

REAL-TIME SIMULATION OF A PARABOLIC TROUGH SOLAR FIELD

by

Matthew J. Tuman

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science
(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2024

ACKNOWLEDGMENTS

Quote

— QUOTED PERSON (YEAR)

Use this section to acknowledge those who have provided support, technical or otherwise, in pursuit of your degree.

CONTENTS

Contents	ii
List of Tables	v
List of Figures	vi
Nomenclature	x
Abstract	xiii
1 Introduction	1
1.1 <i>Solana</i>	6
1.2 <i>Software Overview</i>	8
1.2.1 <i>TRNSYS</i>	8
1.2.2 <i>FORTRAN</i>	9
1.2.3 <i>Python</i>	9
2 Literature Review	10
3 Solar Field Model Methodology	17
3.1 <i>Hydraulics</i>	17
3.1.1 <i>Pumps</i>	18
3.1.2 <i>Pressure Drop</i>	21
3.1.3 <i>Mass Flow Solver</i>	28
3.1.4 <i>Expansion and Overflow System</i>	38
3.2 <i>Thermals</i>	43
3.2.1 <i>Collector Optics</i>	43
3.2.2 <i>Solar Field Loop Thermal Model</i>	53
3.2.3 <i>Receiver Heat Loss Model Overview</i>	58

3.2.4	<i>Neural Network Architecture and Training Protocol</i>	60
3.2.5	<i>Solar Field Sector Model</i>	65
3.2.6	<i>SCA Defocusing</i>	70
3.2.7	<i>Expansion System</i>	71
4	Results	73
4.1	<i>Model Validation</i>	73
4.1.1	<i>Neural Network</i>	73
4.1.2	<i>Solar Field Sector</i>	80
4.2	<i>Case Studies</i>	85
4.2.1	<i>Solar Field Start Up</i>	85
4.2.2	<i>Hydrogen Variation</i>	91
4.2.3	<i>Sector Flow Rate Variability</i>	93
5	Solar Transformer Assistant Neural Network (STANN)	102
5.1	<i>Methodology</i>	104
5.1.1	<i>Q-Learning</i>	104
5.1.2	<i>Preprocessing</i>	106
5.1.3	<i>Observational Embedding</i>	107
5.1.4	<i>Positional Encoding</i>	109
5.1.5	<i>Transformer Encoder</i>	111
5.1.6	<i>Convolutional Neural Network</i>	115
5.1.7	<i>Final Q-Learning Network</i>	116
5.2	<i>Plant Start Up</i>	117
5.2.1	<i>Model Parameters and Training</i>	120
5.2.2	<i>Results</i>	122
6	Conclusions	126
6.1	<i>Future Work</i>	127

6.1.1	<i>Solar Field Model</i>	127
6.1.2	<i>STANN</i>	129
	References	131

LIST OF TABLES

3.1	Loss coefficients for various piping components	24
3.2	Operational modes of expansion system	40
3.3	Neural network architecture utilized to compute heat absorption	60
3.4	Limits for neural network training dataset	61
3.5	Number of samples within the training dataset for each neural network	63
3.6	User-defined parameters utilized to generate solar field loops in a sector . . .	67
4.1	Parameters used in neural network validation simulation	74
4.2	Error between Forristall model and neural network approach for various re- ceiver states	75
4.3	Input values used in case study	76
4.4	Parameters obtained from fine tuning the simulation results	81
5.1	Various measurements used to represent the environment's state along with the number of tokens and bounds associated with each measurement	118
5.2	Possible actions that an agent can make	119

LIST OF FIGURES

1.1	Parabolic trough collector at the Solana power plant	1
1.2	Electricity demand and contribution from renewable energy sources in California [1]	3
1.3	Visual representation of parabolic trough power plant	5
1.4	Google satellite image of Solana outlining one of eight sectors	7
2.1	Illustration of receiver along with the relevant heat transfer mechanisms	11
3.1	Solar field methodology workflow	17
3.2	Different components of a centrifugal pump [2]	18
3.3	Visualization of a fitted pump and NPSHr curve using manufacturer provided data	20
3.4	False Position method utilized to solve Colebrook friction factor formulation .	23
3.5	C_v values as a function of valve position for a butterfly valve	25
3.6	Section of TRNSYS solar field model with various types highlighted	26
3.7	Components of solar field sector used in pressure drop calculation	27
3.8	Hydraulic system solved using Linear Theory Method	29
3.9	Illustration of expansion system comprised of pressurized expansion tanks and overflow tanks	40
3.10	Illustration of expansion system comprised of pressurized expansion tanks and overflow tanks	41
3.11	Solar tracking to maximize solar irradiation upon receiver as sun traverses through the sky	44
3.12	Visualization of	46
3.13	Visualization of sun definition and distribution of angular positions used . . .	49
3.14	Intercept factors computed using various intercept factors	51

3.15 Error between intercept factor curves as a function of rays used during the simulation	52
3.16 Discretization of solar field loop into $N - 1$ control volumes	54
3.17 Simple case study with constant flux applied to HTF in a solar field loop . . .	56
3.18 Comparing analytical solution (line) with numerical model (dots) using different amount of nodes	57
3.19 Instability with numerical model that has nodes in the center of each control volume	58
3.20 Visual representation (not to scale) of neural network used to predict HTF heat absorption (left) and Forristall model energy balance (right)	59
3.21 Various activation functions commonly used in neural networks	61
3.22 Solar field sector model illustrating nodes in finite volume approach used to compute temperatures	66
3.23 User defined hydrogen permeation in the last three receivers with various pressure distributions	68
3.24 Receiver state probabilities	69
3.25 Illustration of tank thermal model	71
4.1 Neural network validation study illustrating the difference in the computed heat absorbed by an entire loop using the Forristall model and the neural network	75
4.2 Parameter sweep investigating heat absorption computed with neural network and Forristall model	77
4.3 Heat absorbed by the HTF as a function of HTF temperature and hydrogen pressure in the annulus	78
4.4 Absolute difference in heat absorbed by the HTF using the Forristall model and the neural network approach while varying HTF temperature and hydrogen pressure in the annulus	79

4.5	Sector visualization with DNI measured at the yellow dot and HTF measurements at the black dot	81
4.6	Solar field sector responses for three separate days comparing simulated sector outlet temperatures (black) and measured outlet temperatures (red dots) . .	82
4.7	Intercept metric comparing heat rejection via defocusing for the model (black) and operation (red)	84
4.8	Preliminary interface for an operator-controlled daily start-up	86
4.9	Solar field sector temperature distribution during start up with cold slug moving through it	87
4.10	User-induced pump cavitation due to insufficient level in the expansion tanks during a startup sequence	89
4.11	Time lag of HTF returning from the solar field back to the central plant	90
4.12	Additional heat loss of a solar field sector as a function of hydrogen pressure in the annulus (see legend) and percentage of receivers in the last SCA with hydrogen permeation	92
4.13	Optical efficiency (top plot) and number of loops defocused in the sector (bottom plot) for simulations with varying mass flow rates	94
4.14	Temperature of 4 th SCA for all loops in a sector (top plot) with colors in reference to loops drawn (bottom plot)	95
4.15	Outlet temperatures of HTF at different flow rates	98
4.16	Impact of varying flow rates on solar field performance parameters	99
5.1	Architecture of the Solar Transformer Assistant Neural Network	102
5.2	Example of taking continuous measurements m and mapping them to tokens x	106
5.3	Transforming tokens to unique vectors of length $n_{dim} = 4$ with observation embedding	108
5.4	Generation of unique vectors for individual tokens using feed-forward neural network	108

5.5	Applying positional encoding to the observational embedding matrix	110
5.6	Creating Query, Key, and Value matrices from the transformer input using linear feed forward networks	111
5.7	Dividing the Query, Key, and Value matrices according to $n_{\text{head}} = 2$	112
5.8	Self-attention matrix generated from Query/Key combination	112
5.9	Generation of final attention matrix for the i^{th} head	113
5.10	Integrating key information learned from each attention head via concatenation and feed forward neural network	114
5.11	INSERT CAPTION	114
5.12	Convolutional neural network used to compress data from transformer encoder	116
5.13	Mapping the compressed data from the CNN to Q-values using a linear feed- forward neural network	116
5.14	GUI for operator-controlled daily start-up	117
5.15	Cumulative rewards obtained with optimal policy as a function of number of simulations included in the training memory	123
5.16	Visualization of STANN providing helpful recommendations that are used to get out of undesirable plant states	124

DISCARD THIS PAGE

NOMENCLATURE

Abbreviations

ANI	Aperture Normal Irradiance
CSP	Concentrating solar power
DEWA	Dubai Electricity and Water Authority
DNI	Direct Normal Irradiance
DOE	Department of Energy
HTF	Heat Transfer Fluid
NPSH _r	Net Positive Section Head Required
NREL	National Renewable Energy Laboratory
PTC	Parabolic Trough Collector
TES	Thermal Energy Storage
SCA	Solar Collector Assembly
STANN	Solar Transformer Assistant Neural Network

Variables [units]

A	Area [m ²]
c	Specific heat [J/kg-K]
C _v	Flow Coefficient [GPM/ $\sqrt{\text{psi}}$]
η	Efficiency [-]
D	Diameter [m]
f	Friction Factor [-]
g	Gravitational Acceleration [m/s ²]
h	Specific Enthalpy [J/kg]
H _L	Head Loss [m]
K	Total Loss Coefficient

$k_{\Delta P}$	Component Flow Loss Coefficient
L	Length [m]
$\dot{m}$	Mass flow rate [kg/s]
m	Mass [kg]
N	Number of Items [-]
ΔP	Pressure Drop [Pa]
P	Pressure [Pa]
$\dot{Q}$	Volumetric Flow Rate [m ³ /s]
SG	Specific Gravity [-]
Re	Reynolds Number [-]
ρ	Density [kg/m ³] or Reflectance [-]
s	Fractional pump Speed [-]
T	Temperature
u	Specific Internal Energy [J/kg]
v	Velocity [m/s]
w	Width [m]
$\dot{W}$	Power [W]
Vol	Volume [m ³]

Subscripts

ave	Average
expansion	Expansion Tanks
header	Any header
hrd,c	Cold Header
hdr,h	Hot Header
i	Index
IOCop	Inlet/Outlet/Cross-Over Piping

loop	Solar Field Loop
network	HTF Piping Network (Everything but expansion system)
overflow	Overflow Tanks
sector	Solar Field Sector
soil	Soiling of collectors (dirt on mirrors)
spacing	Solar Field Loop Spacing
track	Tracking error

ABSTRACT

Concentrating Solar Power (CSP) plants are a crucial component of today's renewable energy technologies. CSP plants offer economies of scale as they increase in size, and they can incorporate thermal energy storage to improve dispatch capabilities as well as extend operation into the night. However, achieving and maintaining high levels of performance in large-scale and complex CSP plants is challenging. Enhancing operator training has been identified as a key area for growth and improvement of the performance of these plants.

