THESIS

DEVELOPMENT OF DETAILED PRIME MOVER MODELS AND DISTRIBUTED GENERATION FOR AN ON-BOARD NAVAL POWER SYSTEM TRAINER

Submitted by

Matthew J. Boley

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Masters of Science

Colorado State University

Fort Collins, Colorado

Summer 2012

ABSTRACT

DEVELOPMENT OF DETAILED PRIME MOVER MODELS AND DISTRIBUTED GENERATION FOR AN ON-BOARD NAVAL POWER SYSTEM TRAINER

A power management platform (PMP) has been developed for an electric generation plant on-board a U.S. naval ship. The control hardware and software interface with a Human Machine Interface (HMI) where the sailor can monitor and control the electric plant state. With the implementation of the PMP, there becomes a need to train the sailors how to effectively use the HMI to manage the power plant. A power system trainer was developed with all the physical parts of the power system modeled in software that communicate to the control software, HMI software, and training software. Previous simulation models of the prime movers created in MATLAB® Simulink® (developed at Woodward, Inc. for control code testing purposes) were inadequate to simulate all the signals the control software receives. Therefore, the goal of this research was to increase the accuracy and detail of the existing prime mover models and add detail to the current electrical grid model for use in a power system trainer while maintaining real-time simulation.

This thesis provides an overview encompassing techniques used to model various prime movers, auxiliary systems, and electrical power system grids collected through literary research as well as creative adaptation. For the prime movers, a mean value model (MVM) was developed for the diesel engine as well as a thermodynamic based steam turbine model. A heat transfer model was constructed for an AC synchronous electrical generator with a Totally Enclosed Air to Water Cooled (TEWAC) cooling arrangement. A modular heat exchanger model was implemented and the electrical grid model was expanded to cover all of the electrical elements.

Models now dynamically simulate all the hardware signals in software and the training simulation executes in real-time.

ACKNOWLEDGMENTS

This research is supported by Woodward, Inc. The author would like to thank Brian Baker, Deanna Richardson, Glen Strandvold, and Marty Malenshek of Woodward, Inc. for their assistance. Brain Baker was instrumental in providing guidance on various stages of the project. His help on modeling techniques, system identification of physical systems, as well as learning how to use Woodward's software tools such as GAP™ was invaluable. He was also helpful in understanding the software interface between MATLAB®, NetSim™, and HMI. Deanna Richardson helped in understanding the existing MATLAB® Simulink® code as well as the power system modeling. Glen Strandvold assisted in understanding the technical programming within Simulink® in addition to the interface between MATLAB® and NetSim™. Marty Malenshek was vital in developing the on-board trainer.

The author would also like to thank Michael Burke of The Mathworks, Inc. He was influential in supporting Mathworks' software products and achieving simulation in real-time of the plant model.

The author would finally like to thank Dr. Christopher L. Hagen for his guidance and support throughout the entire project. Without his contributions this project would not have been successful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A	= Area, *m^2*
a	= Coefficient
AF	= Air to fuel ratio
B	= Engine cylinder bore, *m*
C	= Coefficient
C_2	= Correction factor
c	= Heat capacity of solid, *J/(kg-K)*
c_p	= Heat capacity of a fluid const. *p*, *J/(kg-K)*
$\tilde{c}_{u,opt}$	= Optimal corrected turbine blade speed, *$[(m\text{-}r)/s]/(J/kg)^{1/2}$*
c_{us}	= Turbine blade speed, *$(J/kg)^{1/2}$*
$\tilde{c}_{us}$	= Corrected turbine blade speed, *$[(m\text{-}r)/s]/(J/kg)^{1/2}$*
C_v	= Flow coefficient, *$m^3/(Pa^{1/2}\text{-}s)$* or *$GPM/PSI^{1/2}$*
c_v	= Heat capacity of a fluid const. V, *J/(kg-K)*
D	= Diameter of pipe (internal or external), *m*
D_h	= Hydraulic diameter, *m*
E	= Voltage, *V*
e_ω	= Engine thermal dynamic efficiency based on engine speed
F	= Force, *N*
f	= Friction factor in internal flow
g	= Gravitational acceleration, *m/s^2*
Gr	= Grasshof number
H	= Combined generator and prime mover inertia constant, *kW-sec/kVA*
h	= Local heat transfer coefficient, *$W/(m^2\text{-}K)$* or Specific enthalpy, *J/(kg-K)*
$\bar{h}$	= Average heat transfer coefficient, *$W/(m^2\text{-}K)$*
$\dot{H}$	= Enthalpy flow, *W*
H	= Heating Value of fluid, *J/(kg)*
I	= Current, *A*
k	= Thermal conductivity, *W/(m-K)*
K	= Gain
L	= Length
$\dot{m}$	= Mass flow, *kg/s*
m	= Power coefficient or Mass, kg or Fin heat transfer parameter, *1/m*
N	= Number of
n	= Power coefficient
Nu	= Local Nusselt number
$\overline{Nu}$	= Averaged Nusselt Number
P	= Power, *W or* Perimeter, *m*
p	= Pressure, *Pa*
p_{meof}	= Mean effective pressure loss due to friction, *Pa*
Δp	= Pressure difference caused by a flow resistance, *Pa* or *PSI*
pf	= Power factor [AC Power]
Pr	= Prandtl number

$\dot{Q}$	=	Heat flow, *W*
$\dot{Q}''$		Heat flux-Heat flow per unit area, *W/m^2*
R	=	Specific gas constant, *J/(kg-K)* or Resistance, $Ohms$
r	=	Radius, *m*
Re	=	Reynolds number
Ra	=	Rayleigh number
S	=	Engine stroke, *m* or Apparent Power, *VA or* Pitch, *m*
s	=	Laplace transform/frequency domain operator
T	=	Temperature, *K*
ΔT	=	Temperature difference, *K*
t	=	Thickness, m
U	=	Overall heat transfer coefficient, *W/(m^2-K)*
V	=	Volume, *m^3*
$\dot{V}$	=	Volumetric flow, *m^3/s* or *GPM*
v	=	Fluid velocity, *m/s^2*
w	=	Width, m
X	=	Reactance, $Ohms$
x	=	Position/distance, *m* or Fraction, or Coefficient
Y	=	Electrical Admittance, *S*
Z	=	Electrical impedance, $Ohms$
α	=	Thermal diffusivity of fluid, *m^2/s*
β	=	Volumetric thermal expansion coefficient of fluid, *1/K*
ε	=	Emissivity of solid surface
$\xi_{\#}$	=	Curve fit coefficients for pump characteristics
η	=	Efficiency
Θ	=	Rotational Inertia, *(rad/s$^{2)}$/(N-m)*
κ	=	Specific heat ratio c_p/c_v
μ	=	Dynamic viscosity of fluid, *kg/(m-s)* or Coefficient of Friction
$\tilde{\mu}$	=	Corrected mass flow, *kg/s*
ν	=	Kinematic Viscosity of fluid, *m^2/s*
Π	=	Pressure ratio
ρ	=	Fluid density, *kg/m^3*
σ	=	Stefan-Boltzmann's constant, 5.67*x*10^{-8} *W/(m^2-K^4)*
T	=	Torque, *N-m* or p.u.
ϕ	=	Equivalence Ratio
Ω	=	Number of synchronous generator poles
ω	=	Frequency/rotational speed, *rad/s* or *Hz* or *p.u.*
ω_c	=	Compressor rotational speed, *rad/s*
$\widetilde{\omega}$	=	Corrected rotational speed, *rad/s*

Chapter 1: Introduction

Ships must supply their own AC electrical power through generator(s) when at sea similar to land based island mode power generation. Typically there are multiple AC synchronous generators onboard to meet the ships electrical demands and for redundancy reasons. These generators receive mechanical power through prime movers in order to produce electrical power. A power management control system must be used to coordinate the generators output to match the ships electrical demands where the input of the prime mover is managed by a governor. Woodward, Inc. has developed a power management platform (PMP) which integrates control software and hardware. In recent decades, mechanical governors have been replaced by electrical governors. The electronic (power management and governor) controllers now can communicate to displays known as Human Machine Interfaces (HMI). These graphical displays have replaced switches, analog gauges, and other signals that indicate the power system state as well as prime mover specific information [1].

Woodward, Inc. has developed a Power Management Platform (PMP) which integrates control software and hardware to manage small electrical grids. The PMP has been integrated into an existing electric power plant on board a U.S. naval ship. The control hardware and software interface with an HMI where the sailor can monitor and control the electric plant state. This allowed for automation of various power system functions, improved situational awareness, and provided redundancy. Some power system functions done automatically through the PMP are: zero power transfer, protective relaying, auto recovery, real/reactive load sharing.

With the advent of interface and control software/hardware, there becomes an opportunity to train the sailors how to effectively use the HMI to manage the power plant through Woodward's PMP functions. In order to allow for error and avoiding damage to the physical power plant, the physical power plant needs to be separated from the trainer. This reduces the run hours on the engine, saves fuel, and can still have the real engine running and producing power separate from the training simulation. A trainer must be developed with all the physical parts of the power system modeled in software that communicate to the control software, HMI software, and training software.

The software chosen to develop the power plant model is MATLAB® Simulink®. Schematic models can be developed using function blocks in Simulink®'s graphical programming environment. Simulink Coder™ (formerly Real-Time Workshop®) generates C-code from Simulink® schematics and then using an external compiler, compiles it into an executable file. Built in to the physical model is an interface with the control software, termed software in the loop (SIL).

The goal of this project is to increase the resolution of the exiting prime mover models and add detail to current electric power grid model for use in a power system trainer. This involves developing comprehensive dynamic models for diesel engines and steam turbines as well as a complete electric power grid model. All models developed focus toward reaching the objective of simulating the desired outputs of the plant model required for the trainer.

In previous simulations a simple transfer function delay was used to simulate power production from the prime movers to provide power to the generator [1]. The following gives and overview on how a fuel to power transfer function was used and the method employed to create more detailed and dynamically accurate models in both prime mover cases.

- **For the Diesel Engine:** A fuel governor is needed to control the speed of the unit. A proportional-integral-derivative (PID) controller was applied that takes the generator speed error and converts it into fuel demand. These transfer functions predict the performance of the engines with enough accuracy to verify the integrity of the control software [1]. The problem with using a simple transfer function for predicting the performance of the diesel engine is that it ignores the non-linear dynamics associated with turbocharging and thermodynamics of auxiliary systems such as: aftercooler heat exchange circuit, jacket water heat exchange circuit, and lube oil heat exchange circuit. The transients of the turbo spooling up to steady state speed, as well as transient thermodynamic and flow effects within the engine cylinder, create various delays that are non-linear with load [2]. A technique is developed to modify the engine power transfer function to account for the turbocharger dynamics and predict various fluid properties and surface temperatures as they vary with load/time. The turbocharger dynamics are accounted for by comparing steady state air/fuel ratio (in terms of ϕ) to the current air/fuel ratio and compensating the engine power output creating a simulated turbo lag. Further details of how the engine transfer function is altered will be discussed later. Temperatures and pressures at various points within the engine are predicted using a mean value model (MVM) approach.

- **For the Steam Turbine:** A steam valve governor is needed to control the speed of the unit. A PID similar to the diesel engine is used except for it is calculated in the control software versus the model. It uses the same simple transfer function for fuel demand to power transfer function as the diesel. The simple transfer function does not account for the thermodynamics occurring within the turbine to produce power. Pressures, mass flow, and enthalpies are predicted at various stages within the turbine to facilitate the dynamic calculation of shaft power produced by the turbine based on the turbine control valve position. The control valve demand (through a slight delay in control valve position) adjusts the flow resistance through the control valve and controls the mass flow through the turbine and power produced.

Along with altering the way the diesels and steam turbines calculate shaft power/torque, temperatures within the generator must be simulated. A detailed heat transfer model is developed for the cooling of the stator and field windings within the synchronous generator. Through the application of thermodynamics and heat transfer, temperatures such as stator winding temperature, air temperature entering and exiting the generator cooler, seawater coolant temperature, and air temperature at difference lengths along the rotor-stator air gap are calculated.

External cooling and lubrication circuits were modeled for the diesel engine to assist in detailed engine block heat transfer model along with simulation charge air cooling and generator journal bearing lubrication/cooling. An external lubrication circuit was developed for simulating the lubrication and cooling of various journal bearings within the steam turbine, its corresponding reduction gear, and electric generator.

Although parts of the electric grid model were model, it was inadequate to describe the real electric grid on-board the ship. The functionality of the prime mover's electric generator was not altered. Additional load centers, buses, and bus-ties were simulated to allow for the full use of the electric plant in the HMI. Now the sailor has full control of all the breakers in the HMI on the on-board trainer as he/she would on the real ship.

As the complete electric plant model grew in size and complexity over the course of the project, simulation time of the complete model executable increased and eventually the model became so large that Simulink Coder™ was unable to compile the Simulink® model into an executable file. Techniques were developed, with the assistance of The Mathworks, Inc

Consulting Services, to increase the efficiency of the Simulink$^{®}$ model; gearing the model toward achieving real-time simulation of the executable file. Real-time simulation was achieved after converting parts of the plant model into reference subsystem models, converting the general purpose computing language (C-code) to a “target” language computing syntax (TLC-code), and converting the model states from continuous to discrete.

Chapter 2: Background

2.1Mean Value Model

The style of modeling engine combustion varies depending on the application of the model. Two different ways to approach reciprocating internal combustion engine modeling is to take into account the effect of crank angle dynamics known as discrete-event models (DEM) or use continuous-time lumped parameter model known as mean value models (MVM). MVM uses time average values while ignoring crank angle dynamics to simulate combustion within engines. DEMs are useful when instantaneous parameters of combustion need to be model, like in feedfoward control problems involving air/fuel ratio control, torque production, engine control unit modeling, and gas exchange within individual cylinders; though DEMs are sensitive to errors in modeling delays and timing relations. MVMs are valuable in modeling engine phenomena which can be considered on timescales slower than the rotation of the engine crank such as: air intake system, fuel injection/pump system, and engine thermal systems [3]. Essentially in an engine MVM, the engine is treated as a black box that produces torque [4], exhaust temperature, and auxiliary fluid temperatures as a result of combustion. A MVM was chosen due to the fact that many thermodynamic behaviors occur on large time scales and can be averaged over time to create a mean value. Therefore it is advantageous to take the average value of the engine exhaust gas temperature and propagate through the model. MVMs can also be inherently solved at larger time steps than DEMs requiring less computational power.

The two main types of models that fall within the domain of MVM's are Quasi-Steady method and Filling and Emptying method. The Quasi-Steady method assumes values are "steady" though they are continuously changing until true steady state is reached. Quasi-steady method can be applied to the mapping engine performance parameters such as the compressor and turbine maps for a turbocharger. Empting and Filling method is appropriate for modeling manifolds. The differences in the mass flow in and out of the manifold produces pressure and temperature differences that eventually balance at steady state. The limitation of MVM's is that they can only predict the average behavior of engines and ignores the high frequency modes of combustion. The strength of MVM's is that they can be used for real time engine simulation [2].

Steam turbines flows and control valve positions are not discrete, but can be considered inherently continuous with time. Therefore a MVM does not need to be applied to the steam turbines since they are already continuous and nothing needs to be averaged.

2.2 Per Unit Notation

It is common for designers of synchronous generators to give generator parameters in per unit (p.u.) notation [5]. Values such as voltage (E), current (I), torque (T), and admittance (Y) are non-dimensionalized by a reference value usually the base apparent power S_{base} in voltamperes [VA] that the generator produces. The correct choice of reference value simplifies generator calculations and allows for comparison between different generators in a similar fashion as brake mean effective pressure (BMEP). There are two choices for creating a base unit using real power or apparent power. On the mechanical side of the generator using the rated power output P_{base} in watts (W) as a base unit, the per unit torque values approach a value close to unity for rated load, though per unit current is not unity. On the other hand if the base value is apparent power, then per unit current becomes unity at rated load, while per unit torque is not [5].

To accommodate the per unit generator model, the diesel generator producing the mechanical torque must be converted to the appropriate base dimension. Hence, it makes it convenient to create the engine power transfer function in the same per unit as the base dimension. When the coupled shaft is spinning at rated speed the values is unity. Any deviation of the speed from unity should cause the engine speed control to compensate the amount of fuel being injected, therefore increasing/decreasing shaft power and torque as necessary. An example of per unit conversion of power to torque is shown in Eq. (1) for mechanical torque ($\mathrm{T_M}$):

$$\mathrm{T}_M = \frac{\mathrm{P_M}}{\omega}\frac{P_{base}}{S_{base}} \tag{1}$$

where mechanical power (P_M) and generator rotational speed (ω) are per unit and P_{base} is generally S_{base} multiplied by the rated power factor (pf) [1, 5]. The calculation of P_M and fuel control handeling will be discussed further.

2.3 Literature Review: Shipboard Distributed Generation and PMP

Previous research conducted on modeling electric plant on-board naval ships consists of varying degrees of complexity from brief overview to detail equations and simulation results. Literature providing overview of naval electrical power system architecture and electric propulsion are found in [6, 7]. Hansen et al. [8] developed a power system simulation consisting of multiple diesel synchronous generator and distributed generation for ship electrical loads including ship propeller. This model had four generators, three propulsion drives, and a domestic ship electrical load. This model used a per unit reference frame for the speed governor, automatic voltage regulator, synchronous generator, and electric grid model. Sun et al. [9] created a similar model discussed in less detail than Hansen but focused more on the model's results and implementation of the simulation. For this trainer, the shipboard power system is similar to both simulations done by Hansen and Sun since all had prime movers coupled to synchronous AC generators (with excitation voltage regulators) tied to an isolated electric grid. The trainer simulation is different from the others since propulsion of the ship is not driven by electric motors nor modeled.

The power generation architecture on-board the naval ship consists of a single PMP control per prime mover. Each PMP is responsible for managing various load centers and bus-ties within the electric plant as well as taking in prime mover specific information to display on the HMI. The speed control on-board the ship is control two different ways depending on the prime mover. The diesel engines are controlled by the engine manufacture's Electronic Control Unit (ECU) and accepts a speed bias sent from the PMP to increase or decrease speed depending on the situation. The steam turbines are controlled solely by the PMP where steam valve position demand is calculated in the PMP and a hardware signal of the valve position demand is sent to the electronic valve actuator in the steam chest. Each PMP communicates to each other through Ethernet Modbus, which allows for each prime mover to synchronize with the others for precise load sharing and redundant plant control [1, 10]. A schematic of the architecture of a single PMP unit is shown in Figure 1, where the sailor can operate the entire plant from one PMP through the HMI display.

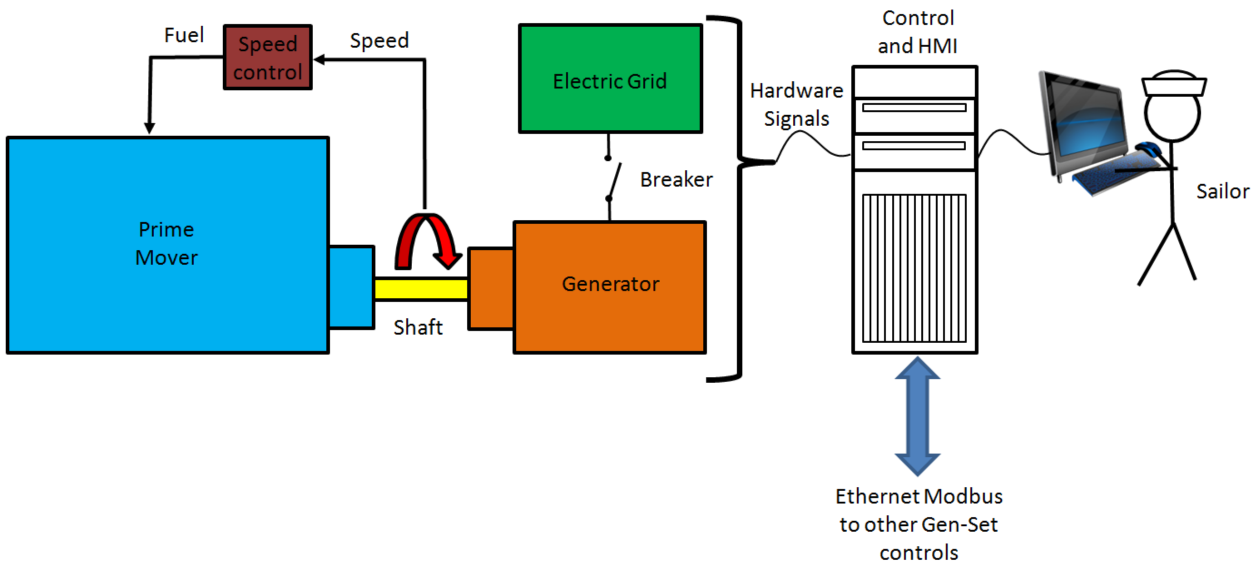


Figure 1. Schematic of Prime Mover and Distributed Generation through PMP Control

For the PMP system, a simple model of the power system was used to validate the control software. The model simulated the electrical power distribution network on-board the ship but didn't model detailed mechanical/thermal systems of the prime mover. The model was able to simulate generators connecting to loads, generators load sharing, zero power transfer, and transfer from ship power to shore power (or utility grid) and shore power to ship power.

2.4 Original Engine/Turbine Model

The engine model that has been used in various projects, including this one, was a simple transfer function for fuel to shaft power generation. This power is then used in a generator model to determine generator speed and alternating current (AC) frequency. Current produced by the generator was fed into the electric grid model. The feedback from the electric grid model to the generator was in terms of bus voltage. Figure 2 shows the block diagram representation of the fuel to power transfer function. This model has been used interchangeably with reciprocating engines and turbines. The compensator for the steam turbines was handled in the control software. However, the diesel engine model had its own compensator within the model and received a speed bias calculated by the control software for load-sharing and synchronizing. The following chapter will discuss the steps taken to separate Figure 2 into two different respective models for a large bore turbocharged diesel engine and a steam turbine along with other dynamic systems needed to be model for development of a power system trainer.

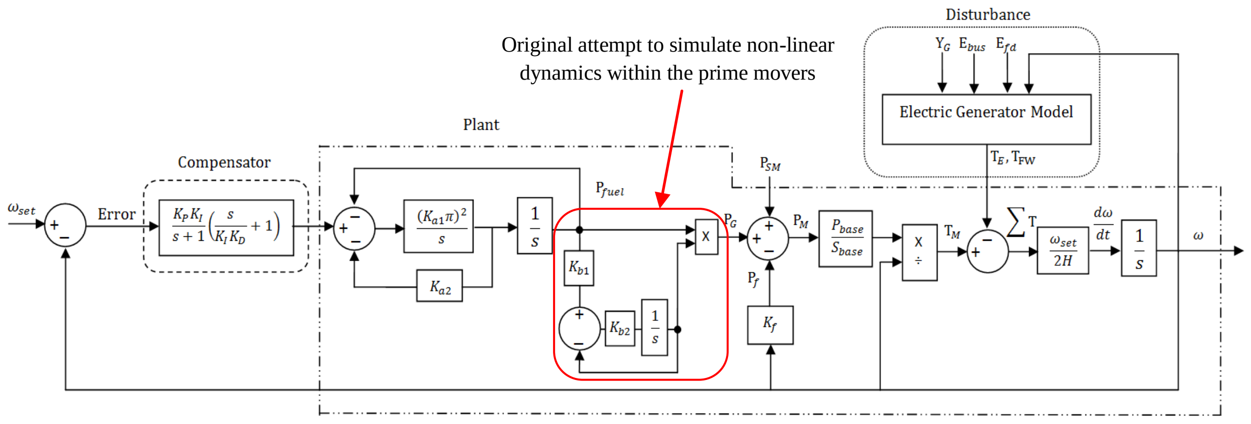


Figure 2. Block Diagram of Speed Control and Torque Production

Chapter 3: Models

This chapter presents the results of the projects research; the goal of developing dynamic relationships for simulating various models. The format of each section within the chapter follow a similar order: first describe/illustrate the physical phenomena, then list the relevant assumptions, next describe the governing equations of that try to quantify the phenomena, and finally state the range of validity of equations used in the model. The figures will help the reader visualize the physical phenomena described by the governing equations as well as graph the relevant inputs and outputs of various equations.

3.1 Proposed Engine Model

This engine model is a MVM which is a lumped parameter model that combines both Quasi-Steady and Filling and Emptying models. Average values of temperature, pressure, mass flow, etc. are used. The model consists of a mix of transfer functions in the s-domain, non-linear algebraic equations, and first order differential equations. The engine fuel and power production is handled by transfer functions with feedback from the generator electrical model. The engines dynamic thermal model takes the power and torque calculated by the transfer function to set the engine load in the thermal model.

To give an idea of the scope of modeling covered in the trainer for the diesel engines, Figure 3 illustrates the main components modeled and relationships between different flows of air, coolant, fuel, and lubrication oil within the diesel engine. The engine model considered in this thesis has two turbochargers one for each engine cylinder bank with a separate cooling circuit for the aftercooler labeled Separate Circuit Aftercooler (SCAC).

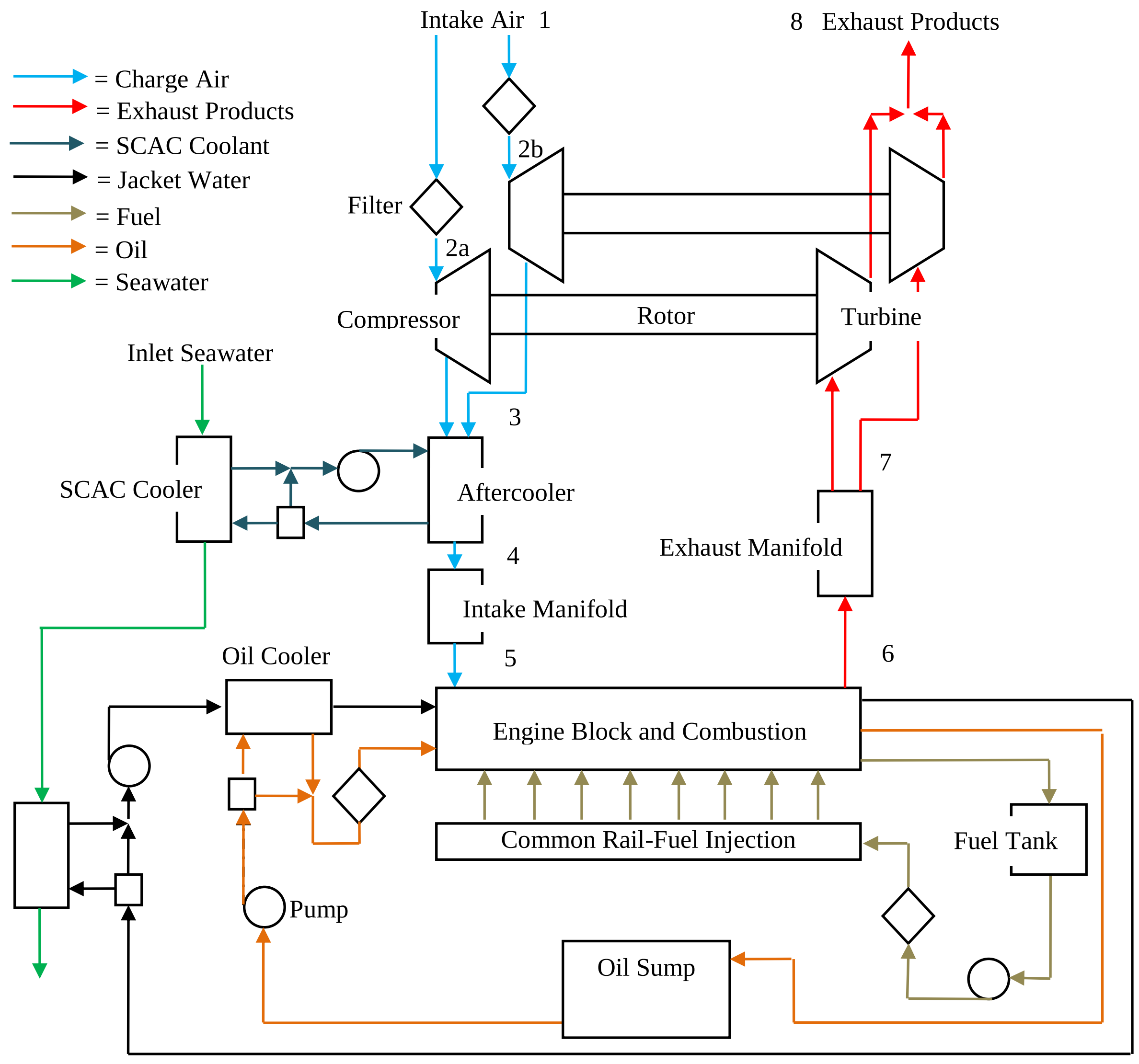


Figure 3. Schematic of Turbocharged Diesel Engine with Coolant, Oil, and Fuel Circuits

3.2 Transfer Function Representation: to estimate power/torque and modification of inherited transfer function

The employment of transfer functions to represent engine power production is useful for generator control purposes, since power production, generator frequency, and fuel injection are the main variables of interest. Figure 4 shows the overall transfer function in block diagram form for the generator frequency output or rotational speed. Frequency and shaft rotational speed in a Ω-pole synchronous machine are related by:

$$\omega = \frac{\Omega}{2}\omega_{shaft} \tag{2}$$

in which ω and coupled shaft speed (ω_{shaft}) are in rad/s. Similarly in p.u. form, ω is unity and ω_{shaft} is 2 when generator is at rated speed. The disturbance portion of Figure 4 is caused by the electric generator model is due to the flux torques between the rotor and stator of the generator caused by the rotating magnetic field (T_E). It is also caused by friction in the generator bearings and power loss in rotor and stator windings, which are lumped into the term T_{FW}. Readers interested in how T_E and T_{FW} are calculated based on generator admittance (Y_G), bus voltage (E_{bus}), and field voltage (E_{fd}) should consult literature [11-14]. Note in Figure 4, the bold portion is what has been changed by the author from Figure 2 specifically the relationship between P_{fuel} and gross power (P_G).

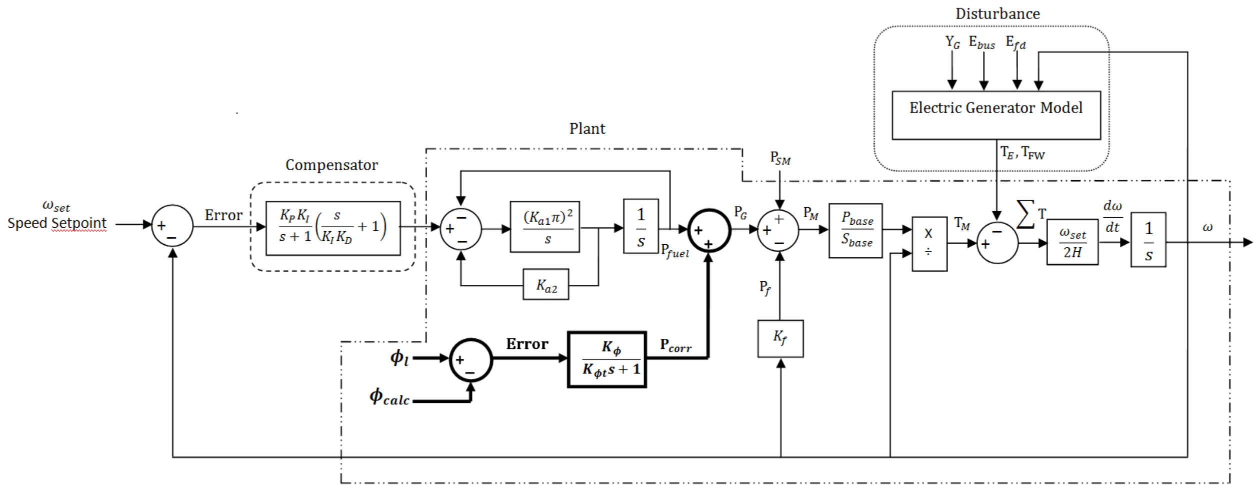


Figure 4. Modified Block Diagram of Speed Control and Torque Production for Reciprocating Engine

The compensator transfer function in Figure 4 is an error amplifier with proportional, integral, and derivative compensation with K_P, K_I, and K_D representing each respective gain. The compensator converts the difference between the actual generator frequency and the frequency set point to a non-dimensional fuel demand value. The fuel demand signal is sent through a summing block where a fuel demand is transformed into non-dimensional P_{fuel}. This transfer function simulates the time delay, known as actuator delay, associated with fuel being injected into engine cylinders and the power extracted from the fuel. For naturally aspirated engines, P_{fuel} equals P_G since the lag related with air being ingested, fuel being injected, and fuel/air mixing are on approximately on the same time scale. Therefore it goes to follow that increase or decrease in fuel injected is directly and linearly proportional to the increase or decrease of power output, with a small time delay. For a turbocharged engine, the transfer function from fuel demand to P_{fuel} does not account for lag in gross power output due to the turbocharger spooling up and non-linear transients for changes in load. As a result, a power correction factor P_{corr} must be applied to P_{fuel} to simulate turbocharger dynamics within the power production transfer function. Equivalence ratio (ϕ) in this model is defined using the air to fuel method in (3):

$$\phi = \frac{AF_{stoich}}{\dot{m}_{\beta} / \dot{m}_{fuel}} \tag{3}$$

where $\dot{m}_{\beta}$ is the mass flow air from intake manifold and $\dot{m}_{fuel}$ is the mass flow of fuel from injectors. Diesel fuel is used for this application and the stoichiometric air to fuel mass ratio (AF_{stoich}) is set at 14.5 [15]. Manufactures have steady state values of equivalence ratio (ϕ_l) tabulated versus engine load. This value ϕ_l is compared to the current value ϕ_{calc} which is calculated in the engine thermal model. This error is then transformed into P_{corr} through the transfer function involving gains K_{ϕ} and $K_{\phi t}$. $K_{\phi t}$ is a time delay factor to account for various delays dealing with fuel and charge air mixing and ϕ realization, mainly due to the distribution of air from the intake manifold to the multiple cylinders of the engine and time scale is on a few hundred milliseconds. The output of this transfer function must be limited to a few percent off of P_{fuel}'s non-dimensional value to create realistic adjustment to P_G. This can be tuned by K_ϕ and also P_{corr} must be limited due to the discontinuity of ϕ being AF_{stoich} divided by zero $\dot{m}_\beta$ over zero $\dot{m}_{fuel}$ at the initial startup of the engine.

To understand how P_{corr} works to create a lag associate with turbocharger dynamics, turbocharger lag must be discussed in detail. For abrupt changes in engine load (i.e. load stepping), the turbocharger is spooled up and rotating at high speed. These lags can be attributed to the change in injected fuel which causes changes in exhaust temperature. Then the changes in exhaust temperatures ultimately lead to changes in turbine power output, which is the driving force of the turbocharger. Inertia of the turbocharger and the dependence on charge air flow on engine power attribute to the resistance to change in turbocharger speed. Initially, in an external electrical load change, the torque absorbed by the compressor closely matches that of the torque generated by the turbine. Then as the engine produces hotter or cooler exhaust gases(depending on load change direction) the torque generated by the turbine changes which changes the speed, which then changes charge air flow rate and pressure. This process feeds back, until again compressor and inertia balance out turbine torque.

P_{corr} adjusts power output of the naturally aspirated transfer function P_{fuel} to simulate the turbocharger lags discussed earlier. At startup it over-predicts P_G. Then when the turbocharger overshoots P_G. P_{corr} fills in these inaccuracies by adding or subtracting power from P_{fuel} to simulate the transient P_G produced. Since ϕ_{calc}, (calculated in the engine MVM) encompasses $\dot{m}_\beta$(which is a function of p_5) and $\dot{m}_{fuel}$, which are the main factors that determining engine power production. It is a good measure of engine performance and determining factor for the engine reaching steady state. Hence, the behavior of ϕ_{calc} a good candidate to most nearly simulate the effects of turbo delay that P_{corr} intends to create with the processing of the error between ϕ_l and ϕ_{calc}.

3.3 Engine Model

In Figure 5, the governor model supplies model variables to the electric generator model (not discussed here but can be deduced from [11-13]) and to the engine MVM. P_G, T_M, P_{fuel}, and ω are inputs to the engine MVM, while ϕ_l and ϕ_{calc} are feedbacks to the governor model.

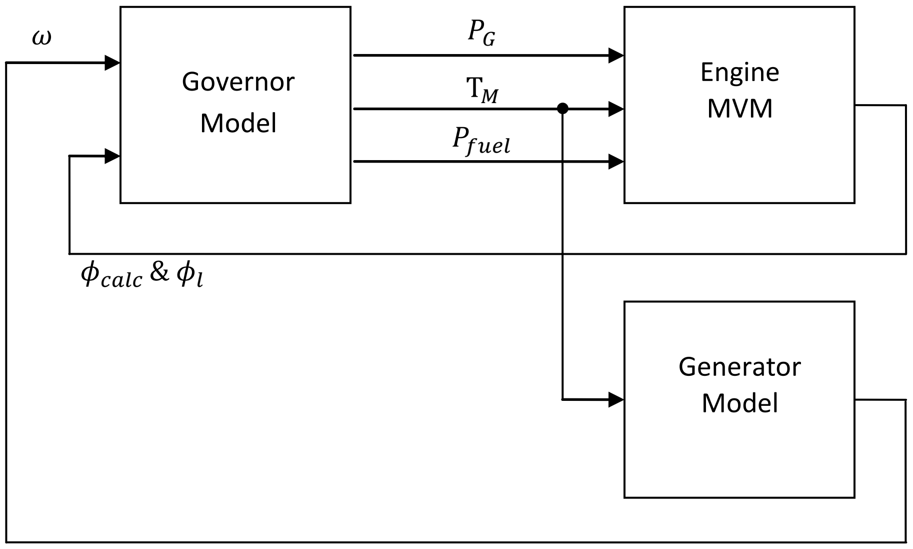


Figure 5. Flow of Variables between Governor Model, MVM Engine Model, and Electrical Generator Model

Air Intake

The intake air filter can be modeled as internal incompressible flow (incompressible since Mach Number $(Ma) < 0.3$ [16]) with a resistance that causes pressure drop. The general equation used to model flow and pressure drop is governed by [17]:

$$\dot{V} = C_v\sqrt{\Delta p} \tag{4}$$

The flow coefficient (C_v) is arbitrary value and can be easily found if volume flow ($\dot{V}$) and pressure drop (Δp) are known and is convenient when trying to fit data. Since in general, it is assumed $\dot{V}$ varies as a function of the square root of Δp; it simulates internal incompressible flow. If using numbering notation in Figure 3 and assuming states 2a and 2b are equal, the pressure drop for the air intake filter becomes:

$$p_2 - p_1 = \left(\frac{\dot{V}_{12}}{C_{v,12}}\right)^2 \tag{5}$$

Turbocharger-Compressor

Steady-state compressor performance is usually characterized by turbocharger manufacturers and formulated into two dimensional compressor maps, with level sets of compressor isentropic efficiency and turbocharger rotor speed. These parameters are related by:

$$\tilde{\mu}_c = f(\widetilde{\omega}_c, \Pi_c) \tag{6}$$

$$\eta_c = f(\widetilde{\omega}_c, \Pi_c) \tag{7}$$

Manufacturers commonly define (using Figure 3 notation) [18]:

$$\tilde{\mu}_c = \dot{m}_c \frac{p_{2,std}}{p_2} \sqrt{\frac{T_2}{T_{2,std}}} \tag{8}$$

$$\widetilde{\omega}_c = \omega_c \sqrt{\frac{T_{2,std}}{T_2}} \tag{9}$$

$$\Pi_c = \frac{p_3}{p_2} \tag{10}$$

where $\tilde{\mu}_c$ is the corrected compressor mass flow rate, η_c is the compressor efficiency, $\dot{m}_c$ is actual compressor mass flow rate, $p_{2,std}$ is the pressure at Standard Pressure and Temperature (STP from EPA [19]) conditions, p_2 is the pressure at the inlet of the compressor, T_2 is the compressor inlet temperature, $T_{2,std}$ is the temperature at STP, $\tilde{\omega}_c$ is the corrected compressor rotational speed, ω_c is the actual compressor rotational speed, Π_c is the compressor pressure ratio, and p_3 is the compressor exit pressure. Figure 6 illustrates typical compressor map. Mass flow through the compressor is found by starting at Π_c (point 1), going across to where Π_c and $\tilde{\omega}_c$ (point 2) meet, η_c and $\dot{\mu}_c$ (points 3 and 4 respectively) can be directly deduced. The compressor map was digitized where linear interpolation and extrapolation were used to find points 3 and 4.

