.8. Observed versus simulated mean yield grade (YG). Line represent y = x.

Plots of residuals versus simulated mean values for each trait are shown in Figure 4.9. The plots for outWT (a), CWT (b) and FT (c) show little evidence that there exist serious problems with constancy of variance or bias. The MS plot (d) reinforced the conclusions from Figure 4.6 depicting an over-prediction in simulated values, but generally over a small range. The LMA (e) plot showed more evidence of over-

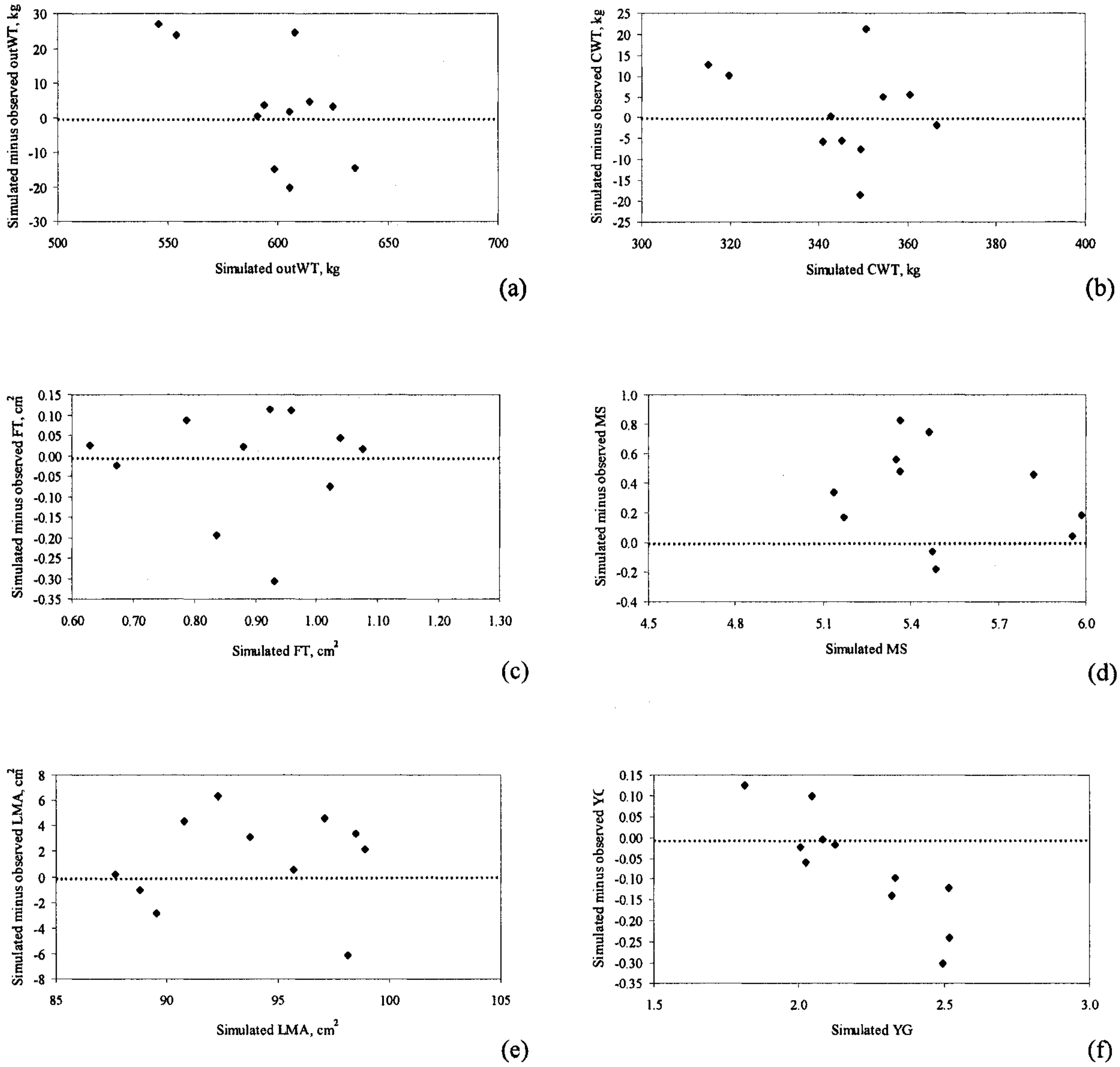

Figure 4.9. Plots of residuals versus simulated mean values for live weight at harvest (a), carcass weight (b), backfat thickness (c), marbling score (d), longissimus muscle area (e) and yield grade (f).

prediction than could be concluded from Figure 4.7, but no variance trend. Figure 4.9(f) shows a negative YG residual trend with increasing YG means. The plot indicates a difficulty in predicting YG at higher levels. Part of the difficulty might be explained by the over-prediction of LMA. LMA is an important component of YG and larger LMA values yield smaller YG values, which would result in under-prediction of YG. The under-prediction is evident and the downward trend may simply be a function of the small number of data points as the points are expected to fluctuate around zero in a random fashion. Plots of studentized residuals versus predicted values (not shown) found no serious outliers.

Table 4.8 shows least squares means and standard errors for *Obs* and *Sim* means, by trait. Mitchell (1997) warned against directly comparing the means using a t-test as the same issues arise as do in regression with validation data, but a subjective evaluation of the means suggests that the model output representing overall production is similar to the actual production, for most traits. The r^2 measures ranged from 55 to 97% and

Table 4.8. Summary statistics for the comparison of observed (Obs) and simulated (Sim) trait means[a].

Trait	Obs[b] (±SE)	Sim[b] (±SE)	r^{2c}	Bias[d]
outWT, kg	597.10 (5.65)	598.08 (0.61)	0.90	-0.60
CWT, kg	344.95 (3.26)	345.05 (0.35)	0.76	-0.44
FT, cm	0.90 (0.03)	0.89 (0.00)	0.55	1.79
MS	5.13 (0.07)	5.51 (0.01)	0.58	-5.91
LMA, cm^2	92.85 (1.07)	93.75 (0.12)	0.54	-1.45
YG	2.27 (0.03)	2.21 (0.00)	0.97	3.17

[a] outWT = live weight at harvest; CWT = carcass weight; FT = backfat thickness; MS = marbling score (small = 5.0); LMA = longissimus muscle area; YG = yield grade

[b] Least squares means

[c] Coefficient of determination for the regression of observed on simulated trait means

[d] Model bias (%); all estimates had $p > 0.10$ where H_0: bias = 0

indicate that the model is able to account for a large amount of the variation in observed values. Each r^2 was significant at the 1% level (except FT, which was significant at the 2% level) supporting the rejection of the hypothesis that each r^2 was obtained by chance. Mean model bias was also low in all cases, ranging from -5.91 to 3.17%, but not differing from zero for any trait ($p > 0.10$). The lack of bias indicates that the model did not suffer from systematic over- or under-prediction of overall trait means.

The mean difference in sire ranking for all traits is depicted in Figure 4.10. The mean ranking differences were very small (< 1) for five of the eleven sires indicating a good relationship between overall *Obs* and *Sim* mean performance. Other ranking differences were in the range -4 to 4. Figure 4.10 discounts the fact that actual differences in trait values may have been small and that the re-ranking may be due to the random variation intrinsic in a stochastic simulation, but it does show that large differences in rank did not exist, suggesting that the model has the ability to correctly

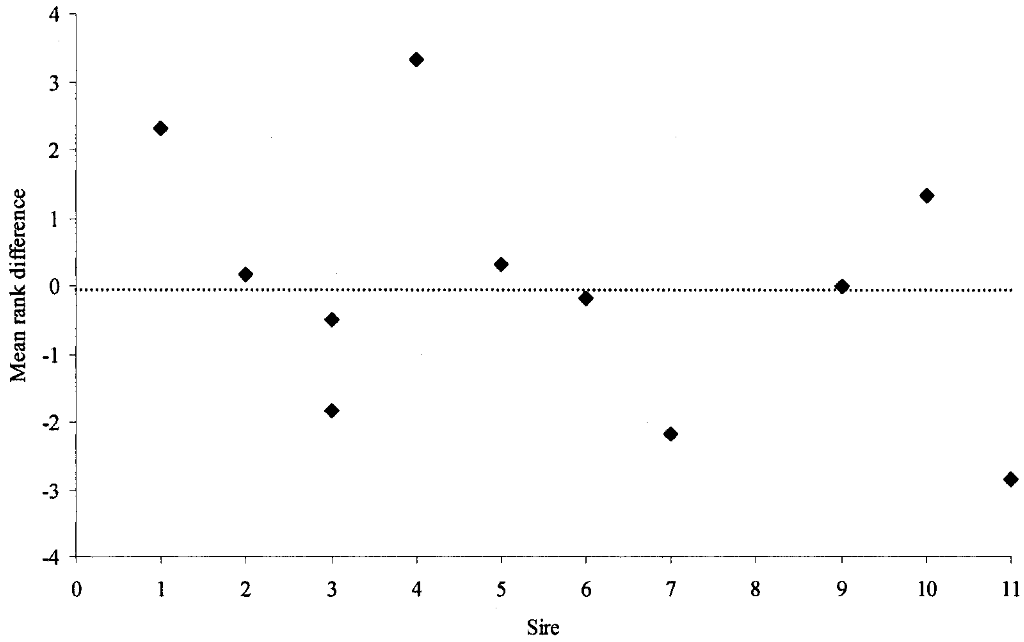

Figure 4.10. Mean difference between sire ranking for observed and simulated trait means.

rank sires. The ranking analysis assumes that sire ranks in the observed population are the true rank and are compared to sire ranks from the simulated population that are predictions of those true values. This assumption may or may not be valid in this small dataset and more observations per sire would allow for a more definitive conclusion as to the actual re-ranking that occurred.

Conclusions. The postweaning model implemented in the DSS appears to account for a sire's growth and carcass genetic potential sufficiently to model progeny growth and carcass composition for the determination of optimum finish endpoints and allow sire selection for optimum profit. As with any model there exists room for improvement, especially in the prediction of marbling score throughout the feeding period, which could very well be improved with the inclusion of random regression marbling score EPD. The model in its current form seems sufficient for use as a research tool, where modeling functions can be objectively evaluated, and as the base for a web-based DSS once needed modifications relating to its desired use have been identified and completed.

Literature Cited

AMS. 2006. Agricultural Marketing Service, U.S. Department of Agriculture. Available at: http://www.ams.usda.gov/lsg/stand/standards/beef-car.pdf. Accessed January 4, 2006.

Analla, M. 1998. Model validation through the linear regression fit to actual versus predicted values. Agric. Syst. 57:115-119.

BIF. Guidelines for uniform beef improvement programs. 8th ed. Beef Improvement Federation. Univ. of Nebraska, Lincoln.

Bourdon, R. M. 1997. Understanding Animal Breeding. Prentice-Hall, Inc., Upper Saddle River, NJ.

Brethour, J. R. 2000. Using serial ultrasound measures to generate models of marbling and backfat thickness changes in feedlot cattle. J. Anim. Sci. 78:2055-2061.

Bruns, K. W., R. H. Pritchard, and D. L. Boggs. The relationship among body weight, body composition, and intramuscular fat content in steers. J. Anim. Sci. 82, 1315-1322. 2004.

Bruns, K. W., R. H. Pritchard, and D. L. Boggs. 2005. The effect of stage of growth and implant exposure on performance and carcass composition in steers. J. Anim. Sci. 83:108-116.

Crews, Jr. D. H., M. Lowerison, N. Canon, and R. A. Kemp. 2004. Genetic parameters among growth and carcass traits of Canadian Charolais cattle. Can. J. Anim. Sci. 84:589-597.

Good, P. 2000. Permutation tests: A practical guide to resampling methods for testing hypotheses. 2nd Ed. Springer-Verlag, New York.

Guiroy, P. J., D. G. Fox, L. O. Tedeschi, and M. J. C. M. D. Baker. 2001. Predicting individual feed requirements of cattle fed in groups. J. Anim. Sci. 79:1983-1995.

Hamlin, K. E., R. D. Green, T. L. Perkins, L. V. Cundiff, and M. F. Miller. 1995. Real-time ultrasonic measurement of fat thickness and longissimus muscle area: I. Description of age and weight effects. J. Anim. Sci. 73:1713-1724.

Harrison, S. R. 1990. Regression of a model on real-system output: An invalid test of model validity. Agric. Syst. 34:183-190.

Kirkpatrick, M., D. Lofsvold, and M. Bulmer. 1990. Analysis of the inheritance, selection, and evolution of growth trajectories. Genetics 124:979-993.

Mayer, D. G. and D. G. Butler. 1993. Statistical validation. Ecol. Model. 68:21-32.

Mitchell, P. L. 1997. Misuse of regression for empirical validation of models. Agric. Syst. 54:313-326.

Nash, S. A., S. N. Harrison, J. H. Packham, R. P. Panting, and S. K. Duckett. 2000. Case study: Monitoring changes in carcass quality across time-on-feed using real-time ultrasound to optimize marketing endpoints. Prof. Anim. Sci. 16:202-205.

Perry, T. C. and D. G. Fox. 1997. Predicting carcass composition and individual feed requirement in live cattle widely varying in body size. J. Anim. Sci. 75:300-307.

Tedeschi, L. O., D. G. Fox, and P. J. Guiroy. 2004. A decision support system to improve individual cattle management. 1. A mechanistic, dynamic model for animal growth. Agric. Syst. 79:171-204.

CHAPTER V

DESCRIPTION OF METHODS AND STUDY DESIGN

Introduction

Three studies were designed to address the stated objectives of this research. The studies were implemented using the postweaning decision-support system (DSS) described in Chapter IV. Following are descriptions of model variables and parameters and the procedures specific to each study used to simulate feedlot animal performance for analysis.

Model parameters

The simulated production system was assumed to be a plains/high plains commercial feedlot with some level of individual cattle management. Progeny of selected sires were assumed to be managed individually to be marketed at the desired endpoint (constant or optimum). Specific values assigned to model parameters are described below.

Animal. Feedlot animals were simulated as the progeny of Red Angus sires and British or British x Continental dams. Mean postweaning performance for Red Angus-cross progeny was reported by Wheeler et al. (2005) in a serial slaughter study evaluating the steer progeny of 21 Red Angus sires and used as the mean performance of the base animal for all simulations. Trait values for model parameterization are listed in Table

5.1. Trait means for a constant finish at 1.1 cm of backfat thickness were used to mirror the common industry practice of marketing cattle at consistent fat endpoints.

Table 5.1. Model parameters for mean animal performance[a].

Parameter[b]	Value
inWT, kg	331.27
Sex	Steer
Age, d	202.00
DOF, d	214.00
outWT[c], kg	572.00
CWT, kg	343.00
CWP, %	59.90
FT, cm	1.10
MS	5.62
LMA, cm^2	78.50
DMI[d], kg	1733.40

[a] Reported by Wheeler et al. (2005)

[b] inWT = live weight at beginning of feeding period; Age = age in days start of feeding period; DOF = days on feed; outWT = live weight at slaughter; CWT = carcass weight; CWP = carcass dressing percent; FT = backfat thickness; MS = marbling score (small = 5.00); LMA = longissimus muscle area; DMI = total dry matter intake

[c] Shrunk weight

[d] From Schoonmaker et al. (2002)

The mean weight of animals at the beginning of the feeding period (inWT) in Table 5.1 was derived from the regression of live weight on days on feed (b = 1.1249) calculated by Wheeler et al., assuming 214 days on feed and a final weight at slaughter (outWT) of 572 kg. The total dry matter intake (DMI) for the feeding period was estimated using daily DMI reported by Schoonmaker et al. (2002) for Angus cross calves started on feed at 202 days of age. They reported a mean DMI of 8.1 kg per day, yielding a total mean DMI of 1733.40 kg for 214 days on feed.

Economics. Economic parameters were divided into costs associated with production and the revenue from harvest of feedlot animals. The costs of production related to feeding and maintenance of animals on feed. Animals were assumed to be fed a typical finishing ration with typical veterinary treatment expenses. Costs were obtained from research in a high plains production system described by Anderson et al. (2005) and were simulated as: \$0.30 per animal per day for yardage (YC); \$0.139 per kg (dry matter basis) (FCkg) for a finishing diet and \$25.00 per animal for fixed production costs (transportation, handling, medication, etc.). Interest costs were charged to the value of an animal and for the value of a loan assumed to be needed to purchase feed and to cover yardage expenses. The annual interest rate (RATE) on an operating loan was 4.6% (FSA, 2005) and assumed to be the same for an animal purchase loan. The amount of the needed operating loan per animal (OLA) was estimated by:

$$\mathrm{OLA} = (\text{total kg feed} \times \mathrm{FCkg}) + (\mathrm{DOF} \times \mathrm{YC}) \qquad [50],$$

yielding an OLA of \$305.14 per animal, assuming mean performance.

Animal value at the beginning and end of the feeding period was determined by the model based on the USDA 10-year average price of feeder and slaughter steers reported by Cattle-Fax (2005). The average price for 318 to 364 kg steers and heifers from 1995 to 2004 was \$81.61 per 45.45 kg live weight resulting in a feeder calf price (FCP) of \$1.80 per kg. The average Choice steer price from 1995 to 2004 was \$70.36 per 45.45 kg live weight and, dividing by the expected CWP of 59.9%, the dressed price was determined to be \$199.25 per 45.45 kg. The carcass base price (BP) was then calculated from the dressed price by:

$$\mathrm{BP} = (100\% - \%\mathrm{CH}) \times \mathrm{CH/SE} + \text{dressed price} \qquad [51]$$

where %CH is the percent of animals expected to have a USDA quality grade of Choice in a particular plant during a specified period of time and CH/SE is the Choice/Select spread. Assuming 60% Choice and a Choice/Select spread of \$11.00 (average in a typical year; Angus GeneNet, 2005) the BP was calculated to be \$123.65 per 45.45 kg carcass weight, for Choice, yield grade 3 carcasses.

Revenue at harvest was determined by the base price and the grid used to assign value to each carcass based on quality and yield characteristics. The carcass specifications were a variation of the Angus GeneNet grid (2005) that is representative of many grids available for Angus and Angus-cross cattle. The grid is considered a "marbling grid" as it places emphasis, in terms of premiums, on marbling score, and thus quality grade. Carcasses outside desired specifications receive large discounts. All carcasses were assumed to be in the "A" maturity category and be free of any defects that would reduce the carcass value. The premiums and discounts used to value carcasses are listed in Table 5.2.

Table 5.2. Carcass grid components used to simulate revenue[a].

	Premium/Discount, $\$\cdot 45.45\ \mathrm{kg}^{-1}$		Premium/Discount, $\$\cdot 45.45\ \mathrm{kg}^{-1}$
Carcass weight		Quality grade	
<243.00 kg	-20.00	Prime	8.00
>432.00 kg	-20.00	High Choice	5.00
Yield grade		Choice	0.00
1	5.00	Select	-6.60
2	3.00	Standard or below	-26.60
3	0.00		
4	-15.00	Base price	123.65
5	-25.00	CH/SE spread	11.00
	%Choice	60.00	

[a]from Angus GeneNet, 2005

Variance components. Variance components for growth and carcass traits were obtained from the Fall 2005 Red Angus genetic prediction analysis (S.E. Speidel, personal communication; Table 5.3). Additive genetic and residual (co)variances for the random regression of weight on days were calculated from the original (co)variances using the procedure described in Chapter IV. The variance components for model parameterization are listed in Table 5.4.

Table 5.3. Additive genetic and residual (co)variance components from the Red Angus analysis for growth and carcass traits[a].

Trait[b]	BW	WW	YW[c]	FT	MS	LMA
BW	**7.31** 5.12	**15.58**	**21.55**			
WW	3.88	**113.84** 244.21	**163.96**			
YW	15.38	284.87	**287.58** 589.99			
FT				**0.02** 0.08		
MS					**0.23** 0.39	
LMA						**16.17** 37.11

[a] Additive genetic (co)variances on upper-diagonal and **bold**; residual (co)variances on lower-diagonal

[b] BW = birth weight (kg); WW = weaning weight (kg); YW = yearling weight (kg); FT = backfat thickness (cm); MS = marbling score; LMA = longissimus muscle area (cm^2)

[c] Calculated from weaning weight and postweaning gain

Other parameters. Other parameter values were defined for simulation of animal performance in the feedlot (Table 5.5). These parameters were not considered central to the model as the model is designed to adjust to the desired growth curve (see Chapter IV) and, thus, realistic average values were assumed. Environmental parameter values were annual averages for the Northern Colorado plains.

Table 5.4. Additive genetic and residual (co)variance components for growth and carcass traits used for model parameterization[a].

Trait[b]	WTb_0	WTb_1	WTb_2	FT	MS	LMA
WTb_0	**7.23**	**0.04**	**-0.000003**			
	5.23					
WTb_1	-0.07	**0.004**	**-0.000006**			
		0.02				
WTb_2	0.0002	-0.00004	**0.00000002**			
			0.0000001			
FT				**0.02**		
				0.08		
MS					**0.23**	
					0.39	
LMA						**16.17**
						37.11

[a] Additive genetic (co)variances on upper-diagonal and **bold**; residual (co)variances on lower-diagonal

[b] WTb_0 = random regression on weight, intercept; WTb_1 = random regression on weight, linear coefficient; WTb_2 = random regression on weight, quadratic coefficient; FT = backfat thickness (cm); MS = marbling score; LMA = longissimus muscle area (cm^2)

Table 5.5. Values used for animal, environment and nutrition model parameters.

Parameter[a]	Value	Parameter	Value
BCS	5	ME, $Mcal{\cdot}kg^{-1}$	3.5
HC	2	NE_g, $Mcal{\cdot}kg^{-1}$	1.4
HCC	2	NE_m, $Mcal{\cdot}kg^{-1}$	2.0
HD	1	PEH, m^3	5.0
HRS, h	12.2	RH, %	40.0
MD, cm	1.0	T_c, °C	10.3
		WS, $km{\cdot}h^{-1}$	6.7

[a] BCS = body condition score; HC = hide code; HCC = hair coat code; HD = hair depth; HRS = hours sunlight; MD = mud depth; ME = metabolizable energy; NE_g = net energy for growth; NE_m = net energy for maintenance; PEH = pen area per head; RH = relative humidity; T_c = average temperature; WS = wind speed

Sire genotypes

A number of sire genotypes were simulated to evaluate the effect on profit of selection of alternative sires for growth and carcass traits. Two genetic levels, high (H) and low (L), were considered for each of four traits: weight, backfat thickness (FT), marbling score (MS) and longissimus muscle area (LMA). Weight initially consisted of weights at birth (BW), weaning (WW) and yearling (YW). The genetic level indicated the trait level (e.g., high or low) desired in most production systems. The (H) level assumed a desire for low BW, high WW and YW, low FT and high MS and LMA, with opposite levels for (L).

The genetic values for each trait were obtained from the Fall 2005 Red Angus sire summary. The EPD were selected independently for each trait to represent sires in the top (H) and bottom (L) 10% of all sires evaluated (Table 5.6). Genetic values for the average sire were also obtained. The random regression weight EPD was then calculated from the BW, WW and YW EPD using the procedure described in Chapter IV (Table5.6). Accuracies were not included as no variation due to prediction error was

Table 5.6. Original growth and carcass EPD from Red Angus sire summary and EPD converted for simulation.

	Original EPD[a]					
Level[b]	BW	WW	YW	FT	MS	LMA
High (H)	-1.32	20.00	34.55	-0.05	0.25	1.35
Low (L)	1.86	8.18	14.55	0.05	-0.13	-1.74
AVG	0.27	14.09	24.55	0.00	0.06	-0.19

	EPD for Simulation[c]					
Level	WTb_0	WTb_1	WTb_2	FT	MS	LMA
High (H)	-1.43	0.11	-0.000037	-0.05	0.25	1.35
Low (L)	1.84	0.03	0.000024	0.05	-0.13	-1.74
AVG	0.20	0.07	-0.000007	0.00	0.06	-0.19

[a] BW = birth weight (kg); WW = weaning weight (kg); YW = yearling weight (kg); FT = backfat thickness (cm); MS = marbling score; LMA = longissimus muscle area (cm^2)

[b] H = top 10% of evaluated sires; L = bottom 10% of evaluated sires; AVG = average EPD

[c] WTb_0 = random regression on weight, intercept; WTb_1 = random regression on weight, linear coefficient; WTb_2 = random regression on weight, quadratic coefficient

assumed for the simulations. All possible combinations of (H) and (L) genetic levels for each trait were simulated, considering the three weight components as one trait and yielding 16 sire genotypes (Table 5.7).

Table 5.7. Sire genotypes for simulation of progeny postweaning performance and profit.

	EPD^a					
Genotype^b	WTb_0	WTb_1	WTb_2	FT	MS	LMA
HHHH	-1.43	0.11	-0.000037	-0.05	0.25	1.35
HLHH	-1.43	0.11	-0.000037	0.05	0.25	1.35
HLHL	-1.43	0.11	-0.000037	0.05	0.25	-1.74
HLLH	-1.43	0.11	-0.000037	0.05	-0.13	1.35
HLLL	-1.43	0.11	-0.000037	0.05	-0.13	-1.74
HHLH	-1.43	0.11	-0.000037	-0.05	-0.13	1.35
HHLL	-1.43	0.11	-0.000037	-0.05	-0.13	-1.74
HHHL	-1.43	0.11	-0.000037	-0.05	0.25	-1.74
LHHH	1.84	0.03	0.000024	-0.05	0.25	1.35
LHLH	1.84	0.03	0.000024	-0.05	-0.13	1.35
LHLL	1.84	0.03	0.000024	-0.05	-0.13	-1.74
LHHL	1.84	0.03	0.000024	-0.05	0.25	-1.74
LLHH	1.84	0.03	0.000024	0.05	0.25	1.35
LLHL	1.84	0.03	0.000024	0.05	0.25	-1.74
LLLH	1.84	0.03	0.000024	0.05	-0.13	1.35
LLLL	1.84	0.03	0.000024	0.05	-0.13	-1.74

[a] WTb_0 = random regression on weight, intercept; WTb_1 = random regression on weight; linear coefficient; WTb_2 = random regression on weight, quadratic coefficient; FT = backfat thickness; MS = marbling score; LMA = longissimus muscle area

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

Statistical Analysis

The stochastic option of the DSS was used to simulate performance and profit of sire progeny at the desired endpoint. Output for all traits was analyzed using the UNIVARIATE procedure of SAS and it was determined that the assumption of normality should be rejected for the data ($p < 0.01$) and thus medians were used for further analysis as more appropriate estimators of population central values (Ott and Longnecker, 2001).

Medians were computed for all traits and comparisons were performed using the nonparametric Wilcoxon Rank Sum Test that replaces data values with ranks to deal with possible extreme values, an option of the NPAR1WAY procedure in SAS. Variance comparisons were performed using Levene's Test for inequality of variances which looks at average distances from the median, an option of the GLM procedure in SAS. Finally, the CORR procedure of SAS was used to calculate the Pearson correlation coefficients to measure the association between variables.

Study 1 description

Study 1 addressed the second project objective and was designed to evaluate changes in expected profit when not accounting for growth and carcass genetic differences when marketing cattle at a pre-determined constant finish endpoint. The objective of this study was to determine the impact on profit of feeding alternative sire progeny to a common backfat thickness endpoint, to assess the difficulty in selecting sires based on traditional EPD and identify the factors important to increased profit in the current production system.

Progeny performance (n = 1000) was simulated stochastically for each progeny of all sire genotypes (n = 16) and compared to performance of average (AVG) sire progeny. The AVG sire progeny represented the average performance that would be expected when feeding Red Angus-sired calves and would yield the average expected profit assuming no knowledge of genetic differences. Each progeny was simulated to remain on feed until reaching 1.1 cm of backfat to evaluate the range of profit possible for different sire genotypes. Due to the stochastic nature of the simulation, cattle on the extreme low end of backfat thickness would not necessarily reach the constant backfat in any realistic amount of time and, thus, up to 350 days on feed were allowed before cattle were harvested, regardless of backfat thickness.

Profit at the constant backfat thickness endpoint was evaluated for each sire genotype and compared to the profit that would be expected when not accounting for growth and carcass genetic potential. Changes in expected profit were assessed by comparing the distribution of profit realizations for each genotype to the AVG genotype to determine if certain genotypes were more likely to yield less than expected profits. A plot of profit by trait levels was used to evaluate the ability to select for profit using growth and carcass EPD alone. Percent changes in feedlot performance with increased profit and correlation coefficients were used to determine factors associated with that increase.

Study 2 description

Study 2 addressed the third project objective and was designed to evaluate the profit impact of managing feedlot growth to market cattle at an individual optimum profit endpoint compared to feeding to the constant endpoint in Study 1. The objective was to investigate the economic advantages, including median profit and risk, of individual cattle management and identification of optimum finish endpoints for alternative sire progeny. Additionally, selection processes and performance associations with profit were compared to Study 1 results.

Progeny performance (n = 1000) was stochastically simulated for each progeny of all sire genotypes (n = 16) until each animal reached the finish endpoint that maximized profit, considered the optimum endpoint. Median performance and profit, and profit distributions, were evaluated for each sire and compared to the results in Study 1 to assess the impact of feeding to optimum endpoints. Specifically the change in potential to produce low profit progeny was assessed. Procedures for these comparisons and other analyses were described in the previous section. Progeny were simulated for a maximum of 350 days on feed assuming that the global profit maximum, at least for a realistic commercial production situation, would occur within this time period.

Study 3 description

Study 3 addressed the fourth project objective and was designed to evaluate the effect of selection for expected improvements in growth and carcass traits over time on profit at the constant finish endpoint in Study 1, the optimum finish endpoint identified for a particular sire genotype in Study 2 and the optimum finish endpoint identified for each sire genotype. The objective was to investigate the impact on profit of ignoring selection when selecting a marketing endpoint for alternative sire progeny and to determine if selection for growth and carcass traits yields improvements in profit at pre-determined endpoints.

Genetic change was simulated in each trait for sire genotype HHHH. Assuming a BIF accuracy of prediction of 0.75, one standard deviation of selection and a generation interval of six years, the maximum amount of genetic change in each individual trait was calculated over five years to yield five new genotypes (Table 5.8). The new genotypes were not meant to necessarily represent the actual amount of genetic change expected over five years, but rather to illustrate the effects that selection could have on profit over time or the type of sires that could be selected each year for desirable trait levels. Selection was considered to occur in the direction of the desired change, e.g. decreased FT and increased MS. Progeny performance (n = 1000) was simulated for each of the new sire genotypes (n = 5) to finish at 1.1 cm of backfat thickness (Study 1), at the median optimum finish backfat thickness endpoint identified for the HHHH sire genotype in Study 2 and at the individual optimum finish endpoint. Simulation procedures for each endpoint option were identical to those in Study 1 and 2. Median profits and profit distributions using the new genotypes were compared to the profit distributions for the HHHH sire genotype in the previous studies. Other analyses were also performed as described in the previous sections.

Table 5.8. Growth and carcass EPD for sire genotype HHHH for five years of selection.

	EPD[a]					
Genotype[b]	WTb_0	WTb_l	WTb_2	FT	MS	LMA
HHHH1	-1.60	0.12	-0.000029	-0.06	0.28	1.61
HHHH2	-1.77	0.12	-0.000021	-0.07	0.31	1.86
HHHH3	-1.93	0.12	-0.000012	-0.08	0.34	2.11
HHHH4	-2.10	0.13	-0.000004	-0.09	0.37	2.36
HHHH5	-2.27	0.13	0.000004	-0.10	0.40	2.61

[a] WTb_0 = random regression on weight, intercept; WTb_l = random regression on weight; linear coefficient; WTb_2 = random regression on weight, quadratic coefficient; FT = backfat thickness; MS = marbling score; LMA = longissimus muscle area

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA; the increasing numbers indicate successive selection for each trait

Literature Cited

Anderson, R. V., R. J. Rasby, T. J. Klopfenstein, and R. T. Clark. 2005. An evaluation of production and economic efficiency of two beef systems from calving to slaughter. J. Anim. Sci. 83:694-704.

Angus GeneNet. 2005. Available at: http://www.genenetbeef.com. Accessed October 17, 2005.

Cattle-Fax. 2005. Available at: http://www.cattle-fax.com. Accessed October 17, 2005.

FSA. 2005. Farm Service Agency. United States Department of Agriculture. Greeley, CO.

Ott, R.L. and M. Longnecker. 2001. An introduction to statistical methods and data analysis. Duxbury, Pacific Grove, CA. 5th ed.

Schoonmaker, J. P., S. C. Loerch, F. L. Fluharty, H. N. Zerby, and T. B. Turner. 2002. Effect of age at feedlot entry on performance and carcass characteristics of bulls and steers. J. Anim. Sci. 80:2247-2254.

Wheeler, T. L., L. V. Cundiff, S. D. Shackelford, and M. Koohmaraie. 2005. Characterization of biological types of cattle (Cycle VII): Carcass, yield, and longissimus palatability traits. J. Anim. Sci. 83:196-207.

CHAPTER VI

RESULTS AND DISCUSSION

The results are presented and discussed separately for each of the studies described in Chapter V. Each study was designed to address specific objectives of this research as defined in previous chapters. Following the study results is a brief example of model output application for producer selection decisions.

Study 1

Simulated feedlot and harvest trait medians when progeny were fed to a constant 1.1 cm backfat thickness (FT) are listed for each genotype in Table 6.1. Performance for progeny of the AVG genotype represents what would be expected for Red Angus-cross calves assuming the average genetic potential for growth and carcass traits from the Red Angus genetic evaluation. Assuming average performance for Red Angus-sired calves under the current production system a producer could expect a median profit of $32.18 (Table 6.1) at the constant FT endpoint and 205 days on feed. This profit is similar to values reported in large feedlot studies for similar weight cattle that were assumed to be marketed at a constant FT (Langemeier et al., 1992; Lawrence et al., 1999). The other median profit values in Table 6.1 illustrate the range of profit possible when accounting for genetic differences in growth and carcass traits and rather than assuming a breed or breed-type average.

Table 6.1. Simulated trait medians for each sire genotype when progeny were fed to a constant 1.1 cm backfat thickness.

	Trait[a]							
Genotype[b]	DOF	outWT	CWT	MS	LMA	QG	YG	Profit
HHHH	208	611.93	365.94	6.28	80.14	CH+	3.23	50.78[x]
HHHL	209	603.24	360.46	6.26	77.12	CH+	3.38	48.55[x]
HHLH	242	626.85	377.00	5.41	86.54	CH	3.02	19.56[x]
HHLL	218	617.33	370.83	5.73	78.72	CH	3.30	27.46[x]
HLHH	225	611.90	366.71	5.74	80.71	CH	3.26	22.22[x]
HLHL	215	602.56	361.66	5.90	75.71	CH	3.41	30.28
HLLH	196	590.99	353.21	5.87	76.25	CH	3.34	28.29[x]
HLLL	206	605.16	362.10	5.67	74.41	CH	3.48	18.63[x]
LHHH	205	586.37	350.65	6.33	79.68	CH+	3.12	36.89[x]
LHHL	209	585.61	350.29	6.26	77.12	CH+	3.26	34.70[x]
LHLH	234	604.39	362.91	5.50	84.83	CH	2.98	16.36[x]
LHLL	228	600.39	360.05	5.58	80.45	CH	3.15	8.66[x]
LLHH	229	591.43	354.91	5.69	81.32	CH	3.14	11.18[x]
LLHL	212	580.75	347.75	5.95	75.24	CH	3.29	16.01[x]
LLLH	197	574.31	342.74	5.88	76.28	CH	3.25	15.54[x]
LLLL	207	585.05	350.37	5.65	74.50	CH	3.36	3.75[x]
AVG[c]	205	593.81	355.01	6.01	76.96	CH+	3.30	32.18

[a] DOF = days on feed; outWT = live weight at harvest (kg); CWT = carcass weight (kg); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); QG = quality grade (CH = Choice; CH+ = upper 2/3 Choice); YG = yield grade

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[c] Genotype with mean EPD for each trait

[x] Differs from the AVG genotype median profit ($p < 0.05$)

Median profit for the 16 sire genotypes ranged from \$3.75 to \$50.78 for 196 to 242 days on feed (Table 6.1). All but one median profit differed from the AVG profit ($P < 0.05$) indicating that profit at a constant FT will be different than expected depending on sire genotype for growth and carcass traits. Considering the genotypes simulated, most (12) had median profits equal to or less than the AVG genotype suggesting that not accounting for sire genotypes may be detrimental to profitability at a constant FT in relation to expected profit, though no genotypes yielded negative median profit. The results indicate that in this case sire genotypes may be less profitable than expected and accounting for genetic differences improves the evaluation of potential profit outcomes in the postweaning phase.

Profit differences between genotypes are important, but the range of predicted profit illustrates the impact of not accounting for genetic differences. Figure 6.1 depicts the range of predicted profit realizations for each sire genotype. All genotypes had positive median profit values, but the range of predicted profits indicates that large negative outcomes are possible for each sire. Expected profit for the average animal (AVG) ranged from a minimum of \$-295.29 to a maximum of \$97.75. All sire genotypes had similar maximum profit values (\$74.61 to \$142.60), but minimum profit realizations were both much less and much more negative than the AVG genotype (\$-170.03 to \$-380.38). The effect of ignoring genetic differences is evident. Progeny produced by certain sire genotypes are at risk of yielding considerably lower profits than expected without the possible reward of receiving higher than expected profits.

An illustration of potential negative profits is the comparison between the AVG and HHLH genotypes. The HHLH genotype had the lowest minimum profit at \$-380.38 and a plot of the realizations (Figure 6.2) shows a wider distribution of HHLH to AVG profit indicating a greater probability of having values at the lower end of the range. The maximum predicted profit for HHLH was \$104.34, only slightly larger than AVG, while the HHLH median profit was smaller than AVG. When average breed performance is assumed, but genetics for growth and carcass traits are actually contributed by sires such as the HHLH genotype, median profits yielded and the range of potential profits are smaller than expected, at a constant FT endpoint. If genetic differences are ignored there

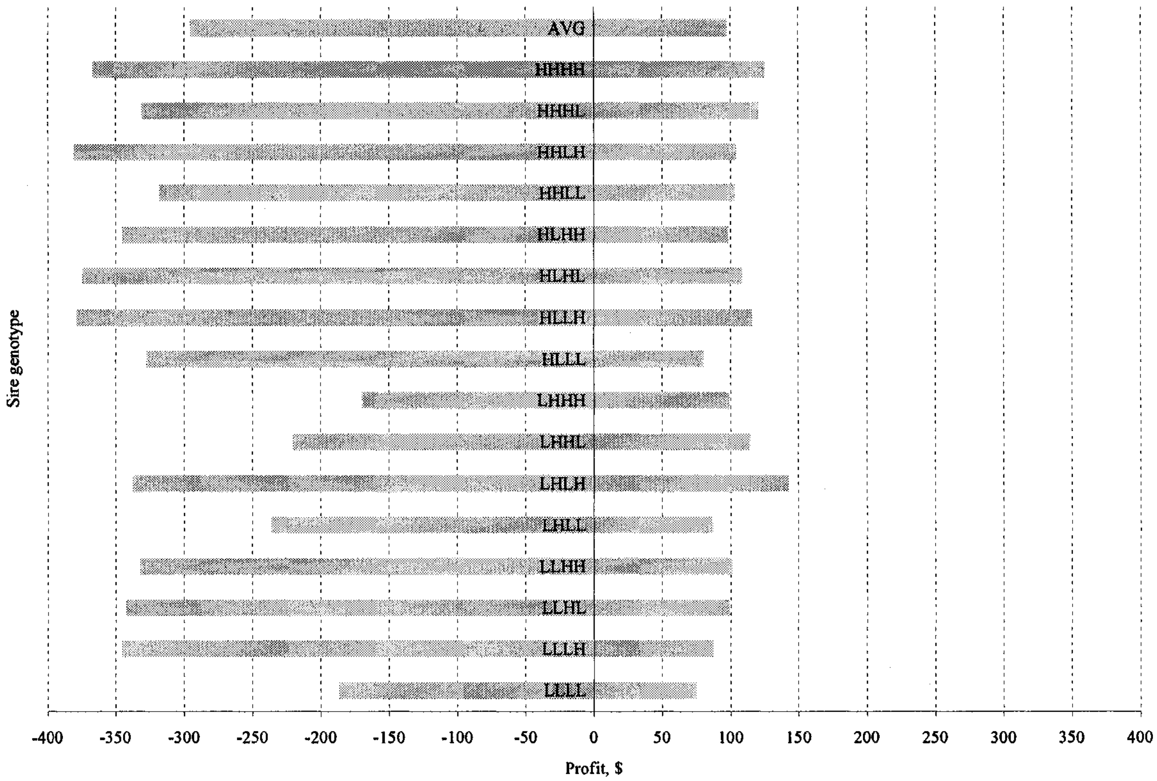

Figure 6.1. Range in profit realizations (n = 1000) for each sire genotype at a constant 1.1 cm backfat thickness. The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

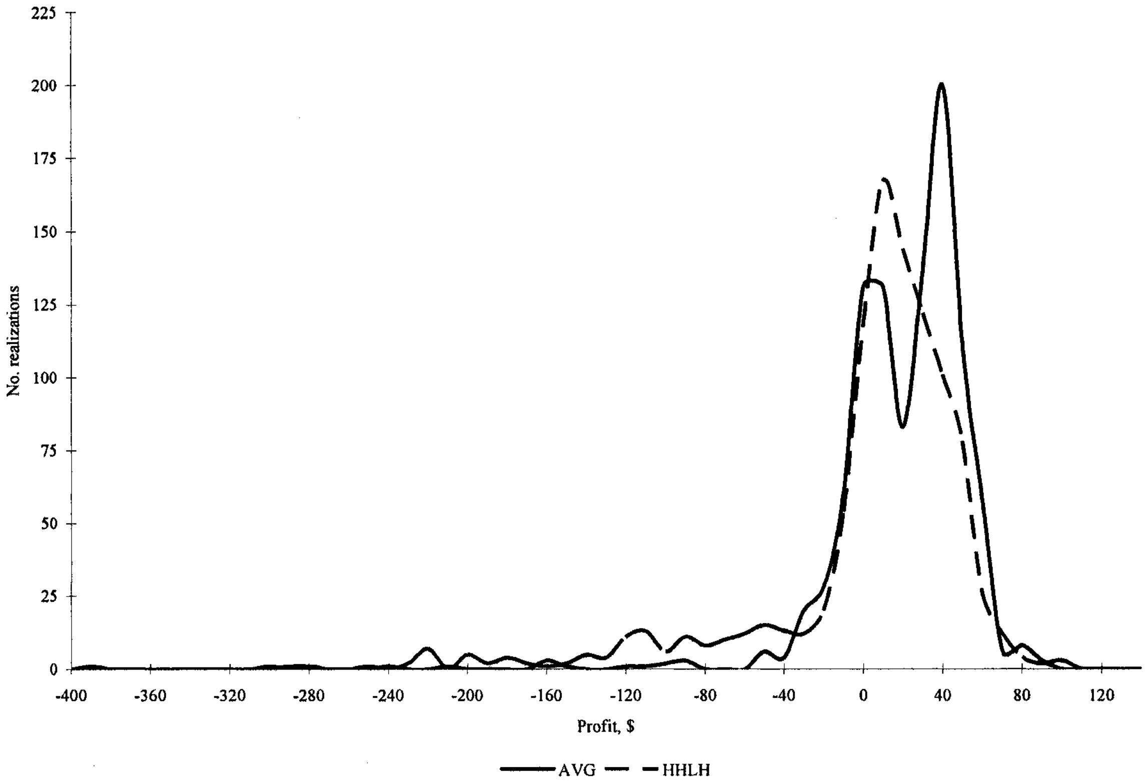

Figure 6.2. Distribution of profit realizations (n = 1000) for the AVG and HHLH sire genotypes at a constant 1.1 cm backfat thickness. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

is a potential negative impact of particular genotypes within a breed on postweaning profitability.

The results indicate that the use of certain genotypes reduced the probability of having lower than expected profit outcomes as some sires had smaller probabilities of negative predicted profit at a FT constant endpoint. Figure 6.3 depicts a comparison of profit realizations for the LHHH and AVG genotypes. The LHHH sire had a slightly larger median and maximum profit than AVG, but the minimum predicted profit was considerably less negative ($-170.03 versus $-295.29). The distributions indicate a smaller probability of yielding negative profits using the LHHH sire. Similarly, Figure 6.4 shows the profit realizations for the HHHH and AVG genotypes. In this case the HHHH median profit is much larger than AVG ($50.78 versus $32.18), but the HHHH predicted minimum is also more negative ($-380.38). Results using the HHHH genotype suggest low potential profits when evaluating only the profit range, but the distribution suggests that the use of the HHHH sire imparts a lower probability of negative profit outcomes. The results support accounting for genetic differences between sires to better predict expected outcomes allowing selection for sires with the most profitable combination of traits when progeny are fed to a constant FT endpoint.