This work presents a high fidelity parabolic trough solar field model capable of emulating CSP plant dynamics for use in an operator training simulator. This flexible, accurate, and computationally efficient model uses a novel neural network methodology to calculate the heat absorbed by the heat transfer fluid (HTF) under various receiver conditions. Validated with real operational data from the Solana plant, the complete solar field model includes 808 solar loops and can simulate responses to operator inputs like pump speeds and valve positions. The complete solar field model solves 20 times faster than real-time. This detailed yet computationally efficient model is not only ideal for training simulators but also offers potential for optimizing future solar field designs and operations.

Additionally, this research presents a new artificial intelligence (AI) model that learns the complex dynamics of CSP plants. This AI model can support operators during training simulations by offering real-time feedback and assisting in the resolution of challenging scenarios. Combining advanced modeling techniques with AI, this approach hopefully will enhance operator training and has the potential to improve overall plant performance.

1 INTRODUCTION

Concentrating solar power (CSP) is one of the most fascinating technological advancements emerging from the transition to combustion-free energy production. The technology utilizes an idea that even most children can understand: concentrating sunlight makes things hot. A simple demonstration of this concept can be achieved by using a magnifying glass to focus sunlight onto a dead leaf which, if focusing the light long enough, will cause the leaf to ignite. Instead of using magnifying glasses, CSP plants utilize mirrors to reflect and concentrate solar energy for heat collection. Although many concepts have been devised to concentrate the sunlight, the two dominant designs today are parabolic trough and power tower plants. While tower designs have been the target of recent research in the CSP community due to improved power cycle efficiencies resulting from higher temperatures, the vast majority of CSP plants in operation today utilize parabolic trough collector (PTC) technology. A visualization of a parabolic trough collector is presented in Figure 1.1.

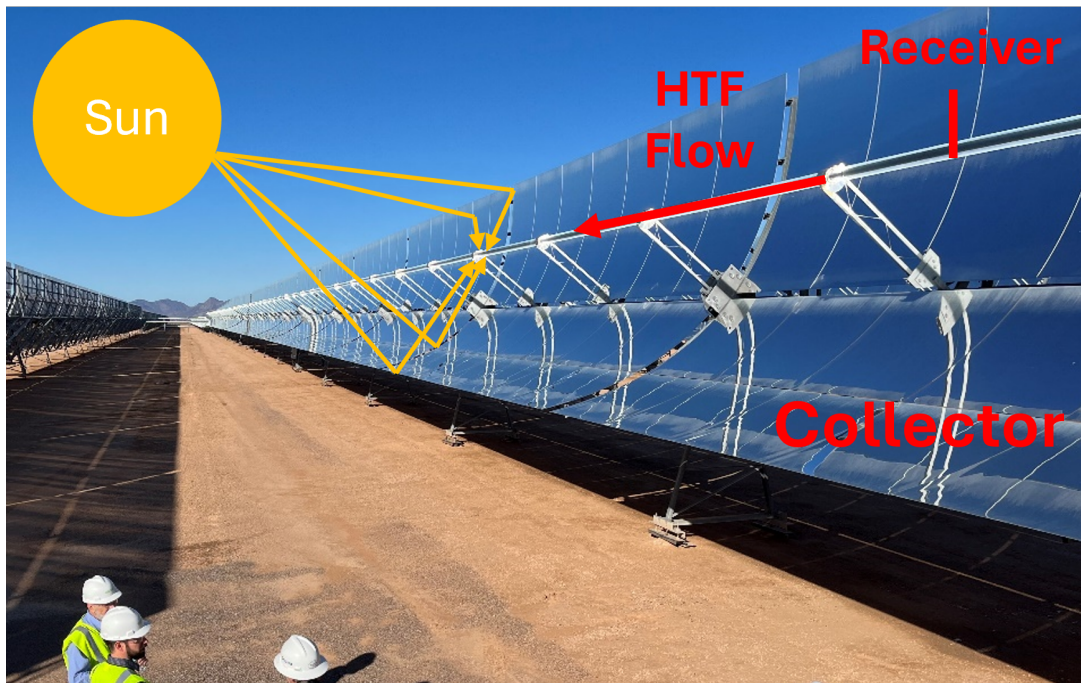


Figure 1.1: Parabolic trough collector at the Solana power plant

A parabolic mirror (collector) is designed to reflect and concentrate solar radiation to a central absorber tube (receiver). A heat transfer fluid (HTF) flowing through the absorber tube collects the radiative solar energy converting it to thermal energy resulting in a temperature rise of the HTF. The external surface of the absorber tube is treated with a coating that maximizes its absorptivity in the short wave length spectrum and minimizes emissivity in the long wave length spectrum. In addition, the absorber tube is enclosed by an evacuated glass tube which is utilized to minimize convective thermal losses. The receiver-collector combination is referred to as a solar collector assembly (SCA). Each SCA is attached to a motor that rotates the collectors about the receiver's horizontal axis in order to maximize the normal solar energy incident upon the mirrors as the sun moves across the sky throughout the day. Furthermore, most operational PTC plants utilize a thermal oil as the HTF which can be heated up to around 740 [F] before the fluid begins to degrade. New PTC plants propose using molten salt which can be heated to about 1000 [F]. Although molten salt can reach higher temperatures, its relatively high freezing point poses challenges. These include the risk of HTF solidification during nighttime operations and extended outages which must be managed carefully. Regardless of the fluid used, PTC technology can collect and transfer a significant amount of thermal energy to the HTF. This heat is most commonly used as input for a power cycle that generates electricity, but there is also increasing interest in using CSP technology for industrial processes that require large amounts of heat typically provided through burning hydrocarbons or resistive heating.

With the goal of transitioning to a clean electricity grid, CSP plants are a crucial component of today's renewable energy technologies. Both parabolic trough and power tower CSP plants offer economies of scale as they increase in size. Additionally, they can incorporate thermal energy storage (TES) to improve dispatch capability and extend operation during periods of low solar availability. For example, CSP plants with TES are typically designed such that the heat absorbed by the solar field is greater than the load

required by the power cycle. Hot HTF that is not required by the power cycle is then either directly stored in large tanks or used to heat a different fluid that is stored in the tanks. Regardless of the mechanism, the excess heat is stored in these hot tanks and the hot fluid within them can be used to produce steam for electricity production when there is zero solar availability: a side effect of clouds or nighttime. This capability is especially attractive when one considers a typical electricity demand curve for the California grid which is presented in Figure 1.2.

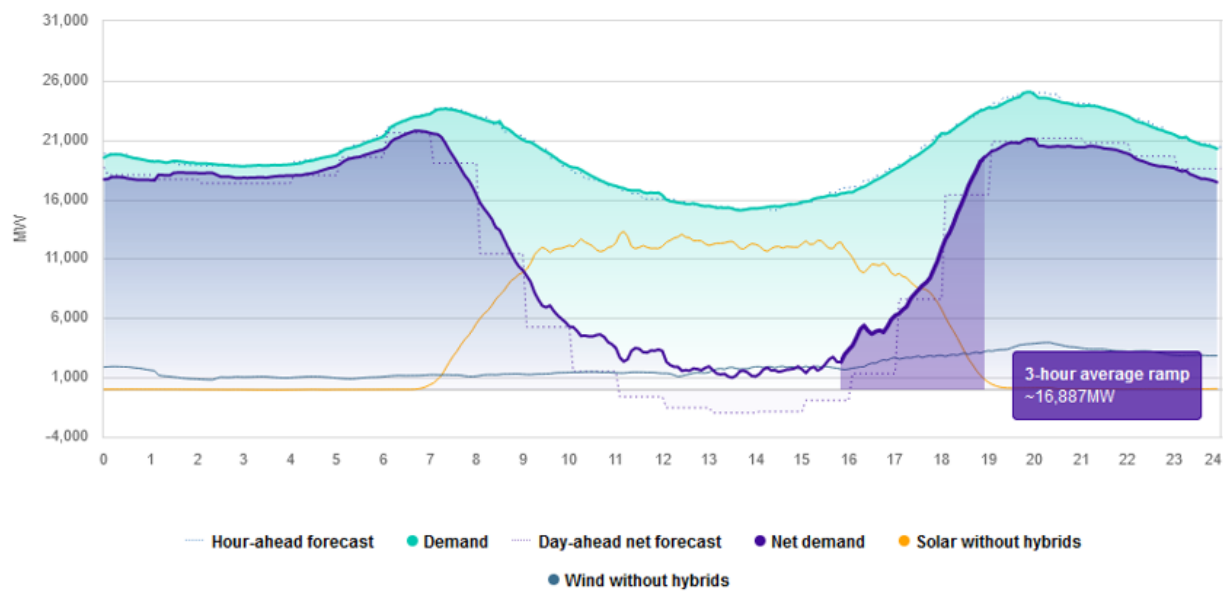


Figure 1.2: Electricity demand and contribution from renewable energy sources in California [1]

The light blue line is the electricity demand, the yellow line is the electricity provided by solar, the dark blue line is the electricity provided by wind, and the purple line is the electricity that must be provided by non-renewable sources. Currently, California's solar energy generation is dominated by photovoltaic panels, so electricity can only be produced in proportion to the radiation available. Interestingly, there are two peaks in electricity demand, and both of these occur when there is either low or zero solar availability. Thus, CSP plants could be incredibly useful for this grid in that they could collect and store thermal energy throughout the day and dispatch it to generate electricity during the morning or evening peak in electricity.