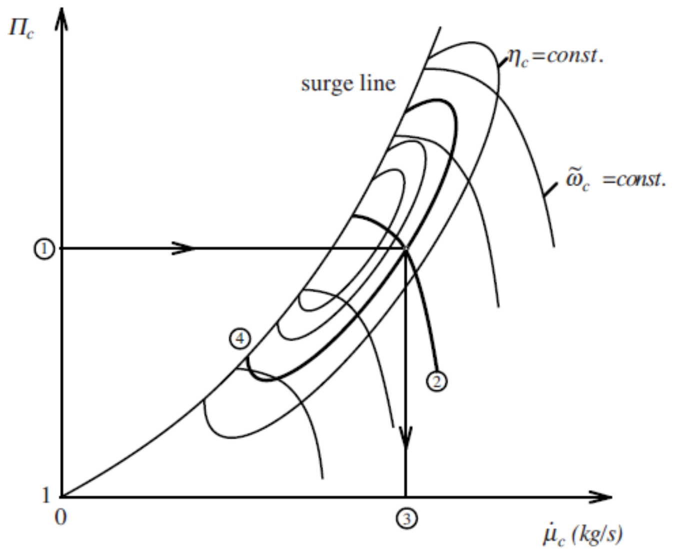


Figure 6. Compressor Map [3]

The power needed to drive to compressor can be derived by finding the compressor power ($P_{c,s}$) that a "perfect" compressor would need to compress the air at specified outlet pressure, then dividing by η_c to obtain the actual compressor power (P_c) shown in Eq. (11). (note c_p is the fluids heat capacity at constant pressure and where κ is the ratio of specific heats; both assumed Quasi-steadily constant)

$$P_c = \frac{P_{c,s}}{\eta_c} = \dot{m}_c c_p T_2 \left[\Pi_c^{\frac{\kappa-1}{\kappa}} - 1 \right] \frac{1}{\eta_c} \tag{11}$$

This compares the theoretical minimum power required to move the air through the compressor to the actual power needed to overcome unavoidable losses due to entropy generation. The torque required to rotate the compressor blades ($\mathrm{T_c}$) is:

$$\mathrm{T_c} = \frac{P_c}{\omega_c} \tag{12}$$

Temperature increase caused by the air compression (T_3) is formulated using isentropic assumption then correcting for the compressor's isentropic efficiency which reduces to:

$$T_3 = T_2 + \left[\Pi_c^{\frac{\kappa-1}{\kappa}} - 1 \right] \frac{T_2}{\eta_c} \tag{13}$$

Equations (11) through (13) are calculated assuming adiabatic conditions, where heat loss through the compressor walls is assumed to be zero. Though in reality the heat loss is non-zero and would decrease the true value of T_3, although slightly. Multiple turbochargers can be handled by simply multiplying the output $\dot{m}_c$ by the number of turbochargers attached in parallel ($N_{tc\parallel}$) to the engine, which makes:

$$\dot{m}_3 = (N_{tc\parallel})\dot{m}_c \tag{14}$$

Aftercooler

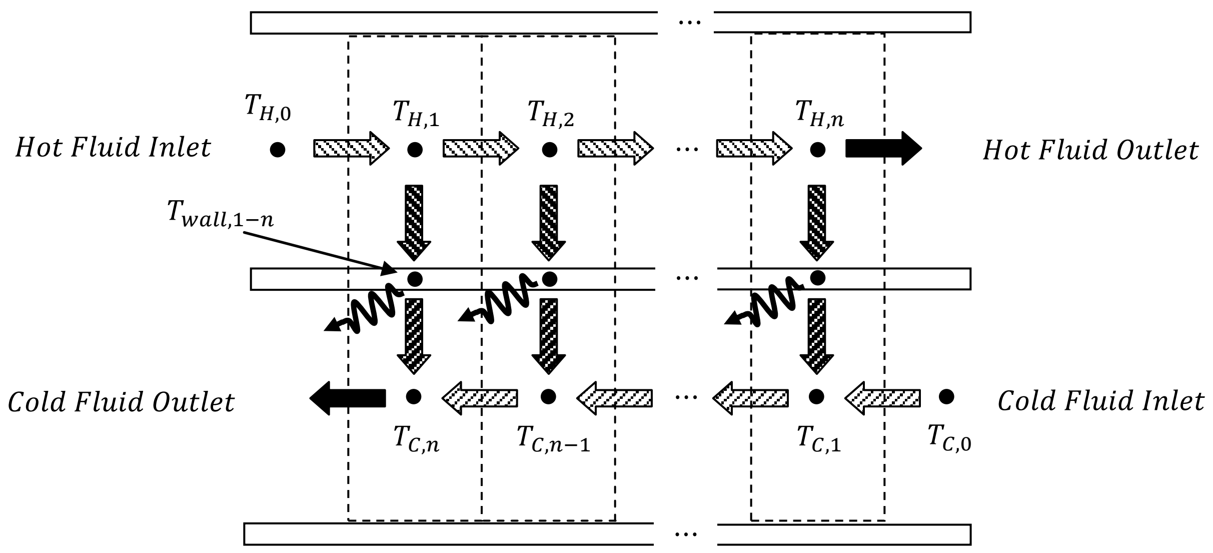


Figure 7. Schematic of General Cross-Flow Heat Exchanger

Table 1. Heat Flows and Respective Symbols

⇨	$\dot{Q}_H = \dot{m}_H c_{p,H} \Delta T_H$	(15)
⬇	$\dot{Q}_{H,w} = \bar{h} A \Delta T_{H-w}$	(16)
⇦	$\dot{Q}_C = \dot{m}_C c_{p,C} \Delta T_C$	(17)
⬇	$\dot{Q}_{C,w} = \bar{h} A \Delta T_{C-w}$	(18)
↜	$\dot{Q}_{loss} = \mathrm{hA}(\mathrm{T_{wall}} - \mathrm{T_\infty}) + \varepsilon A \sigma \left({T_{wall}}^4 - {T_{sur}}^4\right)$	(19)

The aftercooler is modeled similarly to the rest of the heat exchangers in the engine MVM. Figure 7 displays the interaction of heat flows within a cross flow heat exchanger separated by nodes using a finite difference method. The heat exchanger is split into a discrete number of nodes, n, and temperatures and heat flows are calculate at each node. Subscripts h and c denote “hot side” versus “cold side” and subscript w denotes the surface wall. T_∞ does not necessarily have to equal T_{surr} since it is the surrounding surface(s) temperature such as a wall in a room surrounding the heat exchanger, while T_∞ is the temperature of air surrounding the heat exchanger. Equations (15) through (19) are the respective $\dot{Q}$ for heat balances used to calculate T_H, T_C, and T_w given by:

$$\Delta T_H = \frac{\dot{Q}_H}{\dot{m}_H c_{p,H}} \tag{20}$$

$$\Delta T_C = \frac{\dot{Q}_C}{\dot{m}_C c_{p,C}} \tag{21}$$

$$\frac{dT_{wall}}{dt} = \frac{1}{m_{wall} c_{wall}} \left(\dot{Q}_H - \dot{Q}_C - \dot{Q}_{loss}\right) \tag{22}$$

where the hot fluid temperature difference (ΔT_H), cold fluid temperature difference ΔT_C, hot fluid temperature and wall temperature difference (ΔT_{H-w}), cold fluid temperature and wall temperature difference ΔT_{C-w} are defined to keep heat flows positive as: (i denoting the node)

$$\Delta T_H = T_{H,i} - T_{H,i+1} \tag{23}$$

$$\Delta T_C = T_{C,i+1} - T_{C,i} \tag{24}$$

$$\Delta T_{H-w} = T_{H,i} - T_{wall,i} \tag{25}$$

$$\Delta T_{C-w} = T_{wall,i} - T_{C,i} \tag{26}$$

Differences between the different heat exchangers in the engine MVM is dictated by the way the heat transfer coefficient (h) is calculated and can be reviewed in [20-22] for various types of flow. The different types of flows depend on the physical sizes and shapes of the heat exchangers. A common formulation to estimate the value of h is to calculate the Reynolds number (Re) and the Prandtl number (Pr) of the moving fluid and correlate it to the Nusselt number (Nu) defined as:

$$Re_{L_c} = \frac{\rho v L_c}{\mu} \tag{27}$$

$$\overline{Nu}_{L_c} = f\left(Re_{L_c}, Pr\right) \rightarrow a {Re_{L_c}}^{m} Pr^{n} \tag{28}$$

$$\bar{h} = \frac{\overline{Nu}_{L_c} k}{L_c} \tag{29}$$

ρ is the density of the fluid, v is the fluid's velocity, μ is the dynamic viscosity of the fluid, L_c is the characteristic length, and k is the thermal conductivity of the fluid. It must be noted that it is common to find the average Nusselt number $\overline{Nu}_{L_c}$ over the entire length L_c, where as Nu_x denotes the local Nusselt number at position x into the flow and varies along the L_c, hence Eq (28) becomes:

$$Nu_x = f\left(x, Re_{L_c}, Pr\right) \rightarrow a {Re_x}^{m} Pr^{n} \tag{30}$$

by replacing $L_c$with x in Eq. (27). The nodes, defined in Figure 7, help keep ΔT's from being so large that the 2nd law of thermodynamics is not violated. This is caused by the estimation of h's, which can be as far off as ±40% from the true value [20]. The values of h also vary with length into the heat exchanger, due to changes in flow, pressure, and temperature.

Using relationships developed in Eqs. (15)-(24) and creating a discrete number of nodes, the temperature after the aftercooler (T_4) can be calculated knowing inputs T_3, coolant temperature (T_{SCAC}) from Separate Circuit Aftercooler (SCAC) circuit, mass flow intake air ($\dot{m}_4 = \dot{m}_3$), and mass flow coolant ($\dot{m}_{SCAC}$). It is assumed that little to no pressure drop occurs between state 3 to 4 in the aftercooler, therefore $p_3 =$ intake manifold pressure (p_4). It is also assumed to be a shell and tube heat exchanger, where cooling water flow over banks of tubes containing the hot compressed charge air. The detailed version of Eqs. (27)-(29) for a shell and tube heat exchanger will be discussed at end of the chapter in the external engine model section.

Intake Manifold

The intake manifold gathers charge air from the aftercooler and distributes it to the different cylinders. In general it is more useful to use total flow out of the manifold and then divide by total number of cylinders to achieve flow per cylinder. The intake manifold can be modeled as a manifold with one inlet, one outlet, and a receiver state. Figure 8 illustrates the flow of variables between inlet and outlet of a generic manifold. Flows of energy in and out of the manifold are accounted for at the surface of the control volume denoted by the dashed line.

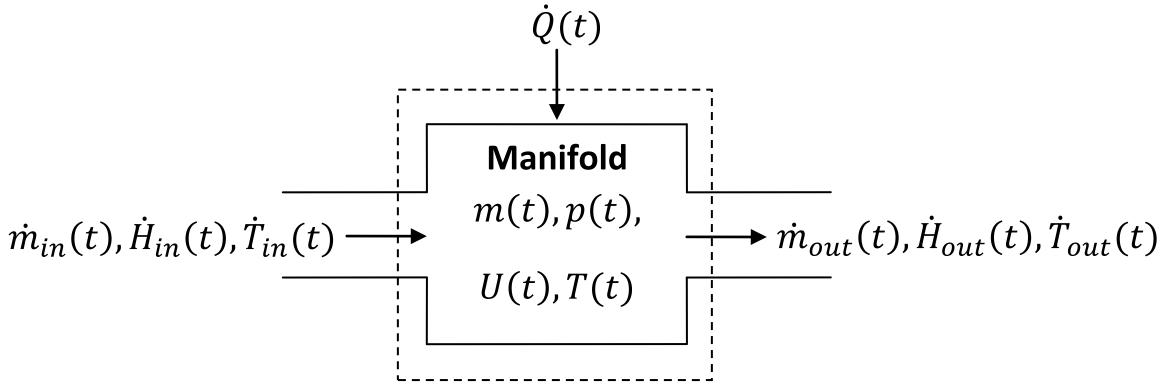

$\dot{Q}(t)$
Manifold
$m(t), p(t),$
$U(t), T(t)$
$\dot{m}_{in}(t), \dot{H}_{in}(t), \dot{T}_{in}(t)$
$\dot{m}_{out}(t), \dot{H}_{out}(t), \dot{T}_{out}(t)$

Figure 8. Manifold Control Volume

By applying the 1st law of thermodynamics to the control volume in Figure 8 (assuming no work is produced and potential energy is zero), the input and output are related by coupled differential equations [2, 3]:

$$\frac{d[m(t)]}{dt} = \dot{m}_{in}(t) - \dot{m}_{out}(t) \tag{31}$$

$$\frac{d[U(t)]}{dt} = \dot{H}_{in}(t) - \dot{H}_{out}(t) + \dot{Q}(t) \tag{32}$$

where m is the fluids mass, $\dot{m}$ is the mass flow rate, U is the fluids internal energy, $\dot{H}$ is the enthalpy flow, and $\dot{Q}$ is the heat flow. Subscripts *in* and *out* indicate inlet and outlet positions of the manifold. Then assuming fluid is an ideal gas governed by:

$$p(t)V = m(t)RT(t) \tag{33}$$

as well as caloric relations for:

$$U(t) = c_v T(t)m(t) = \frac{1}{\kappa - 1} p(t)V \tag{34}$$

$$\dot{H}_{in} = c_p T_{in}(t)\dot{m}_{in}(t) \tag{35}$$

$$\dot{H}_{out} = c_p T(t)\dot{m}_{out}(t) \tag{36}$$

It is assumed that outlet temperature $[T_{out}(t)]$ is equal to internal manifold temperature $[T(t)]$ as a part of the lumped parameter approach. Constant specific heats at constant volume (c_v) and constant pressure (c_p) is assumed constant, also R is the fluids specific gas constant. By substituting Eqs. (33)-(36) into Eq. (31) and (32) and performing algebraic manipulations; manifold pressure and temperature can be obtained in explicit form and described by:

$$\frac{d[p(t)]}{dt} = \frac{\kappa R}{V}[\dot{m}_{in}(t)T_{in}(t) - \dot{m}_{out}(t)T(t)] \tag{37}$$

$$\frac{d[T(t)]}{dt} = \frac{T(t)R}{p(t)Vc_v}\left[c_p\dot{m}_{in}(t)T_{in}(t) - c_p\dot{m}_{out}(t)T(t) - c_v(\dot{m}_{in} - \dot{m}_{out})T(t)\right] \tag{38}$$

The explicit form of Eq. (37) and (38) allows for easy integration to find pressure and temperature within the manifold. The dependence on the balance of $\dot{m}_{in}$ and $\dot{m}_{out}$ is the reason this method is commonly called the “filling” and “emptying” method. For the case in Figure 3 where mass flow into the intake manifold is $\dot{m}_4$ and denoting engine air mass flow from state 5 to 6 as $\dot{m}_\beta$, $\dot{m}_{in}$ and $\dot{m}_{out}$ become $\dot{m}_4$ and $\dot{m}_\beta$ respectively. Also, T_{in} and $T_{out} = T$ become T_4

and T_5, and p equals p_3, p_4, and the exit of the intake manifold and inlet pressure to the engine (p_5) (assuming small pressure drop across aftercooler/intake manifold).

Gas Exchange and Fuel Flow

Engine air mass flow through the engine ($\dot{m}_\beta$) can be calculated by assuming the engine is a volumetric pump where flow is proportional to speed [3]. It is based on the engine's volumetric efficiency (η_V) and $\dot{m}_\beta$ is commonly termed the "speed-density" [2] referring that it depends on engine speed and charge air density. Common formulation of $\dot{m}_\beta$ is posed as:

$$\dot{m}_\beta = \rho_4(t)\dot{V}(t) = \frac{\rho_4(t)\eta_V(p_4,\omega_e)V_d\omega_e(t)}{2\pi N} \tag{39}$$

where $\omega_e$is the crankshaft/flywheel's rotational speed. η_V describes how far the engine differs from a perfect volumetric device (ideal pump) and can be calculated through:

$$\eta_V(p_4,\omega_e) = \eta_V(p_4)\eta_V(\omega_e) \tag{40}$$

and η_V with respect to p_4 is:

$$\eta_V(p_4) = \frac{V_{TDC}+V_d}{V_d} - \left(\frac{p_6}{p_5}\right)^{\frac{1}{\kappa}}\frac{V_{TDC}}{V_d} \tag{41}$$

where V_{TDC} is the cylinder volume at Top Dead Center (TDC), V_d is the cylinder's displaced volume, and p_6 is the pressure at the exhaust of the engine. $\eta_V(\omega_e)$ is based on an engine map found through collection of experimental data.

Engine fuel flow $\dot{m}_{fuel}$ is mainly a function of load and is discretely controlled by the engine control unit (ECU) to manage torque and emissions [3, 23]. This is achieved by controlling the time the solenoid valve is energized which adjust the amount of fuel delivered to each cylinder. The camshaft connected to the crank is continually rotating and fuel is continually flowing through the fuel injectors. Fuel injection during the camshaft cycle is solely determined by time the solenoid valve is energized where the signal is sent from the ECU [15, 24]. In summary, the cam shaft has to be in its lift position and the electronic unit injector nozzle has to receive an electric signal from the ECU for fuel to be injected into each respective cylinders. In the case of the engine MVM, this process can be simplified by realizing $\dot{m}_{fuel}$ directly proportional to P_{fuel} in Figure 4. This non-dimensional value is converted into a volumetric flow based on value of 1 is equal to flow at 100% load at steady state. Manufactures commonly give steady state fuel consumption flow rate ($\dot{V}_{fuel}$) versus load and assuming a nominal fuel density ρ_{fuel}, $\dot{m}_{fuel}$ can be deduced by Eq. (42) .

$$\dot{m}_{fuel} = \dot{V}_{fuel}\rho_{fuel} \tag{42}$$

In solving for Eq. (39) and (42), Eq. (3) can be realized and then used in feedback for the power generation transfer function in Figure 4.

Engine Block

For modeling external engine components (such as jacket water coolant and lubrication system), detailed thermal models of the heat transfer from the combustion process and heat transfer to cooling fluids must be developed. Figure 9 shows the interaction of heat flows within a combustion engine. The balances of heat flows (similar to Eq. 22 and discussed in detail later) lead to the formation of average lumped temperatures for the cylinder wall, engine coolant, engine block, and engine oil temperature, in which the technique is presented in the coming paragraphs. It is assumed everything that is modeled in one cylinder is the same in the rest of the engine cylinders, which leads to the single cylinder model in the MVM.

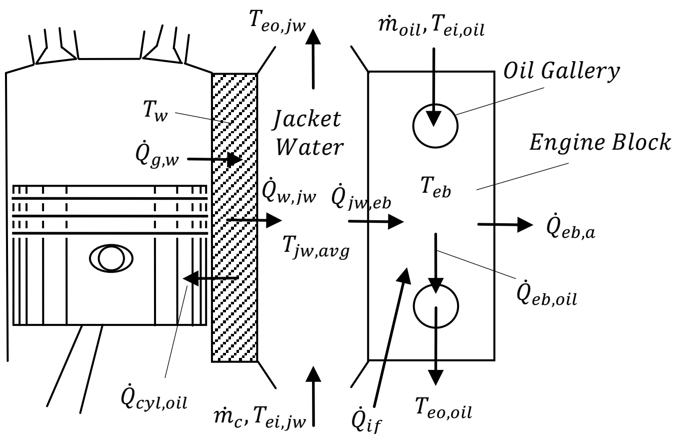


Figure 9. Engine Heat Flows

Engine temperatures are calculated based on heat flow balances using a similar process (review [3, 25, 26]) in finding T_{wall} as in the heat exchanger model presented for the aftercooler. Termed "enthalpy balances" [3], the time derivative of the engine temperatures: cylinder wall temperature (T_w), jacket water outlet temperature ($T_{eo,jw}$), lumped engine block temperature (T_{eb}), and engine oil outlet temperature ($T_{eo,oil}$) become:

$$\frac{dT_w}{dt} = \frac{1}{m_w c_w}\left[\dot{Q}_{g,w} - \dot{Q}_{w,jw} - \dot{Q}_{cyl,oil}\right] \tag{43}$$

$$\frac{dT_{eo,jw}}{dt} = \frac{1}{m_{jw} c_{p,jw}}\left[\dot{Q}_{w,jw} - \dot{Q}_{jw,fl} - \dot{Q}_{jw,eb}\right] \tag{44}$$

$$\frac{dT_{eb}}{dt} = \frac{1}{m_{eb} c_{eb}}\left[\dot{Q}_{jw,eb} - \dot{Q}_{z2} - \dot{Q}_{eb,oil}\right] \tag{45}$$

$$\frac{dT_{eb,oil}}{dt} = \frac{1}{m_{oil} c_{oil}}\left[\dot{Q}_{eb,oil} - \dot{Q}_{oil,fl} - \dot{Q}_{cyl,oil}\right] \tag{46}$$

$$\dot{Q}_{z2} = \dot{Q}_{if} - \dot{Q}_{eb,a} \tag{47}$$

The heat flows associated with convective heat transfer [heat flow from cylinder to jacket water ($\dot{Q}_{w,jw}$), heat flow from cylinder walls/piston to lubrication oil ($\dot{Q}_{oil,cyl}$), heat flow from jacket water to engine block ($\dot{Q}_{jw,eb}$), and heat flow from engine block to gallery oil ($\dot{Q}_{eb,oil}$) were calculated using general equation, known as Newton's law of Cooling [20]:

$$\dot{Q} = hA\Delta T \tag{48}$$

h's were calculated using Nu, Re, and Pr relations similar to Eqs. (27)-(29). In practical applications, h and Nu (which vary along the characteristic length) become the averaged values along the flow $\overline{h}$ and $\overline{Nu}$. $\dot{Q}_{eb,a}$ was calculated by combining natural convection and external radiation to the environment in the same fashion as Eq. (19). The natural convection $\overline{h}$ is a function of the $\overline{Nu}$ based on Grasshof number (Gr) and Pr and approximated by a vertical flat plate of length L:

$$Gr_L = \frac{g\beta(T_s - T_\infty)L^3}{\nu^2} \tag{49}$$

where g is gravitational acceleration, β is the fluid's volumetric thermal expansion coefficient, and ν is the kinematic viscosity of the fluid. The gas surrounding the engine is air and can be modeled as an ideal gas, therefore:

$$\beta = -\frac{1}{\rho}\left(\frac{\partial\rho}{\partial T}\right)_p = \frac{1}{\rho}\frac{p}{RT^2} = \frac{1}{T} \tag{50}$$

The Raleigh number (Ra) based on length L is conveniently defined by combining Gr_L and Pr in Eq. (51) and used in Eqs. (52) and (53).

$$Ra_L = Gr_L Pr = \frac{g\beta(T_s - T_\infty)L^3}{\nu\alpha} \tag{51}$$

$$\overline{Nu}_L = a{Ra_L}^n \tag{52}$$

$$\overline{h} = \frac{\overline{Nu}_L k}{L} \tag{53}$$

where α is the thermal diffusivity of the fluid. For a vertical flat plat: a = 0.59 and n = 1/4 for laminar flow ($10^4 \lesssim Ra_L \lesssim 10^9$), while a = 0.10 and n = 1/3 for turbulent flow ($10^4 \lesssim Ra_L \lesssim 10^9$) [20]. The radiation portion of $\dot{Q}_{eb,a}$ is the simplified form of radiation heat transfer where the engine block radiates to is surroundings in an enclosed room; the walls modeled are as isothermal and "completely" surround the engine.

Fluid flows out of engine ($\dot{Q}_{jw,fl}$ and $\dot{Q}_{oil,fl}$), for engine jacket water and oil respectively, are representative of transferring heat from wall and boundary layer out of the engine. These heat flows are governed mainly by the temperature difference between average fluid temperature and the wall, since $\dot{m}$ of the fluids (oil and jacket water) reaches steady state before the ΔT's do the same. This is qualitatively found by placing control volume around the engine coolant/oil entering the engine and solving for the enthalpy difference between fluid's enthalpy entering (h_{in}) and enthalpy exiting (h_{out}) (assuming the fluid does no work, incompressible, and negligible potential and kinetic energy differences).

$$\dot{Q} = \dot{m}(h_{in} - h_{out}) \tag{54}$$

$$dh = c_p dT \tag{55}$$

$$\dot{Q}_{f,flw} = \dot{m}_{f,flw} c_{p,f} \left(T_{f,in} - T_{f,out}\right) \tag{56}$$

(note subscript f denotes either jacket water or oil and flw is describing that the fluid is flowing in and out of the control volume. Also dh is the enthalpy derivative).

Heat flow generated by the engines internal friction, denoted $\dot{Q}_{if}$, is assumed that all the heat is transferred into the engine block only. Relating the amount of heat transferred to the engine's mean effective pressure loss due to friction (p_{meof}) the heat flow becomes:

$$\dot{Q}_{if} = p_{meof}(t) V_d \frac{\omega_e(t)}{4\pi} \tag{57}$$

The formulation p_{meof}proposed by [3] is:

$$p_{meof}(\omega_e, T_{eb}, \dots) = x_1(T_{eb})(x_2 + x_3 S^2 \omega_e)\Pi_{e,max}\sqrt{\frac{x_4}{B}} \tag{58}$$

where S is the length of the pistion's stroke in the cylinder. Table 2 and Figure 10 show the accompanying values of x_1 to x_4 and how ratio of $x_1(T_{eb})$ and $x_1(T_\infty)$ varies as T_{eb} increases. $\Pi_{e,max}$ is defined as the maximum boost ratio the engine is designed operate at low speeds.

Table 2. Parameters of the ETH Friction Model

	Typical Diesel
x_1	$1.44 \cdot 10^5\ (Pa)$
x_2	$0.50\ (-)$
x_3	$1.3 \cdot 10^{-3}\ (s^2/m^2)$
x_4	0.075

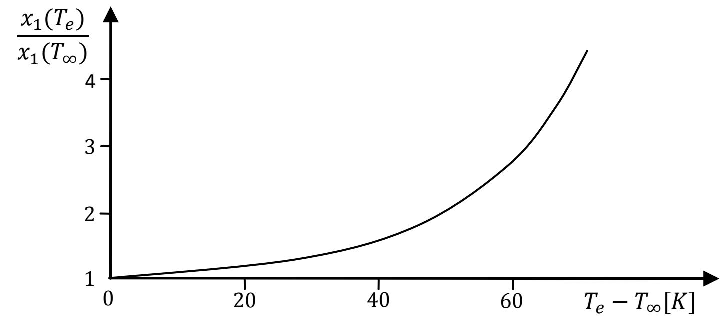


Figure 10. Temperature Dependency on Mechanical Friction

The heat transfer from inside the cylinder to the cylinder wall is referred to as $\dot{Q}_{g,w}$. Various researchers have studied this phenomenon [2, 3, 15, 24-28] and the author's approaches vary widely. However, the technique presented in Heywood proves to be the most useful.

$$Re = \frac{\dot{m}_\beta B}{\mu_g(\pi B^2/4)} = \frac{4\dot{m}_\beta}{\pi \mu_g B} \tag{59}$$

$$\overline{Nu} = aRe^m \tag{60}$$

$$\dot{Q}_{g,w} = \frac{\pi B k_g \left(T_{g,a} - T_w\right)\overline{Nu}}{4} \tag{61}$$

where B is the cylinder's bore and μ_g is the dynamic viscosity of the combustion gas, and k_g is the thermal conductivity of the combustion gas. $T_{g,a}$ is the temperature at which the cylinder wall would stabilize if not heat was loss to the outside, which is found by extrapolating the heat transfer data versus the gas side cylinder temperature and tracing the line back to the zero heat transfer axis[15]. $T_{g,a}$, μ_g, and $k_g$can be experimentally tabulated versus equivalence ratio for internal combustion (IC) with hydrocarbons mixing with air. Refer to [15, 27, 28] for determining coefficient a and power exponent m. For the application of large bore diesel engines, values of 15 for a and 0.75 for m were choosen.

Due to inefficiencies in IC engine combustion, exhaust gases carry large amount of fuels energy, which is realized in cylinder exhaust Temperature. The factors affecting exhaust temperature are air/fuel ratio, valve timing, $\dot{Q}_{g,w}$, among others; which makes it difficult to accurately predict the exhaust gas temperature. A combination of measurements and correcting factors are commonly used to estimate exhaust gas temperature. Another approach would be to calculate complete thermal and chemical equilibrium balance but would be computationally too demanding to solve in real-time. A simplified approach to calculate the exhaust gas temperature (T_{eg}) is carried through by using the air/fuel mixture heating value H_m and dividing by the mixture's c_p to obtain a theoretical ΔT_{eg}, then modifying it by the engine's efficiency $[e_\omega(\omega_e)]$

and fraction of heat that goes to the cylinder wall versus exhaust, x_e. This is best described in Eq. (62)

$$\Delta T_{eg} \approx x_e \frac{H_m}{c_p} e_\omega(\omega_e) \tag{62}$$

where the heating value of the air and fuel mixture is related by the fuel's AF_{stoich} and ϕ through:

$$H_m = \frac{H_l}{\frac{AF_{stoich}}{\phi} + 1} \tag{63}$$

where $e_\omega(\omega_e)$ is found from engine specific maps and is parabolic in form peaking around at its rated speed. This can be explained by the large heat loses through the wall occur at low speeds and while at high speed the combustion reaction time becomes large compared to the expansion stroke's interval [3].

Exhaust Manifold

One important assumption must be made to avoid physically calculating the exhaust gas thermal properties (as a function of exhaust gas species, temperature, and pressure) is that the exhaust gas thermal properties can be reasonably estimated by using thermal properties of air at atmospheric pressure. This greatly simplifies finding the $c_{p,eg}$ and κ_{eg} and is a reasonable assumption since air is primarily composed of 78% Nitrogen (N_2) and most of the exhaust gas is still mainly N_2 (around 70%) because very little N_2 in the charge air actually reacts with the fuel or Oxygen (O_2). The concentrations of water (H_2O) and Carbon Dioxide (CO_2) are small compared to $N_2$and their effect on the overall mixture thermal properties is minimal.

The exhaust manifold is modeled identically the same as the intake manifold using the "emptying and "filling" method. The only difference is accounting for different manifold volume. Equations governing the exhaust manifold are Eqs. (37) and (38), then in referencing

Figure 3; subscripts in and out on pressure, temperature, and mass flow become state 6 and 7, respectively.

Turbocharger-Turbine

The turbine side of the turbocharger is modeled in a similar process as the compressor. Steady state data is supplied by the manufacture in a map. This data (Eqs. (64) and (65)) is then digitized and points are found through linear interpolation methods.

$$\tilde{\mu}_t = f(\widetilde{\omega}_t, \Pi_t) \tag{64}$$

$$\eta_t = f(\widetilde{\omega}_t, \Pi_t) \tag{65}$$

where $\tilde{\mu}_t$ is the corrected turbine mass flow rate, η_c is the turbine efficiency, $\widetilde{\omega}_t$ is the corrected turbine rotational speed, and Π_t is the turbine pressure ratio. To keep nomenclature consistent with Figure 3 and how manufactures define corrected flow and speed, related equations result in (note subscript 7 referring to state after exhaust manifold and turbine inlet, while subscript 8 referring to the exit of the turbine):

$$\tilde{\mu}_t = \dot{m}_t \frac{p_{7,std}}{p_7} \sqrt{\frac{T_7}{T_{7,std}}} \tag{66}$$

$$\widetilde{\omega}_t = \omega_t \sqrt{\frac{T_{7,std}}{T_7}} \tag{67}$$

$$\Pi_t = \frac{p_7}{p_8} \tag{68}$$

The power extracted from the turbine due the change in the exhaust gas enthalpy is (assuming ideal gas and no entropy generation, and then applying an isentropic efficiency):

$$P_t = P_{t,s}\eta_t = \dot{m}_t c_p T_7 \left[1 - \Pi_t^{\frac{\kappa-1}{\kappa}}\right] \eta_t \tag{69}$$

where $P_{t,s}$ is the theoretical maximum power extracted from the turbine while P_t being the actual. Torque ($\mathrm{T_t}$), generated from hot exhaust gas flowing into the turbine blades, is calculated by:

$$\mathrm{T_t} = \frac{P_t}{\omega_t} \tag{70}$$

In most cases maximum turbine efficiency ($\eta_{t,max}$) will be given by manufactures and $\tilde{\mu}_t$ will only depend on Π_t. Since η_t depends on the incidence angle exhaust gas strikes the turbine blades, the "turbine blade speed" ratio (c_{us}) is main parameter affecting η_t for single stage impulsive turbines [3].

$$\tilde{c}_{us} = \frac{r_t \omega_t}{c_{us}} \tag{71}$$

$$c_{us} = \sqrt{2 c_p T_7 \left[1 - \Pi_t^{\frac{\kappa-1}{\kappa}}\right]} \tag{72}$$

$$\eta_t(\tilde{c}_{us}) = \eta_{t,max}\left[\frac{2\tilde{c}_{us}}{\tilde{c}_{u,opt}} - \left(\frac{\tilde{c}_{us}}{\tilde{c}_{u,opt}}\right)^2\right] \tag{73}$$

where $\tilde{c}_{us}$ is the corrected turbine blade speed ratio and $\tilde{c}_{u,opt}$ is the optimal corrected blade speed ratio. Equations (71) through (73) are best visualized by Figure 11. Typical values for $\eta_{t,max} \approx 0.65\ldots0.75$ (though specified in manufacture's turbine map) and $c_{u,opt}$ 0.55…065 [3].

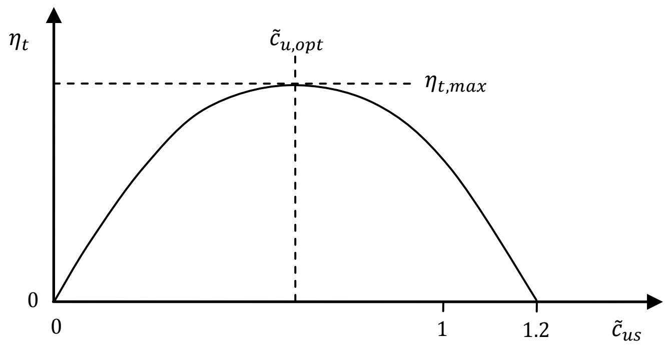


Figure 11. Simplified Turbine Map for Turbine Efficiencies and Blade Speed: Eq. (73)

Turbocharger Rotor Torque Balance

In accounting for compressor, turbine, and turbocharger entropy generating losses, the torque balance based on Newton's second law is:

$$\frac{d\omega_{tc}}{dt} = \frac{1}{\Theta_{tc}}[T_t - T_c - T_{loss}] \tag{74}$$

The time derivative of speed (acceleration) can then be numerically integrated to obtain turbocharger shaft speed (ω_{tc}). If both connected through a single shaft: $\omega_{tc} = \omega_t = \omega_c$. Due to the fact that the compressor and turbine both rely on the coupled shaft speed, Eq. (74) is the crux of the turbocharger model. Obtaining a realistic Θ_{tc} and T_{loss} is essential to obtain a stable turbocharger model. Unfortunately these values are not readily available from turbocharger/engine manufactures which leads to a trial and error method of finding the values Θ_{tc} and T_{loss} that produce the correct ω_{tc} at certain engine loads. T_{loss} was found to be not constant and non-linear with ω_{tc}. This could be caused by the error in estimation of the compressor and turbine efficiencies, losses due to drag within the bearing lubrication and windage (air resistance on rotating blades) between the rotor and stator in both compressor and turbine.

3.4 External Models from Engine

Flow Networks

The external systems connected to the engine (besides air intake/combustion exhaust) can be considered as "closed" systems, which require pumps to provide pressure difference to deliver each respective fluid to the engine to perform its intended purpose. These closed systems are modeled as flow networks. The flow networks for fuel, lubrication oil, and jacket water cooling can be represented as a pump displacing fluid, at a rated volumetric flow rate, where it flows through a series of resistances that lowers the fluid pressure. A simple pump can be modeled by:

$$\dot{V} = f(\Delta p_p) \tag{75}$$

where $f(\Delta p_{pump})$ can be derived from manufacturer's compressor map and generally a second order curve fit is used resulting in:

$$f(\Delta p_p) = -\xi_2 \Delta \mathrm{p}_p{}^2 - \xi_1 \Delta \mathrm{p}_p + \xi_0 \tag{76}$$

then using the fundamental relation described in Eq. (4), C_v's are created for pressure drops within the systems, at rated flow rate, such as the resistances associated with: heat exchangers, filters, pressure regulators, or any other resistances of importance. A form of the emptying and filling method, the flows of the pump and the flows through the system are added then multiplied by a gain K_{system} to simulate capacitance and integrated to find Δp_p, shown in Eq. (77).

$$\frac{d\Delta p_p}{dt} = K_{system}\left(\sum \dot{V}_{in} - \sum \dot{V}_{out}\right) \tag{77}$$

C_v's can be combined in series and parallel to achieve a $C_{v,total}$ for the whole flow network, total network flow resistance, in which the pump has to overcome. For flow resistances in series use Eq. (78) and for resistances in parallel use Eq. (79). The differences in flow paths (series or parallel) can be seen in Figure 12.

$$\left(\frac{1}{C_{v,total}}\right)^2 = \left(\frac{1}{C_{v,1}}\right)^2 + \left(\frac{1}{C_{v,2}}\right)^2 + \cdots + \left(\frac{1}{C_{v,n}}\right)^2 \tag{78}$$

$$C_{v,total} = C_{v,1} + C_{v,2} + \cdots + C_{v,n} \tag{79}$$

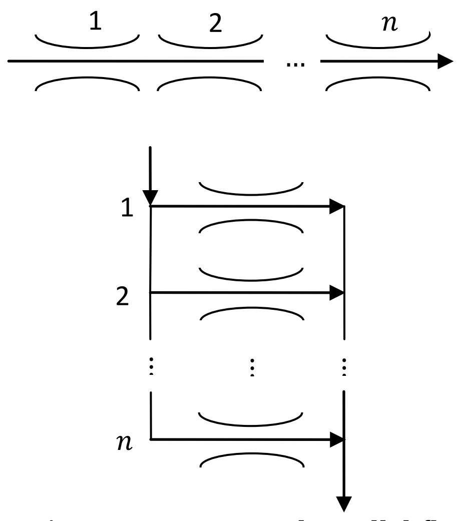


Figure 12. Series and Parallel flow

By combining Eq. (76) through Eq. (79) a simplified flow network can be developed. C_v's can also vary with time and even could simulate a filter clogging over a long period of time if necessary or simulate pressure or temperature regulating valve metering flow based on fluid pressure or temperature.

The main external flow circuits modeled for the diesel engines were the seawater circuit, jacket water circuit, lubrication oil circuit, SCAC circuit, and common rail fuel circuit. The seawater circuit is opened looped, as in the seawater is sucked into the ship, heated by the various coolers, and dumped back into the sea. While the next three circuits are closed looped, continually cycling fluid within the circuit. The last circuit is a combination of an open system (fuel that is sent to the combustion chamber by the fuel injectors and consumed by the engine to create exhaust gas) and a closed system (the fuel that is bypassed by the fuel injectors and sent back to the fuel tank).

Seawater is the major external cooling medium on board the naval ship and can be the determining factor in the effectiveness of the heat exchange, since the inlet seawater temperature varies on location and time of year. Though for modeling purposes, the sea water temperature is assumed constant at a nominal value of 303.15 K (or 77 °F). For the diesel engines, it is the external cooling medium for the jacket water (indirectly lubrication oil) and SCAC. It also feeds the cooling for the electrical generator’s stators and lubrication oil, discussed in later sections. A heat transfer model was developed for the jacket water cooler and SCAC using limited data and specifications. In previous projects at Woodward, a similar external thermal model was developed for cooling jacket water by seawater and jacket water cooling lubrication oil, where the schematic was similar to Figure 3 minus the SCAC circuit. A more detailed model was developed using the counter flow wall technique presented in Figure 7, Table 1, and Eqs. (15)-(29) . This technique will work for both shell and tube and plate heat exchangers, which will be discussed in detail in a later section. Figure 13 shows a schematic for a seawater cooled engine with SCAC taken from a manufactures installation and application guide for engine cooling.

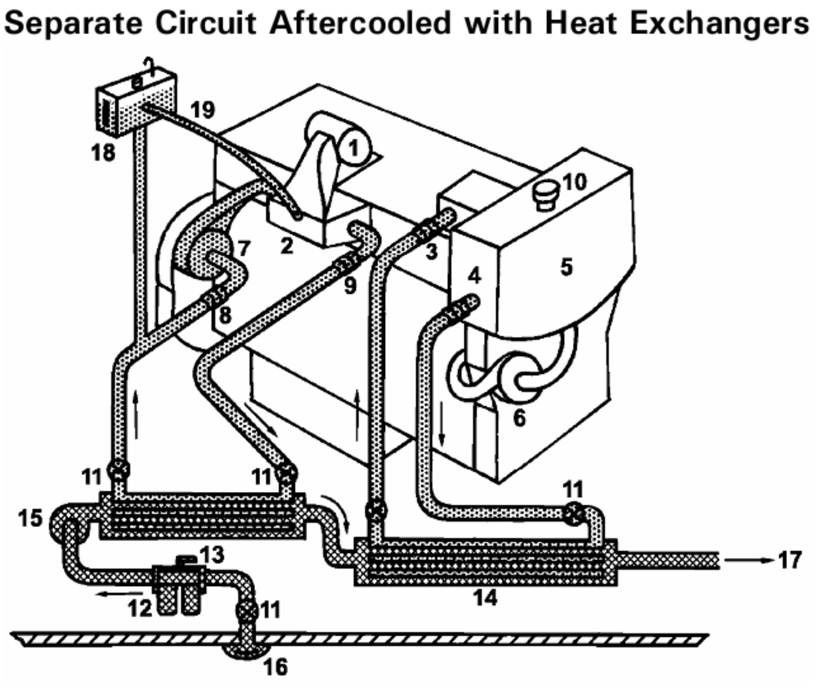


Figure 33

1. Turbocharger
2. Aftercooler, Heat Exchanger Cooler
3. Jacket Water Outlet Connection
4. Jacket Water Inlet Connection
5. Expansion Tank
6. Jacket Water Pump
7. Auxiliary Fresh Water Pump
8. Auxiliary Fresh Water Inlet Connection
9. Aftercooler Outlet Connection
10. Pressure Cap
11. Shut-Off Valve
12. Duplex Full-Flow Strainer
13. Heat Exchanger for Aftercooler
14. Heat Exchanger for Jacket Water
15. Customer–Provided Seawater Pump
16. Seawater Intake
17. Seawater Discharge
18. Expansion Tank for Aftercooler Circuit
19. Vent Line for Aftercooler Circuit

Figure 13. Typical SCAC Circuit for Seawater Cooling [29]

Since not much data was available on flow rates, pump pressures, and pressure drops; the values from that model were adapted to the current model to give an educated guess on the flow network values since both engines (in this project and previous work) were similar in size and power output. The physical sizes (pipe diameters/areas and flow lengths) of the heat exchangers were estimated from heat transfer areas taken from the previous project’s jacket water to seawater cooler and lubrication oil to jacket water cooler. If more detail information on the engines cooling circuit became available such as heat flow, fluid flow, and actual flow area dimensions; Eqs. (20)-(22) and Eqs. (27)-(29) [Also discussed later Eqs.(82)-(94)] could be tuned to mimic actual heat exchanger performance. For now, the previous project's information/data is sufficient in the absence of actual.