Determining the most profitable sire genotype, by evaluating weight (WT), marbling score (MS), longissimus muscle area (LMA) and FT, is difficult as the optimum combination of trait levels is different for every production situation and is likely unknown in most situations. The difficulty in combining traits to maximize profitability is illustrated by Figure 6.5, where median profit is plotted for each combination of EPD

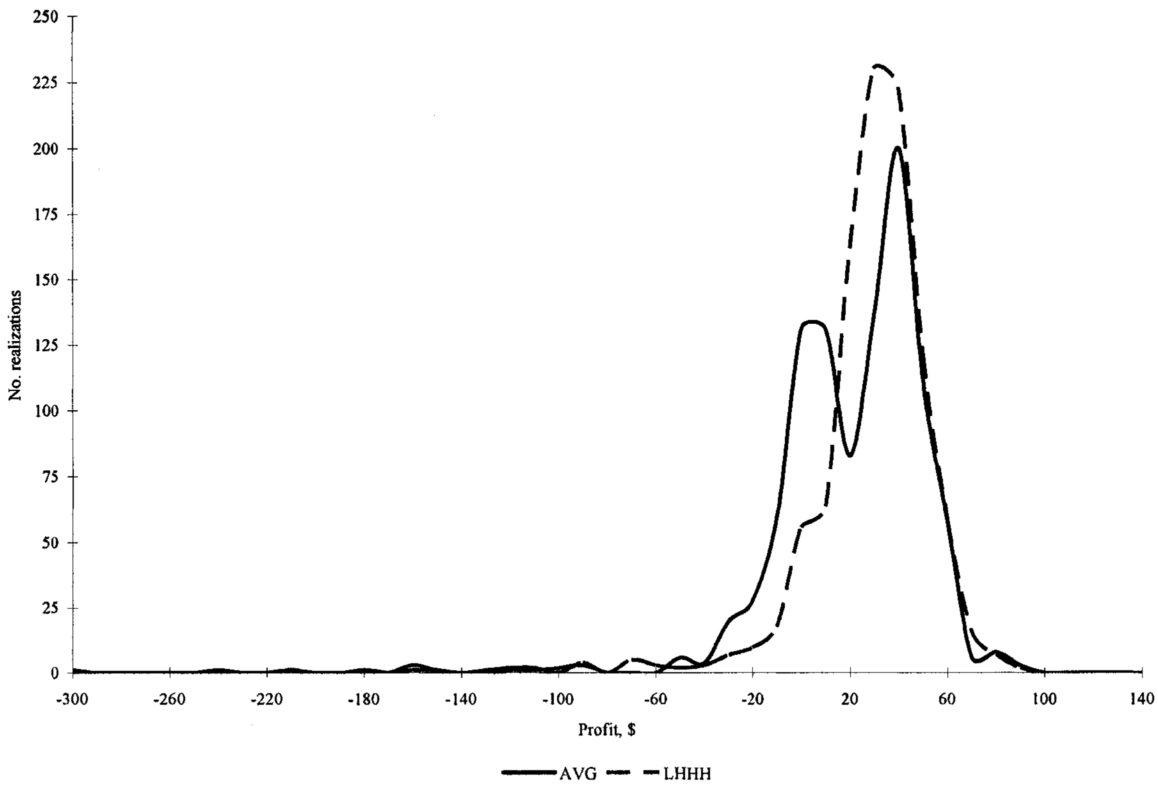

Figure 6.3. Distribution of profit realizations (n = 1000) for the AVG and LHHH sire genotypes at a constant 1.1 cm backfat thickness. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

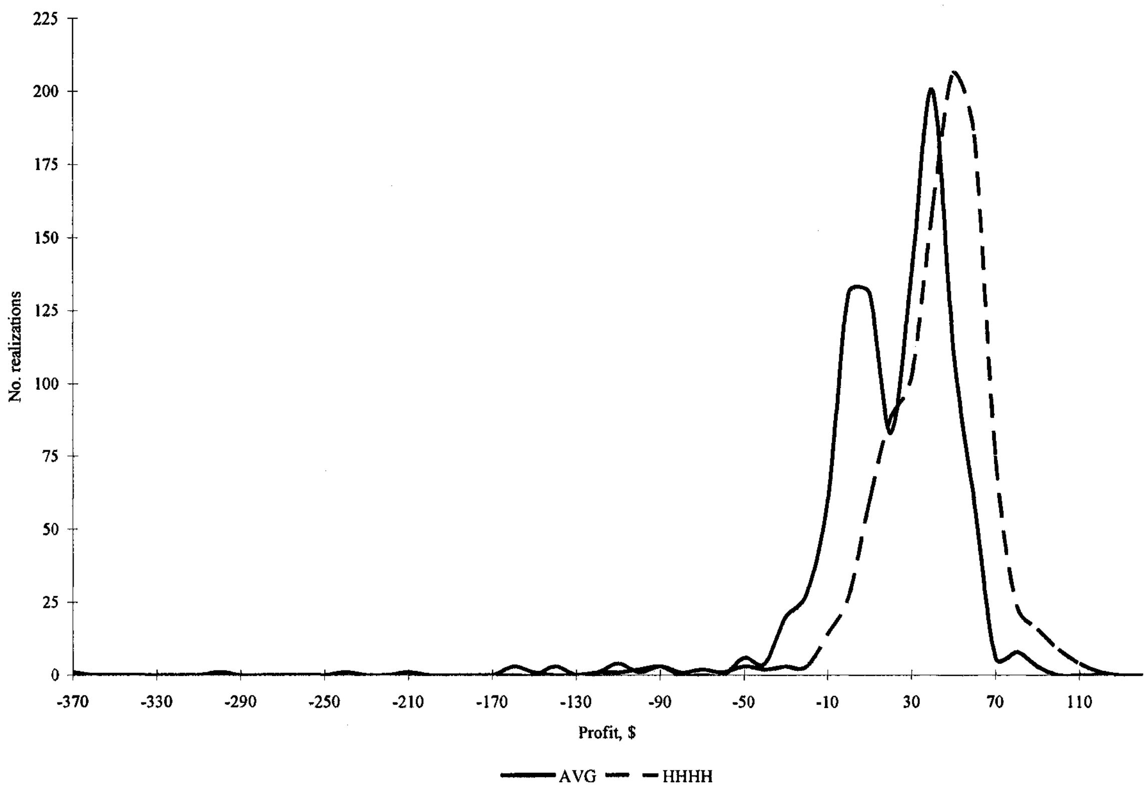

Figure 6.4. Distribution of profit realizations (n = 1000) for the AVG and HHHH sire genotypes at a constant 1.1 cm backfat thickness. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

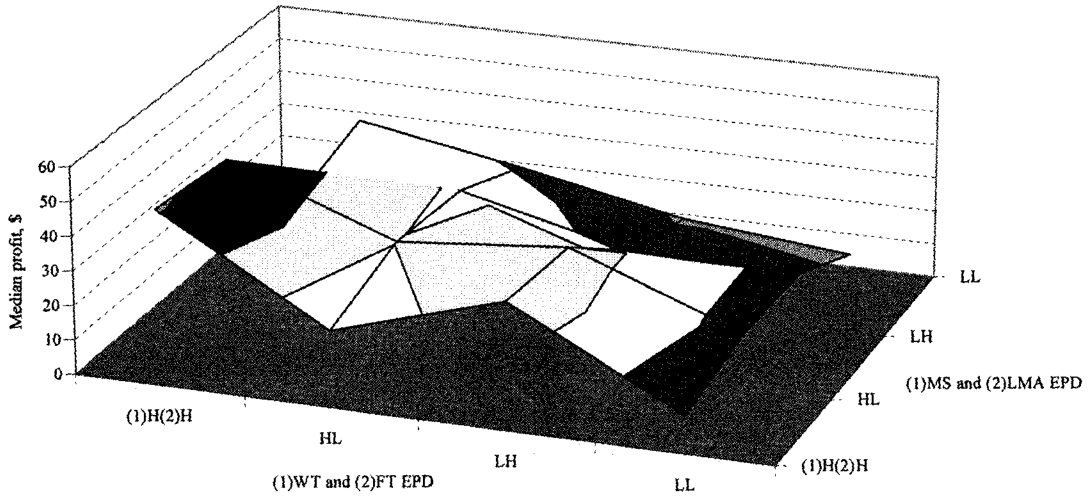

Figure 6.5. Surface plot of median profit at each EPD level, high (H) or low (L) for each trait; weight at a specified day on feed (WT); backfat thickness (FT); marbling score (MS) and longissimus muscle area (LMA) at a constant 1.1. cm backfat thickness.

level, with the highest levels at the left and front. There is some discernable trend moving from high to low EPD levels for WT and FT, MS and LMA, and for the four trait combination, but it is difficult to draw any definite conclusions. The plot indicates a profit peak for intermediate low and high combinations with the expected maximum at HHHH and minimum at LLLL. Between the extremes these results suggest that intuitive selection processes for growth and carcass traits do not exist for maximizing profit for these genotypes at a FT constant endpoint and support Bourdon's (1988) assertion that the identification of optimal production will not stem from readily apparent changes in single traits.

Genotype combinations that maximize profit were not obvious, but evaluating associations between traits and profit may be useful for identifying trait levels that make certain genotypes more profitable under the current management system. Figure 6.6 shows the percent change in trait medians with increasing genotype profitability. Most of the trait values oscillate around either a positive or negative change as profit increases. The levels that might be expected to maximize profit (e.g., increased outWT, LMA and decreased YG) occur in the intermediate range where DOF is increased and MS is decreased. Profit increases to its maximum as the oscillation of trait levels is reduced and the changes are intermediate to those that occurred with less profitable genotypes. These reduced changes occur in conjunction with increased changes in MS and decreased changes in DOF. Profit is at its maximum, considering the simulated genotypes, with small increases in outWT and LMA, a small decrease in YG, a large increase in MS and little change in DOF over the least profitable genotype.

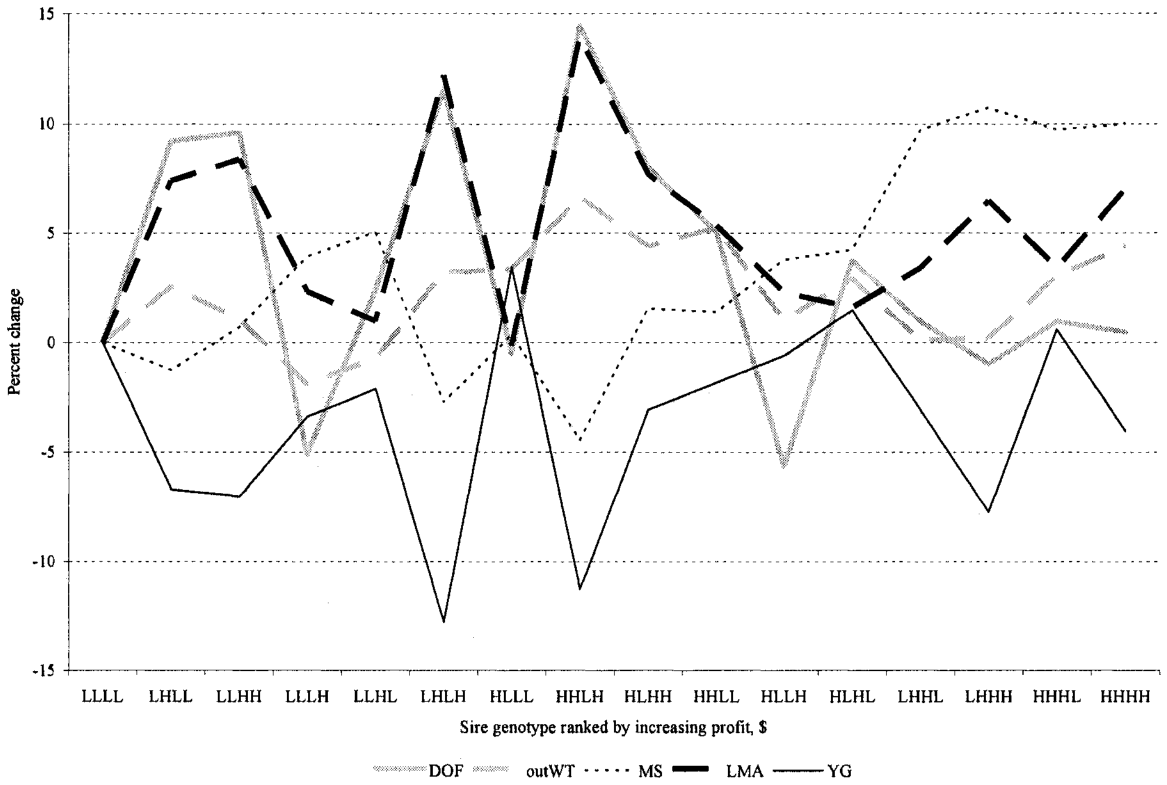

Figure 6.6. Percent change in median days on feed (DOF), live weight at harvest (outWT), marbling score (MS), longissimus muscle area (LMA) and yield grade (YG), for increasing median profit, at a constant 1.1 cm backfat thickness (FT). The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

Correlations between traits and profit were calculated to further evaluate any relationships. The correlation coefficients in Table 6.2 suggest a strong negative relationship between DOF and profit and a strong positive relationship between MS and profit at a FT constant endpoint. This result further supports the importance of these two factors in more profitable genotypes and is in agreement with the findings of Feuz et al. (1993) using a similar marketing system. Little relationship was found between outWT and profit, whereas an association might be assumed with larger carcasses having a larger amount of product. The correlations between LMA, YG and profit were counterintuitive, but suggest that yield characteristics were not important for genotype profitability under this management system, using a marbling-focused grid, and agree with Schroeder and Graff (1999) who reported that the most important non-price factor affecting revenue was QG, as QG premiums and discounts are generally larger than YG price differentials. Pyatt et al. (2005a) found weight and MS to be the most critical carcass factors contributing to differences in profitability at a similar finish endpoint using Simmental-sired calves. The lack of apparent weight effects in this study may be due to difference in weight gain between Continental and British cattle, as calves in this study were not overweight and thus not subject to out-weight discounts. The factors associated with the largest median profits are sensible considering the grid employed, but identification of genotypes that exhibit desired levels for these traits remains difficult as it is the optimum combination of these factors that is important and it is not appropriate to simply select for reduced DOF or increased MS to maximize profit.

Table 6.2. Correlation coefficients for the phenotypic correlation of simulated trait values with simulated profit for each sire genotype, in ascending order of median profit, and for the AVG sire genotype, at a constant 1.1 cm backfat thickness.

	Trait[a]					
Genotype[b]	DOF	outWT	FT	MS	LMA	YG
LLLL	-0.77	-0.01	0.25	0.69	-0.76	0.85
LHLL	-0.66	0.10	0.46	0.55	-0.69	0.85
LLHH	-0.64	0.15	0.54	0.60	-0.62	0.85
LLLH	-0.68	0.15	0.58	0.61	-0.66	0.85
LLHL	-0.65	0.18	0.55	0.57	-0.65	0.87
LHLH	-0.64	0.12	0.54	0.59	-0.63	0.86
HLLL	-0.76	-0.18	0.40	0.69	-0.77	0.84
HHLH	-0.69	-0.05	0.50	0.60	-0.71	0.80
HLHH	-0.65	0.07	0.46	0.57	-0.65	0.81
HHLL	-0.68	0.04	0.51	0.61	-0.68	0.84
HLLH	-0.65	0.30	0.48	0.57	-0.63	0.80
HLHL	-0.62	0.16	0.42	0.51	-0.64	0.83
LHHL	-0.55	0.23	0.43	0.44	-0.60	0.84
LHHH	-0.62	0.15	0.34	0.50	-0.66	0.85
HHHL	-0.66	0.09	0.56	0.54	-0.69	0.82
HHHH	-0.61	0.06	0.55	0.55	-0.63	0.83
AVG[c]	-0.68	0.15	0.51	0.62	-0.65	0.85

[a] DOF = days on feed; outWT = live weight at harvest (kg); FT = backfat thickness (cm); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); YG = yield grade

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[c] Genotype with breed average EPD for each trait

Study 2

Simulated trait medians for each sire genotype when progeny were fed to the OPT endpoint are listed in Table 6.3. Median profit for the AVG genotype was $54.13 at 258 days on feed and median profit ranged from $29.10 to $74.33, for 253 to 262 days on feed, for the other 16 sire genotypes. The AVG profit was similar to a value reported by Amer et al. (1994b) for Angus cattle fed to optimum endpoints. Fewer sires differed from the AVG genotype for median profit at the OPT endpoint (n = 11) versus a constant

Table 6.3. Simulated trait medians for each sire genotype when progeny were fed to their individual optimum endpoint, the point at which profit was maximized.

	Trait[a]								
Genotype[b]	DOF	outWT	CWT	FT	MS	LMA	QG	YG	Profit
HHHH	260[x]	658.91	397.28	1.74	6.62	100.75	CH+	3.16	69.66[y]
HHHL	257	658.60	397.12	1.62	6.48	95.33	CH+	3.29	62.71[y]
HHLH	258[x]	660.39	397.89	1.92	6.53	102.56	CH+	3.24	67.97[y]
HHLL	262	655.87	395.20	1.56	6.00	94.72	CH+	3.24	45.43[y]
HLHH	259	651.25	392.49	1.64	6.03	96.44	CH+	3.22	54.99
HLHL	253[x]	666.26	401.36	1.92	6.75	96.86	CH+	3.54	74.33[y]
HLLH	262[x]	657.03	396.51	1.97	6.01	101.24	CH+	3.34	57.51
HLLL	257[x]	660.59	398.20	1.82	6.01	95.82	CH+	3.44	51.30
LHHH	257	631.61	380.44	1.69	6.68	99.49	CH+	3.05	58.83[y]
LHHL	257[x]	640.42	385.51	1.90	7.00	98.81	CH+	3.30	59.13[y]
LHLH	255[x]	632.72	380.88	1.82	6.45	100.54	CH+	3.08	53.35
LHLL	262[x]	622.85	375.36	1.52	6.01	94.81	CH+	3.08	29.10[y]
LLHH	256[x]	620.37	373.73	1.63	6.17	95.62	CH+	3.10	41.20[y]
LLHL	255[x]	636.37	383.88	2.00	6.91	97.95	CH+	3.43	53.89
LLLH	258	630.17	380.20	1.82	6.00	98.51	CH+	3.17	34.46[y]
LLLL	258	628.20	378.20	1.76	6.00	95.58	CH+	3.28	32.78[y]
AVG[c]	258	643.67	387.64	1.84	6.19	98.67	CH+	3.25	54.13

[a] DOF = days on feed; outWT = live weight at harvest (kg); CWT = carcass weight (kg); FT = backfat thickness (cm); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); QG = quality grade (CH = Choice; CH+ = upper 2/3 Choice); YG = yield grade

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[c] Genotype with breed average EPD for each trait

[x] Differs from the AVG genotype median DOF ($P < 0.05$)

[y] Differs from the AVG genotype median profit ($p < 0.01$)

FT (n = 15) and there were relatively equal numbers above and below the AVG median, suggesting a smaller amount of spread in median profit due to genetic differences. The range of predicted median profit at the OPT endpoint ($45.23) was similar to the FT endpoint ($47.03), as were the ranges in trait values that yield profit at each endpoint (results not shown), with the exception of DOF. Profit variances did not differ between endpoints ($p > 0.05$). These results indicate that variation in median performance for growth, carcass and profit, due to genetic differences, is not reduced by feeding to an optimum endpoint suggesting that some genotypes, on average, will always be less profitable than others, or than expected (AVG), regardless of the management system.

Variation in median DOF was reduced at the OPT endpoint, ranging from 253 to 262 days, compared to 196 to 242 days at the FT endpoint, with differing standard deviations ($p < 0.01$). The majority of sires ($n = 10$) had a median DOF that differed from the AVG sire (Table 6.3), but the differences were small and may not have a large effect in practice. The similarity in DOF seems to suggest that, in this management system, a DOF target for breed average performance might be appropriate for achieving maximal profit as cattle fed for the prescribed number of days will, on average, reach an individual optimum profit regardless of sire genotype. Amer et al. (1994a) suggested such a scheme may be appropriate using average DOF for 11 different genotypes as a single harvest-point criterion. Average individual profit can be optimized in this way, but results show that accounting for genetic differences will still yield greater profitability (Table 6.3).

The differences in medians between the OPT and FT endpoints listed in Table 6.4 indicate that the endpoint of optimal profit for these genotypes (including the Red Angus breed average genotype) occurs at a point beyond 1.1 cm backfat thickness, though Block et al. (2001) suggested a similar FT endpoint for cattle marketed in value-based systems that target marbling superiority and Amer et al. (1994a) found the optimal level of FT to be less than 1.0 cm. With the exception of one non-significant change in MS and seven non-significant changes in YG, performance for all traits at the OPT endpoint differed from performance at the FT endpoint ($P < 0.05$). These differences represent changes in

Table 6.4. Differences in trait medians for each sire genotype when progeny were fed to the optimum endpoint versus feeding to a 1.1 cm backfat thickness endpoint (Study 1)[a].

					Trait[b]				
Genotype[c]	DOF	outWT	CWT	FT	MS	LMA	QG[d]	YG	Profit
HHHH	52	46.98	31.34	0.63	0.34	20.61	0	-0.07[NS]	18.88
HHHL	48	55.36	36.66	0.51	0.22	18.21	0	-0.09	14.16
HHLH	16	33.54	20.89	0.81	1.12	16.02	1	0.22	48.41
HHLL	44	38.54	24.37	0.45	0.27	16.00	1	-0.06	17.97
HLHH	34	39.35	25.78	0.53	0.29	15.73	1	-0.04[NS]	32.77
HLHL	38	63.70	39.70	0.81	0.85	21.15	1	0.13	44.05
HLLH	66	66.04	43.30	0.86	0.14	24.99	1	0.00[NS]	29.22
HLLL	51	55.43	36.10	0.71	0.34	21.41	1	-0.04	32.67
LHHH	52	45.24	29.79	0.58	0.35	19.81	0	-0.07[NS]	21.94
LHHL	48	54.81	35.22	0.79	0.74	21.69	0	0.04	24.43
LHLH	21	28.33	17.97	0.71	0.95	15.71	1	0.10	36.99
LHLL	34	22.46	15.31	0.41	0.43	14.36	1	-0.07[NS]	20.44
LLHH	27	28.94	18.82	0.52	0.48	14.30	1	-0.04	30.02
LLHL	43	55.62	36.13	0.89	0.96[NS]	22.71	1	0.14	37.88
LLLH	61	55.86	37.46	0.71	0.12	22.23	1	-0.08	18.92
LLLL	51	43.15	27.83	0.65	0.35	21.08	1	-0.08[NS]	29.03
AVG[e]	53	49.86	32.63	0.73	0.18	21.71	0	-0.05[NS]	21.95

[a] Difference = value at optimum endpoint – value at FT constant endpoint; all differences were significant unless otherwise noted ($p < 0.05$)

[b] DOF = days on feed; outWT = live weight at harvest (kg); CWT = carcass weight (kg); FT = backfat thickness (cm); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); QG = quality grade; YG = yield grade

[c] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[d] A positive value is an increase in quality grade by the number of levels indicated

[e] Genotype with breed average EPD for each trait

[NS] Difference not significant (alpha = 0.05)

trait medians to levels that are optimum in this production system. Increases in all traits (except YG) are attributable to an increased number of days on feed, ranging from 16 to 61 days. Koontz et al. (2000) found similar increases in DOF, weight and FT when cattle were sorted into marketing groups for harvest at optimal endpoints. Increases in FT are generally associated with overall carcass fat and increases in low-yield carcasses prone to YG discounts, which can have a large impact on profit. In this case, the small changes in YG result from the increased muscling (LMA) that balances the increased amount of fat, minimizing the negative impact on profitability. The change in YG may actually be larger in reality as the model does not account for changes in percent kidney, pelvic and heart fat (a component of the yield grade equation) as the animal fattens and thus extended periods of time on feed may cause the YG to be somewhat underestimated. Differences in YG, though, did not seem to be an important contributor to differences in profit between the two endpoints in this study.

The largest differences between the two endpoints, due to increased days on feed, that likely impacted profit seem to be the increased weight and MS (and subsequent QG). Increases in live weight at harvest ranged from 22.46 to 66.04 kg and progeny of all sire genotypes were fed to an upper Choice quality grade. The combination of larger carcass weights, with similar yields, and QG premiums led to substantial increases in median profits. The difference in profit between the two endpoints is depicted in Figure 6.7 where profit for all genotypes increased at the OPT endpoint, a result also that was expected and reported by other researchers (Amer et al., 1994a; Basarab et al., 1999; Koontz et al., 2000; Pyatt et al., 2005b). The largest increase was experienced by the HHLH genotype and the smallest by the HHHL genotype. Progeny of the HHLH sire would be expected to have an unfavorable MS and favorable LMA (on average), while HHHL progeny would be expected to have the opposite. The HHLH sire, however, had the largest increase in MS, while the HHHL sire had one of the smallest, and both had average increases in LMA. These changes at the optimum endpoint reduced the performance and profit differences between the two sires and illustrate the difficulty in predicting changes in progeny performance at alternative harvest endpoints based on sire genotype.

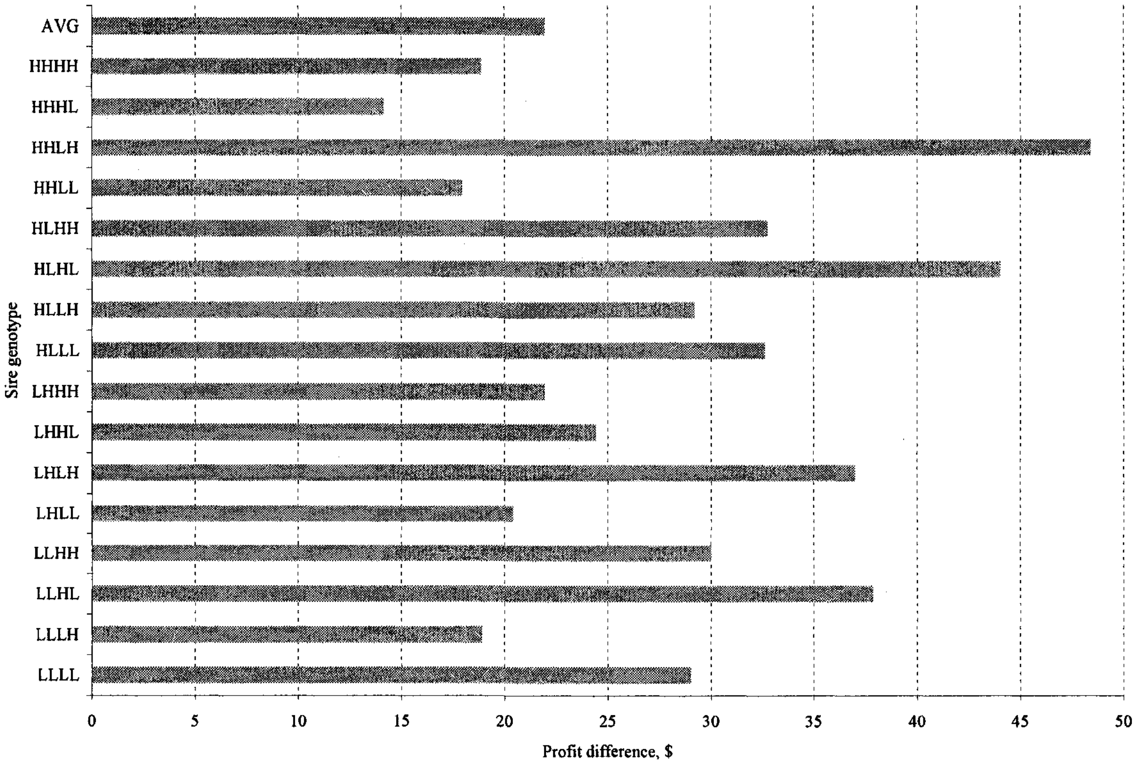

Figure 6.7. Difference in median profit for each sire genotype when progeny were fed to the optimum endpoint versus a backfat thickness constant endpoint. The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

Sire selection is complicated by the re-ranking that occurs when the finish endpoint changes. Sire re-ranking for median profit at each endpoint is shown in Figure 6.8 and supports the re-ranking at alternative endpoints reported by numerous studies (Lamb et al., 1992; Amer et al., 1994a; Cleveland, 2002). The plot shows the rank deviations when progeny were fed to an OPT versus a FT finish endpoint and depicts larger deviations with increasing profitability. Re-ranking was minimal for the four least profitable sire genotypes and increased to a maximum re-ranking of six places for HHLH. Re-ranking for the most profitable sires appears to decrease following the maximum re-ranking for sires in the intermediate profit range, but is still larger than in the lower profit range. This trend might suggest that the relative rank of certain sire genotypes is stable regardless of endpoint and, thus, could be predicted from a single endpoint, whereas the relative ranking of other genotypes is more endpoint dependent.

Sire genotypes with the largest increases in rank from the FT to the OPT endpoint correspond to the genotypes with the greatest profit difference between endpoints. Likewise, the genotypes with the largest decrease in rank were those that had the smallest profit difference between endpoints (Table 6.4). This suggests that some sire genotypes were closer to their optimum at the FT endpoint, while others were much farther away, but at the optimum they are more profitable. The differences between genotype profits are not static, however, indicating a genotype by management interaction. This interaction is evident in the re-ranking of the HHLH and HHHL genotypes, where the difference between median profit for each genotype at the OPT endpoint is much smaller than at the FT endpoint. Interactions complicate the ranking of sires for profit at

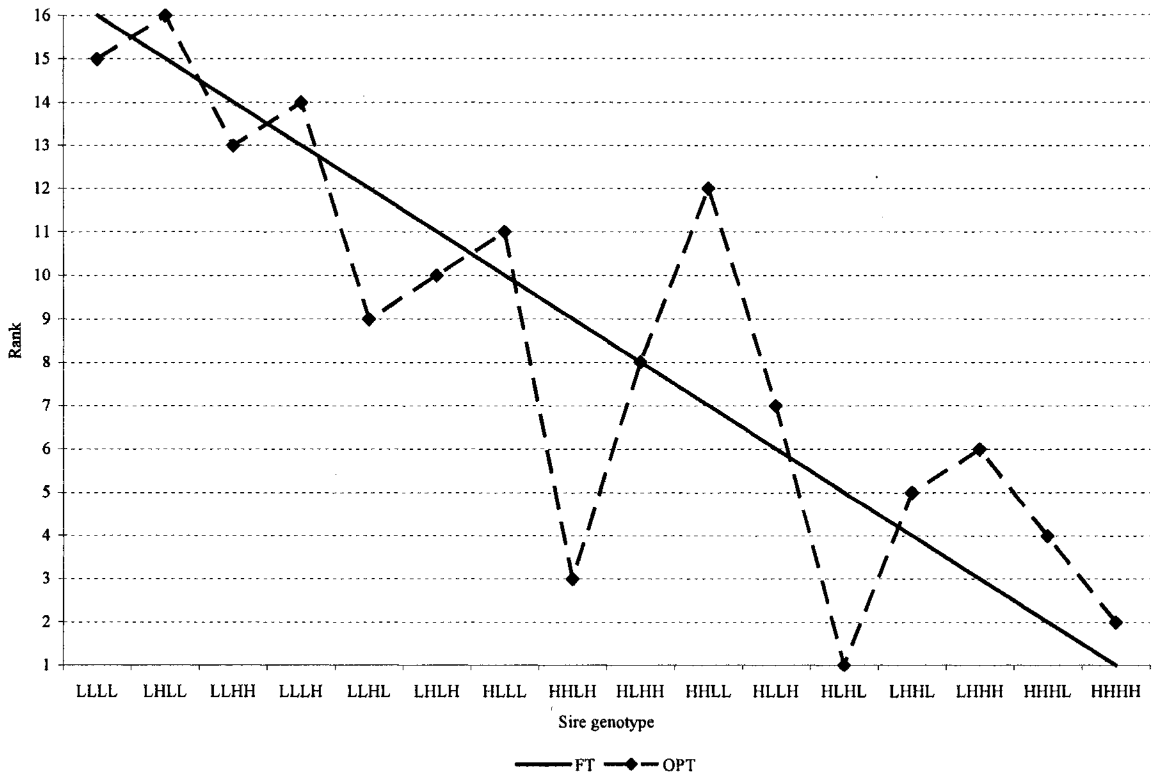

Figure 6.8. Sire genotype rank (1 = most profitable) for median profit at a 1.1 cm FT constant endpoint (FT) and at the optimum endpoint (OPT). The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

alternative harvest endpoints, but illustrate the importance of appropriately accounting for genetic differences.

Identifying genetic differences at the FT constant endpoint reduced potential negative profits for some genotypes and was expected to further decrease at the OPT endpoint. Figure 6.9 depicts the range of profit realizations for all sire genotypes at each endpoint. Expected profit for the AVG genotype ranged from \$-85.63 to \$165 for the OPT endpoint; a maximum value similar to all genotypes. Only one genotype (HHHH) had a minimum predicted profit that was more positive than AVG, indicating that most sire genotypes were capable of yielding profits that were less than would be expected from the breed average, while not yielding greater than expected maximum returns. At the OPT endpoint it appears that most genotypes confer more chance of having negative profits than expected, but an evaluation of profit distribution will determine if this is actually the case.

The profit comparison between the AVG and LHLL genotypes illustrates potential negative profit impact at the OPT endpoint. The LHLL genotype had the most negative minimum median profit (\$-210.90) and a plot of profit realizations (Figure 6.10) shows a distribution with a heavier negative tail than the distribution of AVG profit. The maximum predicted LHLL profit (\$156.53) was smaller than AVG, as was the profit median (\$29.10). Selection of this sire would be expected to yield a larger proportion of progeny with negative profits than would be expected considering the breed average. A comparison between the AVG and LLLL genotypes (Figure 6.11) shows two distributions that appear to contain equal proportions in the higher end of profit values. The LLLL sire, though, is similar to the LHLL genotype for median, minimum and

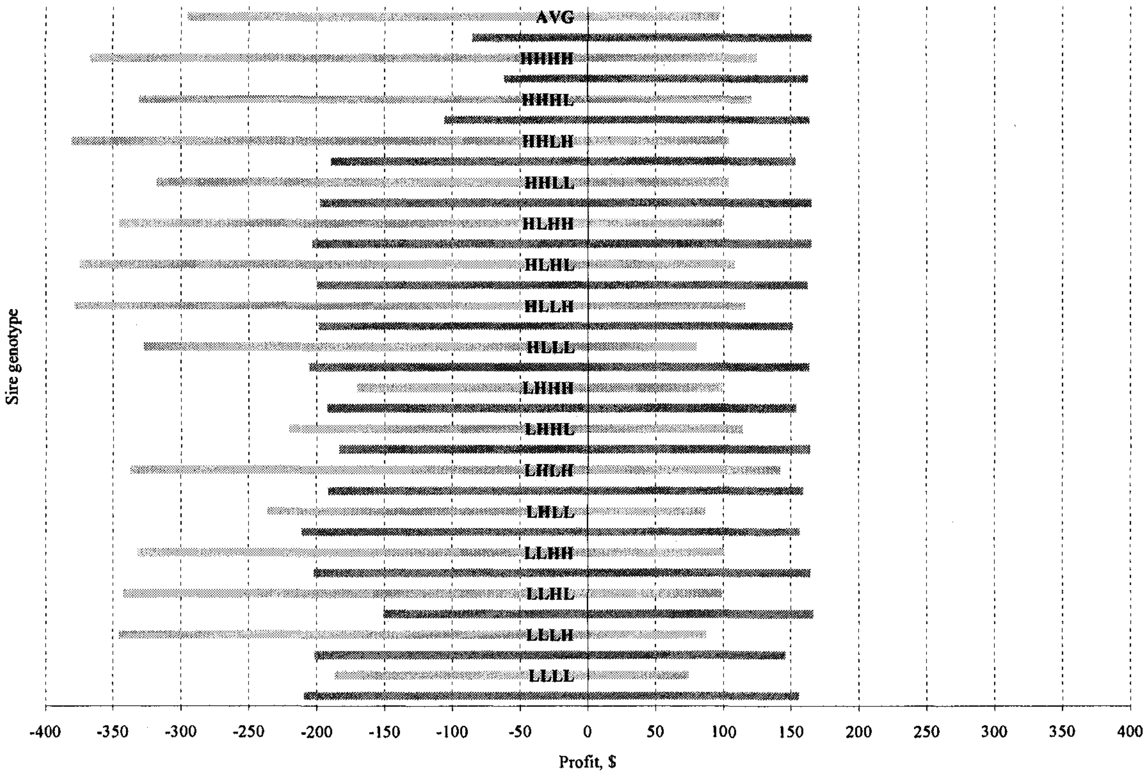

Figure 6.9. Range in profit realizations (n = 1000) for each sire genotype at a constant 1.1 cm backfat thickness (▒) and at the optimum finish endpoint (▨). The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

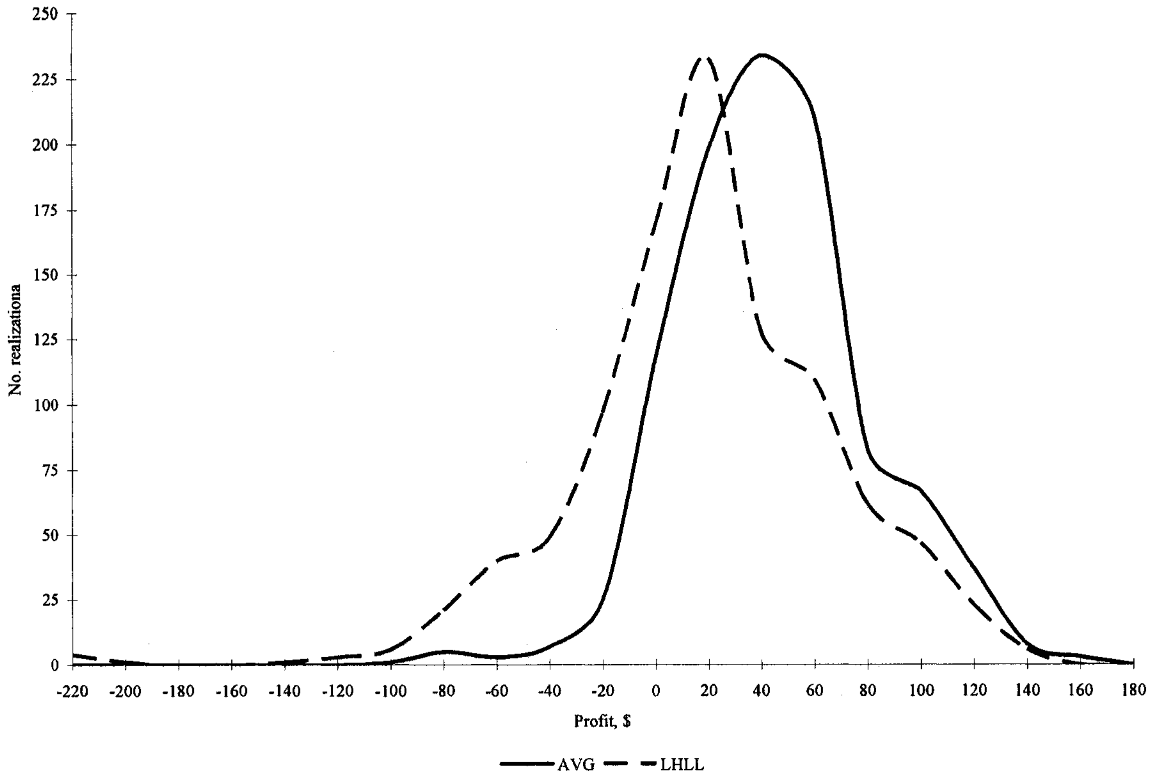

Figure 6.10. Distribution of profit realizations (n = 1000) for the AVG and LHLL sire genotypes at the optimum finish endpoint. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

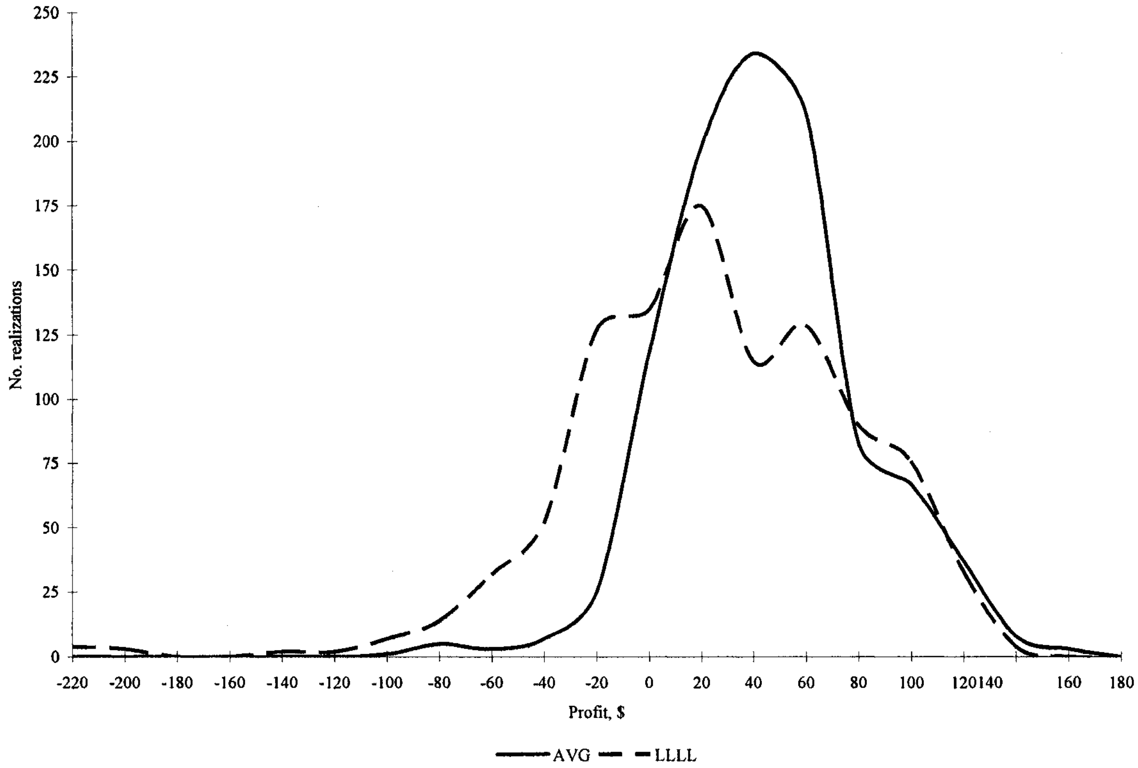

Figure 6.11. Distribution of profit realizations (n = 1000) for the AVG and LLLL sire genotypes at the optimum finish endpoint. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

maximum and confers a similar chance of yielding lower than expected profits. Two other genotype evaluations showed an apparent increase in potential negative profit over what would be expected indicating that, even at optimum endpoints, certain genotypes may have a larger impact on postweaning profitability.

Certain sire genotypes appear to impact profit more or less based on profit ranges only. The LHLH sire genotype had median and maximum profit values similar to the AVG genotype, but a minimum profit value that was much more negative ($-191.30) suggesting the possibility of progeny with large negative profits. A comparison of profit distributions (Figure 6.12), though, shows similar tails for AVG and LHLL genotypes indicating a similar chance of negative profit values. The HHHH genotype had a larger median, but a more positive minimum profit ($-61.39) than the AVG sire. The distribution of profit realizations (Figure 6.13) shows a heavier positive tail for the HHHH genotype, suggesting a greater proportion of positive profits, but similar negative tails, with a slightly larger proportion of negative profit values compared to the AVG genotype. Similarly, the HLHL genotype had the largest median profit ($74.33) and a larger proportion of progeny in the positive profit range, but the distribution of profits in the negative range was similar to the AVG genotype (Figure 6.14). At the OPT endpoint profit ranges and medians do not provide sufficient information to identify the risk of selection decisions.

The comparison of profit distributions at the OPT endpoint showed that some genotypes had more chance for negative profits, but more interestingly, no genotypes had less chance than was expected of the breed average animal, unlike finish at the FT constant endpoint. Accounting for genetic differences in this case does not seem to

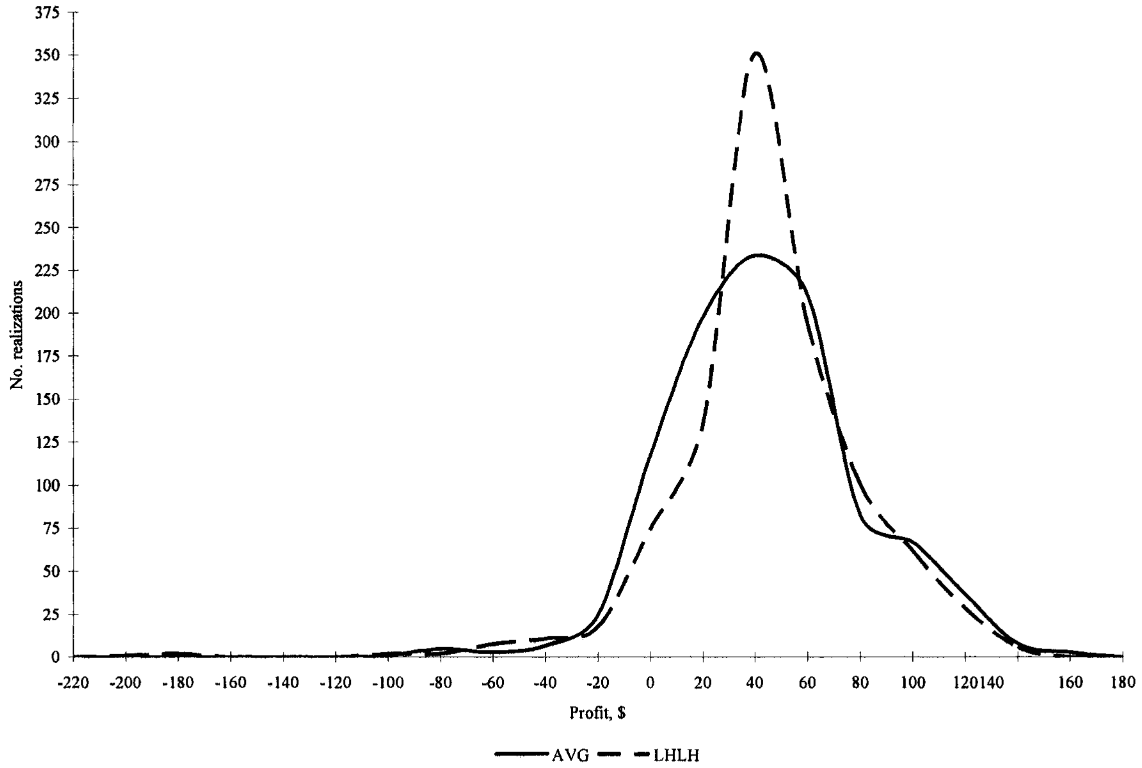

Figure 6.12. Distribution of profit realizations (n = 1000) for the AVG and LHLH sire genotypes at the optimum finish endpoint. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

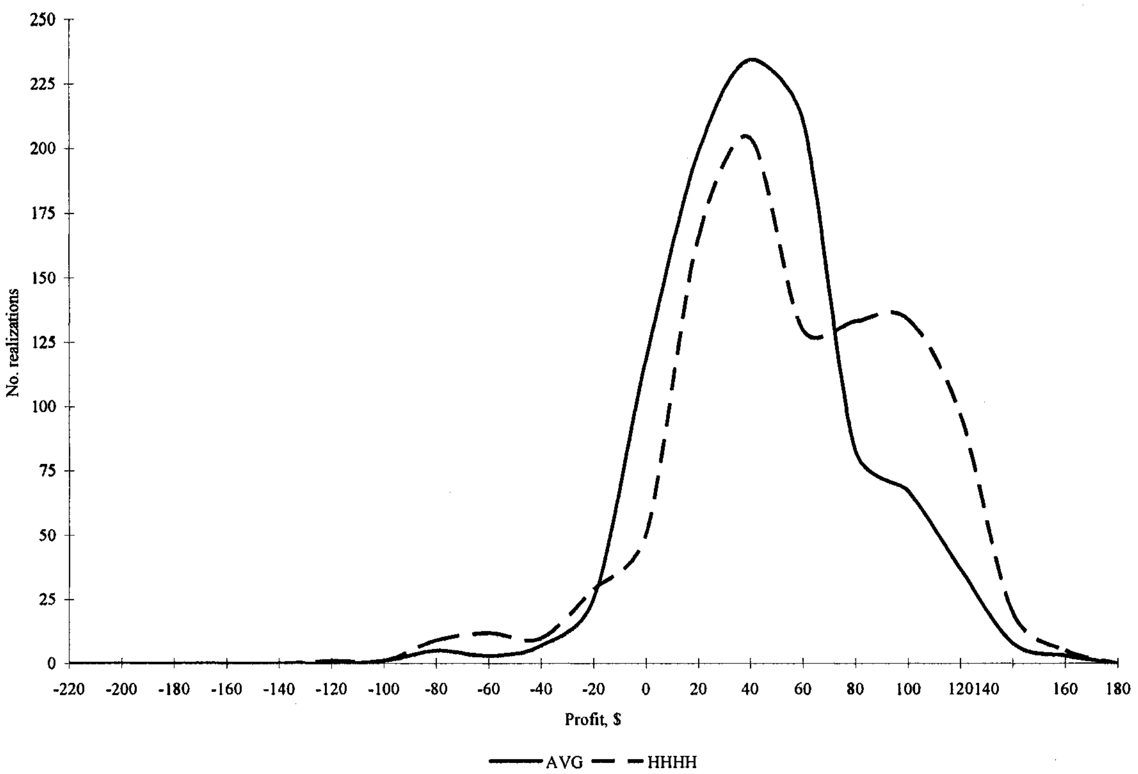

Figure 6.13. Distribution of profit realizations (n = 1000) for the AVG and HHHH sire genotypes at the optimum finish endpoint. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

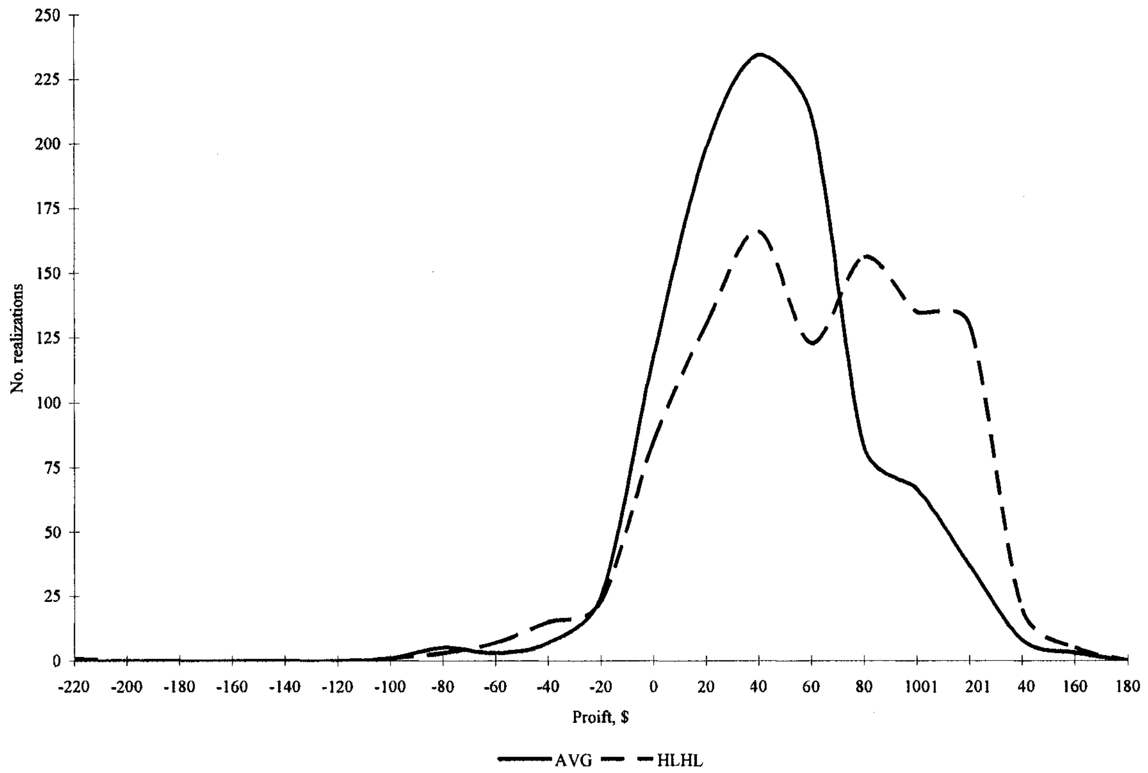

Figure 6.14. Distribution of profit realizations (n = 1000) for the AVG and HLHL sire genotypes at the optimum finish endpoint. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. AVG indicates a genotype with breed average EPD for each trait.

reduce probability of negative profits, but did create more opportunities for larger positive profits. At the FT endpoint larger medians or less negative minimum profit values could be inferred to reduce negative profit potential (Figures 6.3; 6.4), and smaller medians with more negative minimum profit to increase negative profit potential (Figure 6.2), but no such trends were evident at the OPT endpoint. It is possible that the optimization of profit also minimized this potential to a certain level for this production system, and that minimization was shared by all genotypes.