Although the idea and advantages of CSP technology are straight forward, CSP plants are extremely complex, so achieving and maintaining high levels of performance at a large-scale is challenging. The underwhelming performance of existing CSP plants in the U.S. partly stems from the ongoing maturation of technology, which will naturally improve over time with more operational experience. Yet, there are also issues such as insufficient operator skill development and barriers to information sharing among plants that need addressing. A U.S. based team of CSP experts [3] conducted a comprehensive effort to compile and communicate best practices for engineering, design, construction, start-up, and ongoing operations and maintenance of CSP parabolic trough and power tower systems. Specifically, they point out "[p]lant simulators would likely be beneficial for training and familiarization of plant procedures and operations. It is believed that simulation for CSP plant operations for training purposes is in its infancy; yet, technology seems to be available to develop this concept more thoroughly. This could be an area of future development and improvement among the CSP community." This work aims to fill this gap by developing a realistic parabolic trough solar field model that can effectively emulate CSP plant dynamics and can be used as a pilot operator training simulator. With this goal in mind, the model must have sufficient complexity to accurately predict the thermal energy collection, including transient effects, yet it also must run faster than real time meaning. That is, if the simulation utilizes one second time-steps, the model must solve faster than one second. Creating a highly detailed model that is also computationally efficient was the greatest challenge of this work.

The main components of a CSP plant are the solar field, thermal energy storage, and a power block that generates electricity. A simplified illustration of a parabolic trough CSP plant is presented in Figure 1.3.

The solar field is responsible for collecting the solar thermal energy used by the plant, and it is comprised of numerous solar field loops in parallel. Each loop contains

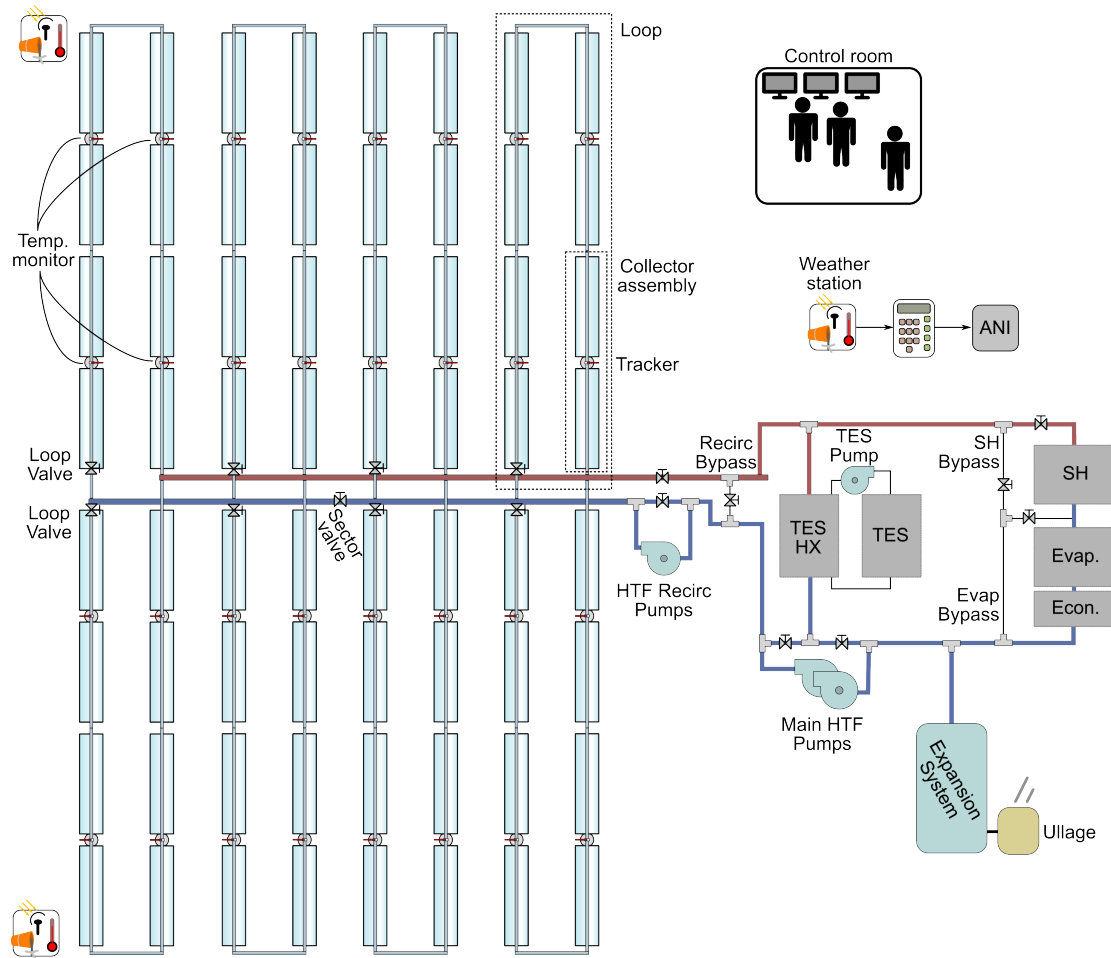


Figure 1.3: Visual representation of parabolic trough power plant

multiple SCA's in series each equipped with their own tracking drive and temperature sensor. As illustrated in Figure 1.3 cold HTF (blue) is sent out to a solar field via an inlet header pipe that delivers flow to the solar field loops. Once the HTF is heated in the loops (red), it returns back to the plant via a return header pipe where it can be used to generate electricity through the production of superheated steam, heat a secondary fluid for thermal energy storage, or to be recirculated back to the solar field. While the thermal energy storage and power block are key components of the plant, this work focuses on the development of a solar field model. While most PTC plant models obviously include the solar field, this model also includes detailed models of often over-looked equipment such as control valves, main HTF pumps, piping at the central plant, and expansion system

that are highly relevant for training new operators.

This work is presented in six chapters. Chapter 2 explores the previously developed models for PTC CSP plants and clearly distinguishes the additional capabilities this model provides. Chapter 3 outlines the methodology of this model. This includes an in-depth discussion of methods used to compute flow rates of HTF throughout the complex piping network determined by operator decisions along with a novel method for quickly computing heat loss for receivers in various states using a neural network. Chapter 4 validates the methodologies presented in the previous chapter utilizing data from an operational CSP plant, and it also explores three case studies illustrating the model's capabilities. First, a preliminary simulation tool is presented that can test a user's ability to start up a plant from sunrise. Second, plant performance as a function of varying amounts of hydrogen in solar field receivers is investigated. Third, the trade-off between HTF outlet temperature and defocusing is explored, and a recommendation for operation of a solar field is provided. Lastly, with the intent to help train operators how to operate the plant, Chapter 5 presents an artificial intelligence (AI) tool developed for this purpose. This tool, called the Solar Transformer Assistant Neural Network, leverages many of the recent advancements in AI to learn complex behavior of a CSP plant. Based on the entirety of the work discussed, conclusions and future work are presented in Chapter 6.

1.1 Solana

While the solar field model developed is capable of representing a wide range of PTC CSP plants, this work focuses on a solar field model that mimics a plant similar to Solana. Solana is 280-MW steam Rankine cycle plant with a TES system that resides about an hour outside of Phoenix, AZ. If the hot TES tanks are filled to capacity during the day, the plant can generate electricity for an additional 6 hours after sunset thereby increasing renewable contribution to meet the evening peak in electricity demand. Annually, Solana

generates around 900,000 MWhr of electricity and prevents 480,000 metric tons of carbon dioxide emissions [4]. A satellite image of Solana is provided in Figure 1.4.



Figure 1.4: Google satellite image of Solana outlining one of eight sectors

The sheer size of Solana is quite impressive as it has a solar field that is about 2 miles wide and has a total aperture area of collectors just under a square mile. Due to its size, the solar field is discretized into eight distinct sectors. Each sector contains one inlet header pipe and return header pipe connected to numerous solar field loops in parallel. At Solana, there are 808 loops in total with the sectors containing 104 and 98 loops for the far field sectors and near field sectors respectively (Sector 1 in Figure 1.4 is a far field). Furthermore, each loop is comprised of four solar collector assemblies (SCA's) which are labeled sequentially from the inlet to the outlet. For example, the SCA nearest to the HTF inlet of the loop is labeled as SCA #1 and the nearest to the HTF outlet is SCA #4.

At the time of writing, Solana is the largest operational PTC CSP plant (DEWA 600 MW project in construction). Because of its size, there are numerous complexities that

exist at Solana that are not present at other smaller plants. These complexities will be discussed further in Chapter 3. Nevertheless, by tackling these unique challenges at Solana, the robust methodologies developed are applicable to a broad spectrum of CSP projects, from large-scale projects like Solana to simpler single-loop plants for industrial heating.

1.2 Software Overview

The modeling tools developed in this work are written in FORTRAN and Python. FORTRAN is the language utilized by the simulation software TRNSYS which manages the entire model, and Python is used for the various machine learning applications used.

1.2.1 TRNSYS

The solar field model in this work was developed and runs entirely in TRNSYS, a software tool designed for transient simulations. TRNSYS, the Transient System Simulation Tool, was originally developed at the University of Wisconsin - Madison, and it is widely used in industry for modeling complex systems such as power cycles or heating for buildings. Essentially, it solves large systems of coupled, nonlinear equations using successive substitution at each timestep in a transient simulation. TRNSYS models are built using a modular approach by connecting "Types" which can be thought of as functions that have user defined parameters, take in input values, and return output values. A TRNSYS model is constructed by connecting outputs of Types to inputs of other Types. For example, the output flow rate from a pump Type would be connected to the input flow rate for a header pipe Type sending flow out to the solar field. User-defined Types and TRNSYS itself are written in FORTRAN yielding impressive computational efficiency. While there are numerous predefined types, all the methodologies discussed in Chapter 3 are implemented in user-defined Types. The flexibility to create and write the code for

user-defined types is one of the main advantages of TRNSYS.

1.2.2 FORTRAN

FORTRAN is a compiled programming language that is well-suited for scientific computing. It was originally developed in the 1950s and was the first programming language that allowed scientists to input their own code as opposed to requiring a programmer to translate their ideas to machine code. While its roots are old, it has been continually updated and is still a highly relevant language today. Its main benefit is speed. FORTRAN was designed for computationally intensive applications, and compilers have been developed to optimize the speed of executed FORTRAN code.

1.2.3 Python

Python is an interpreted programming language that is also well-suited for scientific computing. However, because Python is interpreted, it lacks the computational efficiency provided by FORTRAN. Nonetheless, Python is arguably one of the most popular programming languages in existence today, and numerous open-source machine learning libraries have been developed using Python as their source code. Open-source libraries such as JAX (developed by Google), TensorFlow (developed by Google), and PyTorch (developed by Meta) allow users to easily generate and train user-defined neural networks. These advancements have been a gift to the scientific community in that many of the difficult tasks, such as computing gradients of weights in complex neural network architectures, are completely automated. The machine learning applications presented in this work would be much more difficult to implement without these libraries.