Pressure/Flow, and Temperature Bypass

The fuel injectors and pressure regulation valve discretely change the resistance of the flow network when fuel is injected and bypassed to the tank when not needed. The pressure regulating valve helps keep a constant pressure in the unit fuel injectors. By taking the time averaged fuel consumption (which flows out of the flow network) and modeling the pressure regulator dynamics (practically on/off behavior), the bypass can be modeled by the changing C_v of the pressure regulating valve and bypass flow through it. Bypass flow in heat exchangers can be modeled in a similar way, where temperature regulators direct fluid flow to heat exchanger or bypass based on fluid temperature. The temperature regulator can be simplified into a relationship of temperature vs. amount of bypass (u_{byp}), then the amount of bypass is used to vary the C_v of flow going to heat exchanger and to the bypass, which are in parallel with each other. The different flows are described as followed:

$$\dot{m}_{hx} = \rho\dot{V}_{hx} = \rho\left[C_{v,hx}\left(1 - u_{byp}\right)\sqrt{\Delta p_{hx}}\right] \tag{80}$$

$$\dot{m}_{byp} = \rho\dot{V}_{byp} = \rho\left[C_{v,byp}u_{byp}\sqrt{\Delta p_{byp}}\right] \tag{81}$$

where Δp_{hx} is equal to Δp_{byp} at steady flow conditions.

Heat Exchangers: Shell-Tube and Plate

The two types of heat exchangers used in this model are shell and tube type and plate type, though the specific heat exchangers on the naval ship are not discussed here explicitly. The splitting heat exchangers into separate sections/nodes can be applied to both types. This technique helps to avoid over-calculating the heat transferred from the hot fluid to the cooling fluid and violating the second law of thermodynamics.

Shell and tube heat exchangers involve a bundle of tubes surrounded by an outer shell with baffles to direct flow over the bundle of tubes. Figure 14 shows a general shell-tube heat exchanger arrangement along with how it was split-up into separate sections for heat transfer calculations.

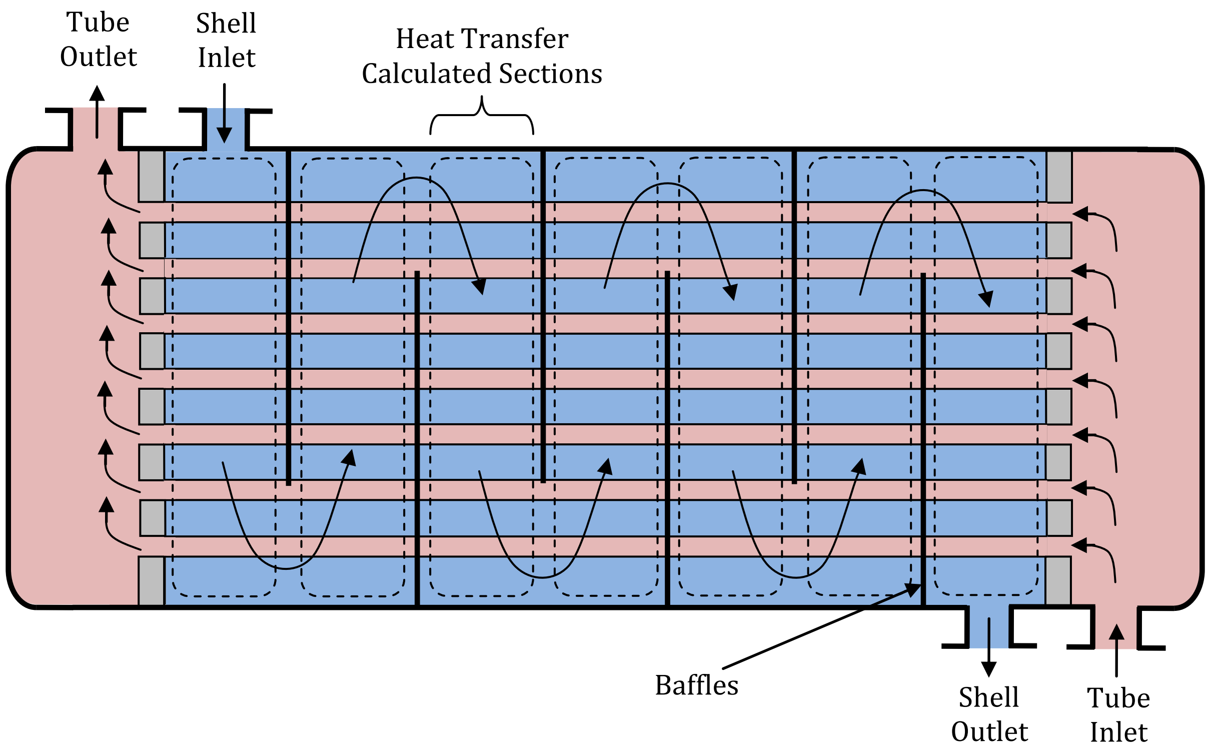


Figure 14. General Single Pass Shell and Tube Heat Exchanger

The sections can be thought of shell flow over bundles of tubes and within those tubes can be thought of pure internal tube flow. The technique applied in Figure 7, Table 1, and Eqs. (15)-(29) was used to calculate the heat flow from the hot fluid in the tube, through the wall, and to coolant fluid in the surrounding shell. Using correlations based on Re, flow, and other parameters; three correlations were used for the tube side based on laminar, transition, and turbulent regimes. For laminar conditions ($Re < 2300$) assuming constant heat flux ($\dot{Q}''$), the correlation inside the tube is [20-22]:

$$Nu_D = \frac{hD}{k} = 4.364 \tag{82}$$

where D is the diameter of the tube which in this case is the characteristic length L_c. For the transition regime where the part of the flow is starting to separate and become turbulent ($2{,}300 \leq Re < 10{,}000$): this relation was used [22]:

$$\overline{Nu}_D = \frac{\overline{h}D}{k} = 0.166(Re_D^{\frac{2}{3}} - 125)Pr^{\frac{1}{3}}\left(\frac{\mu}{\mu_w}\right)^{0.14}\left(1 + \frac{D}{L_f}\right)^{\frac{2}{3}} \tag{83}$$

where μ is the dynamic viscosity of the bulk fluid, μ_w is the dynamic viscosity of the fluid at the surface of the tube wall, L_f is the length of tube into the flow starting at the entrance of the tube [22]. For the turbulent flow conditions ($Re \geq 10{,}000$), correlation developed by Gnielinski was used [20-22]:

$$\overline{Nu}_D = \frac{\overline{h}D}{k} = \frac{(f/8)(Re_D - 1000)Pr}{1 + 12.7(f/8)^{\frac{1}{2}}(Pr^{\frac{2}{3}} - 1)} \tag{84}$$

Note in Eqs. (83) and (84), the fluids properties need to be evaluated at the section's mean temperature. The friction factor (f) can be found in many ways depending on the information on hand. If information is known about the internal flow surface's relative roughness (e/D : e is the surface roughness), the Moody Diagram may be used which is a function of Re_D and e/D [20-22]. But in most cases the surface roughness can only be guessed at, so an approximation is needed for f. Therefore, efforts have been made to develop equations that approximate f over a broad Re_D range. One implicit equation developed by Prandtl for Re_D values up to 10^6 [21]:

$$\frac{1}{f^{\frac{1}{2}}} = 1.737\ln(Re_D f^{\frac{1}{2}}) - 0.396 \tag{85}$$

While another approach developed by Petukhov yields an explicit function of f for Re_D values ranging between 3000 to 5 x 10^6 [20]:

$$f = (0.79\ln Re_D - 1.64)^{-2} \tag{86}$$

Though both work over a wide range of Re_D, Eq. (86) is easier to solve and implement into usable code.

For the heat transfer of the shell side, the fluid is modeled in each section as fluid flowing over banks of tubes similar to Figure 15. The bank of tubes can be arranged in aligned or staggered positions described in Figure 16.

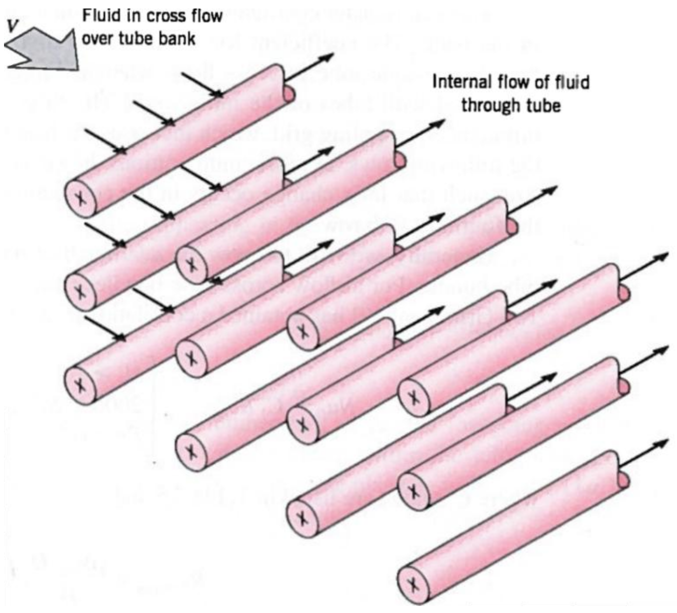

V
Fluid in cross flow over tube bank
Internal flow of fluid through tube

Figure 15. Schematic of Cross Flow Over Banks of Tubes [20]

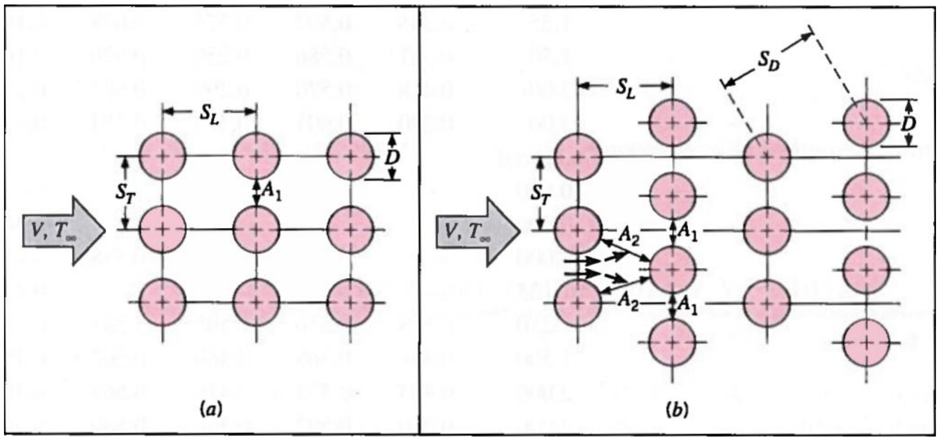


Figure 16. Tube bank Arrangements (*a*) Aligned (*b*) Staggered [20]

The configuration in Figure 16 is defined by the pipe diameter D, the transverse pitch S_T, the longitudinal pitch S_L, and the diagonal pitch S_D (each pitch measured from the center of the pipes). A technique developed by Zukauskas presented in various literature [20-22] proved the best approach to model heat transfer from fluid flowing over banks of tubes and can be applied over large ranges of Reynolds numbers (based on maximum flow velocity occurring in the tube bank (v_{max}) and outer pipe D termed $Re_{D,max}$). Zukauskas derivation is as follows; $Re_{D,max}$ is defined as:

$$Re_{D,max} = \frac{\rho v_{max} D}{\mu} \tag{87}$$

where v_{max} is calculated using parameters shown in Figure 16 for aligned arrangement:

$$v_{max} = \frac{S_T}{S_T - D} v \tag{88}$$

or for staggered arrangement if:

$$2(S_D - D) < (S_T - D) \tag{89}$$

then use:

$$v_{max} = \frac{S_T}{2(S_D - D)} v \tag{90}$$

else use Eq. (88). Once $Re_{D,max}$ is known, the $\overline{Nu}_D$ has been found to relate to it by:

$$\overline{Nu}_D = \frac{\bar{h}D}{k} = CRe_{D,max}{}^{m}Pr^{0.36}\left(\frac{Pr}{Pr_s}\right)^{\frac{1}{4}} \tag{91}$$

subject to the constraints of:

$$\begin{gathered} N_L \geq 20 \\ 0.7 \lesssim Pr \lesssim 500 \\ 1000 \lesssim Re_{D,max} \lesssim 2\ \times 10^6 \end{gathered} \tag{92}$$

where the constants C is defined in Table 3.

Table 3. Constants for Eq. (91) [20-22]

Configuration	$Re_{D,max}$	C	m
Aligned	$10 - 10^2$	0.80	0.40
Staggered	$10 - 10^2$	0.90	0.40
Aligned	$10^2 - 10^3$	0.51	0.50
Staggered	$10^2 - 10^3$	0.51	0.50
Aligned ($S_T/S_L > 0.7$)	$10^3 - 2 \times 10^5$	0.27	0.63
Staggered ($S_T/S_L < 2$)	$10^3 - 2 \times 10^5$	$0.35(S_T/S_L)^{\frac{1}{4}}$	0.60
Staggered ($S_T/S_L > 2$)	$10^3 - 2 \times 10^5$	0.40	0.60
Aligned	$2 \times 10^5 - 2 \times 10^6$	0.021	0.84
Staggered	$2 \times 10^5 - 2 \times 10^6$	0.022	0.84

For bundles of tubes with number of rows (N_L) less than 20, correction factors are used to modify Eq. (92) by factor C_2 shown in Eq. (93) and given in table. Another requirement is that the fluids properties ($\mu, \rho, k, and\ Pr$) must be evaluated at the fluid's mean temperature from the inlet and outlet temperatures in each section.

$$\overline{Nu}_D\big|_{(N_L<20)} = C_2\overline{Nu}_D\big|_{(N_L\geq 20)} \tag{93}$$

Table 4. Correction Factor C_2 of Eq. (91) [20-22]

N_L	1	2	3	4	5	7	10	13	16
Aligned ($Re_{D,max} > 1000$)	0.70	0.80	0.86	0.90	0.92	0.95	0.97	0.98	0.99
Staggered ($Re_{D,max} 100 - 1000$)	0.83	0.87	0.91	0.94	0.95	0.97	0.98	0.98	0.99
Staggered ($Re_{D,max} > 1000$)	0.64	0.76	0.84	0.89	0.92	0.95	0.97	0.98	0.99

Plate type heat exchangers involve integrating many thin grooved plates and effectively transferring heat from the hot fluid through the thin wall to the cold fluid on the other side of the plate shown in Figure 17.

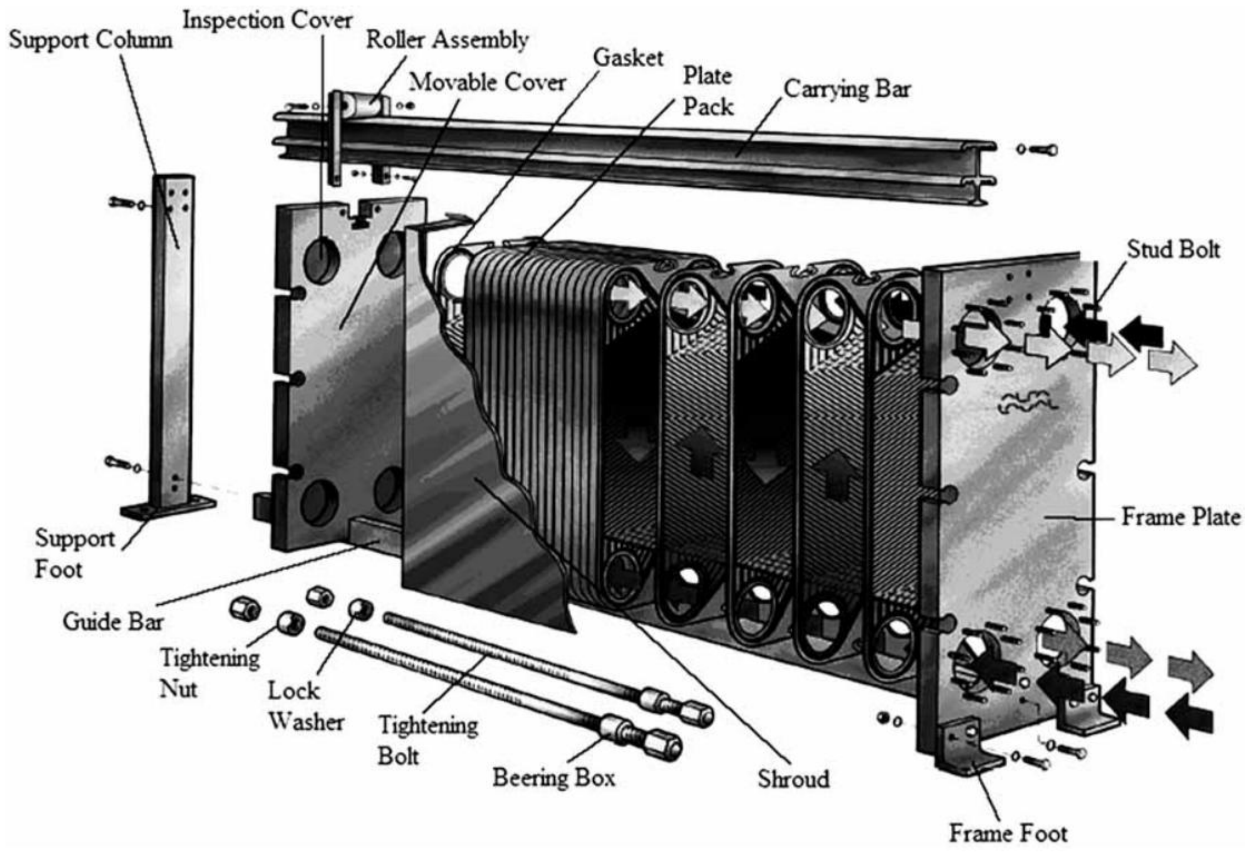


Figure 17. Counter Flow Plate Type Heat Exchanger [48]

Flow within plate heat exchangers between the gasketed plates is complex but since internal flow occurs Eqs. (83) and (84) may be used considering the hydraulic diameter (D_h) of the flow defined as:

$$D_h = \frac{4A_c}{P} \tag{94}$$

where A_c is the flow cross-sectional area and P is the wetted perimeter. Since flow is usually only laminar for a short period of time at startup, Eq. (82) can be used as a reasonable approximation for laminar flow in non-circular cross-section (through it's meant truly for flow in circular pipes). Values for laminar flow in non-circular cross-section $\overline{Nu}_D$ vary between approximately 2.5 to 8 [20-22] and the $\overline{Nu}_D$ being 4.364 for short period of time does not affect the steady state heat transfer performance.

3.5 Steam Turbine Model

The steam turbine was modeled as a multi-stage impulse turbine. Since the condenser and feedwater pump pressures and temperatures were not needed for the trainer, they were not modeled along with the boiler. Boiler pressure and temperature were assumed to be constant inputs into the turbine model. If the condenser was modeled, a technique similar to the one developed for a shell and tube heat exchanger would have been used and the feedwater pump would have been model similar to Eq. (76). Readers interested in boiler dynamic simulation and control should consult literature [30-38] and for detailed boiling heat transfer then use the Chen Correlation [21]. Figure 18 shows a schematic of the type of steam turbine modeled in the simulation with N_z number of control valve/nozzles, N_c number of Curtis stages, and N_r number of Rateau stages.

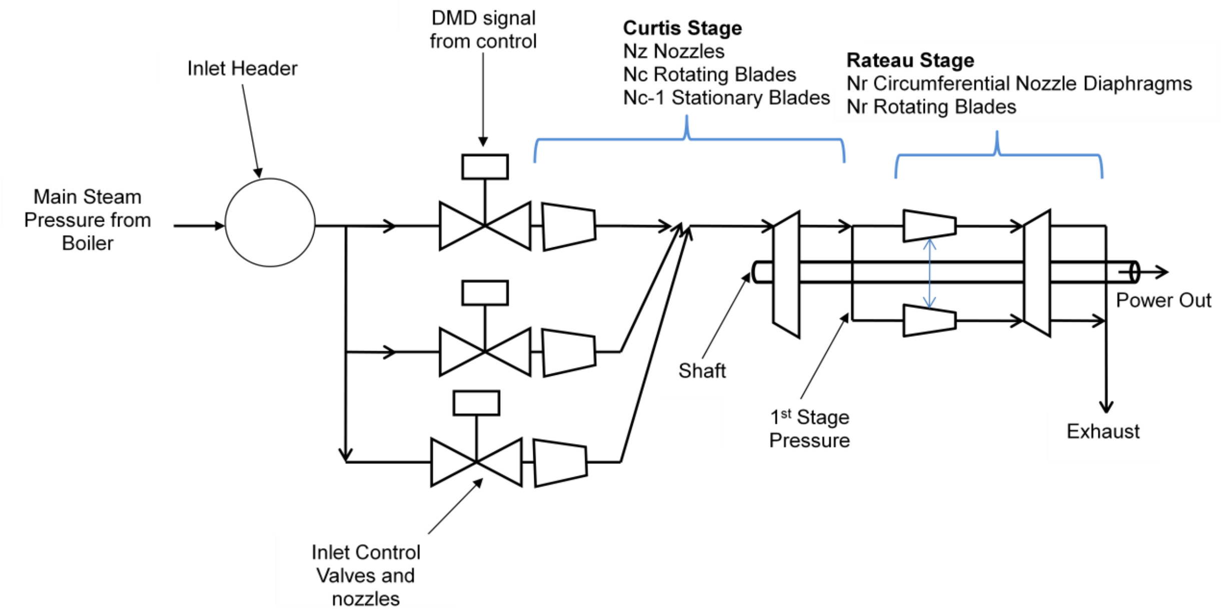


Figure 18. Steam Turbine Model Schematic

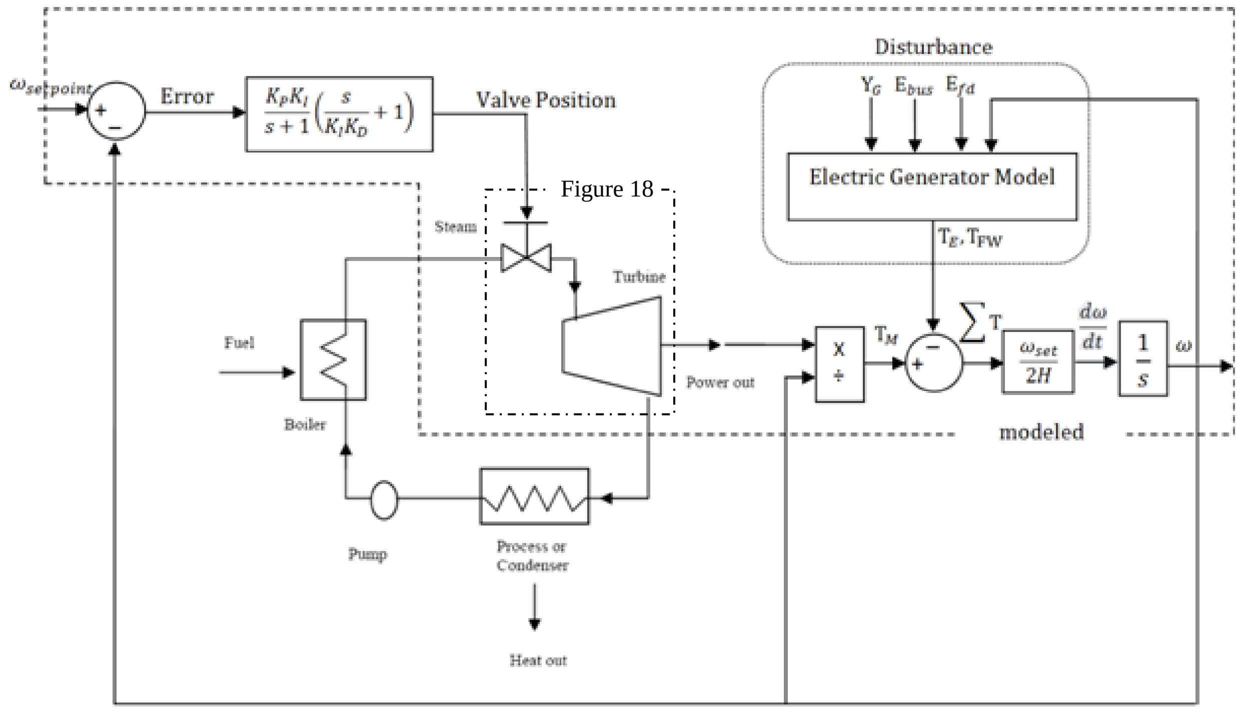


Figure 19. Block Diagram of Steam Turbine Control, Power Production, and Electrical Generator.

Figure 19 shows the relationship between prime mover model (Figure 18), generator model, and control. Only the turbine portion of the Rankine cycle is modeled. In this case, the valve position demand was an input to the model from the Woodward control.

Turbine Blade Stages

The two types of turbine stages modeled in the simulation are pressure-velocity compounding (Curtis) stages and pressure compounding (Rateau) stages, both are forms of impulse type stages. The pressure-velocity relationship of both stages can be seen in Figure 20 and Figure 21 (note “M” and “F” stand for moving and fixed respectively). For the Curtis stages the steam flow is accelerated through the nozzle while the pressure drops sharply across the nozzle. Then pressure remains constant while the steam velocity decreased over the moving and fixed blades, but more velocity dropped over the moving blades. In the Rateau stages the flow is accelerated through the nozzle diaphragms while pressure drops. The moving blades then decelerate the flow while remaining at constant pressure. The nozzle areas expand as flow progress through turbine due to the expanding steam to achieve repeating stages (constant velocity profiles and mean radius). In simulation, both of these cases are assumed by a rated pressure drop at maximum flow condition where velocity changes have no effect, but mass flow does as pressure drop across stage is a function of mass flow.

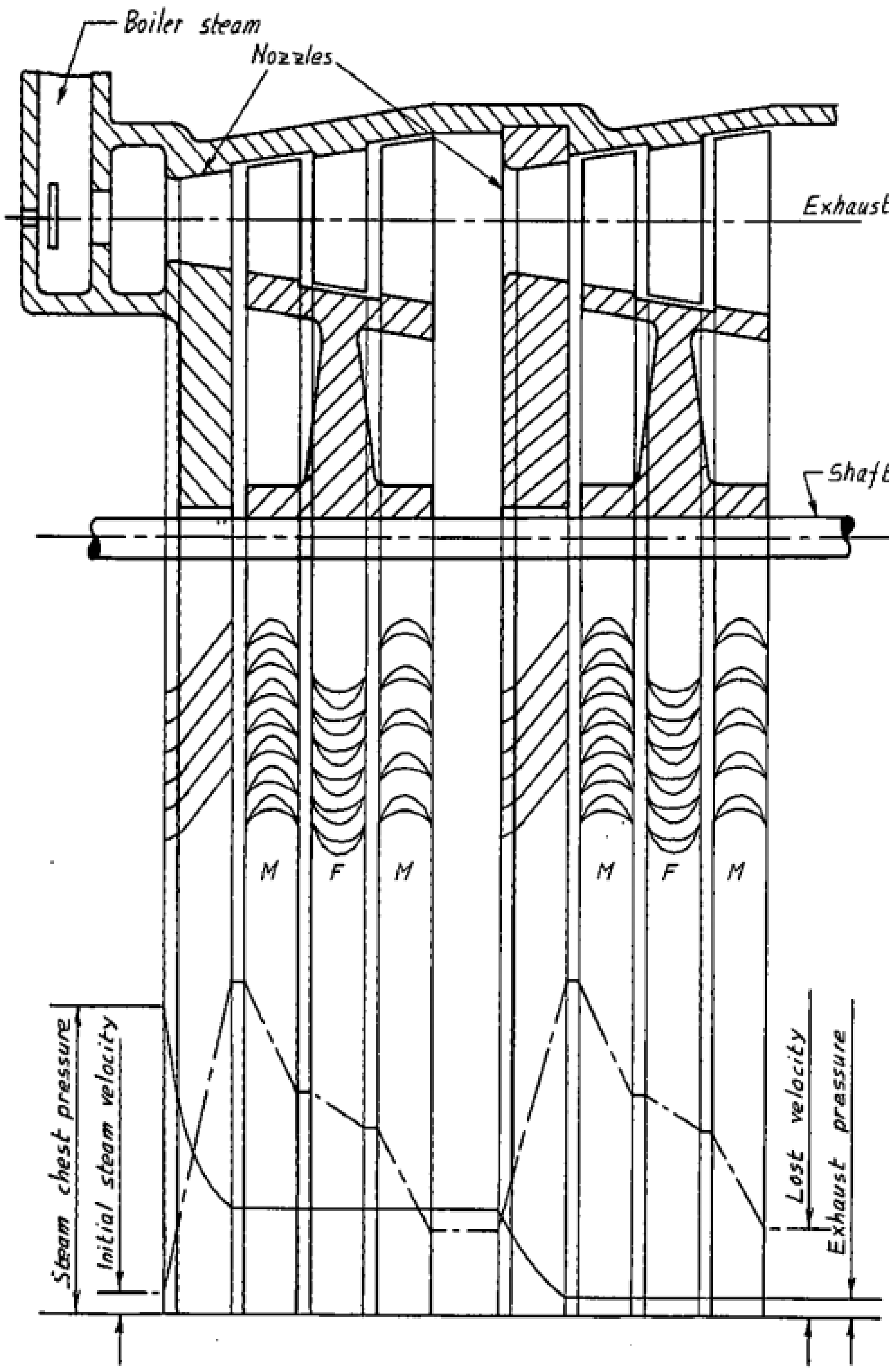


Figure 20. Pressure-Velocity Diagram for Curtis Stage [39]

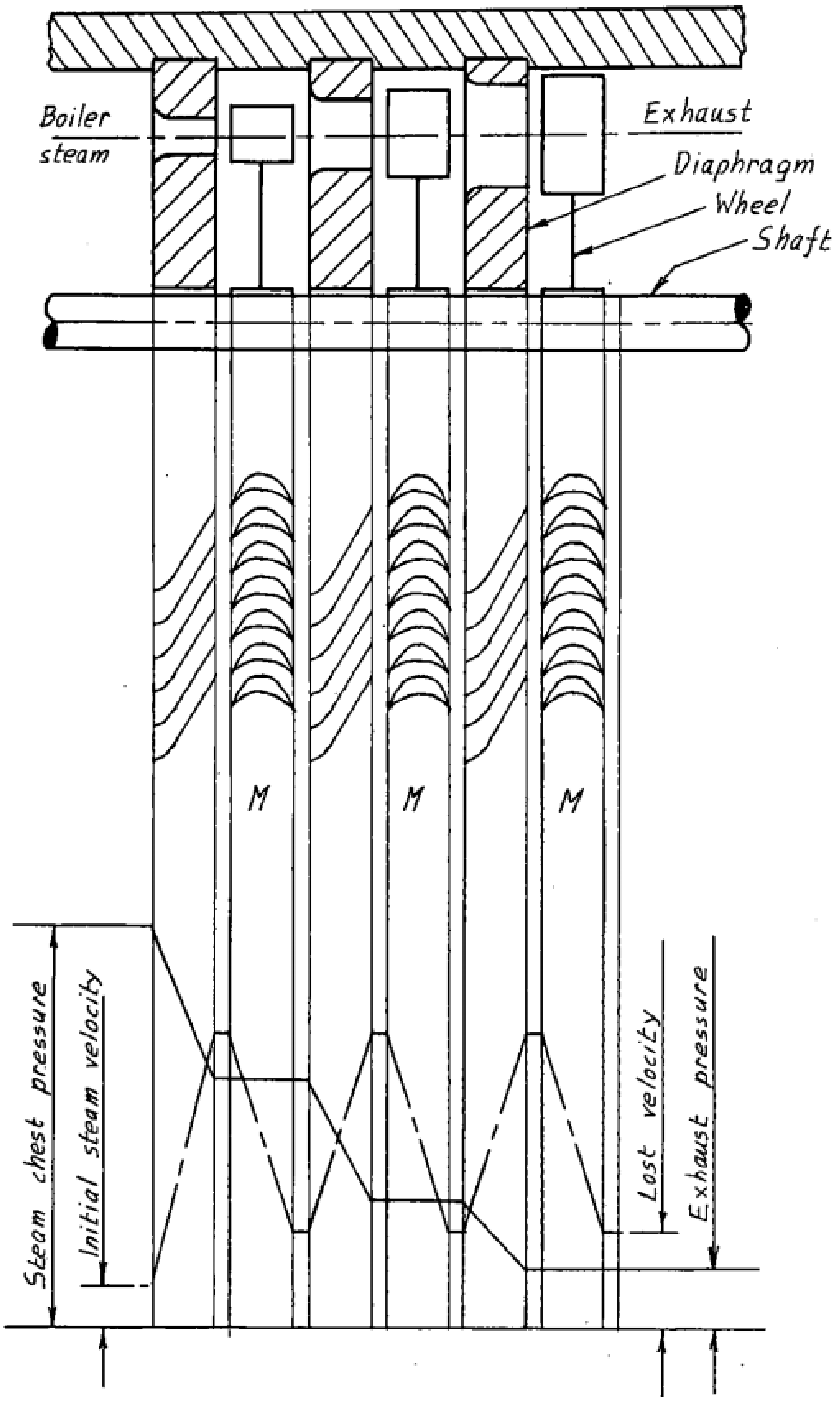


Figure 21. Pressure-Velocity Diagram for Rateau Stage [39]

Simulation method

Since detailed information about velocity diagrams (which can be deduced from blade geometry and techniques to derive power output in [40]), were unavailable and are only useful at design conditions [1], a pressure drop and enthalpy drop approach was taken to model power output. Since condenser and boiler are not dynamically modeled, the steam properties at the inlet and outlet of the steam turbine are fixed as constant boundary conditions. The main parameters calculated are the derivatives of specific enthalpy ($\dot{h}$) and pressure ($\dot{p}$) based on steam flow through the first stage and second stage and flow resistances/restrictions. The first stage with N_c number of Curtis stages can be simplified and lumped into one single flow resistance. The same can be said for the N_r number of stages in the second stage. Therefore if two enthalpy derivatives and two pressure derivatives are accounted for; the turbine state and power output is fixed. As power output is the main modeling objective; power output is found by correlating the steam mass flow, enthalpy drop, assumed isentropic efficiency, and valve position demand.

Within the steam power model, the control valve signal is used to vary the flow coefficient simulating the valve. The steam then flows through the nozzles and concentrated jets are shot at turbine blades (First Stage), which are modeled by another resistance. The pressure between the control valve and first stage is calculated by comparing 4 cases: first being both control valve and nozzle are choked (typical of higher loads), only control valve is choked, only nozzle is choked, and neither control valve nor nozzles are choked. These 4 cases are calculated simultaneously and the case with the least mass flow difference between control valve and nozzles is chosen. This is done because of the discontinuity caused by different equations used to calculate mass flow based on non-choked or choked flow described in Eq. (95) which is non-choked flow and Eq. (96) which is choked flow for an ideal gas:

$$\dot{m} = CA\sqrt{\rho\Delta p} \tag{95}$$

$$\dot{m} = CA\sqrt{\kappa\rho p_{in}\left(\frac{2}{k+1}\right)^{\frac{\kappa+1}{\kappa-1}}} \tag{96}$$

where $\dot{m}$ is the mass flow, C is the meter coefficient, A is the cross-sectional area, ρ is the fluid density, Δp is the pressure drop, p_{in} is the inlet pressure to flow through an orifice, and κ is the specific heat ratio. It is common in practice to define:

$$K = CA \tag{97}$$

where K is the flow coefficient (similar to C_v) encompassing area and flow. And for incompressible flow (ρ being constant) Eq. (95) looks similar to Eq. (4) where C_v and Eq. (4) become:

$$C_v = \frac{CA\sqrt{\rho}}{\rho} \; or \; C_v = \frac{K\sqrt{\rho}}{\rho} \tag{98}$$

$$\begin{gathered} \dot{V} = \left(\frac{CA\sqrt{\rho}}{\rho}\right)\sqrt{\Delta p} \\ and \; \dot{m} = \rho\dot{V} = \left(CA\sqrt{\rho}\right)\sqrt{\Delta p} = K\sqrt{\rho\Delta p} \end{gathered} \tag{99}$$

An important parameter not specified is the pressure between the control valve and nozzles/Curtis stage (p_{mid}) which needs to be calculated to find Δp between control valve and nozzle/Curtis stage. In the four cases in p_{mid} is calculated by forcing equal flow between both resistances; for both choked flow:

$$p_{mid} = \frac{{K_c}^2\rho_c\beta p_c}{{K_{curtis}}^2\rho_{mid}} \tag{100}$$

where β is defined as:

$$\beta = \kappa \left(\frac{2}{\kappa + 1}\right)^{\frac{\kappa+1}{\kappa-1}} \tag{101}$$

for control valve choked and nozzle un-choked:

$$p_{mid} = \frac{{K_c}^2 \rho_c \beta p_c + {K_{curtis}}^2 \rho_{mid} p_n}{{K_{curtis}}^2 \rho_{mid}} \tag{102}$$

for control valve un-choked and nozzle choked:

$$p_{mid} = \frac{{K_c}^2 \rho_c p_c}{{K_{curtis}}^2 \rho_{mid} \beta + {K_c}^2 \rho_c} \tag{103}$$

for both control valve and nozzle un-choked:

$$p_{mid} = \frac{{K_c}^2 \rho_c p_c + {K_{curtis}}^2 \rho_{mid} p_n}{{K_{curtis}}^2 \rho_c + {K_{curtis}}^2 \rho_{mid}} \tag{104}$$

where K_c and K_{curtis} are the flow resistance coefficients for the control valve and nozzle/Curtis Stage, ρ_c and ρ_{mid} are the densities at the control valve and nozzle inlet, p_c and p_n are the pressures at the control valve inlet and nozzle ext. A simplification that was made was that the inlet steam density was used to determine the flow through the control valve and 1st stage nozzles along with ratio of specific heats, though in reality density and ratio of specific heats will vary through valve and nozzle. Since when choked flow occurs, $\dot{m}$ calculated in Eq. (95) becomes too large, so the lowest mass flow is chosen to indicate flow is choked and then is purely a function of inlet pressure. The same is done for the second stage except only one mass flow is calculated based on the pressure difference or stage inlet pressure depending on non-choked or choked flow. The differences in mass flow exiting the 1st stage and entering the 2nd stage then can be used to find second stage inlet pressure which is determined by:

$$p \propto \int \Delta\dot{m} \tag{105}$$

$$\dot{p} = K_{pressure} \Delta \dot{m}$$

$$p = \int \dot{p}$$

where $\Delta \dot{m}$ is the difference between mass flows in and out of the stage inlet and $K_{pressure}$ is the gain to simulate capacitance. Now enthalpies can be calculated assuming and isentropic efficiency based on stage pressures. Using digitized steam tables and enthalpy derivatives, $\dot{h}_{12}$ (specific enthalpy derivative between 1st and 2nd stages) and $\dot{h}_{2,exit}$ (specific enthalpy derivative between 2nd stage and exit) calculated by:

$$\dot{h}_{out} = \frac{\dot{m}[\eta_t(h_{in} - h_{out,isen}) - h_{out}] - \dot{Q}_{loss}}{m}$$

$$h_{out} = \int \dot{h}_{out} \tag{106}$$

where η_t is the isentropic stage efficiency, h_{in} is the enthalpy in, $h_{out,isen}$ is the outlet enthalpy of the stage, and h_{out} is the current stage exit enthalpy. $\dot{Q}_{loss}$ is the heat loss to the environment/turbine blade enclosure, and m is the mass of steam present in the stage. Equation (106) essentially compares the steady state enthalpy calculated through constant efficiency to the current integrated stage enthalpy. Then once all the stage enthalpies are known power can be derived using the 1st law of thermodynamics for and open system:

$$\frac{dE_{cv}}{dt} = \dot{Q} - \dot{W} + \dot{m}_i\left(h_i + \frac{{V_i}^2}{2} + gz_i\right) - \dot{m}_e\left(h_e + \frac{{V_e}^2}{2} + gz_e\right) \tag{107}$$

and assuming steady state conditions where $\dot{E}_{cv}$ equals zero and $\dot{m}_i$ is equals $\dot{m}_e$ as well as kinetic energy, potential energy, and heat flows are all negligible compared to the steam's enthalpy difference; power generated by the turbine in the two stages on the shaft becomes:

$$P_{t,shaft} = \sum_{i=1}^{2} \left[\eta_{shaft} \dot{m}(h_{in} - h_{out})\right]_i \tag{108}$$

where n_{shaft} is the efficiency of steam power ($\dot{m}$ times Δh) transferred to the coupled generator shaft ($P_{t,shaft}$). The lubrication oil system for the steam turbine rotating bearings was also modeled using flow network and heat exchanger techniques discussed in Chapter 3.4 External Models from Engine along with bearing friction and heat transfer discussed later. Look up tables in the form of S-functions (details discussed in later chapter) for steam properties were based on ASME Standard [1] and used to calculate enthalpies and entropies at the steam turbine inlet, after the first stage, and exhaust of the turbine.

Parameter Calculations

Making the assumption that pressure drop across the control valve, nozzle, and turbine stages is linear with mass flow; K_c, K_{curtis}, and K_{rateau} are best found at a maximum rated condition/power output. Manufactures commonly give rated inlet pressure and steam mass flow, which are used with successive pressure drops (calculated through assumed control valve pressure drop, assumed first stage pressure at rated condition, and rated exhaust pressure) to find the K's for each pressure drop. The K's are reverse calculated by:

$$K = \frac{\dot{m}}{\sqrt{\min\,(case1, case2)}} \tag{109}$$

$$case1 = \rho \Delta p \tag{110}$$

$$case2 = \rho p_{in} \beta \tag{111}$$

where p_{in} is the inlet pressure to the specific flow resistance and Δp is the pressure drop across flow resistance.