Feeding cattle to an optimum finish endpoint reduces the probability of negative profits compared to the FT constant endpoint. The range of predicted profit was smaller at the OPT endpoint for most genotypes and was shifted towards positive profit values (Figure 6.9). Predicted maximum profit realizations were somewhat larger ($146.07 to $166.72) and the range smaller than at the FT endpoint. Predicted minimum profits were smaller ($-61.39 to $-209.33), with a slightly smaller range, at the OPT endpoint for most, but not all, genotypes indicating that there was some increased negative profit potential at the OPT endpoint for certain genotypes. In general there was greater potential to avoid negative profit by feeding to the OPT endpoint.

The distribution of profit realizations at each endpoint for the HHLL genotype illustrates the reduction in negative profit potential by feeding to the OPT endpoint (Figure 6.15). The change in median profit from the FT to OPT endpoint for the HHLL sire was the smallest of all genotypes, but the distribution indicates a larger proportion of positive profit values at the OPT endpoint with similar proportions in the negative range at each endpoint. The largest change in median profit from the FT to the OPT endpoint was for the HHLH sire genotype. The distribution of profit realizations at each endpoint

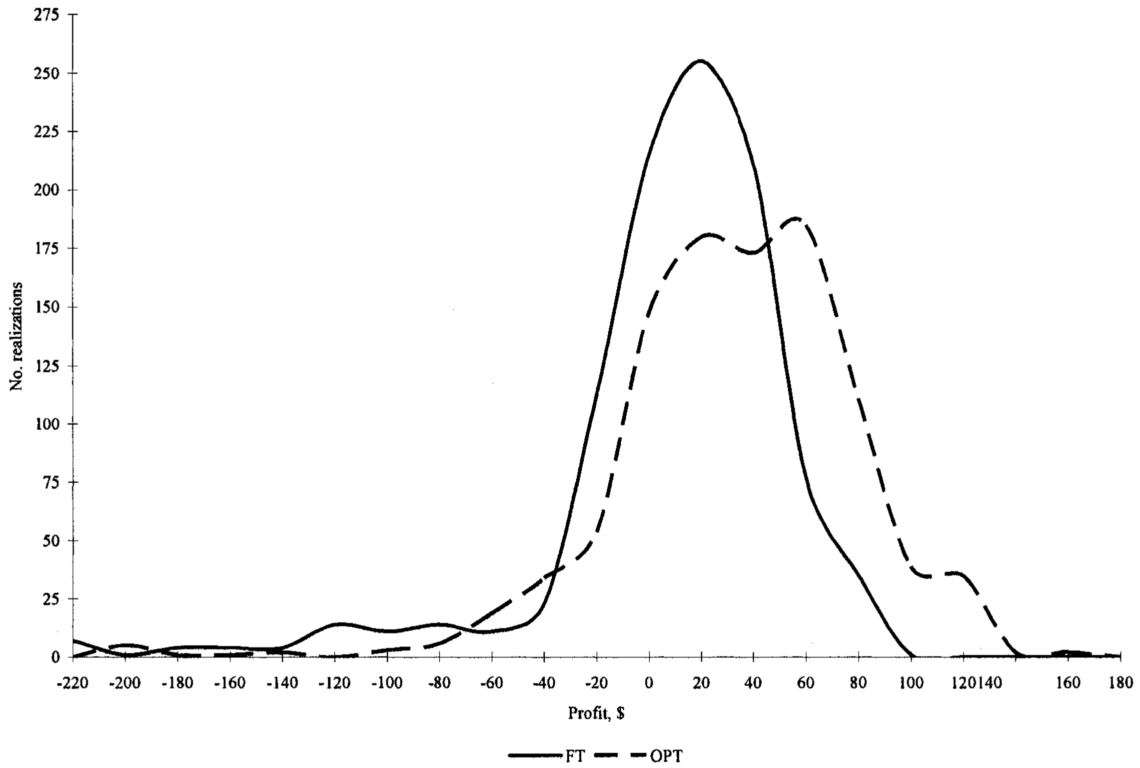

Figure 6.15. Distribution of profit realizations (n = 1000) for the HHLL sire genotype at a constant 1.1 cm backfat thickness (FT) and at the optimum finish endpoint (OPT). The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

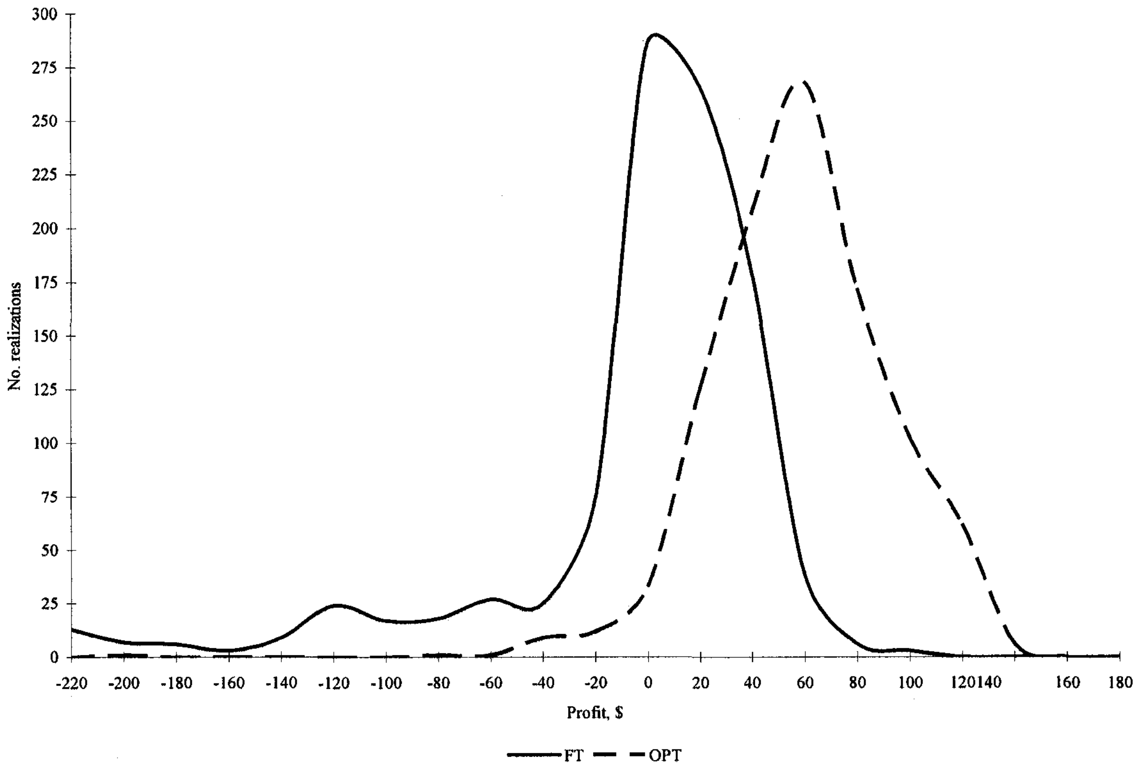

Figure 6.16. Distribution of profit realizations (n = 1000) for the HHLH sire genotype at a constant 1.1 cm backfat thickness (FT) and at the optimum finish endpoint (OPT). The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

(Figure 6.16) shows a much decreased potential for negative profitability and a larger proportion of profit in the positive range. Accounting for genetic differences at the OPT endpoint did not decrease negative profit potential in relation to breed average, but it was reduced compared to the FT endpoint and the identification of differences between genotypes allows for selection of sires that maximize profit.

The most profitable sires at the OPT endpoint are determined by the optimal combination of WT, FT, MS and LMA and their impact on costs and revenue. Selecting sires for this profit-maximizing combination remains difficult at the OPT endpoint, as it was at the FT constant endpoint. Figure 6.17 shows median profit plotted for each combination of EPD level, with the highest levels at the left and front. There is a discernable trend moving from high to low levels of WT and FT, but a peak was evident along the axis representing high MS and low LMA. In general, though, more high level EPD in a genotype equated to higher median profits. There exists more of a trend at the OPT endpoint than at the FT endpoint, but the plot is less than satisfying as support for an intuitive selection process for growth and carcass traits to maximize profit. These results highlight the difficulty in ranking sires for profit based on genetic potential for growth and carcass traits.

Traits important to increased profit were evaluated to identify any associations between profit and trait level at the OPT endpoint. Figure 6.18 depicts the percent change in postweaning performance for each sire genotype, ranked by increasing profit. Changes in most traits were small and not negative compared to the least profitable genotype. Backfat thickness was quite variable with increasing profit, while DOF varied

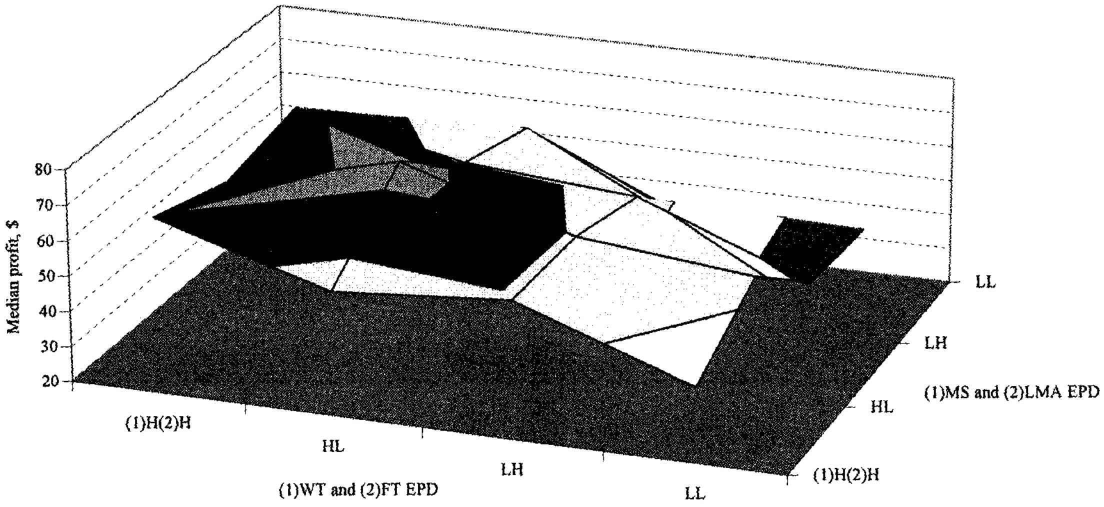

Figure 6.17. Surface plot of median profit at each EPD level, high (H) or low (L) for each trait; weight at a specified day on feed (WT); backfat thickness (FT); marbling score (MS) and longissimus muscle area (LMA) at the optimum finish endpoint.

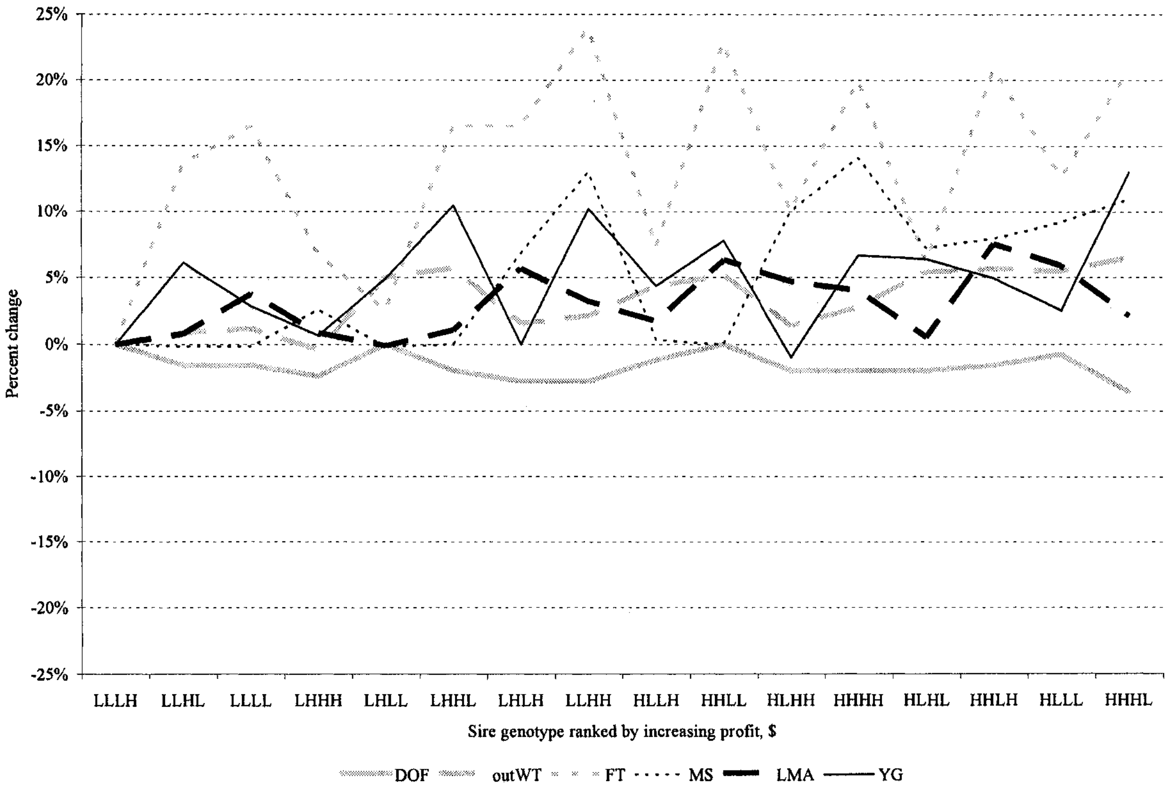

Figure 6.18. Percent change in median days on feed (DOF), live weight at harvest (outWT), backfat thickness (FT), marbling score (MS), longissimus muscle area (LMA) and yield grade (YG), for increasing median profit, at the optimum finish endpoint. The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

little. Increased profit appeared to be associated with overall increases in FT and MS, though MS seems to become less important, and a slight decrease in DOF. Yield grade was also increased in the most profitable sire, a function of increased fat with the extended days on feed and a decrease in LMA, due to the sire's genetic potential for muscling. Generally, outWT and LMA were increasing with profit. The difference between the least and most profitable sire appeared to be attributable to a large difference in FT, moderate differences in MS and YG, and small differences in outWT, LMA and DOF.

Coefficients for the correlation of simulated trait values with profit support the association of FT and MS with increased profit at the OPT endpoint (Table 6.5). Days on feed was minimally correlated with profit indicating DOF was not an important factor in profit maximization. Weight at harvest, LMA and YG were also highly positively correlated with profit, results that are not necessarily apparent in Figure 6.18, or, in the case of YG, intuitively obvious. A larger outWT is associated with more pounds of product, as is a larger LMA, increasing revenue as long as weights are below grid cutoffs. The relationship between YG and profit seems to be a function of the large increases in FT coupled with smaller increases in LMA causing YG to rise, but not to a level that is detrimental to profit. The changes in these traits to their optimal level, given the current production system, are surprising in light of research on maximizing feedlot profitability, especially considering the large increases in FT and DOF over optimum levels reported in other studies (Amer et al., 1994ab; Basarab et al., 1999).

Table 6.5. Correlation coefficients for the phenotypic correlation of simulated trait values with simulated profit for each sire genotype, in ascending order of median profit, and for the AVG sire genotype, at the optimum finish endpoint.

	Trait[a]					
Genotype[b]	DOF	outWT	FT	MS	LMA	YG
LHLL	0.19	0.76	0.80	0.82	0.71	0.82
LLLL	0.16	0.77	0.81	0.80	0.75	0.88
LLLH	0.16	0.75	0.76	0.84	0.70	0.83
LLHH	0.30	0.80	0.82	0.80	0.79	0.85
HHLL	0.02	0.70	0.71	0.79	0.60	0.80
HLLL	-0.08	0.70	0.76	0.80	0.65	0.88
LHLH	0.36	0.76	0.81	0.79	0.78	0.83
LLHL	0.31	0.84	0.81	0.76	0.78	0.91
HLHH	0.04	0.77	0.76	0.76	0.66	0.88
HLLH	0.02	0.74	0.72	0.83	0.65	0.82
LHHH	0.28	0.79	0.79	0.78	0.74	0.81
LHHL	0.45	0.87	0.79	0.74	0.80	0.86
HHHL	0.15	0.82	0.76	0.75	0.71	0.87
HHLH	0.15	0.79	0.76	0.77	0.70	0.87
HHHH	0.23	0.81	0.76	0.75	0.73	0.82
HLHL	0.23	0.85	0.75	0.71	0.73	0.91
AVG[c]	0.08	0.79	0.78	0.82	0.71	0.88

[a] DOF = days on feed; outWT = live weight at harvest (kg); FT = backfat thickness (cm); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); YG = yield grade

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[c] Genotype with breed average EPD for each trait

Factors important to increased profit at the optimum endpoint seemed to differ from those important at the FT constant endpoint. There was less apparent variability between genotypes in traits at the OPT endpoint, with the exception of outWT and MS, which had similar patterns of change. Days on feed was a notable difference as it appeared to be quite important to profit at the FT endpoint (Figure 6.6; Table 6.2) and not as important at the OPT endpoint. The difference in DOF between least and most profitable genotypes at each endpoint, though, was small, indicating that maximizing profit was more a function of DOF optimally combined with the other traits. At the FT endpoint DOF may have worked against sire profit, while it was not a factor at the OPT endpoint. With the exception of FT and YG, the most profitable genotype at each endpoint had similar changes for all traits. The trait with the largest fluctuations at the OPT endpoint was FT, where a large increase in FT was attributed to the most profitable sire. Backfat thickness appeared to be more important at the OPT endpoint (Tables 6.2; 6.5). Pyatt et al. (2005) reported that weight became less important with increased days on feed, but this study found weight to be more important to profit based on the large increase in correlation coefficients at the OPT endpoints (Table 6.5). These differences suggest that a re-ranking of factor importance occurs at alternative endpoints making a prediction of the most profitable combination of trait levels for a particular endpoint difficult.

Study 3

The HHHH sire genotype was most profitable at the FT constant endpoint (Table 6.1) and was chosen for the base genotype on which to reference the simulation of genetic improvement. Higher profits were associated with increased weight, MS and LMA and decreased FT at the FT endpoint and it was assumed that continued selection for these traits to improve profitability was reasonable in the absence of other selection criteria. Results from previous sections led to the conclusion that intuitive selection processes to optimize profit at a FT or optimum endpoint were not apparent considering the available genotypes. The genotypes in this study represent possible future sires where selection has focused on the aforementioned growth and carcass traits and resulted in changes in these traits in the desired direction or the selection of sires with desired trait levels. These genotypes are used to assess future profitability when feeding progeny to a 1.1 cm FT constant endpoint, the HHHH sire genotype optimum backfat thickness endpoint (1.74 cm FT) or to individual optimum endpoints.

Simulated feedlot and harvest trait medians at each of the three endpoints for each sire genotype are listed in Table 6.6. Median profit for successive years at the 1.1 cm FT endpoint ranged from \$43.19 to \$66.64 for 203 to 236 days on feed. All profits differed from the HHHH genotype ($p < 0.01$), where median profit declined in the first year of selection but subsequently increased and seemed to stabilize in the last three years. At the 1.74 cm FT endpoint profit ranged from \$71.83 to \$88.32 for 242 to 269 days on feed. A similar profit trend occurred in that profit declined in the first years before increasing in the last two years, over HHHH. Only one median profit was less than HHHH ($P < 0.01$), but it occurred in year three of selection. Profit at the optimum endpoint ranged from \$74.44 to \$96.94 for 258 to 264 days on feed. In this case profit increased in the first year, declined for the two subsequent years, before increasing in the fourth. None of the profits at the optimum endpoint were less than the HHHH genotype indicating that selection improved profitability in each case.

At each of the endpoints the HHHH4 sire genotype was most profitable, though differences were small when compared to the HHHH5 sire. This result indicates that the HHHH4 sire may have the optimal combination of traits levels and is most appropriate for alternative endpoints in this production system. There is no knowledge of selection beyond the fifth year, but it is possible that profit has been optimized in the fourth year and will not increase due to subsequent selection, though profit has been improved over HHHH.

Table 6.6. Simulated trait medians for the HHHH sire genotype accounting for genetic improvements due to growth and carcass trait selection when progeny were fed to a constant 1.1 cm backfat thickness, the median optimum backfat thickness endpoint identified for the HHHH sire in Study 2 (1.74 cm) and the optimum endpoint for each successive year of selection.

	Trait[a]								
Genotype[b]	DOF	outWT	CWT	FT	MS	LMA	QG	YG	Profit
	1.1 cm backfat thickness endpoint								
HHHH	208	611.93	365.94	-	6.28	80.14	CH+	3.23	50.78
HHHH1[c]	231	629.70	378.15	-	5.97	84.66	CH	3.11	43.19[x]
HHHH2	220	620.99	372.03	-	6.17	83.10	CH+	3.14	54.26[x]
HHHH3	203	615.94	368.14	-	6.53	80.91	CH+	3.22	63.70[x]
HHHH4	228	632.99	379.77	-	6.16	85.50	CH+	3.14	66.64[x]
HHHH5	236	646.09	387.00	-	6.12	87.75	CH+	3.03	65.59[x]
	1.74 cm backfat thickness endpoint								
HHHH	260	658.91	397.28	-	6.62	100.75	CH+	3.16	69.66
HHHH1	242	653.59	392.59	-	7.00	98.06	CH+	3.21	72.07
HHHH2	269	663.99	399.79	-	6.50	102.43	CH+	3.10	71.83[x]
HHHH3	265	659.87	397.35	-	6.62	102.11	CH+	3.06	62.48[x]
HHHH4	246	669.60	402.20	-	7.09	99.89	CH+	3.20	88.32[x]
HHHH5	256	674.62	406.28	-	6.92	101.54	CH+	3.11	85.31[x]
	Individual optimum endpoint								
HHHH	260	658.91	397.28	1.74	6.62	100.75	CH+	3.16	69.66
HHHH1	258	674.77	406.94	1.90	6.95	102.76	CH+	3.30	81.84[x]
HHHH2	263	670.44	404.35	1.65	6.36	100.63	CH+	3.10	79.01[x]
HHHH3	260	663.31	399.90	1.43	6.18	96.97	CH+	2.99	74.44[x]
HHHH4	259	686.19	413.66	1.91	7.10	104.40	CH+	3.29	96.94[x]
HHHH5	264	677.11	408.26	1.42	6.29	99.54	CH+	2.99	88.42[x]

[a] DOF = days on feed; outWT = live weight at harvest (kg); CWT = carcass weight (kg); FT = backfat thickness (cm); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); QG = quality grade (CH = Choice; CH+ = upper 2/3 Choice); YG = yield grade

[b] The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[c] The number indicates the year of selection past HHHH

[x] Differs from the HHHH genotype median profit, within endpoint ($P < 0.01$)

Sire performance and predicted median profit were similar at each of the optimum endpoints (1.74 cm FT and individual optimum). Days on feed were also similar and more than DOF at the 1.1 cm FT endpoint. This suggests that the optimum endpoint does not differ markedly as genetic levels change due to selection for improvements in growth and carcass traits. An identified optimum endpoint, therefore, may be appropriate for successive generations with only small changes in the expected median profit.

Selection for growth and carcass traits appears to have improved profitability at each endpoint, but profit decreases would need to be absorbed before reaching the optimal profit level. Continued selection does not appear to maintain optimum profitability, assuming a static market, and may increase the risk associated with negative profits. Figure 6.19 shows the distribution of profit realizations for the HHHH and HHHH5 sire genotypes at a 1.1 cm FT constant endpoint. Similar distributions are evident, but the HHHH5 genotype distribution appears to have a heavier negative profit tail indicating a higher probability of negative profits than are expected of the HHHH genotype. The flatter HHHH5 distribution also yields a greater proportion of values in the upper profit range. Selection for growth and carcass traits increased the median profit and probability of large profits, but also increased negative profit potential at the 1.1 cm FT endpoint. This increase in negative potential, however, was only evident after five years of selection (results not shown) suggesting that optimal selection practices at this endpoint may increase profitability without increases in negative profit potential.

The distribution of profit realizations for the HHHH and HHHH5 genotypes at the HHHH optimum endpoint (1.74 cm FT) is depicted in Figure 6.20. Selection appears to have narrowed the distribution of profits compared to the HHHH genotype and generally

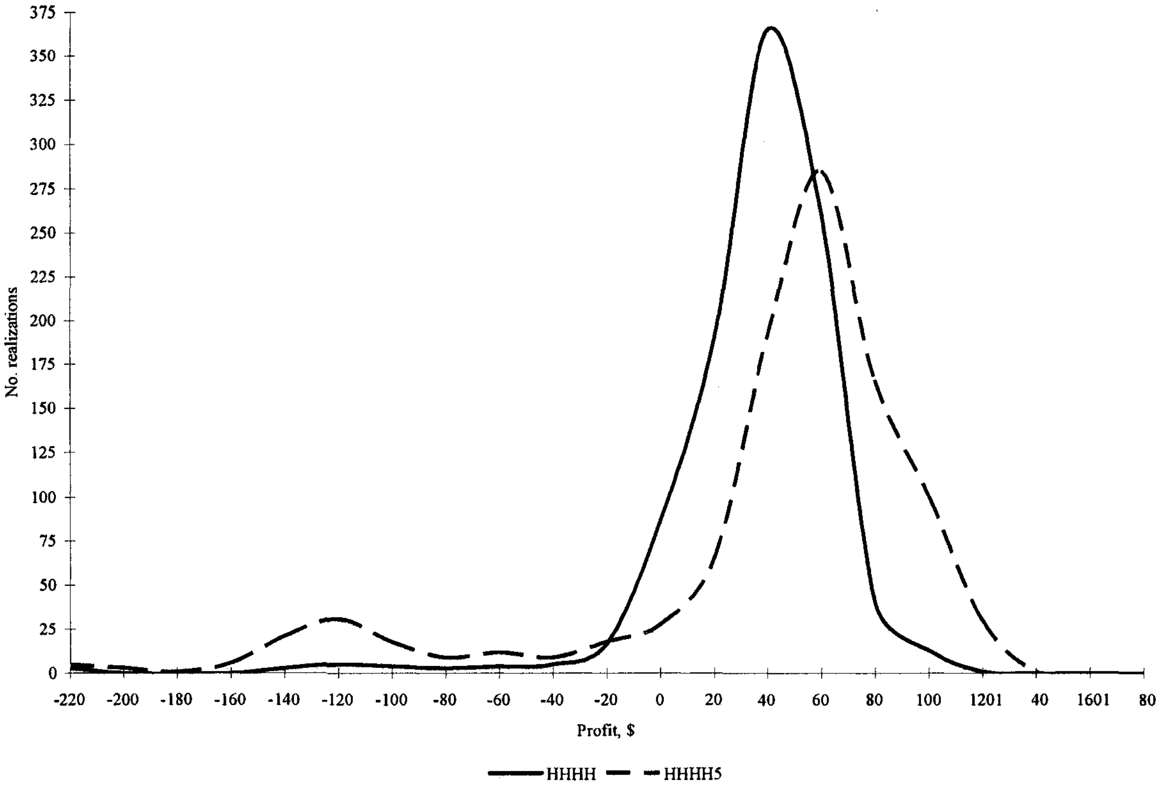

Figure 6.19. Distribution of profit realizations (n = 1000) for the HHHH and HHHH5 sire genotypes at a 1.1 cm constant backfat thickness, where HHHH5 represents maximal genetic improvement for growth and carcass traits. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

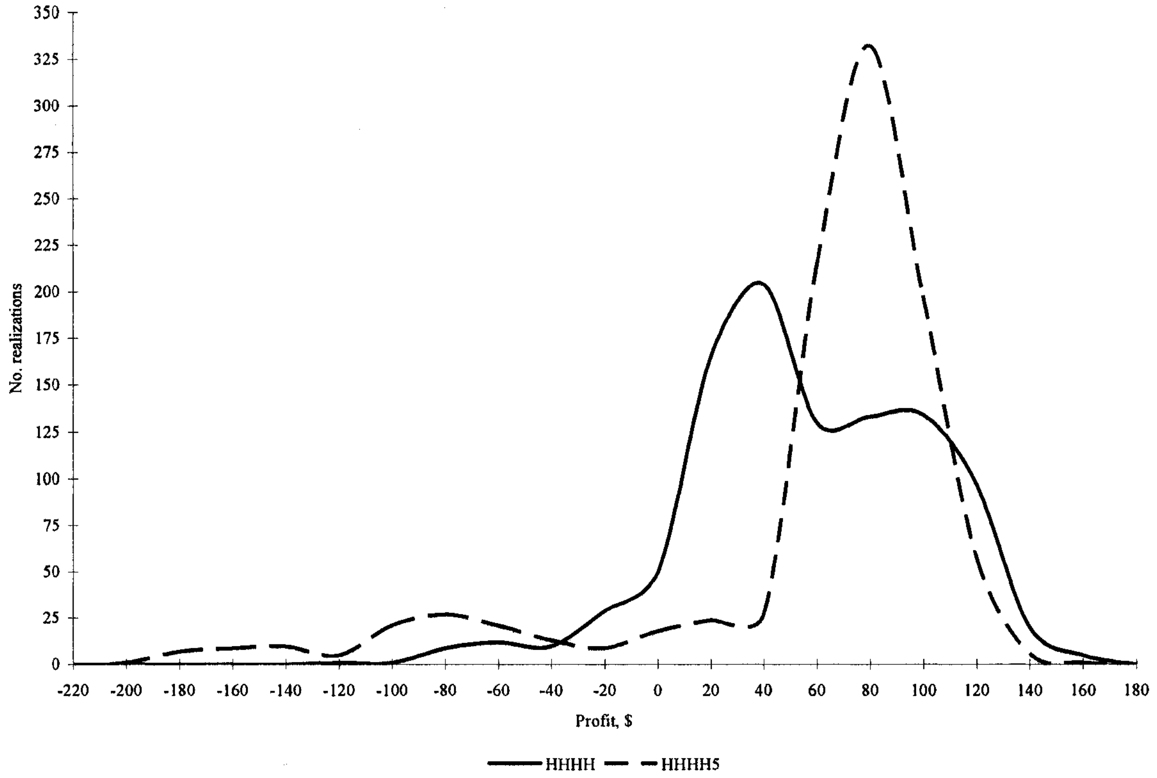

Figure 6.20. Distribution of profit realizations (n = 1000) for the HHHH and HHHH5 sire genotypes at the HHHH optimum finish endpoint (1.74 cm backfat thickness), where HHHH5 represents maximal genetic improvement for growth and carcass traits. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

reduced the chance of negative profit as more values were in the higher profit range, but also reduced the proportion of values in the extreme positive range. There is a reduction in negative profit potential, but there was a higher probability of extreme negative profit values imparted by the HHHH5 genotype. Such large negative profits, though rare, could offset larger profit values and reduce the benefits of selecting for this sire genotype at a constant 1.74 cm FT endpoint. Distributions for the previous four years were similar to Figure 6.20 (results not shown) as each successive year of selection yielded profits in the extreme negative profit range, while increasing profits in the moderate range.

Accounting for individual optimum endpoints concurrent with selection appears to improve profitability and reduce negative profit potential. The distribution of profit realizations for the HHHH and HHHH5 genotypes at the individual optimum endpoint depicts a narrowing for HHHH5 compared to HHHH (Figure 6.21). The HHHH5 distribution falls within the HHHH distribution with similar proportions of values in the extreme positive range. The probability of values in the extreme negative range appear to be smaller for the HHHH5 genotype, compared to HHHH, and much smaller than the probability at the constant 1.74 cm FT endpoint (Figure 6.20). Each of the previous year's genotypes had similar distributions and reduced negative profits compared to HHHH (results not shown). The chance for negative profits appears to increase with continued selection considering previously determined constant endpoints, but the chance can be reduced by accounting for selection when identifying finish endpoints.

Changes in all traits that are the target of selection may not necessarily be associated with changes in profitability. To evaluate any relationship between trait levels and profit the percent change in median postweaning performance for each sire genotype

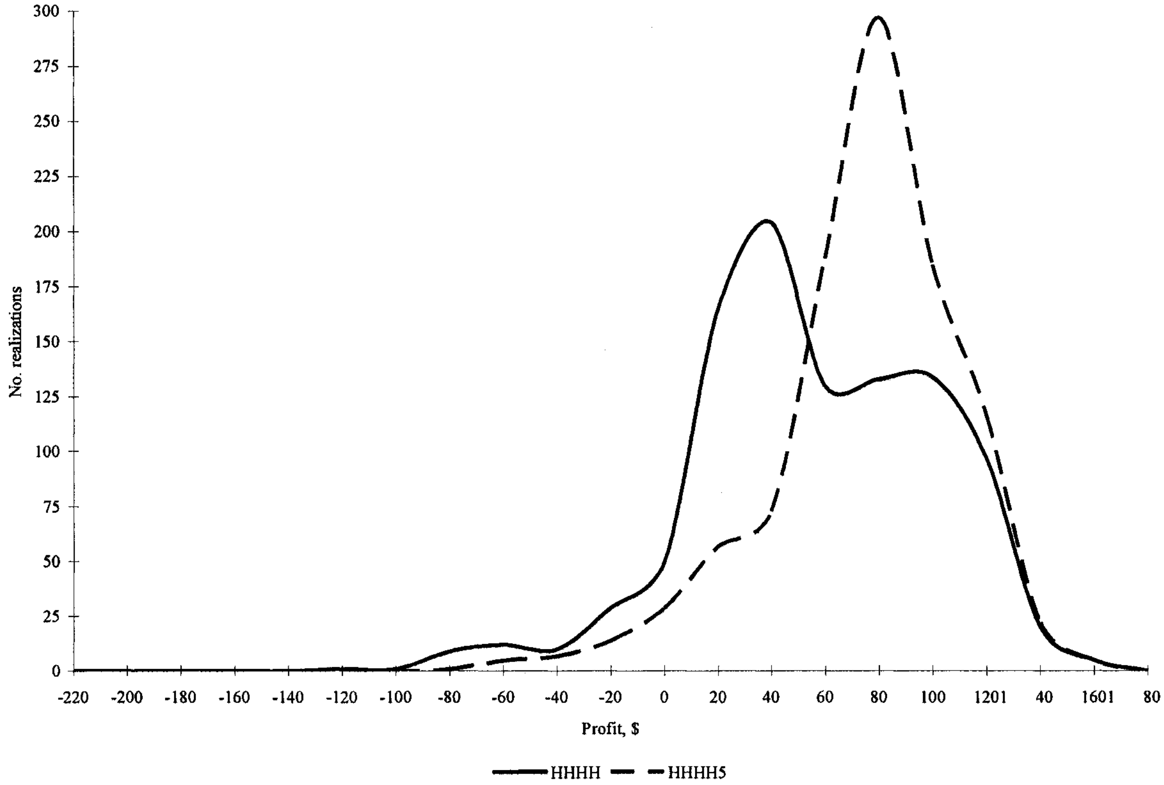

Figure 6.21. Distribution of profit realizations (n = 1000) for the HHHH and HHHH5 sire genotypes at the individual optimum finish endpoint, where HHHH5 represents maximal genetic improvement for growth and carcass traits. The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA.

was plotted with profit (Figure 6.22). Changes in DOF, outWT and LMA were generally small and non-negative, while changes in MS and YG were generally negative when compared to the HHHH genotype. Increased profit appeared to be associated with small changes in all traits (except MS) coupled with a peak increase in MS, or increases in DOF, outWT and LMA and decreases in MS and YG. The decrease in MS for the most profitable sire (HHHH4) was not enough to impact revenue, as QG was not affected (Table 6.6), and the less-than-maximum DOF and increases in outWT appears to have contributed to increased profit. Though carcass traits changed with selection, trait levels for sires in the second through fifth year were similar and thus differences in quality and yield characteristics did not seem to be important. Optimal DOF and increased outWT were most the most important factors for increases in profit with selection at the 1.1 cm FT endpoint.

Factors important to profitability at the HHHH optimum endpoint appear, at first glance, to differ from those at the 1.1 cm FT endpoint, accounting for selection. Figure 6.23 shows the percent change in median performance and profit for each genotype. Changes in trait levels were small and, with the exception of DOF and YG, all were non-negative relative to the HHHH sire. Profit optimization (HHHH4) was associated with increases in MS, outWT and YG and a decrease in DOF. The most important trait combination appears to be the peak change in MS coupled with a near-peak change in outWT and reduction in DOF. An increase in YG would normally be associated with reductions in revenue, and thus profit, but carcass and yield characteristics are similar so that small changes do not impact profit. Marbling score appears to be important

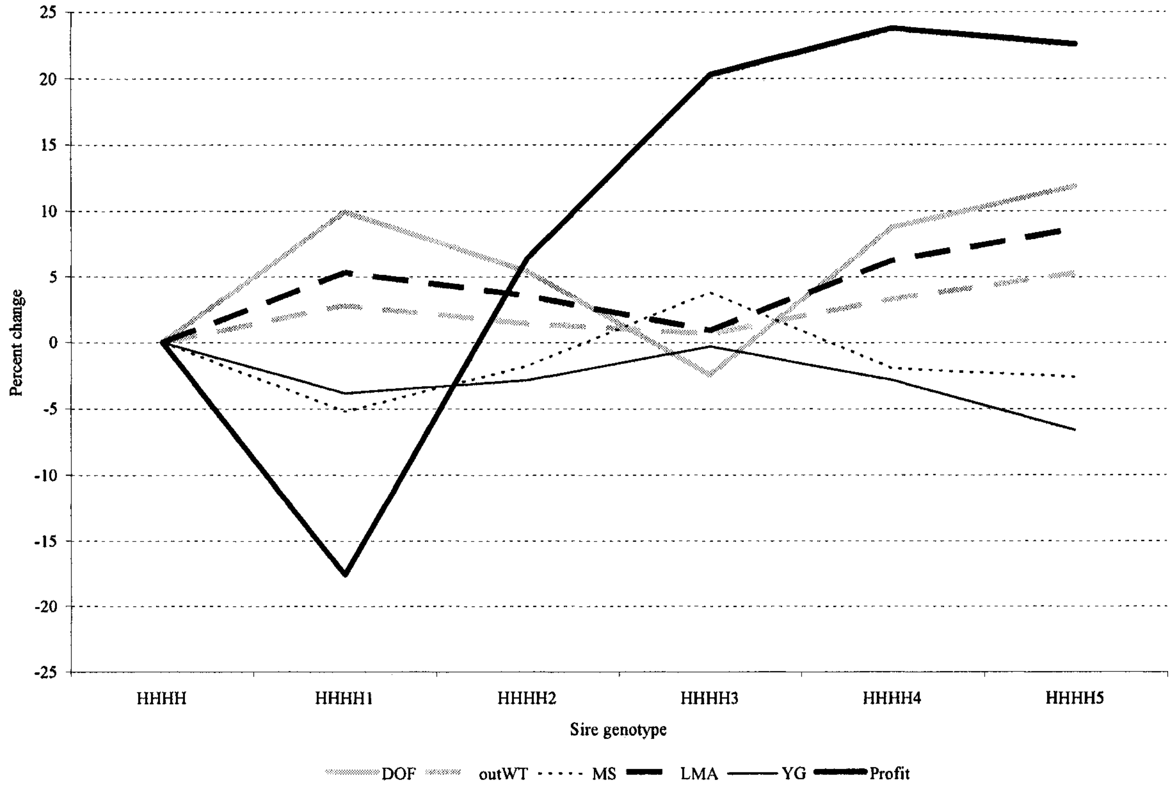

Figure 6.22. Percent change in median days on feed (DOF), live weight at harvest (outWT), backfat thickness (FT), marbling score (MS), longissimus muscle area (LMA), yield grade (YG) and profit at a 1.1 cm constant FT endpoint. The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. Number suffixes on genotypes indicate increasing genetic improvement for growth and carcass traits (outWT; FT; MS; LMA).

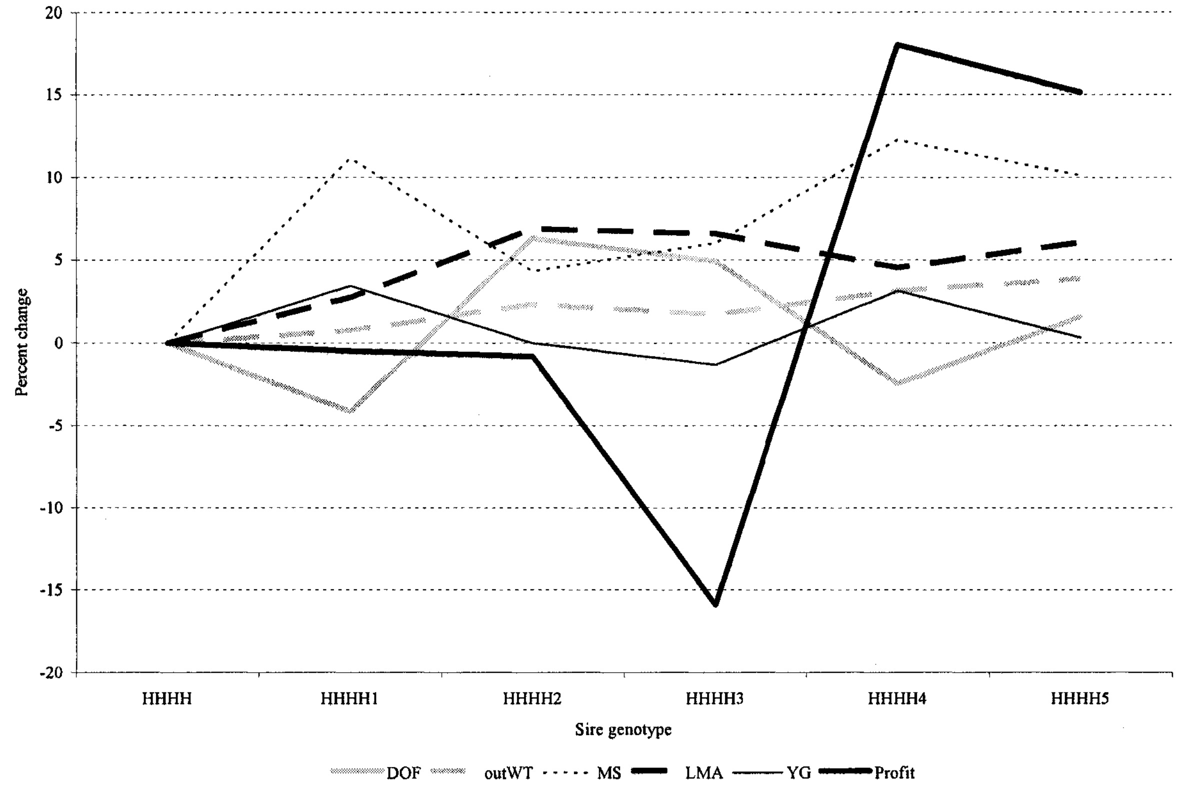

Figure 6.23. Percent change in median days on feed (DOF), live weight at harvest (outWT), backfat thickness (FT), marbling score (MS), longissimus muscle area (LMA), yield grade (YG) and profit at the HHHH optimum endpoint (1.74 cm FT). The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. Number suffixes on genotypes indicate increasing genetic improvement for growth and carcass traits (outWT; FT; MS; LMA).

to increased profit in this system, but the increase does not have any practical consequences as all sire genotypes had a high Choice median QG (Table 6.6) and so profit was not affected. Like the 1.1 cm FT endpoint, DOF and outWT seem to be the most important factors associated with profit at the 1.74 cm FT endpoint.