2 LITERATURE REVIEW

PTC solar field modeling is not new. Originally, Excelergy [5] was used as the first parabolic trough model in NREL's System Advisor Model (SAM) [6]. This model was based largely on empirical curves that were fit to the performance of operating plants, so it could not be used evaluate new designs or technologies. To address this issue, the physical trough model was developed and integrated into the System Advisor Model [7]. The physical trough model utilizes a physics based approach to capture the thermal behavior of a solar field. Similar to this work, the physical trough model was also developed for implementation into TRNSYS. It was constructed as a performance model designed for predicting daily and annual plant behavior, and the physical trough model is one of the most detailed and flexible simulation tools to date. One of its key features is its ability to compute receiver heat loss using the Forristall model [8]. To better understand this model, it is first necessary to discuss the design of the receivers utilized in PTC solar fields. An illustration of a receiver is provided in Figure 2.1.

The receiver is designed to maximize heat absorbed by the HTF and minimize heat loss to the environment. With that in mind, the absorbed pipe that HTF flows through is surrounded by a glass envelope. During manufacturing, the gas in the annulus space between the glass envelope is evacuated in order to create a vacuum to minimize convective losses between the absorber tube and the ambient environment. A perfectly performing glass envelope would have 100% transmittance and the glass surface would operate at the ambient environment temperature which results in radiation being the only mode of heat transfer between the absorber and the envelope. To further decrease the heat loss, the absorber is coated with a material that acts as a selective surface in that it has a high absorptivity for radiation at short wavelengths originating from the sun and a low emissivity at long wavelengths that correspond to thermal losses from the tube surface

to the surrounding environment. This design, combining the annulus space vacuum with the selective surface coating on the absorber, results in very little heat lost to the environment. Receivers in this state are referred to as "pristine". However, as plants age thermal and aerodynamic stresses cause damage to the receiver. This damage can result in the glass cracking allowing for ambient air to fill the annulus space or the glass can completely shatter, exposing the absorber to ambient conditions. Additionally, as hot HTF exceeds its temperature limit, it degrades and releases hydrogen gas. This hydrogen gas can permeate through the absorber pipe and become trapped within the annulus space between the absorber and glass envelope. Hydrogen is a highly energetic particle, so even relatively low pressures of hydrogen in the annulus space results in significantly more heat loss to the environment via convection between the absorber and glass envelope.

The Foristall model is incredibly useful as it solves for the temperatures of the absorber and glass tube to satisfy the energy balances that govern the system for receivers that are pristine, have broken glass, have lost the vacuum, or have a build up of hydrogen in the annulus space. However, one of the physical trough model's limitations is that it simulates one loop for each assumed receiver condition and extrapolates the results to full field performance. While this methodology is useful for efficiently predicting the performance of plants over days/years, it neglects some of the transient details that occur on the order of minutes that cannot be captured with

large time-steps. Furthermore, the hydraulic solver used to compute HTF flow rates in SAM assumes perfect operator control in order to yield an optimal plant output such as a defined solar field outlet temperature. This means that the model effectively solves

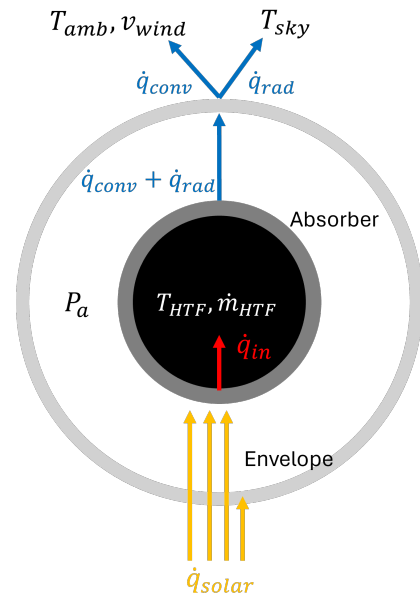


Figure 2.1: Illustration of receiver along with the relevant heat transfer mechanisms

for valve positions and pump speeds as opposed to letting the user define these control decisions.

Around the same time the physical trough model was being developed in the US, Spain's renewable energy department was also developing a performance model using Wolfram's Mathematica 7 software [9]. This model was also highly detailed and validated with operational plant data. Unlike SAM, the Spanish model could simulate transient events as it utilized timesteps of 10 seconds. However, its drawbacks are that the heat loss of the receivers are computed using empirical curve fits from [10] and that the HTF flow rates are computed in a manner similar to SAM where flow is defined such that the plant can achieve optimal performance.

Additionally, many researchers in Europe simulating PTC CSP applications utilize the modeling software Dymola which is built on the Modelica programming language. Recently, there has been an explosion of publications using the Dymola/Modelica [11] [12] [13] [14]. All of these works utilize an open source Modelica library called ThermoCycle [15]. In addition to many other thermodynamic models, this library contains a parabolic trough model that is easily integrated to complex systems modeled in Dymola. As defined by the user, the model computes HTF heat absorption either using the Forristall model or empirical curves derived from experimental measurements.

Currently, the most detailed transient simulation tool for CSP training applications was developed by researchers at the German Aerospace Center (DLR) [16] [17]. This impressive model, called the Virtual Solar Field (VSF), couples a hydraulic solver that computes mass flow rates through the various loops in the solar field based on user-defined valve positions with a detailed transient thermal model that computes HTF temperatures. VSF was the first model that accomplished this. The purpose for the development of VSF over the past several years is to better understand how to run a PTC-CSP plant using molten salt as the HTF instead of the traditionally used oil-based

HTF's. To better understand this new technology, DLR and other partners are building a PTC-CSP test plant that uses molten salt. In preparation for the operation of this test plant, DLR has constructed a training simulator that allows future operators to practice scenarios that they may face in the future [18]. The Virtual Solar Field model has been successfully integrated with a LabView interface that allows future operator to explore the plant's behavior to their actions.

While VSF has proven to be incredibly useful, it does have some limitations that this work seeks to overcome. First, the VSF model utilizes a curve from a textbook that maps the degree of SCA defocusing to optical efficiencies [16]. However, optical efficiency as a function of SCA defocusing is unique to each collector-receiver combination. Thus, this work employs SolTrace [19], a Monte Carlo ray tracing algorithm tailored for CSP applications, to develop defocusing curves that are specific to Solana. This approach is fast and easily adaptable to other plants as users simply have to input the geometry of their collectors and receivers.

Next, while VSF is implemented in C++, this work is written in FORTRAN for use in TRNSYS. Both utilize compiled languages which provide significant computational speed. However, by developing the model for TRNSYS, this work can be easily integrated with existing or new power block models also developed in TRNSYS. The modular approach in TRNSYS allows for easy integration for this model with various projects.

Furthermore, unlike VSF which focuses solely on modeling the solar field, this work includes critical components of the HTF network that are often overlooked such as pumps and expansion systems. The expansion system plays a vital role in all PTC CSP plants utilizing thermal oil as their HTF. This is because the HTF level in the expansion tanks determines the pressure throughout the HTF piping network. This pressure is a crucial variable for identifying and preventing costly operational issues. One such issue is pump cavitation which occurs when the pumps are operating at a high speed without sufficient

inlet pressure to the pump. Modeling this phenomenon serves as a valuable training tool for operators and will hopefully reduce operational costs by minimizing the need for pump maintenance.

VSF also utilizes empirical curves for pristine collectors to compute heat loss in the solar field loops. In order to advance the state of the art, this work presents a novel method for quickly computing heat loss of HTF in the solar field with receivers in various conditions. This advancement is particularly relevant for aging operational plants that have a growing percentage of receivers in their solar field no longer in pristine condition. To efficiently compute heat absorption into the HTF with receivers in various conditions, neural networks are utilized to predict Forristall model.

There have been numerous applications of neural networks (NN) for flat-plate solar collectors [20]. For example, for one of the first use cases, a neural network was trained to predict the outlet temperature of a fluid flowing through a flat-plate solar collector as a function of inlet fluid temperature, ambient temperature, and incident radiation [21]. This model was trained using interpolated experimental data and achieved a high degree of accuracy. Additionally, other forms of machine learning has been utilized to reduce the runtime of CSP performance simulations [22] [23]. While [23] utilizes a neural network in the workflow, the methodologies at large utilize data clustering to reduce runtime. This is different from this work in that a neural network is used to replace a computationally expensive energy balance solver.

Lastly, this work introduces a new reinforcement learning algorithm to aid operators during training. Reinforcement learning is a type of machine learning where a model, referred to as an agent, learns to make decisions by interacting with an environment. It aims to maximize cumulative rewards by taking actions based on past experiences. Through trial and error, the agent learns optimal strategies by receiving feedback in the form of rewards or penalties for its actions. Currently, applications of reinforcement

learning in the field of CSP are fairly limited. One of the first applications used a foundational reinforcement learning algorithm to identify optimal operation schedules for a solar hot water system [24]. While this method proved to be successful, it only made control decisions every few hours which is in stark contrast to a CSP plant where operators are making decisions on the order of minutes to seconds. Recently, a relatively more advanced reinforcement learning algorithm was utilized to optimize thermal output of a solar field loop and maintain HTF temperatures within specified bounds by controlling defocusing and flow rates [25]. This algorithm achieved better performance relative to a state of the art model predictive controller.

In reinforcement learning, a "state" refers to the current situation or configuration of the environment in which an agent is operating. For CSP applications, a state could be comprised of HTF mass flow rate, number of collectors defocused, and HTF solar field outlet temperature. In the studies referenced, the state is comprised of measurements made at the current time step in the simulation. This differs from the algorithm used in this work as the state is defined as a history of measurements. A sequence of measurements is particularly useful for plant control as there are often transient delays between correlated measurements. For example, during start up, the temperature of HTF at the solar field outlet is higher than the temperature of the HTF at the central plant. This occurs because it takes the HTF a few minutes to return back to the plant from the solar field. Therefore, providing a history of measurements instead of a snapshot of the present should allow an agent to make more informed decisions.