Stage isentropic efficiency may also be estimated using the max design conditions by assuming a percentage of power is extracted in each lumped stage using the rated mass flow

$$h_{out,1} = h_{in,1} - \frac{x_{extract}P}{\eta_{shaft}\dot{m}} \tag{112}$$

where $x_{extract}$ is the percent of power extracted by the turbine blades in the 1st stage. The estimated enthalpy after the first stage then can be used with the inlet and exit enthalpy to derive both stage isentropic efficiencies. At turbine startup and low loads, the pressure at the first stage is not linear with mass flow, so Rateau stage flow coefficient K_{rateau} must be altered to account for most/to all of the pressure being dropped in the Curtis stage and very little pressure drop in the Rateau stage. A gain can be applied to K_{rateau} to increase the coefficient, increasing flow while decreasing the resistance, and in the case of the model Figure 22 best describes this relationship.

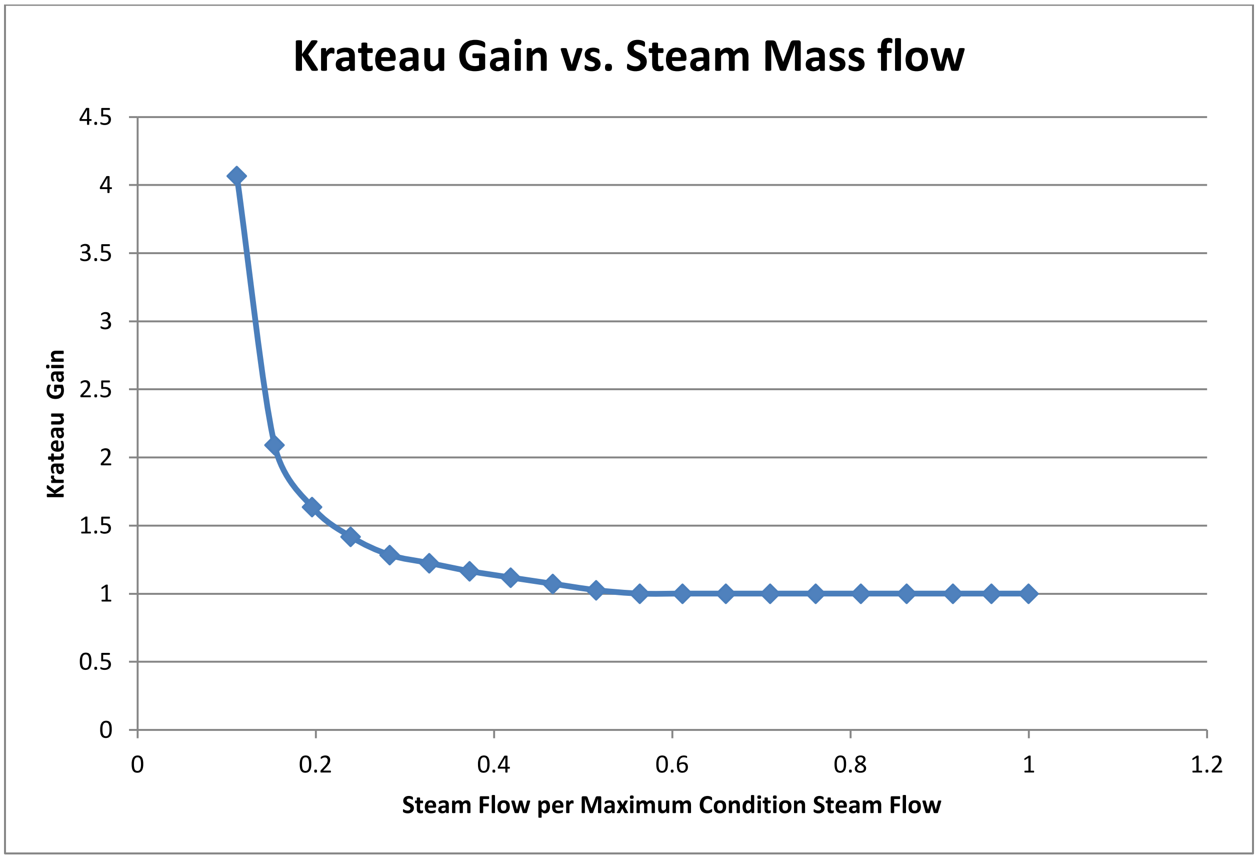


Figure 22. K_{rateau} Gain for Turbine Start-up and Low Loads

K_{rateau} has to be heavily modified at low steam flow to reflect the non-linear 1st stage steam pressure behavior but at higher steam mass flows the gain is not needed where flow and pressure drop become linear. It also must be noted that turbine valve position to K_c is not linear as well. In the case where zero percent valve opened equals zero turbine power output and 100 percent valve opened equals max turbine power output. At 100 percent valve opened, K_c (control valve flow coefficient) Gain equals 1, so the value equals what was calculated in Eq. (109) at maximum condition. K_c's non-linear behavior with valve position is illustrated in Figure 23.

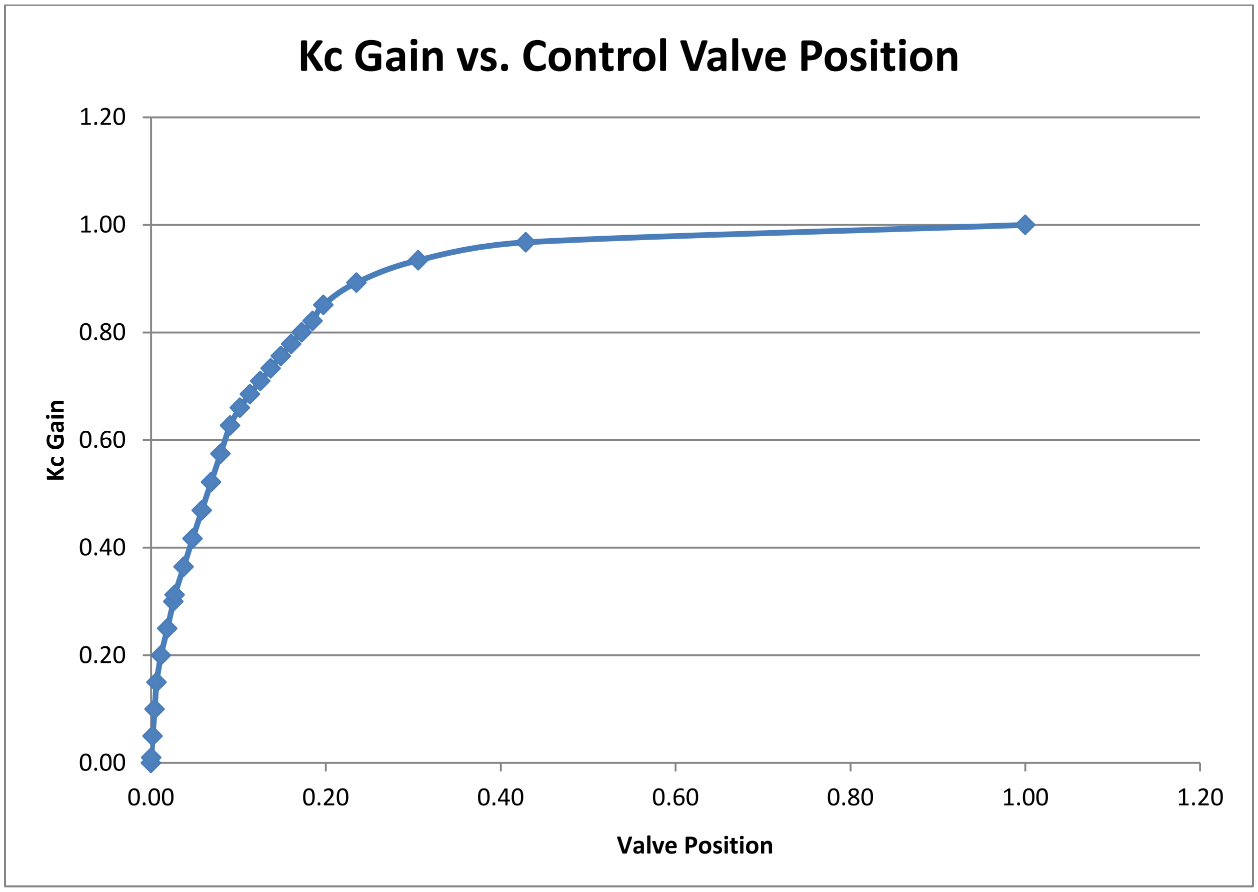


Figure 23. K_c Gain as it Varies with Control Valve Position

3.6 Generator Thermal Model

The main ways an electric generator produces heat is through the friction in the rotating bearings and through resistive heating caused by electrical power loss in the stator and field windings. Though the two heat flows might have a slight interaction with each other, it is assumed that either have no effect on each other except for the exchange of coolant seawater. In all cases of the prime movers, it is assumed that each electric generator has a Totally Enclosed Water to Air Cooler (TEWAC). Though, lubrication oil is handled differently depending on if prime mover and generator share a lubrication oil cooler or have separate cooling circuits for each.

TEWAC

TEWAC cooler construction incorporates a heat exchanger enclosed on top of the generator stator core and windings. Figure 24 illustrates the basic operation of an electrical generator TEWAC cooler. Air is circulated by density differences and in some cases assisted by a shaft mounted blower. Starting at the shaft mounted blower, air is accelerated through the fan into the air gap between the rotor and stator core. Then air absorbs heat from flowing over the stator windings and flowing vertically between the stator cores. The hot air rises and is returned to the heat exchanger, where the cooled air is returned to the shaft mounted fan. It can be assumed that synchronous exciter/ Automatic Voltage Regulator (AVR) produces negligible heat compared to the stator and field windings and its heat transfer neglected. Flow through rotor and stator air gap can be either symmetric or asymmetric, both described in Figure 25. It is assumed that in this application, the TEWAC cooler for each prime mover has asymmetric air flow. The technique to be presented to model heat transfer will also work for modeling half of symmetric air flow operation assuming cooling on both sides is symmetrically distributed.

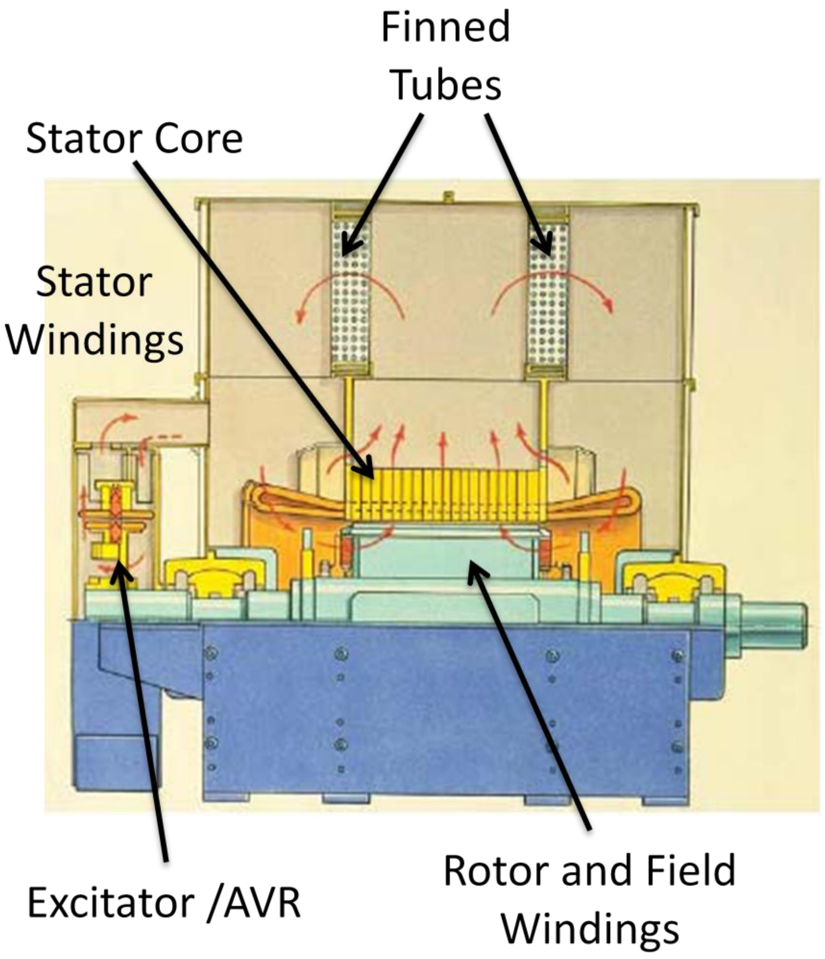


Figure 24. TEWAC Operation [41]

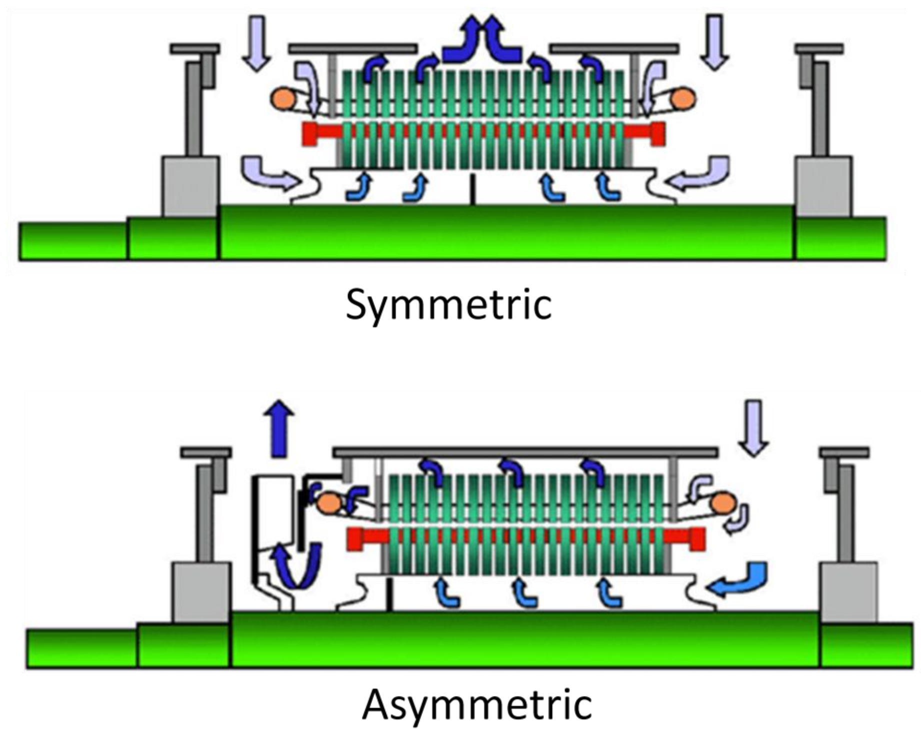


Figure 25. Comparison of Air flow: Symmetric vs. Asymmetric [41]

To estimate the heat transfer occurring within the stator core assembly and rotor, a control volume (or node) was created around each stator core, its gap between the stator cores, and corresponding axial rotor positions. Though the flow through the air gap and stator cores is complex three-dimensional flow through a cylindrical annulus (axial) and thin circular disks (vertical), only a singular cross-section is modeled radially from the center line and the heat transfer at the cross-section is assumed to be the same in the θ direction. The control volume separation and air flow network is depicted in (a preliminary sketch) Figure 26. This sketch gives an idea of the use of nodes to separate heat transfer calculation and the use of one cross-section to model the radial heat transfer.

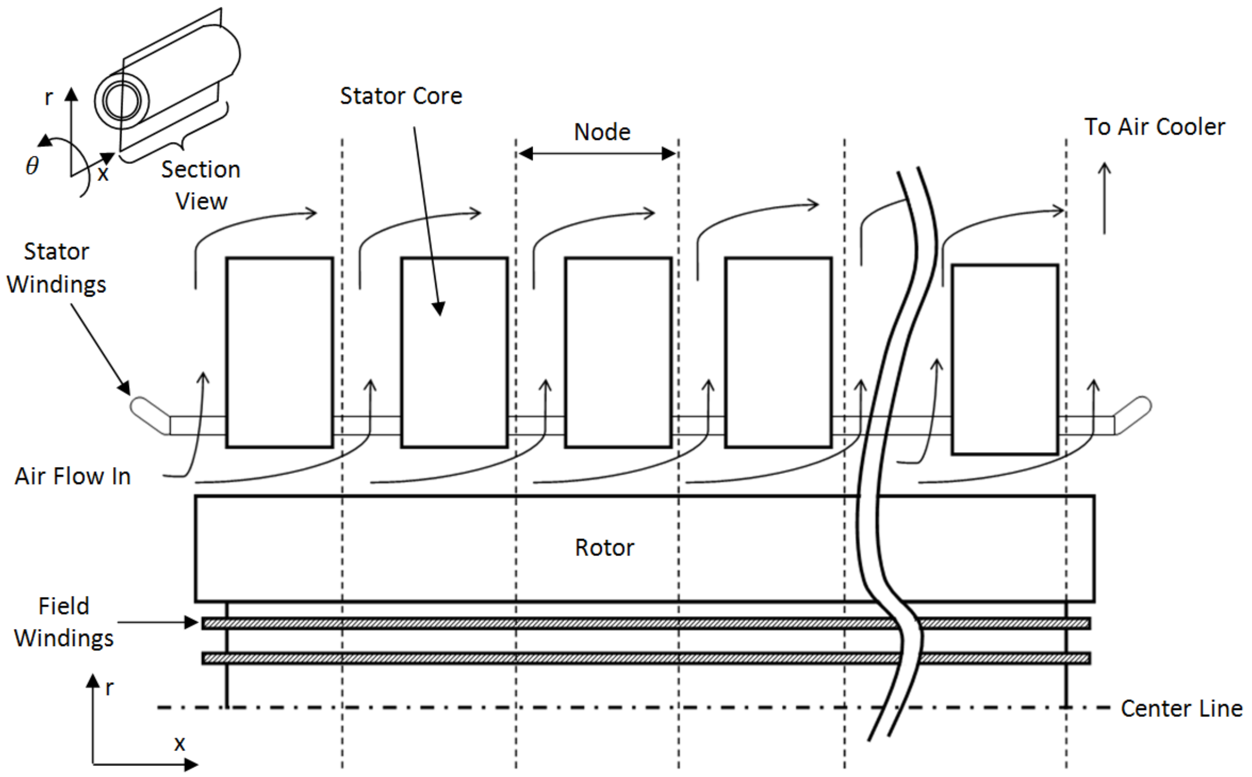


Figure 26. Generator Stator and Rotor Nodal Flow Network

Generator Heat Transfer

Using Figure 26 as a base, a heat transfer model was developed for the rotor and stator core assembles of the generator. Defining heat transfer per node, and replicating for how many stator cores on the generator, has the same effect as defining sections within a heat exchanger discussed earlier. Along with conforming to the 2nd law of Thermodynamics, heat transfer is not the same through every stator core. Since air flow is continually divided between stator core passages and the air gap between rotor and stator core, mass flow through the stator core passages is different for every passage, affecting the convective heat transfer. Air temperatures in and out of the nodes also vary due to air absorbing heat from the rotor, stator, and stator core. A simplification made is that the stator core is assumed to have a high thermal conductivity and transient 2-D/3-D thermal conduction is ignored. Therefore, we can assume a lumped capacitance approach where the stator core has one uniformly distributed temperature greatly reducing the complexity of heat transfer (though rigorously proven by calculating the objects Biot number ($Bi = hL_c/k$) is less than 0.1 [20-22]). The same assumption is made for the rotor as well. Placement of the nodes proved difficult define due to the heat transfer on the "back-side" of the stator core. The placement of the end of the node was eventually defined as just past the boundary layer of the "back-side" of the stator core. Figure 27 illustrates the interaction between conduction, convection, and radiation heat transfer.

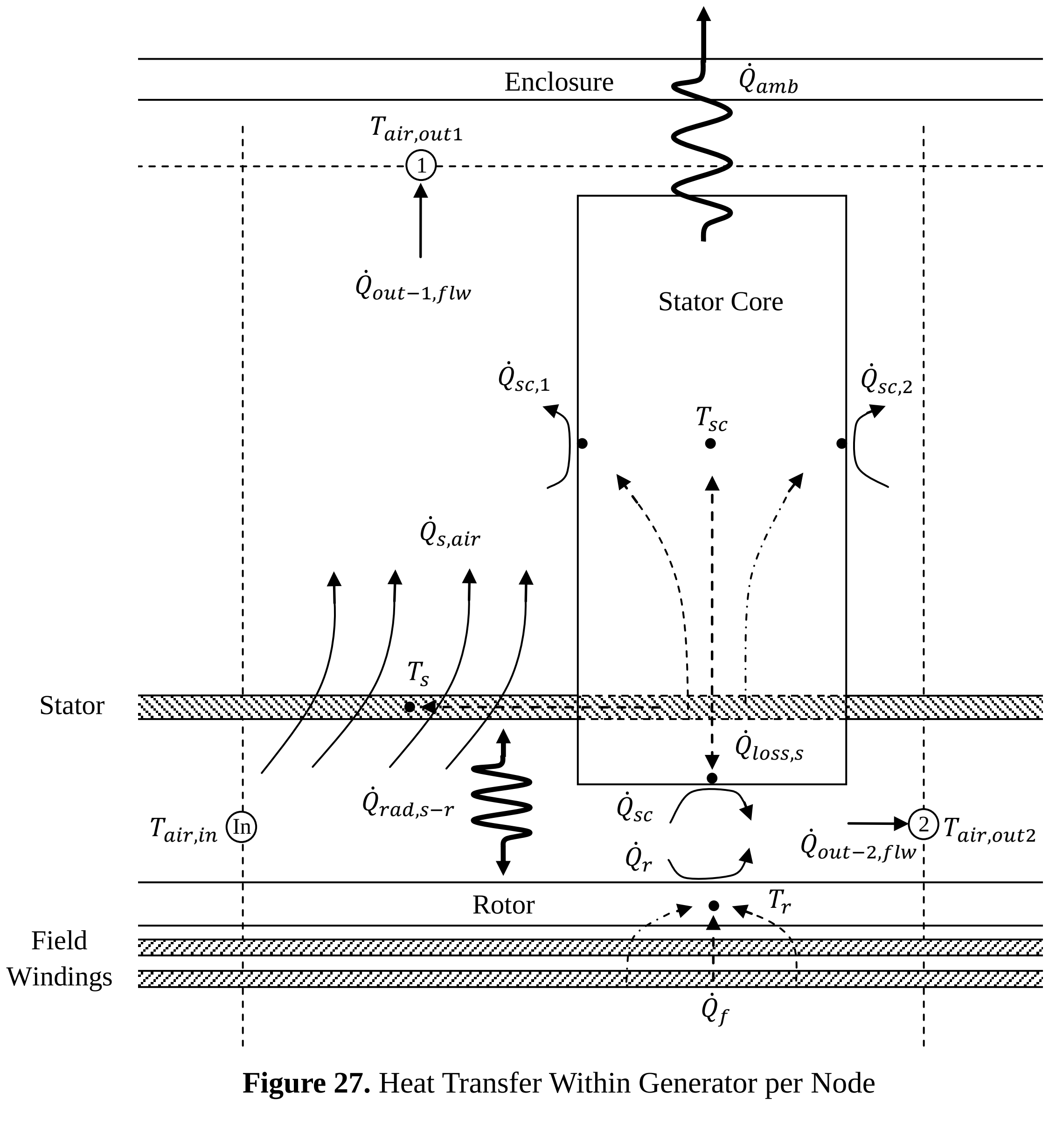


Figure 27. Heat Transfer Within Generator per Node

The five temperatures calculated per node are: the temperature leaving top of stator core gap ($T_{air,out1}$), the temperature leaving the rotor/stator air gap ($T_{air,out2}$), the lumped stator core temperature (T_{sc}), lumped stator winding temperature (T_s), and lumped rotor temperature (T_r). Heat flows and temperature derivatives are handled in a similar manner to the engine heat flows and Eqs. (43)-(53). The heat balances are as flows:

$$\frac{dT_s}{dt} = \frac{1}{m_s c_s}\left(\dot{Q}_{loss,s} - \dot{Q}_{rad,s-r} - \dot{Q}_{s,air}\right) \tag{113}$$

$$\frac{dT_{sc}}{dt} = \frac{1}{m_{sc} c_{sc}}\left(\dot{Q}_{loss,s} - \dot{Q}_{sc} - \dot{Q}_{sc1} - \dot{Q}_{sc2} - \dot{Q}_{amb}\right) \tag{114}$$

$$\frac{dT_r}{dt} = \frac{1}{m_r c_r}\left(\dot{Q}_{loss,f} + \dot{Q}_{rad,s-r} - \dot{Q}_r\right) \tag{115}$$

where subscripts s, sc, and r refer to stator, stator core, and rotor respectively. m and c are the mass and heat capacitance of the object. $\dot{Q}_{loss,s}$ and $\dot{Q}_{loss,f}$ refer to the resistive heating power loss (I^2R) in the stator and field windings. $\dot{Q}_{sc}$, $\dot{Q}_{sc1}$, and $\dot{Q}_{sc2}$ are the convective heat transfer from: the bottom of the stator core to the stator/rotor air gap, the "front-side" of the stator core to stator core air gap, and the "back-side" of the stator core. $\dot{Q}_{rad,s-r}$ is the heat radiated from the stator windings toward the rotor. $\dot{Q}_{s,air}$ is the convective heat transferred from the stator windings to the stator core air gap. $\dot{Q}_{amb}$is the convective and radiative heat transfer from stator core/enclosure to the surrounding environment.

For the heat transfer coefficient on all sides of the stator core, an isothermal flat plate correlation was used with Eq. (28) and coefficients found in [20-22]. $\dot{Q}_{rad,s-r}$ was modeled as purely radiative heat transferred of long (infinite) concentric cylinders (i.e. rotor within the stator windings) governed by [20]:

$$\dot{Q}_{rad,s-r} = \frac{\sigma A_r\left({T_s}^4 - {T_r}^4\right)}{\frac{1}{\varepsilon_r} + \frac{1-\varepsilon_s}{\varepsilon_s}\left(\frac{r_r}{r_s}\right)} \tag{116}$$

where ε is the materials emissivity, r is the radius from centerline, σ is Stefan-Boltzmann's constant, and A_r is the surface area of rotor exposed to the stator windings. There are some discrepancies that must be noted, since it is calculated per node. The arrangement is not truly long infinite concentric cylinders but sections of rotating rotor of Ω-pole sections (at lumped temperature T_r) surrounded by stator windings which bundled together around the θ direction (at lumped temperature T_s) represent the concentric cylinders. Without going into detailed view factor calculations, Eq. (116) greatly simplifies the radiative heat transfer and is good approximation. $\dot{Q}_{s,air}$ can be considered heat transfer from flow over aligned bundled tubes using Eqs. (87)-(93) and Table 3 and Table 4, except number of rows N_L equal to 1. $\dot{Q}_{amb}$ was calculated in a similar fashion as Eq. (19). $\dot{Q}_r$ was calculated using Re and Nu relations for 4-pole salient rotor in synchronous generators, found in literature [42], based on rotor rotational speed, rotor salient pole width, and rotor salient pole height.

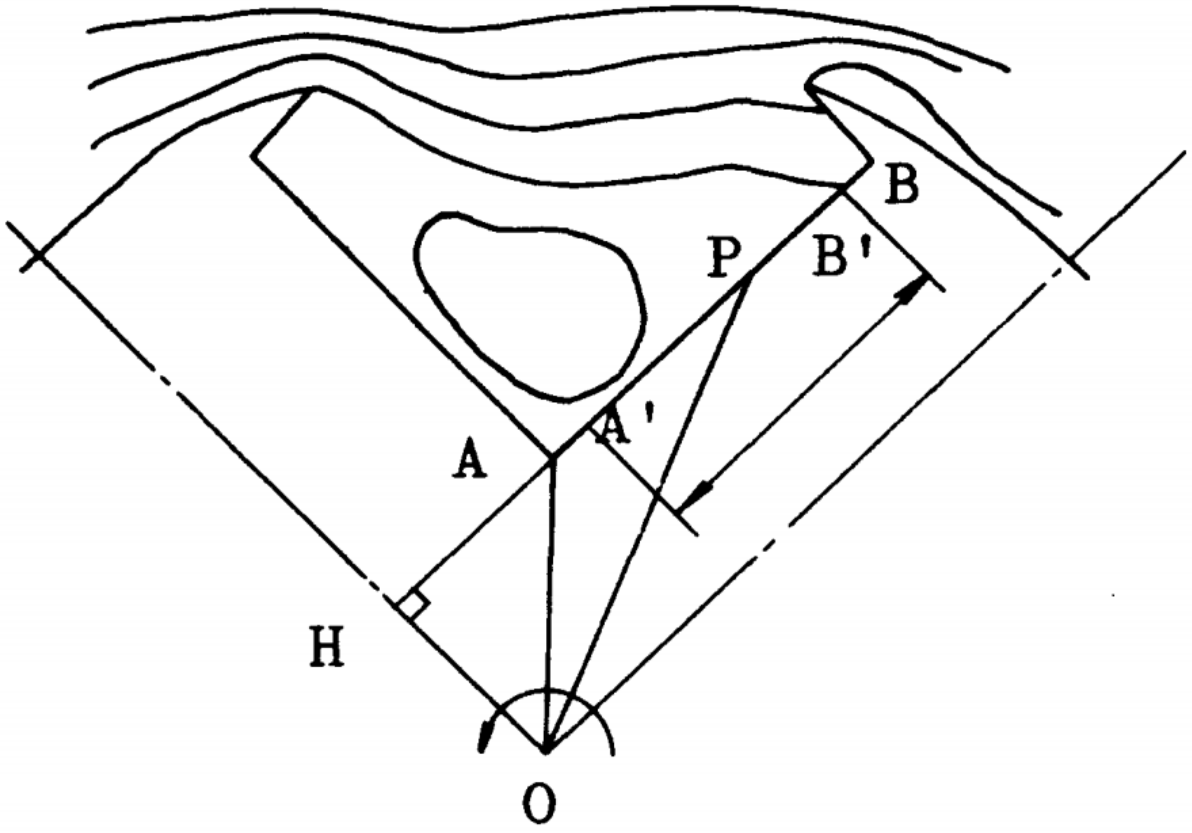


Figure 28. Gas Stream Lines around Poles and Windings [42]

The Re based on rotational speed Re_{ω} becomes:

$$Re_{\omega} = \frac{vL_h}{\nu} = \frac{\omega L_h L_w}{\nu} \tag{117}$$

where v is the gas velocity shown in Figure 28, ω is the rotor rotational speed, ν is the kinematic viscosity of the fluid. L_h and L_w are the lengths $\overline{AB}$ and $\overline{AH}$ in Figure 28, respectively. There are two cases of heat transfer considered; the Nusselt number for the field winding on the axial end portion ($\overline{Nu}_{\omega 1}$) is:

$$\overline{Nu}_{\omega 1} = \frac{\overline{h}L_h}{k} = 0.0171{Re_{\omega}}^{0.78} \tag{118}$$

and for the field windings on the straight portion is the average of the front and back side of the rotation ($\overline{Nu}_{\omega 2}$) is:

$$\overline{Nu}_{\omega 2} = \frac{\bar{h}L_h}{k} = \frac{0.148{Re_\omega}^{0.65} + 0.649{Re_\omega}^{0.54}}{2} \tag{119}$$

It is assumed that the heat transferred occurring at the field windings contributes to the temperature of the rotor by lumping the field windings mass into that of the rotor and considering the temperature to be the same in the rotor and field windings.

The outlet temperatures ($T_{air,out1}$ and $T_{air,out2}$) were calculated using technique described in Eq. (56) where $\dot{Q}_{f,flw}$ becomes $\dot{Q}_{out-1,flw}$ and $\dot{Q}_{out-2,flw}$, then solving for the outlet temperatures with known inlet temperature, mass flow, heat capacity, and heat flow. Referring to Figure 27, the heat added to the air flows at exit 1 and 2 are:

$$\dot{Q}_{out-1,flw} = \dot{Q}_{sc1} + \dot{Q}_{sc2} + \dot{Q}_{s,air} \tag{120}$$

$$\dot{Q}_{out-2,flw} = \dot{Q}_{sc} + \dot{Q}_{r} \tag{121}$$

where $\dot{Q}_{sc2}$ is added to the current node and the future node heat transfer of $\dot{Q}_{s,air}$ is ignored and only adds heat to the future node.

In the absence of air flow and pressure data, the resistances to flow between the stator cores and stator/rotor air gap were assumed to be constant. Therefore, air flow is assumed to continually spilt between each section by a certain percentage (i.e. 20 percent of flow goes up the gap between the stator cores and 80 percent continues to flow through the stator/rotor air gap). The shaft mounted blower was assumed to supply all the pressure to achieve required flow and density driven air flow affects (though present) were neglected. This gives an approximation of the mass flow decreasing through each node as it air travels through the stator core assembly and rotor/stator air gap towards the end. The decreasing mass flows lowers the air velocity and therefore reduces the heat transfer (heat transfer coefficients decrease for the same amount of heat transfer area) between the stator, windings, and rotor surfaces to the air. This creates lumped air and surface temperatures that gradually increase at each node as cooling fluid enters further into generator. This overall thermodynamic effect simulates what happens in an actual generator. Though many assumptions were made in the technique presented, this only roughly approximates generator (stator core and rotor assembly) thermodynamic behavior.

Finned Tube Heat Exchanger

The heat exchangers assumed to be on top of the stator core assembly are finned tube heat exchangers with a double pass of the seawater coolant. With multiple fins, conduction through the fins becomes significant and the cross flow heat exchanger model developed in earlier sections must be altered to incorporate an overall heat transfer coefficient for the air flowing over the fins, air flowing over the tubes, and conduction within the fins. The heat transfer was modeled in two sections for both passes of seawater coolant shown in Figure 29. This effectively splits the heat exchanger in half where the air temperature from pass 2 is sent to the inlet of pass 1 (at $T_{air,2}$). Conversely the seawater temperature from the outlet of pass 1 is passed to the inlet of pass 2 (at $T_{sw,2}$). The inlet and resultant outlet of heat exchange from both passes are $T_{air,1}$ and $T_{air,3}$ respectively. The same is denoted for seawater in and out with $T_{sw,1}$ and $T_{sw,3}$.

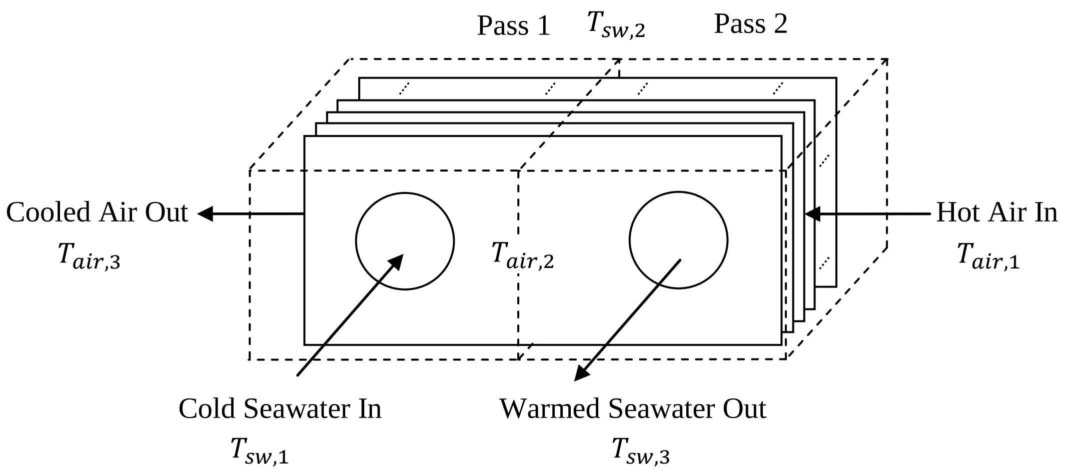


Figure 29. Counterflow Double Pass Finned Tube Heat Exchanger Flows

The heat flows within the heat exchanger are illustrated in Figure 30 for each pass. The subscripts in and out in this figure refer to each respective pass described in Figure 29. Heat flows balance in a similar way to previously presented heat exchangers. For the lumped temperature derivative of the tube base, the heat balance was:

$$\frac{dT_w}{dt} = \frac{1}{m_w c_w}\left(\dot{Q}_{UA} - \dot{Q}_{amb} - \dot{Q}_{sw}\right) \tag{122}$$

where $\dot{Q}_{UA}$ is the combined heat transfer from convection due to air flowing over tube base, convection due to air flowing over extended fins, and effects of conduction within the fins. $\dot{Q}_{amb}$ and $\dot{Q}_{sw}$ are heat flows from tube base to outside of heat exchanger and seawater coolant respectively. m_w is the mass and c_w is the heat capacity of the tube base.

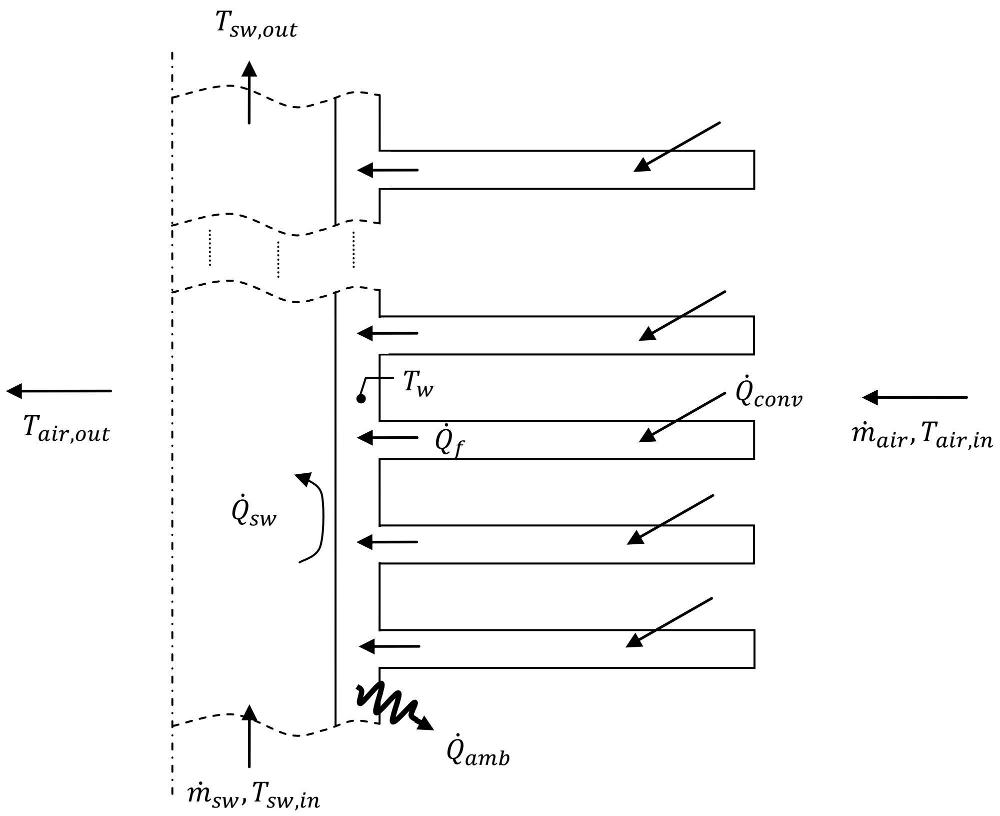


Figure 30. Heat Flow within Finned Tube Heat Exchanger per Pass

The convection portion of $\dot{Q}_{UA}$ is based on Re and Pr to $\overline{Nu}$ relations using Eqs. (27)-(29) for flow over flat plates (laminar and turbulent) and correlations found in [20] for flow over a single cylindrical tube. An equation that (varies in form from the default Eq. (28)) for flow over tubes for all Re_D and $Pr \gtrsim 0.2$ is (Nu and Re based on outer tube diameter D):

$$\overline{Nu}_D = \frac{\overline{h}D}{k} = 0.3 + \frac{0.62Re_D^{\frac{1}{2}}Pr^{\frac{1}{3}}}{\left[1 + \left(\frac{0.4}{Pr}\right)^{\frac{2}{3}}\right]^{\frac{1}{4}}}\left[1 + \left(\frac{Re_D}{282{,}000}\right)^{\frac{5}{8}}\right]^{\frac{4}{5}} \tag{123}$$

The heat transfer from the fins can be obtained by finding the heat transfer coefficient of one fin and factoring in the fins conduction thermal efficiency η_f as well as the array of fins overall surface efficiency η_o. The parameter m is used to simplify calculation of η_f and is defined as:

$$m = \sqrt{\frac{\overline{h}P}{kA_c}} \tag{124}$$

where P is the perimeter of the fin's cross-section, k is the thermal conductivity of the fin, A_c is the cross-sectional area of the fin, and $\overline{h}$ is the average heat transfer coefficient of the fin estimated by a flat plate. For the case of straight uniform fins, Figure 31 defines the parameters for x, L, w, t, and A_c. The fin efficiency defined as the amount of heat transfer through conduction ($\dot{Q}_f$ versus the maximum amount through convection. Using parameter m and assuming an adiabatic tip (since t is smaller compared to w and L), η_f becomes [20-22]:

$$\eta_f = \frac{\dot{Q}_f}{\overline{h}A_f(T_b - T_\infty)} = \frac{\tan(\mathrm{mL})}{mL} \tag{125}$$

where A_f the surface area of the fin. Accounting for effect of multiple fins in an array on the base surface, η_o becomes:

$$\eta_o = \frac{\dot{Q}_t}{\bar{h} A_t (T_b - T_\infty)} = 1 - \frac{N_f A_f}{A_t}(1 - \eta_f) \tag{126}$$

where $\dot{Q}_t$ is the total heat transferred from the surface area A_t that encompasses the total fin area and exposed portion of the tube base. N_f is the number of fins attached to the tubes surface, A_t is defined as:

$$A_t = N_f A_f + A_b \tag{127}$$

where A_b is the surface area exposed on the tube base [20].