Changes in trait levels at the individual optimum endpoint were more orderly than at each of the previous endpoints. Figure 6.24 shows the percent change in median performance and profit for each sire genotype. All traits, with the exception of DOF, changed in the general direction, but not magnitude, of profit. The most profitable genotype (HHHH4) was associated with peak changes in outWT, FT, MS and LMA compared to the HHHH sire. Days on feed remained fairly stable but did decrease for the optimum profit genotype. Large increases in FT were offset by similar increases in LMA, avoiding YG discounts, while changes in MS made no improvements to profit in terms of QG premiums. The factors most important to profit at the individual optimum endpoint appear to be increases in LMA, to offset FT, increases in outWT and a decrease in DOF.

Correlation coefficients for the correlation of simulated trait values with profit generally support the associations discussed for each of the endpoints, accounting for selection (Table 6.7). Decreases in DOF were associated with increased profit at the FT constant endpoints, but the two traits had little association at the optimum endpoint. All other traits were positively associated with profit at the optimum endpoint, differing from the FT endpoints for outWT and LMA. The increases in FT were assumed to be mostly offset by the increases in LMA at the optimum endpoint, with the increases in YG

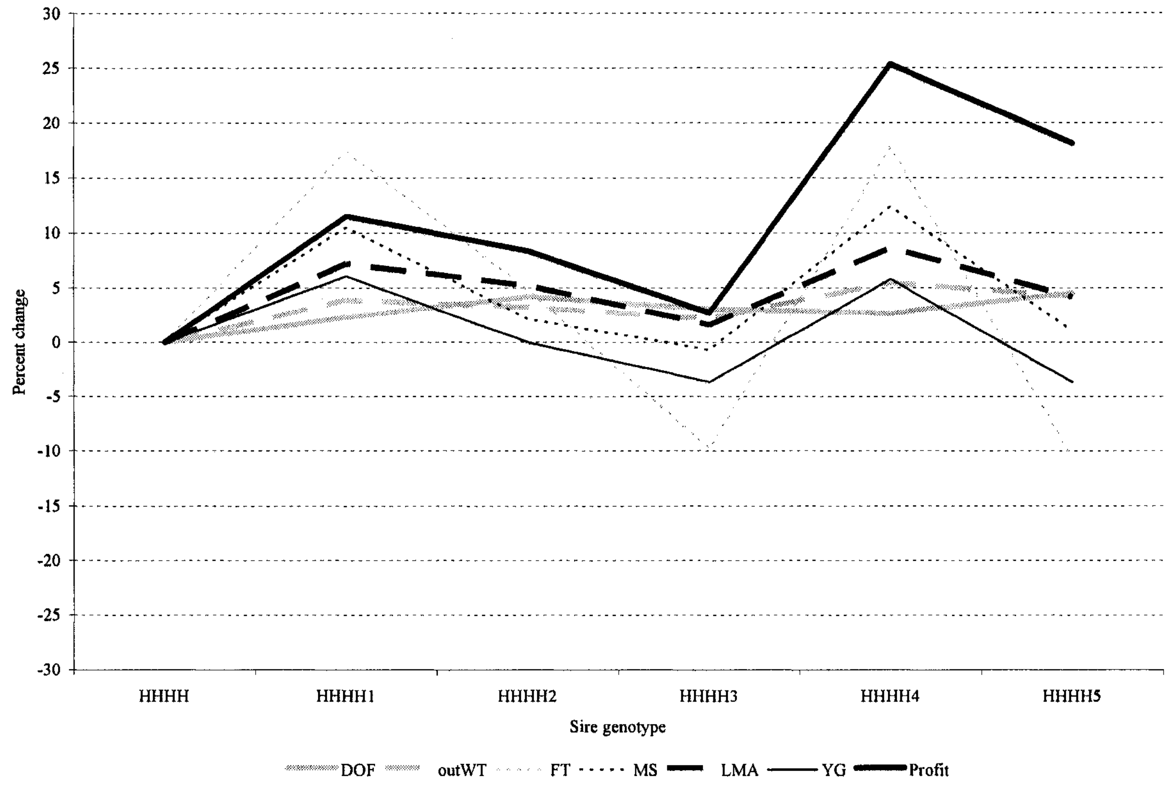

Figure 6.24. Percent change in median days on feed (DOF), live weight at harvest (outWT), backfat thickness (FT), marbling score (MS), longissimus muscle area (LMA), yield grade (YG) and profit at the individual optimum endpoint. The first letter (H or L) = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA. Number suffixes on genotypes indicate increasing genetic improvement for growth and carcass traits (outWT, FT, MS, LMA).

Table 6.7. Correlation coefficients for the phenotypic correlation of simulated trait values with simulated profit for the HHHH genotype accounting for selection when progeny were fed to a constant 1.1 cm backfat thickness, the median optimum finish backfat thickness endpoint for the HHHH sire in Study 2 (1.74 cm) and the optimum endpoint for each successive year of selection.

	Trait[a]					
Genotype[b]	DOF	outWT	FT	MS	LMA	YG
	1.1 cm backfat thickness endpoint					
HHHH	-0.61	0.06	0.55	0.55	-0.63	0.83
HHHH1[c]	-0.59	0.11	0.42	0.48	-0.61	0.78
HHHH2	-0.67	-0.14	0.54	0.53	-0.72	0.81
HHHH3	-0.40	0.23	0.57	0.26	-0.50	0.85
HHHH4	-0.60	-0.24	0.32	0.49	-0.64	0.70
HHHH5	-0.69	-0.23	0.50	0.59	-0.71	0.74
	1.74 cm backfat thickness endpoint					
HHHH	0.23	0.81	0.76	0.75	0.73	0.82
HHHH1[c]	-0.68	0.07	0.47	0.64	-0.65	0.64
HHHH2	-0.66	-0.15	0.33	0.61	-0.66	0.58
HHHH3	-0.67	-0.19	0.40	0.63	-0.66	0.50
HHHH4	-0.66	-0.08	0.38	0.61	-0.65	0.56
HHHH5	-0.65	-0.32	0.37	0.60	-0.65	0.56
	Individual optimum endpoint					
HHHH	0.23	0.81	0.76	0.75	0.73	0.82
HHHH1[c]	0.21	0.82	0.67	0.66	0.64	0.83
HHHH2	0.17	0.79	0.75	0.77	0.71	0.80
HHHH3	0.00	0.76	0.73	0.73	0.62	0.82
HHHH4	0.04	0.78	0.63	0.69	0.59	0.76
HHHH5	0.03	0.75	0.67	0.70	0.58	0.76

[a] DOF = days on feed; outWT = live weight at harvest (kg); FT = backfat thickness (cm); MS = marbling score (5.0 = small); LMA = longissimus muscle area (cm^2); YG = yield grade

[b] The first letter = the weight EPD at a specified day on feed; the second letter = FT; the third letter = MS; the fourth letter = LMA

[c] The number indicates the year of selection past HHHH

not detrimental to profitability. Associations between performance and profit exist, but many are not of practical importance to changes in profitability. The relative importance of factors at each of the endpoints does not appear to be affected in any meaningful way by changes in genetic levels due to selection.

Application of model output

The postweaning decision support system (DSS) was designed to assist producers in ranking sires for selection when progeny are fed to a constant endpoint or to the endpoint that maximizes individual profitability. The goal of such a system should be to provide users with easy-to-understand, likely phenotypic outcomes (or ranges of outcomes) that properly account for genetic differences for growth and carcass traits and allow the user to compare sires in terms of profitability at harvest, given a specified production system. Additionally, supplying ranges (or some indication of data spread) for stochastically simulated performance allows decision-making that accounts for the risk-aversion or acceptance of a particular user. The DSS presented in this research is intended as one tool (focusing on the postweaning phase) in the decision-making process encompassing the entire production system.

The DSS output could be presented in any number of ways to rank sires for profit. One possible presentation is depicted in Figure 6.25 where the sire genotypes described earlier are ordered by profit at the optimum endpoint and the interquartile range is displayed. This range represents 25% of the profit values that fall above and below the median profit. Presenting the output in this way allows the user to determine the most profitable sire, but also illustrates the risk that may be associated with a particular decision. For example, the HLHL genotype is most profitable at the optimum endpoint, but the interquartile range is much larger than the next most profitable sire (HHHH). A risk-averse user may wish to accept the reduction in median profit and possible maximum profit to avoid profits in the lower range. The comparison is valid assuming the optimum endpoint can be identified.

Additional output may be displayed to provide a management context for the ranking of profitability at the optimum endpoint. Figure 6.26 shows the interquartile range and median DOF for profit ranked sires. Such a graphical presentation can add information to selection decisions by accounting for ranges of predicted performance of alternative sire progeny. A sire with smaller DOF range may be more attractive as progeny may be more easily managed to the optimum endpoint. Ranges of performance for DOF, outWT, FT or MS may also be used to initially sort cattle into similar groups based on sire genotype. The DSS is designed to be flexible and provide a wide range of phenotypic output for sire comparison to suit the needs of the user.

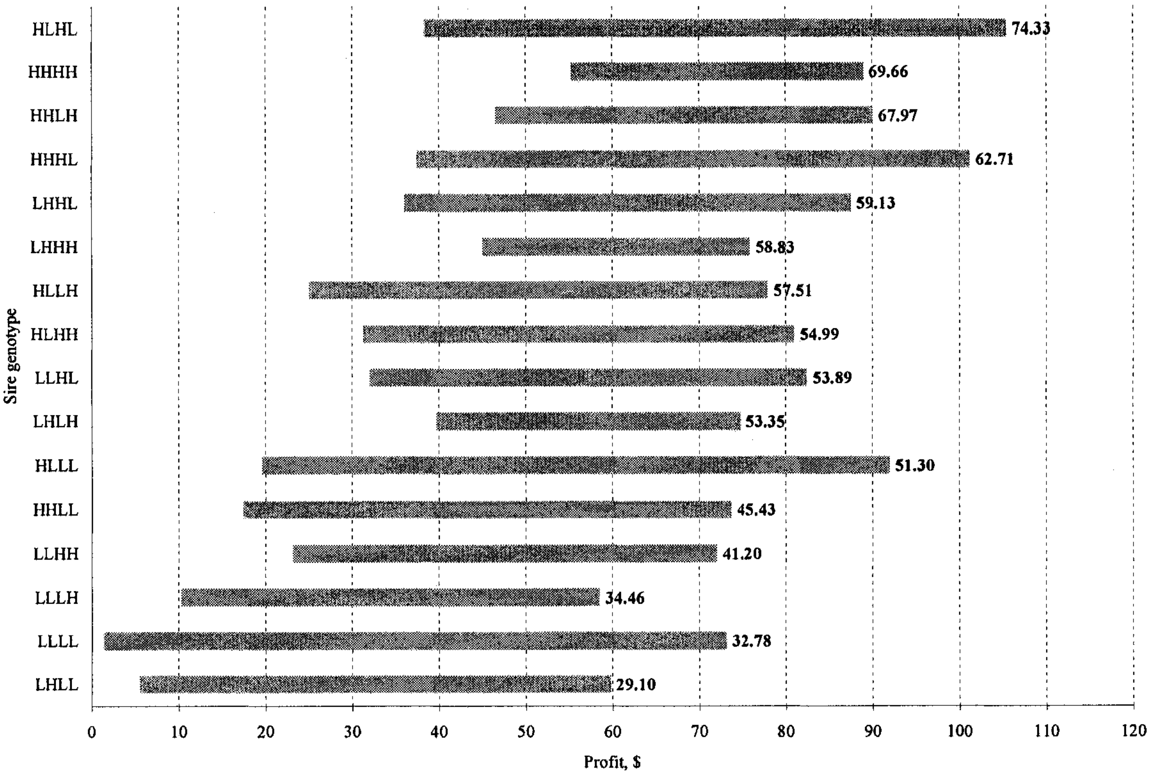

Figure 6.25. Interquartile range (25 to 75% quartiles) and median (to right of range) profit when progeny are fed to an optimum finish endpoint, where sires are ranked in descending order of median profit.

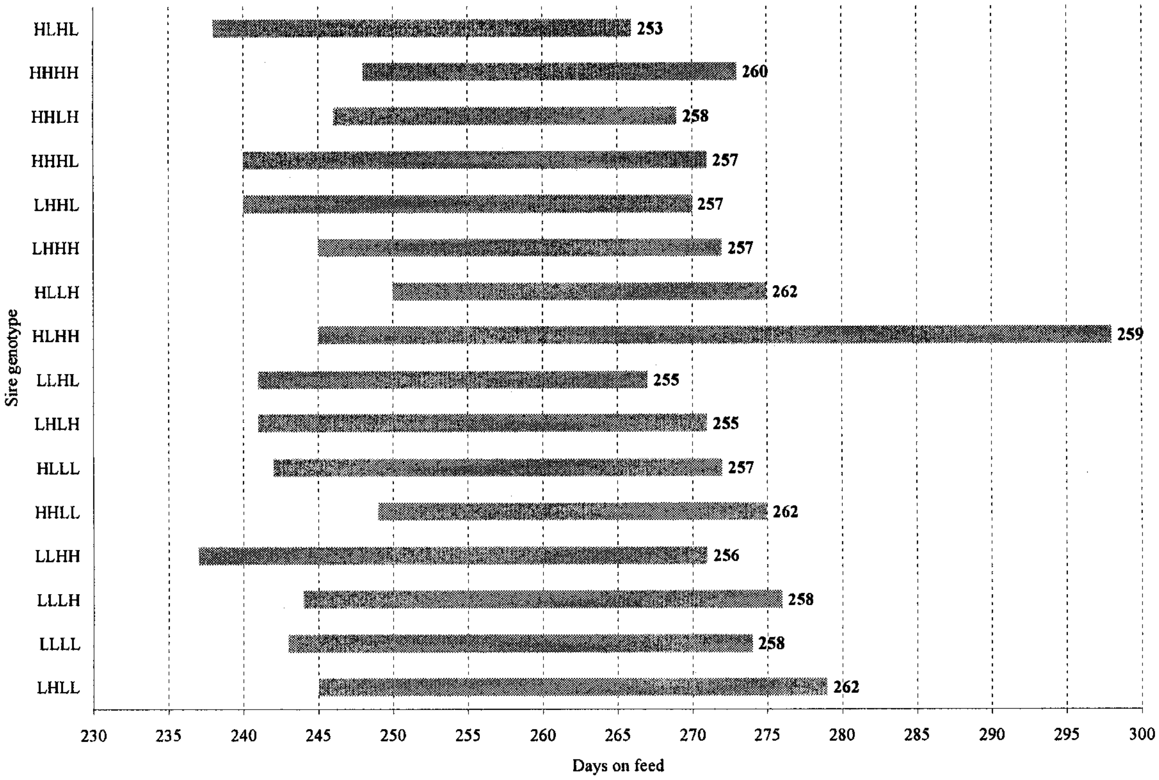

Figure 6.26. Interquartile range (25 to 75% quartiles) and median (to right of range) days on feed when progeny are fed to an optimum endpoint, where sires are ranked in descending order of median profit.

Literature Cited

Amer, P. R., R. A. Kemp, J. G. Buchanan-Smith, G. C. Fox, and C. Smith. 1994a. A bioeconomic model for comparing beef cattle genotypes at their optimal economic slaughter end point. J. Anim. Sci. 72:38-50.

Amer, P. R., R. A. Kemp, G. C. Fox, and C. Smith. 1994b. An economic comparison of beef cattle genotypes for feedlot traits at their optimal slaughter end point. Can. J. Anim. Sci. 74:7-14.

Basarab, J. A., J. R. Brethour, D. R. ZoBell, and B. Graham. 1999. Sorting feeder cattle with a system that integrates ultrasound backfat and marbling estimates with a model that maximizes feedlot profitability in value-based marketing. Can. J. Anim. Sci. 79:327-335.

Block, H. C., J. J. McKinnon, A. F. Mustafa, and D. A. Christensen. 2001. Manipulation of cattle growth to target carcass quality. J. Anim. Sci. 79:133-140.

Bourdon, R. M. 1988. Bovine Nirvana - From the perspective of a modeler and purebred breeder. J. Anim. Sci. 66:1892-1898.

Cleveland, M.A. 2002. Sire selection for optimum endpoints using simulation. Thesis. Colorado State University, Fort Collins.

Feuz, D. M., S. W. Fausti, and J. J. Wagner. 1993. Analysis of the efficiency of four marketing methods for slaughter cattle. Agribusiness 9:453-463.

Koontz, S. R., D. L. Hoag, J. L. Walker, and J. R. Brethour. 2000. Returns to sorting and market timing of animals within pens of fed cattle. Western Agricultural Economics Association Annual Meetings, Vancouver, BC.

Lamb, M. A., M. W. Tess, and O. W. Robison. 1992. Evaluation of mating systems involving five breeds for integrated beef production systems: II. Feedlot segment. J. Anim. Sci. 70:700-713.

Langemeier, M., T. Schroeder, and J. Mintert. 1992. Determinants of cattle finishing profitability. S. J. Agric. Econ. 24:41-47.

Lawrence, J. D., A. Wang, and D. Loy. 1999. Elements of cattle feeding profitability in midwest feedlots. J. Agric. Appl. Econ. 31:349-357.

Pyatt, N. A., L. L. Berger, D. B. Faulkner, P. M. Walker, and Rodriguez-Zas. 2005a. Factors affecting carcass value and profitability in early-weaned Simmental steers: I. Five-year average pricing. J. Anim. Sci. 83:2918-2925.

Pyatt, N. A., L. L. Berger, D. B. Faulkner, P. M. Walker, and Rodriguez-Zas. 2005b. actors affecting carcass value and profitability in early-weaned Simmental steers: II. Days on feed endpoints and sorting strategies. J. Anim. Sci. 83:2926-2937.

Schroeder, T. C. and J. L. Graff. 1999. Comparing live weight, dressed weight, and grid pricing: Assessing the value of cattle quality information. Research Bulletin 1-99, Research Institute on Livestock Pricing, Virginia Tech., Blacksburg, VA.

CHAPTER VII

CONCLUSIONS AND IMPLICATIONS

A decision support system (DSS) was presented as a tool to aid in the selection of sires for profit in the postweaning phase of beef production. The DSS appropriately accounts for genetic differences between sires for growth and carcass traits given a specified production system and finish endpoint as current genetic predictions of growth and carcass traits, alone, are not suitable selection criteria for improving profitability. The DSS addresses the need for a decision-making framework when the goal is to maximize profit in postweaning production. Such a system has application for breeder selection decisions, when appropriate random regression expected progeny differences (EPD) become available, and for use in research where an infinite number of sire genotype, finish endpoint and management decision combinations can be evaluated for impact on profit. The design of the DSS makes it suitable for implementation in a number of environments, including web-based applications.

Scenarios simulated using the DSS indicated the impact on expected profit associated with not accounting for genetic differences in growth and carcass traits at a constant finish endpoint. Results showed that certain genotypes were less profitable on average than would be expected considering only breed or biological type averages and conferred a greater risk of large negative profit values. The probability of negative profits was found to decrease, and profit increased, when progeny of all sires were fed to an optimum endpoint, but certain sire genotypes still had a negative impact on profitability. Variation in days on feed was significantly reduced at the optimum endpoint, however, indicating that single harvest criteria for profit maximization may exist within a breed. Re-ranking for profit was observed between the two endpoints highlighting the importance of accounting for finish endpoints when making selection decisions. Furthermore, it was found that accounting for genetic differences within a breed is necessary to correctly predict expected outcomes allowing selection for sires with the most profitable combination of traits when progeny are fed to a given finish endpoint.

The difficulty in determining the optimal combination of traits to maximize profit at each endpoint was illustrated. At the backfat thickness endpoint, other than extreme high and low EPD levels there was no obvious relationship between sire genotype and profit. A slightly better relationship was evident at the optimum endpoint, but it can be concluded that intuitive selection processes do not exist for postweaning traits when the goal is profit maximization. This conclusion is re-enforced by the results of the third study that showed continued selection for individual improvements in growth and carcass traits did not maintain profitability at any endpoint, assuming static market conditions. It is also difficult to predict performance based on genotypes alone as the importance of factors determining profit changes at alternative endpoints.

Further work is needed to refine the marbling model in the DSS to more accurately account for genetic differences in the prediction of marbling score. Development of appropriate random regression EPD for each trait will likely enhance the usefulness of the system. Continued validation of the system when data become available would also be appropriate to ensure that results are representative of any biological type in any production system. The simulation scenarios could also be expanded to evaluate the impact on profitability, and the re-ranking that may occur, with changes in management or marketing systems as well as changes in market conditions.

APPENDIX I

JAVADOC SUPPORTING DECISION SUPPORT SYSTEM

Following is the Java documentation describing classes, methods and parameters used in the decision support system. This documentation is produced in HTML and is designed as a web-based navigational tool for the classes and methods included in an application.

postweaningdss

Class RunDSS2

public class **RunDSS2**

Title: Postweaning DSS

Description: Runs classe from postweaning package to simulate animal performance in the feedlot to set or identify optimum finish endpoints, in terms of growth, carcass composition and profitability. The **runBase** method is used prior to the other methods to parameterize the model to match simulated animal performance with expected performance. Adjustments to several parameters are stored in the .ini parameter file and used for each subsequent run.

The **runDeterm** method is used to model average progeny of a specific sire and dam combination to a desired finish endpoint (weight; backfat; marbling). Progeny performance, carcass composition and profitability are output at that endpoint. Progeny growth and carcass performance are adjusted by random regression EPD for weight, backfat and marbling on a specific day on feed.

The **runStoch** method is used to model progeny performance of a specific sire and dam with adjustments to starting weight due to Mendelian sampling. This mehtod also allows the simulation of individual or pens of animals. Progeny are simulated on feed for a large number of days and the day of maximum profit is identified. Progeny performance, carcass composition and profitability are output for that day. A large number of rounds are run to give ranges for the output. Progeny growth and carcass performance are adjusted by random regression EPD for weight, backfat and marbling on a specific day on feed.

Copyright: August 31, 2005

Company: Colorado State University - Department of Animal Sciences

Command Line Inputs: Parameter Filename (.ini), Output Filename (.csv), Run Type (1 = base animal; 2 = regular run)

Version:
1.0
Author:
M.A. Cleveland

Method Detail

main

`public static void main(String[] args)`

Main method for RunDSS2

Parameters:

`args` - String[]

Throws:

`IOException` -

---

runBase

```
private static void runBase(String paramFileName)
```

Run simulation for base animal and compare output to expected output input by user. Adjust animal requirements and iterate until outputs are the same.

Parameters:

`paramFileName` - String

---

runDeterm

```
private static void runDeterm(String paramFileName,
                              String outFileName)
```

run deterministic model

Parameters:

`paramFileName` - String

`outFileName` - String

Throws:

`IOException` -

---

runStoch

```
private static void runStoch(String paramFileName,
                             String outFileName)
```

Method to run stochastic model

Parameters:

`paramFileName` - String

`outFileName` - String

Throws:

`IOException` -

postweaningdss

Class CarcassGrid

public class **CarcassGrid**

Title: Postweaning DSS

Description: Class to describe a grid to determine the value of a carcass

Copyright: June 17, 2005

Company: Colorado State University - Department of Animal Sciences

Version:

1.0

Author:

M.A. Cleveland

Constructor Detail

CarcassGrid

```
public CarcassGrid()
```

default constructor for CarcassGrid

CarcassGrid

```
public CarcassGrid(double basePrice,
                   double overWeight,
                   double underWeight,
                   double outWeightDeviation,
                   double primeGradeDeviation,
                   double upperChoiceGradeDeviation,
                   double choiceGradeDeviation,
                   double selectGradeDeviation,
                   double underSelectDeviation,
                   double yg1Deviation,
                   double yg2Deviation,
                   double yg3Deviation,
                   double yg4Deviation,
                   double yg5Deviation)
```

Constructor for CarcassGrid

Parameters:

`basePrice` - double

`overWeight` - double

`underWeight` - double

`outWeightDeviation` - double

`primeGradeDeviation` - double
`upperChoiceGradeDeviation` - double
`choiceGradeDeviation` - double
`selectGradeDeviation` - double
`underSelectDeviation` - double
`yg1Deviation` - double
`yg2Deviation` - double
`yg3Deviation` - double
`yg4Deviaiton` - double
`yg5Deviation` - double

Method Detail

basePrice

`public double basePrice()`

Get base price of carcass

Returns:

double

overWeight

`public double overWeight()`

Get the upper limit of normal carcass weight range

Returns:

double

underWeight

`public double underWeight()`

Get the lower limit of the normal weight range

Returns:

double

outWeightDeviation

`public double outWeightDeviation()`

Get the deviation in value (per 45.45kg) for carcasses outside the normal weight range

Returns:

double

primeGradeDeviation

`public double primeGradeDeviation()`

Get the deviation in value (per 45.45kg) for carcasses in the Prime quality grade category

Returns:

double

---

upperChoiceGradeDeviation

`public double upperChoiceGradeDeviation()`

Get the deviation in value (per 45.45kg) for carcasses in the upper 2/3 Choice quality grade category

Returns:

double

---

choiceGradeDeviation

`public double choiceGradeDeviation()`

Get the deviation in value (per 45.45kg) for carcasses in the Choice quality grade category

Returns:

double

---

selectGradeDeviation

`public double selectGradeDeviation()`

Get the deviation in value (per 45.45kg) for carcasses in the Select quality grade category

Returns:

double

---

underSelectDeviation

`public double underSelectDeviation()`

Get the deviation in value (per 45.45kg) for carcasses in quality grade categories under the Select quality grade

Returns:

double

---

yg1Deviation

`public double` **`yg1Deviation`**`()`

Get the deviation in value (per 45.45kg) for carcasses in the yield grade 1 category

Returns:

double

---

yg2Deviation

`public double` **`yg2Deviation`**`()`

Get the deviation in value (per 45.45kg) for carcasses in the yield grade 2 category

Returns:

double

---

yg3Deviation

`public double` **`yg3Deviation`**`()`

Get the deviation in value (per 45.45kg) for carcasses in the yield grade 3 category

Returns:

double

---

yg4Deviation

`public double` **`yg4Deviation`**`()`

Get the deviation in value (per 45.45kg) for carcasses in the yield grade 4 category

Returns:

double

---

yg5Deviation

`public double` **`yg5Deviation`**`()`

Get the deviation in value (per 45.45kg) for carcasses in the yield grade 5 category

Returns:

double

---

postweaningdss

Class Economics

public class **Economics**

Title: Postweaning DSS

Description: Class to describe variable and fixed daily costs associated with postweaning production and methods to calculate total costs and revenue

Copyright: June 17, 2005

Company: Colorado State University - Department of Animal Sciences

Version:
1.0

Author:
M.A. Cleveland

Constructor Detail

Economics

```
public Economics()
```

Default constructor

Economics

```
public Economics(double perKgFeedCost,
                 double feederCalfPrice,
                 double feedLoanAmount,
                 double dailyYardage,
                 double interestRate,
                 double fixedCost)
```

Constructor for Economics

Parameters:

`perKgFeedCost` - double
`feederCalfPrice` - double
`feedLoanAmount` - double
`dailyYardage` - double
`interestRate` - double
`fixedCost` - double

Method Detail

updateDailyCost

```
public void updateDailyCost(Growth currentGrowth,
                            FeedRation feed,
                            TerminalAnimal currentAnimal)
```

Method to update each category of cost on a daily basis. The updated daily feed cost is determined by the price of feed and the amount of intake for that day.

Parameters:

`currentGrowth` - Growth
`feed` - FeedRation
`currentAnimal` - TerminalAnimal

---

cumulativeTotalCost

```
public double cumulativeTotalCost()
```

Returns the cumulative total cost (including feed cost, variable cost and fixed cost) for all days on feed to the point method is called

Returns:

double

---

totalRevenue

```
public double totalRevenue(TerminalAnimal currentAnimal,
                           CarcassGrid currentGrid)
```

Returns total revenue determined by the specified carass grid in CarcassGrid and animal composition in TerminalAnimal

Parameters:

`currentAnimal` - TerminalAnimal
`currentGrid` - CarcassGrid

Returns:

double

---

postweaningdss

Class Environment

public class **Environment**

Title: Postweaning DSS

Description: Class to describe factors external to the animal

Copyright: August 23, 2005

Company: Colorado State University - Department of Animal Sciences

Version:
1.0

Author:
M.A. Cleveland

Constructor Detail

Environment

```
public Environment()
```

Default constructor

Environment

```
public Environment(double avgTemp,
                   double relHumid,
                   double windSpeed,
                   double hoursSun,
                   double mudDepth)
```

Constructor for Environment

Parameters:

`avgTemp` - double

`relHumid` - double

`windSpeed` - double

`hoursSun` - double

`mudDepth` - double

Method Detail

averageTemp

```
public double averageTemp()
```

Returns the average temperature (C)

Returns:
double

---

relativeHumidity

```
public double relativeHumidity()
```

Returns the average relative humidity (%)

Returns:
double

windSpeed

`public double windSpeed()`

Returns the average wind speed (km/h) for a given month

Returns:

double

hoursSunlight

`public double hoursSunlight()`

Returns the average hours of sunlight

Returns:

double

mudDepth

`public double mudDepth()`

Returns the mud depth (cm)

Returns:

double

month

`public int month(int dayOfYear)`

Returns the month as an integer from the input day of year

Parameters:

`dayOfYear` - int

Returns:

int

dayOfYear

`public int dayOfYear(int month)`

Returns the day of year for the first day of the input month

Parameters:

`month` - int

Returns:

int

postweaningdss

Class FeedRation

public class **FeedRation**

Title: Postweaning DSS

Description: Class to describe energy value of feed ration for growing or finishing

Copyright: June 17, 2005

Company: Colorado State University - Department of Animal Sciences

Version:
1.0

Author:
M.A. Cleveland

Constructor Detail

FeedRation

```
public FeedRation()
```

Default constructor

FeedRation

```
public FeedRation(double me,
                  double neMaint,
                  double neGrowth,
                  double percentDryMatter)
```

Constructor for FeedRation

Parameters:

`me` - double
`neMaint` - double
`neGrowth` - double
`percentDryMatter` - double

Method Detail

neMaint

```
public double neMaint()
```

Returns NE for maintenance

Returns:
double

---

me

`public double me()`

Returns ME

Returns:
double

---

neGrowth

`public double neGrowth()`

Returns NE for growth

Returns:
double

---

percentDryMatter

`public double percentDryMatter()`

Returns percent dry matter of feed

Returns:
double

---

postweaningdss

Class Growth

public class **Growth**

Constructor Detail

Growth

`public Growth()`

Default constructor for Growth

Growth

`public Growth(TerminalAnimal currentAnimal,`

```
              SamplingAdjustment sa_g,
              SamplingAdjustment sa_r)
```

Constructor for Growth

Parameters:

`currentAnimal` - TerminalAnimal
`sa_g` - SamplingAdjustment
`sa_r` - SamplingAdjustment

Method Detail

setAdjMEcs

`public void setAdjMEcs(double adjMEcs)`

Set the adjustment for ME requirement (metabolizable energy)

Parameters:

`adjMEcs` - double

---

setAdjNEmr

`public void setAdjNEmr(double adjNEmr)`

Set the adjustment for NEm requirement (net energy for maint.)

Parameters:

`adjNEmr` - double

---

setAdjDMI

`public void setAdjDMI(double adjDMI)`

Set the adjustment for DMI (dry matter intake)

Parameters:

`adjDMI` - double

---

setAdjRE

`public void setAdjRE(double adjRE)`

Set the adjustment for RE (retained energy)

Parameters:

`adjRE` - double

---

setAdjREA

`public void setAdjREA(double adjREA)`

Set the adjustment for REA (ribeye area)

Parameters:

`adjREA` - double

---

setAdjCWP

```
public void setAdjCWP(double adjCWP)
```

Set the adjustment for carcass dressing percent

Parameters:

`adjCWP` - double

---

setShrink

```
public void setShrink(double shrink)
```

Set the estimated amount of shrink

Parameters:

`shrink` - double

---

dmi

```
public double dmi()
```

Returns the dry matter intake (kg)

Returns:

double

---

grow

```
public void grow(TerminalAnimal currentAnimal,
                 FeedRation ration,
                 Environment currentEnvironment,
                 Management theManagement,
                 Parent sireParent,
                 Parent damParent,
                 double expFT,
                 double expMS,
                 double expDOF,
                 double expOutWt,
                 double expCWT)
```

Calculates the growth and composition for one day on feed, given animal, feed, environment and management characteristics. Assigns weight change and carcass composition to the current TerminalAnimal object. Requires expected backfat thickness (expFT), expected marbling score (expMS), expected days on feed (expDOF), expected weight at harvest (expOutWt) and expected carcass weight (expCWT) assuming animals are harvested at a low Choice quality grade.

Parameters:

`currentAnimal` - TerminalAnimal
`ration` - FeedRation
`currentEnvironment` - Environment
`theManagement` - Management
`sireParent` - Parent
`damParent` - Parent
`expFT` - double
`expMS` - double
`expDOF` - double
`expOutWt` - double
`expCWT` - double

---

postweaningdss

Class Management

public class **Management**

Title: Postweaning DSS

Description: Class to describe feedlot characteristics and management decisions

Copyright: June 17, 2005

Company: Colorado State University - Department of Animal Sciences

Version:
1.0
Author:
M.A. Cleveland

Constructor Detail

Management

```
public Management()
```

Default constructor

Management

```
public Management(boolean implants,
                  double penAreaPerHead,
                  boolean ionophoreUse)
```

Constructor for Management

Parameters:

`implants` - boolean
`penAreaPerHead` - int
`ionophoreUse` - boolean

Method Detail

implants

`public boolean implants()`

Returns the implant status

Returns:

boolean

---

penAreaPerHead

`public double penAreaPerHead()`

Returns the pen area per head in square meters

Returns:

double

---

ionophoreUse

`public boolean ionophoreUse()`

Returns true if an ionophore is used

Returns:

boolean

---

postweaningdss

Class Parent

public class **Parent**

Title: Package PostweaningDSS

Description: Describes the genetic potential of a sire or dam for random regression growth and carcass traits, prediction accuracy and the variation associated with each trait, and has methods to stochastically generate EPD accounting for possible change.

Copyright: August 23, 2005

Company: Colorado State University - Department of Animal Sciences

Version:
1.0
Author:
M.A. Cleveland

Constructor Detail

Parent

```
public Parent()
```

Default constructor for Parent

Parent

```
public Parent(double WT_RRa,
              double WT_RRb,
              double WT_RRc,
              double FT_RRa,
              double FT_RRb,
              double FT_RRc,
              double MS_RRa,
              double MS_RRb,
              double MS_RRc,
              double REA_epd,
              double WT_Ga,
              double WT_Gb,
              double WT_Gc,
              double FT_Ga,
              double FT_Gb,
              double FT_Gc,
              double MS_Ga,
              double MS_Gb,
              double MS_Gc,
              double REA_G,
              double accWT,
              double accFT,
              double accMS,
              double accREA)
```

Constructor for Parent including random regression EPD, accuracies and additive genetic variances

RRa, b or c = random regression intercept, linear or quadratic coefficient

G = additive genetic variance

acc = accuracy of prediction

Parameters:

`WT_RRa` - double
`WT_RRb` - double
`WT_RRc` - double
`FT_RRa` - double
`FT_RRb` - double
`FT_RRc` - double
`MS_RRa` - double
`MS_RRb` - double
`MS_RRc` - double
`REA_epd` - double
`WT_Ga` - double
`WT_Gb` - double
`WT_Gc` - double
`FT_Ga` - double
`FT_Gb` - double
`FT_Gc` - double
`MS_Ga` - double
`MS_Gb` - double
`MS_Gc` - double
`REA_G` - double
`accWT` - double
`accFT` - double
`accMS` - double
`accREA` - double

Method Detail

WT_RRa

`public double WT_RRa()`

Returns the random regression weight intercept EPD

Returns:

double

WT_RRb

`public double WT_RRb()`

Returns the random regression weight linear coefficient EPD

Returns:

double

WT_RRc

`public double WT_RRc()`

Returns the random regression weight quadratic coefficient EPD

Returns:

double

FT_RRa

`public double FT_RRa()`

Returns the random regression backfat intercept EPD

Returns:

double

FT_RRb

`public double FT_RRb()`

Returns the random regression backfat linear coefficient EPD

Returns:

double

FT_RRc

`public double FT_RRc()`

Returns the random regression backfat quadratic coefficient EPD

Returns:

double

MS_RRa

`public double MS_RRa()`

Returns the random regression marbling score intercept EPD

Returns:

double

MS_RRb

`public double MS_RRb()`

Returns the random regression marbling score linear coefficient EPD

Returns:

double

MS_RRc

`public double MS_RRc()`

Returns the random regression marbling score quadratic coefficient EPD

Returns:

double

REA_epd

`public double REA_epd()`

Returns the REA EPD

Returns:

double

---

accWT

`public double accWT()`

Returns the weight EPD accuracy

Returns:

double

---

accFT

`public double accFT()`

Returns the backfat EPD accuracy

Returns:

double

---

accMS

`public double accMS()`

Returns the marbling score EPD accuracy

Returns:

double

---

accREA

`public double accREA()`

Returns the REA EPD accuracy

Returns:

double

---

setWT_RRa

`public void setWT_RRa(double epdWT_RRa)`

Set the WT_RRa EPD

Parameters:

`epdWT_RRa` - double

---

setWT_RRb

`public void setWT_RRb(double epdWT_RRb)`

Set the WT_RRc EPD

Parameters:

`epdWT_RRb` - double

---

setWT_RRc

`public void setWT_RRc(double epdWT_RRc)`

Set the WT_RRc EPD

Parameters:

`epdWT_RRc` - double

---

setFT_RRa

`public void setFT_RRa(double epdFT_RRa)`

Set the FT_RRa EPD

Parameters:

`epdFT_RRa` - double

---

setFT_RRb

`public void setFT_RRb(double epdFT_RRb)`

Set the FT_RRb EPD

Parameters:

`epdFT_RRb` - double

---

setFT_RRc

`public void setFT_RRc(double epdFT_RRc)`

Set the FT_RRc EPD

Parameters:

`epdFT_RRc` - double

---

setMS_RRa

`public void setMS_RRa(double epdMS_RRa)`

Set the MS_RRa EPD

Parameters:

`epdFT_RRa` - double

---

setMS_RRb

`public void setMS_RRb(double epdMS_RRb)`

Set the MS_RRb EPD

Parameters:

`epdFT_RRb` - double

---

setMS_RRc

`public void setMS_RRc(double epdMS_RRc)`

Set the MS_RRc EPD

Parameters:

`epdFT_RRc` - double

---

setREAepd

`public void setREAepd(double epdREA)`

Set the REA EPD

Parameters:

`epdREA` - double

---

generatedEPD

```
public double generatedEPD(double epd,
                           double additiveVar,
                           double acc)
```

Generate EPD from a distribution based on possible change

Parameters:

`epd` - double

`additiveVar` - double

`acc` - double

Returns:

double

---

postweaningdss

Class PossibleChange

public class **PossibleChange**

Constructor Detail

PossibleChange

`public PossibleChange()`

Default constructor for PossibleChange

Method Detail

possibleChange

```
public static double possibleChange(double additiveVar,
                                    double acc)
```

Returns possible change (standard error of prediction) EPD, determined by EPD accuracy and additive genetic variance (from BIF)

Parameters:

`additiveVar` - double

`acc` - double

Returns:

double

---

samplingAdjustmentDM

`public static double samplingAdjustmentDM(double additiveVar)`

Returns the adjustment to progney performance accounting for dam and Mendelian sampling

Parameters:

`additiveVar` - double

Returns:

double

---

samplingAdjustmentR

`public static double samplingAdjustmentR(double residualVar)`

Returns the residual contribution to an animal's phenotype sampled from a distribution of possible values

Parameters:

`residualVar` - double

Returns:

double

---

postweaningdss

Class SamplingAdjustment

public class **SamplingAdjustment**

Title: Postweaning DSS

Description: Describes variables randomly sampled from a normal distribution with specified mean and variance used to simulate progeny phenotype to account for genetic and residual effects.

Copyright: June 17, 2005

Company: Colorado State University - Department of Animal Sciences

Version:

1.0

Author:

M.A. Cleveland

Constructor Detail

SamplingAdjustment

`public SamplingAdjustment()`

Default constructor

Method Detail

setWT

`public void setWT(double WT)`

Set the adjustment to WT performance

Parameters:

`WT` - double

setFT

`public void setFT(double FT)`

Set the adjustment to FT performance

Parameters:

`FT` - double

---

setMS

`public void setMS(double MS)`

Set the adjustment to MS performance

Parameters:

`MS` - double

---

setREA

`public void setREA(double REA)`

Set the adjustment to REA performance

Parameters:

`REA` - double

---

adjWT

`public double adjWT()`

Returns the adjustment for WT

Returns:

double

---

adjFT

`public double adjFT()`

Returns the adjustment for FT

Returns:

double

---

adjMS

`public double adjMS()`

Returns the adjustment for MS

Returns:

double

---

adjREA

```
public double adjREA()
```

Returns the adjustment for REA

Returns:

double

---

postweaningdss

Class TerminalAnimal

public class **TerminalAnimal**

Title: Postweaning DSS

Description: A class to describe growth and carcass characteristics of a feeedlot steer or heifer

Copyright: August 23, 2005

Company: Colorado State University - Department of Animal Sciences

Version:

1.0

Author:

M.A. Cleveland

Constructor Detail

TerminalAnimal

```
public TerminalAnimal()
```

default constructor

TerminalAnimal

```
public TerminalAnimal(double inWeightMale,
                      double sdInWeightMale,
                      double inWeightFemale,
                      double sdInWeightFemale,
                      double proportionFemaleReplacements,
                      int sexChoice,
                      double ageInDays)
```

Constructor for TerminalAnimal

Parameters:

`inWeightMale` - double

`sdInWeightMale` - double
`inWeightFemale` - double
`sdInWeightFemale` - double
`proportionFemaleReplacements` - double
`sexChoice` - int
`ageInDays` - double

Method Detail

updateAgeInDays

`public void **updateAgeInDays**()`

Increment the age in days by one

---

updateDaysOnFeed

`public void **updateDaysOnFeed**()`

Increment the current number of days on feed from weaning by one

---

setInWeight

`public void **setInWeight**(double inWeight)`

Set the in weight of the current animal

Parameters:

`inWeight` - double

---

setLiveWeight

`public void **setLiveWeight**(double liveWeight)`

Set the current live weight (kg)

Parameters:

`liveWeight` - double

---

setBodyWeightAtChoice

`public void **setBodyWeightAtChoice**(double bodyWeightAtChoice)`

Set the body weight expected at the Choice quality grade (approx. 28% EBF) - will be dependent on breed, frame size, sex, etc.

Parameters:

`bodyWeightAtChoice` - double

---

setCarcassWeight

`public void setCarcassWeight(double carcassWeight)`

Set the current carcass weight (kg)

Parameters:

`carcassWeight` - double

---

setCarcassFatThickness

`public void setCarcassFatThickness(double carcassFatThickness)`

Set the current carcass backfat thickness (cm)

Parameters:

`carcassFatThickness` - double

---

setCarcassMarbling

`public void setCarcassMarbling(double carcassMarbling)`

Set the current carcass marbling score

Parameters:

`carcassMarbling` - double

---

setDressingPercent

`public void setDressingPercent(double dressingPercent)`

Set the current dressing percent

Parameters:

`dressingPercent` - double

---

setQualityGrade

`public void setQualityGrade(double qualityGrade)`

Set the current carcass USDA quality grade

Parameters:

`qualityGrade` - int

---

setYieldGrade

`public void setYieldGrade(double yieldGrade)`

Set the current carcass USDA yield grade

Parameters:

`yieldGrade` - double

---

setREA

`public void setREA(double rea)`

Set the current ribeye area (cm^2)

Parameters:

`rea` - double

---

setHairCoatCode

`public void setHairCoatCode(int hairCoatCode)`

Set the hair coat code

Parameters:

`hairCoatCode` - int

---

setHairDepth

`public void setHairDepth(double hairDepth)`

Set the hair depth (cm)

Parameters:

`hairDepth` - double

---

setHideCode

`public void setHideCode(int hideCode)`

Set the hide code

Parameters:

`hideCode` - int

---

setBCS

`public void setBCS(double bodyConditionScore)`

Set the body condition score

Parameters:

`bodyConditionScore` - double

---

setShrink

`public void setShrink(double shrink)`

Set the estimated amount of shrink

Parameters:

`shrink` - double

---

sex

`public int sex()`

Returns the animal's sex

Returns:

int

---

ageInDays

`public double ageInDays()`

Returns the current age in days of animal

Returns:

int

---

inWeight

`public double inWeight()`

Returns animal in weight (kg)

Returns:

double

---

daysOnFeed

`public double daysOnFeed()`

Returns the current number of days on feed from weaning

Returns:

int

---

liveWeight

`public double liveWeight()`

Returns the current animal live weight (kg)

Returns:

double

---

bodyWeightAtChoice

`public double bodyWeightAtChoice()`

Returns the body weight at Choice quality grade (approx. 28% EBF) (kg)

Returns:

double

---

carcassWeight

`public double carcassWeight()`

Returns the current animal carcass weight (kg)

Returns:

double

---

carcassFatThickness

`public double carcassFatThickness()`

Returns the current animal carcass backfat thickness (cm)

Returns:

double

---

carcassMarbling

`public double carcassMarbling()`

Returns the current animal carcass marbling score

Returns:

double

---

dressingPercent

`public double dressingPercent()`

Returns the dressing percent

Returns:

double

---

qualityGrade

`public double qualityGrade()`

Returns the current carcass USDA quality grade

Returns:

int

---

yieldGrade

`public double yieldGrade()`

Returns the current carcass USDA yield grade

Returns:

int

---

rea

`public double rea()`

Return the current REA

Returns:

double

---

hairCoatCode

`public int hairCoatCode()`

Returns the hair coat code

Returns:

int

---

hairDepth

`public double hairDepth()`

Returns the hair depth (cm)

Returns:

double

---

hideCode

`public int hideCode()`

Returns the hide code

Returns:

int

---

bodyConditionScore

`public double bodyConditionScore()`

Returns the body condition score

Returns:

double

---

generateSex

`private int generateSex(double proportionFemaleReplacements)`

Returns sex of animal as determined by probability of being male or female

Parameters:

`proportionFemaleReplacements` - double

Returns:

int

---

generateWeight

```
private double generateWeight(double inWeight,
                              double sdInWeight)
```

Returns in weight of animal from a distribtuion of values around the mean in weight, having a particular standard deviation

Parameters:

`inWeight` - double

`sdInWeight` - double

Returns:

double

APPENDIX II

POSTWEANING DECISION SUPPORT SYSTEM USE INSTRUCTIONS AND CODE

Following are instructions for using the decision support system in a Linux environment, a sample parameter file and the Java code for each of the classes that make up the application. Classes developed for this project are part of the ***postweaningdss*** package.

The main driver class for the program is **RunDSS2**.

Following the classes developed for the postweaning decision support system are classes from outside sources (**Matrix and INIFile**) necessary for running the program. Authors are attributed in the code.

Use instructions

1. Move all source files (.java) to desired location and compile using the javac command. All source files from the ***postweaningdss, Jama,*** and ***com.freeware.inifiles*** packages are needed.