Furthermore, this work leverages the transformer architecture [26] which has been used to develop the widely popular large language models such as Generate Pre-trained Transformers (GPT) [27] and Bidirectional Encoder Representations from Transformers (BERT) [28]. Using the transformer architecture for reinforcement learning tasks was initially proposed in a work that utilized sequences of state-action-reward pairs from

historical data to obtain an optimal policy [29]. One of the main differences compared to this work is that our agent has the ability to interact with an environment. The architecture used in this work is very similar to the Transformer Deep Q Network [30]. The transformer is particularly powerful because it is exceptional at learning relationships and dependencies between different parts of the input sequence through the use of a self-attention mechanism. The transformer architecture was used in this work with the hope that it can learn and identify dependencies to operators in order to improve their understanding of the plant. While there is active research trying to determine whether the self-attention mechanism is providing an accurate interpretation of the model's understanding of the system [31], it may provide useful information in this context.

3 SOLAR FIELD MODEL METHODOLOGY

This chapter outlines the methodologies utilized to construct the solar field model. The workflow for the solar field model is broken up into two separate components: hydraulics and thermals. A representation of the workflow is presented in Figure 3.1.

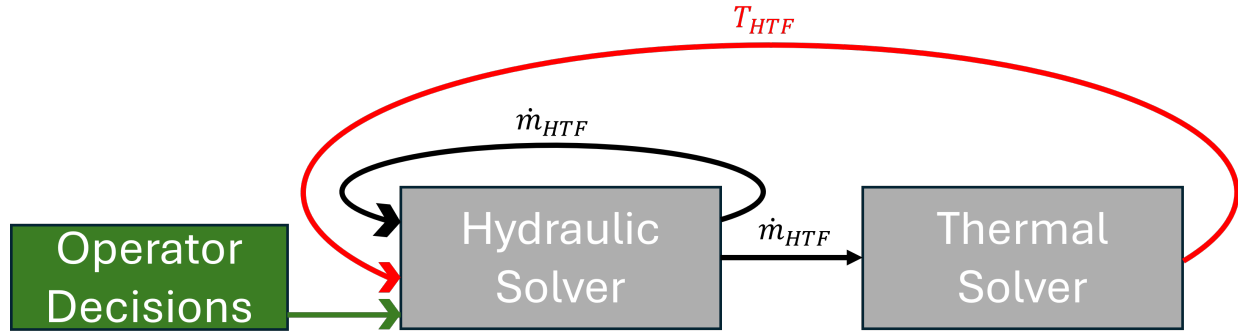


Figure 3.1: Solar field methodology workflow

At each time-step in the simulation, users input control decisions such as HTF valve positions or pump speeds. Next, a hydraulic solver iteratively computes flow rates of HTF throughout the solar field based on current HTF temperatures and operator control decisions. Once the hydraulic solver converges, the mass flow rates and weather conditions are used within a thermal model that computes new temperatures of HTF in the solar field piping network. These temperatures are then updated for the next time-step and the procedure repeats itself. The following sections and subsections outline the details of each solver.

3.1 Hydraulics

Computing HTF flow rates accurately and efficiently is of paramount importance for any relevant parabolic trough CSP plant model. Operators effectively control the plant with HTF flow by altering pump speeds and valve positions. Thus, this model was developed with the intent to respond to operator decisions. While almost none of this methodology

is novel, this section provides details that relate to hydraulic modeling of the solar field and will be relevant for readers attempting to replicate or re-purpose the model of the Solana solar field.

3.1.1 Pumps

Humanity enjoys the luxuries that can come from moving fluids from one place to another. For example, the Romans utilized the power of gravity to move water with the invention of aqueducts. This invention enabled the growth of cities and agriculture while also supporting public baths and sanitation systems that were crucial for public health and urban development. Fast forwarding to present day, whether it be water supply, wastewater treatment, HVAC systems, oil/gas transportation, or various industrial processes, our civilization completely relies on fluids being transported from one location to another. Fluid transportation is facilitated largely through the use of pumps. With regard to CSP plants, centrifugal pumps are used to generate HTF flow through the solar field piping network. An illustration of a centrifugal pump is provided in Figure 3.2.

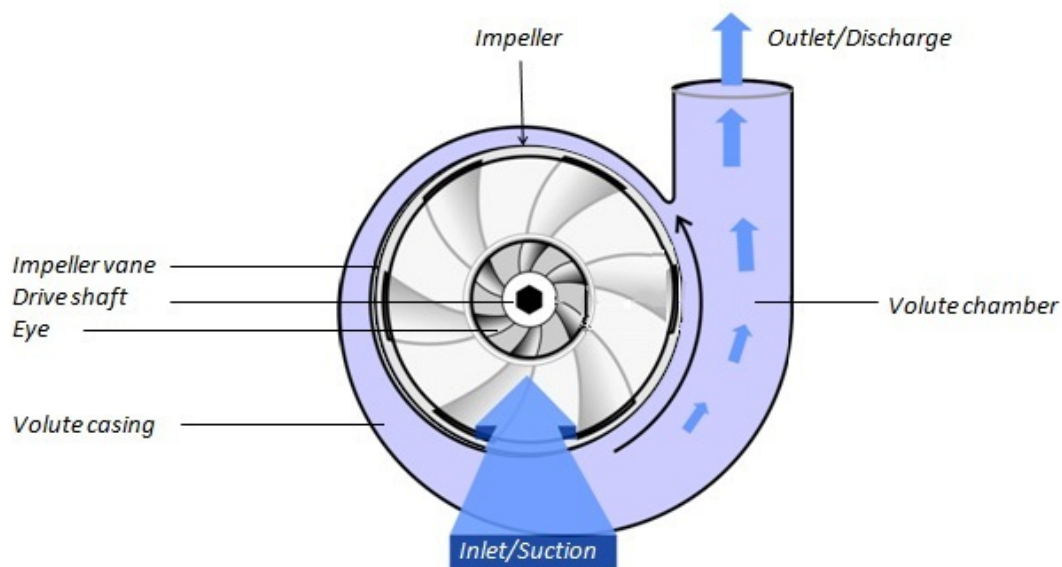


Figure 2. Volute case design

Figure 3.2: Different components of a centrifugal pump [2]

A centrifugal pump draws fluid into its center, known as the eye of the impeller. From there, the rotating impeller imparts kinetic energy to the fluid, causing it to accelerate. This acceleration generates centrifugal force, which pushes the fluid outward towards the edges of the impeller vanes. The fluid gains velocity and pressure energy as it moves from the eye to the edge of the impeller, and it is ultimately discharged through the outlet.

The pumps in the hydraulic solver are modeled using pump curves that correspond to the installed HTF pumps at the Solana plant. Pump performance is typically represented by these curves, which communicate the flow rate a pump can produce given the pressure head differential it must overcome within the piping network. Each plant has pump curves provided by their supplier, so this approach offers the flexibility to easily model various plants. In addition to pump curves, the supplier also provides a net positive suction head required (NPSHr) curve. The NPSHr curve establishes the minimum inlet pressure of the fluid, as a function of flow rate, necessary for the pump to operate without cavitation. Cavitation, a costly consequence of low inlet pump pressure, occurs when pockets of low-pressure fluid near the pump impeller vane vaporize and subsequently implode rapidly. These implosions can cause damage to the pump impeller and lead to a significant decrease in pump efficiency over time. Figure 3.3 presents a representative pump curve (black) and NPSHr curve (red).

While interpolating data from a look-up table is an option, these curves are quadratic and it is simple to fit a quadratic curve to the data. Furthermore, evaluating a quadratic function is much faster than using a lookup table and interpolating between data points. Thus, the NPSHr and pump curve are transformed into a quadratic function as presented in Equations 3.1 and 3.2.

$$H_L = A + B\dot{Q} + C\dot{Q}^2 \quad (3.1)$$

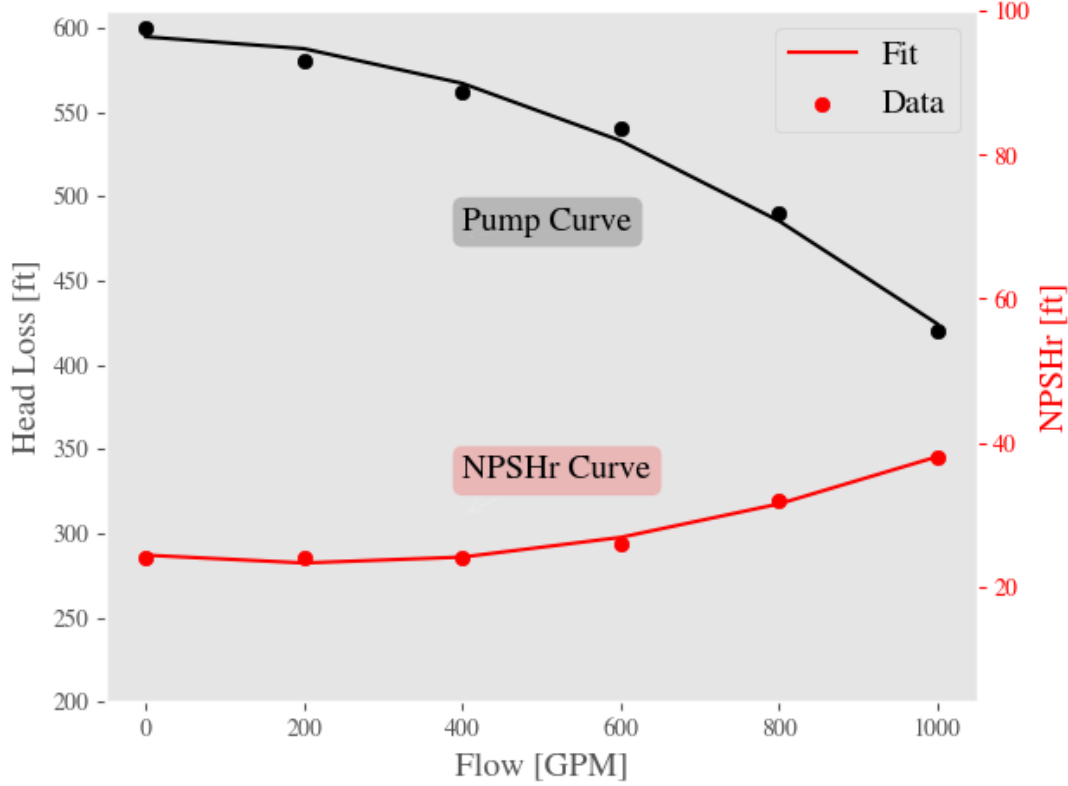


Figure 3.3: Visualization of a fitted pump and NPSHr curve using manufacturer provided data

$$\text{NPSHr} = D + F\dot{Q} + G\dot{Q}^2 \quad (3.2)$$

Here, H_L is the head loss the pump can overcome in units of length, $\dot{Q}$ is the volumetric flow rate, and $[A, B, C, D, F, G]$ are the coefficients obtained from the two curve fits. Given a flow rate through the piping system, equations 3.1 and 3.2 provide an efficient method for evaluating the pressure head a pump can overcome or the NPSHr.