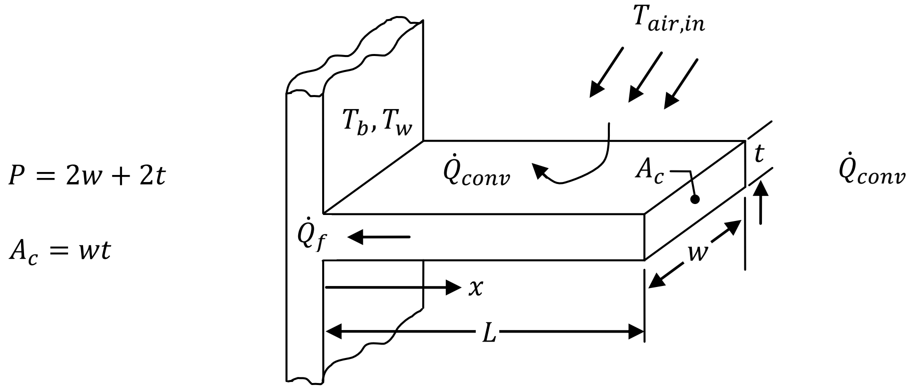


Figure 31. Diagram of Straight Fins of Uniform Cross Section

With the convection heat transfer coefficients and fin efficiencies know, the overall heat transferred from convection and conduction through the fin ($\dot{Q}_{UA}$) can be found. The overall heat transfer coefficient (U) multiplied by the contact area (A) is:

$$UA = \bar{h}_t A_b + \eta_o \bar{h}_f N_f A_f \tag{128}$$

where $\bar{h}_t$ and $\bar{h}_f$ are average heat transfer coefficient for flow over base tube and flow along the fin respectively. Using the lumped base temperature (T_b or T_w) and air temperature flow into each pass ($T_{air,in}$: either $T_{air,1}$ or $T_{air,2}$) the overall heat transferred to the fins from the air is (positive in the way temperatures are defined):

$$\dot{Q}_{UA} = UA(T_{air,in} - T_w) \tag{129}$$

Heat transfer for the seawater fluid inside the tube, $\dot{Q}_{sw}$, was calculated using internal flow Nu and Re based on diameter using Eqs. (82)-(84) and Eq. (48). The fluid temperatures $T_{air,out}$ and $T_{sw,out}$ coming out each pass is calculated in a similar fashion to Eq. (56) and solving for the exit temperature knowing $\dot{Q}_{f,flw}$ for each side of tube wall ($\dot{Q}_{UA}$ for air side and $\dot{Q}_{sw}$ for the seawater side).

Bearing Lubrication Oil Heat Transfer

It is assumed that all the prime movers have journal bearings holding the coupled shaft from prime mover to generator rotor shaft in place. Also, the steam turbines have bearings on the gear-train that convert the higher rotational speed steam turbine shaft to synchronous generator rotational speed. The lubrication oil flowing through the bearings serves a dual purpose where it lubricates the contact between the shaft and bearing and cools the contact area where the friction force between shaft and bearing is dissipated by heat. To simplify calculations only convection heat transfer is considered in the heat flow from the shaft and bushing to the lubrication oil. The journal bearings are assumed to be in a similar configuration to Figure 32 where lubrication enters at the top of the bushing and is drained out the bottom.

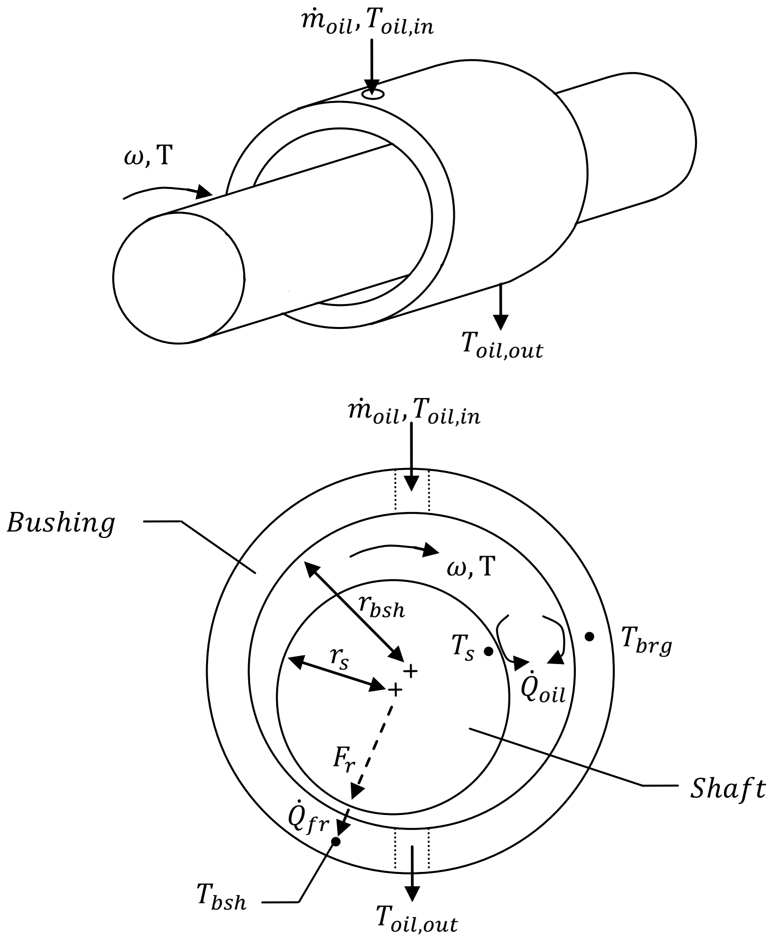


Figure 32. Journal Bearing Heat Transfer

The oil flowing out of the journal bearings is either sent to its own dedicated lube oil cooler for the generator side or is combined with the oil on the prime mover side and both are cooled by one heat exchanger. The separate lubrication oil coolers for the generator side are assumed to be shell and tube type coolers with seawater cooling calculations are the same as in the section Heat Exchangers: Shell-Tube and Plate.

Oil flow within the journal bearing is complicated by the rotation of the shaft within the bushing at speed ω and the offset of the between the bushing center point and journal center point. To simplify calculations: oil flow is assumed to flow axial within concentric tube annulus from the inlet to outlet, the temperature of the bushing (T_{bsh}) is assumed to be lumped into one temperature, the outside of the bushing is insulated (no heat is transfer to the environment), and the convective heat transfer occurring on both sides of the oil (the bushing and the shaft) is the same, essentially the shaft temperature (T_s) is the same as T_{bsh} making T_s one lumped temperature as well. This simplification avoids having to calculate the temperature distribution within the bushing and shaft created by the offset causing the friction force to be concentrated where the shaft and bearing meet. It also allows use of heat transfer correlations known for axial flow within an annulus and simplifies calculation of hydraulic diameter, D_h.

Heat generation due to the contact between the shaft and bearing ($\dot{Q}_{fr}$ or $\dot{Q}_{fr}''$) is based on ω and F_r. Equation found to correlate friction to heat transfer in literature [43]:

$$\dot{Q}_{fr}'' = \mu p r_s \omega \tag{130}$$

where μ is the coefficient of friction, p is the contact pressure, and r_s is the shaft radius. Knowing that the heat flow is constant heat flux multiplied area of heat transfer and p is force divided by area of contact (A_c) Eq. (130) becomes (assuming contact area equals heat transfer area):

$$\dot{Q}_{fr} = \dot{Q}_{fr}'' A_c = \mu p r_s \omega A_c = \mu \frac{F_r}{A_c} r_s \omega A_c = \mu F_r r_s \omega \tag{131}$$

where F_r is the resultant contact force of the shaft on the bushing. $\dot{Q}_{oil}$ is calculated in similar fashion to Eqs. (82)-(84) and Eq. (48) but using D_h for D. In the case for the assumption of concentric tube annulus the hydraulic diameter becomes (inner and outer diameter D_i and D_o respectively) [20]:

$$D_h = \frac{4(\pi/4)\left(D_o{}^2 - D_i{}^2\right)}{\pi D_o + \pi D_i} = D_o - D_i = 2r_{bsh} - 2r_s \tag{132}$$

where r_{bsh} is the bushing radius. Knowing the heat generated through friction and heat transfer through convection, the heat balance for the temperature derivative for T_{bsh} is:

$$\frac{dT_{bsh}}{dt} = \frac{1}{m_{bsh,s}c_{bsh,s}}\left(\dot{Q}_{fr} - \dot{Q}_{oil}\right) \tag{133}$$

where $m_{bsh,s}$ is the combined mass of the bushing and shaft. The same subscript notation for the heat capacitance of the combined objects, $c_{bsh,s}$. The outlet temperature of the oil $T_{oil,out}$ is calculated in a similar fashion to Eq. (56) and solving for the exit temperature knowing $\dot{Q}_{f,flw}$ to be $\dot{Q}_{oil}$.

3.7 Electrical Plant Grid and Bus Matrix

The technique used to model small AC power systems has been developed at Woodward, Inc. for use in simulating the electrical plant that Woodward's power management platform (PMP) controls are trying to manage. This allows for testing and validation of the PMP control software [1]. The technique for power system network calculations is developed in [44] and the main equations are discussed in this section, while an example power system grid will be worked in Appendix C. An example grid is used in place of the real grid due to International Traffic in Arms Regulations (ITAR) governing the real power system on board the Navy ship.

A small electrical power system grid is defined by the connection of power sources to load centers through nodes or buses, as well as bus-ties that connect different buses together (note bus-ties only have reactive load and no real load). Through the use of bus admittance matrices, the state of the electric grid (E, I, and Y) can be determined knowing the current added to the buses (I_{gen} or I_{bus}) and the current admittances on the grid (Y_{ab}), then calculating the bus voltage (E_{bus}). The subscripts on Y_{ab} are in order of effect-cause: as in the first subscript denotes the node/bus that the current is being expressed, and the second subscript is that of the voltage on the bus driving this component current. The collection admittances within the system (whether a connection exists or not) are collected into a symmetrical matrix (around principal diagonal) and matrix is termed Y_{bus}. The size of $\boldsymbol{Y_{bus}}$ is determined by the number of buses/nodes in the system (N_{bus}) and is a $N_{bus} \times N_{bus}$ square matrix. The admittances along the principal diagonal are termed self-admittances of the buses and each equals the sum of all admittances connected to bus, identified by the repeated subscript. All the other admittances are termed mutual admittances of the buses and each equal the negative of the sum of all admittances connected directly between the nodes identified by the subscripts [44]. The system is governed by the fundamental ohms law:

$$E = IR$$
$$Y = \frac{1}{R} \tag{134}$$
$$I = YE$$

and the general form for the source current toward the bus b of a network having N_{bus} number of independent buses is:

$$I_i = \sum_{n=1}^{N_{bus}} Y_{bn} E_n \tag{135}$$

In terms of generators, the current sourced by the generator to the bus it is connected to is the product of the generator's electromotive force (emf) voltage (E_{gen}) and admittance (Y_{gen}). For buses that do not have a current source but receive current through bus-ties (voltage is fixed due to the connecting bus-ties), I is set to zero. In allowing matrices and solving for unknown bus voltage for V_{bus}, Eq. (134) becomes:

$$\boldsymbol{V} = \boldsymbol{Y_{bus}}^{-1}\boldsymbol{I}$$

$$letting\ N_{bus} = n$$

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \dots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \dots & Y_{nn} \end{bmatrix}^{-1} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \tag{136}$$

It must be noted that admittances with the same subscripts but different order have the same admittance (i.e. $Y_{ab} = Y_{ba}$). Also, for systems with measurable capacitance the term impedance (Z) is used for R:

$$Z = R + jX \tag{137}$$

where X is the reactance and j is imaginary number of value $\sqrt{-1}$ creating a complex number having real and imaginary parts. Then Y with real and imaginary components becomes:

$$Y = \frac{1}{Z} = \frac{1}{R + jX} \tag{138}$$

As discussed earlier a detailed example electrical grid will be developed in Appendix C to demonstrate the usefulness of this technique using an admittance matrix to represent an electrical grid with generators, buses, breakers, and load centers (terminals of known amount of load connect to the bus(s)). The key advantage of admittance matrix is that it can capture the network's ability to allow current to flow without having to converting the network into its Thévenin or Norton equivalent circuit. Also, modification of the electrical network is made easier since only the matrix needs to be recreated versus having to recalculate the equivalent circuit which becomes cumbersome the more generators, buses, breakers, and load centers added to the network. Whereas in the matrix form, only rows and columns need to added or deleted to represent the new electrical network.

Chapter 4: Application and Results of Models

In this chapter, application and results of presented models will be discussed. Aspects of using MATLAB® Simulink® to simulate electric plant model as well as techniques learned through the project to increase the efficiency of generated code by Simulink Coder™ will be presented. The interface between the model, NetSim™, HMI, and supervisory training executive will be discussed in detail.

4.1 Application

All of the models discussed in the previous chapter are a part of an electrical grid plant model simulation an ocean-faring ship. Since this electric plant model is used for purposes of sailor training, it is necessary to run this model at or as close as possible to real time. It then becomes a tradeoff between simulation accuracy and the speed at which the model can be executed. The goal in modeling the engine was to simulate the dynamics of the turbocharger along with important temperatures and pressures the sailor sees on the HMI screen. Similarly, the goal of the steam turbine model was to simulate the overall thermodynamics occurring within the turbine section and the goal of the generator stator core thermal model was to capture the key dynamics of the heat transfer within a TEWAC generator.

Starting with the inherited plant model in MATLAB® Simulink®, the equations/models developed in the previous chapter were added (using function blocks) into multiple subsystems within Simulink®. The electric generator model functionality was not changed. The steam turbine transfer function for power was replaced with the dynamic enthalpy calculation discussed earlier. The diesel engine power transfer function was modified by the difference in ϕ_l and ϕ_{calc} and then the engine MVM was added using the values in Figure 5 as inputs. The Generator stator core thermal model was added in a similar fashion to all prime mover electrical generators with parameters such as the values of stator current and field current. The electrical grid model was modified to capture all of the buses, bus-ties, and load centers not previously modeled in the inherited model.

The completed model in Simulink® was then coded into C-code through the use of Simulink Coder™ (formerly Real-Time Workshop®). The C-code then was compiled in to an executable file, through the use of an external compiler, that interfaces with Woodward's control software simulation tool NetSim™. Through the use of NetSim™, the model sends/receives "hardware" inputs/out from the control GAP™. The interface is done through the computer's shared memory where a C-code based S-function within the Simulink® code receives (from control) and sends (to control) inputs and outputs of the model and keeps track of simulation time between the control software and Simulink® model. NetSim™ also interfaces the control to the HMI where hardware signals show up on the screen and through a graphical interface of the HMI; the operator has control over the simulated electrical plant. The NetSim™ interface can be best illustrated by Figure 33. The power system trainer was then realized using a supervisory executive. The training executive interfaced with NetSim™ simulation platform and HMI (shown in Figure 33). The supervisory executive is in control of tasks such as: user login, loading training scenarios, introducing tunables to control software code, and tracking user performance.

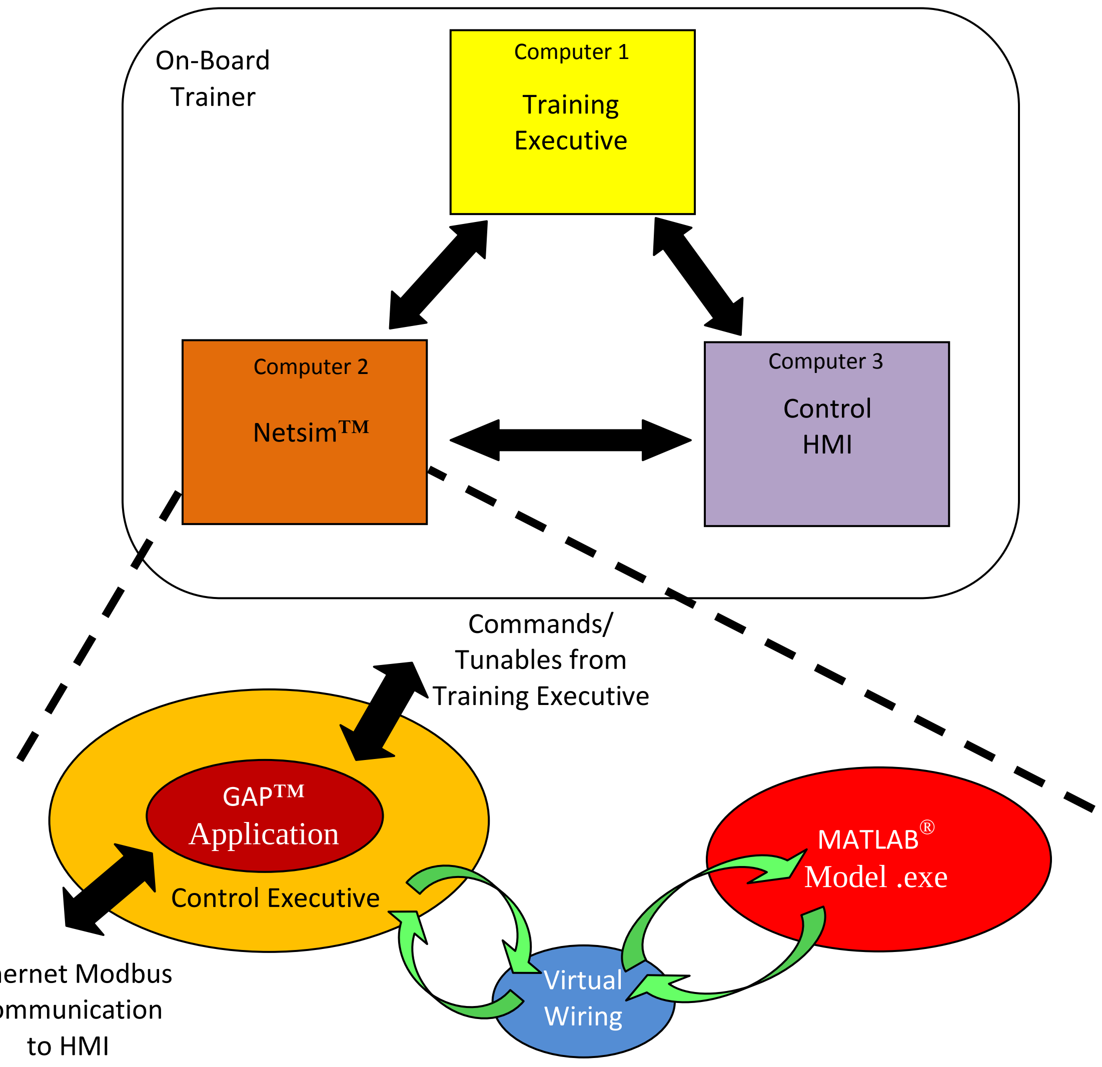


Figure 33. On-Board Trainer Architecture with Netsim™ and HMI

4.2 Results

Diesel Engine

Due to the lack of available transient data and physical specifications for the engines, the model was tuned to match steady state values versus set load. This said, various model parameters within the engine MVM could be adjusted to match transient engine performance should transient information become available. Engine parameters that were known were bore, stroke, displacement, rated fuel consumption, and rated power. Parameters such as turbocharger map (compressor and turbine), aftercooler dimensions, intake and outlet manifold sizes, external heat exchanger sizes, external pumps and circuit pressure gains/drops, and various engine efficiencies were all deduced from online sources[3, 29], previous Woodward projects[1], or reasonable and prudent judgments.

Data was available for key performance parameters versus engine/generator load (though the specific engine and its tabulated values versus load are protected from public viewing). Data such as fuel flow rate and steady state equivalent ratio were coded into the model as a function of engine load. Parameters unknown but critical to the performance of the engine MVM, such as the turbocharger, were borrowed from sources of engines similar in size. The turbocharger map [18] was borrowed from a publicly available turbocharger that closely represented the pressure, temperatures, mass flow rates, and power output found in the diesel engine on-board the ship.

The result of tuning the engine MVM to achieve steady state values of important performance parameters is that the model dynamics are "quasi"-validated through the reaching of steady state parameters versus certain loads. That is, the steady state value is achieved within a logical amount of time without noise or discontinuities. Figure 34 best describes the "reasonable" transients from an engine load step and parameters reaching steady state (compressor mass flow $\dot{m}_c$, non-dimensional engine speed ω_e, and calculated air to fuel ratio ϕ_{calc}). The engine MVM feedback ϕ_{calc} and ϕ_l for the power production transfer function (in Figure 4) was also adjusted to match reasonably with the non-linear behavior of the turbocharger. The values of the gains K_ϕ and $K_{\phi t}$ (from Figure 35) were adjusted to create a lag between naturally aspirated fuel power (P_{fuel}) to the gross supplied combustion power (P_G). If there was transient data on the power supplied by the engine for particular load steps available, the values of K_ϕ and $K_{\phi t}$ could be altered to match the non-linear performance of the real engine. Figure 35 shows the adjustment of correct power (P_{corr}) makes on P_G for the same load step used in Figure 34.

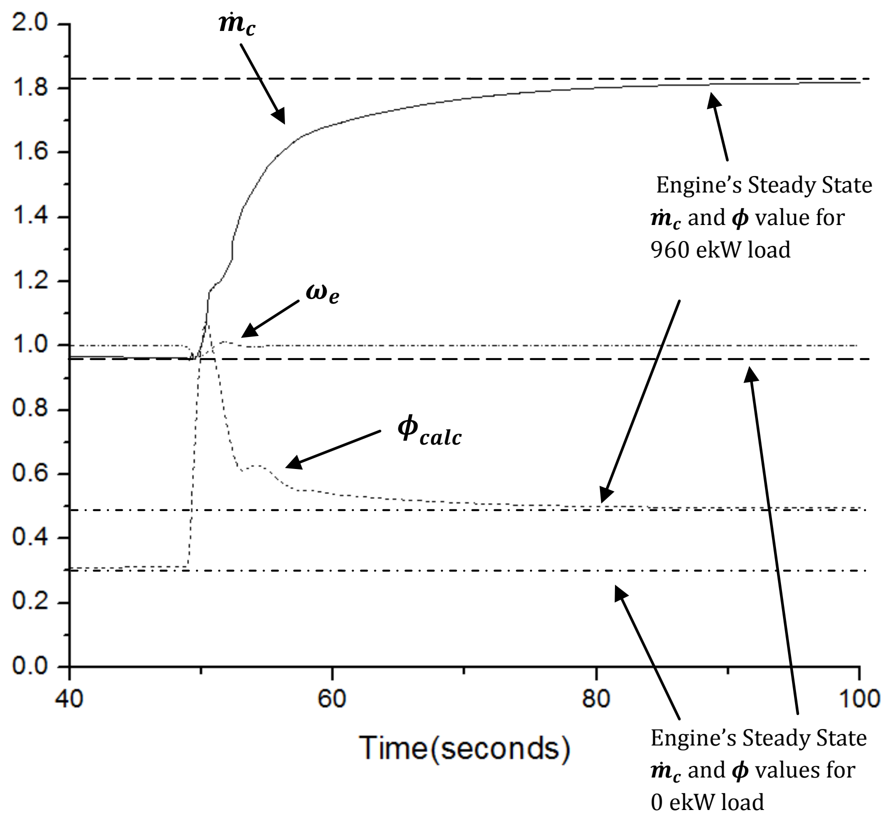


Figure 34. $\dot{m}_c$, ω_e, and ϕ_{calc} [model results]

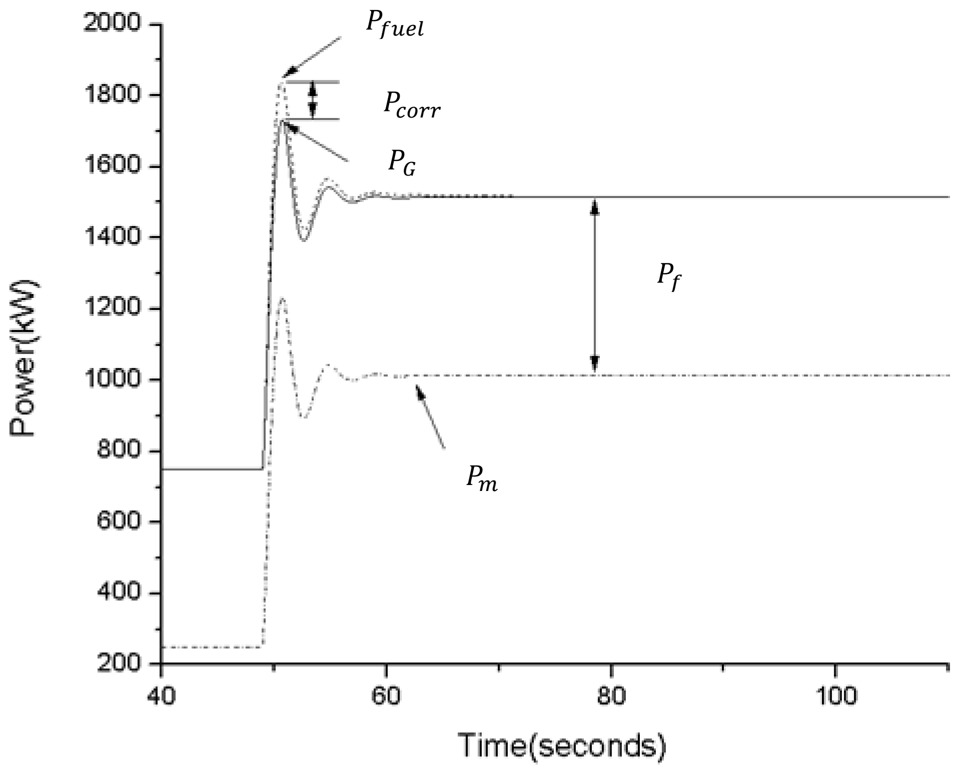


Figure 35. Power Production from Engine Transfer Function [model result]

Figure 36 through Figure 38 show other engine performance parameters for a certain load step from no load at rated speed to a different load than in previous Figure 34 and Figure 35. Then the diesel engine's circuit breaker is opened and unloaded back to no load at rated speed. For Figure 36 to Figure 38, load was added to the engine at 57 seconds and unloaded at 136 seconds. These figures show the transient response and steady state performance obtained from the mean value engine model.

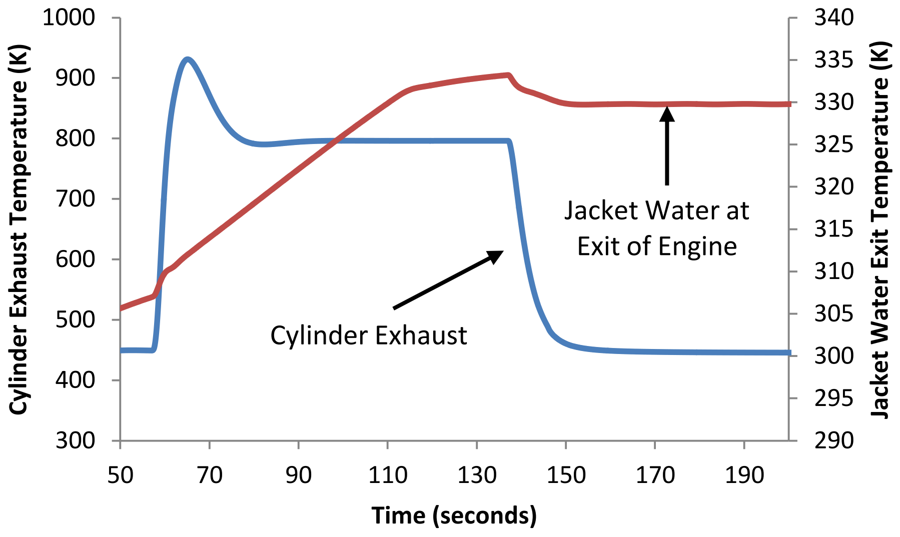


Figure 36. Cylinder Exhaust and Jacket Water Temperatures

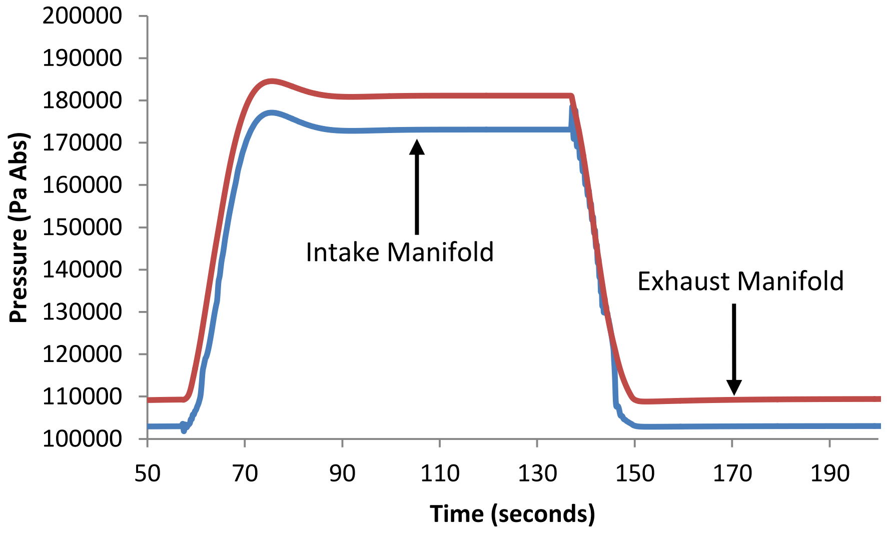


Figure 37. Intake and Exhaust Manifold Pressures

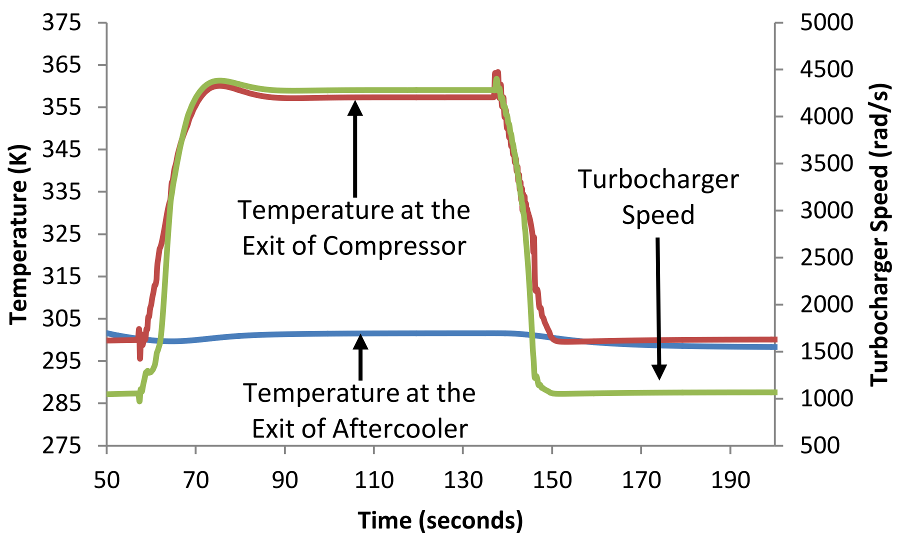


Figure 38. Aftercooler Air Inlet and Exit Temperatures with Turbocharger Speed

Table 5 shows how accurate steady state values for engine performance parameters were achieved with the MVM model versus manufacture's data. The slight differences in compressor outlet temperature and intake manifold temperatures are created by the different inlet ambient air temperatures and (indirectly through SCAC coolant) the inlet seawater temperatures. This can be noticed because the error for rated speed and 50% load conditions have the same exact error difference of approximately 5.0 K and 9.0 K for the compressor outlet and the intake manifold. While the values for compressor flow ($\dot{m}_c$), compressor pressure ratio (Π_c), Combustion temperature increase (ΔT_{eg}), and equivalence ratio (ϕ_{calc}) were tuned through various parameters (η_V , x_e, T_{loss}, η_c , η_t , etc.) to achieve the value within approximately 0.1%.

Table 5: Steady State Error from MVM

Parameter	Rated Speed		50% Load	
	S.S Error	Error %	S.S. Error	Error %
Mass Flow Compressor [kg/s]	1.912E-04	0.021	0.000678	0.041
Compressor Pressure Ratio	7.177E-04	0.069	0.002487	0.145
Combustion Temperature Increase [K]	1.9	0.676	0.3	0.061
Equivalence ratio	1.333E-04	0.049	0.000184	0.037
Compressor outlet temperature [K]	5.15	1.657	5.05	1.393
Intake Manifold Temperature [K]	9.05	2.941	9.25	2.976

Steam Turbine

The steam turbine model was tuned to reach steady values for a certain valve position demand correlation to a steam turbine power or speed set point. The main values that affected the steady state power output of the turbine model were the Rateau stage flow coefficient (K_{rateau}) and the control valve flow coefficient (K_c) which were shown in Figure 22 and Figure 23 respectively. These values along with the rated inlet pressure, inlet temperature, inlet steam flow, and turbine exhaust/condenser pressure were used to adjust the turbines power output and 1st stage pressure to match given data. 1st stage steam pressure strongly correlates to the generator output power, and is usually an accurate estimate turbine power output knowing the 1st stage pressure easily obtained from a pressure transducer. Figure 39 points out this relationship where power and 1st stage pressure are linearly related except for low load conditions, where steam control valve is barely open and steam is trickling in the turbine [1]. Also, note that data was only available for approximately 55 percent load and rest of data was linearly extrapolated to estimate the 1st stage pressure at loads 55 percent and above. The data for this comparison was recorded from the PMP control on-board the ship. Figure 40 also shows that 1st stage pressure and steam mass flow are linearly related as well which is the case for actual turbines as well [1].

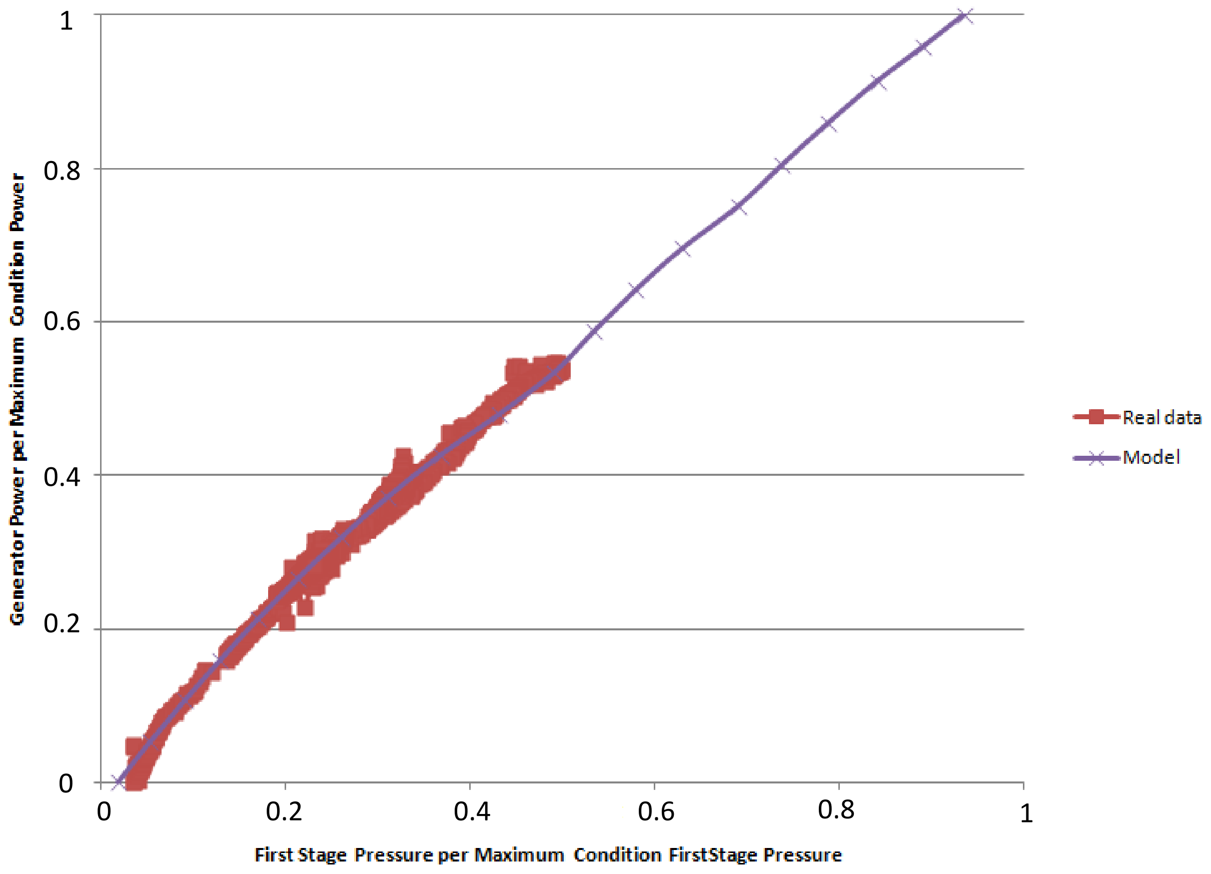


Figure 39. Comparison between Turbine Model and Actual Turbine for First Stage Pressure and Generator Power

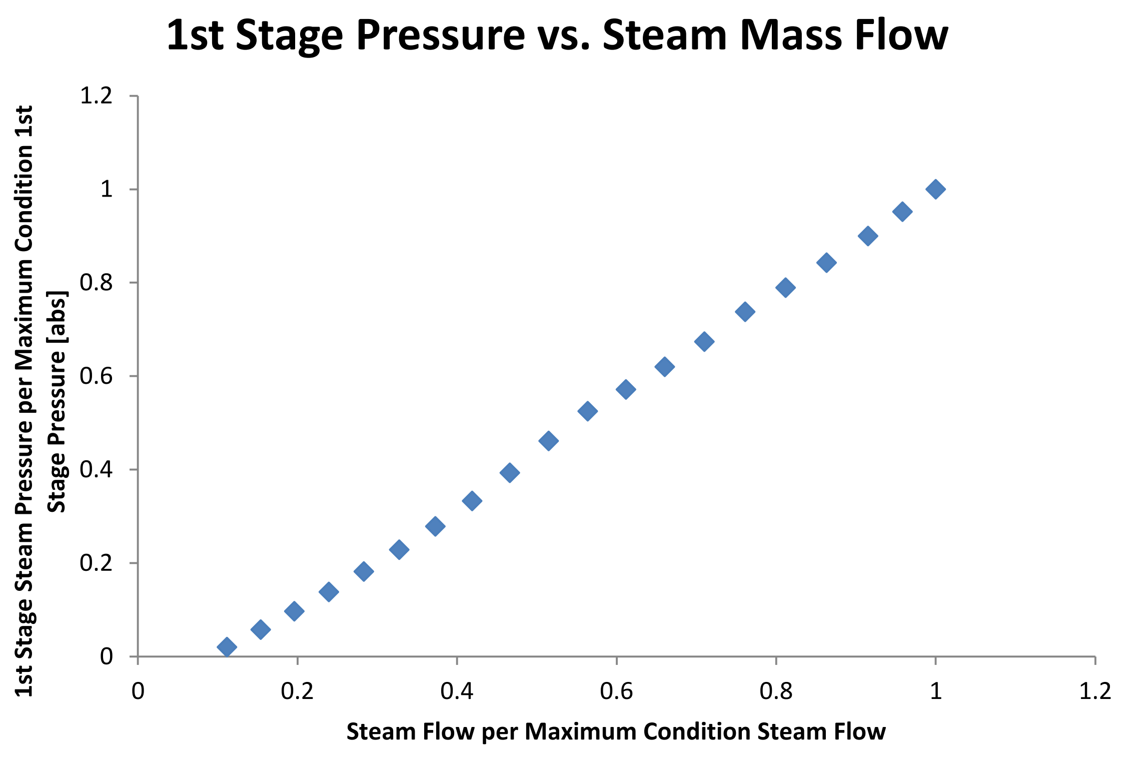


Figure 40. Relationship between 1st Stage Pressure and Steam Mass Flow

Factors affecting transient response with little effect on steady state performance of the turbine were the pressure derivative gain ($K_{pressure}$), the heat loss from the steam to turbine enclosure/outside environment ($\dot{Q}_{loss}$), and mass of steam within each stage (m: though assumed to be constant to simplify calculations). $K_{pressure}$ affects the speed at which the differences in mass flows, in and out of turbine stage, drive changes in stage pressure. At steady state the differences in mass flows is essentially zero and pressure settles to a constant value. $\dot{Q}_{loss}$, though the system was assumed to be adiabatic to the environment, will affect the speed at which enthalpies at each stage balance at steady state. The same goes for m, which is proportional to the mass flow of steam within each stage (i.e. the integral of steam mass flow equals steam mass in stage).

Stator Core Heat Transfer

The TEWAC generator thermal model performance is function of current flowing through the stator and field windings. The current increases through the stator windings as load increases, while the current remains relatively constant keeping the output voltage of the generator constant. In Figure 41, the average stator winding temperature and cooler air inlet temperature summarize the interactions between the heat flows generating temperature differences (shown in Figure 27) for a load step up to 50% load and unloaded back down to no load. The stator windings transfer heat to the air flowing over them increasing the air temperature, and then the cooling air rejects heat to the seawater reducing the air's temperature at the exit of the cooler. Figure 41 shows the TEWAC cooling system approaching steady state equilibrium for a constant generator load and the equilibrium state for unloading of the generator.