2. Run the **RunDSS2** class using the java command as:

```
Java –classpath path to class files postweaningdss.RunDSS2
parameterFile outputFileName runType
```

where:

parameterFile = an .ini file listing model parameters (illustrated below)
outputFileName = the name of the .csv file output by the model
runType = the type of model run (1 = runBase; 2 = runDeterm or runStoch)

Parameter file

The parameter file should contain the following fields exactly as they are below. Parameter definitions are given in Chapter IV. The parameter file requires an EPD file (epdFilename) that is illustrated below.

```
[Model]
modelType =
endpointType =
endpoint =

[Animal]
inWtMale =
inWtFemale =
sdInWtMale =
sdInWtFemale =
sexChoice =
ageInDays =
expOutWt =
expDOF =
expDMI =
expCWT =
expDP =
expREA =
expFT =
expMS =
hairCoatCode =
hairDepth =
hideCode =
bodyConditionScore =
```

```
[Management]
propFemaleReplace =
implants =
ionophoreUse =
penAreaPerHead =
animalsPerPen =
shrink =

[Environment]
avgTemp =
relHumid =
windSpeed =
hoursSun =
mudDepth =

[Economic]
basePrice =
upperWeight =
lowerWeight =
outWtDev =
primeDev =
upperCHDev =
choiceDev =
selectDev =
underSEDev =
yg1Dev =
yg2Dev =
yg3Dev =
yg4Dev =
yg5Dev =
perKgFeedCost =
feederCalfPrice =
feedLoanAmount =
fixedCost =
dailyYardage =
interestRate =

[Feed]
ME =
NEg =
NEm =
percentDM =
;ME adjustment
adjME_rq =
;NEm adjustment
```

```
adjNEm_rq =
;DMI adjustment
adjDMI =
;RE adjustment
adjRE =
;REA adjustment
adjREA =
;DP adjustment
adjDP =

[Genetic]
addG_WTa =
addG_WTb =
addG_WTc =
addG_FTa =
addG_FTb =
addG_FTc =
addG_MSa =
addG_MSb =
addG_MSc =
addG_REA =
addCOV_WTab =
addCOV_WTac =
addCOV_WTbc =
addCOV_FTab =
addCOV_FTac =
addCOV_FTbc =
addCOV_MSab =
addCOV_MSac =
addCOV_MSbc =
epdFilename = epd.txt

[Residual]
R_WTa =
R_WTb =
R_WTc =
R_FTa =
R_FTb =
R_FTc =
R_MSa =
R_MSb =
R_MSc =
R_REA =
RCOV_WTab =
RCOV_WTac =
RCOV_WTbc =
```

EPD file

The EPD file contains EPD and accuracies for sire and dams for random regression on weight, backfat thickness, marbling score and the traditional longissimus muscle area EPD. The following columns are required in the EPD file (as a text file, tab or space delimited, with just values – no desciptors). Multiple sire and dam combinations may be represented in a single file as separate records.

sireEPD_WTa = (sire intercept EPD for WT)
sireEPD_WTb = (sire linear coefficient EPD for WT)
sireEPD_WTc = (sire quadratic coefficient EPD for WT)
sireEPD_FTa = (sire intercept EPD for FT)
sireEPD_FTb = (sire linear coefficient EPD for FT)
sireEPD_FTc = (sire quadratic coefficient EPD for FT)
sireEPD_MSa = (sire intercept EPD for MS)
sireEPD_MSb = (sire linear coefficient for MS)
sireEPD_MSc = (sire linear coefficient EPD for MS)
sireEPD_REA = (sire EPD for REA)
damEPD_WTa = (dam intercept EPD for WT)
damEPD_WTb = (dam linear coefficient EPD for WT)
damped_WTc = (dam quadratic coefficient EPD for WT)
damEPD_FTa = (dam intercept EPD for FT)
damEPD_FTb = (dam linear coefficient EPD for FT)
damped_FTc = (dam quadratic coefficient EPD for FT)
damEPD_MSa = (dam intercept EPD for MS)
damEPD_MSb = (dam linear coefficient for MS)
damped_MSc = (dam quadratic coefficient EPD for MS)
damEPD_REA = (dam EPD for REA)
sireACC_WT = (sire WT EPD accuracy)
sireACC_FT = (sire FT EPD accuracy)
sireACC_MS = (sire MS EPD accuracy)
sireACC_REA = (sire REA EPD accuracy)
damACC_WT = (dam WT EPD accuracy)
dameACC_FT = (dam FT EPD accuracy)
damACC_MS = (dam MS EPD accuracy)
damACC_REA = (dam REA EPD accuracy)

RunDSS2 class

```
package postweaningdss;

import com.freeware.inifiles.INIFile;
import java.io.*;
import java.util.*;
import java.text.DecimalFormat;


/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Runs classe from postweaning package to simulate animal performance in the
* feedlot to set or identify optimum finish endpoints, in terms of growth, carcass composition and
* profitability. The <B>runBase</B> method is used prior to the other methods to parameterize the
* model to match simulated animal performance with expected performance. Adjustments to several
* parameters are stored in the .ini parameter file and used for each subsequent run. </p>
* <p></p>
* <p>The <B>runDeterm</B> method is used to model average progeny of a specific sire and dam
* combination to a desired finish endpoint (weight; backfat; marbling). Progeny performance,
* carcass composition and profitability are output at that endpoint. Progeny growth and carcass
* performance are adjusted by random regression EPD for weight, backfat and marbling on a specific
* day on feed.</p>
* <p></p>
* <p>The <B>runStoch</B> method is used to model progeny performance of a specific sire and dam
* with adjustments to starting weight due to Mendelian sampling. This mehtod also allows the
* simulation of individual or pens of animals. Progeny are simulated on feed for a large number
* of days and the day of maximum profit is identified. Progeny performance, carcass composition and
* profitability are output for that day. A large number of rounds are run to give ranges for the
* output. Progeny growth and carcass performance are adjusted by random regression EPD for weight,
* backfat and marbling on a specific day on feed.</p>
* <p></p>
 * <p>Copyright: August 31, 2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * <p>Command Line Inputs: Parameter Filename (.ini), Output Filename (.csv), Run Type
*    (1 = base animal; 2 = regular run)</p>
 * @author M.A. Cleveland
 * @version 1.0
 */
public class RunDSS2 {

 /**
  * Default constructor for RunDSS2
  */
 /* public RunDSS2() {
  }*/

 /**
  * Main method for RunDSS2
  *
  * @param args String[]
  * @throws IOException
  */
 public static void main(String[] args) throws IOException {
  /*validate that parameter and output filenames are entered on command line*/
```

```
try { //try to read filenames
  String checkParamFile = new String(args[0]);
  String checkOutFile = new String(args[1]);
  int runType = Integer.parseInt(args[2]);
}
catch (ArrayIndexOutOfBoundsException exception) {
  //exception thrown if two filenames and runType not entered
  System.out.println(
     "Error: Two filenames must be entered, one parameter file and one output file");
  System.out.println(
     "Additionally, a run type must be entered: 1 = base run; 2 = regular run");
  System.out.println("");
}
/*end try/catch*/

/*read in filenames for parameter file and designated output file*/
String paramFile = new String(args[0]);
String outFile = new String(args[1]);
int runType = Integer.parseInt(args[2]);

/*determine which type of model run*/
if (runType == 1) { //run base animal model to match growth curves
  RunDSS2.runBase(paramFile);
}
else { //run regular model (determ or stoch)
  /*get model type*/
  int modelType;
  INIFile inFile = new INIFile(paramFile);
  modelType = (inFile.getIntegerProperty("Model", "modelType")).intValue();

  /*determine which method to call (which model to run)*/
  if (modelType == 1) {
    RunDSS2.runDeterm(paramFile, outFile); //run the deterministic model
  }
  else if (modelType == 2) {
    RunDSS2.runStoch(paramFile, outFile); //run the stochastic model
  }
  else {
    System.out.println("Error: Model type must be 1 or 2");
    System.out.println("");
  }
}

} //end public static void main(String[] args)

/**
 * Run simulation for base animal and compare output to expected output input by user.  Adjust
 * animal requirements and iterate until outputs are the same.
 *
 * @param paramFileName String
 */
private static void runBase(String paramFileName) {

 System.out.println("Start runBase method of RunDSS2");
 System.out.println("");
```

```
/*Initialize file object*/
INIFile inFile = new INIFile(paramFileName);

/*DECLARE AND GET PARAMETERS FROM FILE*/

/***Animal***/
double inWtMale = (inFile.getDoubleProperty("Animal", "inWtMale")).
   doubleValue();
double ageInDays = (inFile.getDoubleProperty("Animal", "ageInDays")).doubleValue();
//expected performance to low choice
double expOutWt = (inFile.getDoubleProperty("Animal", "expOutWt")).
   doubleValue();
double expDOF = (inFile.getDoubleProperty("Animal", "expDOF")).doubleValue();
double expDMI = (inFile.getDoubleProperty("Animal", "expDMI")).doubleValue();
double expCWT = (inFile.getDoubleProperty("Animal", "expCWT")).doubleValue();
double expDP = (inFile.getDoubleProperty("Animal", "expDP")).doubleValue();
double expREA = (inFile.getDoubleProperty("Animal", "expREA")).doubleValue();
double expFT = (inFile.getDoubleProperty("Animal", "expFT")).doubleValue();
double expMS = (inFile.getDoubleProperty("Animal", "expMS")).doubleValue();
//other animal characteristics
int hairCoatCode = (inFile.getIntegerProperty("Animal", "hairCoatCode")).
   intValue();
double hairDepth = (inFile.getDoubleProperty("Animal", "hairDepth")).
   doubleValue();
int hideCode = (inFile.getIntegerProperty("Animal", "hideCode")).intValue();
double bodyConditionScore =
   (inFile.getDoubleProperty("Animal", "bodyConditionScore")).doubleValue();

/***Management***/
boolean implants = (inFile.getBooleanProperty("Management", "implants")).
   booleanValue();
boolean ionophoreUse = (inFile.getBooleanProperty("Management", "implants")).
   booleanValue();
double penAreaPerHead = (inFile.getDoubleProperty("Management",
   "penAreaPerHead")).doubleValue();
double shrink = (inFile.getDoubleProperty("Management", "shrink")).doubleValue();

/***Environment***/
double avgTemp = (inFile.getDoubleProperty("Environment", "avgTemp")).
   doubleValue();
double relHumid = (inFile.getDoubleProperty("Environment", "relHumid")).
   doubleValue();
double windSpeed = (inFile.getDoubleProperty("Environment", "windSpeed")).
   doubleValue();
double hoursSun = (inFile.getDoubleProperty("Environment", "hoursSun")).
   doubleValue();
double mudDepth = (inFile.getDoubleProperty("Environment", "mudDepth")).
   doubleValue();

/***Feed***/
double ME = (inFile.getDoubleProperty("Feed", "ME")).doubleValue();
double NEg = (inFile.getDoubleProperty("Feed", "NEg")).doubleValue();
double NEm = (inFile.getDoubleProperty("Feed", "NEm")).doubleValue();
double percentDM = (inFile.getDoubleProperty("Feed", "percentDM")).
   doubleValue();
```

```
/***Genetic***/
//additive genetic variance for random regression traits
double addG_WTa = 0;
double addG_WTb = 0;
double addG_WTc = 0;
double addG_FTa = 0;
double addG_FTb = 0;
double addG_FTc = 0;
double addG_MSa = 0;
double addG_MSb = 0;
double addG_MSc = 0;
double addG_REA = 0;
//random regression EPD for sire and dam
double sireEPD_WTa = 0;
double sireEPD_WTb = 0;
double sireEPD_WTc = 0;
double sireEPD_FTa = 0;
double sireEPD_FTb = 0;
double sireEPD_FTc = 0;
double sireEPD_MSa = 0;
double sireEPD_MSb = 0;
double sireEPD_MSc = 0;
double sireEPD_REA = 0;
double damEPD_WTa = 0;
double damEPD_WTb = 0;
double damEPD_WTc = 0;
double damEPD_FTa = 0;
double damEPD_FTb = 0;
double damEPD_FTc = 0;
double damEPD_MSa = 0;
double damEPD_MSb = 0;
double damEPD_MSc = 0;
double damEPD_REA = 0;
//EPD acc for sire and dam
double sireACC_WT = 0;
double sireACC_FT = 0;
double sireACC_MS = 0;
double sireACC_REA = 0;
double damACC_WT = 0;
double damACC_FT = 0;
double damACC_MS = 0;
double damACC_REA = 0;
/*****END GET PARAMETERS FROM FILE******/

/*set requirements adjustments to 1*/
inFile.setDoubleProperty("Feed", "adjME_rq", 1, "ME adjustment");
inFile.setDoubleProperty("Feed", "adjNEm_rq", 1, "NEm adjustment");
inFile.setDoubleProperty("Feed", "adjDMI", 1, "DMI adjustment");
inFile.setDoubleProperty("Feed","adjRE",1,"RE adjustment");
inFile.setDoubleProperty("Feed","adjREA",1,"REA adjustment");
inFile.setDoubleProperty("Feed","adjDP",1,"DP adjustment");
inFile.save();

int round = 0;
int exit = 0;
/*loop for comparing simulated and expected outcomes*/
```

```
while (exit == 0) { //exit when diffences are small

 /*declare other variables*/
 double cumulativeIntake = 0;
 round++;
 /*instantiate sire and dam objects*/
 Parent sire = new Parent(sireEPD_WTa, sireEPD_WTb, sireEPD_WTc, sireEPD_FTa,
                    sireEPD_FTb, sireEPD_FTc,
                    sireEPD_MSa,
                    sireEPD_MSb, sireEPD_MSc, sireEPD_REA, addG_WTa, addG_WTb, addG_WTc,
                    addG_FTa,
                    addG_FTb, addG_FTc, addG_MSa,
                    addG_MSb, addG_MSc, addG_REA, sireACC_WT, sireACC_FT,
                    sireACC_MS, sireACC_REA);
 Parent dam = new Parent(damEPD_WTa, damEPD_WTb, damEPD_WTc, damEPD_FTa,
                   damEPD_FTb,damEPD_FTc,
                    damEPD_MSa,
                    damEPD_MSb, damEPD_MSc, damEPD_REA, addG_WTa, addG_WTb,addG_WTc,
                    addG_FTa, addG_FTb,addG_FTc,
                    addG_MSa,
                    addG_MSb, addG_MSc, addG_REA, damACC_WT, damACC_FT,
                    damACC_MS, damACC_REA);

 /*instantiate progeny object*/
 TerminalAnimal progeny = new TerminalAnimal(inWtMale, 0, 0, 0, 0, 2,
                             ageInDays);
 progeny.setBodyWeightAtChoice(expOutWt); //set based on user input of expected out weight
 progeny.setHairCoatCode(hairCoatCode);
 progeny.setHairDepth(hairDepth);
 progeny.setHideCode(hideCode);
 progeny.setBCS(bodyConditionScore);
 progeny.setShrink(shrink);

 /*instantiate management object*/
 Management manage = new Management(implants, penAreaPerHead, ionophoreUse);

 /*instantiate ration object*/
 FeedRation ration = new FeedRation(ME, NEm, NEg, percentDM);

 /*instantiate environment object*/
 Environment env = new Environment(avgTemp, relHumid, windSpeed, hoursSun,
                          mudDepth);

 /*vary sire EPD to account for dam and Mendelian sampling effects*/
   SamplingAdjustment sa_g = new SamplingAdjustment();
   SamplingAdjustment sa_r = new SamplingAdjustment();

 /*instantiate Growth object*/
 Growth g = new Growth(progeny, sa_g, sa_r);
 g.setShrink(shrink);
 /*set animal requirements adjustments from param file*/
 g.setAdjMEcs(inFile.getDoubleProperty("Feed", "adjME_rq").doubleValue());
 g.setAdjNEmr(inFile.getDoubleProperty("Feed", "adjNEm_rq").doubleValue());
 g.setAdjDMI(inFile.getDoubleProperty("Feed", "adjDMI").doubleValue());
 g.setAdjRE(inFile.getDoubleProperty("Feed", "adjRE").doubleValue());
 g.setAdjREA(inFile.getDoubleProperty("Feed", "adjREA").doubleValue());
```

```
    g.setAdjCWP(inFile.getDoubleProperty("Feed", "adjDP").doubleValue());

    /*iterate grow method for number of days expected to reach finish, input by user*/
    for (int i = 0; i < expDOF; i++) {
     /*run growth*/
     g.grow(progeny, ration, env, manage, sire, dam, expFT, expMS, expDOF, expOutWt,expCWT);

     /*track intake*/
     cumulativeIntake += (g.dmi());
    } //end for

    /*****WRITE OUTPUT TO SCREEN*********/
     /*System.out.println("round, dof, lwt, cwt, ft, ms");
     System.out.println(round +" " +progeny.daysOnFeed() +" " +progeny.liveWeight() +" "
       +progeny.carcassWeight() +" " +progeny.carcassFatThickness() +" "
                 +progeny.carcassMarbling());

    System.out.println("**************Total DMI = " +cumulativeIntake);
    System.out.println("Carcass: weight, dp, qg, yg, rea");
    System.out.println(progeny.carcassWeight() +" " +progeny.dressingPercent() +" "
             +progeny.qualityGrade() +" " +progeny.yieldGrade() +"," +progeny.rea());

/*************************************************************************************/

    /*simulataneously compare composition and DMI from simulatiom to expected composition and DMI
     from user input*/
    if ( ((Math.abs(progeny.liveWeight() - expOutWt)) / expOutWt) > 0.00001 ||
       ((Math.abs(cumulativeIntake - expDMI)) / expDMI) > 0.00001 ||
       ((Math.abs(progeny.carcassFatThickness() - expFT)) / expFT) > 0.00001 ||
       ((Math.abs(progeny.dressingPercent() - expDP) / expDP) > 0.00001) ||
       ((Math.abs(progeny.rea() - expREA) / expREA) > 0.00001)) {

       if (progeny.liveWeight() > expOutWt) {
        /*if the animal reaches the specified weight sooner than the specified number of days,
            then the maint requirements should be increased, leaving less energy for growth*/
        /*increment the requirements adjustment*/
        double adj = 1 - (expOutWt / progeny.liveWeight());
        inFile.setDoubleProperty("Feed", "adjME_rq",
                       (inFile.getDoubleProperty("Feed","adjME_rq").doubleValue() *
                        (1 + adj)),"ME adjustment");

        inFile.setDoubleProperty("Feed", "adjNEm_rq",
                       (inFile.getDoubleProperty("Feed","adjNEm_rq").doubleValue() *
                        (1 + adj)),"NEm adjustment");
        inFile.save();
       } //end if(progeny.liveWeight() < expOutWt)
       else {
        /*the animal has less weight and thus the maint requirments need to be reduced,
              leaving more energy for growth*/
        /*decrement the requirements*/
        double adj = 1 - (progeny.liveWeight() / expOutWt);
        inFile.setDoubleProperty("Feed", "adjME_rq",
                       (inFile.getDoubleProperty("Feed","adjME_rq").doubleValue() *
                        (1 - adj)),
                       "ME adjustment");
```

```
  inFile.setDoubleProperty("Feed", "adjNEm_rq",
                    (inFile.getDoubleProperty("Feed",
     "adjNEm_rq").doubleValue() *
                     (1 - adj)),
                    "NEm adjustment");
  inFile.save();
 } //end else

 /*compare DMI to expected DMI*/
 if (cumulativeIntake < expDMI) {
  /*increment the dmi adjustment if intake is less than expected*/
  double adj = 1 - (cumulativeIntake / expDMI);
  inFile.setDoubleProperty("Feed", "adjDMI",
                    (inFile.getDoubleProperty("Feed", "adjDMI").
                     doubleValue() *
                     (1 + adj)),
                    "DMI adjustment");
  inFile.save();
 } //end if
 else {
  /*decrement dmi adjustment if intake is moe than expectd*/
  double adj = 1 - (expDMI / cumulativeIntake);
  inFile.setDoubleProperty("Feed", "adjDMI",
                    (inFile.getDoubleProperty("Feed", "adjDMI").
                     doubleValue() *
                     (1 - adj)),
                    "DMI adjustment");
  inFile.save();
 } //end else

 /*compare FT to expected FT*/
 if (progeny.carcassFatThickness() < expFT) {
  double adj = (1 - (progeny.carcassFatThickness() / expFT))*0.5;
  /*increment the RE adjustment if intake is less than expected*/
  inFile.setDoubleProperty("Feed", "adjRE",
                    (inFile.getDoubleProperty("Feed", "adjRE").
                     doubleValue() *
                     (1 + adj)),
                    "RE adjustment");
  inFile.save();
 } //end if
 else {
  /*decrement RE adjustment if intake is more than expectd*/
  double adj = (1 - (expFT / progeny.carcassFatThickness()))*0.5;
  inFile.setDoubleProperty("Feed", "adjRE",
                    (inFile.getDoubleProperty("Feed", "adjRE").
                     doubleValue() *
                     (1 - adj)),
                    "RE adjustment");
  inFile.save();
 } //end else

 /*compare REA to expected REA*/
 if (progeny.rea() > expREA) {
  double adj = (1 - (expREA / progeny.rea()));
  /*increment the REA adjustment if REA is greater than expected*/
```

```
        inFile.setDoubleProperty("Feed", "adjREA",
                        (inFile.getDoubleProperty("Feed", "adjREA").
                         doubleValue() *
                         (1 + adj)),
                        "REA adjustment");
        inFile.save();
      } //end if
      else {
        /*decrement REA adjustment if REA is samller than expectd*/
        double adj = (1 - (progeny.rea() / expREA));
        inFile.setDoubleProperty("Feed", "adjREA",
                        (inFile.getDoubleProperty("Feed", "adjREA").
                         doubleValue() *
                         (1 - adj)),
                        "REA adjustment");
        inFile.save();
      } //end else

      /*compare DP to expected DP*/
      if (progeny.dressingPercent() > expDP) {
        double adj = (1 - (expDP / progeny.dressingPercent()));
        /*decrement the DP adjustment if DP is greater than expected*/
        inFile.setDoubleProperty("Feed", "adjDP",
                        (inFile.getDoubleProperty("Feed", "adjDP").
                         doubleValue() *
                         (1 - adj)),
                        "DP adjustment");
        inFile.save();
      } //end if
      else {
        /*increment DP adjustment if DP is samller than expectd*/
        double adj = (1 - (progeny.dressingPercent() / expDP));
        inFile.setDoubleProperty("Feed", "adjDP",
                        (inFile.getDoubleProperty("Feed", "adjDP").
                         doubleValue() *
                         (1 + adj)),
                        "DP adjustment");
        inFile.save();
      } //end else
      } //end  if((Math.abs(progeny.liveWeight() - expOutWt)) / expOutWt > 0.05)
      else {
        exit = 1; //exit loop
      } //end else
    } //end while(exit == 0)

    System.out.println("runBase finished in " +round +" rounds");
    System.out.println("");

  } //end private static void runBase

  /**
   * run deterministic model
   *
   * @param paramFileName String
   * @param outFileName String
   * @throws IOException
```

```
 */
private static void runDeterm(String paramFileName, String outFileName) throws
   IOException {
 /*run deterministic model for each sire/dam in EPD file*/

 System.out.println("Start runDeterm method in RunDSS2");
 System.out.println("");

 /*Initialize file object*/
 INIFile inFile = new INIFile(paramFileName);

 /*DECLARE AND GET PARAMETERS FROM FILE*/
 /***Model***/
 //possible values: weight; backfat; marbling
 String endpointType = inFile.getStringProperty("Model", "endpointType");
 double endpoint = (inFile.getDoubleProperty("Model", "endpoint")).
   doubleValue();

 /***Animal***/
 double inWtMale = (inFile.getDoubleProperty("Animal", "inWtMale")).
   doubleValue();
 double ageInDays = (inFile.getDoubleProperty("Animal", "ageInDays")).doubleValue();
 //expected performance to low choice
 double expOutWt = (inFile.getDoubleProperty("Animal", "expOutWt")).
   doubleValue(); //****************shrunk weight at low choice**************
 double expCWT = (inFile.getDoubleProperty("Animal", "expCWT")).
   doubleValue();
 //other animal characteristics
 int hairCoatCode = (inFile.getIntegerProperty("Animal", "hairCoatCode")).
   intValue();
 double hairDepth = (inFile.getDoubleProperty("Animal", "hairDepth")).
   doubleValue();
 int hideCode = (inFile.getIntegerProperty("Animal", "hideCode")).intValue();
 double bodyConditionScore =
   (inFile.getDoubleProperty("Animal", "bodyConditionScore")).doubleValue();
 double expDOF = (inFile.getDoubleProperty("Animal", "expDOF")).doubleValue();
 double expFT = (inFile.getDoubleProperty("Animal", "expFT")).doubleValue();
 double expMS = (inFile.getDoubleProperty("Animal", "expMS")).doubleValue();

 /***Management***/
 boolean implants = (inFile.getBooleanProperty("Management", "implants")).
   booleanValue();
 boolean ionophoreUse = (inFile.getBooleanProperty("Management", "implants")).
   booleanValue();
 double penAreaPerHead = (inFile.getDoubleProperty("Management",
   "penAreaPerHead")).doubleValue();
 double shrink = (inFile.getDoubleProperty("Management", "shrink")).doubleValue();

 /***Environment***/
 double avgTemp = (inFile.getDoubleProperty("Environment", "avgTemp")).
   doubleValue();
 double relHumid = (inFile.getDoubleProperty("Environment", "relHumid")).
   doubleValue();
 double windSpeed = (inFile.getDoubleProperty("Environment", "windSpeed")).
   doubleValue();
 double hoursSun = (inFile.getDoubleProperty("Environment", "hoursSun")).
```

```
   doubleValue();
double mudDepth = (inFile.getDoubleProperty("Environment", "mudDepth")).
   doubleValue();

/***Economic***/
//grid parameters (per 45.45kg of CWT)
double basePrice = (inFile.getDoubleProperty("Economic", "basePrice")).
   doubleValue();
double upperWeight = (inFile.getDoubleProperty("Economic", "upperWeight")).
   doubleValue();
double lowerWeight = (inFile.getDoubleProperty("Economic", "lowerWeight")).
   doubleValue();
double outWtDev = (inFile.getDoubleProperty("Economic", "outWtDev")).
   doubleValue();
double primeDev = (inFile.getDoubleProperty("Economic", "primeDev")).
   doubleValue();
double upperCHDev = (inFile.getDoubleProperty("Economic", "upperCHDev")).
   doubleValue();
double choiceDev = (inFile.getDoubleProperty("Economic", "choiceDev")).
   doubleValue();
double selectDev = (inFile.getDoubleProperty("Economic", "selectDev")).
   doubleValue();
double underSEDev = (inFile.getDoubleProperty("Economic", "underSEDev")).
   doubleValue();
double yg1Dev = (inFile.getDoubleProperty("Economic", "yg1Dev")).
   doubleValue();
double yg2Dev = (inFile.getDoubleProperty("Economic", "yg2Dev")).
   doubleValue();
double yg3Dev = (inFile.getDoubleProperty("Economic", "yg3Dev")).
   doubleValue();
double yg4Dev = (inFile.getDoubleProperty("Economic", "yg4Dev")).
   doubleValue();
double yg5Dev = (inFile.getDoubleProperty("Economic", "yg5Dev")).
   doubleValue();
//cost parameters
double perKgFeedCost = (inFile.getDoubleProperty("Economic",
   "perKgFeedCost")).doubleValue();
double feederCalfPrice = (inFile.getDoubleProperty("Economic",
   "feederCalfPrice")).doubleValue();
double feedLoanAmount = (inFile.getDoubleProperty("Economic",
   "feedLoanAmount")).doubleValue();
double fixedCost = (inFile.getDoubleProperty("Economic", "fixedCost")).
   doubleValue();
double dailyYardage = (inFile.getDoubleProperty("Economic", "dailyYardage")).
   doubleValue();
double interestRate = (inFile.getDoubleProperty("Economic", "interestRate")).
   doubleValue();

/***Feed***/
double ME = (inFile.getDoubleProperty("Feed", "ME")).doubleValue();
double NEg = (inFile.getDoubleProperty("Feed", "NEg")).doubleValue();
double NEm = (inFile.getDoubleProperty("Feed", "NEm")).doubleValue();
double percentDM = (inFile.getDoubleProperty("Feed", "percentDM")).
   doubleValue();

/***Genetic***/
```

```
//additive genetic variance for random regression traits
double addG_WTa = (inFile.getDoubleProperty("Genetic", "addG_WTa")).
   doubleValue();
double addG_WTb = (inFile.getDoubleProperty("Genetic", "addG_WTb")).
   doubleValue();
double addG_WTc = (inFile.getDoubleProperty("Genetic", "addG_WTc")).
   doubleValue();
double addG_FTa = (inFile.getDoubleProperty("Genetic", "addG_FTa")).
   doubleValue();
double addG_FTb = (inFile.getDoubleProperty("Genetic", "addG_FTb")).
   doubleValue();
double addG_FTc = (inFile.getDoubleProperty("Genetic", "addG_FTc")).
   doubleValue();
double addG_MSa = (inFile.getDoubleProperty("Genetic", "addG_MSa")).
   doubleValue();
double addG_MSb = (inFile.getDoubleProperty("Genetic", "addG_MSb")).
   doubleValue();
double addG_MSc = (inFile.getDoubleProperty("Genetic", "addG_MSc")).
   doubleValue();
double addG_REA = (inFile.getDoubleProperty("Genetic", "addG_REA")).
   doubleValue();
//random regression EPD file for sire and dam
String epdFilename = inFile.getStringProperty("Genetic", "epdFilename");
System.out.println("    Finished getting parameters from file");
System.out.println("");
/*****END GET PARAMETERS FROM FILE******/

/*instantiate object to read EPD file*/
FileReader fr = new FileReader(epdFilename);
BufferedReader epdFile = new BufferedReader(fr);
String line = epdFile.readLine();
StringTokenizer tk;

/*instantiate object to write file*/
FileWriter fw = new FileWriter(outFileName);
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter outFile = new PrintWriter(bw);

/*print header for outFile*/
String header = new String("dof" + "," + "lwt" + ","
                    +"cwt" + ","
                    + "ft" + "," + "ms" + "," + "rea" +"," +"dp" + "," + "qg" +
                    "," + "yg" + "," +"dmi" +"," + "rev"
                    + "," + "cost" + "," + "profit");
outFile.println(header);

while (line != null) {
 /*read file and run simulation for each sire/dam*/
 tk = new StringTokenizer(line);

 /*read EPD and ACC*/
 String sireID = tk.nextToken();
 double sireEPD_WTa = Double.parseDouble(tk.nextToken());
 double sireEPD_WTb = Double.parseDouble(tk.nextToken());
 double sireEPD_WTc = Double.parseDouble(tk.nextToken());
 double sireEPD_FTa = Double.parseDouble(tk.nextToken());
```

```
    double sireEPD_FTb = Double.parseDouble(tk.nextToken());
    double sireEPD_FTc = Double.parseDouble(tk.nextToken());
    double sireEPD_MSa = Double.parseDouble(tk.nextToken());
    double sireEPD_MSb = Double.parseDouble(tk.nextToken());
    double sireEPD_MSc = Double.parseDouble(tk.nextToken());
    double sireEPD_REA = Double.parseDouble(tk.nextToken());
    double damEPD_WTa = Double.parseDouble(tk.nextToken());
    double damEPD_WTb = Double.parseDouble(tk.nextToken());
    double damEPD_WTc = Double.parseDouble(tk.nextToken());
    double damEPD_FTa = Double.parseDouble(tk.nextToken());
    double damEPD_FTb = Double.parseDouble(tk.nextToken());
    double damEPD_FTc = Double.parseDouble(tk.nextToken());
    double damEPD_MSa = Double.parseDouble(tk.nextToken());
    double damEPD_MSb = Double.parseDouble(tk.nextToken());
    double damEPD_MSc = Double.parseDouble(tk.nextToken());
    double damEPD_REA = Double.parseDouble(tk.nextToken());
    //EPD acc for sire and dam
    double sireACC_WT = Double.parseDouble(tk.nextToken());
    double sireACC_FT = Double.parseDouble(tk.nextToken());
    double sireACC_MS = Double.parseDouble(tk.nextToken());
    double sireACC_REA = Double.parseDouble(tk.nextToken());
    double damACC_WT = Double.parseDouble(tk.nextToken());
    double damACC_FT = Double.parseDouble(tk.nextToken());
    double damACC_MS = Double.parseDouble(tk.nextToken());
    double damACC_REA = Double.parseDouble(tk.nextToken());

    /*Declare other variables*/
    double totalCosts, totalRevenue, profit;

    /*instantiate sire and dam objects*/
    Parent sire = new Parent(sireEPD_WTa, sireEPD_WTb, sireEPD_WTc, sireEPD_FTa,
                    sireEPD_FTb, sireEPD_FTc,
                    sireEPD_MSa,
                    sireEPD_MSb, sireEPD_MSc, sireEPD_REA, addG_WTa, addG_WTb, addG_WTc,
                    addG_FTa,
                    addG_FTb, addG_FTc, addG_MSa,
                    addG_MSb, addG_MSc, addG_REA, sireACC_WT, sireACC_FT,
                    sireACC_MS, sireACC_REA);
    Parent dam = new Parent(damEPD_WTa, damEPD_WTb, damEPD_WTc, damEPD_FTa,
damEPD_FTb,damEPD_FTc,
                    damEPD_MSa,
                    damEPD_MSb, damEPD_MSc, damEPD_REA, addG_WTa, addG_WTb,addG_WTc,
                    addG_FTa, addG_FTb,addG_FTc,
                    addG_MSa,
                    addG_MSb, addG_MSc, addG_REA, damACC_WT, damACC_FT,
                    damACC_MS, damACC_REA);


    /*instantiate progeny object*/
    TerminalAnimal progeny = new TerminalAnimal(inWtMale, 0, 0, 0, 0, 2,
                        ageInDays);
    progeny.setBodyWeightAtChoice(expOutWt); //set based on user input of expected out weight
    progeny.setHairCoatCode(hairCoatCode);
    progeny.setHairDepth(hairDepth);
    progeny.setHideCode(hideCode);
    progeny.setBCS(bodyConditionScore);
```

```
progeny.setShrink(shrink);

/*instantiate ration object*/
FeedRation ration = new FeedRation(ME, NEm, NEg, percentDM);

/*instantiate environment object*/
Environment env = new Environment(avgTemp, relHumid, windSpeed, hoursSun,
                    mudDepth);

/*instantiate management object*/
Management manage = new Management(implants, penAreaPerHead, ionophoreUse);

/*instantiate CarcassGrid object*/
CarcassGrid grid = new CarcassGrid(basePrice, upperWeight, lowerWeight,
                    outWtDev,
                    primeDev, upperCHDev, choiceDev,
                    selectDev, underSEDev, yg1Dev, yg2Dev,
                    yg3Dev, yg4Dev, yg5Dev);

/*instantiate Econimics object*/
Economics econ = new Economics(perKgFeedCost, feederCalfPrice,
                  feedLoanAmount,
                  dailyYardage, interestRate, fixedCost);

/*vary sire EPD to account for dam and Mendelian sampling effects*/
  SamplingAdjustment sa_g = new SamplingAdjustment(); //adjustments are zero using this model
  SamplingAdjustment sa_r = new SamplingAdjustment();

/*instantiate Growth object*/
Growth g = new Growth(progeny, sa_g, sa_r);
g.setShrink(shrink);
/*set animal requirements adjustments from param file*/
g.setAdjMEcs(inFile.getDoubleProperty("Feed", "adjME_rq").doubleValue());
g.setAdjNEmr(inFile.getDoubleProperty("Feed", "adjNEm_rq").doubleValue());
g.setAdjDMI(inFile.getDoubleProperty("Feed", "adjDMI").doubleValue());
g.setAdjRE(inFile.getDoubleProperty("Feed", "adjRE").doubleValue());
g.setAdjREA(inFile.getDoubleProperty("Feed", "adjREA").doubleValue());
g.setAdjCWP(inFile.getDoubleProperty("Feed", "adjDP").doubleValue());

double cumulativeIntake = 0;

/*loop growth until selected endpoint is reached*/
if (endpointType.compareToIgnoreCase("weight") == 0) {
  while (progeny.liveWeight() < endpoint) {
    /*run growth*/
    g.grow(progeny, ration, env, manage, sire, dam, expFT, expMS, expDOF, expOutWt, expCWT);

    /*track intake*/
    cumulativeIntake += (g.dmi());
    /*update costs*/
    econ.updateDailyCost(g, ration, progeny);
  } //end while()
} //end if(endpointType == "weight")

else if (endpointType.compareToIgnoreCase("backfat") == 0) {
  while (progeny.carcassFatThickness() < endpoint) {
```

```
      /*run growth*/
      g.grow(progeny, ration, env, manage, sire, dam, expFT, expMS, expDOF, expOutWt,expCWT);
      /*track intake*/
      cumulativeIntake += (g.dmi());
      /*update costs*/
      econ.updateDailyCost(g, ration, progeny);
    } //end while()
  } //end else if(endpointType =="backfat")

  else if (endpointType.compareToIgnoreCase("marbling") == 0) {
    while (progeny.carcassMarbling() < endpoint) {
      /*run growth*/
      g.grow(progeny, ration, env, manage, sire, dam, expFT, expMS, expDOF, expOutWt,expCWT);
      /*track intake*/
      cumulativeIntake += (g.dmi());
      /*update costs*/
      econ.updateDailyCost(g, ration, progeny);
    } //end while()
  } //end else if(endpointType == "marbling")

  else {
    System.out.println("Error: RunDSS Class - incompatible endpointType");
    System.out.println("");
  } //end else

  /*calculate costs, revenue and profit*/
  totalCosts = econ.cumulativeTotalCost();
  totalRevenue = econ.totalRevenue(progeny, grid);
  profit = totalRevenue - totalCosts;

  /*create a decimal formatter*/
  DecimalFormat fmt = new DecimalFormat("0.###");
  DecimalFormat fmt_c = new DecimalFormat("0.00");

  /*print performance to file*/
  String out = new String(sireID +"," +progeny.daysOnFeed() +","
                  + fmt.format(progeny.liveWeight()) + ","
                  + fmt.format(progeny.carcassWeight()) + "," +
                  fmt.format(progeny.carcassFatThickness())
                  + "," +fmt.format(progeny.carcassMarbling())
                  + "," +fmt.format(progeny.rea()) +","
                  + fmt.format(progeny.dressingPercent()) + "," +
                  fmt.format(progeny.qualityGrade()) + ","
                  + fmt.format(progeny.yieldGrade()) + "," +fmt.format(cumulativeIntake)
                  +"," +fmt_c.format(totalRevenue) + "," + fmt_c.format(totalCosts)
                  + "," + fmt_c.format(profit));
  outFile.println(out);

  line = epdFile.readLine();
} //end while(line != null)

epdFile.close();
outFile.close();

System.out.println("runDeterm method finished");
System.out.println("");
```

```
} //end  private static void RunDeterm()

/**
 * Method to run stochastic model
 *
 * @param paramFileName String
 * @param outFileName String
 * @throws IOException
 */
private static void runStoch(String paramFileName, String outFileName) throws
   IOException {

  System.out.println("Start runStoch method in RunDSS2");
  System.out.println("");

  /*****declare constants for model loops******/
  final int DAYS = 350; //number of days to run growth model
  final int ROUNDS = 1000; //number of rounds to run model

  /*Initialize file object*/
  INIFile inFile = new INIFile(paramFileName);

  /***Animal***/
  double inWtMale = (inFile.getDoubleProperty("Animal", "inWtMale")).doubleValue();
  double inWtFemale = (inFile.getDoubleProperty("Animal", "inWtFemale")).doubleValue();
  double sdInWtMale = (inFile.getDoubleProperty("Animal", "sdInWtMale")).doubleValue();
  double sdInWtFemale = (inFile.getDoubleProperty("Animal", "sdInWtFemale")).doubleValue();
  int sexChoice = (inFile.getIntegerProperty("Animal", "sexChoice")).intValue();
  double ageInDays = (inFile.getDoubleProperty("Animal", "ageInDays")).doubleValue();
  //expected performance to low choice
  double expOutWt = (inFile.getDoubleProperty("Animal", "expOutWt")).doubleValue();
  double expDOF = (inFile.getDoubleProperty("Animal", "expDOF")).doubleValue();
  double expCWT = (inFile.getDoubleProperty("Animal", "expCWT")).doubleValue();
  //other animal characteristics
  int hairCoatCode = (inFile.getIntegerProperty("Animal", "hairCoatCode")).intValue();
  double hairDepth = (inFile.getDoubleProperty("Animal", "hairDepth")).doubleValue();
  int hideCode = (inFile.getIntegerProperty("Animal", "hideCode")).intValue();
  double bodyConditionScore =
     (inFile.getDoubleProperty("Animal", "bodyConditionScore")).doubleValue();
  double expFT = (inFile.getDoubleProperty("Animal", "expFT")).doubleValue();
  double expMS = (inFile.getDoubleProperty("Animal", "expMS")).doubleValue();

  /***Management***/
  double propFemaleReplace =
     (inFile.getDoubleProperty("Management", "propFemaleReplace")).doubleValue();
  boolean implants = (inFile.getBooleanProperty("Management", "implants")).booleanValue();
  boolean ionophoreUse = (inFile.getBooleanProperty("Management","ionophoreUse")).booleanValue();
  double penAreaPerHead =
(inFile.getDoubleProperty("Management","penAreaPerHead")).doubleValue();
  int animalsPerPen = (inFile.getIntegerProperty("Management","animalsPerPen")).intValue();
  double shrink = (inFile.getDoubleProperty("Management", "shrink")).doubleValue();

  /***Environment***/
  double avgTemp = (inFile.getDoubleProperty("Environment", "avgTemp")).doubleValue();
  double relHumid = (inFile.getDoubleProperty("Environment", "relHumid")).doubleValue();
```

```
double windSpeed = (inFile.getDoubleProperty("Environment", "windSpeed")).doubleValue();
double hoursSun = (inFile.getDoubleProperty("Environment", "hoursSun")).doubleValue();
double mudDepth = (inFile.getDoubleProperty("Environment", "mudDepth")).doubleValue();

/***Economic***/
//grid parameters (per 45.45kg of CWT)
double basePrice = (inFile.getDoubleProperty("Economic", "basePrice")).doubleValue();
double upperWeight = (inFile.getDoubleProperty("Economic", "upperWeight")).doubleValue();
double lowerWeight = (inFile.getDoubleProperty("Economic", "lowerWeight")).doubleValue();
double outWtDev = (inFile.getDoubleProperty("Economic", "outWtDev")).doubleValue();
double primeDev = (inFile.getDoubleProperty("Economic", "primeDev")).doubleValue();
double upperCHDev = (inFile.getDoubleProperty("Economic", "upperCHDev")).doubleValue();
double choiceDev = (inFile.getDoubleProperty("Economic", "choiceDev")).doubleValue();
double selectDev = (inFile.getDoubleProperty("Economic", "selectDev")).doubleValue();
double underSEDev = (inFile.getDoubleProperty("Economic", "underSEDev")).doubleValue();
double yg1Dev = (inFile.getDoubleProperty("Economic", "yg1Dev")).doubleValue();
double yg2Dev = (inFile.getDoubleProperty("Economic", "yg2Dev")).doubleValue();
double yg3Dev = (inFile.getDoubleProperty("Economic", "yg3Dev")).doubleValue();
double yg4Dev = (inFile.getDoubleProperty("Economic", "yg4Dev")).doubleValue();
double yg5Dev = (inFile.getDoubleProperty("Economic", "yg5Dev")).doubleValue();
//cost parameters
double perKgFeedCost = (inFile.getDoubleProperty("Economic","perKgFeedCost")).doubleValue();
double feederCalfPrice = (inFile.getDoubleProperty("Economic","feederCalfPrice")).doubleValue();
double feedLoanAmount = (inFile.getDoubleProperty("Economic","feedLoanAmount")).doubleValue();
double fixedCost = (inFile.getDoubleProperty("Economic", "fixedCost")).doubleValue();
double dailyYardage = (inFile.getDoubleProperty("Economic", "dailyYardage")).doubleValue();
double interestRate = (inFile.getDoubleProperty("Economic", "interestRate")).doubleValue();

/***Feed***/
double ME = (inFile.getDoubleProperty("Feed", "ME")).doubleValue();
double NEg = (inFile.getDoubleProperty("Feed", "NEg")).doubleValue();
double NEm = (inFile.getDoubleProperty("Feed", "NEm")).doubleValue();
double percentDM = (inFile.getDoubleProperty("Feed", "percentDM")).doubleValue();

/***Genetic***/
//additive genetic variance for random regression traits
double addG_WTa = (inFile.getDoubleProperty("Genetic", "addG_WTa")).
    doubleValue();
 double addG_WTb = (inFile.getDoubleProperty("Genetic", "addG_WTb")).
    doubleValue();
 double addG_WTc = (inFile.getDoubleProperty("Genetic", "addG_WTc")).
    doubleValue();
 double addG_FTa = (inFile.getDoubleProperty("Genetic", "addG_FTa")).
    doubleValue();
 double addG_FTb = (inFile.getDoubleProperty("Genetic", "addG_FTb")).
    doubleValue();
 double addG_FTc = (inFile.getDoubleProperty("Genetic", "addG_FTc")).
    doubleValue();
 double addG_MSa = (inFile.getDoubleProperty("Genetic", "addG_MSa")).
    doubleValue();
 double addG_MSb = (inFile.getDoubleProperty("Genetic", "addG_MSb")).
    doubleValue();
 double addG_MSc = (inFile.getDoubleProperty("Genetic", "addG_MSc")).
    doubleValue();
 double addG_REA = (inFile.getDoubleProperty("Genetic", "addG_REA")).
    doubleValue();
```

```
double addCOV_WTab = (inFile.getDoubleProperty("Genetic", "addCOV_WTab")).
   doubleValue();
double addCOV_WTac = (inFile.getDoubleProperty("Genetic", "addCOV_WTac")).
   doubleValue();
double addCOV_WTbc = (inFile.getDoubleProperty("Genetic", "addCOV_WTbc")).
   doubleValue();

/*******Residual*********/
//residual variance for traits
double R_WTa = (inFile.getDoubleProperty("Residual", "R_WTa")).doubleValue();
double R_WTb = (inFile.getDoubleProperty("Residual", "R_WTb")).doubleValue();
double R_WTc = (inFile.getDoubleProperty("Residual", "R_WTc")).doubleValue();
double R_FTa = (inFile.getDoubleProperty("Residual", "R_FTa")).doubleValue();
double R_FTb = (inFile.getDoubleProperty("Residual", "R_FTb")).doubleValue();
double R_FTc = (inFile.getDoubleProperty("Residual", "R_FTc")).doubleValue();
double R_MSa = (inFile.getDoubleProperty("Residual", "R_MSa")).doubleValue();
double R_MSb = (inFile.getDoubleProperty("Residual", "R_MSb")).doubleValue();
double R_MSc = (inFile.getDoubleProperty("Residual", "R_MSc")).doubleValue();
double R_REA = (inFile.getDoubleProperty("Residual", "R_REA")).doubleValue();
double RCOV_WTab = (inFile.getDoubleProperty("Residual", "RCOV_WTab")).doubleValue();
double RCOV_WTac = (inFile.getDoubleProperty("Residual", "RCOV_WTac")).doubleValue();
double RCOV_WTbc = (inFile.getDoubleProperty("Residual", "RCOV_WTbc")).doubleValue();

//random regression EPD file for sire and dam
String epdFilename = inFile.getStringProperty("Genetic", "epdFilename");
/*****END GET PARAMETERS FROM FILE******/

System.out.println("     Finished getting parameters from file");
System.out.println("");

/*instantiate object to read EPD file*/
FileReader fr = new FileReader(epdFilename);
BufferedReader epdFile = new BufferedReader(fr);
String line = epdFile.readLine();
StringTokenizer tk;

/*instantiate object to write file*/
FileWriter fw = new FileWriter(outFileName);
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter outFile = new PrintWriter(bw);

/*print outFile Header*/
String header = new String("sire_id" +"," +"pen_size" +"," +"dof" +"," +"total_lwt" +"," +
                "avg_lwt" +"," +"total_cwt" +"," +"avg_cwt" +"," +"avg_ft" +"," +
                "avg_ms" +"," +"avg_rea" +"," +"avg_dp" +"," +"avg_qg" +"," +"avg_yg"
                +"," +
                "total_dmi" +"," +"avg_dmi" +"," +"total_rev" +"," +"avg_rev" +"," +
                "total_cost" +"," +"avg_cost" +"," +"total_profit" +","
                +"avg_profit");
outFile.println(header);

/*track sire/dam combination*/
int line_num = 0;

while (line != null) {
  line_num++;
```

```
 /*read file and run simulation for each sire/dam*/
 tk = new StringTokenizer(line);

 /*read EPD and ACC*/
String  sireID = tk.nextToken();
double sireEPD_WTa = Double.parseDouble(tk.nextToken());
double sireEPD_WTb = Double.parseDouble(tk.nextToken());
double sireEPD_WTc = Double.parseDouble(tk.nextToken());
double sireEPD_FTa = Double.parseDouble(tk.nextToken());
double sireEPD_FTb = Double.parseDouble(tk.nextToken());
double sireEPD_FTc = Double.parseDouble(tk.nextToken());
double sireEPD_MSa = Double.parseDouble(tk.nextToken());
double sireEPD_MSb = Double.parseDouble(tk.nextToken());
double sireEPD_MSc = Double.parseDouble(tk.nextToken());
double sireEPD_REA = Double.parseDouble(tk.nextToken());
double damEPD_WTa = Double.parseDouble(tk.nextToken());
double damEPD_WTb = Double.parseDouble(tk.nextToken());
double damEPD_WTc = Double.parseDouble(tk.nextToken());
double damEPD_FTa = Double.parseDouble(tk.nextToken());
double damEPD_FTb = Double.parseDouble(tk.nextToken());
double damEPD_FTc = Double.parseDouble(tk.nextToken());
double damEPD_MSa = Double.parseDouble(tk.nextToken());
double damEPD_MSb = Double.parseDouble(tk.nextToken());
double damEPD_MSc = Double.parseDouble(tk.nextToken());
double damEPD_REA = Double.parseDouble(tk.nextToken());
//EPD acc for sire and dam
double sireACC_WT = Double.parseDouble(tk.nextToken());
double sireACC_FT = Double.parseDouble(tk.nextToken());
double sireACC_MS = Double.parseDouble(tk.nextToken());
double sireACC_REA = Double.parseDouble(tk.nextToken());
double damACC_WT = Double.parseDouble(tk.nextToken());
double damACC_FT = Double.parseDouble(tk.nextToken());
double damACC_MS = Double.parseDouble(tk.nextToken());
double damACC_REA = Double.parseDouble(tk.nextToken());


 /*Declare other variables*/
 double totalCosts, totalRevenue, profit;

 /*instantiate sire and dam objects*/
 /*want sire and dam to have same EPD for each animal in the total rounds - EPD is varied
     below (by the Parent class) to account for different dams and Mendelian sampling.
     To look at the variation in EPD due to accuracy, the sire and dam need to be run
     through runStoch multiple times*/
 Parent sire = new Parent(sireEPD_WTa, sireEPD_WTb, sireEPD_WTc, sireEPD_FTa,
                    sireEPD_FTb, sireEPD_FTc,
                    sireEPD_MSa,
                    sireEPD_MSb, sireEPD_MSc, sireEPD_REA, addG_WTa, addG_WTb, addG_WTc,
                    addG_FTa,
                    addG_FTb, addG_FTc, addG_MSa,
                    addG_MSb, addG_MSc, addG_REA, sireACC_WT, sireACC_FT,
                    sireACC_MS, sireACC_REA);
   Parent dam = new Parent(damEPD_WTa, damEPD_WTb, damEPD_WTc, damEPD_FTa,
damEPD_FTb,damEPD_FTc,
                   damEPD_MSa,
```

```
                    damEPD_MSb, damEPD_MSc, damEPD_REA, addG_WTa, addG_WTb,addG_WTc,
                    addG_FTa, addG_FTb,addG_FTc,
                    addG_MSa,
                    addG_MSb, addG_MSc, addG_REA, damACC_WT, damACC_FT,
                    damACC_MS, damACC_REA);

//int i; //use when using a constant endpoint
  /*run stochastic simulation for n rounds*/
  for (int j = 0; j < ROUNDS; j++) {

    /*declare variables to accumulate weight and carcass predictions*/
    double[] total_lwt = new double[DAYS];
    double[] total_cwt = new double[DAYS];
    double[] total_ft = new double[DAYS];
    double[] total_ms = new double[DAYS];
    double[] total_rea = new double[DAYS];
    double[] total_dp = new double[DAYS];
    double[] total_qg = new double[DAYS];
    double[] total_yg = new double[DAYS];
    double[] total_dmi = new double[DAYS];
    double[] total_rev = new double[DAYS];
    double[] total_cost = new double[DAYS];

    //create array to store profit each day
    double[] profitVect = new double[DAYS];

  //i = 0; //for use in growth loop when using a constant endpoint
    for(int h = 0; h < animalsPerPen; h++) {

      double cumulativeIntake = 0;

      /*Assign dam and Mendelian sampling effects*/
      /********ASSUMES THAT DAM EPD ARE UNKNOWN**********/
      SamplingAdjustment sa_g = new SamplingAdjustment(); //adj. for unknown genetic effects
      /*When RR EPD are available for FT and MS the adjustment will just be a random
      normal deviate and the phenotypic adjustment will be calculated in the Growth class*/
      sa_g.setFT(PossibleChange.samplingAdjustmentDM(addG_FTa));
      sa_g.setMS(PossibleChange.samplingAdjustmentDM(addG_MSa));
      sa_g.setREA(PossibleChange.samplingAdjustmentDM(addG_REA));
      /********************************************************/

      /*Assign residual effects*/
      SamplingAdjustment sa_r = new SamplingAdjustment();
      /*When RR EPD are available for FT and MS the adjustment will just be a random
      normal deviate and the phenotypic adjustment will be calculated in the Growth class*/
      sa_r.setFT(PossibleChange.samplingAdjustmentR(R_FTa));
      sa_r.setMS(PossibleChange.samplingAdjustmentR(R_MSa));
      sa_r.setREA(PossibleChange.samplingAdjustmentR(R_REA));
     /***********************************************************************/
      Random rand = new Random();
      sa_g.setWT(rand.nextGaussian()); //use same random deviate for every day of growth
      sa_r.setWT(rand.nextGaussian());
      /*************/

      /*instantiate progeny object*/
      TerminalAnimal progeny = new TerminalAnimal(inWtMale, sdInWtMale,
```

```
      inWtFemale,
      sdInWtFemale, propFemaleReplace,
      sexChoice, ageInDays);
   progeny.setBodyWeightAtChoice(expOutWt); //set based on user input of expected out weight
   progeny.setHairCoatCode(hairCoatCode);
   progeny.setHairDepth(hairDepth);
   progeny.setHideCode(hideCode);
   progeny.setBCS(bodyConditionScore);
   progeny.setShrink(shrink);

   /*instantiate ration object*/
   FeedRation ration = new FeedRation(ME, NEm, NEg, percentDM);

   /*instantiate environment object*/
   Environment env = new Environment(avgTemp, relHumid, windSpeed,
                        hoursSun,
                        mudDepth);

   /*instantiate management object*/
   Management manage = new Management(implants, penAreaPerHead,
                        ionophoreUse);

   /*instantiate CarcassGrid object*/
   CarcassGrid grid = new CarcassGrid(basePrice, upperWeight, lowerWeight,
                        outWtDev,
                        primeDev, upperCHDev, choiceDev,
                        selectDev, underSEDev, yg1Dev,
                        yg2Dev,
                        yg3Dev, yg4Dev, yg5Dev);

   /*instantiate Econimics object*/
   Economics econ = new Economics(perKgFeedCost, feederCalfPrice,
                      feedLoanAmount,
                      dailyYardage, interestRate, fixedCost);

   /*instantiate Growth object*/
   Growth g = new Growth(progeny, sa_g, sa_r);
   g.setShrink(shrink);
   /*set animal requirements adjustments from param file*/
   g.setAdjMEcs(inFile.getDoubleProperty("Feed", "adjME_rq").doubleValue());
   g.setAdjNEmr(inFile.getDoubleProperty("Feed", "adjNEm_rq").doubleValue());
   g.setAdjDMI(inFile.getDoubleProperty("Feed", "adjDMI").doubleValue());
   g.setAdjRE(inFile.getDoubleProperty("Feed", "adjRE").doubleValue());
   g.setAdjREA(inFile.getDoubleProperty("Feed", "adjREA").doubleValue());
   g.setAdjCWP(inFile.getDoubleProperty("Feed", "adjDP").doubleValue());

   /******/
   /*run growth loop to find the global profit optimum*/
  for (int i = 0; i < DAYS; i++) {

 //use while loop when using a constant endpoint
//while(i < 258) {
//while(progeny.carcassFatThickness() <= 1.1 && i <350) {
    /*run growth*/

    g.grow(progeny, ration, env, manage, sire, dam, expFT, expMS, expDOF, expOutWt,expCWT);
```

```
  /*track intake*/
  cumulativeIntake += (g.dmi());

  /*update costs*/
  econ.updateDailyCost(g, ration, progeny);

  /*calculate costs, revenue and profit*/
  totalCosts = econ.cumulativeTotalCost();
  totalRevenue = econ.totalRevenue(progeny, grid);
  profit = totalRevenue - totalCosts;

  /*store progeny and economic information in arrays*/
  /*accumulate progeny and economic predictions for all animals in pen per day*/
  total_lwt[i] +=  progeny.liveWeight();
  total_cwt[i] += progeny.carcassWeight();
  total_ft[i] += progeny.carcassFatThickness();
  total_ms[i] += progeny.carcassMarbling();
  total_rea[i] += progeny.rea();
  total_dp[i] += progeny.dressingPercent();
  total_qg[i] += progeny.qualityGrade();
  total_yg[i] += progeny.yieldGrade();
  total_dmi[i] += cumulativeIntake;
  total_rev[i] += totalRevenue;
  total_cost[i] += totalCosts;
  profitVect[i] += profit;

 //i++; //when using a constant endpoint
 } //end for(int i = 0; i < DAYS; i++)
 /********/
} //end  for(int h = 0; h < animalsPerPen; h++}

 /*find profit maximum*/
 double maximum = profitVect[0]; // start with the first value
 int maxIndex = 0;
 for (int k = 1; k < profitVect.length; k++) {
  if (profitVect[k] > maximum) {
    maximum = profitVect[k]; // new maximum
    maxIndex = k; //index of profit max
  } //end if
 } //end for(k = 1; k < profitVect.length; k++)

 int ps = animalsPerPen; //for calculating average

 /*create a decimal formatter*/
 DecimalFormat fmt = new DecimalFormat("0.########");
 DecimalFormat fmt_c = new DecimalFormat("0.00");

 /***************NEED TOP CHANGE*********/
//int maxIndex = i-1;  //when using a constant endpoint

 /*write to file information at each round for maximum profit*/
 String out = new String(sireID +"," +ps +"," +(maxIndex+1) +","
                    +fmt.format(total_lwt[maxIndex]) +","
                    +fmt.format(total_lwt[maxIndex]/ps) +","
                    +fmt.format(total_cwt[maxIndex]) +","
```

```
                +fmt.format((total_cwt[maxIndex]/ps)) +","
                +fmt.format((total_ft[maxIndex]/ps)) +","
                +fmt.format(total_ms[maxIndex]/ps) +","
                +fmt.format(total_rea[maxIndex]/ps) +","
                +fmt.format((total_dp[maxIndex]/ps)) +","
                +fmt.format((total_qg[maxIndex]/ps)) +","
                +fmt.format((total_yg[maxIndex]/ps)) +","
                +fmt.format(total_dmi[maxIndex]) +","
                +fmt.format((total_dmi[maxIndex]/ps)) +","
                +fmt_c.format(total_rev[maxIndex]) +","
                +fmt_c.format(total_rev[maxIndex]/ps) +","
                +fmt_c.format(total_cost[maxIndex]) + ","
                +fmt_c.format(total_cost[maxIndex]/ps) +","
                +fmt_c.format(profitVect[maxIndex]) +","
                +fmt_c.format(profitVect[maxIndex]/ps));
    outFile.println(out);
    System.out.println("Round " +(j+1) +" complete");

   } //end for(int j = 0; j < ROUNDS; j++)

   line = epdFile.readLine(); //increment readline
  } //end while(line != null)

  epdFile.close();
  outFile.close();

  System.out.println("");
  System.out.println("runStoch method finished");
  System.out.println("");

 } //end private static void RunStoch()
} //end public class RunDSS2
```