This methodology is and can be utilized to fully capture the behavior of single speed pumps. However, most plants utilize variable speed pumps in order to optimize plant performance under various solar conditions. For example, when the sun is rising or setting, there is less solar radiation incident upon the solar field. To achieve the rated HTF outlet temperatures achieved during mid-day operations, the HTF flow rate must

decrease. To account for these changes in pump speeds, pump affinity laws are utilized [32]. These laws provide a relationship between pump speed and pressure head along with pump speed and flow rate. Specifically, there is a quadratic relationship between pressure head and pump speed while there is a linear relationship between flow rate and pump speed. Substituting these relationships into Equations 3.1 and 3.2 yields Equations 3.3 and 3.4 which communicates head loss and NPSHr as a function of flow rate and pump speed, s .

$$H_L = s^2 A + s B \dot{Q} + C \dot{Q}^2 \quad (3.3)$$

$$\text{NPSHr} = s^2 D + s F \dot{Q} + G \dot{Q}^2 \quad (3.4)$$

The pump speed, s , is defined as the operating angular velocity of the pump (typically measured in rotations per minute) divided by the angular velocity of the pump used during the pump performance testing. Therefore, s typically ranges from 0 to 1, but it is possible to have values that are greater than 1 if the pump is operating at a speed greater than what was used during the performance test. Ultimately, this approach allows the model to compute the pressure head a pump can overcome and the NPSHr as operators change pump speeds to address the needs of the plant.

3.1.2 Pressure Drop

In the HTF piping network, flow is driven by one or more pumps that create a pressure differential between the pump outlet and inlet. Within the network, the HTF interacts with the internal pipe surface causing frictional losses, and the HTF also loses energy due to form losses from fittings, changes in pipe direction, or the presence of valves. This loss in HTF energy then causes a drop in pressure. For a balanced system, the pressure

increase generated by the pump must equal the pressure decrease caused by the HTF interacting with the piping network. Using the methodology in the previous subsection, the pressure increase from the pump can be computed. This subsection will focus on the computation of HTF pressure drop. The methodology used is nearly identical to the physical trough model in SAM [7].

There are three components of the HTF network that require pressure drop computation. The first is pressure drop through a pipe due to frictional losses. This pressure drop is calculated using physical properties of the fluid, Reynolds number, and friction factor.

$$\Delta P_{\text{pipe}} = \frac{f \rho L_{\text{pipe}} v_{\text{ave}}^2}{2 D_{\text{pipe}}} \quad (3.5)$$

This pressure drop computation is dependent upon the diameter of the pipe (D_{pipe}), average velocity of the fluid through the pipe (v_{ave}), pipe length (L_{pipe}), density of the fluid (ρ), and friction factor (f). The friction factor is a function of Reynolds number (Re) and the relative roughness of the internal pipe surface. For laminar flow ($Re < 2750$), the friction factor is easily computed as $\frac{64}{Re}$. However, when the flow is turbulent (as it often is in the solar field), the calculation of the friction factor requires an iterative approach. First, a new variable, X , is introduced.

$$f = \frac{1}{X^2} \quad (3.6)$$

This substitution is especially useful for the implicit Colebrook friction factor formulation. This equation can be solved iteratively using Equation 3.7.

$$F_x = X + 2 \log_{10} \left[\frac{\text{Roughness}}{3.7} + 2.51 \frac{X}{Re} \right] \quad (3.7)$$

A solution to the Colebrook equation is achieved when F_x equals zero. Therefore, an iterative approach is utilized to determine a value of X that sets Equation 3.7 to zero. This value of X can then be used to compute the friction factor using Equation 3.6. The iterative approach utilizes the False Position method. At a high level, the algorithm identifies values of F_x that are greater than and less than zero. Then a linear line is fit to those points and the value of X that intersects with $F_x = 0$ is used. The process then repeats itself until convergence is achieved. A visualization of this algorithm is presented in Figure 3.4.

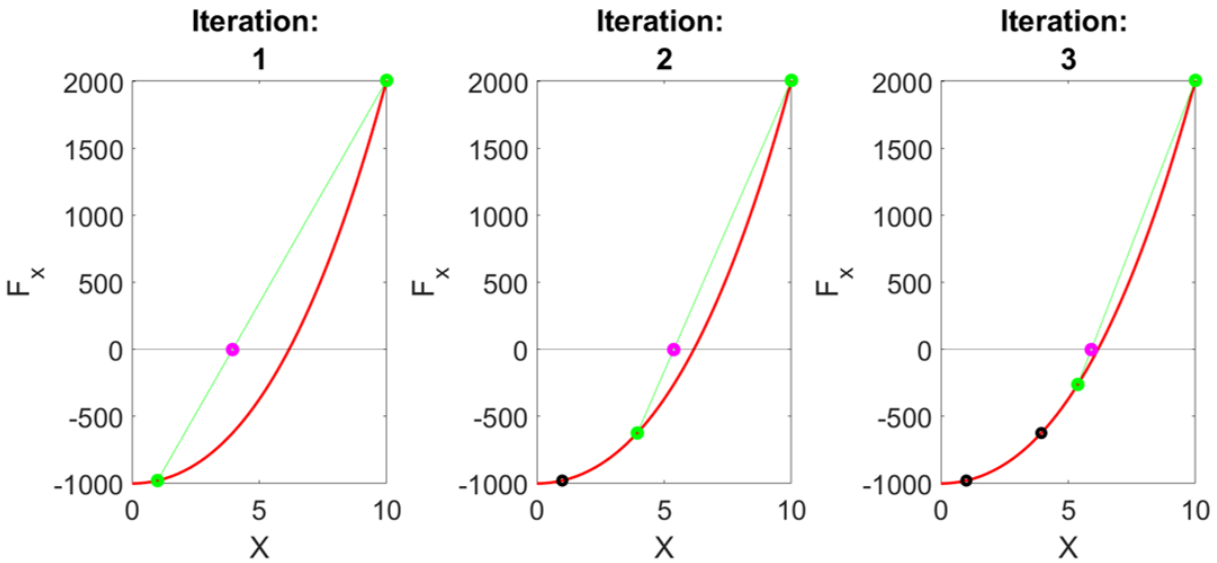


Figure 3.4: False Position method utilized to solve Colebrook friction factor formulation

Although only the first three iterations are presented, one can imagine how this approach will eventually converge to the solution of X that solves for $F_x = 0$. One can speed up the computation time of this algorithm by using guess values near the solution.

The next pressure drop in the HTF network is due to form losses, such as fittings, elbows, and valves. Each type of form loss is characterized using a loss coefficient, $k_{\Delta P}$. The values for these loss coefficients, presented in Table 3.1, are identical to those used in the physical trough model [7].

The calculation used to compute pressure drops across the various piping components listed in Table 3.1 is the same. The formulation used is Equation 3.8.

Table 3.1: Loss coefficients for various piping components

Component	$k_{\Delta P}$
Expansion	0.50
Contraction	0.50
Standard Elbow	0.90
Medium Elbow	0.75
Long Elbow	0.60
Gate Valve	0.19
Globe Valve	10
Check Valve	2.5
Loop Weldolet	1.8
Loop Control Valve	10
Ball Joint Assembly	8.69

$$\Delta P_i = \frac{1}{2} k_{\Delta P, i} \rho v_{ave}^2 N_i \quad (3.8)$$

This equation computes the pressure drop across the i^{th} piping component. The only new variable here is N_i which represents the number of the i^{th} component in the piping section of interest. Therefore, the model keeps track of how many of each piping components are within each piping segment.

The last component to discuss is pressure drops due to valves in various positions. To implement this, the model used C_v values. C_v values, also known as flow coefficients, are numerical values used to characterize the flow capacity of valves and other fluid control devices. C_v values are determined experimentally by conducting flow tests on the valve under controlled conditions, where the flow rate of water (or another specified fluid) is measured at various valve openings and pressure differentials. The relationship between C_v and pressure drop for any fluid is provided in Equation 3.9.

$$\Delta P_{valve} = \frac{(SG)\dot{Q}^2}{C_v^2} \quad (3.9)$$

$\dot{Q}$ is volumetric flow rate in units of gallons per minute, and SG is the specific gravity of the fluid. Note, units are important in this calculation because C_v values are obtained from experimental measurements using English units. Therefore, unit conversions are necessary if one is not using English units.

To further the discussion of valve positions, consider Figure 3.5 that presents the C_v values for a concentric butterfly valve as a function of valve opening.