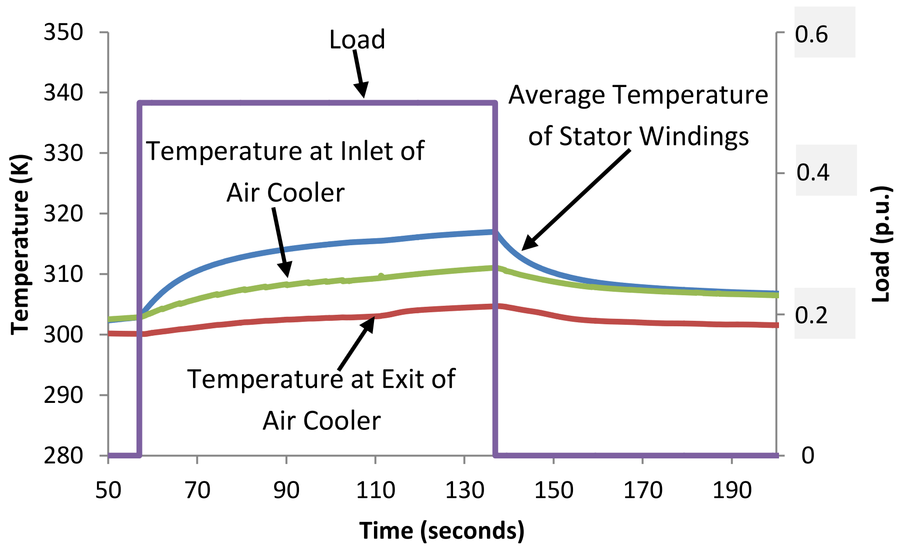


Figure 41. Air Cooler Temperatures and Average Generator Stator Temperature

Journal Bearing Heat Transfer

The bearing lubrication system in both diesel engines (generator bearings) and steam turbines (turbine and generator bearings) used the journal bearing heat transfer technique to calculate temperatures and flow networks technique to calculate pressures and mass flows. Figure 42 shows the result of the journal bearing model and oil cooler for the steam turbine where speed and temperatures are plotted versus time. Frequency is shown against the temperatures since the rotational speed is proportional to the frictional heat generated in the journal bearing using Eq. (131). Other oil bearing temperatures vary due to bearing force on bushing and oil flow rate within bearing. The figure shows the transient behavior of the system and the cooling system reaching a steady state equilibrium maintaining a temperature difference between the inlet and outlet.

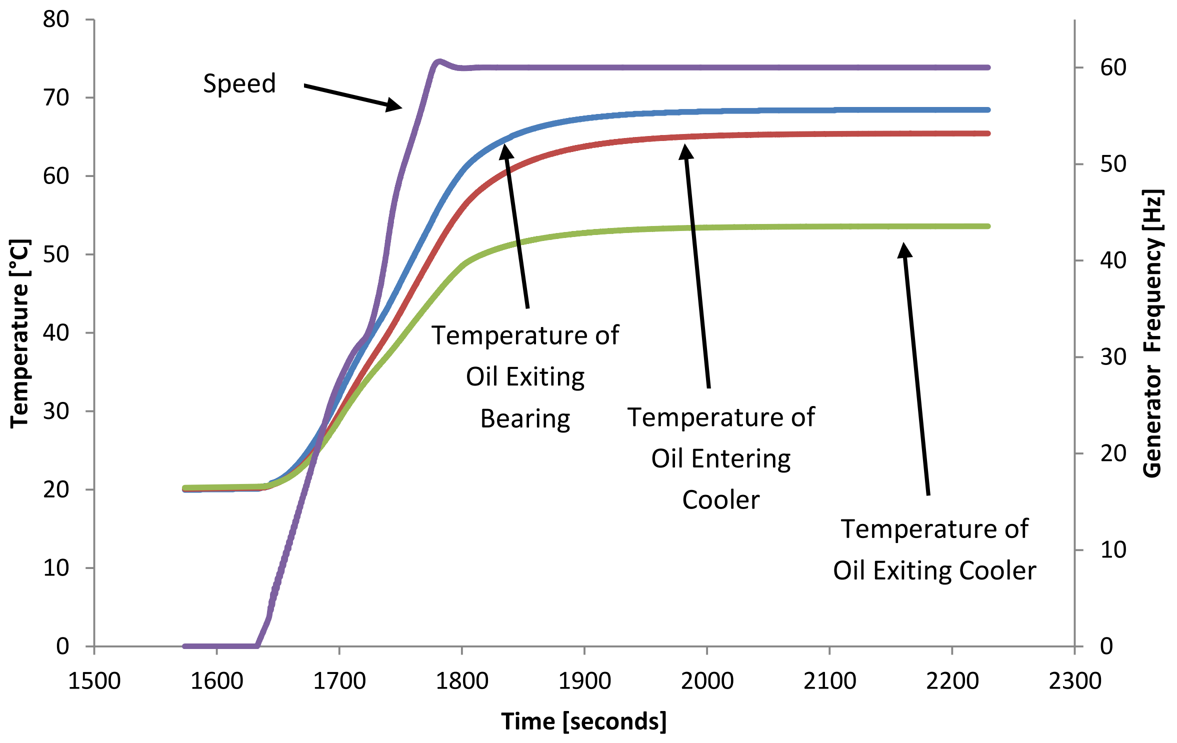


Figure 42. Generator Lube Oil Temperatures – Startup to Steady State

4.3 Simulink®

Issues Encountered

After spending months of researching and developing individual pieces of Simulink® models to debug and test separately; an issue arose because the size of the code generated by Simulink coder™ became so large and inefficient for coding purposes. Therefore, the external compiler was unable to compile the Simulink® code. Though the workstation used to compile had 24 GB of RAM, the compiler was running on a 32-bit process and was running out of memory to store the model/C-code and resulting executable file in RAM. This is due to the fact that the size of the ship electric plant Simulink model grew from only 5,500 kB to 76,000 kB. Simulink Coder™ had to organize numerous virtual subsystems, thousands of function blocks, over 20,000 virtual wires and buses, and a few C-code S-functions into a single complete C-code file. When Simulink® systems get this large, Simulink Coder™ requires additional work in terms of configuration and architecture to create efficient C-code [45].

Another problem that arose in plant model development was the failure for the executable file of the model to run in real-time. That is 1 second in model time correlates to real measure of 1 second, or 1:1 ratio. This was mainly due to two reasons: the size of the model (even with reference models) and the number of continuous integrators and transfer functions. These continuous integrators and transfer functions create continuous states which require additional computational power over discrete states.

It was determined that nobody at Woodward had experience handling or compiling such large model, so the decision made was to consult with The Mathworks, Inc Consulting Services to assist in altering the Simulink® model (while keep the same functionality) to compile. The consultant that worked with the project was very helpful. With his help, the issues were solved and performance of the executable model running in real-time also increased. The concepts and techniques developed through the consultation will be discussed further.

A common technique to overcome the issue of large Simulink® models is to create reference (non-virtual subsystem) models. This is especially useful when code is reused in multiple instances. The idea behind reference models is that instead of copying and pasting a virtual subsystem, a single reference model is used in multiple instances. If a change is made in the reference model, it automatically applies to all the instances. The drawback of reference models over virtual subsystem is that reference models require greater level of interface specification to enable reuse. If it is planned for in the beginning of model development, then it is relatively straightforward to follow the reference model rules. If it is not planned for and the model is relatively large, MATLAB® scripts need to be developed to automatically convert the subsystems, function blocks, lines, and S-functions to comply with reference model requirements. Scripts were also written to convert model from continuous states to discrete states.

The technique and requirements for reference models along with conversion of model from continuous to discrete states will be discussed in detail in later sections.

Reference Models

Reference models improve the Simulink® model compiled code efficiency and reduce the time Simulink Coder™ takes to compile the model. The efficiency is gained in reduction of model size due to the reference models generating reusable functions. They are non-virtual subsystems. The MATLAB® documentation defines it best as, "The primary difference is that non-virtual subsystems provide the ability to control when the contents of the subsystem are evaluated. Non-virtual subsystems are executed as a single unit (atomic execution) by the Simulink® engine. A subsystem is virtual unless the block is conditionally executed…" [46]. As discussed earlier, one model only needs to be changed to alter multiple instances referencing the same reference model. A reference model is a model in which a subsystem is placed outside of the large model in a separate model with the inputs and outputs on the top level. Simulink Coder™ splits the complete top model into its reference model portion and un-referenced/virtual subsystems. Then it compiles (outside of MATLAB® in an external compiler such as Microsoft Visual Studio®) the reference models first creating an independent C-code file. Then it links those compiled models within the larger model and finally compiles everything else not within the reference models into a separate C-code file. All of C-code files are then instrumented into an executable .exe file. This process is illustrated in Figure 43.

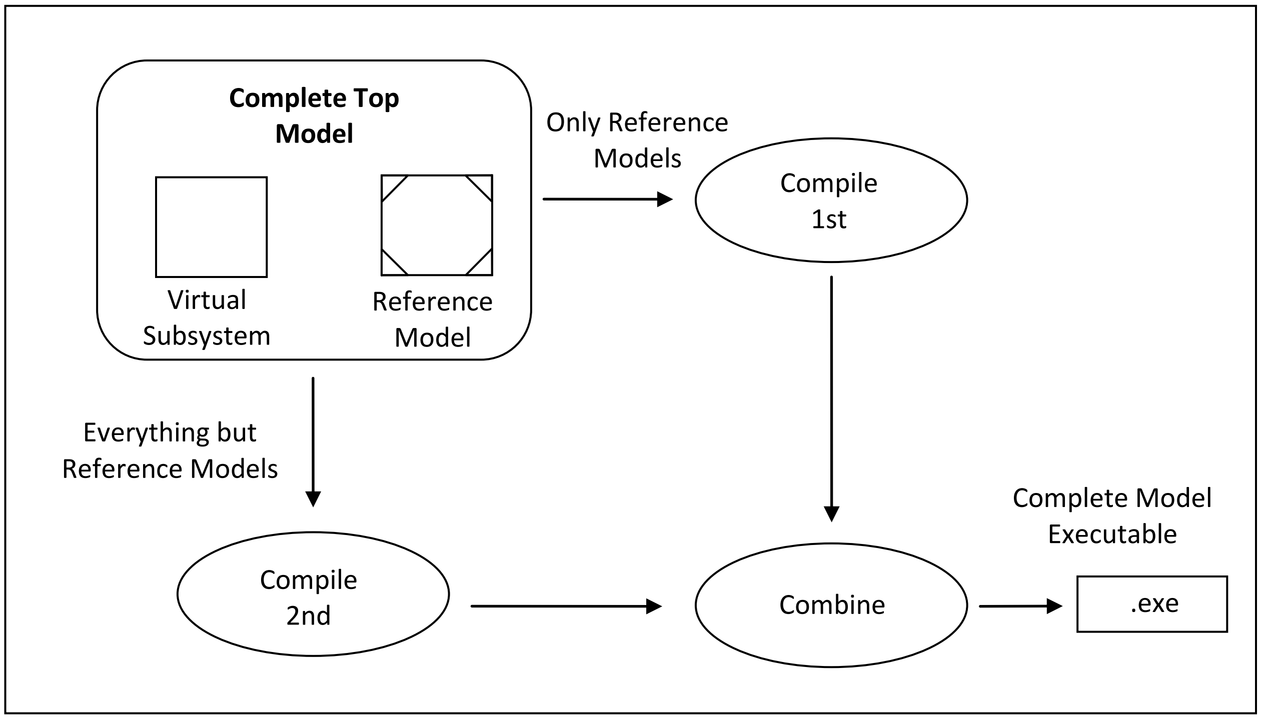


Figure 43. Example of Compile Flow of Simulink Coder™

Figure 44 shows the visual difference between virtual subsystem and reference model. Each have the same exact code underneath the subsystem (same function blocks, lines, subsystems) but since Simulink Coder™ compiles the reference model separately from the rest of the model, it has a easier time compiling the rest of the model.

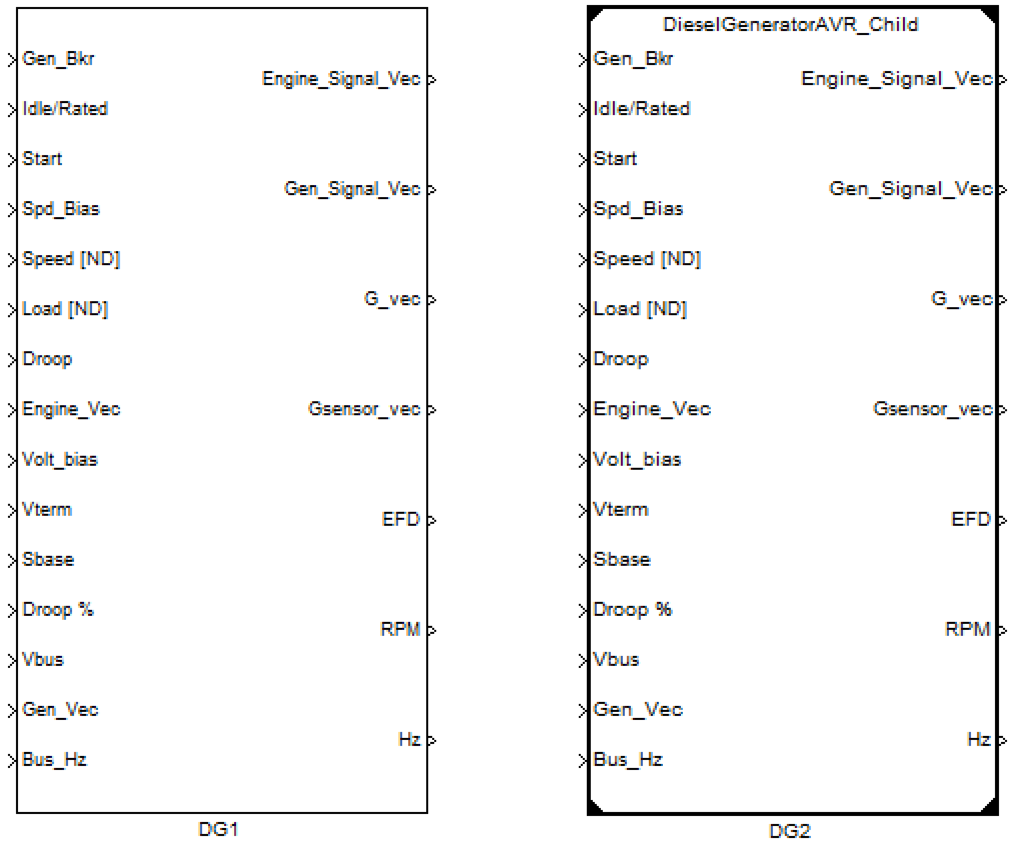


Figure 44. Virtual Subsystem [Left] vs. Reference Model [Right] Subsystem in Simulink®

A reference model can be simulated in 4 modes: Normal, Accelerator, Software-in-the-Loop (SIL), and Processor-in-the-Loop (PIL). Normal mode executes the reference submodel interpretively in Simulink®. Advantages of Normal mode are it works with more Simulink® and Stateflow® tools as well as supports more S-functions that the Accelerator does, though Normal mode executes slower than the Accelerator mode does. Accelerator mode executes the reference submodel by creating a MEX-file (or simulation target), then running MEX-file. SIL and PIL are useful in production code and simulating hardware. SIL executes production code on host platform while PIL executes production code on a target processor connected to the host computer [46]. To achieve the goal of real-time simulation, the reference model mode chosen for the project was Accelerator mode.

The requirement/limitations of reference models (Accelerator) are as follows from latest version of MATLAB documentation [46]:

- **Signal Propagation** - “The signal name must explicitly appear on any signal line connected to an Outport block of a referenced model. A signal connected to an unlabeled line of an Outport block of a referenced model cannot propagate out of the Model block to the parent model.”
- **Bus Usage** - “A bus that propagates between a parent model and a referenced model must be nonvirtual. Use the same bus object to specify the properties of the bus in both the parent and the referenced model. Define the bus object in the MATLAB workspace.”
- **S-functions** - “You cannot use the Simulink Coder S-function target in a referenced model in Accelerator mode.” And “A referenced model in Accelerator mode cannot use S-functions generated by the Simulink Coder software.”
- **Global Goto’s** – “global means that From and Goto blocks using the same tag can be anywhere in the model except in locations that span nonvirtual subsystem boundaries.”

The summary behind the documentation is that bus objects must be used to define named buses of combined signals or a single signal which are to be used outside the reference model (i.e. signals like temperatures and pressures within the engine MVM which are simulated to match inputs sent to the power management control and HMI are combined into a bus to pass signals from within the reference model to NetSim™ interface blocks/S-function in the top model). Also, S-functions within the reference models (Accelerator) have a Target Language Compiler (TLC) code interface for Simulink Coder™ (though note code still written in C-code while calling infrastructure is defined in TLC). The steam property functions for the steam turbine were written in C-code and needed to be converted to TLC for use in Accelerated mode reference models. MATLAB® provides a process to generate a TLC wrapper that can be realized through the use of a Legacy Code Tool.

Bus Objects

Bus objects are MATLAB® workspace variables that define a signal bus: it has labeled inputs and outputs along with information such as input signal dimension, complexity (real or complex), and maximum and/or minimum value. Signal names for the inputs and outputs must follow a certain structure to be valid in C programming language [45]. There must be no spaces or invalid characters/symbols such as "\" or "#" in the signal name, but underscore "_" is valid. The algorithm to convert normal signal buses to signal buses that reference bus objects first starts with converting the signal names into valid names (since most likely valid signal names were not planned for in initial development but if all signal names are valid first step can be ignored). Then, it involves updating the signal names within the Simulink® model to the valid names. Finally, storing those names and creating structures for bus objects where the bus objects can be saved from the workspace. The algorithm is visualized in Figure 45 and the MATLAB® script (developed by the Mathworks consultant and modified by the author for implementation in this project) is shown in the Appendix D. A common problem that one encounters using the script is that the lines entering the signal bus must be labeled; this problem usually occurs when combining multiple signal buses into one signal bus where it is not required to label the signal for Simulink® to understand and propagate signal names to the combined signal bus. For script to work, **ALL** lines entering and exiting every signal bus targeted to be converted into bus objects **MUST** be labeled with valid names. The “update signal with valid names” portion of the script does not take into account lines not labeled (due to propagation) in the Simulink® model and must be done manually.

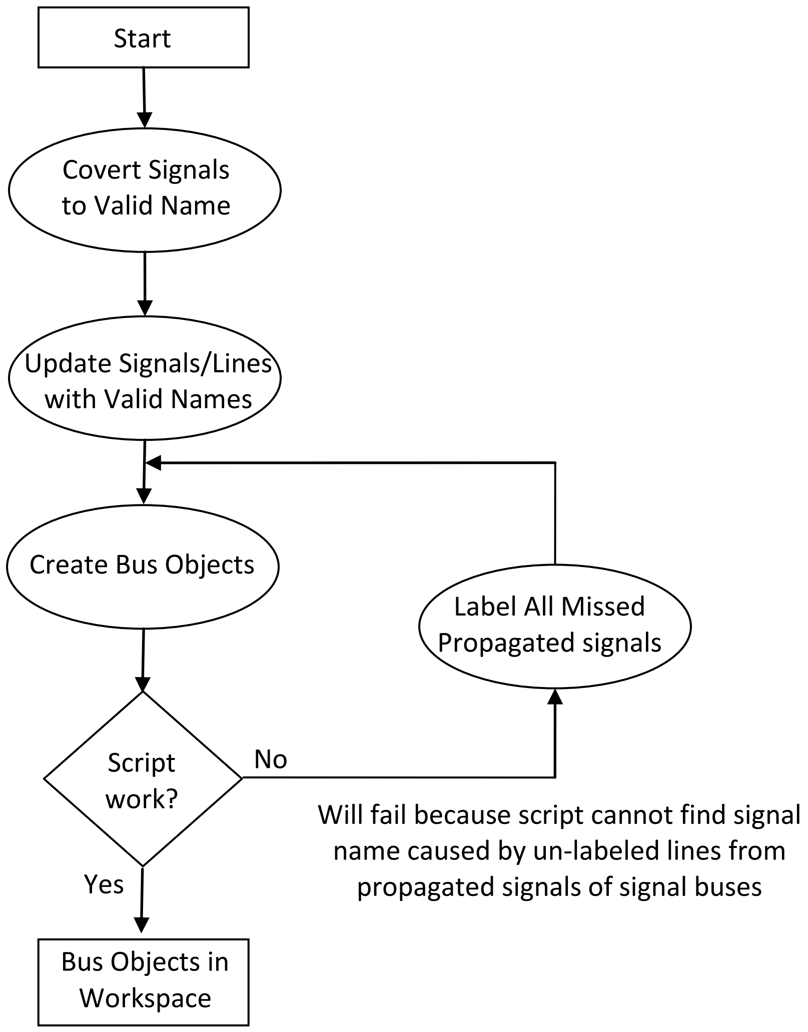
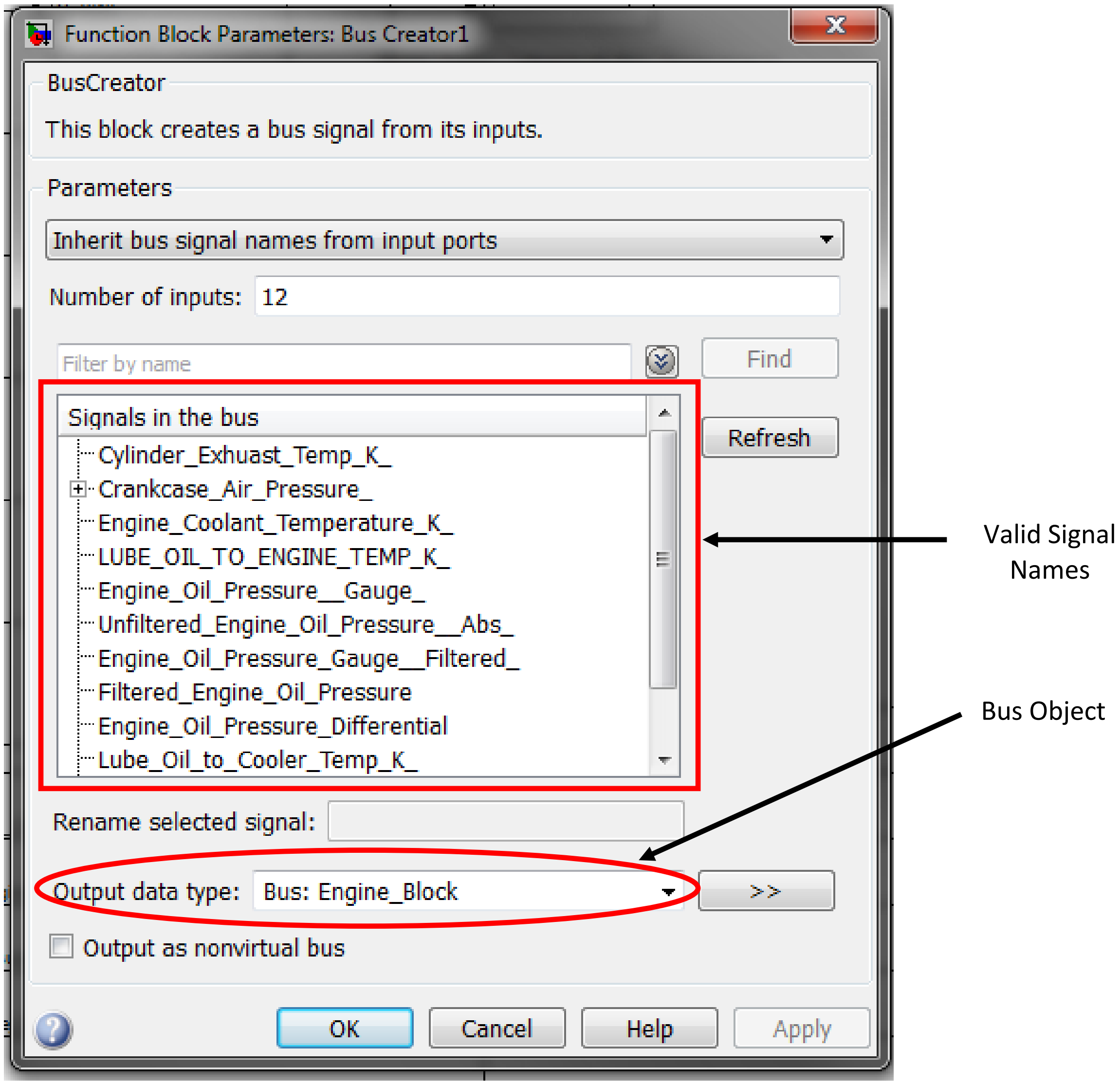

Start
Covert Signals to Valid Name
Update Signals/Lines with Valid Names
Create Bus Objects
Label All Missed Propagated signals
Script work?
No
Will fail because script cannot find signal name caused by un-labeled lines from propagated signals of signal buses
Yes
Bus Objects in Workspace

Figure 45. Flow of Bus Object Conversion Script

After the bus objects are created, the signal bus will reference an “Output data type” which is the bus object in the MATLAB®. An example of a signal bus (in the bus creator dialog box) with it referencing a bus object with valid signal names can be seen in Figure 46.


Function Block Parameters: Bus Creator1

BusCreator

This block creates a bus signal from its inputs.

Parameters

Inherit bus signal names from input ports

Number of inputs: 12

Signals in the bus

Cylinder_Exhuast_Temp_K_
Crankcase_Air_Pressure_
Engine_Coolant_Temperature_K_
LUBE_OIL_TO_ENGINE_TEMP_K_
Engine_Oil_Pressure__Gauge_
Unfiltered_Engine_Oil_Pressure__Abs_
Engine_Oil_Pressure_Gauge__Filtered_
Filtered_Engine_Oil_Pressure
Engine_Oil_Pressure_Differential
Lube_Oil_to_Cooler_Temp_K_

Rename selected signal:

Output data type: Bus: Engine_Block

Output as nonvirtual bus

Figure 46. Bus Creator Dialog Box for Signal Bus

Legacy and TLC-Code

As discussed earlier, there is a strict limitation of C-code based S-functions within reference models. Therefore, it was necessary to convert C-code based S-functions into C-code with a TLC wrapper. MATLAB® has a build in function to integrate a TLC wrapper to legacy code (C and C++ code) called Legacy Code Tool. This was a very useful tool since it avoids having to manually create a TLC wrapper, which would have been time-consuming. MATLAB® documentation describes the Legacy Code Tool as, “The legacy_code function creates a

MATLAB structure for registering the specification for existing C or C++ code and the S-function being generated. In addition, the function can generate, compile and link, and create a masked block for the specified S-function" [46]. The setup script for converting C-code based S-functions to TLC-code involves; first initializing the function data structure, defining the files (source and header), defining the interface of the S-function, and defining the simulation version. Then the tool generates a TLC based S-function source file and compiles it. Finally the tool creates a masked S-function block. The MATLAB® script that sets up and executes Legacy Code Tool functions can be referred to in the Appendix D. In the case of this project, Figure 47 illustrates the comparison between the unmasked original S-function and the masked TLC S-function for the look up of the steam properties.

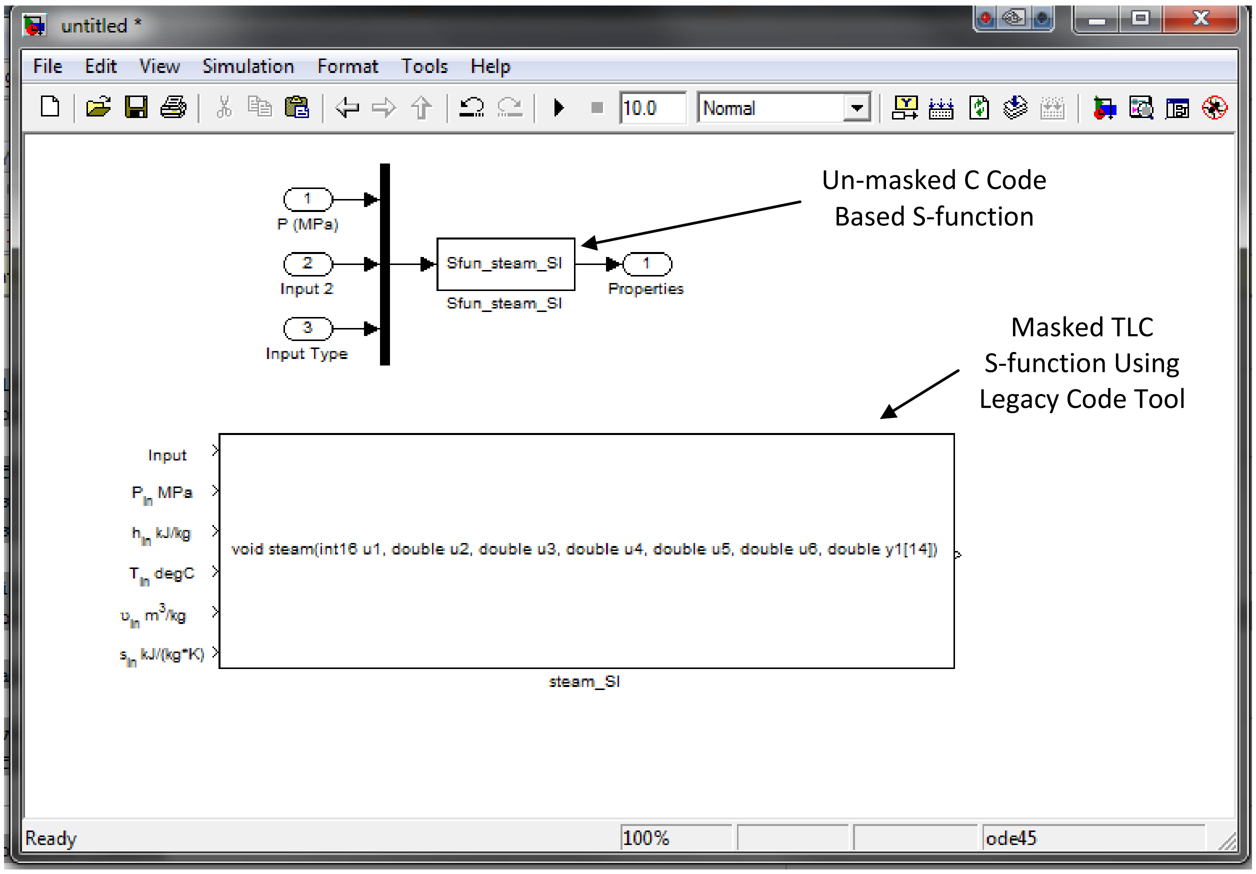

untitled *
File Edit View Simulation Format Tools Help
10.0
Normal
Un-masked C Code Based S-function
1 P (MPa)
2 Input 2
3 Input Type
Sfun_steam_SI
Sfun_steam_SI
1 Properties
Masked TLC S-function Using Legacy Code Tool
Input
P_{in} MPa
h_{in} kJ/kg
T_{in} degC
υ_{in} m^3/kg
s_{in} kJ/(kg*K)
void steam(int16 u1, double u2, double u3, double u4, double u5, double u6, double y1[14])
steam_SI
Ready
100%
ode45

Figure 47. Comparison of S-functions

Continuous to Discrete States

Continuous states are created in a Simulink® model when continuous function blocks are used. The solver chosen for the model will recognize these continuous states and adjust the output of the continuous blocks to show a continuous line even though the solver executes a solution in discrete fixed or variable time step. Similarly in all function blocks, Simulink® will calculate the discrete version of the block and then process the output to be continuous if required by a continuous function block. Since discrete blocks avoid the extra process, the solution is inherently solved much faster [45]. The differences in the solution are negligible and can hardly be noticed in Figure 49, the output of the scope of the simple model in Figure 48. In Figure 50, differences between continuous can be noticed if zoomed in close enough to notice the time step increment (time step in this case 0.01 sec).

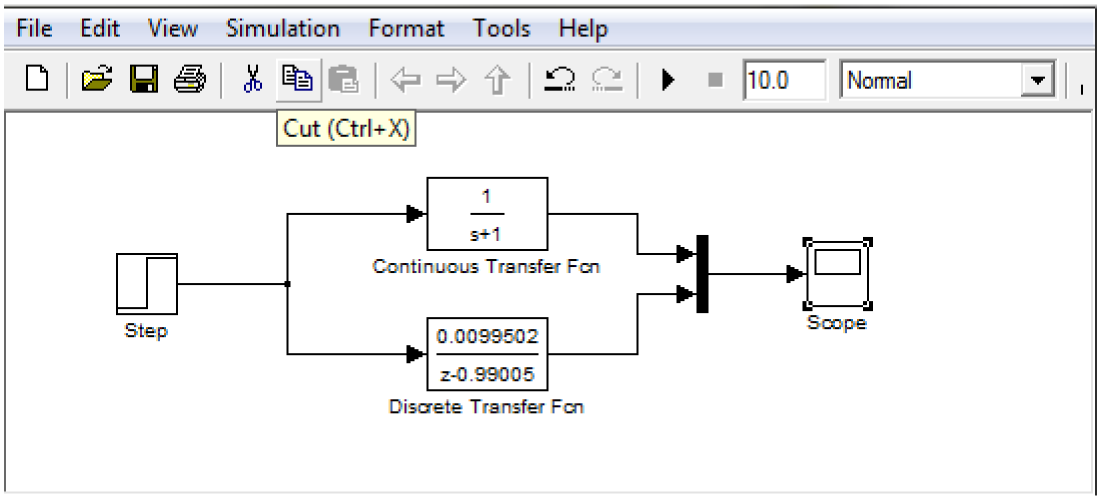


Figure 48. Continuous vs. Discrete Example Model

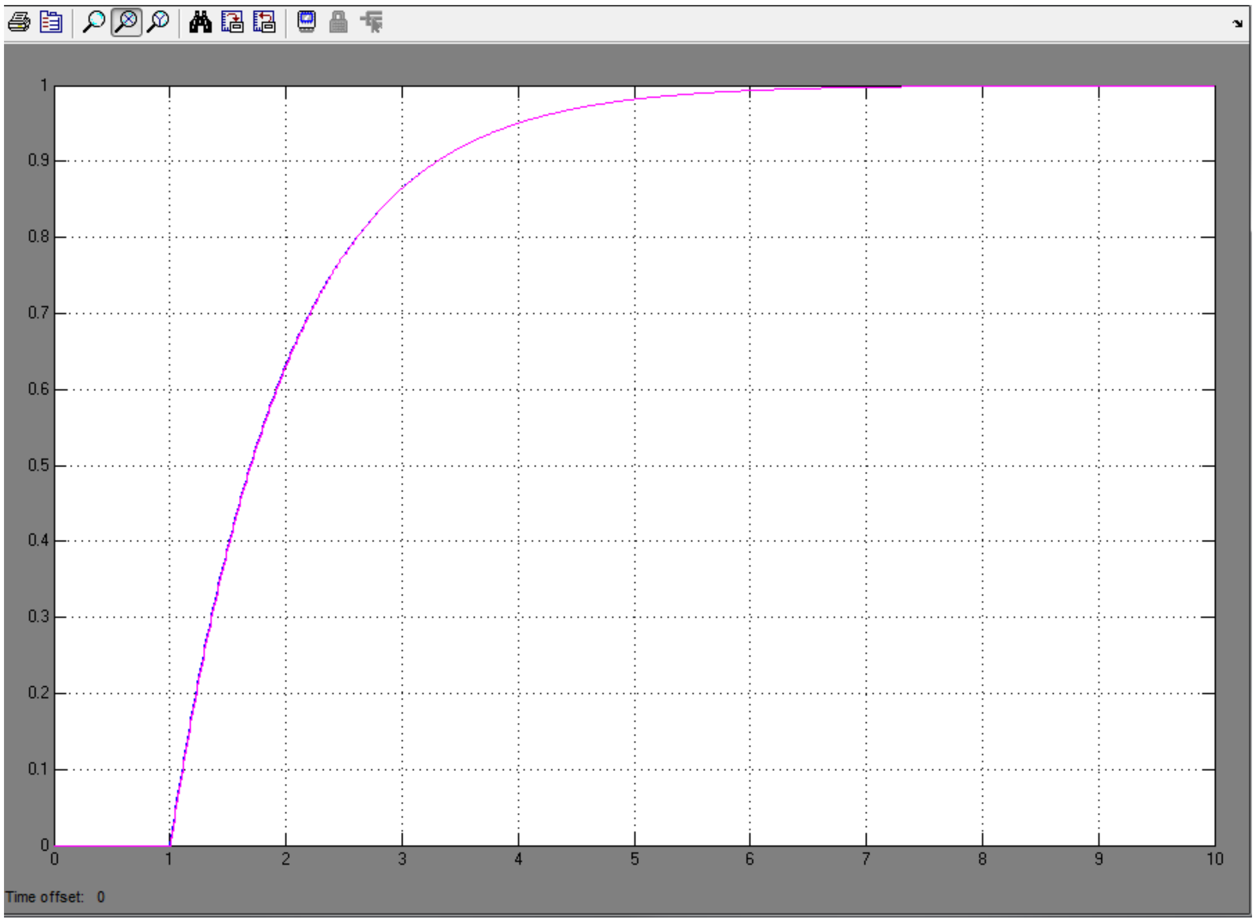


Figure 49. Output of Scope in Example Model

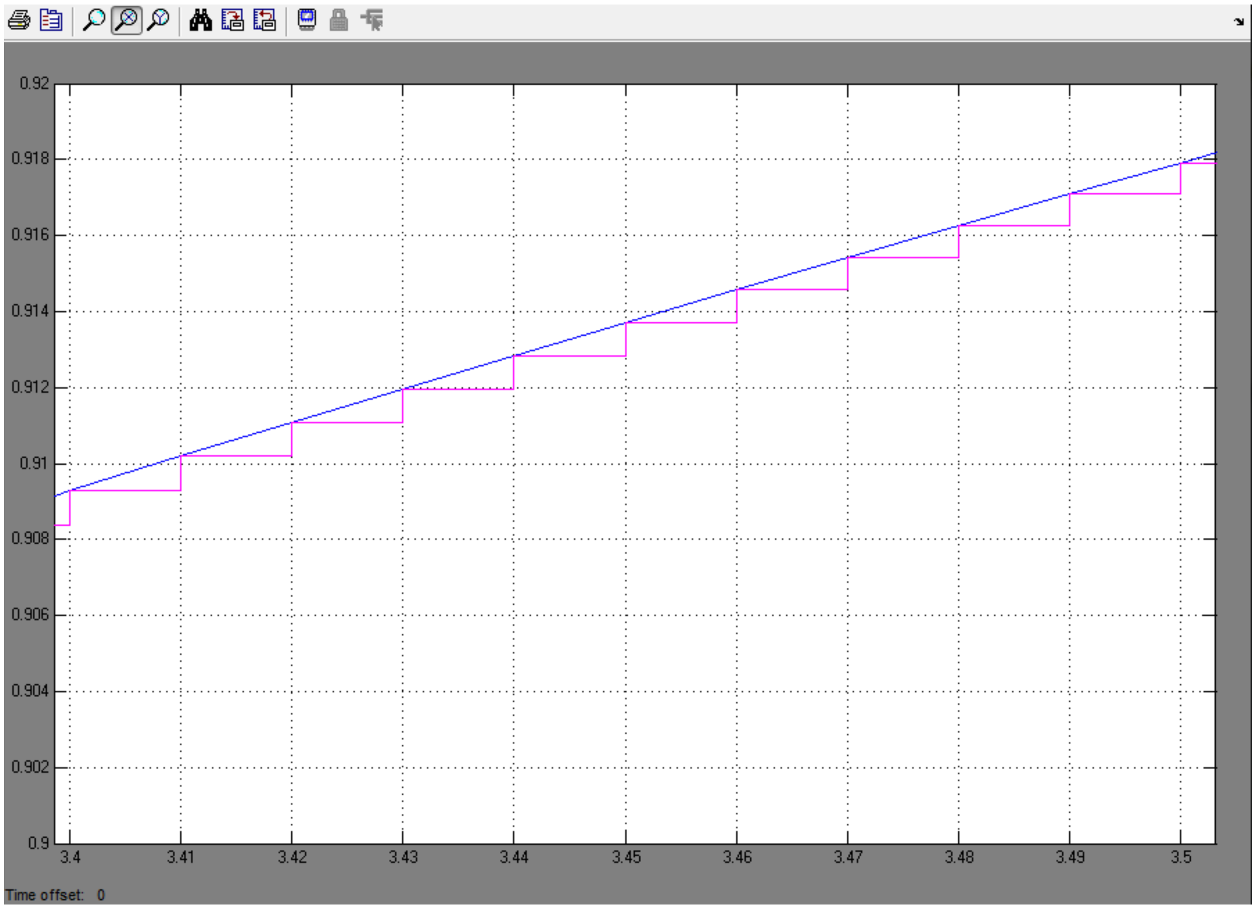


Figure 50. Zoomed in Output of Example Model (blue continuous and purple is discrete)

To take advantage of the faster solution times of discrete function blocks and fully discrete state model, MATLAB® scripts were written to convert all continuous function blocks within the plant model to its discrete equivalent. The two main function blocks that constitute for all of the continuous states within the plant model were the continuous integrators and continuous transfer functions. Although state space function blocks are also continuous, they were not used in the plant model and are not converted by the developed script. The script involved first finding all of the continuous function blocks within plant model. Then the script stored the function blocks location, name, and position. Afterwards, the script stored parameters specific to the continuous integrator and the 3 common types of continuous transfer functions (no initial conditions, only initial output, and both initial input and output). Then using the stored parameters, each function block was then converted by MATLAB® to discrete transfer function form. For the transfer functions, they were converted from transfer function to state space form using “tf2ss()” function. Then MATLAB® is able to convert from continuous state space domain to discrete state space domain using “c2d()” function. The numerator and denominator of the discrete transfer function were then stored. Finally, discrete integrators and transfer functions could replace the continuous ones (in each respective location and position) by using “delete_block()” and “add_block()” functions and automatically inputting relevant stored parameters into the dialog boxes. The resulting MATLAB® Script can be referred to in the Appendix D.

Results

The results from converting to reference models and almost entirely discrete states (sans electric generator model) within the plant model were dramatic. The Simulink® plant model went from not being able to even compile into an executable file to being able to compile as well as the executable model ran at 4 times real-time. The goal of the executable model running in real-time was achieved with some margin. This clearly demonstrates the efficiency gains achieved within the Simulink® code.