CarcassGrid class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 *
 * <p>Description: Class to describe a grid to determine the value of a carcass
 *
 * <p>Copyright: June 17, 2005</p>
 *
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 *
 * @author M.A. Cleveland
 * @version 1.0
 */
public class CarcassGrid {

 /*declare variables*/
 private double basePrice, overWeight, underWeight, outWeightDeviation;
 private double primeGradeDeviation, upperChoiceGradeDeviation, choiceGradeDeviation;
```

```
private double selectGradeDeviation, underSelectDeviation;
private double yg1Deviation, yg2Deviation, yg3Deviation, yg4Deviation, yg5Deviation;

 /**
 * default constructor for CarcassGrid
 */
public CarcassGrid() {

}

/**
 * Constructor for CarcassGrid
 *
 * @param basePrice double
 * @param overWeight double
 * @param underWeight double
 * @param outWeightDeviation double
 * @param primeGradeDeviation double
 * @param upperChoiceGradeDeviation double
 * @param choiceGradeDeviation double
 * @param selectGradeDeviation double
 * @param underSelectDeviation double
 * @param yg1Deviation double
 * @param yg2Deviation double
 * @param yg3Deviation double
 * @param yg4Deviaiton double
 * @param yg5Deviation double
 */
public CarcassGrid(double basePrice, double overWeight, double underWeight,
            double outWeightDeviation,
            double primeGradeDeviation, double upperChoiceGradeDeviation,
            double choiceGradeDeviation, double selectGradeDeviation,
            double underSelectDeviation, double yg1Deviation, double yg2Deviation,
            double yg3Deviation, double yg4Deviation, double yg5Deviation) {

 /*set instance variables*/
 this.basePrice = basePrice;
 this.overWeight = overWeight;
 this.underWeight = underWeight;
 this.outWeightDeviation = outWeightDeviation;
 this.primeGradeDeviation = primeGradeDeviation;
 this.upperChoiceGradeDeviation = upperChoiceGradeDeviation;
 this.choiceGradeDeviation = choiceGradeDeviation;
 this.selectGradeDeviation = selectGradeDeviation;
 this.underSelectDeviation = underSelectDeviation;
 this.yg1Deviation = yg1Deviation;
 this.yg2Deviation = yg2Deviation;
 this.yg3Deviation = yg3Deviation;
 this.yg4Deviation = yg4Deviation;
 this.yg5Deviation = yg5Deviation;

 /******************DEVIATIONS ARE PER 45.45KG**************************/

} //end CarcassGrid

/*get methods*/
```

```
/**
 * Get base price of carcass
 *
 * @return double
 */
public double basePrice() {

  return this.basePrice;
} //end public double basePrice

/**
 * Get the upper limit of normal carcass weight range
 *
 * @return double
 */
public double overWeight() {

  return this.overWeight;
} //end public overWeight

/**
 * Get the lower limit of the normal weight range
 *
 * @return double
 */
public double underWeight() {

  return this.underWeight;
} //end public underWeight

/**
 * Get the deviation in value (per 45.45kg) for carcasses outside the normal weight range
 *
 * @return double
 */
public double outWeightDeviation() {

  return this.outWeightDeviation;
} //end public outWeightDeviation

/**
 * Get the deviation in value (per 45.45kg) for carcasses in the Prime quality grade category
 *
 * @return double
 */
public double primeGradeDeviation() {

  return this.primeGradeDeviation;
} //end public primeGradeDeviation
/**
 * Get the deviation in value (per 45.45kg) for carcasses in the upper 2/3 Choice quality grade
 * category
 *
 * @return double
 */
public double upperChoiceGradeDeviation() {
```

```
  return this.upperChoiceGradeDeviation;
} //end public upperChoiceGradeDeviation

/**
 * Get the deviation in value (per 45.45kg) for carcasses in the Choice quality grade category
 *
 * @return double
 */
public double choiceGradeDeviation() {

  return this.choiceGradeDeviation;
} //end public choiceGradeDeviation

/**
 * Get the deviation in value (per 45.45kg) for carcasses in the Select quality grade category
 *
 * @return double
 */
public double selectGradeDeviation() {

  return this.selectGradeDeviation;
} //end public selectGradeDeviation

/**
 * Get the deviation in value (per 45.45kg) for carcasses in quality grade categories under the
 * Select quality grade
 *
 * @return double
 */
public double underSelectDeviation() {

  return this.underSelectDeviation;
} //end public underSelectDeviation

/**
 * Get the deviation in value (per 45.45kg) for carcasses in the yield grade 1 category
 *
 * @return double
 */
public double yg1Deviation() {

  return this.yg1Deviation;
} //end public yg1Deviation

/**
 * Get the deviation in value (per 45.45kg) for carcasses in the yield grade 2 category
 *
 * @return double
 */
public double yg2Deviation() {

  return this.yg2Deviation;
} //end public yg2Deviation

/**
```

```
  * Get the deviation in value (per 45.45kg) for carcasses in the yield grade 3 category
  *
  * @return double
  */
 public double yg3Deviation() {

   return this.yg3Deviation;
 } //end public yg3Deviation

 /**
  * Get the deviation in value (per 45.45kg) for carcasses in the yield grade 4 category
  *
  * @return double
  */
 public double yg4Deviation() {

   return this.yg4Deviation;
 } //end public yg4Deviation

 /**
  * get the deviation in value (per 45.45kg) for carcasses in the yield grade 5 category
  *
  * @return double
  */
 public double yg5Deviation() {

   return this.yg5Deviation;
 } //end public yg5Deviation

} //end public class CarcassGrid
```

Economics class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Class to describe variable and fixed daily costs associated with postweaning
*               production and methods to calculate total costs and revenue
 * <p>Copyright: June 17,  2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * @author M.A. Cleveland
 * @version 1.0
 */

public class Economics {

 /*declare instance variables*/
 private double perKgFeedCost; //cost of feed per kilogram
 private double feederCalfPrice; //per kg
 private double feedLoanAmount; //amount financed for feed and yardage
 private double dailyYardage; //amount of daily yardage per head
 private double interestRate; //annual interest rate
 private double fixedCost; //includes trucking, processing, etc.
```

```
private double feederCalfValue; //value of feeder calf based on price and weight

private double cumulativeDailyVariableCost = 0; //sums costs each day

/**
 * Default constructor
 */
public Economics() {

}

/**
 * Constructor for Economics
 *
 * @param perKgFeedCost double
 * @param feederCalfPrice double
 * @param feedLoanAmount double
 * @param dailyYardage double
 * @param interestRate double
 * @param fixedCost double
 */
public Economics(double perKgFeedCost, double feederCalfPrice, double feedLoanAmount,
            double dailyYardage, double interestRate, double fixedCost) {

  this.perKgFeedCost = perKgFeedCost;
  this.feederCalfPrice = feederCalfPrice;
  this.feedLoanAmount = feedLoanAmount;
  this.dailyYardage = dailyYardage;
  this.interestRate = interestRate;
  this.fixedCost = fixedCost;

} //end public Economics

/*set methods*/
/**
 * Updates each category of cost on a daily basis. The updated daily feed cost is
 * determined by the price of feed and the amount of intake for that day.
 *
 * @param currentGrowth Growth
 * @param feed FeedRation
 * @param currentAnimal TerminalAnimal
 */
public void updateDailyCost(Growth currentGrowth, FeedRation feed, TerminalAnimal currentAnimal) {

  //calculate daily intake dependent on % dry matter of feed
  double dailyIntake = currentGrowth.dmi() / feed.percentDryMatter();

  //determine daily feed costs from intake and cost of feed
  double dailyFeedCost = this.perKgFeedCost * dailyIntake;

  //determine value of feeder calf
  this.feederCalfValue = (currentAnimal.inWeight() * this.feederCalfPrice);

  //determine interest charges for the current day
  double dailyAnimalInterest = (this.interestRate / 365) * this.feederCalfValue;
  double dailyFeedInterest = (this.interestRate / 365) * this.feedLoanAmount;
```

```
  //sum all cossts and track cumulative costs each day
  this.cumulativeDailyVariableCost += (dailyFeedCost + dailyYardage +
                                  dailyAnimalInterest + dailyFeedInterest);

} //end public void updateDailyCost

/**
 * Returns the cumulative total cost (including feed cost, variable cost and fixed cost) for all
 * days on feed to the point method is called
 *
 * @return double
 */
public double cumulativeTotalCost() {

    return (this.cumulativeDailyVariableCost + this.fixedCost + this.feederCalfValue);
} //end public double cumulativeTotalCost()

/**
 * Returns total revenue determined by the specified carass grid in CarcassGrid and animal
 * composition in TerminalAnimal
 *
 * @param currentAnimal TerminalAnimal
 * @param currentGrid CarcassGrid
 * @return double
 */
public double totalRevenue(TerminalAnimal currentAnimal,
                        CarcassGrid currentGrid) {
  double revenue = 0; //return
  double adjPrice = 0; //base price adjusted for carcass characteristics (per 45.45.kg)
  double basePrice = currentGrid.basePrice(); //get carcass base price (per 45.45kg)

  /*variables to store deviations from base value (per 45.45kg)*/
  double outWeightDev = 0;
  double qualityDev = 0;
  double yieldDev = 0;

  /*get carcass characteristics*/
  double qualityGrade = currentAnimal.qualityGrade();
  double yieldGrade = currentAnimal.yieldGrade();
  double cwt = currentAnimal.carcassWeight();

  /*determine out weight deviation*/
  if(cwt <= currentGrid.overWeight() && cwt >= currentGrid.underWeight()) {
    outWeightDev = 0;
  } //end if
  else {
    outWeightDev = currentGrid.outWeightDeviation();
  } //end else

  /*determine quality grade deviation*/
  if(qualityGrade < 2) {
    qualityDev = currentGrid.primeGradeDeviation();
  }
  else if(qualityGrade >=2 && qualityGrade < 3) {
    qualityDev = currentGrid.upperChoiceGradeDeviation();
```

```
    }
    else if(qualityGrade >=3 && qualityGrade < 4) {
      qualityDev = currentGrid.choiceGradeDeviation();
    }
    else if(qualityGrade >=4 && qualityGrade < 5) {
      qualityDev = currentGrid.selectGradeDeviation();
    }
    else {
      qualityDev = currentGrid.underSelectDeviation();
    }

    /*determine yield grade deviation*/
    if(yieldGrade < 2) {
      yieldDev = currentGrid.yg1Deviation();
    }
    else if(yieldGrade >= 2 && yieldGrade < 3) {
      yieldDev = currentGrid.yg2Deviation();
    }
    else if(yieldGrade >= 3 && yieldGrade < 4) {
      yieldDev = currentGrid.yg3Deviation();
    }
    else if(yieldGrade >= 4 && yieldGrade < 5) {
      yieldDev = currentGrid.yg4Deviation();
    }
    else if(yieldGrade >= 5) {
      yieldDev = currentGrid.yg5Deviation();
    }
    else {
      System.out.println("Class Economics: not a valid yield grade");
    }

      /*adjust base price using deviations above*/
      adjPrice = basePrice + outWeightDev + qualityDev + yieldDev; //per 45.45kg

      /*calculate revenue - adjust carcass weight*/
      revenue = (cwt / 45.45) * adjPrice;

    return revenue;
  } //end public double totalRevenue()

} //end public class Economics
```

Environment class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Class to describe factors external to the animal
 * <p>Copyright: August 23, 2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * @author M.A. Cleveland
 * @version 1.0
 */
```

```
public class Environment {

 /*declare instance variables*/
 private double Tc; //current average temp. in C
 private double RHc; //current relative humidity in %
 private double WS; //wind speed in km/hour
 private double HRS; //hours of sunlight
 private double Mud; //mud depth in cm

  /**
  * Default constructor
  */
 public Environment() {

 }

 /**
  * Constructor for Environment
  *
  * @param avgTemp double
  * @param relHumid double
  * @param windSpeed double
  * @param hoursSun double
  * @param mudDepth double
  */
 public Environment(double avgTemp, double relHumid, double windSpeed,
              double hoursSun, double mudDepth) {

  /*initialize variables*/
    this.Tc = avgTemp;
    this.RHc = relHumid;
    this.WS = windSpeed;
    this.HRS = hoursSun;
    this.Mud = mudDepth;

 } //end public Environment

 /*get methods*/
 /**
  * Returns the average temperature (C)
  *
  * @return double
  */
 public double averageTemp() {
  return this.Tc;

 } //end public double averageTemp()

 /**
  * Returns the average relative humidity (%)
  *
  * @return double
  */
 public double relativeHumidity() {
  return this.RHc;
```

```
} //end public double relativeHumidity()

/**
 * Returns the average wind speed (km/h) for a given month
 *
 * @return double
 */
public double windSpeed() {
 return this.WS;

} //public double windSpeed()

/**
 * Returns the average hours of sunlight
 *
 * @return double
 */
public double hoursSunlight() {
 return this.HRS;

} //public double hoursSunlight()

/**
 * Returns the mud depth (cm)
 *
 * @return double
 */
public double mudDepth() {
 return this.Mud;

} //end public double mudDepth()

/**
 * Returns the month as an integer from the input day of year
 *
 * @param dayOfYear int
 * @return int
 */
public int month(int dayOfYear) {
 int month;

 if(dayOfYear <= 31) {
  month = 1;
 }
 else if(dayOfYear > 31 && dayOfYear <= 59) {
  month = 2;
 }
 else if(dayOfYear > 59 && dayOfYear <= 90) {
  month = 3;
 }
 else if(dayOfYear > 90 && dayOfYear <= 120) {
  month = 4;
 }
 else if(dayOfYear > 120 && dayOfYear <= 151) {
  month = 5;
```

```
 }
 else if(dayOfYear > 151 && dayOfYear <= 181) {
  month = 6;
 }
 else if(dayOfYear > 181 && dayOfYear <= 212) {
  month = 7;
 }
 else if(dayOfYear > 212 && dayOfYear <= 243) {
  month = 8;
 }
 else if(dayOfYear > 243 && dayOfYear <= 273) {
  month = 9;
 }
 else if(dayOfYear > 273 && dayOfYear <= 304) {
  month = 10;
 }
 else if(dayOfYear > 304 && dayOfYear <= 334) {
  month = 11;
 }
 else if(dayOfYear > 334 && dayOfYear <=365) {
  month = 12;
 }
 else {
  month = 0;
 }
 return month;
} //end public int month()

/**
 * Returns the day of year for the first day of the input month
 *
 * @param month int
 * @return int
 */
public int dayOfYear(int month) {
 int day;

 if(month == 1) {
  day = 1;
 }
 else if(month == 2) {
  day = 32;
 }
 else if(month == 3) {
  day = 60;
 }
 else if(month == 4) {
  day = 91;
 }
 else if(month == 5) {
  day = 121;
 }
 else if(month == 6) {
  day = 152;
 }
 else if(month == 7) {
```

```
    day = 182;
  }
  else if(month == 8) {
    day = 213;
  }
  else if(month == 9) {
    day = 244;
  }
  else if(month == 10) {
    day = 274;
  }
  else if(month == 11) {
    day = 305;
  }
  else if(month == 12) {
    day = 335;
  }
  else {
    day = 0;
  }
  return day;
 } //end public int dayOfYear()

} //end public class Environment
```

FeedRation class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Class to describe energy value of feed ration for growing or finishing
 * <p>Copyright: June 17,  2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * @author M.A. Cleveland
 * @version 1.0
 */

public class FeedRation {

 /*declare instance variables*/
 //me = metabolizable energy; ne = net energy
 private double neMaint;
 private double neGrowth;
 private double me;
 private double percentDryMatter;

 /**
  * Default constructor
  */
 public FeedRation() {

 } //end public FeedRation
```

```
//constructor
/**
 * Constructor for FeedRation
 *
 * @param me double
 * @param neMaint double
 * @param neGrowth double
 * @param percentDryMatter double
 */
public FeedRation(double me, double neMaint,
            double neGrowth, double percentDryMatter) {

 /*set instance variables*/
 this.me = me;
 this.neMaint = neMaint;
 this.neGrowth = neGrowth;
 this.percentDryMatter = percentDryMatter;

} //end public FeedRation

 /*get methods*/

/**
 * Returns NE for maintenance
 *
 * @return double
 */
public double neMaint() {
 return this.neMaint;

} //public double neMaint()

/**
 * Returns ME
 *
 * @return double
 */
public double me() {
 return this.me;

} // public double me()

/**
 * Returns NE for growth
 *
 * @return double
 */
public double neGrowth() {
 return this.neGrowth;

} // public double neGrowth()

/**
 * Returns percent dry matter of feed
 *
 * @return double
```

```
  */
  public double percentDryMatter() {
    return this.percentDryMatter;

  } // public double percentDryMatter()


} //end public class FeedRation
```

Growth class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 *
 * <p>Description: Class to describe the daily weight and composition of animals in the feedlot
 *
 * <p>Copyright: August 23, 2005</p>
 *
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 *
 * @author M.A. Cleveland
 * @version 1.0
 */

import Jama.Matrix; //package for matrix manipulation

public class Growth {

  /*declare instance variables*/
  private double dmi; //dry matter intake for one day
  private double FATinitial; //initial amount of empty body fat
  private double FAT; //cumulative amount of empty body fat
  private double LIVE_TO_SHRUNK_FACTOR = 1; //1 - percent shrink

  /*adjustment factors to set average growth curve*/
  private double adjMEcs;
  private double adjNEmr;
  private double adjDMI;
  private double adjRE;
  private double adjREA;
  private double adjCWP;

  /*variables to store adjustments to performance due to genetic effects*/
  private double sampWT;
  private double sampFT;
  private double sampMS;
  private double sampREA;

  /*variables to store adjustments to performance due to residual effects*/
  private double rsampWT;
  private double rsampFT;
  private double rsampMS;
```

```
private double rsampREA;

private double avgBW; //average body weight
private Matrix Rrr, Grr; //declare genetic and residual matrices

/**
 * Default constructor for Growth
 */
public Growth() {

}

/**
 * Constructor for Growth
 *
 * @param currentAnimal TerminalAnimal
 * @param sa_g SamplingAdjustment
 * @param sa_r SamplingAdjustment
 */
public Growth(TerminalAnimal currentAnimal, SamplingAdjustment sa_g, SamplingAdjustment sa_r) {

  double SBW; //shrunk live body weight

  /*caculate initial amount of body fat (kg)*/
  SBW = currentAnimal.liveWeight() * LIVE_TO_SHRUNK_FACTOR;
  this.FATinitial = (0.00054 * (SBW * 0.891) * 2 + 0.037 * (SBW * 0.891) -
                0.61) * 0.85; //eq 41
  this.FAT = this.FATinitial;

  /*get dam and Mendelian sampling effects to adjust performance*/
  this.sampWT = sa_g.adjWT(); //just a random normal deviate - not the actual effect
    //These are actual affects calculated in PossibleChange
  this.sampFT = sa_g.adjFT();
  this.sampMS = sa_g.adjMS();
  this.sampREA = sa_g.adjREA();

  /*get residual effects to adjust performance*/
  this.rsampWT = sa_r.adjWT(); //just a random normal deviate - not the actual effect
    //These are actual affects calculated in PossibleChange
  this.rsampFT = sa_r.adjFT();
  this.rsampMS = sa_r.adjMS();
  this.rsampREA = sa_r.adjREA();

  this.avgBW = currentAnimal.liveWeight();
  }

/*set methods*/
/**
 * Set the adjustment for ME requirement (metabolizable energy)
 *
 * @param adjMEcs double
 */
public void setAdjMEcs(double adjMEcs) {
  this.adjMEcs = adjMEcs;
} //end public void setAdjMEcs
```

```
/**
 * Set the adjustment for NEm requirement (net energy for maint.)
 *
 * @param adjNEmr double
 */
public void setAdjNEmr(double adjNEmr) {
 this.adjNEmr = adjNEmr;
} //end public void setAdjNEmr

/**
 * Set the adjustment for DMI (dry matter intake)
 *
 * @param adjDMI double
 */
public void setAdjDMI(double adjDMI) {
 this.adjDMI = adjDMI;
} //end public void setAdjDMI

/**
 * Set the adjustment for RE (retained energy)
 *
 * @param adjRE double
 */
public void setAdjRE(double adjRE) {
 this.adjRE = adjRE;
} //end public void setAdjRE

/**
 * Set the adjustment for REA (ribeye area)
 *
 * @param adjREA double
 */
public void setAdjREA(double adjREA) {
 this.adjREA = adjREA;
} //end public void setAdjREA

/**
 * Set the adjustment for carcass dressing percent
 *
 * @param adjCWP double
 */
public void setAdjCWP(double adjCWP) {
 this.adjCWP = adjCWP;
} //end public void setAdjCWP

/**
 * Set the estimated amount of shrink
 *
 * @param shrink double
 */
public void setShrink(double shrink) {
 this.LIVE_TO_SHRUNK_FACTOR = 1 - shrink;
} //end public void setShrink

/*get methods*/
/**
```

```
 * Returns the dry matter intake (kg)
 *
 * @return double
 */
public double dmi() {
  return this.dmi;
} //end public double dmi()

/*other methods*/
/**
 * Calculates the growth and composition for one day on feed, given animal, feed, environment and
 * management characteristics. Assigns weight change and carcass composition to the current
 * TerminalAnimal object. Requires expected backfat thickness (expFT), expected marbling score
 * (expMS), expected days on feed (expDOF), expected weight at harvest (expOutWt) and expected
 * carcass weight (expCWT) assuming animals are harvested at a low Choice quality grade.
 *
 * @param currentAnimal TerminalAnimal
 * @param ration FeedRation
 * @param currentEnvironment Environment
 * @param theManagement Management
 * @param sireParent Parent
 * @param damParent Parent
 * @param expFT double
 * @param expMS double
 * @param expDOF double
 * @param expOutWt double
 * @param expCWT double
 */
public void grow(TerminalAnimal currentAnimal, FeedRation ration,
            Environment currentEnvironment,
            Management theManagement, Parent sireParent,
            Parent damParent, double expFT, double expMS, double expDOF,
            double expOutWt, double expCWT) {

 /************Equations 1-50 from: Tedeschi et al. Agric. Syst. 79:171-204 (2004)*******/

 /*declare and assign values to input variables*/
 final double SRW = 478; //standard reference weight in kg
 double AFSBW = currentAnimal.bodyWeightAtChoice(); //expect. adjusted final shrunk body weight
 double currentBW = this.avgBW; //current live body weight of average animal
 double Tc = currentEnvironment.averageTemp(); //current month average temp. in C
 double RHc = currentEnvironment.relativeHumidity(); //relative humidity in %
 double WS = currentEnvironment.windSpeed(); //wind speed in km/hour
 double HRS = currentEnvironment.hoursSunlight(); //hours of sunlight
 double Mud = currentEnvironment.mudDepth(); //depth of mud in cm
 double ageInDays = currentAnimal.ageInDays(); //current age of animal
 double NEm = ration.neMaint(); //dietary content of net energy for maintenance in Mcal/kg
 int HCCcode = currentAnimal.hairCoatCode(); //hair coat code
 int HideCode = currentAnimal.hideCode(); //hide depth code
 double HD = currentAnimal.hairDepth(); //hair depth in cm
 double BCS = currentAnimal.bodyConditionScore(); //body condition score
  //get physical activity factor (from table 3)
 double PA;
 double penArea = theManagement.penAreaPerHead(); //ped area per head in square meters
 if (penArea >= 3 && penArea <= 5) {
  PA = 1.12;
```

```
}
else if (penArea > 5 && penArea <= 9) {
  PA = 1.16;
}
else if (penArea > 9 && penArea <= 19) {
  PA = 1.16;
}
else if (penArea >= 19) {
  PA = 1.17;
}
else {
  PA = 1; //no adjustment
} //end
final double a1 = 0.072; //fasting heat production coefficient (Mcal/kg^-0.75 * day)
  //default value for beef cattle (table 1)
  //determine ionophore adjustment factor
boolean ionophoreUse = theManagement.ionophoreUse(); //ionophore use true/false
final double IonophoreADGFactor; //adjustment factor for ionophore use
if (ionophoreUse = true) {
  IonophoreADGFactor = 1.12;
}
else {
  IonophoreADGFactor = 1;
} //end
double NEg = ration.neGrowth(); //dietary content of net energy for growth (Mcal/kg)
double ME = ration.me(); //dietary content of metabolizable energy (Mcal/kg)

/*declare calculated variables*/
double EqSBW; //equivalent shrunk body weight
double SBW; //shrunk body weight
double newSBW; //shrunk body weight, including current day's gain
double BFAF; //body fat adjustment factor, adjusts DMI for EBF content
double CETI; //current month effective temp. index
double DMINC; //DMI night cooling adjustment
double DMIAF; //DMI adjustment factor with night cooling
double MudDMI; //DMI adjustment factor for mud depth
double DMI; //dry matter intake
double MudME; //mud adjustment factor for external insulation
double HideME; //hide depth adjustment for external insulation
double EI; //external insulation
double TI; //tissue (internal) insulation
double IN; //total insulation (degrees C/ Mcal m^2 day)
double PN; //adjustment of NEm for previous nutrition
double NEmr; //animal requirement for net energy for maintenance (Mcal/day)
double RE; //retained energy (Mcal/day)
double SA; //surface area (m^2)
double HE; //heat production (Mcal/day)
double LCT; //lower critical temperature
double MEcs; //animal requirement for ME adjusted for cold stress (Mcal/day)
double NEmrcs; //animal requirement for NE for growth adjusted for cold stress
double FFM; //Feed intake to support animal requirement for maintenance (kg/day)
double FFG; //feed intake available to support animal growth (kg/day)
double EFG; //net energy available for growth (Mcal/day)
double SWG = 0; //shrunk weight gain - added to SBW for new total shrunk weight
double EqCW; //equivalent carcass weight (kg)
double CWP = 0; //carcass dressing percent
```

```
double CW = 0; //carcass weight (kg)
double EWG; //empty weight gain (kg/day);
double FIG; //fat in gain
double EBF = 0; //content of empty body fat (%)
double FT = 0; //12th rib backfat thickness
double MS = 0; //marbling score
double REA = 0; //ribeye area
double adjSBW; //shrunk body weight adjusted by EPD
double adjFT; //fat thickness adjusted by EPD
double adjMS; //marbling score adjusted by EPD
double adjREA; //rea adjusted by EPD
double QG; //quality grade
double YG = 0; //yield grade
double REp; //retained energy as protein
double PIG; //protein in gain
double kG; //conversion efficiency of ME to NEg
double AdjNEg; //adjusted NEg

/**********************/
/*GROWTH CALCULATIONS*/
/*********************/
/*get shrunk body weight from full live weight*/
SBW = LIVE_TO_SHRUNK_FACTOR * currentBW;

/*EQ 1*/
/*calculate equivalent shrunk body weight*/
EqSBW = (SRW * SBW) / AFSBW;

/*EQ 2 & 3*/
/*determine DMI adjustment factor for EBF*/
if (EqSBW >= 350) {
  BFAF = 0.7714 + (0.00196 * EqSBW) - (0.00000371 * (Math.pow(EqSBW, 2)));
}
else {
  //no adjustment
  BFAF = 1;
}

/*EQ 4*/
/*calculate current month effective temperature index*/
CETI = 27.88 - (0.456 * Tc) + (0.010754 * (Math.pow(Tc, 2))) -
   (0.4905 * RHc) +
   (0.00088 * (Math.pow(RHc, 2))) + (1.1507 * (1000 / 3600) * WS) -
   (0.126447 * ( (1000 / 3600) * WS) * 2) + (0.019876 * Tc * RHc) -
   (0.046313 * Tc * ( (1000 / 3600) * WS)) + (0.4167 * HRS);

/*EQ 5*/
/*calculate DMI night cooling adjustment*/
DMINC = (119.62 - 0.9708 * CETI) / 100;

/*EQ 6, 7, 8 & 9*/
/*calculate DMI adjustment factor with night cooling*/
if (Tc <= -20) {
  DMIAF = 1.16;
}
else if (Tc > -20 && Tc <= 20) {
```

```
 DMIAF = 1.0433 - 0.0044 * Tc + 0.0001 * Math.pow(Tc, 2);
}
else if (Tc > 20 && Tc <= 28) {
 DMIAF = ( (1 - DMINC) * 0.75 + DMINC) / 100 + 1 + 0.05;
}
else if (Tc > 28) {
 DMIAF = ( (1 - DMINC) * 0.75 + DMINC) / 100 + 1;
}
else {
 DMIAF = 1; //no adjustment
}

/*EQ 10*/
/*calculate DMI adjustment factor for mud depth*/
MudDMI = 1 - 0.01 * Mud;

/*EQ 11 & 12*/
/*calculate dry matter intake - based on age of animal*/
if (ageInDays <= 365) {
 DMI = (( (Math.pow(SBW, 0.75) *
      (0.2435 * NEm - 0.0466 * Math.pow(NEm, 2) - 0.1128)) / NEm) *
    BFAF * DMIAF * MudDMI)*this.adjDMI;
}
else { //ageInDays > 365
 DMI = (( (Math.pow(SBW, 0.75) *
      (0.2435 * NEm - 0.0466 * Math.pow(NEm, 2) - 0.0869)) / NEm) *
    BFAF * DMIAF * MudDMI)*this.adjDMI;
}
/*assign DMI to instance varable*/
this.dmi = DMI;

/*EQ 13, 14, 15, 16, 17 & 18*/
/*calculate animal insulation*/
if (HCCcode <= 2) {
 MudME = 1 - (HCCcode - 1) * 0.2; //eq 13
}
else { //HCCcode > 2
 MudME = 0.8 - (HCCcode - 2) * 0.3; //eq 14
}

HideME = 0.8 + (HideCode - 1) * 0.2; //eq 15

EI = (7.36 - (0.296 * WS) + (2.55 * HD)) * MudME * HideME; //eq 16

TI = 5.25 + 0.75 * BCS; //eq 17

IN = TI + EI; //eq 18

/*EQ 20*/
/*calculate adjustment of basal metabolism NEm requirement for the effects of previous
  nutrition*/
PN = 0.8 + (BCS - 1) * 0.05;

/*EQ 21*/
/*calculate net energy for maintenance*/
NEmr = ((Math.pow(SBW, 0.75)) * ( (a1 * PA) * PN))* this.adjNEmr; //adjust requirements
```

```
 //original equation contains an additional + a2, but is not implemented

/*iterative solution for NEg*/
int difference = 0; //use for while loop evaluation
while(difference == 0) {
 /*EQ 22*/
 /*calculate retained energy*/
 RE = ((DMI - (NEmr / (NEm * IonophoreADGFactor))) * NEg)*this.adjRE; //adjust RE

 /*EQ 23*/
 /*calculate surface area*/
 SA = 0.09 * Math.pow(SBW, 0.67);

 /*EQ 24*/
 /*calculate heat production*/
 HE = ( (ME * DMI) - RE) / SA;

 /*EQ 25*/
 /*calculate lower critical temp*/
 LCT = 39 - (IN * HE * 0.85);

 /*EQ 26, 27, 28 & 29*/
 /*calculate energy required for maintenance due to cold stress*/
 if (LCT > Tc) {
  MEcs = (SA * (LCT - Tc) / IN)*this.adjMEcs; //eq 26
 }
 else { //LCT <= Tc
  MEcs = 0; //eq 27
 } //end
 if (MEcs > 0) {
  NEmrcs = NEmr + (NEm / ME) * MEcs; //eq 28
 }
 else { //MEcs <= 0
  NEmrcs = NEmr; //eq 29
 } //end

 /*EQ 30*/
 /*re-compute feed required for maintenance after accounting for energy requirement for
  cold stress*/
 FFM = NEmrcs / (NEm * IonophoreADGFactor);

 /*EQ 31*/
 /*calculate feed available for growth*/
 FFG = DMI - FFM;

 /*EQ 32*/
 /*calculate energy available for growth*/
 EFG = FFG * NEg;

 /*EQ 33 & 34*/
 /*calculate shrunk weight gain*/
 if (EFG > 0) {
  SWG = 13.91 * (Math.pow(EqSBW, -0.6837)) * (Math.pow(EFG, 0.9116)); //eq 33
 }
 else { //EFG <= 0
  SWG = 0; //eq 34
```

```
    }

    /*EQ 35*/
    /*calculate total shrunk weight - including gain*/
    newSBW = SBW + SWG;
    this.avgBW = newSBW * (1 + (1 - LIVE_TO_SHRUNK_FACTOR)); //update live weight of average
animal

    /*EQ 36, 37, 38*/
    /*calculate carcass weight and dressing percent*/
    EqCW = (0.891 * EqSBW - 32.39) / 1.316; //eq36

    CWP = (EqCW * 100 / EqSBW) * this.adjCWP; //dressing percent - eq 37 - adjust to match
                                  //expected dressing percent

    CW = CWP * SBW / 100; //carcass weight - eq 38

    /*EQ 39*/
    /*caculate empty weight gain*/
    EWG = SWG * 0.956;

    /*EQ 40*/
    /*calculate fat in gain*/
    if (EWG > 0) {
      FIG = 0.123 * RE / EWG - 0.154;
    }
    else { //EWG <= 0
      FIG = 0;
    }

    /*EQ 42 & 43*/
    /*calculate empty body fat weight (kg) and empty body fat percent*/
    this.FAT = this.FAT + FIG * EWG * 0.85; //fat accumulates in current growth object

    //empty body fat (kg)
    EBF = this.FAT * 100 / (SBW * 0.891); //empty body fat (%)

    /*EQ 44*/
    /*calculate yield grade*/
    //YG = -0.604 + 0.127 * EBF; //see below**

    /*EQ 45 & 46 are method 1 to calculate REp and PIG*/

    /*EQ 47*/
    /*method 2*/
    PIG = 0.253908 - 0.0271067 * (RE / EWG);

    /*EQ 48*/
    REp = PIG * EWG * (5.686 / 1000) / RE;

    /*EQ 49*/
    kG = (3 / (4 + 11 * REp));

    /*EQ 50*/
    AdjNEg = kG * ME;
```

```
  /*compare adjusted NEg and original NEg*/
  double diff = Math.abs(AdjNEg - NEg);

  if((diff / NEg) < 0.05) {
    difference = 1;
  }
  NEg = AdjNEg;

} //end while(difference == 0)

/*calculate backfat thickness*/
FT = -1.29286 + (8.53571 * (EBF / 100));

/*calculate marbling score*/
MS = 0.94359 + (17.12018 * (EBF / 100));
//MS = -3.99 + (0.313 * EBF); //from Brethour (2004) JAS