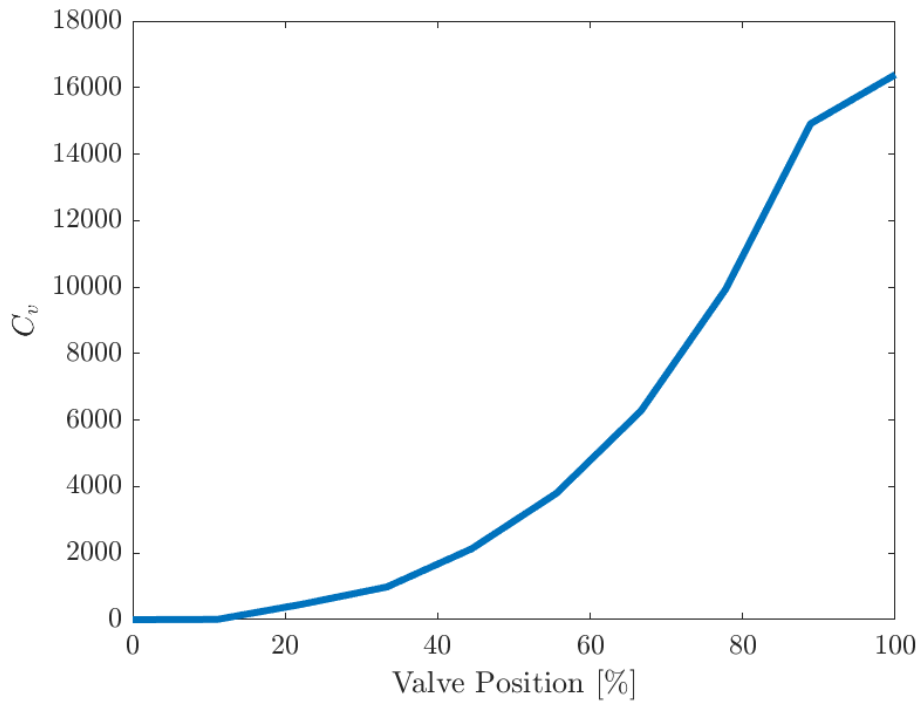


Figure 3.5: C_v values as a function of valve position for a butterfly valve

A valve is fully open when "Valve Position" is at 100% and fully closed at 0%. As illustrated in Figure 3.5, the C_v value increases with valve position. Because C_v is in the denominator in Equation 3.9, a larger C_v value corresponds to a smaller ΔP_{valve} . On the other hand, as the valve closes, it approaches a C_v value of zero resulting in ΔP_{valve} approaching infinity. For modeling purposes, a C_v value of zero is never used as this would result in a division by zero. Therefore, whenever a valve is closed in a model, the C_v value is set to a very small number which results in effectively zero flow traveling

through the valve.

Using the pressure drop Equations 3.5, 3.8, and 3.9, pressure drops can be computed across the entire HTF piping network. These equations are implemented in the TRNSYS types that are used to construct the solar field model. For example, Figure 3.6 presents a screenshot of a portion of the solar field model developed with TRNSYS. The model illustrates various pipe, valve, and solar field sectors Types that are connected.

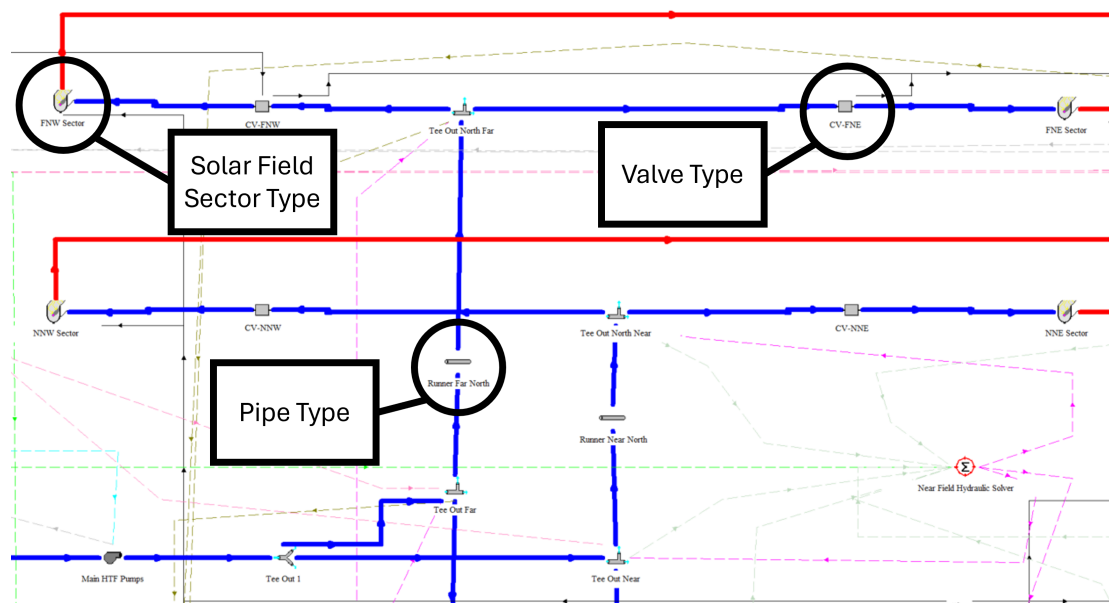


Figure 3.6: Section of TRNSYS solar field model with various types highlighted

The pressure drop across a pipe type is computed using Equation 3.15 (friction losses) and Equation 3.8 (form losses). For each "Pipe Type", the user must specify the piping geometry and the number of form loss components. As an example, a runner pipe from the central plant to the solar field sector often has numerous expansion loops to alleviate thermal stresses due to heating and cooling of HTF. Therefore, for a pipe type representing a runner pipe, the user would specify that there are N standard elbows according to how many expansion loops are in that section of piping. Next, the pressure drop across a valve type is computed using C_v values with Equation 3.9. C_v values are unique for various valve sizes and valve types. Because of this, the user must also specify relevant geometric

parameters along with the type of the valve, such as butterfly, gate, or globe. Lastly, regarding the solar field sectors, a simplified approach is utilized to compute pressure drops. Each sector has a cold header (hdr, c), hot header (hdr, h), inlet/outlet/cross-over piping (IOCop), and solar field loop (Loop). The different sections of a solar field sector are highlighted in Figure 3.7.

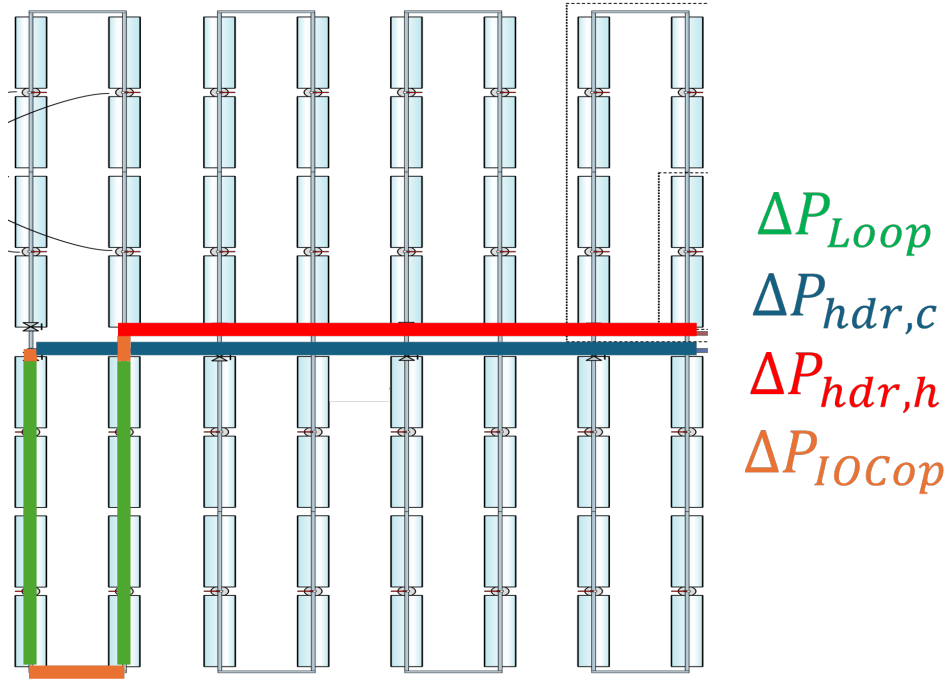


Figure 3.7: Components of solar field sector used in pressure drop calculation

For each of these different sections of piping, the pressure drop is calculated in the same manner that pressure drops are computed for the pipe types. With the pressure drop across each section determined, the total pressure drop across a solar field sector is computed with Equation 3.10.

$$\Delta P_{sector} = \Delta P_{Loop} + \Delta P_{hdr,c} + \Delta P_{hdr,h} + \Delta P_{IOCOp} \quad (3.10)$$

One of the key assumptions in this framework is that HTF flow into a solar field sector is equally distributed across all the loops in the sector. This decision was driven by

the fact that, at most large-scale CSP plants, operators do not have the ability to control individual solar field loop control valves. Operators can only adjust flow rates to the various sectors by using sector control valves, so computing flow rates of each loop would not contribute to operator's training. Furthermore, by making this flow rate assumption, the runtime of the hydraulic solver is significantly reduced. Lastly, although flow is assumed to be equally distributed for pressure drop calculations, there is a prescribed mass flow variation in each loop used for the thermal model. This mass flow variation was a result of feedback from industry partners that indicated loop control valves are set annually such that loop flow rates are within $\pm X\%$ of the average flow rate. Thus, users have the ability to define $X\%$ based on their confidence in their loop control valve positions used to equally distribute flow.

3.1.3 Mass Flow Solver

Mass flow rates are computed by coupling the pressure drop model with the pump model. This work utilizes an iterative method based on the Linear Theory Method [33] to solve for HTF flow rates through a piping network. At its core, the mass flow solver computes HTF mass flow rates while enforcing the following:

- mass flow entering a node equals the mass flow exiting (conservation of mass)
- the total pressure loss experienced by the fluid exactly equals that provided by the pump (conservation of energy)
- the pressures of two fluids converging at a tee are equal

The solved flow rates are entirely dependent on the thermal state of the HTF throughout the plant and operator decisions such as pump speeds and valve positions. To better understand the methodology, consider the system in Figure 3.8 representing a simplified piping schematic of an HTF network.