The efficiency gains can be separated into two categories: due to reference models and due to discrete states. The reference models created boundaries for Simulink Coder™ to handle compiling and Accelerator mode of the reference models increased the speed at which the solver can simulate the reference model. The combined effect only has a slight effect on the simulation speed. The majority of the gains came from converting the model states from continuous to discrete. The computation power needed for continuous states is much greater due to the added step of converting the discrete state solution to a continuous state. Multiply this effect by thousands of function block states, the performance of the solver and the executable model begins to suffer. Therefore, unnecessary calculations are avoided when a model has all or mostly all of functions blocks set as discrete.

In some cases it is not advantageous to convert from continuous states to discrete states. It was noted in testing that conversion from continuous to discrete state within electric generator subsystem adversely affected solver and the stability of the solution. The electric generators with discrete states were observed to exhibit erratic and unstable behavior when trying to load share with another (or multiple) generator(s). Eventually the instability in the solution would erroneously cause one or multiple generators to over-current or reverse power, causing the control system protect and bring off-line the generator in question. The solution was to simply convert the discrete integrators within the electric model back to continuous. This in total affected about 20 integrators (while the rest of the plant model stayed discrete) and it was determined to negligibly affect the ability for the plant model to run in real-time

In reference to the discussion earlier about the Simulink® model size, the reference models accounted for about 60,450 kB while the rest of virtual subsystems accounted for about 11,840 kB. The reference model figure comes from the size of the single reference model (about 5,000-7,000 kB), then multiplied by the times used in the plant model. And finally each type multiplied by times used is summed to arrive at the value of 60,450 kB. Bringing down the size of the reference model to thousands of kB versus 70,000 kB for the entire model allowed for Simulink Coder™ to realistically handle the code Simulink® and compile into a C file. Instead of having Simulink Coder™ deal with organizing and compiling such an enormous model, it was able to now compile much smaller reference models and a smaller top model one at a time.

Chapter5: Conclusion

The purpose of the thesis is to disseminate simulation techniques/knowledge that encompasses multiple disciplines of engineering that were required to model an on-board electrical power system with prime movers, electrical generators, and a small electrical grid. More specifically, the thesis combines knowledge found in various textbook and research publications in the fields of mechanical engineering, electrical engineering, classical controls, and software engineering. The electrical power system model was developed along with training executive to serve the purpose of training new sailors how to operate the electric plant through the HMI. The initial inherited models were stated along with definitions of MVM and per unit notation (commonly used in synchronous AC electrical generator models). The development of a MVM engine model and modification of existing fuel to power transfer function was discussed along with implementation of auxiliary engine systems. A heat exchanger model was created and applied for various fluids/heat exchangers based on idea of calculations per node with pertinent heat transfer correlations and thermodynamic effects modeled. A steam turbine model was implemented to convert pressure drop and steam flow rate (based on control valve position) into output shaft power. A heat transfer model for heat flow within a TEWAC synchronous generator was constructed to simulate the general behavior of heat transfer from the interior surfaces, stator/stator core and field windings/rotor surfaces, to the circulating air against those surfaces. Also, a heat transfer model for the cooling of the air within the finned tube seawater heat exchanger was developed as well. At the system level, an electric grid plant model was also discussed and more detailed technique presented in Appendix C. While the electrical dynamics for synchronous AC generators are not discussed, general governing equations can be found in electrical power system literature to complete the power system plant model.

Application of the models presented (combined into a complete model) was described using MATLAB® Simulink®, NetSim™, GAP™, and accompanying HMI. Model results for the diesel engine and steam turbine were to shown to illustrate the validity of the models. Additionally, the details on what issues were encountered and details on how issues were overcome within the project due to Simulink Coder™ and Simulink® limitations were presented. Reference models, bus objects, and the differences between continuous and discrete states were explained in detail and all scripts used in the process are in recreated in Appendix D.

Recommendations

- Based on the limited amount of physical and performance data for the diesel engine, steam turbine, or TEWAC cooling; more research is need in determining the accuracy of the models when compared to performance versus transient load data, while steady state performance for all models is sufficient.
- If, at the beginning of a Simulink® project using Simulink Coder™, the size of the Simulink® model is known to become large (greater than approximately 40,000 kB), reference models should be used segment the code to allow Simulink Coder™ to handle compiling code properly.

References

[1] Woodward, I. E. S. a. S., "Personal Communication."
[2] Kao, M. H., and Moskwa, J. J., 1995, "Turbocharged Diesel-Engine Modeling For Nonlinear Engine Controls and State Estimation," Journal of Dynamic Systems Measurement and Control-Transactions of the ASME, 117(1), pp. 20-30.
[3] Guzzella, L., 2010, "Introduction to modeling and control of internal combustion engine systems," C. H. Onder, ed., Springer, Berlin, p. 354.
[4] Flory, M., and Hiltner, J., "Engine Control System Development Using Rapid Prototyping Hardware and Software.," Proc. 25th CIMAC World Congress on Combustion Engine.
[5] 1987, "IEEE Recommended Practice: Definitions of Basic Per-Unit Quantities for Ac Rotating Machines," IEEE Std 86-1987, p. 0_1.
[6] Doerry, N. H., Clayton, D. H., and Ieee, 2005, "Shipboard electrical power quality of service," 2005 IEEE Electric Ship Technologies Symposium, pp. 274-279.
[7] Ericsen, T., and Ieee, "The Ship Power Electronic Revolution: Issues and Answers," Proc. 55th Annual Petroleum and Chemical Industry Conference, Ieee, pp. 221-231.
[8] Hansen, J. F., Adnanes, A. K., and Fossen, T. I., 2001, "Mathematical modelling of diesel-electric propulsion systems for marine vessels," Mathematical and Computer Modelling of Dynamical Systems, 7(3), pp. 323-355.
[9] Sun, J. B., and Guo, C., 2010, PC-Based Modeling and Simulation of Large Marine Propulsion Plant, Springer-Verlag Berlin, Berlin.
[10] Adams, J., and Borowski, D., 2008, "Machinery Systems: Redefining Power System Automation," SEAFRAME Carderock Division Publication, NAVSEA Naval Surface Warfare Center Carderock Division.
[11] Sauer, P. W., and Pai, M. A., 1998, Power system dynamics and stability, Prentice Hall, Upper Saddle River, N.J.
[12] Kundur, P., Balu, N. J., and Lauby, M. G., 1994, Power system stability and control, McGraw-Hill, New York.
[13] 2006, "IEEE Recommended Practice for Excitation System Models for Power System Stability Studies," IEEE Std 421.5-2005 (Revision of IEEE Std 421.5-1992), pp. 0_1-85.
[14] Anderson, P. M., and Fouad, A. A., "Power System Control and Stability (Second Edition)," Wiley-IEEE Press.
[15] Heywood, J. B., 1988, Internal combustion engine fundamentals, McGraw-Hill, New York.
[16] White, F. M., 2008, Fluid mechanics, McGraw-Hill, New York.
[17] Yeaple, F. D., 1990, Fluid power design handbook, M. Dekker, New York.
[18] Honeywell, G. b., 2012, "Turbocharger Guide: Turbocharger, Intercoolers, Upgrades, Accessories, and Tutorials."
[19] 42 U.S.C 7041, e. s., 2012, "Title 40: Protection of Enviroment," Part 50-National Primary and Secondary Ambient Air Quality Standards, U. S. E. P. Agency, ed.
[20] Incropera, F. P., 2007, Fundamentals of heat and mass transfer, John Wiley, Hoboken, NJ.

[21] Bejan, A., and Kraus, A. D., 2003, Heat transfer handbook, J. Wiley, New York.
[22] Rohsenow, W. M., Hartnett, J. P., and Cho, Y. I., 1998, Handbook of heat transfer, McGraw-Hill, New York.
[23] Woodyard, D., "Pounder's Marine Diesel Engines and Gas Turbines (9th Edition)," Elsevier.
[24] 2000, Automotive handbook, Robert Bosch GmbH: Distributed by SAE, Society of Automotive Engineers, Stuttgart.
[25] Cortona, E., Onder, C. H., and Guzzella, L., 2002, "Engine thermomanagement with electrical components for fuel consumption reduction," International Journal of Engine Research, 3(3), pp. 157-170.
[26] Wagner, J., Paradis, I., Marotta, E., and Dawson, D., 2002, "Enhanced automotive engine cooling systems - A mechatronics approach," International Journal of Vehicle Design, 28(Compendex), pp. 214-240.
[27] Finol, C. A., and Robinson, K., 2006, "Thermal modelling of modern engines: a review of empirical correlations to estimate the in-cylinder heat transfer coefficient," Proceedings of the Institution of Mechanical Engineers Part D-Journal of Automobile Engineering, 220(D12), pp. 1765-1781.
[28] Taylor, C. F., and Toong, T. Y., "Heat transfer in internal-combustion engines," Proc. ASME Meeting, Aug 11-15 1957, American Society of Mechanical Engineers (ASME), p. 21.
[29] Inc., C., 2006, "Application and Installation Guide-Cooling Systems."
[30] 1973, "MW Response of Fossil Fueled Steam Units," IEEE Transactions on Power Apparatus and Systems, PA92(2), pp. 455-463.
[31] Demello, F. P., 1991, "Dynamic-Models for Fossil Fueled Steam Units in Power-System Studies," IEEE Transactions on Power Systems, 6(2), pp. 753-761.
[32] Dharmalingam, S., Sivakumar, L., and Kumar, K. K. A., 2010, "Simplified approximations for the accumulation value and time constant for an evaporator system in drum-type boilers," Proceedings of the Institution of Mechanical Engineers Part a-Journal of Power and Energy, 224(A1), pp. 59-67.
[33] Fang, F., Le, W., and Ieee, 2009, "Pressure Equilibrium Control for Boiler-turbine Units," 2009 Ieee International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 143-147.
[34] Fang, F., and Wei, L., 2011, "Backstepping-based nonlinear adaptive control for coal-fired utility boiler-turbine units," Applied Energy, 88(3), pp. 814-824.
[35] Li, S. Y., Liu, H. B., Cai, W. J., Soh, Y. C., and Xie, L. H., 2005, "A new coordinated control strategy for boiler-turbine system of coal-fired power plant," Ieee Transactions on Control Systems Technology, 13(6), pp. 943-954.
[36] Liu, H. B., Li, S. Y., Chai, T. Y., and Aac, "Intelligent control of power-plant main steam pressure and power output," Proc. Annual American Control Conference (ACC 2003), Ieee, pp. 2857-2862.
[37] Naghizadeh, R. A., Vahidi, B., and Tavakoli, M. R. B., 2011, "Estimating the Parameters of Dynamic Model of Drum Type Boilers Using Heat Balance Data as an Educational Procedure," Ieee Transactions on Power Systems, 26(2), pp. 775-782.

[38] Neuman, P., Sulc, B., and Dlouhy, T., 2000, "Non-linear model of coal fired steam boiler applied to engineering simulator," Power Plants and Power Systems Control 2000, pp. 37-45.
[39] Jachens, W. B., "Steam Turbines -Their Construction, Selection and Operation," Proc. The South African Sugar Technologist's Association, pp. 113-131.
[40] Dixon, S. L., and Hall, C. A., 2010, Fluid mechanics and thermodynamics of turbomachinery, Butterworth-Heinemann/Elsevier, Burlington, MA.
[41] Dabbousi, R., Balfaqih, H., Anundsson, Y., and Savinovic, D., "A comparison of totally enclosed motor coolers commonly used in the COG industry," Proc. Petroleum and Chemical Industry Conference Europe - Electrical and Instrumentation Applications, 2008. PCIC Europe 2008. 5th, pp. 1-5.
[42] Nonaka, S., Murata, K., Yamamoto, M., and Takeda, Y., 1979, "Experimental Study on Cooling of Rotor in a Salient 4-Pole Synchronous Machine," Power Apparatus and Systems, IEEE Transactions on, PAS-98(1), pp. 310-317.
[43] Wen, J., Khonsari, M. M., and Hua, D. Y., 2011, "Three-Dimensional Heat Transfer Analysis of Pin-Bushing System With Oscillatory Motion: Theory and Experiment," Journal of Tribology, 133(1), pp. 011101-011110.
[44] Stevenson, W. D., 1982, Elements of power system analysis, McGraw-Hill, New York.
[45] The Mathworks, I. C. S., "Personal Communication."
[46] The Mathworks, I., 1984-2012, "R2012a Documentation-Simulink."
[47] Sharqawy, M. H., Lienhard, J. H., and Zubair, S. M., 2010, "Thermophysical properties of seawater: a review of existing correlations and data," Desalination and Water Treatment, 16(1-3), pp. 354-380.
[48] Fernandes, C. S., Dias, R. P., and Maia, J. M., 2008, "New Plates for Different Types of Plate Heat Exchangers," Recent Patents on Mechanical Engineering 2008, pp. 198-205.

Appendix A – Mapping Turbocharger

Most turbocharger performance maps distributed by manufactures will use corrected pressure ratio and turbocharger rotational speed to determine the compressor's efficiency and corrected mass flow. The surface plots (or level sets) of efficiency create efficiency islands where the mass flow and pressure ratio are plotted for selected speeds. Using Eqs. (106)-(86), the state of the compressor can be determined. This relies on how accurately this 2 dimensional map can be reproduced in simulation software. This is done by linearly interpolating values on the map. Artificial lines labeled "R-Lines" are created to assist with linearly interpolating between speed and pressure ratio to obtain compressor mass flow. The turbocharger adapted for this project was a Garret GT6041 turbocharger shown in Figure 51 [18]. The map used for interpolation with R-lines is shown in Figure 52. The map had to be slightly modified since the original map had two solutions for the mass flow with the same rotational speed. The resulting map is shown in Figure 53.

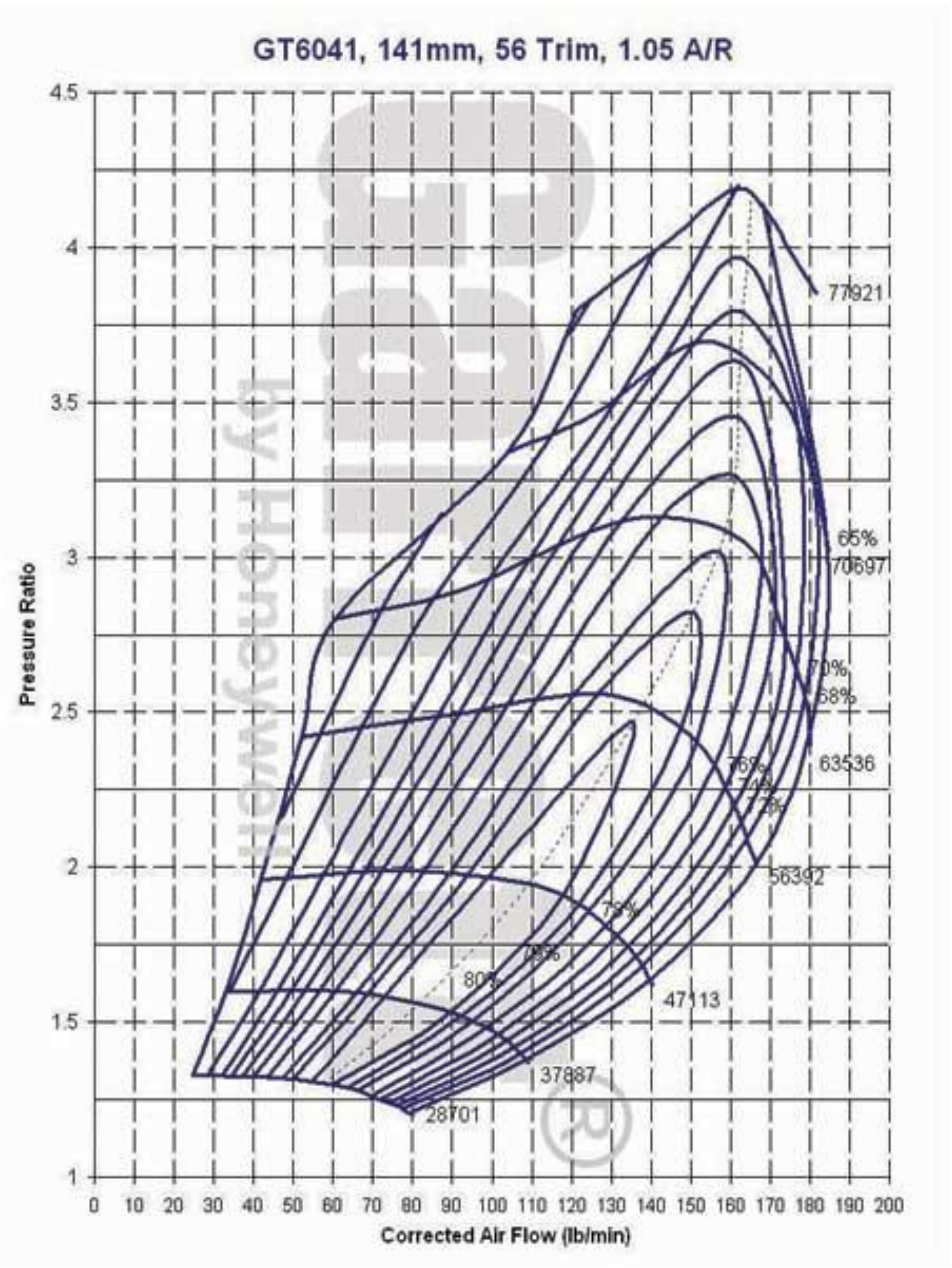


Figure 51. Garret GT6041 Compressor Map [18]

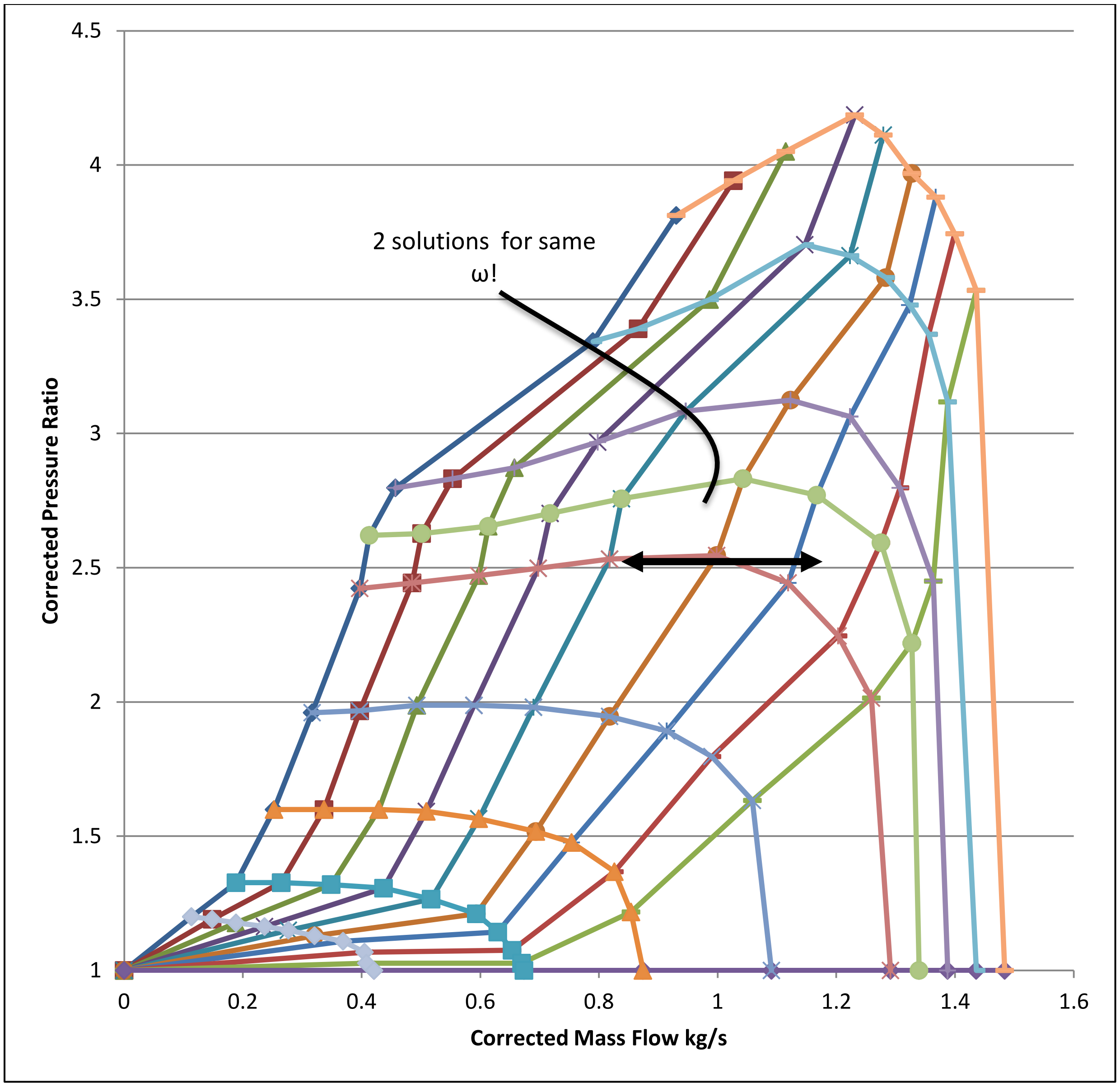


Figure 52. Garrett GT6041 Interpolation Points: R-lines and Speed

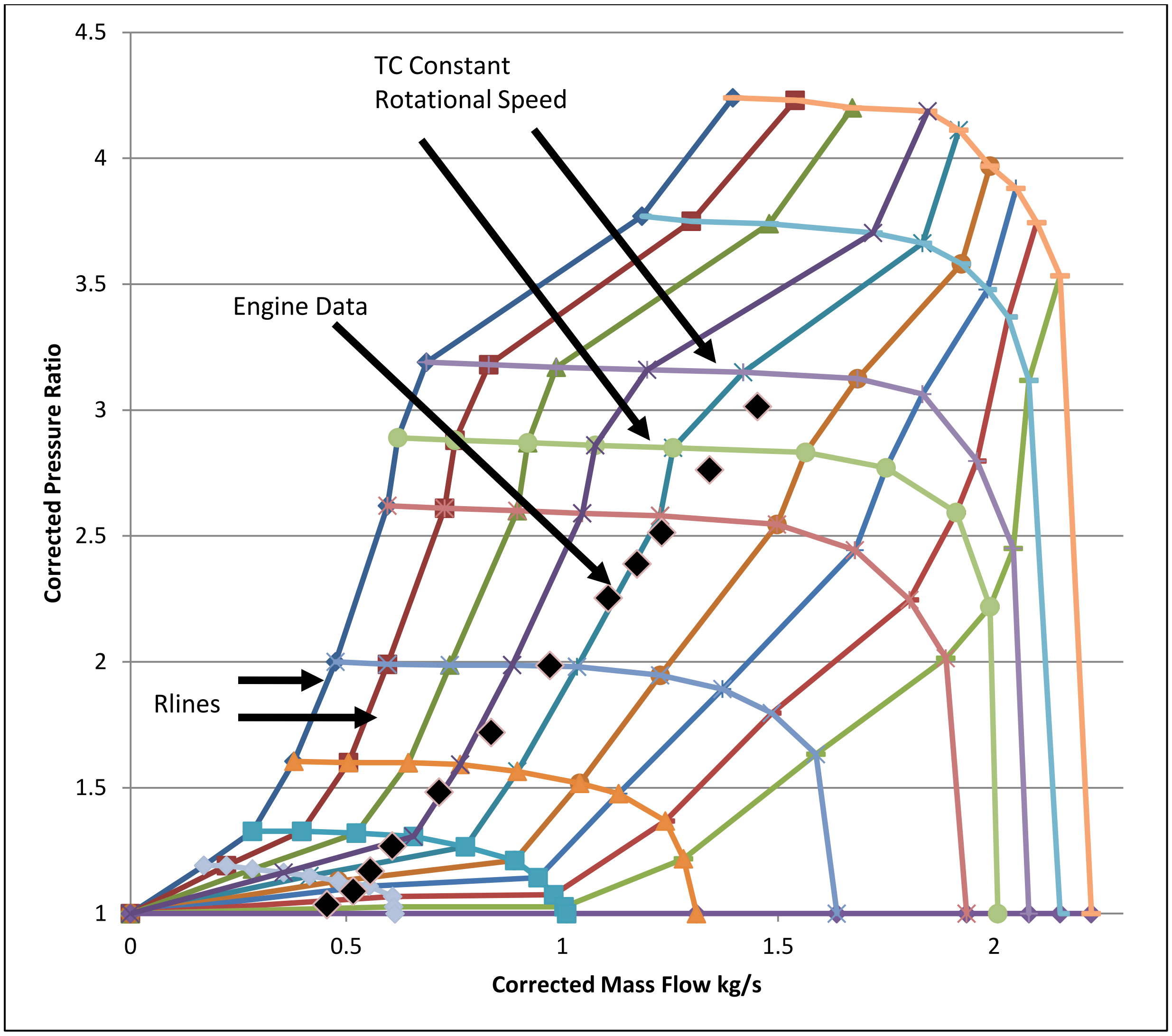


Figure 53. Modified Compressor Map Used in Diesel Model

The level sets of efficiencies and speeds in Figure 51 also had to be modified to account for slightly larger mass flow, but technique to map efficiencies applies to figure as well. First the peak efficiency line needs to be defined and is roughly defined as the point where the efficiency islands become a maximum and minimum on the pressure ratio axis. Then, along this line; values of mass flow, speed, and efficiency are logged to create a "Peak Flow vs. Speed" and Peak Efficiency vs. Speed" tables. Finally, a percentage of deviation of flow versus efficiency needs to be defined where the difference in actual mass flow is compared to the flow along the speed line that goes through the maximum efficiency point. This deviation is assumed to hold true for all speeds on the map to simplify the calculation. Figure 54 shows the modified turbocharger map and Figure 55-Figure 57 are the efficiency correlations just explained.

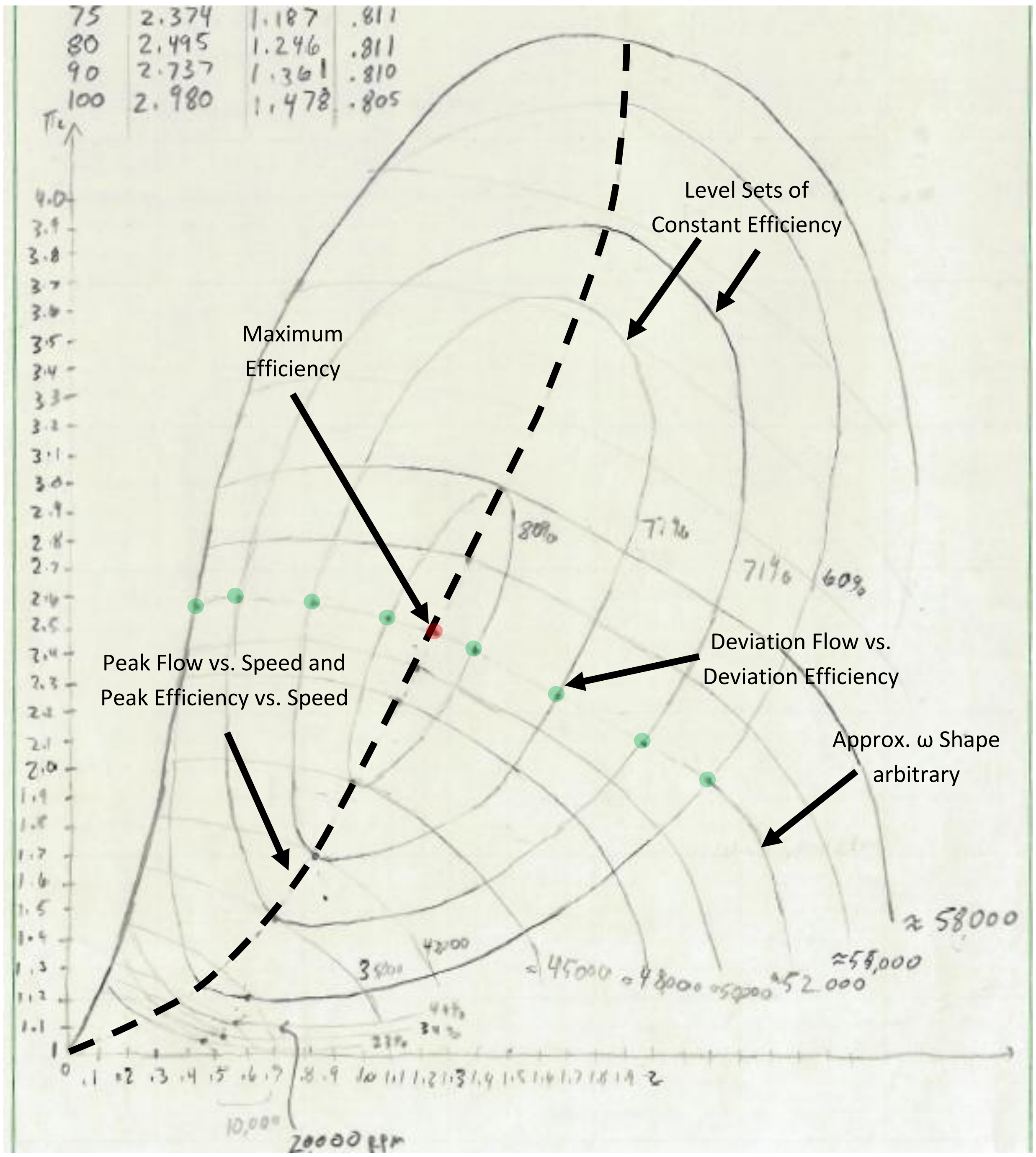


Figure 54. New Islands of Compressor Efficiency and Rotational Speed

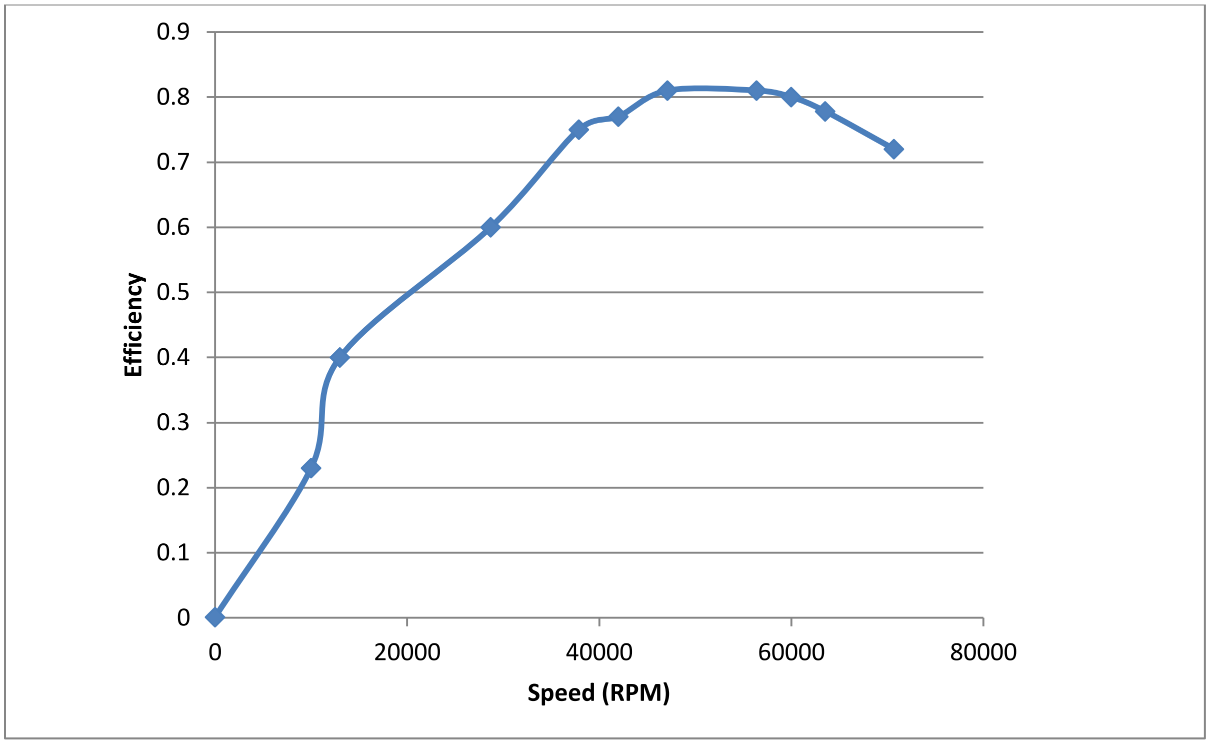


Figure 55. Compressor Peak Efficiency versus Rotational Speed

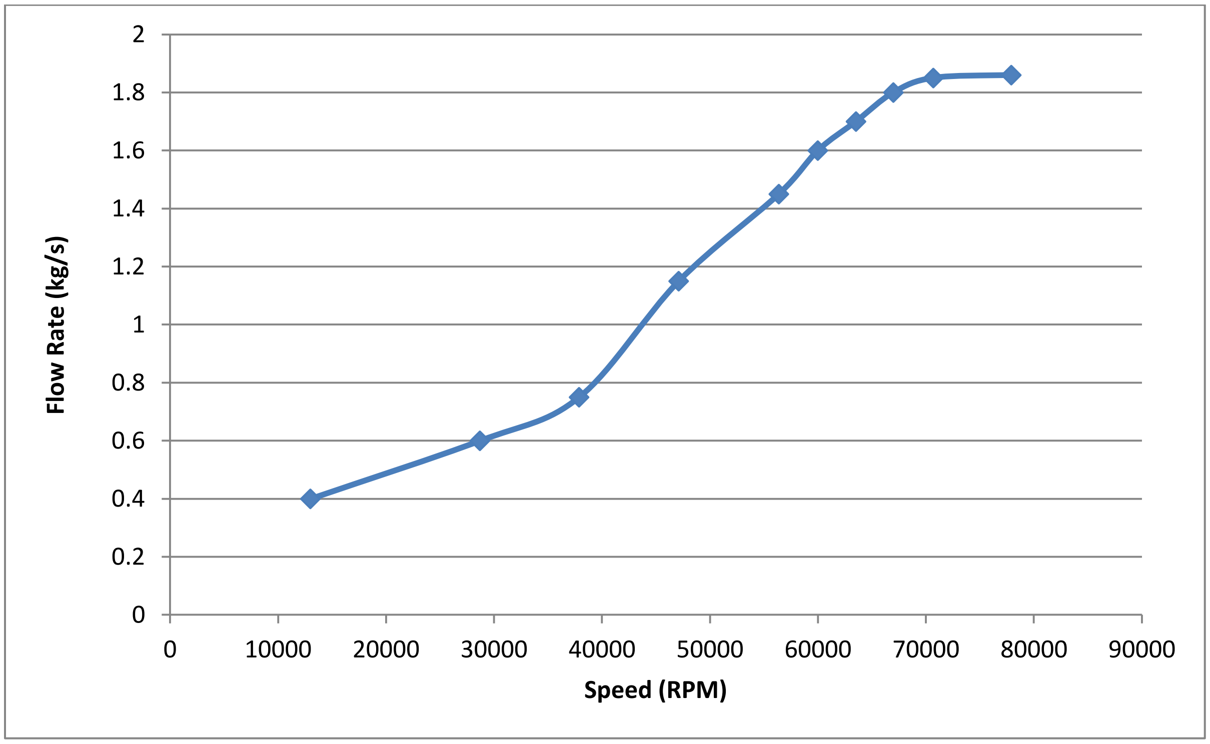


Figure 56. Compressor Peak Mass Flow versus Rotational Speed

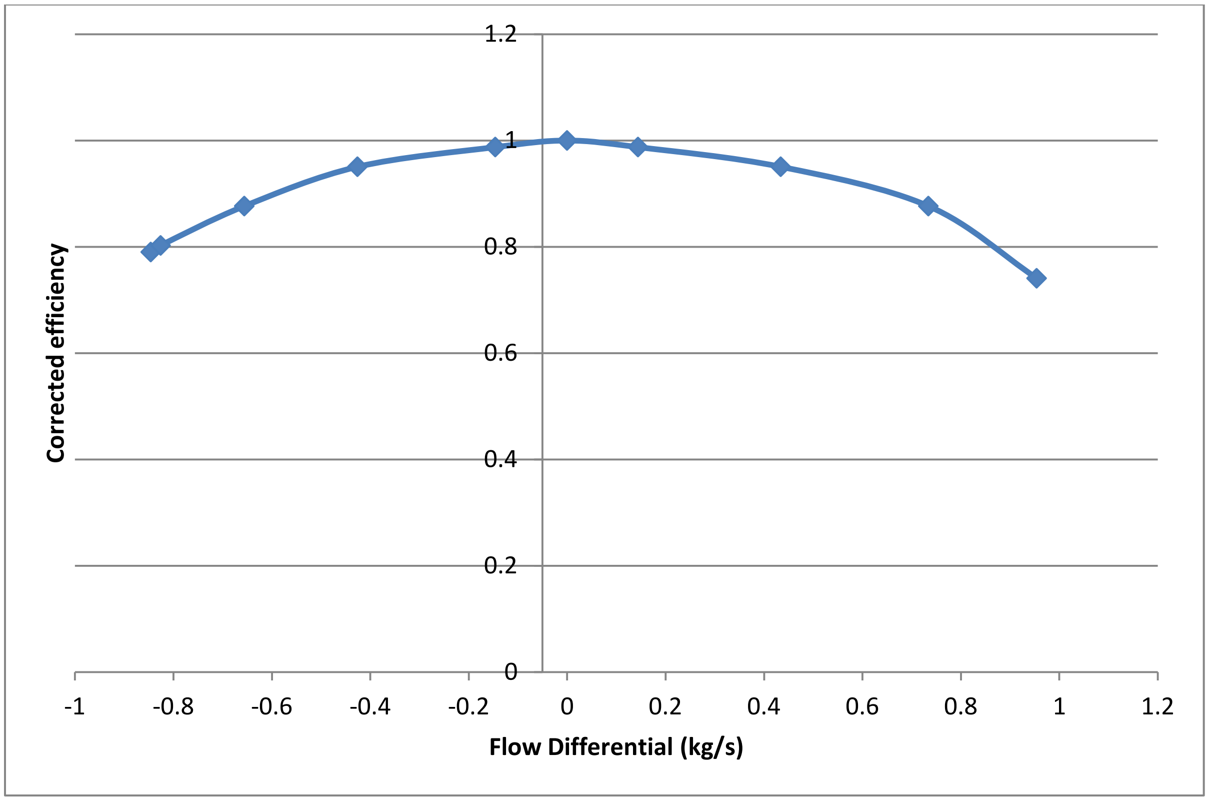


Figure 57. Corrected Compressor Efficiency versus Flow Differential from Peak Flow

Appendix B – Property Tables

For all the fluid properties used in the engine mean value model as well as in the various heat exchangers modeled. The fluid dynamic properties for air, water, lubrication oil, and seawater are presented.