/*calculate REA*/
//from Hamlin et al., 1995 JAS:1713 - intercept adjusted
REA = (-12.05 * this.adjREA) + (0.3 * (currentBW *
     this.LIVE_TO_SHRUNK_FACTOR)) -
    (0.00022 * Math.pow((currentBW *
    this.LIVE_TO_SHRUNK_FACTOR),2)); //used below - uses adjustment factor to match expected
                               //changes in REA

/*calculate YG*/
YG = 2.5 + (2.5 * (FT / 2.54)) + (0.2 * 3) +
    (0.0038 * (CW * 2.2)) -
    (0.32 * (REA / 6.4516));

/*get EPD*/
double sireWT_RRa = sireParent.WT_RRa() ;
double sireWT_RRb = sireParent.WT_RRb() ;
double sireWT_RRc = sireParent.WT_RRc() ;
double sireFT_RRa = sireParent.FT_RRa();
double sireFT_RRb = sireParent.FT_RRb();
double sireFT_RRc = sireParent.FT_RRc();
double sireMS_RRa = sireParent.MS_RRa();
double sireMS_RRb = sireParent.MS_RRb();
double sireMS_RRc = sireParent.MS_RRc();
double sireREA_epd = sireParent.REA_epd();
double damWT_RRa = damParent.WT_RRa();
double damWT_RRb = damParent.WT_RRb();
double damWT_RRc = damParent.WT_RRc();
double damFT_RRa = damParent.FT_RRa();
double damFT_RRb = damParent.FT_RRb();
double damMS_RRa = damParent.MS_RRa();
double damMS_RRb = damParent.MS_RRb();
double damREA_epd = damParent.REA_epd();

/*****Calculate variances (genetic and residual) for WT for current day on feed*****/
/*determine residual variance for current day and sample residual effect*/
Matrix Zi = new Matrix(3,3); //calculate matrix with current age
Zi.set(0,0,1);
Zi.set(0,1,1);
Zi.set(0,2,1);
```

```
Zi.set(1,0,1);
Zi.set(1,1,205);
Zi.set(1,2,Math.pow(Zi.get(1,1),2));
Zi.set(2,0,1);
Zi.set(2,1,currentAnimal.ageInDays());
Zi.set(2,2,Math.pow(Zi.get(2,1),2));
Matrix ZiT = Zi.transpose();
Matrix R = (Zi.times(Rrr)).times(ZiT);

/*determine additive variance for current day and sample dam and Mendelian sampling effects*/
Matrix G = (Zi.times(Grr)).times(ZiT);

/*sample dam + Mendelian sampling and residual effects to get adjustment for phenotype*/
double residualWT = Math.sqrt(R.get(2,2)) * this.rsampWT;
double other_geneticWT = Math.sqrt(G.get(2,2)*0.75) * this.sampWT;

/*calculate EPD for current day on feed*/
double sireWT_EPD = sireWT_RRa + (sireWT_RRb * currentAnimal.ageInDays()) +
   (sireWT_RRc * Math.pow(currentAnimal.ageInDays(),2));
double damWT_EPD = damWT_RRa + (damWT_RRb * currentAnimal.ageInDays()) +
   (damWT_RRc * Math.pow(currentAnimal.ageInDays(),2));

/***for use with RR EPD - when available***/ /***need to calculate other effects***/
//double sireFT_EPD = sireFT_RRa + (sireFT_RRb * currentAnimal.daysOnFeed()) +
   //(sireFT_RRb * Math.pow(currentAnimal.daysOnFeed(),2)); //assuming a quadratic
//double sireMS_EPD = sireMS_RRa + (sireMS_RRb * currentAnimal.daysOnFeed()) +
   //(sireMS_RRb * Math.pow(currentAnimal.daysOnFeed(),2)); //assuming a quadratic
//double damFT_EPD = damFT_RRa + (damFT_RRb * currentAnimal.daysOnFeed()) +
  // (damFT_RRb * Math.pow(currentAnimal.daysOnFeed(),2)); //assuming a quadratic
//double damMS_EPD = damMS_RRa + (damMS_RRb * currentAnimal.daysOnFeed()) +
  // (damMS_RRb * Math.pow(currentAnimal.daysOnFeed(),2)); //assuming a quadratic

/***************************************************************/
/**Phenotype = sireEPD + damEPD + Mendelian sampling + residual***/
/***************************************************************/

/*Adjust current weight based on genetic and resiudal effects*/
adjSBW = (currentBW + sireWT_EPD + damWT_EPD + other_geneticWT +residualWT)
   * LIVE_TO_SHRUNK_FACTOR;

//adjFT = FT + sireFT_EPD + damFT_EPD;
//adjMS = MS + sireMS_EPD + damMS_EPD;

/*************************************************************************/
 /*RANDOM REGRESSION EPD ARE NOT YET AVAILABLE FOR FT AND MS, SO REGULAR
  EPD ARE USED TO ADJUST CARCASS PERFORMANCE EACH DAY.  THE ADJUSTMENTS
  ASSUMES THAT THE EPD ARE CALCULATED AT A DAY NEAR THE DAY OF HARVEST
  FOR ANIMALS SIMULATED IN THE MODEL. ONCE RR EPD ARE AVAILABLE, THE
  FOLLOWING TWO SECTIONS CAN BE REMOVED*/
/******************************************************************************/

/*adjust FT in absence of RR FT EPD*/
/*assuming that FT EPD adjusts FT performance at harvest*/
/*FT EPD is in parameter file as RR intercept EPD for FT*/
if(sireFT_RRa != 0 || sireFT_RRa == 0) { //only use alternate method to calc FT if
                         //EPD is not zero
```

```
    double adjExpFT = expFT + sireFT_RRa + damFT_RRa +this.sampFT + this.rsampFT;
            //adjust FT at harvest by FT EPD and other effects
    double day = currentAnimal.daysOnFeed() - expDOF;

    //try different methods to determine FT
    /*Brethour, 2000 JAS:2055*/
    adjFT = (adjExpFT * Math.exp(0.0117*day)); //MATCHES BEST SO FAR
    //adjFT = adjExpFT * Math.exp(0.0096*day);

    /*Nash etal., 2000 PAS:202*/
    //adjFT = adjExpFT + (0.005 * day);

    /*Hamlin et al., 1995 JAS:1731*/
    //adjFT = adjExpFT - (0.0018*day) + (0.0000021*(Math.pow(day, 2)));

    //adjFT = FT +sireFT_RRa + damFT_RRa + this.sampFT_a + this.rsampFT_a;
    //adjFT = FT;
   }
   else {
    adjFT = FT + this.sampFT + this.rsampFT;
   }

   /*adjust MS in absence of RR MS EPD*/
   /*assuming that MS EPD adjust MS performance at harvest*/
   /*MS EPD is in parameter file as RR intercept EPD for MS*/
   if(sireMS_RRa !=0 || sireMS_RRa == 0) { //only use alternate method to calc MS if
                                   //EPD is not zero

    //use following two lines with a appropriate MS function
    //double adjExpMS = expMS + sireMS_RRa + damMS_RRa + this.sampMS + this.rsampMS;
    //double day = currentAnimal.daysOnFeed() - expDOF;

    //try different methods to determine MS
    /*Brethour, 2000 JAS:2055*/
    //adjMS = 0.000214 * (Math.pow(( Math.pow((adjExpMS - MS) / 0.000214,(1/1.55)) + day),
                  //(1.55))) + MS;

    //adjMS = 0.00000000123642 * (Math.pow(( Math.pow((adjExpMS - MS) /
0.00000000123642,(1/3.42))
     //+ day),(3.42))) + MS;

     /*Bruns et al., 2004 JAS:1315*/
     //adjMS = ((adjExpMS*100) + (1.654 * ((CWP * adjSBW / 100) - expCWT)))/100;

     /*Bruns et al., 2005 JAS:108*/
     //adjMS = ((adjExpMS*100) + (0.74567 * (adjSBW - currentAnimal.bodyWeightAtChoice()
      //                          * LIVE_TO_SHRUNK_FACTOR)))/100;

      //an appropriate adjustment was not found
     adjMS = MS + sireMS_RRa + damMS_RRa + this.sampMS + this.rsampMS;
   }
   else {
   adjMS = MS + this.sampMS + this.rsampMS;
   }

   /*adjust REA by EPD*/
```

```
/*assume that REA EPD adjust REA performance at harvest*/
if(sireREA_epd != 0 || sireREA_epd == 0) { //only use alternate method to calc REA if
                                           //EPD is not zero
 /*adjust expected REA by EPD*/
 double adjExpREA = ((-12.05 * this.adjREA) + (0.3 * (currentAnimal.bodyWeightAtChoice() *
    LIVE_TO_SHRUNK_FACTOR)) -
   (0.00022 * Math.pow((currentAnimal.bodyWeightAtChoice() * LIVE_TO_SHRUNK_FACTOR),2)))
+
   sireREA_epd + damREA_epd + this.sampREA + this.rsampREA;
 double day = currentAnimal.daysOnFeed() - expDOF;

 //try different methods to determine REA
 /*Nash et al., 2000 PAS:202*/
 //adjREA = adjExpREA + (0.38*(day));

 /*Hamlin et al., 1995 JAS:1713*/
 adjREA = adjExpREA + (0.47*day) - (0.0004*Math.pow(day,2));
}
else {
 adjREA = REA + this.sampREA + this.rsampREA;
}

/*re-calculate yield grade with new variables*/
YG = 2.5 + (2.5 * (adjFT / 2.54)) + (0.2 * 3) +
  (0.0038 * ((CWP * adjSBW / 100) * 2.2)) -
  (0.32 * (adjREA / 6.4516));

/*assign quality grade*/
/*Marbling scores:
   1=devoid; 2=practically devoid; 3=traces; 4=slight; 5=small; 6=modest
   7=moderate; 8=slightly abundant; 9=moderately abundant; 10=abundant
 Quality Grades:
   5=below select; 4=select; 3=choice; 2=high choice; 1=prime
 */
//assign quality grade based on adjusted marbling score
if (adjMS < 4.0) {
 QG = 5;
}
else if (adjMS >= 4.0 && adjMS < 5.0) {
 QG = 4;
}
else if (adjMS >= 5.0 && adjMS < 6.0) {
 QG = 3;
}
else if (adjMS >= 6.0 && adjMS < 8.0) {
 QG = 2;
}
else if (adjMS >= 8.0 && adjMS < 11.0) {
 QG = 1;
}
else {
 QG = 0;
}

/*set the inWeight for the current animal adjusted by the wt EPD*/
if(currentAnimal.daysOnFeed() == 1) {
```

```
    currentAnimal.setInWeight(currentAnimal.inWeight() + (sireWT_EPD + damWT_EPD +
                    other_geneticWT +residualWT)* LIVE_TO_SHRUNK_FACTOR);
   }

   /*assign adjusted values to current animal*/
   currentAnimal.setLiveWeight( (adjSBW * (1 + (1 - LIVE_TO_SHRUNK_FACTOR))));
   currentAnimal.setCarcassWeight(CWP * adjSBW / 100);
   currentAnimal.setDressingPercent(CWP);
   currentAnimal.setCarcassFatThickness(adjFT);
   currentAnimal.setCarcassMarbling(adjMS);
   currentAnimal.setREA(adjREA);
   currentAnimal.setQualityGrade(QG);
   currentAnimal.setYieldGrade(YG);

   /*update age of animal and number of days on feed*/
   currentAnimal.updateAgeInDays();
   currentAnimal.updateDaysOnFeed();

  } //end public void grow()

} //end public class Growth
```

Management class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Class to describe feedlot characteristics and management decisions
 * <p>Copyright: June 17, 2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * @author M.A. Cleveland
 * @version 1.0
 */

public class Management {

 /*declare instance variables*/
 private boolean implants; //implants: true or false?
 private double penAreaPerHead; //area in m^2 per head for feedlot pen
 private boolean ionophoreUse; //true = ionophore used

 /**
  * Default constructor
  */
 public Management() {

 } //end public class Management

 /**
  * Constructor for Management
  *
  * @param implants boolean
  * @param penAreaPerHead int
```

```
 * @param ionophoreUse boolean
 */
public Management(boolean implants, double penAreaPerHead, boolean ionophoreUse) {
  /*set initial values for management options*/

  this.implants = implants;
  this.penAreaPerHead = penAreaPerHead;
  this.ionophoreUse = ionophoreUse;

} //end public class Management

/*get methods*/
/**
 * Returns the implant status
 *
 * @return boolean
 */
public boolean implants() {
  return this.implants;

} //end public boolean implants

/**
 * Returns the pen area per head in square meters
 *
 * @return double
 */
public double penAreaPerHead() {
  return this.penAreaPerHead;

} //end public double penAreaPerHead

/**
 * Returns true if an ionophore is used
 *
 * @return boolean
 */
public boolean ionophoreUse() {
  return this.ionophoreUse;

} //end public boolean ionophoreUse()

} //end public class Management
```

Parent class

```
package postweaningdss;
import java.util.Random;

/**
*<p>Title: Package PostweaningDSS</p>
* <p>Description: Describes the genetic potential of a sire or dam for random regression
*   growth and carcass traits, prediction accuracy and the variation associated with each
*   trait, and has methods to stochastically generate EPD accounting for possible change.
```

```
* <p>Copyright: August 23, 2005</p>
* <p>Company: Colorado State University - Department of Animal Sciences</p>
* @author M.A. Cleveland
* @version 1.0

 */
public class Parent {

 /*declare instance variables*/
 private double WT_RRa, WT_RRb, WT_RRc; //intercept and slope EPD for weight
 private double FT_RRa, FT_RRb, FT_RRc; //intercept, slope EPD for backfat
 private double MS_RRa, MS_RRb, MS_RRc; //intercept, slope EPD for marbling
 private double REA_epd; //EPD for ribeye area
 private double accWT, accFT, accMS, accREA; //accuracies for EPD

 /**
  * Default constructor for Parent
  */
 public Parent() {

 } //end public Parent

 /**
  * Constructor for Parent including random regression EPD, accuracies and additive genetic
  * variances</P>
  *    <P>RRa, b or c = random regression intercept, linear or quadratic coefficient</P>
  *    <P>G = additive genetic variance</P>
  *    <P>acc = accuracy of prediction</P>
  *
  * @param WT_RRa double
  * @param WT_RRb double
  * @param WT_RRc double
  * @param FT_RRa double
  * @param FT_RRb double
  * @param FT_RRc double
  * @param MS_RRa double
  * @param MS_RRb double
  * @param MS_RRc double
  * @param REA_epd double
  * @param WT_Ga double
  * @param WT_Gb double
  * @param WT_Gc double
  * @param FT_Ga double
  * @param FT_Gb double
  * @param FT_Gc double
  * @param MS_Ga double
  * @param MS_Gb double
  * @param MS_Gc double
  * @param REA_G double
  * @param accWT double
  * @param accFT double
  * @param accMS double
  * @param accREA double
  */
 public Parent(double WT_RRa, double WT_RRb, double WT_RRc, double FT_RRa, double FT_RRb,
         double FT_RRc, double MS_RRa, double MS_RRb, double MS_RRc, double REA_epd,
```

```
        double WT_Ga, double WT_Gb, double WT_Gc, double FT_Ga, double FT_Gb, double FT_Gc,
        double MS_Ga, double MS_Gb, double MS_Gc, double REA_G,
        double accWT, double accFT, double accMS, double accREA) {

 /*generate new EPD*/
 this.WT_RRa = this.generatedEPD(WT_RRa, WT_Ga, accWT);
 this.WT_RRb = WT_RRb;
 this.WT_RRc = WT_RRc;
 this.FT_RRa = this.generatedEPD(FT_RRa, FT_Ga, accFT);
 this.FT_RRb = FT_RRb;
 this.FT_RRc = FT_RRc;
 this.MS_RRa = this.generatedEPD(MS_RRa, MS_Ga, accMS);
 this.MS_RRb = MS_RRb;
 this.MS_RRc = MS_RRc;
 this.REA_epd = this.generatedEPD(REA_epd, REA_G, accREA);
 this.accWT = accWT;
 this.accFT = accFT;
 this.accMS = accMS;
 this.accREA = accREA;

} //end public Parent

/*get methods*/
/**
 * Returns the random regression weight intercept EPD
 *
 * @return double
 */
public double WT_RRa() {
 return this.WT_RRa;
}

/**
 * Returns the random regression weight linear coefficient EPD
 *
 * @return double
 */
public double WT_RRb() {
 return this.WT_RRb;
}

/**
 * Returns the random regression weight quadratic coefficient EPD
 *
 * @return double
 */
public double WT_RRc() {
 return this.WT_RRc;
}

/**
 * Returns the random regression backfat intercept EPD
 *
 * @return double
 */
public double FT_RRa() {
```

```
 return this.FT_RRa;
}

/**
 * Returns the random regression backfat linear coefficient EPD
 *
 * @return double
 */
public double FT_RRb() {
 return this.FT_RRb;
}

/**
 * Returns the random regression backfat quadratic coefficient EPD
 *
 * @return double
 */
public double FT_RRc() {
 return this.FT_RRc;
}

/**
 * Returns the random regression marbling score intercept EPD
 *
 * @return double
 */
public double MS_RRa() {
 return this.MS_RRa;
}

/**
 * Returns the random regression marbling score linear coefficient EPD
 *
 * @return double
 */
public double MS_RRb() {
 return this.MS_RRb;
}

/**
 * Returns the random regression marbling score quadratic coefficient EPD
 *
 * @return double
 */
public double MS_RRc() {
 return this.MS_RRc;
}

/**
 * Returns the REA EPD
 *
 * @return double
 */
public double REA_epd() {
 return this.REA_epd;
}
```

```
/**
 * Returns the weight EPD accuracy
 *
 * @return double
 */
public double accWT() {
   return this.accWT;
}

/**
 * Returns the backfat EPD accuracy
 *
 * @return double
 */
public double accFT() {
  return this.accFT;
}

/**
 * Returns the marbling score EPD accuracy
 *
 * @return double
 */
public double accMS() {
  return this.accMS;
}

/**
 * Returns the REA EPD accuracy
 *
 * @return double
 */
public double accREA() {
  return this.accREA;
}

/*set methods*/
/**
 * Set the WT_RRa EPD
 *
 * @param epdWT_RRa double
 */
public void setWT_RRa(double epdWT_RRa) {
  this.WT_RRa = epdWT_RRa;
}

/**
 * Set the WT_RRc EPD
 *
 * @param epdWT_RRb double
 */
public void setWT_RRb(double epdWT_RRb) {
  this.WT_RRb = epdWT_RRb;
}
```

```
/**
 * Set the WT_RRc EPD
 *
 * @param epdWT_RRc double
 */
public void setWT_RRc(double epdWT_RRc) {
  this.WT_RRc = epdWT_RRc;
}

/**
 * Set the FT_RRa EPD
 *
 * @param epdFT_RRa double
 */
public void setFT_RRa(double epdFT_RRa) {
  this.FT_RRa = epdFT_RRa;
}

/**
 * Set the FT_RRb EPD
 *
 * @param epdFT_RRb double
 */
public void setFT_RRb(double epdFT_RRb) {
  this.FT_RRb = epdFT_RRb;
}

/**
 * Set the FT_RRc EPD
 *
 * @param epdFT_RRc double
 */
public void setFT_RRc(double epdFT_RRc) {
  this.FT_RRc = epdFT_RRc;
}

/**
 * Set the MS_RRa EPD
 *
 * @param epdFT_RRa double
 */
public void setMS_RRa(double epdMS_RRa) {
  this.MS_RRa = epdMS_RRa;
}

/**
 * Set the MS_RRb EPD
 *
 * @param epdFT_RRb double
 */
public void setMS_RRb(double epdMS_RRb) {
  this.MS_RRb = epdMS_RRb;
}

/**
 * Set the MS_RRc EPD
```

```
  *
  * @param epdFT_RRc double
  */
 public void setMS_RRc(double epdMS_RRc) {
  this.MS_RRc = epdMS_RRc;
 }

 /**
  * Set the REA EPD
  *
  * @param epdREA double
  */
 public void setREAepd(double epdREA) {
  this.REA_epd = epdREA;
 }

 /*other methods*/
 /**
  * Generate EPD from a distribution based on possible change
  *
  * @param epd double
  * @param additiveVar double
  * @param acc double
  * @return double
  */
 public double generatedEPD(double epd, double additiveVar, double acc) {
  double pc = PossibleChange.possibleChange(acc, additiveVar);

    /*generate new EPD*/
    Random generator = new Random();
    return epd += (generator.nextGaussian() * pc);

 } //end public double generatedEPD
} //end public class Parent
```

PossibleChange class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Class to provide method for calculating possible change EPD
 * <p>Copyright: August 23, 2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * @author M.A. Cleveland
 * @version 1.0
 */
import java.util.*;
import Jama.Matrix;

public class PossibleChange {

 /**
  * Default constructor for PossibleChange
```

```
*/
public PossibleChange() {
}

/**
 * Returns possible change (standard error of prediction) EPD, determined by EPD accuracy and
 * additive genetic variance
 * (from BIF)
 *
 * @param additiveVar double
 * @param acc double
 * @return double
 */
public static double possibleChange(double additiveVar, double acc) {
 double pc; //possible change

 /*return possible change of zero if no variance or accuracy*/
 if(additiveVar > 0 && acc > 0) {
  pc = (1 - acc) * (Math.sqrt(additiveVar)) * 0.5;
 }
 else {
  pc = 0;
 }

 return pc;
} //end public static double possibleChange()

/**
 * Returns the adjustment to progney performance accounting for dam and Mendelian sampling
 *
 * @param additiveVar double
 * @return double
 */
public static double samplingAdjustmentDM(double additiveVar) {
 double adj;

 if(additiveVar > 0) {
  Random rand = new Random();
  double sd = (Math.sqrt(additiveVar*0.75)); //take 3/4 of additive variance for standard dev
                             //of random normal distribution
  adj = rand.nextGaussian() * sd; //selects number from random normal distribution with mean 0
                        // and standard dev = sd
 }
 else {
  adj = 0;
 }

 return adj;

} //end public static double samplingAdjustmentDM

/**
 * Returns the residual contribution to an animal's phenotype sampled from a
 * distribution of possible values
 *
 * @param residualVar double
```

```
 * @return double
 */
public static double samplingAdjustmentR(double residualVar) {
 /*where the residual ~N(0,Var(resid))*/
 double adj;

 if(residualVar > 0) {
   Random rand = new Random();
   double sd = Math.sqrt(residualVar);
   adj = rand.nextGaussian() * sd;
 }
 else {
   adj = 0;
 } //end public static double samplingAdjustmentR(double residualVar)

 return adj;
} //end public static double samplingAdjustmentR

public static Matrix samplingAdjustmentDM_RR(Matrix g) {
 /*return a vector of sampling adjustments for progeny performance accounting for dam and
    Mendelian sampling effects - used for quadratic random regression*/
 Matrix L = new Matrix(3,3);
 Matrix q = new Matrix(3,1);
 Matrix adj = new Matrix(3,1);
 Random rand = new Random();

 /*get Cholesky decomposition of g*/
 L = g.chol().getL();

 /*add random normally distruibuted numbers to q*/
 q.set(0,0,(rand.nextGaussian()*Math.sqrt(.75)));
 q.set(1,0,(rand.nextGaussian()*Math.sqrt(.75)));
 q.set(2,0,(rand.nextGaussian()*Math.sqrt(.75)));

 /*get sampling adjustments*/
 adj = L.times(q);

 //Test decomposition
 //Matrix test = new Matrix(3,3);
 //test = L.times(L.transpose());

 return adj;
} //end public static Matrix samplingAdjustmentDM_RR

public static Matrix samplingAdjustmentR_RR(Matrix r) {
   /*return a vector of sampling adjustments for progeny performance accounting for dam and
      Mendelian sampling effects - used for quadratic random regression*/
   Matrix L = new Matrix(3,3);
   Matrix q = new Matrix(3,1);
   Matrix adj = new Matrix(3,1);
   Random rand = new Random();

   /*get Cholesky decomposition of g*/
   L = r.chol().getL();

   /*add random normally distruibuted numbers to q*/
```

```
    q.set(0,0,rand.nextGaussian());
    q.set(1,0,rand.nextGaussian());
    q.set(2,0,rand.nextGaussian());

    /*get sampling adjustments*/
    adj = L.times(q);

    //Test decomposition
    //Matrix test = new Matrix(3,3);
    //test = L.times(L.transpose());

    return adj;
  } //end public static Matrix samplingAdjustmentR_RR
} //end public class PossibleChange
```

SamplingAdjustment class

```
package postweaningdss;

/**
 * <p>Title: Postweaning DSS</p>
 * <p>Description: Describes variables randomly sampled from a normal distribution with
*    specified mean and variance used to simulate progeny phenotype to account for genetic
*    and residual effects.</p>
 * <p>Copyright: June 17,  2005</p>
 * <p>Company: Colorado State University - Department of Animal Sciences</p>
 * @author M.A. Cleveland
 * @version 1.0
 */

public class SamplingAdjustment {
  /**
   * Default constructor
   */
  public SamplingAdjustment() {
  }

  /*variables*/
  private double WT = 0;
  private double FT = 0;
  private double MS = 0;
  private double REA = 0;

  /*set methods*/
  /**
   * Set the adjustment to WT performance
   *
   * @param WT double
   */
  public void setWT(double WT) {
    this.WT = WT;
  }

  /**
```

```
 * Set the adjustment to FT performance
 *
 * @param FT double
 */
public void setFT(double FT) {
  this.FT = FT;
}

/**
 * Set the adjustment to MS performance
 *
 * @param MS double
 */
public void setMS(double MS) {
 this.MS = MS;
}

/**
 * Set the adjustment to REA performance
 *
 * @param REA double
 */
public void setREA(double REA) {
 this.REA = REA;
}

/*get methods*/
/**
 * Returns the adjustment for WT
 *
 * @return double
 */
public double adjWT(){
 return this.WT;
}

/**
 * Returns the adjustment for FT
 *
 * @return double
 */
public double adjFT(){
 return this.FT;
}

/**
 * Returns the adjustment for MS
 *
 * @return double
 */
public double adjMS(){
 return this.MS;
}

/**
 * Returns the adjustment for REA
```

```
   *
   * @return double
   */
  public double adjREA(){
   return this.REA;
  }

} //end public class SamplingAdjustment
```

TerminalAnimal class

```
package postweaningdss;
import java.util.Random;

/**
* <p>Title: Postweaning DSS</p>
* <p>Description: A class to describe growth and carcass characteristics of a
*   feeedlot steer or heifer
* <p>Copyright: August 23, 2005</p>
* <p>Company: Colorado State University - Department of Animal Sciences</p>
* @author M.A. Cleveland
* @version 1.0

 */
public class TerminalAnimal {

  /*declare instance variables*/
  private double LIVE_TO_SHRUNK_FACTOR = 1; //convert live weight to shrunk weight
  private int sex; //1 = heifer, 2 = steer
  private int sexChoice; //1 = all heifer; 2 = all steer; 3 = mixed
  private double inWeight; //weight at start of feeding
  private double ageInDays; //age at start of feeding (days)
  private double daysOnFeed;
  private double bodyWeightAtChoice; //28% EBF
  private double liveWeight; //current live body weight
  private double carcassWeight, carcassFatThickness, carcassMarbling;
  private double dressingPercent;
  private double qualityGrade;
  private double yieldGrade;
  private double rea; //ribeye area
  private int hairCoatCode; //1 = dry and clean; 2 = some mud on lower body; 3 = wet and matted;
                         //4 = covered with wet snow or mud
  private double hairDepth; //in cm
  private int hideCode; //hide depth code: 1 = thin; 2 = average; 3 = thick
  private double bodyConditionScore; //9 possible scores

  /**
   * default constructor
   */
  public TerminalAnimal() {

  } //end public TerminalAnimal

  /**
```

```
 * Constructor for TerminalAnimal
 *
 * @param inWeightMale double
 * @param sdInWeightMale double
 * @param inWeightFemale double
 * @param sdInWeightFemale double
 * @param proportionFemaleReplacements double
 * @param sexChoice int
 * @param ageInDays double
 */
public TerminalAnimal(double inWeightMale, double sdInWeightMale,
                double inWeightFemale, double sdInWeightFemale,
                double proportionFemaleReplacements, int sexChoice, double ageInDays) {
 /*set initial characteristics and composition of animal*/

 double inWeight, sdInWeight;

 /*evaluate sex choice*/
 if(sexChoice == 1) {
  this.sex = 1;
 }
 else if(sexChoice == 2) {
  this.sex = 2;
 }
 else { //generate sex for a mixed pen
   this.sex = generateSex(proportionFemaleReplacements);
 }

 if(this.sex == 1) {
  inWeight = inWeightFemale;
  sdInWeight = sdInWeightFemale;
 }
 else {
  inWeight = inWeightMale;
  sdInWeight = sdInWeightMale;
 }

 /*generate inWeight based on sex*/
 if(sdInWeight > 0) {
  this.inWeight = generateWeight(inWeight, sdInWeight);
 }
 else {
  this.inWeight = inWeight;
 }

 this.daysOnFeed = 0;
 this.liveWeight = this.inWeight;
 this.ageInDays = ageInDays;

} //end public TerminalAnimal


/*set methods*/

/**
 * Increment the age in days by one
```

```
 */
public void updateAgeInDays() {
 this.ageInDays += 1;
}

/**
 * Increment the current number of days on feed from weaning by one
 */
public void updateDaysOnFeed() {
 this.daysOnFeed += 1;
}

/**
 * Set the in weight of the current animal
 *
 * @param inWeight double
 */
public void setInWeight(double inWeight) {
 this.inWeight = inWeight;
}

/**
 * Set the current live weight (kg)
 *
 * @param liveWeight double
 */
public void setLiveWeight(double liveWeight) {
 this.liveWeight = liveWeight;
}

/**
 * Set the body weight expected at the Choice quality grade (approx. 28% EBF) - will be dependent
 * on breed, frame size, sex, etc.
 *
 * @param bodyWeightAtChoice double
 */
public void setBodyWeightAtChoice(double bodyWeightAtChoice) {
 /*assume that entered weight is liveweight and trnaslate to shrunk weight*/
 this.bodyWeightAtChoice = (bodyWeightAtChoice * LIVE_TO_SHRUNK_FACTOR);
}

/**
 * Set the current carcass weight (kg)
 *
 * @param carcassWeight double
 */
public void setCarcassWeight(double carcassWeight) {
 this.carcassWeight = carcassWeight;
}

/**
 * Set the current carcass backfat thickness (cm)
 *
 * @param carcassFatThickness double
 */
public void setCarcassFatThickness(double carcassFatThickness) {
```

```
 this.carcassFatThickness = carcassFatThickness;
}

/**
 * Set the current carcass marbling score
 *
 * @param carcassMarbling double
 */
public void setCarcassMarbling(double carcassMarbling) {
 this.carcassMarbling = carcassMarbling;
}

/**
 * Set the current dressing percent
 *
 * @param dressingPercent double
 */
public void setDressingPercent(double dressingPercent) {
 this.dressingPercent = dressingPercent;
}

/**
 * Set the current carcass USDA quality grade
 *
 * @param qualityGrade int
 */
public void setQualityGrade (double qualityGrade) {
 this.qualityGrade = qualityGrade;
 /*quality grade codes:
    prime = 1
    upper 2/3 choice = 2
    choice = 3
    select = 4
    under select = 5
 */
}

/**
 * Set the current carcass USDA yield grade
 *
 * @param yieldGrade double
 */
public void setYieldGrade(double yieldGrade) {
 this.yieldGrade = yieldGrade;
}

/**
 * Set the current ribeye area (cm^2)
 *
 * @param rea double
 */
public void setREA(double rea) {
 this.rea = rea;
}

/**
```

```
 * Set the hair coat code
 *
 * @param hairCoatCode int
 */
public void setHairCoatCode(int hairCoatCode) {
  this.hairCoatCode = hairCoatCode;
}

/**
 * Set the hair depth (cm)
 *
 * @param hairDepth double
 */
public void setHairDepth(double hairDepth) {
  this.hairDepth = hairDepth;
}

/**
 * Set the hide code
 *
 * @param hideCode int
 */
public void setHideCode(int hideCode) {
  this.hideCode = hideCode;
}

/**
 * Set the body condition score
 *
 * @param bodyConditionScore double
 */
public void setBCS(double bodyConditionScore) {
  this.bodyConditionScore = bodyConditionScore;
}

/**
 * Set the estimated amount of shrink
 *
 * @param shrink double
 */
public void setShrink(double shrink) {
this.LIVE_TO_SHRUNK_FACTOR = 1 - shrink;
}

/*get methods*/
/**
 * Returns the animal's sex
 *
 * @return int
 */
public int sex() {
  return this.sex;
}

/**
 * Returns the current age in days of animal
```

```
 *
 * @return int
 */
public double ageInDays() {
  return this.ageInDays;
}

/**
 * Returns animal in weight (kg)
 *
 * @return double
 */
public double inWeight() {
  return this.inWeight;
}

/**
 * Returns the current number of days on feed from weaning
 *
 * @return int
 */
public double daysOnFeed() {
  return this.daysOnFeed;
}

/**
 * Returns the current animal live weight (kg)
 *
 * @return double
 */
public double liveWeight() {
  return this.liveWeight;
}

/**
 * Returns the body weight at Choice quality grade (approx. 28% EBF) (kg)
 *
 * @return double
 */
public double bodyWeightAtChoice() {
  return this.bodyWeightAtChoice;
}

/**
 * Returns the current animal carcass weight (kg)
 *
 * @return double
 */
public double carcassWeight() {
  return this.carcassWeight;
}

/**
 * Returns the current animal carcass backfat thickness (cm)
 *
 * @return double
```

```
 */
public double carcassFatThickness() {
 return this.carcassFatThickness;
}

/**
 * Returns the current animal carcass marbling score
 *
 * @return double
 */
public double carcassMarbling() {
 return this.carcassMarbling;
}

/**
 * Returns the dressing percent
 *
 * @return double
 */
public double dressingPercent() {
 return this.dressingPercent;
}

/**
 * Returns the current carcass USDA quality grade
 *
 * @return int
 */
public double qualityGrade() {
 return this.qualityGrade;
}

/**
 * Returns the current carcass USDA yield grade
 *
 * @return int
 */
public double yieldGrade() {
 return this.yieldGrade;
}

/**
 * Return the current REA
 *
 * @return double
 */
public double rea() {
 return this.rea;
}

/**
 * Returns the hair coat code
 *
 * @return int
 */
public int hairCoatCode() {
```

```
 return this.hairCoatCode;
}

/**
 * Returns the hair depth (cm)
 *
 * @return double
 */
public double hairDepth() {
 return this.hairDepth;
}

/**
 * Returns the hide code
 *
 * @return int
 */
public int hideCode() {
 return this.hideCode;
}

/**
 * Returns the body condition score
 *
 * @return double
 */
public double bodyConditionScore() {
 return this.bodyConditionScore;
}

/*other methods*/
/**
 * Returns sex of animal as determined by probability of being male or female
 *
 * @param proportionFemaleReplacements double
 * @return int
 */
private int generateSex(double proportionFemaleReplacements) {
 int sex = 0;
 double rand = 0;
 double propAdjMale;
 double propMale = 0;

 if(proportionFemaleReplacements > 0) {
  propAdjMale = proportionFemaleReplacements / 2;
 }
 else {
  propAdjMale = 0;
 }

 /*determine probability of males*/
 propMale = 0.5 * (1 + propAdjMale);

 /*select sex with greater probability of male if female replacements are kept*/
 /*assumes there is always an equal or greater chance of selecting male*/
 Random generator = new Random();
```

```
    rand = generator.nextDouble();

    if(rand < propMale) {
      /*animal is male*/
      sex = 2;
    }
    else {
      /*animal is female*/
      sex = 1;
    }
    return sex;
  } //private int generateSex()

  /**
   * Returns in weight of animal from a distribtuion of values around the mean in weight, having a
   * particular standard deviation
   *
   * @param inWeight double
   * @param sdInWeight double
   * @return double
   */
  private double generateWeight(double inWeight, double sdInWeight) {
    double weight = 0;

    /*select wwt from normal distribution around averageWWT*/
    Random generator = new Random();
    weight = inWeight + (sdInWeight * generator.nextGaussian());

    return weight;
  } //end private double generateWeight()

} //end public class TerminalAnimal
```

Matrix class

(downloaded from http://math.nist.gov/javanumerics/jama/ on August 30, 2005)

```
package Jama;

import java.text.NumberFormat;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.Locale;
import java.text.FieldPosition;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.StreamTokenizer;
import Jama.util.*;

/**
   Jama = Java Matrix class.
<P>
   The Java Matrix Class provides the fundamental operations of numerical
   linear algebra.  Various constructors create Matrices from two dimensional
```

```
  arrays of double precision floating point numbers. Various "gets" and
  "sets" provide access to submatrices and matrix elements. Several methods
  implement basic matrix arithmetic, including matrix addition and
  multiplication, matrix norms, and element-by-element array operations.
  Methods for reading and printing matrices are also included. All the
  operations in this version of the Matrix Class involve real matrices.
  Complex matrices may be handled in a future version.
<P>
  Five fundamental matrix decompositions, which consist of pairs or triples
  of matrices, permutation vectors, and the like, produce results in five
  decomposition classes. These decompositions are accessed by the Matrix
  class to compute solutions of simultaneous linear equations, determinants,
  inverses and other matrix functions. The five decompositions are:
<P><UL>
  <LI>Cholesky Decomposition of symmetric, positive definite matrices.
  <LI>LU Decomposition of rectangular matrices.
  <LI>QR Decomposition of rectangular matrices.
  <LI>Singular Value Decomposition of rectangular matrices.
  <LI>Eigenvalue Decomposition of both symmetric and nonsymmetric square matrices.
</UL>
<DL>
<DT><B>Example of use:</B></DT>
<P>
<DD>Solve a linear system A x = b and compute the residual norm, ||b - A x||.
<P><PRE>
    double[][] vals = {{1.,2.,3},{4.,5.,6.},{7.,8.,10.}};
    Matrix A = new Matrix(vals);
    Matrix b = Matrix.random(3,1);
    Matrix x = A.solve(b);
    Matrix r = A.times(x).minus(b);
    double rnorm = r.normInf();
</PRE></DD>
</DL>

@author The MathWorks, Inc. and the National Institute of Standards and Technology.
@version 5 August 1998
*/

public class Matrix implements Cloneable, java.io.Serializable {

/* ------------------------
  Class variables
 * ------------------------ */

  /** Array for internal storage of elements.
  @serial internal array storage.
  */
  private double[][] A;

  /** Row and column dimensions.
  @serial row dimension.
  @serial column dimension.
  */
  private int m, n;

/* ------------------------
```

```
Constructors
* ------------------------ */

/** Construct an m-by-n matrix of zeros.
@param m    Number of rows.
@param n    Number of colums.
*/

public Matrix (int m, int n) {
  this.m = m;
  this.n = n;
  A = new double[m][n];
}

/** Construct an m-by-n constant matrix.
@param m    Number of rows.
@param n    Number of colums.
@param s    Fill the matrix with this scalar value.
*/

public Matrix (int m, int n, double s) {
  this.m = m;
  this.n = n;
  A = new double[m][n];
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = s;
    }
  }
}

/** Construct a matrix from a 2-D array.
@param A    Two-dimensional array of doubles.
@exception  IllegalArgumentException All rows must have the same length
@see        #constructWithCopy
*/

public Matrix (double[][] A) {
  m = A.length;
  n = A[0].length;
  for (int i = 0; i < m; i++) {
    if (A[i].length != n) {
      throw new IllegalArgumentException("All rows must have the same length.");
    }
  }
  this.A = A;
}

/** Construct a matrix quickly without checking arguments.
@param A    Two-dimensional array of doubles.
@param m    Number of rows.
@param n    Number of colums.
*/

public Matrix (double[][] A, int m, int n) {
  this.A = A;
```

```
      this.m = m;
      this.n = n;
   }

   /** Construct a matrix from a one-dimensional packed array
   @param vals One-dimensional array of doubles, packed by columns (ala Fortran).
   @param m    Number of rows.
   @exception  IllegalArgumentException Array length must be a multiple of m.
   */

   public Matrix (double vals[], int m) {
      this.m = m;
      n = (m != 0 ? vals.length/m : 0);
      if (m*n != vals.length) {
         throw new IllegalArgumentException("Array length must be a multiple of m.");
      }
      A = new double[m][n];
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {
            A[i][j] = vals[i+j*m];
         }
      }
   }

/* ------------------------
   Public Methods
 * ------------------------ */

   /** Construct a matrix from a copy of a 2-D array.
   @param A    Two-dimensional array of doubles.
   @exception  IllegalArgumentException All rows must have the same length
   */

   public static Matrix constructWithCopy(double[][] A) {
      int m = A.length;
      int n = A[0].length;
      Matrix X = new Matrix(m,n);
      double[][] C = X.getArray();
      for (int i = 0; i < m; i++) {
         if (A[i].length != n) {
            throw new IllegalArgumentException
               ("All rows must have the same length.");
         }
         for (int j = 0; j < n; j++) {
            C[i][j] = A[i][j];
         }
      }
      return X;
   }

   /** Make a deep copy of a matrix
   */

   public Matrix copy () {
      Matrix X = new Matrix(m,n);
      double[][] C = X.getArray();
```

```
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = A[i][j];
    }
  }
  return X;
}

/** Clone the Matrix object.
*/

public Object clone () {
  return this.copy();
}

/** Access the internal two-dimensional array.
@return     Pointer to the two-dimensional array of matrix elements.
*/

public double[][] getArray () {
  return A;
}

/** Copy the internal two-dimensional array.
@return     Two-dimensional array copy of matrix elements.
*/

public double[][] getArrayCopy () {
  double[][] C = new double[m][n];
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = A[i][j];
    }
  }
  return C;
}

/** Make a one-dimensional column packed copy of the internal array.
@return     Matrix elements packed in a one-dimensional array by columns.
*/

public double[] getColumnPackedCopy () {
  double[] vals = new double[m*n];
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      vals[i+j*m] = A[i][j];
    }
  }
  return vals;
}

/** Make a one-dimensional row packed copy of the internal array.
@return     Matrix elements packed in a one-dimensional array by rows.
*/

public double[] getRowPackedCopy () {
```

```
    double[] vals = new double[m*n];
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
        vals[i*n+j] = A[i][j];
      }
    }
    return vals;
  }

  /** Get row dimension.
  @return     m, the number of rows.
  */

  public int getRowDimension () {
    return m;
  }

  /** Get column dimension.
  @return     n, the number of columns.
  */

  public int getColumnDimension () {
    return n;
  }

  /** Get a single element.
  @param i    Row index.
  @param j    Column index.
  @return     A(i,j)
  @exception  ArrayIndexOutOfBoundsException
  */

  public double get (int i, int j) {
    return A[i][j];
  }

  /** Get a submatrix.
  @param i0   Initial row index
  @param i1   Final row index
  @param j0   Initial column index
  @param j1   Final column index
  @return     A(i0:i1,j0:j1)
  @exception  ArrayIndexOutOfBoundsException Submatrix indices
  */

  public Matrix getMatrix (int i0, int i1, int j0, int j1) {
    Matrix X = new Matrix(i1-i0+1,j1-j0+1);
    double[][] B = X.getArray();
    try {
      for (int i = i0; i <= i1; i++) {
        for (int j = j0; j <= j1; j++) {
          B[i-i0][j-j0] = A[i][j];
        }
      }
    } catch(ArrayIndexOutOfBoundsException e) {
      throw new ArrayIndexOutOfBoundsException("Submatrix indices");
```

```
  }
  return X;
}

/** Get a submatrix.
@param r    Array of row indices.
@param c    Array of column indices.
@return     A(r(:),c(:))
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/

public Matrix getMatrix (int[] r, int[] c) {
  Matrix X = new Matrix(r.length,c.length);
  double[][] B = X.getArray();
  try {
    for (int i = 0; i < r.length; i++) {
      for (int j = 0; j < c.length; j++) {
        B[i][j] = A[r[i]][c[j]];
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
  return X;
}

/** Get a submatrix.
@param i0   Initial row index
@param i1   Final row index
@param c    Array of column indices.
@return     A(i0:i1,c(:))
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/

public Matrix getMatrix (int i0, int i1, int[] c) {
  Matrix X = new Matrix(i1-i0+1,c.length);
  double[][] B = X.getArray();
  try {
    for (int i = i0; i <= i1; i++) {
      for (int j = 0; j < c.length; j++) {
        B[i-i0][j] = A[i][c[j]];
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
  return X;
}

/** Get a submatrix.
@param r    Array of row indices.
@param i0   Initial column index
@param i1   Final column index
@return     A(r(:),j0:j1)
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/
```

```
public Matrix getMatrix (int[] r, int j0, int j1) {
  Matrix X = new Matrix(r.length,j1-j0+1);
  double[][] B = X.getArray();
  try {
    for (int i = 0; i < r.length; i++) {
      for (int j = j0; j <= j1; j++) {
        B[i][j-j0] = A[r[i]][j];
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
  return X;
}

/** Set a single element.
@param i   Row index.
@param j   Column index.
@param s   A(i,j).
@exception  ArrayIndexOutOfBoundsException
*/

public void set (int i, int j, double s) {
  A[i][j] = s;
}

/** Set a submatrix.
@param i0  Initial row index
@param i1  Final row index
@param j0  Initial column index
@param j1  Final column index
@param X   A(i0:i1,j0:j1)
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/

public void setMatrix (int i0, int i1, int j0, int j1, Matrix X) {
  try {
    for (int i = i0; i <= i1; i++) {
      for (int j = j0; j <= j1; j++) {
        A[i][j] = X.get(i-i0,j-j0);
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
}

/** Set a submatrix.
@param r   Array of row indices.
@param c   Array of column indices.
@param X   A(r(:),c(:))
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/

public void setMatrix (int[] r, int[] c, Matrix X) {
```

```
  try {
    for (int i = 0; i < r.length; i++) {
      for (int j = 0; j < c.length; j++) {
        A[r[i]][c[j]] = X.get(i,j);
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
}

/** Set a submatrix.
@param r   Array of row indices.
@param j0   Initial column index
@param j1   Final column index
@param X    A(r(:),j0:j1)
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/

public void setMatrix (int[] r, int j0, int j1, Matrix X) {
  try {
    for (int i = 0; i < r.length; i++) {
      for (int j = j0; j <= j1; j++) {
        A[r[i]][j] = X.get(i,j-j0);
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
}

/** Set a submatrix.
@param i0   Initial row index
@param i1   Final row index
@param c    Array of column indices.
@param X    A(i0:i1,c(:))
@exception  ArrayIndexOutOfBoundsException Submatrix indices
*/

public void setMatrix (int i0, int i1, int[] c, Matrix X) {
  try {
    for (int i = i0; i <= i1; i++) {
      for (int j = 0; j < c.length; j++) {
        A[i][c[j]] = X.get(i-i0,j);
      }
    }
  } catch(ArrayIndexOutOfBoundsException e) {
    throw new ArrayIndexOutOfBoundsException("Submatrix indices");
  }
}

/** Matrix transpose.
@return    A'
*/

public Matrix transpose () {
```

```
  Matrix X = new Matrix(n,m);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[j][i] = A[i][j];
    }
  }
  return X;
}