Table 6. Air at Atmospheric Pressure [20]

Temperature	Thermal Conductivity	Prandtl Number	Kinematic Viscosity	Heat Capacity @ Constant Pressure
K	W/(m*K)		m^2/s	J/(kg*K)
100	0.00934	0.786	0.00000200	1032
150	0.0138	0.758	0.00000443	1012
200	0.0181	0.737	0.00000759	1007
250	0.0223	0.72	0.00001144	1006
300	0.0263	0.707	0.00001589	1007
350	0.03	0.7	0.00002092	1009
400	0.0338	0.69	0.00002641	1014
450	0.0373	0.686	0.00003239	1021
500	0.0407	0.684	0.00003879	1030
550	0.0439	0.683	0.00004557	1040
600	0.0469	0.685	0.00005269	1051
650	0.0497	0.69	0.00006021	1063
700	0.0524	0.695	0.00006810	1075
750	0.0549	0.702	0.00007637	1087
800	0.0573	0.709	0.00008493	1099

Table 7. Water at Atmospheric Pressure [20]

Temperature	Kinematic Viscosity	Thermal Conductivity	Prandtl Number	Density	Dynamic Viscosity	Specific Heat @ Constant Pressure
K	m^2/s	W/(m*K)		kg/m^3	N*s/(m^2)	J/(kg*K)
273.15	0.00000175	0.569	12.99	1000	0.00175	4217
275	0.000001652	0.574	12.22	1000	0.001652	4211
280	0.000001422	0.582	10.26	1000	0.001422	4198
285	0.000001225	0.59	8.81	1000	0.001225	4189
290	1.08108E-06	0.598	7.56	999.001	0.00108	4184
295	9.60918E-07	0.606	6.62	998.004	0.000959	4181

300	8.57565E-07	0.613	5.83	997.009	0.000855	4179
305	7.72845E-07	0.62	5.2	995.0249	0.000769	4178
310	6.99865E-07	0.628	4.62	993.0487	0.000695	4178
315	6.36679E-07	0.634	4.16	991.0803	0.000631	4179
320	5.83347E-07	0.64	3.77	989.1197	0.000577	4180
325	5.34864E-07	0.645	3.42	987.1668	0.000528	4182
330	4.96824E-07	0.65	3.15	984.252	0.000489	4184
335	4.61154E-07	0.656	2.88	982.3183	0.000453	4186
340	4.2882E-07	0.66	2.66	979.4319	0.00042	4188
345	3.98336E-07	0.668	2.45	976.5625	0.000389	4191
350	3.74855E-07	0.668	2.29	973.7098	0.000365	4195
355	3.8419E-07	0.671	2.14	970.8738	0.000373	4199
360	3.38118E-07	0.674	2.02	967.118	0.000327	4203
365	3.17628E-07	0.677	1.91	963.3911	0.000306	4209
370	3.00849E-07	0.679	1.8	960.6148	0.000289	4214
373.15	2.91276E-07	0.68	1.76	957.8544	0.000279	4217
375	2.8633E-07	0.681	1.7	956.9378	0.000274	4220
380	2.7274E-07	0.683	1.61	953.2888	0.00026	4226
385	2.61144E-07	0.685	1.53	949.6676	0.000248	4232
390	2.50746E-07	0.686	1.47	945.1796	0.000237	4239
400	2.31539E-07	0.688	1.34	937.2071	0.000217	4256

Table 8. Oil at Atmospheric Pressure [20]

Temperature	Thermal Conductivity	Prandtl Number	Kinematic Viscosity	Heat Capacity @ Constant Pressure
K	W/(m*K)		m^2/s	J/(kg*K)
273	0.147	47000	0.00428	1796
280	0.144	27500	0.00243	1827
290	0.145	12900	0.00112	1868
300	0.145	6400	0.00055	1909
310	0.145	3400	0.000288	1951
320	0.143	1965	0.000161	1993
330	0.141	1205	0.0000966	2035
340	0.139	793	0.0000617	2076
350	0.138	546	0.0000417	2118
360	0.138	395	0.0000297	2161
370	0.137	300	0.000055	2206

380	0.136	233	0.0000169	2255
390	0.135	187	0.0000133	2294
400	0.134	152	0.0000106	2337
410	0.133	125	0.00000852	2381
420	0.133	103	0.00000694	2427
430	0.132	88	0.00000583	2471

For the following: Seawater at atmospheric pressure [47]

Table 9. Seawater Density kg/m^3

Salinity, g/kg												
0	10	20	30	40	50	60	70	80	90	100	110	Temp C
999.8	1007.9	1016	1024	1032	1040	1048	1056.1	1064.1	1072.1	1080.1	1088.1	0
999.7	1007.4	1015.2	1023	1030.9	1038.7	1046.6	1054.4	1062.2	1070.1	1077.9	1085.7	10
998.2	1005.7	1013.4	1021.1	1028.8	1036.5	1044.1	1051.8	1059.5	1067.2	1074.9	1082.6	20
995.7	1003.1	1010.7	1018.2	1025.8	1033.4	1040.9	1048.5	1056.1	1063.6	1071.2	1078.7	30
992.2	999.7	1007.1	1014.6	1022.1	1029.5	1037	1044.5	1052	1059.4	1066.9	1074.4	40
988	995.5	1002.9	1010.3	1017.7	1025.1	1032.5	1039.9	1047.3	1054.7	1062.1	1069.5	50
983.2	990.6	998	1005.3	1012.7	1020	1027.4	1034.7	1042.1	1049.5	1056.8	1064.2	60
977.8	985.1	992.5	999.8	1007.1	1014.5	1021.8	1029.1	1036.5	1043.8	1051.2	1058.5	70
971.8	979.1	986.5	993.8	1001.1	1008.5	1015.8	1023.1	1030.5	1037.8	1045.1	1052.5	80
965.3	972.6	980	987.3	994.7	1002	1009.4	1016.8	1024.1	1031.5	1038.8	1046.2	90
958.4	965.7	973.1	980.5	987.9	995.2	1002.6	1010	1017.4	1024.8	1032.2	1039.6	100
950.9	958.3	965.8	973.2	980.6	988.1	995.5	1003	1010.4	1017.8	1025.3	1032.7	110
943.1	950.6	958.1	965.6	973.1	980.6	988.1	995.6	1003.1	1010.6	1018.1	1025.6	120

Table 10. Seawater Thermal Conductivity W/(m*K)

Salinity g/kg													
0	10	20	30	40	50	60	70	80	90	100	110	120	Temp C
0.572	0.571	0.57	0.57	0.569	0.569	0.568	0.568	0.567	0.566	0.566	0.565	0.565	0
0.588	0.588	0.587	0.587	0.586	0.585	0.585	0.584	0.584	0.583	0.583	0.582	0.582	10
0.604	0.603	0.602	0.602	0.601	0.601	0.6	0.6	0.599	0.599	0.598	0.598	0.597	20
0.617	0.617	0.616	0.616	0.615	0.615	0.614	0.614	0.613	0.613	0.612	0.612	0.611	30
0.63	0.629	0.629	0.628	0.628	0.627	0.627	0.626	0.626	0.625	0.625	0.624	0.624	40
0.641	0.64	0.64	0.639	0.639	0.638	0.638	0.637	0.637	0.636	0.636	0.635	0.635	50
0.65	0.65	0.649	0.649	0.648	0.648	0.647	0.647	0.647	0.646	0.646	0.645	0.645	60

0.658	0.658	0.658	0.657	0.657	0.656	0.656	0.655	0.655	0.655	0.654	0.654	0.653	70
0.665	0.665	0.665	0.664	0.664	0.663	0.663	0.663	0.662	0.662	0.661	0.661	0.661	80
0.671	0.671	0.67	0.67	0.67	0.669	0.669	0.669	0.668	0.668	0.667	0.667	0.667	90
0.676	0.675	0.675	0.675	0.674	0.674	0.674	0.673	0.673	0.673	0.672	0.672	0.672	100
0.679	0.679	0.679	0.678	0.678	0.678	0.677	0.677	0.677	0.676	0.676	0.676	0.675	110
0.682	0.681	0.681	0.681	0.68	0.68	0.68	0.679	0.679	0.679	0.679	0.678	0.678	120

Table 11. Seawater Prandtl Number

Salinity g/kg													
0	10	20	30	40	50	60	70	80	90	100	110	120	Temp C
13.18	13.21	13.25	13.31	13.4	13.5	13.63	13.78	13.94	14.13	14.34	14.56	14.81	0
9.32	9.36	9.41	9.48	9.56	9.65	9.76	9.87	10	10.14	10.3	10.46	10.64	10
6.95	7	7.06	7.12	7.19	7.27	7.36	7.46	7.56	7.67	7.79	7.92	8.05	20
5.4	5.45	5.51	5.57	5.63	5.7	5.78	5.86	5.94	6.03	6.13	6.23	6.33	30
4.34	4.38	4.43	4.49	4.54	4.6	4.67	4.74	4.81	4.88	4.96	5.04	5.13	40
3.57	3.61	3.66	3.71	3.76	3.81	3.87	3.93	3.99	4.05	4.12	4.18	4.25	50
3	3.04	3.08	3.12	3.17	3.22	3.27	3.32	3.37	3.42	3.48	3.54	3.6	60
2.57	2.6	2.64	2.68	2.72	2.76	2.81	2.85	2.9	2.94	2.99	3.04	3.09	70
2.23	2.27	2.3	2.33	2.37	2.41	2.45	2.49	2.53	2.57	2.61	2.66	2.7	80
1.97	2	2.03	2.06	2.09	2.13	2.16	2.2	2.23	2.27	2.31	2.35	2.39	90
1.76	1.78	1.81	1.84	1.87	1.9	1.93	1.96	1.99	2.03	2.06	2.1	2.13	100
1.59	1.61	1.63	1.66	1.69	1.71	1.74	1.77	1.8	1.83	1.86	1.89	1.93	110
1.45	1.47	1.49	1.51	1.54	1.56	1.59	1.61	1.64	1.67	1.7	1.73	1.7	120

Table 12. Seawater Kinematic Viscosity x 10^7 m^2/s

Salinity g/kg													
0	10	20	30	40	50	60	70	80	90	100	110	120	Temp C
17.92	18.06	18.23	18.43	18.65	18.9	19.16	19.46	19.77	20.11	20.46	20.84	21.24	0
13.07	13.2	13.35	13.51	13.69	13.89	14.1	14.33	14.57	14.82	15.09	15.38	15.67	10
10.04	10.16	10.29	10.43	10.58	10.75	10.92	11.1	11.3	11.5	11.71	11.93	12.17	20
8.01	8.12	8.23	8.36	8.49	8.63	8.77	8.93	9.09	9.26	9.43	9.61	9.8	30
6.58	6.68	6.78	6.89	7	7.13	7.25	7.38	7.52	7.66	7.81	7.96	8.11	40
5.53	5.62	5.71	5.81	5.91	6.02	6.13	6.24	6.36	6.48	6.61	6.74	6.87	50
4.74	4.82	4.91	4.99	5.08	5.18	5.28	5.38	5.48	5.59	5.7	5.81	5.93	60
4.13	4.2	4.28	4.36	4.44	4.52	4.61	4.7	4.79	4.89	4.98	5.08	5.19	70
3.65	3.71	3.78	3.85	3.93	4	4.08	4.16	4.25	4.33	4.42	4.51	4.6	80

3.26	3.32	3.38	3.45	3.51	3.58	3.65	3.73	3.8	3.88	3.96	4.04	4.12	90
2.94	3	3.05	3.11	3.17	3.24	3.3	3.37	3.44	3.51	3.58	3.65	3.73	100
2.68	2.73	2.78	2.84	2.89	2.95	3.01	3.07	3.13	3.2	3.26	3.33	3.4	110
2.46	2.51	2.55	2.6	2.65	2.71	2.76	2.82	2.88	2.93	3	3.06	3.1	120

Table 13. Seawater Specific Heat at Constant Pressure J/(kg*K)

Salinity g/kg												
0	10	20	30	40	50	60	70	80	90	100	110	Temp
4206.8	4142.1	4079.9	4020.1	3962.7	3907.8	3855.3	3805.2	3757.6	3712.4	3669.7	3629.3	0
4196.7	4136.7	4078.8	4022.8	3968.9	3916.9	3867.1	3819.2	3773.3	3729.5	3687.7	3647.9	10
4189.1	4132.8	4078.2	4025.3	3974.1	3924.5	3876.6	3830.4	3785.9	3743	3701.8	3662.3	20
4183.9	4130.5	4078.5	4027.8	3978.6	3930.8	3884.4	3839.4	3795.8	3753.6	3712.7	3673.3	30
4181	4129.7	4079.6	4030.7	3982.9	3936.4	3891	3846.7	3803.7	3761.8	3721.1	3681.6	40
4180.6	4130.8	4081.9	4034.1	3987.3	3941.5	3896.6	3852.9	3810.1	3768.3	3727.5	3687.8	50
4182.7	4133.7	4085.5	4038.3	3992	3946.5	3902	3858.3	3815.5	3773.7	3732.7	3692.6	60
4187.1	4138.5	4090.6	4043.6	3997.3	3951.9	3907.4	3863.6	3820.6	3778.5	3737.2	3696.7	70
4194	4145.3	4097.3	4050.1	4003.7	3958.1	3913.3	3869.2	3825.9	3783.5	3741.7	3700.8	80
4203.4	4154.2	4105.9	4058.3	4011.5	3965.4	3920.2	3875.7	3832	3789.1	3746.9	3705.6	90
4215.2	4165.4	4116.4	4068.2	4020.9	3974.3	3928.5	3883.6	3839.4	3796	3753.5	3711.7	100
4229.4	4178.8	4129.1	4080.2	4032.2	3985.1	3938.7	3893.3	3848.6	3804.9	3761.9	3719.9	110
4246.1	4194.7	4144.2	4094.6	4045.9	3998.2	3951.3	3905.4	3860.3	3816.2	3773	3730.7	120

Appendix C – Example Bus Matrix

An example small electric power system grid (or island) seen in Figure 58, has 3 generators, 8 load centers, 8 buses, and 15 bus-ties with breaker control. It is assume for this case that all of the bus-ties have the same admittance, all the generators the same generator parameters and power output. The load centers real power output will be labeled Watt and reactance power consumption will be labeled VAR.

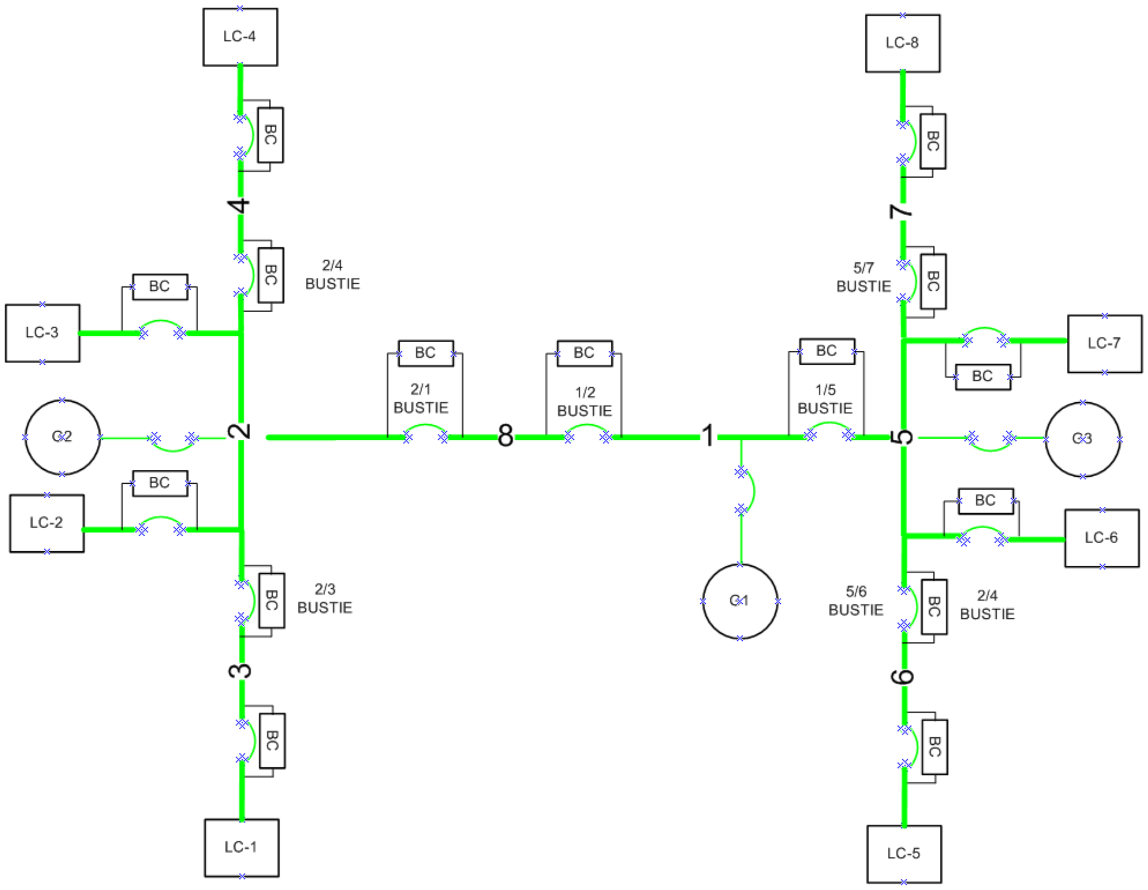


Figure 58. Sample Electric Power System Grid

If the electric power system grid model and generators are per-unit based, the first step is to convert all values into per-unit (p.u.). There are various forms of per-unit base systems [5] but it was chosen to use voltage E_{base}, apparent power S_{base} and frequency ω_{base} of the generator to be the base unit. If generator sizes connected to the small electric grid are different sizes, then the largest apparent power generator is used for the base values. For this case, E_{base} is 450 volts, S_{base} is 2,500,000 watts, and ω_{base} is 60 Hz. Other base values calculated of these are:

$$Z_{base} = \frac{{E_{base}}^2}{S_{base}}\ , Y_{base} = \frac{1}{Z_{base}}\ , I_{base} = \frac{S_{base}}{\sqrt{3}E_{base}} \tag{139}$$

For this example the various parameters become:

$$\begin{gathered} Z_{base} = \frac{{E_{base}}^2}{S_{base}} = \frac{450^2}{2{,}500{,}000} = 0.081\ ohms \\ Y_{base} = \frac{1}{Z_{base}} = \frac{1}{0.081} = 12.346\ siemens \\ I_{base} = \frac{S_{base}}{\sqrt{3}E_{base}} = \frac{2{,}500{,}000}{\sqrt{3}(450)} = 3{,}207.5\ amps \end{gathered} \tag{140}$$

Now the second step is to find all of the admittances within the system and convert to per-unit base. For the generators the admittance becomes (if generator's stator resistance (R_{stator}) and reactance (${X_d}'$) are 0.00104 ohms and 0.0194):

$$Y_g = \frac{1}{\left[\left(\frac{R_{stator}}{Z_{base}}\right) + j\left(\frac{{X_d}'}{Z_{base}}\right)\right]} = \frac{1}{\left[\left(\frac{0.00104}{0.081}\right) + j\left(\frac{0.194}{0.081}\right)\right]} = 0.225 - j4.1635\ p.u. \tag{141}$$

The admittance of the load centers found in a similar fashion is:

$$\begin{gathered} Y_{LC} = \frac{WattZ_{base}}{{E_{base}}^2} - j\frac{VARZ_{base}}{{E_{base}}^2} = \frac{Watt.081}{450^2} - j\frac{VAR.081}{450^2} \\ = (4\text{E}^{-7})Watt - j(4\text{E}^{-7})VAR \end{gathered} \tag{142}$$

Admittance of an open bus-tie breaker is zero or resistance is infinite:

$$Y_{CB-open} = 0 - j0\ p.u. \tag{143}$$

Admittance of a closed bus-tie breaker, though has little resistance has some reactance around .001:

$$Y_{CB-closed} = \frac{1}{\left[\left(\frac{R_{stator}}{Z_{base}}\right) + j\left(\frac{X_d{}'}{Z_{base}}\right)\right]} = \frac{1}{\left[\left(\frac{0}{0.081}\right) + j\left(\frac{0.001}{0.081}\right)\right]} = 0 - j81.3\ p.u. \tag{144}$$

Now that all the admittances are calculated, the admittance matrix now must be formed. The size of the matrix for this example for 8 buses is an 8x8 matrix and 64 admittances must be accounted for in the matrix. Starting with bus-ties that are physically connected, table below illustrates connections:

Table 14. Admittances of Physically Connected Bus-ties

From bus #	To bus #	R p.u.	X p.u.	Y p.u.	Symbol
1	8	0	.001/.081=.0123	0-j1/.0123= -j81.3	Y_{18}
1	5	0	.0123	-j81.3	Y_{15}
8	2	0	.0123	-j81.3	Y_{82}
2	4	0	.0123	-j81.3	Y_{24}
2	3	0	.0123	-j81.3	Y_{25}
5	6	0	.0123	-j81.3	Y_{56}
5	7	0	.0123	-j81.3	Y_{57}

Admittances of the inverse order are the same:

$$Y_{18} = Y_{81}, Y_{15} = Y_{51}, Y_{82} = Y_{28}, Y_{24} = Y_{24}, Y_{25} = Y_{52}, Y_{56} = Y_{65}, Y_{57} = Y_{75} \tag{145}$$

For all paths not connected the admittances are zero:

$$Y_{12} = Y_{21}, Y_{13} = Y_{31}, Y_{14} = Y_{14}, Y_{16} = Y_{61}, Y_{17} = Y_{71}$$

$$Y_{25} = Y_{52}, Y_{26} = Y_{62}, Y_{27} = Y_{72}$$

$$Y_{34} = Y_{43}, Y_{35} = Y_{53}, Y_{36} = Y_{36}, Y_{37} = Y_{73}, Y_{38} = Y_{83}$$

$$Y_{45} = Y_{54}, Y_{46} = Y_{64}, Y_{47} = Y_{74}, Y_{48} = Y_{84} \tag{146}$$

$$Y_{58} = Y_{85}$$

$$Y_{67} = Y_{76}, Y_{68} = Y_{86}$$

$$Y_{78} = Y_{87}$$

Next the sum of each bus's admittance (self admittance) is found, where load centers enter matrix:

$$
\begin{gathered}
Y_{11} = Y_{g1} + Y_{18} + Y_{15} \\
Y_{22} = Y_{g2} + Y_{LC2} + Y_{LC3} + Y_{23} + Y_{24} + Y_{28} \\
Y_{33} = Y_{23} + Y_{LC1} \\
Y_{44} = Y_{24} + Y_{LC4} \\
Y_{55} = Y_{g3} + Y_{LC6} + Y_{LC7} + Y_{56} + Y_{57} + Y_{15} \\
Y_{66} = Y_{56} + Y_{LC5} \\
Y_{77} = Y_{57} + Y_{LC8} \\
Y_{88} = Y_{28} + Y_{18}
\end{gathered}
\tag{147}
$$

Afterwards the current sources need to be identified and defined. In general a current source could be a generator, battery, breaker tied to the main utility grid, or anything sourcing AC current. In most cases, current is found by multiplying the sources electromotive force (emf) voltage by its admittance:

$$I = EY \tag{148}$$

For the case of this example only buses 1, 2, and 5 have generators sourcing current: I_{g1}, I_{g2}, I_{g3}. Current is calculated using per by (using subscript # to represent 1, 2, and 5):

$$I_{g\#} = E_{g\#} Y_{g\#} \tag{149}$$

where the absolute value of the emf $E_{g\#}$ of would stay relatively constant at 450 volts (with an excitation system) but its phasor (or voltage in the direct axis and quadrature axis using the common three phase d-q transformation) varies with load. Finally solving for the unknown voltages on all the buses using Eq. (93), the admittance matrix and current/voltages are as follows:

$$\begin{bmatrix} V_{11} \\ V_{22} \\ V_{33} \\ V_{44} \\ V_{55} \\ V_{66} \\ V_{77} \\ V_{88} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} & Y_{15} & Y_{16} & Y_{17} & Y_{18} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} & Y_{25} & Y_{26} & Y_{27} & Y_{28} \\ Y_{31} & Y_{32} & Y_{33} & Y_{34} & Y_{35} & Y_{36} & Y_{37} & Y_{38} \\ Y_{41} & Y_{42} & Y_{43} & Y_{44} & Y_{45} & Y_{46} & Y_{47} & Y_{48} \\ Y_{51} & Y_{52} & Y_{53} & Y_{54} & Y_{55} & Y_{56} & Y_{57} & Y_{58} \\ Y_{61} & Y_{62} & Y_{63} & Y_{64} & Y_{65} & Y_{66} & Y_{67} & Y_{68} \\ Y_{71} & Y_{72} & Y_{73} & Y_{74} & Y_{75} & Y_{76} & Y_{77} & Y_{78} \\ Y_{81} & Y_{82} & Y_{83} & Y_{84} & Y_{85} & Y_{86} & Y_{87} & Y_{88} \end{bmatrix}^{-1} \begin{bmatrix} I_{11} \\ I_{22} \\ I_{33} \\ I_{44} \\ I_{55} \\ I_{66} \\ I_{77} \\ I_{88} \end{bmatrix} \tag{150}$$

where red highlighted admittances are the “self admittances” in Eq. (131) and green admittances are in Table 14 and Eqs. (143) and (144). All the rest of the admittances are zero.

Appendix D – MATLAB® Scripts

Bus Object Script:

```
%% Example model converter
% Auth/Revision:  Michael Burke
%                 Copyright 2012, The MathWorks Consulting Group.
% Revised to suit application by Matthew Boley


%% Step 1: Define inputs
model='modelname'
%% Step 2: Get the line names
[oldNames,newNames,allLines] = GetSignalNamesAndReturnValidNames(model);

%% Step 3: Update the line Names
UpdateInvalidSignalNames(model,oldNames,newNames,allLines);

%% Step 4: Create the bus objects...
[busObjects] = GenerateBusObjectForBusCreator(model)

%% Step 5: Save the bus objects that where created....
```

Bus Object Supporting Functions:

1st Function

```
%% Example Function:  GetSignalNamesAndReturnValidNames
%
% For use on the Woodward project
%
% This script shows how to inspect a model and update "invalid" signal
% names.  A signal is considered "valid" if it is valid in the C programing
% language.
%
% Invalid characters in the names will be replaced with the underscore
% symbol "_"
% Signal names that start with number will have an "A" added to the front
% of the signal name.
%
% Inputs:
%   model --> the name of the model
%
% Outputs:
%   oldNames      --> The original signal names (filtered to just changed)
%   newNames      --> The new signal names (filtered to just changed)
%   allLines      --> The line handles to the lines that need to be changed
%   originalNames --> All the names
%
% Auth/Revision:  Michael Burke
%                 Copyright 2012, The MathWorks Consulting Group.
% Revised to suit application by Matthew Boley

function [oldNames,newNames,allLines] =
GetSignalNamesAndReturnValidNames(model)
```

```
%% Step 0: Define Valid Characters
validLow = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',...
            'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',...
            'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
validNum = {'0','1','2','3','4','5','6','7','8','9'};
validChar = [validLow, upper(validLow),'_',validNum];

%% Define invalid characters
invalidChar = {'/','\','[',']','{','}','!','%','@','#',...
               '^','&','*','(',')',',',';','?',' ','-'};
%% Step 1: find all signal names
    allLines = find_system(model, 'FindAll', 'on', 'type', 'line');
    oldNames = get_param(allLines,'Name');
    % For return use only
    originalNames = oldNames;
%% Step 2: filter out the propogated names and the "empty" names
notProp = [];
for inx = 1  : numel(oldNames)
    if (~isempty(oldNames{inx})) && (~strcmp(oldNames{inx}(1),'<'))
        notProp(end+1) = inx;
    end
end

allLines = allLines(notProp);
oldNames = oldNames(notProp);
newNames = oldNames;

%% Step 3: Now use the power of regexp to clean up the names...

for inx = 1 : numel(oldNames)
    newNames{inx} = regexprep(oldNames{inx},invalidChar,'_');
    % now take care of the starts with a number
    if any(strcmp(newNames{inx}(1),validNum))
        newNames{inx} = ['A',newNames{inx}];
    end
end

%% Step 4: Finally filter down to just the changed names
wasChangeIndex = find(~strcmp(newNames,oldNames));

newNames = newNames(wasChangeIndex);
oldNames = oldNames(wasChangeIndex);
allLines = allLines(wasChangeIndex);
```

2nd Function:

```
% Function: UpdateInvalidSignalNames(model,OldNames,NewNames,lineHandle)
%
% This function updates invalid signal line names with new valid names
%
% A common application of this function is systematic, reliable,
comprehensive
% list-based parameter and signal name changes to large models as naming
conventions evolve over
```

```
% the course of a customer product development process
%
%  Input Arguments:
%  model    :  The name of the model on which the opertation is run
%  OldNames:  A cell array of existing signal and parameter names within a
%             model
%
%  NewNames:  A cell array of new signal and parameter names directly
%             corresponding to the OldNames cell array
%
%  Output Arguments:
%
%     ChangedNames:  A cell array containing a list of the signal and
%     parameter names changed by the function
%
%  Example applications:
%
%  open_system('f14');
%
%  Change name of time-constant parameter 'Ta' to 'TauA'
%
%  ChangedNames=ChangeModelSigAndParmNames(bdroot,{'Ta'},{'TauA'})
%
%
%  Ver 0.1 Created January 29, 2012 by Pete Maloney, MathWorks Consulting
%  Ver 0.2 Created Feb 4, 2012 by Michael Burke
% Revised to suit application by Matthew Boley


function UpdateInvalidSignalNames(model,OldNames,NewNames,lineHandle)

    %% Step 1: Update the names to the "new names"
   for inx = 1 : numel(OldNames)
       set_param(lineHandle(inx),'Name',NewNames{inx})
   end

   %% Step 2: Start updating the bus blocks
   busSelector = find_system(model, 'BlockType', 'BusSelector');


   for inx = 1 : numel(busSelector)
       curOutputs = get_param(busSelector{inx},'OutputSignals');
       newOutputs = [];
       %% this is returned as a comma and dot seperated string, break it into
parts.
       [array,num,token] = strToArray(curOutputs,',','.');
       for jnx = 1 : num
           %index for line names
           index = find(strcmp(array{jnx},OldNames));
           if token{jnx}==1 %comma
                if (isempty(index))
                    newOutputs = [newOutputs,',',array{jnx}];
                else
                    index1{1}=index(1);
                    newOutputs = [newOutputs,',',NewNames{index1{1}}];
                end
```

```
            else        %dot
                if (isempty(index));
                     newOutputs = [newOutputs,'.',array{jnx}];
                else
                     index1{1}=index(1);
                     newOutputs = [newOutputs,'.',NewNames{index1{1}}];
                end
            end
        end
        % Trim the leading ","
        newOutputs = newOutputs(2:end);
        %% Now update the bus outputs...

        set_param(busSelector{inx},'OutputSignals',newOutputs)


    end

end

%String to Array, with two types of delimiters
function [arrayOut,arraySize,delim] = strToArray(str,sep1,sep2)

    loc        = findstr(str,sep1); %find index of seperator 1
    loc1       =findstr(str,sep2);  %find index of seperator2

    if (isempty(loc1)) && (isempty(loc))%if only 1 variable is passed through
    arraySize=1;
    arrayOut{1}=str(1:end);
    delim{1}=1;

    elseif (isempty(loc1)) %no propogated bus line names neeeded
    arraySize = numel(loc) + 1;
    arrayOut{1} = str(1:loc(1)-1);
    delim{1}=1;
    arrayOut{arraySize} = str(loc(end)+1:end);
    delim{arraySize}=1;
        for inx = 2: arraySize - 1
            arrayOut{inx} = str(loc(inx-1)+1:loc(inx)-1);
            delim{inx}=1;                                   %delim=1
comma,delim=0 dot
        end

    else %lets include bus line names
    arraySize = numel(loc) + numel(loc1)+1;
    %combine loc and loc1 into one along with corresponding token
    a=length(loc);
    b=length(loc1);
    A(1:a)=1;
    B(1:b)=0;
    Newloc=[1,loc,loc1;1,A,B];  %has dummy column
    Newloc1=Newloc';
    Newloc2=sortrows(Newloc1,1); %sort indexes but keep tokens with correct
indexes
    NewlocSort=Newloc2';
    %for simplicity seperate rows
    loc3=NewlocSort(1,1:end);
    loc4=NewlocSort(2,1:end);

    %create an array based on delimited
    arrayOut{1} = str(1:loc3(2)-1);
    delim{1}=loc4(1);
    arrayOut{arraySize} = str(loc3(end)+1:end);
    delim{arraySize}=loc4(arraySize);
        for inx = 2: arraySize - 1
            arrayOut{inx} = str(loc3(inx)+1:loc3(inx+1)-1);
            delim{inx}=loc4(inx);
        end
    end
end % ends function
```

3rd Function:

```
%% Example Function:  GenerateBusObjectForBusCreator
%
% For use on the Woodward project
%
% This function looks for bus creator blocks in the model and uses the
% inputs of the blocks to create bus objects.
%
% Inputs:
%   model --> the name of the model
%
% Outputs:
%   busObjects    --> A collection of bus objects.
%
% Notes and limitations:
%
% Auth/Revision:  Michael Burke
%                 Copyright 2012, The MathWorks Consulting Group.
% Revised to suit application by Matthew Boley

function [busObjects] = GenerateBusObjectForBusCreator(model)
    busCreator = find_system(model, 'BlockType', 'BusCreator')
    busObjects = []

    % This function requires the model to be put into compiled mode.
    eval([model,'([],[],[],''compile'')'])
    for inx = 1 : numel(busCreator)
        %% Create the "root" bus
        outName = get_param(busCreator{inx},'OutputSignalNames')
        if (isempty(outName)) || (~isempty(strfind(outName{1},',')))
            outName = {['BusObj_',num2str(inx)]}
        end
        busName{inx} = outName{1}
        busObjects.(outName{1}) = Simulink.Bus
        %% get the list of inputs to the bus creator
        sigNames = get_param(busCreator{inx},'InputSignalNames')
        % Check to see if the input is a bus itself
        isBus = isaBus(busCreator{inx})
        for jnx = 1 : length(sigNames)
```

```
            if isBus(jnx)==1
                be = Simulink.BusElement
                be.Name = sigNames{jnx}
                be.DataType = ['Bus: ',sigNames{jnx}]
                busObjects.(outName{1}).Elements(jnx) = be
            else
                be = Simulink.BusElement
                be.Name = sigNames{jnx}
                busObjects.(outName{1}).Elements(jnx) = be
            end
        end
        % Now add it to the base workspace
        assignin('base',outName{1},busObjects.(outName{1}))
        %% Now set the bus type of the bus creator to that Bus Object
    end
    %% Take the model out of compiled mode
    eval([model,'([],[],[],''term'')'])
    % Now loop through the buses and set the data type
    for inx = 1 : numel(busCreator)
        set_param(busCreator{inx},'OutDataTypeStr',['Bus: ',busName{inx}])
    end

end

function [isBus] = isaBus(busCreator)

    portH  = get_param(busCreator,'PortHandles');
    portIO = get_param(portH.Inport,'CompiledBusType');
    notBus = find(strcmp(portIO,'NOT_BUS'));
    isBus1=find(strcmp(portIO,'VIRTUAL_BUS'));
    isBus(notBus) = 0;
    isBus(isBus1) = 1;

end
```

Legacy Code Tool Setup Script:

```
%% setUpLCT:
% This function creates a legacy code block for the steam table functions.
% The source files "steamLCT.c" and "steamLCT.h" are derived from the
% Sfun_steam_SI.c file.  The are the same file with the "mdl<FUN>" stripped
% out and the interface updated.
%
%
% Auth/Revision:  Michael Burke
%                 MathWorks Consulting
%
% Dependencies: None
%
% Copyright 2012 MathWorks, Inc



%% Set up the legacy code tool for the existing Steam calculations
def = legacy_code('initialize');
```

```
%% Define the files
def.SourceFiles = {'steamLCT.c'};
def.HeaderFiles = {'steamLCT.h'};

%% Define the interface:
def.OutputFcnSpec= 'void steam(int16 u1, double u2, double u3, double u4,
double u5, double u6, double y1[14])';

def.SFunctionName = 'steam';

%% Simulation version
legacy_code('sfcn_cmex_generate',def);

%% compile it
legacy_code('compile',def);

%% get the S-function block
legacy_code('slblock_generate',def);

%% TLC file
legacy_code('sfcn_tlc_generate',def);
```

Continuous to Discrete Model State Conversion Script:

```
%% Find Transfer Functions with Intial conditions
model='model';

TS=.01;     %simulation time step

%% Find all continous TFs
trnsfnc=find_system(model,'Blocktype','TransferFcn');
trnsfncnum=get_param(trnsfnc,'Numerator');
trnsfncden=get_param(trnsfnc,'Denominator');


%% Find all continous TFs with outputs
trnsfnc1=find_system(model,'MaskType','Transfer Function with Initial
Outputs');
N1=get_param(trnsfnc1, 'N');
D1=get_param(trnsfnc1, 'D');
Y0=get_param(trnsfnc1, 'Y0');
U0=get_param(trnsfnc1, 'U0');

%% Find all continous TFs with Intial state
trnsfnc2=find_system(model,'MaskType','Transfer Function with Initial
States');
N2=get_param(trnsfnc2, 'N');
D2=get_param(trnsfnc2, 'D');
X0=get_param(trnsfnc2, 'X0');

%% Find All Continous Integrators
trnsfnc3=find_system(model,'Blocktype','Integrator');
```

```
trnsfnculmt=get_param(trnsfnc3,'UpperSaturationLimit');   %Find Upper Sat.
Limit
trnsfncllmt=get_param(trnsfnc3,'LowerSaturationLimit');   %Find Lower Sat.
Limit
trnsfncic=get_param(trnsfnc3,'InitialConditionSource');   %Determine IC
Source
trnsfncic1=get_param(trnsfnc3,'InitialCondition');        %If internal IC
find, or it will be 0 either external or if IC is set at 0
reset=get_param(trnsfnc3, 'ExternalReset');               %External Reset
type
stateport=get_param(trnsfnc3, 'ShowStatePort');           %State Port on?
satport=get_param(trnsfnc3, 'ShowSaturationPort');        %Sat Port on?
ignlimit=get_param(trnsfnc3, 'IgnoreLimit');              %ignore limit when
linearizing?

%% Create Arrays from 3 different TF types + Integrators along with each
respective properties in the location based on thier length
%create string of zeros
for tnx=1:length(trnsfnc3)
    Zero1='[0]';
    Zero2=cellstr(Zero1);
    Zero3(tnx,1)=Zero2;
end

%Lenghts
p=length(trnsfnc);
k=length(trnsfnc1);
l=length(trnsfnc2);
w=length(trnsfnc3);

%Blocks Assign into 1 cell array
Blocks(1:p,1)=trnsfnc;
Blocks(p+1:(p+k),1)=trnsfnc1;
Blocks((p+k)+1:(p+k+l),1)=trnsfnc2;
Blocks((p+k+l)+1:(p+k+l+w),1)=trnsfnc3;


%Numerator
N3=cell(p+k+l+w,1);
N3(1:p,1)=trnsfncnum;
N3(p+1:(p+k),1)=N1;
N3((p+k)+1:(p+k+l),1)=N2;
N3((p+k+l)+1:(p+k+l+w),1)=Zero3;


%Denominator
D3=cell(p+k+l+w,1);
D3(1:p,1)=trnsfncden;
D3(p+1:p+k,1)=D1;
D3(p+k+1:(p+k+l),1)=D2;
D3((p+k+l)+1:(p+k+l+w),1)=Zero3;

%Y0
Y01=cell(p+k+l+w,1);
```

```
Y01(p+1:p+k,1)=Y0;

%U0
U01=cell(p+k+l+w,1);
U01(p+1:p+k,1)=U0;

%X0
X01=cell(p+k+l+w,1);
X01(p+k+1:(p+k+l),1)=X0;

%Integrator Upper Limit
trnsfnculmt1=cell(p+k+l+w,1);
trnsfnculmt1(p+k+l+1:(p+k+l+w),1)=trnsfnculmt;

%Integrator Lower Limit
trnsfncllmt1=cell(p+k+l+w,1);
trnsfncllmt1(p+k+l+1:(p+k+l+w),1)=trnsfncllmt;

%Integrator IC source
trnsfncic2=cell(p+k+l+w,1);
trnsfncic2(p+k+l+1:(p+k+l+w),1)=trnsfncic;

%Integrator IC
trnsfncic3=cell(p+k+l+w,1);
trnsfncic3(p+k+l+1:(p+k+l+w),1)=trnsfncic1;

%Integrator Reset
reset1=cell(p+k+l+w,1);
reset1(p+k+l+1:(p+k+l+w),1)=reset;

%Integrator Stateport
stateport1=cell(p+k+l+w,1);
stateport1(p+k+l+1:(p+k+l+w),1)=stateport;

%Integrator Satport
satport1=cell(p+k+l+w,1);
satport1(p+k+l+1:(p+k+l+w),1)=satport;

%Integrator Ingnore Limit When Linearizing?
ignlimit1=cell(p+k+l+w,1);
ignlimit1(p+k+l+1:(p+k+l+w),1)=ignlimit;

%Find All names and Positons
parents = get_param(Blocks,'Parent');                    %Find Parents
Names   = get_param(Blocks,'Name');                      %Find Block Names
pos     = get_param(Blocks,'Position');                 %Find Positions
BlockMirror=get_param(Blocks,'BlockMirror');             %mirrored?
BlockRotation=get_param(Blocks,'BlockRotation');         %rotation


%% Loop to convert from continous to discrete
for inx=1:length(Blocks)
    K=str2num(N3{inx});
    Q=str2num(D3{inx});
    %convert zeros to something that will convert but wont ever use
        if  K==0
            K=[1];
        else
            %nada-tostada
        end
        if  Q==0
            Q=[1 1];
        else
            %nada-tostada
        end
    %create TF then convert to discrete
    [a,b,c,d]=tf2ss(K,Q);
    StateSpace_S =ss(a,b,c,d);
    StateSpace_Z = c2d(StateSpace_S,TS,'zoh');
    TF_Z = tf(StateSpace_Z);

    %Index cells of discrete values for numerator and denominator
    [num(inx),den(inx)]=tfdata(TF_Z);
end

%% Data Processing
%Transpose it
N=num';
D=den';

%Make TS a string
TS1=num2str(TS);

%% Loop to convert from number to string, then index in cell array
for unx=1:length(Blocks)
    %Numerator
    N4=cell2mat(N(unx));
    N5=mat2str(N4);
    N6=cellstr(N5);
    N7(unx,1)=N6;

    %Denominator
    D4=cell2mat(D(unx));
    D5=mat2str(D4);
    D6=cellstr(D5);
    D7(unx,1)=D6;

end



%% Loop to Replace Blocks
for inx=1:p %Reg TF's
    delete_block(Blocks{inx})
    add_block('built-
in/DiscreteTransferFcn',[parents{inx},'/',Names{inx}],...
        'Position',pos{inx},'Denominator',D7{inx},'Numerator',N7{inx},...
        'SampleTime',TS1,'BlockMirror',BlockMirror{inx},...
        'BlockRotation',BlockRotation{inx})
end
```

```
for jnx=p+1:p+k %TF's with Initial outputs
    delete_block(Blocks{jnx})
    add_block('simulink_extras/Additional Discrete/Discrete Transfer Fcn
(with initial outputs)',...
        [parents{jnx},'/',Names{jnx}],'Position',pos{jnx},'D',D7{jnx},...
        'N',N7{jnx},'TS',TS1,'Y0',Y01{jnx},'U0',U01{jnx},...
        'BlockMirror',BlockMirror{jnx},'BlockRotation',BlockRotation{jnx})
end

for knx=p+k+1:p+k+l %TF's with Initial State
delete_block(Blocks{knx})
    add_block('simulink_extras/Additional Discrete/Discrete Transfer Fcn
(with initial states)',...
        [parents{knx},'/',Names{knx}],'Position',pos{knx},'D',D7{knx},...
        'N',N7{knx},'TS',TS1,'X0',X01{knx},...
        'BlockMirror',BlockMirror{jnx},'BlockRotation',BlockRotation{jnx})
end

for mnx=p+k+l+1:p+k+l+w %Integrators
    delete_block(Blocks{mnx})
    add_block('built-in/DiscreteIntegrator',[parents{mnx},'/',Names{mnx}],...
               'Position',pos{mnx},'InitialConditionSource',
trnsfncic2{mnx},...
               'ExternalReset', reset1{mnx},'ShowStatePort',
stateport1{mnx},...
               'ShowSaturationPort', satport1{mnx}, 'LimitOutput', 'on',...
               'UpperSaturationLimit',trnsfnculmt1{mnx},...
               'LowerSaturationLimit',trnsfncllmt1{mnx},...
               'InitialCondition', trnsfncic3{mnx},'IgnoreLimit',
ignlimit1{mnx},...

'BlockMirror',BlockMirror{mnx},'BlockRotation',BlockRotation{mnx},'SampleTime
',TS1);

end
disp('Continous to Discrete Conversion Complete')
```