/** One norm
@return    maximum column sum.
*/

public double norm1 () {
  double f = 0;
  for (int j = 0; j < n; j++) {
    double s = 0;
    for (int i = 0; i < m; i++) {
      s += Math.abs(A[i][j]);
    }
    f = Math.max(f,s);
  }
  return f;
}

/** Two norm
@return    maximum singular value.
*/

public double norm2 () {
  return (new SingularValueDecomposition(this).norm2());
}

/** Infinity norm
@return    maximum row sum.
*/

public double normInf () {
  double f = 0;
  for (int i = 0; i < m; i++) {
    double s = 0;
    for (int j = 0; j < n; j++) {
      s += Math.abs(A[i][j]);
    }
    f = Math.max(f,s);
  }
  return f;
}

/** Frobenius norm
@return    sqrt of sum of squares of all elements.
*/

public double normF () {
  double f = 0;
```

```
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      f = Maths.hypot(f,A[i][j]);
    }
  }
  return f;
}

/** Unary minus
@return   -A
*/

public Matrix uminus () {
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = -A[i][j];
    }
  }
  return X;
}

/** C = A + B
@param B    another matrix
@return     A + B
*/

public Matrix plus (Matrix B) {
  checkMatrixDimensions(B);
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = A[i][j] + B.A[i][j];
    }
  }
  return X;
}

/** A = A + B
@param B    another matrix
@return     A + B
*/

public Matrix plusEquals (Matrix B) {
  checkMatrixDimensions(B);
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = A[i][j] + B.A[i][j];
    }
  }
  return this;
}

/** C = A - B
```

```
@param B    another matrix
@return     A - B
*/

public Matrix minus (Matrix B) {
  checkMatrixDimensions(B);
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = A[i][j] - B.A[i][j];
    }
  }
  return X;
}

/** A = A - B
@param B    another matrix
@return     A - B
*/

public Matrix minusEquals (Matrix B) {
  checkMatrixDimensions(B);
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = A[i][j] - B.A[i][j];
    }
  }
  return this;
}

/** Element-by-element multiplication, C = A.*B
@param B    another matrix
@return     A.*B
*/

public Matrix arrayTimes (Matrix B) {
  checkMatrixDimensions(B);
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = A[i][j] * B.A[i][j];
    }
  }
  return X;
}

/** Element-by-element multiplication in place, A = A.*B
@param B    another matrix
@return     A.*B
*/

public Matrix arrayTimesEquals (Matrix B) {
  checkMatrixDimensions(B);
  for (int i = 0; i < m; i++) {
```

```
    for (int j = 0; j < n; j++) {
      A[i][j] = A[i][j] * B.A[i][j];
    }
  }
  return this;
}

/** Element-by-element right division, C = A./B
@param B    another matrix
@return     A./B
*/

public Matrix arrayRightDivide (Matrix B) {
  checkMatrixDimensions(B);
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = A[i][j] / B.A[i][j];
    }
  }
  return X;
}

/** Element-by-element right division in place, A = A./B
@param B    another matrix
@return     A./B
*/

public Matrix arrayRightDivideEquals (Matrix B) {
  checkMatrixDimensions(B);
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = A[i][j] / B.A[i][j];
    }
  }
  return this;
}

/** Element-by-element left division, C = A.\B
@param B    another matrix
@return     A.\B
*/

public Matrix arrayLeftDivide (Matrix B) {
  checkMatrixDimensions(B);
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = B.A[i][j] / A[i][j];
    }
  }
  return X;
}
```

```
/** Element-by-element left division in place, A = A.\B
@param B    another matrix
@return     A.\B
*/

public Matrix arrayLeftDivideEquals (Matrix B) {
  checkMatrixDimensions(B);
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = B.A[i][j] / A[i][j];
    }
  }
  return this;
}

/** Multiply a matrix by a scalar, C = s*A
@param s    scalar
@return     s*A
*/

public Matrix times (double s) {
  Matrix X = new Matrix(m,n);
  double[][] C = X.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      C[i][j] = s*A[i][j];
    }
  }
  return X;
}

/** Multiply a matrix by a scalar in place, A = s*A
@param s    scalar
@return     replace A by s*A
*/

public Matrix timesEquals (double s) {
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = s*A[i][j];
    }
  }
  return this;
}

/** Linear algebraic matrix multiplication, A * B
@param B    another matrix
@return     Matrix product, A * B
@exception  IllegalArgumentException Matrix inner dimensions must agree.
*/

public Matrix times (Matrix B) {
  if (B.m != n) {
    throw new IllegalArgumentException("Matrix inner dimensions must agree.");
  }
  Matrix X = new Matrix(m,B.n);
```

```
  double[][] C = X.getArray();
  double[] Bcolj = new double[n];
  for (int j = 0; j < B.n; j++) {
    for (int k = 0; k < n; k++) {
      Bcolj[k] = B.A[k][j];
    }
    for (int i = 0; i < m; i++) {
      double[] Arowi = A[i];
      double s = 0;
      for (int k = 0; k < n; k++) {
        s += Arowi[k]*Bcolj[k];
      }
      C[i][j] = s;
    }
  }
  return X;
}

/** LU Decomposition
@return     LUDecomposition
@see LUDecomposition
*/

public LUDecomposition lu () {
  return new LUDecomposition(this);
}

/** QR Decomposition
@return     QRDecomposition
@see QRDecomposition
*/

public QRDecomposition qr () {
  return new QRDecomposition(this);
}

/** Cholesky Decomposition
@return     CholeskyDecomposition
@see CholeskyDecomposition
*/

public CholeskyDecomposition chol () {
  return new CholeskyDecomposition(this);
}

/** Singular Value Decomposition
@return     SingularValueDecomposition
@see SingularValueDecomposition
*/

public SingularValueDecomposition svd () {
  return new SingularValueDecomposition(this);
}

/** Eigenvalue Decomposition
@return     EigenvalueDecomposition
```

```
@see EigenvalueDecomposition
*/

public EigenvalueDecomposition eig () {
  return new EigenvalueDecomposition(this);
}

/** Solve A*X = B
@param B    right hand side
@return     solution if A is square, least squares solution otherwise
*/

public Matrix solve (Matrix B) {
  return (m == n ? (new LUDecomposition(this)).solve(B) :
              (new QRDecomposition(this)).solve(B));
}

/** Solve X*A = B, which is also A'*X' = B'
@param B    right hand side
@return     solution if A is square, least squares solution otherwise.
*/

public Matrix solveTranspose (Matrix B) {
  return transpose().solve(B.transpose());
}

/** Matrix inverse or pseudoinverse
@return     inverse(A) if A is square, pseudoinverse otherwise.
*/

public Matrix inverse () {
  return solve(identity(m,m));
}

/** Matrix determinant
@return     determinant
*/

public double det () {
  return new LUDecomposition(this).det();
}

/** Matrix rank
@return     effective numerical rank, obtained from SVD.
*/

public int rank () {
  return new SingularValueDecomposition(this).rank();
}

/** Matrix condition (2 norm)
@return     ratio of largest to smallest singular value.
*/

public double cond () {
  return new SingularValueDecomposition(this).cond();
```

```
}

/** Matrix trace.
@return     sum of the diagonal elements.
*/

public double trace () {
  double t = 0;
  for (int i = 0; i < Math.min(m,n); i++) {
    t += A[i][i];
  }
  return t;
}

/** Generate matrix with random elements
@param m    Number of rows.
@param n    Number of colums.
@return     An m-by-n matrix with uniformly distributed random elements.
*/

public static Matrix random (int m, int n) {
  Matrix A = new Matrix(m,n);
  double[][] X = A.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      X[i][j] = Math.random();
    }
  }
  return A;
}

/** Generate identity matrix
@param m    Number of rows.
@param n    Number of colums.
@return     An m-by-n matrix with ones on the diagonal and zeros elsewhere.
*/

public static Matrix identity (int m, int n) {
  Matrix A = new Matrix(m,n);
  double[][] X = A.getArray();
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      X[i][j] = (i == j ? 1.0 : 0.0);
    }
  }
  return A;
}


/** Print the matrix to stdout.  Line the elements up in columns
  * with a Fortran-like 'Fw.d' style format.
@param w    Column width.
@param d    Number of digits after the decimal.
*/

public void print (int w, int d) {
```

```
  print(new PrintWriter(System.out,true),w,d); }

/** Print the matrix to the output stream.  Line the elements up in
  * columns with a Fortran-like 'Fw.d' style format.
@param output Output stream.
@param w     Column width.
@param d     Number of digits after the decimal.
*/

public void print (PrintWriter output, int w, int d) {
  DecimalFormat format = new DecimalFormat();
  format.setDecimalFormatSymbols(new DecimalFormatSymbols(Locale.US));
  format.setMinimumIntegerDigits(1);
  format.setMaximumFractionDigits(d);
  format.setMinimumFractionDigits(d);
  format.setGroupingUsed(false);
  print(output,format,w+2);
}

/** Print the matrix to stdout.  Line the elements up in columns.
  * Use the format object, and right justify within columns of width
  * characters.
  * Note that is the matrix is to be read back in, you probably will want
  * to use a NumberFormat that is set to US Locale.
@param format A  Formatting object for individual elements.
@param width    Field width for each column.
@see java.text.DecimalFormat#setDecimalFormatSymbols
*/

public void print (NumberFormat format, int width) {
  print(new PrintWriter(System.out,true),format,width); }

// DecimalFormat is a little disappointing coming from Fortran or C's printf.
// Since it doesn't pad on the left, the elements will come out different
// widths.  Consequently, we'll pass the desired column width in as an
// argument and do the extra padding ourselves.

/** Print the matrix to the output stream.  Line the elements up in columns.
  * Use the format object, and right justify within columns of width
  * characters.
  * Note that is the matrix is to be read back in, you probably will want
  * to use a NumberFormat that is set to US Locale.
@param output the output stream.
@param format A formatting object to format the matrix elements
@param width  Column width.
@see java.text.DecimalFormat#setDecimalFormatSymbols
*/

public void print (PrintWriter output, NumberFormat format, int width) {
  output.println();  // start on new line.
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      String s = format.format(A[i][j]); // format the number
      int padding = Math.max(1,width-s.length()); // At _least_ 1 space
      for (int k = 0; k < padding; k++)
        output.print(' ');
```

```
      output.print(s);
    }
    output.println();
  }
  output.println();  // end with blank line.
}

/** Read a matrix from a stream. The format is the same the print method,
  * so printed matrices can be read back in (provided they were printed using
  * US Locale). Elements are separated by
  * whitespace, all the elements for each row appear on a single line,
  * the last row is followed by a blank line.
@param input the input stream.
*/

public static Matrix read (BufferedReader input) throws java.io.IOException {
  StreamTokenizer tokenizer= new StreamTokenizer(input);

  // Although StreamTokenizer will parse numbers, it doesn't recognize
  // scientific notation (E or D); however, Double.valueOf does.
  // The strategy here is to disable StreamTokenizer's number parsing.
  // We'll only get whitespace delimited words, EOL's and EOF's.
  // These words should all be numbers, for Double.valueOf to parse.

  tokenizer.resetSyntax();
  tokenizer.wordChars(0,255);
  tokenizer.whitespaceChars(0, ' ');
  tokenizer.eolIsSignificant(true);
  java.util.Vector v = new java.util.Vector();

  // Ignore initial empty lines
  while (tokenizer.nextToken() == StreamTokenizer.TT_EOL);
  if (tokenizer.ttype == StreamTokenizer.TT_EOF)
       throw new java.io.IOException("Unexpected EOF on matrix read.");
  do {
    v.addElement(Double.valueOf(tokenizer.sval)); // Read & store 1st row.
  } while (tokenizer.nextToken() == StreamTokenizer.TT_WORD);

  int n = v.size();  // Now we've got the number of columns!
  double row[] = new double[n];
  for (int j=0; j<n; j++)  // extract the elements of the 1st row.
    row[j]=((Double)v.elementAt(j)).doubleValue();
  v.removeAllElements();
  v.addElement(row);  // Start storing rows instead of columns.
  while (tokenizer.nextToken() == StreamTokenizer.TT_WORD) {
    // While non-empty lines
    v.addElement(row = new double[n]);
    int j = 0;
    do {
      if (j >= n) throw new java.io.IOException
        ("Row " + v.size() + " is too long.");
      row[j++] = Double.valueOf(tokenizer.sval).doubleValue();
    } while (tokenizer.nextToken() == StreamTokenizer.TT_WORD);
    if (j < n) throw new java.io.IOException
      ("Row " + v.size() + " is too short.");
  }
```

```
      int m = v.size(); // Now we've got the number of rows.
      double[][] A = new double[m][];
      v.copyInto(A); // copy the rows out of the vector
      return new Matrix(A);
    }


 /* ------------------------
    Private Methods
  * ------------------------ */

    /** Check if size(A) == size(B) **/

    private void checkMatrixDimensions (Matrix B) {
      if (B.m != m || B.n != n) {
        throw new IllegalArgumentException("Matrix dimensions must agree.");
      }
    }

 }
```

INIFile class

```
/*------------------------------------------------------------------------------
 * PACKAGE: com.freeware.inifiles
 * FILE   : iniFile.java
 * CREATED: Jun 30, 2004
 * AUTHOR : Prasad P. Khandekar
 *------------------------------------------------------------------------------
 * Change Log:
 * 05/07/2004   - Added support for date time formats.
 *              Added support for environment variables.
 * 07/07/2004   - Added support for data type specific getters and setters.
 *              Updated main method to reflect above changes.
 * 26/08/2004   - Added support for section level and property level comments.
 *              Introduction of seperate class for property values.
 *              Added addSection method.
 *              Sections and properties now retail their order (LinkedHashMap)
 *              Method implementation changes.
 *-----------------------------------------------------------------------------*/
package com.freeware.inifiles;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.Writer;
import java.sql.Timestamp;
import java.text.DateFormat;
import java.text.ParseException;
```

```
import java.text.SimpleDateFormat;
import java.util.Collections;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.NoSuchElementException;
import java.util.Properties;
import java.util.Set;

/**
 * INIFile class provides methods for manipulating (Read/Write) windows ini files.
 *
 * @author Prasad P. Khandekar
 * @version 1.0
 * @since 1.0
 */
public final class INIFile
{
  /** Variable to represent the date format */
  private String mstrDateFmt = "yyyy-MM-dd";

  /** Variable to represent the timestamp format */
  private String mstrTimeStampFmt = "yyyy-MM-dd HH:mm:ss";

  /** Variable to denote the successfull load operation. */
  private boolean mblnLoaded = false;

  /** Variable to hold the ini file name and full path */
  private String mstrFile;

  /** Variable to hold the sections in an ini file. */
  private LinkedHashMap mhmapSections;

  /** Variable to hold environment variables **/
  private Properties mpropEnv;

  /**
   * Create a iniFile object from the file named in the parameter.
   * @param pstrPathAndName The full path and name of the ini file to be used.
   */
  public INIFile(String pstrPathAndName)
  {
    this.mpropEnv = getEnvVars();
    this.mhmapSections = new LinkedHashMap();
    this.mstrFile = pstrPathAndName;
    // Load the specified INI file.
    if (checkFile(pstrPathAndName)) loadFile();
  }

/*------------------------------------------------------------------------------
 * Getters
------------------------------------------------------------------------------*/
  /**
   * Returns the ini file name being used.
   * @return the INI file name.
```

```
 */
public String getFileName()
{
   return this.mstrFile;
}

/**
 * Returns the specified string property from the specified section.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be retrieved.
 * @return the string property value.
 */
public String getStringProperty(String pstrSection, String pstrProp)
{
   String      strRet   = null;
   INIProperty objProp  = null;
   INISection  objSec   = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec != null)
   {
      objProp = objSec.getProperty(pstrProp);
      if (objProp != null)
      {
         strRet = objProp.getPropValue();
         objProp = null;
      }
      objSec = null;
   }
   return strRet;
}

/**
 * Returns the specified boolean property from the specified section.
 * This method considers the following values as boolean values.
 * <ol>
 *      <li>YES/yes/Yes - boolean true</li>
 *      <li>NO/no/No  - boolean false</li>
 *      <li>1 - boolean true</li>
 *      <li>0 - boolean false</li>
 *      <li>TRUE/True/true - boolean true</li>
 *      <li>FALSE/False/false - boolean false</li>
 * </ol>
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be retrieved.
 * @return the boolean value
 */
public Boolean getBooleanProperty(String pstrSection, String pstrProp)
{
   boolean     blnRet  = false;
   String      strVal  = null;
   INIProperty objProp = null;
   INISection  objSec  = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec != null)
```

```
  {
    objProp = objSec.getProperty(pstrProp);
    if (objProp != null)
    {
      strVal = objProp.getPropValue().toUpperCase();
      if (strVal.equals("YES") || strVal.equals("TRUE") ||
        strVal.equals("1"))
      {
        blnRet = true;
      }
      objProp = null;
    }
    objSec = null;
  }
  return new Boolean(blnRet);
}

/**
 * Returns the specified integer property from the specified section.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be retrieved.
 * @return the integer property value.
 */
public Integer getIntegerProperty(String pstrSection, String pstrProp)
{
  Integer    intRet  = null;
  String     strVal  = null;
  INIProperty objProp = null;
  INISection  objSec  = null;

  objSec = (INISection) this.mhmapSections.get(pstrSection);
  if (objSec != null)
  {
    objProp = objSec.getProperty(pstrProp);
    try
    {
      if (objProp != null)
      {
        strVal = objProp.getPropValue();
        if (strVal != null) intRet = new Integer(strVal);
      }
    }
    catch (NumberFormatException NFExIgnore)
    {
    }
    finally
    {
      if (objProp != null) objProp = null;
    }
    objSec = null;
  }
  return intRet;
}

/**
 * Returns the specified long property from the specified section.
```

```
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be retrieved.
 * @return the long property value.
 */
public Long getLongProperty(String pstrSection, String pstrProp)
{
  Long       lngRet  = null;
  String     strVal  = null;
  INIProperty objProp = null;
  INISection  objSec  = null;

  objSec = (INISection) this.mhmapSections.get(pstrSection);
  if (objSec != null)
  {
    objProp = objSec.getProperty(pstrProp);
    try
    {
      if (objProp != null)
      {
        strVal = objProp.getPropValue();
        if (strVal != null) lngRet = new Long(strVal);
      }
    }
    catch (NumberFormatException NFExIgnore)
    {
    }
    finally
    {
      if (objProp != null) objProp = null;
    }
    objSec = null;
  }
  return lngRet;
}

/**
 * Returns the specified double property from the specified section.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be retrieved.
 * @return the double property value.
 */
public Double getDoubleProperty(String pstrSection, String pstrProp)
{
  Double     dblRet  = null;
  String     strVal  = null;
  INIProperty objProp = null;
  INISection  objSec  = null;

  objSec = (INISection) this.mhmapSections.get(pstrSection);
  if (objSec != null)
  {
    objProp = objSec.getProperty(pstrProp);
    try
    {
      if (objProp != null)
      {
```

```
            strVal = objProp.getPropValue();
            if (strVal != null) dblRet = new Double(strVal);
         }
      }
      catch (NumberFormatException NFExIgnore)
      {
      }
      finally
      {
         if (objProp != null) objProp = null;
      }
      objSec = null;
   }
   return dblRet;
}

/**
 * Returns the specified date property from the specified section.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be retrieved.
 * @return the date property value.
 */
public Date getDateProperty(String pstrSection, String pstrProp)
{
   Date       dtRet   = null;
   String     strVal  = null;
   DateFormat  dtFmt   = null;
   INIProperty objProp = null;
   INISection  objSec  = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec != null)
   {
      objProp = objSec.getProperty(pstrProp);
      try
      {
         if (objProp != null) strVal = objProp.getPropValue();
         if (strVal != null)
         {
            dtFmt = new SimpleDateFormat(this.mstrDateFmt);
            dtRet = dtFmt.parse(strVal);
         }
      }
      catch (ParseException PExIgnore)
      {
      }
      catch (IllegalArgumentException IAEx)
      {
      }
      finally
      {
         if (objProp != null) objProp = null;
      }
      objSec = null;
   }
   return dtRet;
```

```
    }

    /**
     * Returns the specified date property from the specified section.
     * @param pstrSection the INI section name.
     * @param pstrProp the property to be retrieved.
     * @return the date property value.
     */
    public Date getTimestampProperty(String pstrSection, String pstrProp)
    {
      Timestamp   tsRet   = null;
      Date        dtTmp   = null;
      String      strVal  = null;
      DateFormat  dtFmt   = null;
      INIProperty objProp = null;
      INISection  objSec  = null;

      objSec = (INISection) this.mhmapSections.get(pstrSection);
      if (objSec != null)
      {
        objProp = objSec.getProperty(pstrProp);
        try
        {
          if (objProp != null) strVal = objProp.getPropValue();
          if (strVal != null)
          {
            dtFmt = new SimpleDateFormat(this.mstrDateFmt);
            dtTmp = dtFmt.parse(strVal);
            tsRet = new Timestamp(dtTmp.getTime());
          }
        }
        catch (ParseException PExIgnore)
        {
        }
        catch (IllegalArgumentException IAEx)
        {
        }
        finally
        {
          if (objProp != null) objProp = null;
        }
        objSec = null;
      }
      return tsRet;
    }

  /*------------------------------------------------------------------------------
   * Setters
   ------------------------------------------------------------------------------*/
    /**
     * Sets the comments associated with a section.
     * @param pstrSection the section name
     * @param pstrComments the comments.
     */
    public void addSection(String pstrSection, String pstrComments)
    {
```

```
    INISection objSec   = null;

    objSec = (INISection) this.mhmapSections.get(pstrSection);
    if (objSec == null)
    {
        objSec = new INISection(pstrSection);
        this.mhmapSections.put(pstrSection, objSec);
    }
    objSec.setSecComments(delRemChars(pstrComments));
    objSec = null;
}

/**
 * Sets the specified string property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @pstrVal the string value to be persisted
 */
public void setStringProperty(String pstrSection, String pstrProp,
                              String pstrVal, String pstrComments)
{
    INISection objSec   = null;

    objSec = (INISection) this.mhmapSections.get(pstrSection);
    if (objSec == null)
    {
        objSec = new INISection(pstrSection);
        this.mhmapSections.put(pstrSection, objSec);
    }
    objSec.setProperty(pstrProp, pstrVal, pstrComments);
}

/**
 * Sets the specified boolean property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @param pblnVal the boolean value to be persisted
 */
public void setBooleanProperty(String pstrSection, String pstrProp,
                               boolean pblnVal, String pstrComments)
{
    INISection objSec   = null;

    objSec = (INISection) this.mhmapSections.get(pstrSection);
    if (objSec == null)
    {
        objSec = new INISection(pstrSection);
        this.mhmapSections.put(pstrSection, objSec);
    }
    if (pblnVal)
        objSec.setProperty(pstrProp, "TRUE", pstrComments);
    else
        objSec.setProperty(pstrProp, "FALSE", pstrComments);
}

/**
```

```
 * Sets the specified integer property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @param pintVal the int property to be persisted.
 */
public void setIntegerProperty(String pstrSection, String pstrProp,
                                            int pintVal, String pstrComments)
{
   INISection objSec   = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec == null)
   {
      objSec = new INISection(pstrSection);
      this.mhmapSections.put(pstrSection, objSec);
   }
   objSec.setProperty(pstrProp, Integer.toString(pintVal), pstrComments);
}

/**
 * Sets the specified long property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @param plngVal the long value to be persisted.
 */
public void setLongProperty(String pstrSection, String pstrProp,
                                   long plngVal, String pstrComments)
{
   INISection objSec   = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec == null)
   {
      objSec = new INISection(pstrSection);
      this.mhmapSections.put(pstrSection, objSec);
   }
   objSec.setProperty(pstrProp, Long.toString(plngVal), pstrComments);
}

/**
 * Sets the specified double property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @param pdblVal the double value to be persisted.
 */
public void setDoubleProperty(String pstrSection, String pstrProp,
                                            double pdblVal, String pstrComments)
{
   INISection objSec   = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec == null)
   {
      objSec = new INISection(pstrSection);
      this.mhmapSections.put(pstrSection, objSec);
   }
```

```
    objSec.setProperty(pstrProp, Double.toString(pdblVal), pstrComments);
}

/**
 * Sets the specified java.util.Date property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @param pdtVal the date value to be persisted.
 */
public void setDateProperty(String pstrSection, String pstrProp,
                                    Date pdtVal, String pstrComments)
{
    INISection objSec    = null;

    objSec = (INISection) this.mhmapSections.get(pstrSection);
    if (objSec == null)
    {
        objSec = new INISection(pstrSection);
        this.mhmapSections.put(pstrSection, objSec);
    }
    objSec.setProperty(pstrProp, utilDateToStr(pdtVal, this.mstrDateFmt),
                pstrComments);
}

/**
 * Sets the specified java.sql.Timestamp property.
 * @param pstrSection the INI section name.
 * @param pstrProp the property to be set.
 * @param ptsVal the timestamp value to be persisted.
 */
public void setTimestampProperty(String pstrSection, String pstrProp,
                                            Timestamp ptsVal, String pstrComments)
{
    INISection objSec    = null;

    objSec = (INISection) this.mhmapSections.get(pstrSection);
    if (objSec == null)
    {
        objSec = new INISection(pstrSection);
        this.mhmapSections.put(pstrSection, objSec);
    }
    objSec.setProperty(pstrProp, timeToStr(ptsVal, this.mstrTimeStampFmt),
                pstrComments);
}

/**
 * Sets the format to be used to interpreat date values.
 * @param pstrDtFmt the format string
 * @throws IllegalArgumentException if the if the given pattern is invalid
 */
public void setDateFormat(String pstrDtFmt) throws IllegalArgumentException
{
    if (!checkDateTimeFormat(pstrDtFmt))
        throw new IllegalArgumentException("The specified date pattern is invalid!");
    this.mstrDateFmt = pstrDtFmt;
}
```

```
  /**
   * Sets the format to be used to interpreat timestamp values.
   * @param pstrTSFmt the format string
   * @throws IllegalArgumentException if the if the given pattern is invalid
   */
  public void setTimeStampFormat(String pstrTSFmt)
  {
    if (!checkDateTimeFormat(pstrTSFmt))
      throw new IllegalArgumentException("The specified timestamp pattern is invalid!");
    this.mstrTimeStampFmt = pstrTSFmt;
  }

/*------------------------------------------------------------------------------
 * Public methods
------------------------------------------------------------------------------*/
  public int getTotalSections()
  {
    return this.mhmapSections.size();
  }

  /**
   * Returns a string array containing names of all sections in INI file.
   * @return the string array of section names
   */
  public String[] getAllSectionNames()
  {
    int       iCntr  = 0;
    Iterator  iter   = null;
    String[]  arrRet = null;

    try
    {
      if (this.mhmapSections.size() > 0)
      {
        arrRet = new String[this.mhmapSections.size()];
        for (iter = this.mhmapSections.keySet().iterator();;iter.hasNext())
        {
          arrRet[iCntr] = (String) iter.next();
          iCntr++;
        }
      }
    }
    catch (NoSuchElementException NSEExIgnore)
    {
    }
    finally
    {
      if (iter != null) iter = null;
    }
    return arrRet;
  }

  /**
   * Returns a string array containing names of all the properties under specified section.
   * @param pstrSection the name of the section for which names of properties is to be retrieved.
```

```
 * @return the string array of property names.
 */
public String[] getPropertyNames(String pstrSection)
{
   String[]   arrRet = null;
   INISection objSec = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec != null)
   {
      arrRet = objSec.getPropNames();
      objSec = null;
   }
   return arrRet;
}

/**
 * Returns a map containing all the properties under specified section.
 * @param pstrSection the name of the section for which properties are to be retrieved.
 * @return the map of properties.
 */
public Map getProperties(String pstrSection)
{
   Map        hmRet = null;
   INISection objSec = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec != null)
   {
      hmRet = objSec.getProperties();
      objSec = null;
   }
   return hmRet;
}

/**
 * Removed specified property from the specified section. If the specified
 * section or the property does not exist, does nothing.
 * @param pstrSection the section name.
 * @param pstrProp the name of the property to be removed.
 */
public void removeProperty(String pstrSection, String pstrProp)
{
   INISection objSec = null;

   objSec = (INISection) this.mhmapSections.get(pstrSection);
   if (objSec != null)
   {
      objSec.removeProperty(pstrProp);
        objSec = null;
   }
}

/**
 * Removes the specified section if one exists, otherwise does nothing.
 * @param pstrSection the name of the section to be removed.
```

```
 */
public void removeSection(String pstrSection)
{
   if (this.mhmapSections.containsKey(pstrSection))
      this.mhmapSections.remove(pstrSection);
}

/**
 * Flush changes back to the disk file. If the disk file does not exists then
 * creates the new one.
 */
public boolean save()
{
   boolean   blnRet    = false;
   File      objFile   = null;
   String    strName   = null;
   String    strTemp   = null;
   Iterator  itrSec    = null;
   INISection objSec   = null;
   FileWriter objWriter = null;

   try
   {
      if (this.mhmapSections.size() == 0) return false;
      objFile = new File(this.mstrFile);
      if (objFile.exists()) objFile.delete();
      objWriter = new FileWriter(objFile);
      itrSec = this.mhmapSections.keySet().iterator();
      while (itrSec.hasNext())
      {
         strName = (String) itrSec.next();
         objSec = (INISection) this.mhmapSections.get(strName);
         strTemp = objSec.toString();
         objWriter.write(strTemp);
         objWriter.write("\r\n");
         objSec = null;
      }
      blnRet = true;
   }
   catch (IOException IOExIgnore)
   {
   }
   finally
   {
      if (objWriter != null)
      {
         closeWriter(objWriter);
         objWriter = null;
      }
      if (objFile != null) objFile = null;
      if (itrSec != null) itrSec = null;
   }
   return blnRet;
}

/*------------------------------------------------------------------------
```

```
* Helper functions
*--------------------------------------------------------------------------*/
  /**
   * Procedure to read environment variables.
   * Thanx to http://www.rgagnon.com/howto.html for this implementation.
   */
  private Properties getEnvVars()
  {
    Process p = null;
    Properties envVars = new Properties();

    try
    {
      Runtime r = Runtime.getRuntime();
      String OS = System.getProperty("os.name").toLowerCase();

      if (OS.indexOf("windows 9") > -1)
      {
        p = r.exec("command.com /c set");
      }
      else if ((OS.indexOf("nt") > -1) ||
             (OS.indexOf("windows 2000") > -1) ||
             (OS.indexOf("windows xp") > -1))
      {
        p = r.exec("cmd.exe /c set");
      }
      else
      {
        // our last hope, we assume Unix (thanks to H. Ware for the fix)
        p = r.exec("env");
      }
      BufferedReader br = new BufferedReader(new InputStreamReader(p.getInputStream()));
      String line;
      while((line = br.readLine()) != null)
      {
        int idx = line.indexOf('=');
        String key = line.substring(0, idx);
        String value = line.substring(idx + 1);
        envVars.setProperty(key, value);
      }
    }
    catch (Exception ExIgnore)
    {
    }
    return envVars;
  }

  /**
   * Helper function to check the date time formats.
   * @param pstrDtFmt the date time format string to be checked.
   * @return true for valid date/time format, false otherwise.
   */
  private boolean checkDateTimeFormat(String pstrDtFmt)
  {
    boolean   blnRet = false;
    DateFormat objFmt = null;
```

```
    try
    {
       objFmt = new SimpleDateFormat(pstrDtFmt);
       blnRet = true;
    }
    catch (NullPointerException NPExIgnore)
    {
    }
    catch (IllegalArgumentException IAExIgnore)
    {
    }
    finally
    {
       if (objFmt != null) objFmt = null;
    }
    return blnRet;
 }

 /**
  * Reads the INI file and load its contentens into a section collection after
  * parsing the file line by line.
  */
 private void loadFile()
 {
    int          iPos      = -1;
    String       strLine   = null;
    String       strSection = null;
    String       strRemarks = null;
    BufferedReader objBRdr   = null;
    FileReader   objFRdr   = null;
    INISection   objSec    = null;

    try
    {
       objFRdr = new FileReader(this.mstrFile);
       if (objFRdr != null)
       {
          objBRdr = new BufferedReader(objFRdr);
          if (objBRdr != null)
          {
             while (objBRdr.ready())
             {
                iPos = -1;
                strLine  = null;
                strLine = objBRdr.readLine().trim();
                if (strLine == null)
                {
                }
                else if (strLine.length() == 0)
                {
                }
                else if (strLine.substring(0, 1).equals(";"))
                {
                   if (strRemarks == null)
                      strRemarks = strLine.substring(1);
```

```
               else if (strRemarks.length() == 0)
                  strRemarks = strLine.substring(1);
               else
                  strRemarks = strRemarks + "\r\n" + strLine.substring(1);
            }
            else if (strLine.startsWith("[") && strLine.endsWith("]"))
            {
               // Section start reached create new section
               if (objSec != null)
                  this.mhmapSections.put(strSection.trim(), objSec);
               objSec = null;
               strSection = strLine.substring(1, strLine.length() - 1);
               objSec = new INISection(strSection.trim(), strRemarks);
               strRemarks = null;
            }
            else if ((iPos = strLine.indexOf("=")) > 0 && objSec != null)
            {
               // read the key value pair 012345=789
               objSec.setProperty(strLine.substring(0, iPos).trim(),
                              strLine.substring(iPos + 1).trim(),
                              strRemarks);
               strRemarks = null;
            }
         }
         if (objSec != null)
            this.mhmapSections.put(strSection.trim(), objSec);
         this.mblnLoaded = true;
      }
   }
}
catch (FileNotFoundException FNFExIgnore)
{
   this.mhmapSections.clear();
}
catch (IOException IOExIgnore)
{
   this.mhmapSections.clear();
}
catch (NullPointerException NPExIgnore)
{
   this.mhmapSections.clear();
}
finally
{
   if (objBRdr != null)
   {
      closeReader(objBRdr);
      objBRdr = null;
   }
   if (objFRdr != null)
   {
      closeReader(objFRdr);
      objFRdr = null;
   }
   if (objSec != null) objSec = null;
}
```

```
}

/**
 * Helper function to close a reader object.
 * @param pobjRdr the reader to be closed.
 */
private void closeReader(Reader pobjRdr)
{
  if (pobjRdr == null) return;
  try
  {
    pobjRdr.close();
  }
  catch (IOException IOExIgnore)
  {
  }
}

/**
 * Helper function to close a writer object.
 * @param pobjWriter the writer to be closed.
 */
private void closeWriter(Writer pobjWriter)
{
  if (pobjWriter == null) return;

  try
  {
    pobjWriter.close();
  }
  catch (IOException IOExIgnore)
  {
  }
}

/**
 * Helper method to check the existance of a file.
 * @param the full path and name of the file to be checked.
 * @return true if file exists, false otherwise.
 */
private boolean checkFile(String pstrFile)
{
  boolean blnRet  = false;
  File    objFile = null;

  try
  {
    objFile = new File(pstrFile);
    blnRet = (objFile.exists() && objFile.isFile());
  }
  catch (Exception e)
  {
    blnRet = false;
  }
  finally
  {
```

```
    if (objFile != null) objFile = null;
  }
  return blnRet;
}

/**
 * Converts a java.util.date into String
 * @param pd Date that need to be converted to String
 * @param pstrFmt The date format pattern.
 * @return String
 */
private String utilDateToStr(Date pdt, String pstrFmt)
{
  String strRet = null;
  SimpleDateFormat dtFmt = null;

  try
  {
    dtFmt = new SimpleDateFormat(pstrFmt);
    strRet = dtFmt.format(pdt);
  }
  catch (Exception e)
  {
    strRet = null;
  }
  finally
  {
    if (dtFmt != null) dtFmt = null;
  }
  return strRet;
}

/**
 * Converts the given sql timestamp object to a string representation. The format
 * to be used is to be obtained from the configuration file.
 *
 * @param pobjTS the sql timestamp object to be converted.
 * @param pblnGMT If true formats the string using GMT  timezone
 * otherwise using local timezone.
 * @return the formatted string representation of the timestamp.
 */
private String timeToStr(Timestamp pobjTS, String pstrFmt)
{
  String strRet = null;
  SimpleDateFormat dtFmt = null;

  try
  {
    dtFmt = new SimpleDateFormat(pstrFmt);
    strRet = dtFmt.format(pobjTS);
  }
  catch (IllegalArgumentException  iae)
  {
    strRet = "";
  }
  catch (NullPointerException npe)
```

```
    {
        strRet = "";
    }
    finally
    {
        if (dtFmt != null) dtFmt = null;
    }
    return strRet;
}

/**
 * This function deletes the remark characters ';' from source string
 * @param pstrSrc the source  string
 * @return the converted string
 */
private String delRemChars(String pstrSrc)
{
    int    intPos = 0;

    if (pstrSrc == null) return null;
    while ((intPos = pstrSrc.indexOf(";")) >= 0)
    {
        if (intPos == 0)
            pstrSrc = pstrSrc.substring(intPos + 1);
        else if (intPos > 0)
            pstrSrc = pstrSrc.substring(0, intPos) + pstrSrc.substring(intPos + 1);
    }
    return pstrSrc;
}

/**
 * This function adds a remark character ';' in source string.
 * @param pstrSrc source string
 * @return converted string.
 */
private String addRemChars(String pstrSrc)
{
    int intLen  = 2;
    int intPos  = 0;
    int intPrev = 0;

    String strLeft  = null;
    String strRight = null;

    if (pstrSrc == null) return null;
    while (intPos >= 0)
    {
        intLen = 2;
        intPos = pstrSrc.indexOf("\r\n", intPrev);
        if (intPos < 0)
        {
            intLen = 1;
            intPos = pstrSrc.indexOf("\n", intPrev);
            if (intPos < 0) intPos = pstrSrc.indexOf("\r", intPrev);
        }
        if (intPos == 0)
```

```
         {
            pstrSrc = ";\r\n" + pstrSrc.substring(intPos + intLen);
            intPrev = intPos + intLen + 1;
         }
         else if (intPos > 0)
         {
            strLeft = pstrSrc.substring(0, intPos);
            strRight = pstrSrc.substring(intPos + intLen);
            if (strRight == null)
               pstrSrc = strLeft;
            else if (strRight.length() == 0)
               pstrSrc = strLeft;
            else
               pstrSrc = strLeft + "\r\n;" + strRight;
            intPrev = intPos + intLen + 1;
         }
      }
      if (!pstrSrc.substring(0, 1).equals(";"))
         pstrSrc = ";" + pstrSrc;
      pstrSrc = pstrSrc + "\r\n";
      return pstrSrc;
   }
/*------------------------------------------------------------------------------
 * Main entry point to test the functionality.
 *----------------------------------------------------------------------------*/
   /**
    * The main entry point for testing.
    * @param pstrArgs the command line arguments array if any.
    */
   public static void main(String[] pstrArgs)
   {
      INIFile objINI = null;
      String  strFile = null;

      if (pstrArgs.length == 0) return;

      strFile = pstrArgs[0];
      // Following call will load the strFile if one exists.
      objINI = new INIFile(strFile);

//     objINI.addSection("QADatabase", "QA database connection details\nUsed for QA Testing");
//     objINI.setStringProperty("QADatabase", "SID", "ORCL", null);
//     objINI.setStringProperty("QADatabase", "UserId", "System", null);
//     objINI.setStringProperty("QADatabase", "Password", "Manager", null);
//     objINI.setStringProperty("QADatabase", "HostName", "DBServer", null);
//     objINI.setIntegerProperty("QADatabase", "Port", 1521, null);
//     objINI.setStringProperty("QADatabase", "OracleHome", "%ORACLE_HOME%", null);
//
      //objINI.setSectionComments("Folders", "Directories where generated files are stored");
      objINI.setStringProperty("Folders", "folder1", "G:\\Temp", null);
      objINI.setStringProperty("Folders", "folder2", "G:\\Temp\\Backup", null);

      // Save changes back to strFile.
      objINI.save();
      objINI = null;
   }
```

```
/*---------------------------------------------------------------------------
 * Private class representing the INI Section.
 *---------------------------------------------------------------------------*/
  /**
   * Class to represent the individual ini file section.
   * @author Prasad P. Khandekar
   * @version 1.0
   * @since 1.0
   */
  private class INISection
  {
    /** Variable to hold any comments associated with this section */
    private String mstrComment;

    /** Variable to hold the section name. */
    private String mstrName;

    /** Variable to hold the properties falling under this section. */
    private LinkedHashMap mhmapProps;

    /**
     * Construct a new section object identified by the name specified in
     * parameter.
     * @param pstrSection The new sections name.
     */
    public INISection(String pstrSection)
    {
      this.mstrName =  pstrSection;
      this.mhmapProps = new LinkedHashMap();
    }

    /**
     * Construct a new section object identified by the name specified in
     * parameter and associated comments.
     * @param pstrSection The new sections name.
     * @param pstrComments the comments associated with this section.
     */
    public INISection(String pstrSection, String pstrComments)
    {
      this.mstrName =  pstrSection;
      this.mstrComment = delRemChars(pstrComments);
      this.mhmapProps = new LinkedHashMap();
    }

    /**
     * Returns any comments associated with this section
     * @return the comments
     */
    public String getSecComments()
    {
      return this.mstrComment;
    }

    /**
     * Returns name of the section.
```

```
 * @return Name of the section.
 */
public String getSecName()
{
   return this.mstrName;
}

/**
 * Sets the comments associated with this section.
 * @param pstrComments the comments
 */
public void setSecComments(String pstrComments)
{
   this.mstrComment = delRemChars(pstrComments);
}

/**
 * Sets the section name.
 * @param pstrName the section name.
 */
public void setSecName(String pstrName)
{
   this.mstrName = pstrName;
}

/**
 * Removes specified property value from this section.
 * @param pstrProp The name of the property to be removed.
 */
public void removeProperty(String pstrProp)
{
   if (this.mhmapProps.containsKey(pstrProp))
      this.mhmapProps.remove(pstrProp);
}

/**
 * Creates or modifies the specified property value.
 * @param pstrProp The name of the property to be created or modified.
 * @param pstrValue The new value for the property.
 * @param pstrComments the associated comments
 */
public void setProperty(String pstrProp, String pstrValue, String pstrComments)
{
   this.mhmapProps.put(pstrProp, new INIProperty(pstrProp, pstrValue, pstrComments));
}

/**
 * Returns a map of all properties.
 * @return a map of all properties
 */
public Map getProperties()
{
   return Collections.unmodifiableMap(this.mhmapProps);
}

/**
```

```
 * Returns a string array containing names of all the properties under
 * this section.
 * @return the string array of property names.
 */
public String[] getPropNames()
{
   int     iCntr  = 0;
   String[] arrRet = null;
   Iterator iter   = null;

   try
   {
      if (this.mhmapProps.size() > 0)
      {
         arrRet = new String[this.mhmapProps.size()];
         for (iter = this.mhmapProps.keySet().iterator();iter.hasNext();)
         {
            arrRet[iCntr] = (String) iter.next();
            iCntr++;
         }
      }
   }
   catch (NoSuchElementException NSEExIgnore)
   {
      arrRet = null;
   }
   return arrRet;
}

/**
 * Returns underlying value of the specified property.
 * @param pstrProp the property whose underlying value is to be etrieved.
 * @return the property value.
 */
public INIProperty getProperty(String pstrProp)
{
   INIProperty objRet = null;

   if (this.mhmapProps.containsKey(pstrProp))
      objRet = (INIProperty) this.mhmapProps.get(pstrProp);
   return objRet;
}

/* (non-Javadoc)
 * @see java.lang.Object#toString()
 */
public String toString()
{
   Set         colKeys = null;
   String       strRet  = "";
   Iterator     iter    = null;
   INIProperty  objProp = null;
   StringBuffer objBuf  = new StringBuffer();

   if (this.mstrComment != null)
      objBuf.append(addRemChars(this.mstrComment));
```

```
            objBuf.append("[" + this.mstrName + "]\r\n");
            colKeys = this.mhmapProps.keySet();
            if (colKeys != null)
            {
                iter = colKeys.iterator();
                if (iter != null)
                {
                    while (iter.hasNext())
                    {
                        objProp = (INIProperty) this.mhmapProps.get(iter.next());
                        objBuf.append(objProp.toString());
                        objBuf.append("\r\n");
                        objProp = null;
                    }
                }
            }
            strRet = objBuf.toString();

            objBuf  = null;
            iter    = null;
            colKeys = null;
            return strRet;
        }
    }

/*------------------------------------------------------------------------------
 * Private class representing the INI Property.
 *----------------------------------------------------------------------------*/
    /**
     * This class represents a key value pair called property in an INI file.
     * @author Prasad P. Khandekar
     * @version 1.0
     * @since 1.0
     */
    private class INIProperty
    {
        /** Variable to hold name of this property */
        private String mstrName;
        /** Variable to hold value of this property */
        private String mstrValue;
        /** Variable to hold comments associated with this property */
        private String mstrComments;

        /**
         * Constructor
         * @param pstrName the name of this property.
         * @param pstrValue the value of this property.
         */
        public INIProperty(String pstrName, String pstrValue)
        {
            this.mstrName = pstrName;
            this.mstrValue = pstrValue;
        }

        /**
         * Constructor
```

```
 * @param pstrName the name of this property.
 * @param pstrValue the value of this property.
 * @param pstrComments the comments associated with this property.
 */
public INIProperty(String pstrName, String pstrValue, String pstrComments)
{
   this.mstrName = pstrName;
   this.mstrValue = pstrValue;
   this.mstrComments = delRemChars(pstrComments);
}

/**
 * Returns the string identifier (key part) of this property.
 * @return the string identifier of this property.
 */
public String getPropName()
{
   return this.mstrName;
}

/**
 * Returns value of this property. If value contains a reference to
 * environment avriable then this reference is replaced by actual value
 * before the value is returned.
 * @return the value of this property.
 */
public String getPropValue()
{
   int    intStart = 0;
   int    intEnd   = 0;
   String strVal   = null;
   String strVar   = null;
   String strRet   = null;

   strRet = this.mstrValue;
   intStart = strRet.indexOf("%");
   if (intStart >= 0)
   {
      intEnd = strRet.indexOf("%", intStart + 1);
      strVar = strRet.substring(intStart + 1, intEnd);
      strVal = mpropEnv.getProperty(strVar);
      if (strVal != null)
      {
         strRet = strRet.substring(0, intStart) + strVal +
                      strRet.substring(intEnd + 1);
      }
   }
   return strRet;
}

/**
 * Returns comments associated with this property.
 * @return the associated comments if any.
 */
public String getPropComments()
{
```

```
            return this.mstrComments;
        }

        /**
         * Sets the string identifier (key part) of a property
         * @param pstrName the string identifier of a property
         */
        public void setPropName(String pstrName)
        {
            this.mstrName = pstrName;
        }

        /**
         * Sets the property value
         * @param pstrValue the value for the property
         */
        public void setPropValue(String pstrValue)
        {
            this.mstrValue = pstrValue;
        }

        /**
         * Sets the comments for a property
         * @param pstrComments the comments
         */
        public void setPropComments(String pstrComments)
        {
            this.mstrComments = delRemChars(pstrComments);
        }

        /* (non-Javadoc)
         * @see java.lang.Object#toString()
         */
        public String toString()
        {
            String strRet = "";

            if (this.mstrComments != null)
                strRet = addRemChars(mstrComments);
            strRet = strRet + this.mstrName + " = " + this.mstrValue;
            return strRet;
        }
    }
}
```