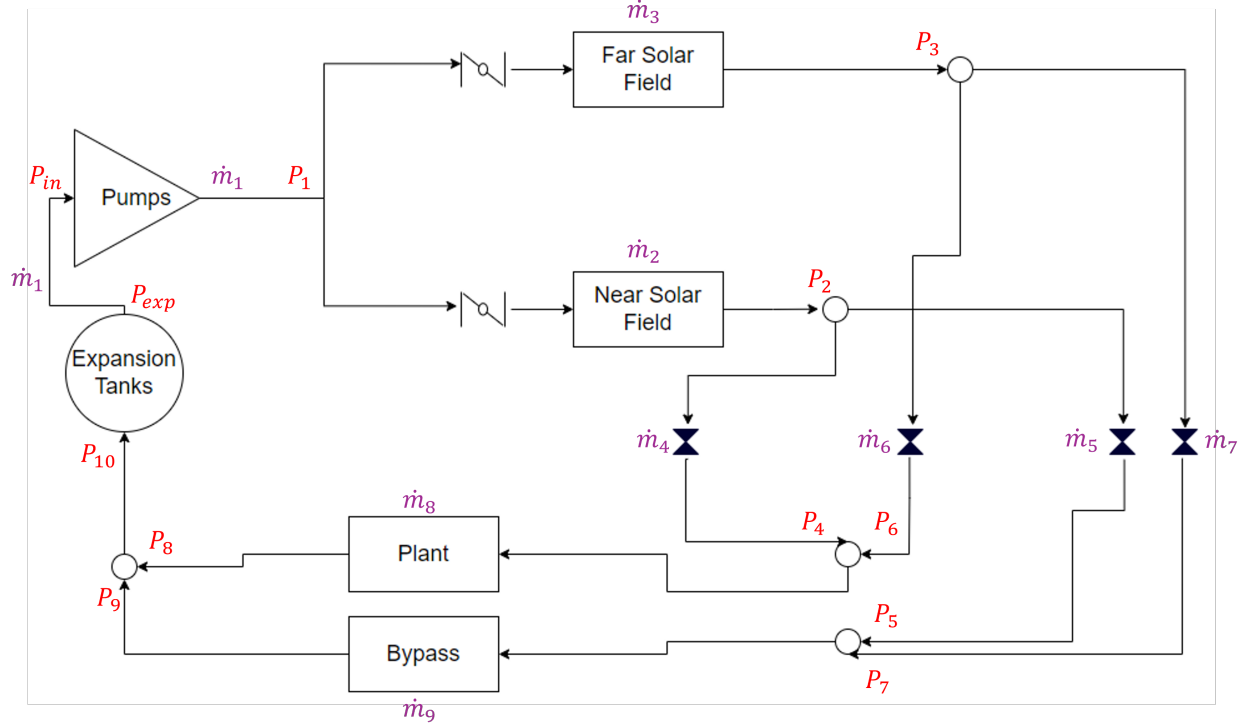


Figure 3.8: Hydraulic system solved using Linear Theory Method

This system has all the necessary components of a solar field HTF network. Flow is driven by main HTF pumps, and it can be diverted through near or far solar field sectors to collect heat before returning back to the central plant. Here, HTF can either be sent to the plant to generate steam or through a bypass if the HTF is not hot enough. Lastly, as will be discussed at length in Subsection 3.1.4, the expansion tanks set the minimum pressure of the HTF.

In this system, the 9 unknown variables are the 9 unknown mass flow rates indicated with purple text. Thus, 9 equations are needed to find a unique solution for this system. To begin, continuity equations are used to ensure the conservation of mass. In this system, there are 5 continuity equations. Equation 3.11 presents the first of these that ensures the mass flow into a node equals the mass flow out of it.

$$-\dot{m}_1 + \dot{m}_2 + \dot{m}_3 = 0 \quad (3.11)$$

Next, there are 3 equations that relate to pressure constraints. These equations enforce that the pressure of two fluids converging at a tee are equal. For example, Equation 3.12 ensures that the pressure of the fluids converging from line 4 and line 6 are equal.

$$P_4 - P_6 = 0 \quad (3.12)$$

Lastly, to enforce the conservation of energy, the total pressure loss experience by the fluid must exactly equal that provided by the pump. In the model, the pump provides energy by increasing the pressure and the piping network removes energy through various pressure drop computations. To ensure that the energy added equals the energy lost, P_{10} must be equal to P_{exp} . Note, this is only true because P_{exp} , the pressure set by the expansion tanks, serves as the reference pressure for the system and is unchanging within each time step. Therefore, when P_{10} is equal to P_{exp} , the model has computed a set of mass flow rates that results in a pressure drop throughout the system that is equal to the pressure increase generated by the pumps.

Here, 9 equations are now available to solve for the 9 unknown mass flow rates. To understand the challenge of solving flow rates, it is first important to define the computation used for pressure drops. This computation is provided in Equation 3.13 which can be used to determine the pressure of the HTF at the tee indicated by P_3 .

$$P_3 = P_1 - \Delta P_3 = P_1 - K_3 \dot{m}_3^2 \quad (3.13)$$

K_3 is the loss coefficient for the piping line associated with $\dot{m}_3$. This coefficient captures pressure drop effects due to frictional and form losses. However, due to the dependence of friction factor on Reynolds number, the loss coefficient is also a function of mass flow rate. Hence, a challenge emerges from the pressure constraints as these equations are

non-linear and implicit in the sense that the loss coefficients are a function of mass flow rate.

Before addressing these challenges, the pressure constraint equations must be formally defined. Using the longest constraint as the example, Equation 3.14 ensures that P_{10} is equal to P_{exp} .

$$P_{10} = P_{exp} - \Delta P_1 + \Delta P_{pump} - \Delta P_2 - \Delta P_4 - \Delta P_8 - \Delta P_{10} = P_{exp} \quad (3.14)$$

There are a few items worth lingering on here. As the solver converges to a solution, the pressures of HTF converging at a tee throughout the network will not be equal. For example, P_4 will not equal P_6 in many of the iterations before a solution is achieved. To ensure convergence, it is incredibly important that the model used to compute pressure drops sends through the pressure used in the constraint equations. For example, consider the tee where $\dot{m}_4$ and $\dot{m}_6$ combine. With the definition for P_{10} in Equation 3.14, it is crucial the the value of P_4 is set as the pressure of the tee instead of P_6 or some average of the two of them. This must also be applied for the tee where $\dot{m}_8$ and $\dot{m}_9$ combine. Failure to do so will result in the methodology not converging to the correct solution. Second, there is a new term, ΔP_{pump} , which is the pressure provided to the system from the pump. Recall from Section 3.1.1 that pump curves are utilized to communicate the pressure head a pump can provide as a function of flow rate. Thus, Equation 3.3 is rearranged into a new form, Equation 3.15, that computes the pressure increase a pump can provide as a function of mass flow.

$$\Delta P_{pump} = gs^2(\rho A + B \frac{\dot{m}_1}{sN_{pump}} + C \frac{\dot{m}_1^2}{\rho^2 s^2}) \quad (3.15)$$

New variables in this equation are gravitational acceleration, g , and the number of

pumps in parallel, N_{pump} .

With the ability to define all 9 equations, an iterative approach is now be used to solve for the unknown mass flow rates. During the first iteration, a guess for the mass flow rates is provided to the system. The detailed TRNSYS model then computes pressure drops according to valve positions and flow rates through each line. The pressure drops propagated through the piping network are those defined in the pressure constraint equations. Once all pressure drops are calculated, the loss coefficients can be approximated. For example, the loss coefficient for the piping line associated with $\dot{m}_3$ is presented in Equation 3.16.

$$K_{3,0} = \frac{P_{1,0} - P_{3,0}}{\dot{m}_{3,0}^2} \quad (3.16)$$

Note that there is a new subscript on all the terms. This secondary subscript indicates the current iteration with indexing beginning at 0. More generally, $K_{i,j}$ represents the loss coefficient associated with the i^{th} piping line computed using mass flow rates from the j^{th} iteration. All of the relevant loss coefficients for this system are computed in the model using the code snippet below. Note, the mass flow and pressure values are loaded in from the previous iteration.

Listing 3.1: Computation of loss coefficients

```

1 ! Compute K values based on m_dots and pressures from previous iteration
2 K2 = abs((P_1 - P_2)/(m_dot_2)**2)
3 K3 = abs((P_1 - P_3)/(m_dot_3)**2)
4 K4 = abs((P_2 - P_4)/(m_dot_4)**2)
5 K5 = abs((P_2 - P_5)/(m_dot_5)**2)
6 K6 = abs((P_3 - P_6)/(m_dot_6)**2)
7 K7 = abs((P_3 - P_7)/(m_dot_7)**2)
8 K8 = abs((P_4 - P_8)/(m_dot_8)**2)
9 K9 = abs((P_7 - P_9)/(m_dot_9)**2)
10 K_fin = abs((P_8 - P_10)/(m_dot_1)**2)

```

```
11 K_in = abs((P_exp - P_in)/(m_dot_1)**2)
```

Next, to linearize the pressure constraint equations, the Linear Theory Method is utilized. Using the pressure drop across line 3 as an example, Equation 3.17 presents the new approach.

$$\Delta P_3 = (K_{3,j} \dot{m}_{3,j}) \hat{m}_{3,j+1} \quad (3.17)$$

Ultimately, with this new linear relationship between mass flow and pressure drop, a complete linear system can be defined. Equation 3.18 presents the matrix representing the linear system for the piping network illustrated in Figure 3.8.

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 \\ 0 & -K_2 \dot{m}_2 & K_3 \dot{m}_3 & -K_4 \dot{m}_4 & 0 & K_6 \dot{m}_6 & 0 & 0 & 0 & 0 \\ 0 & -K_2 \dot{m}_2 & K_3 \dot{m}_3 & 0 & -K_5 \dot{m}_5 & 0 & K_7 \dot{m}_7 & 0 & 0 & 0 \\ 0 & K_2 \dot{m}_2 & -K_3 \dot{m}_3 & K_4 \dot{m}_4 & 0 & 0 & -K_7 \dot{m}_7 & K_8 \dot{m}_8 & -K_9 \dot{m}_9 & 0 \\ \dot{m}_1 (-K_1 - K_{10} + \frac{gC}{\rho N_{pump}^2}) + \frac{gsB}{n_p} & -K_2 \dot{m}_2 & 0 & -K_4 \dot{m}_4 & 0 & 0 & 0 & 0 & -K_8 \dot{m}_8 & 0 \end{bmatrix} \quad (3.18)$$

For sizing limitations, the second subscript indicating iteration number has been removed. Assume that all loss coefficients and mass flow rates in $\mathbf{A}$ are from the previous iteration. This matrix is built up in FORTRAN using the following code. Note, only one of the continuity equations and two of the pressure constraints are presented because they are fairly repetitive.

Listing 3.2: Define system of equations by defining values in A matrix

```

1 !! Update A matrix
2 !!!!!!!!!!!!!!!!!!!!!!!
3 ! Enforce continuity in A
4 A(:, :) = 0.d0
5
6 A(1, 1) = -1.d0
7 A(1, 2) = 1.d0
8 A(1, 3) = 1.d0
9
10
11 ! Enforce Pressure Drop Relations
12 rho = Density_SF(Fluid_ID, T_1, 0.d0)
13 k1_tot = -K_in - K_fin + g/rho/n_pumps**2*coef_c
14 K1_hat = k1_tot*abs(m_dot_1) + g*coef_b*s/n_pumps
15 A(9, 1) = K1_hat
16
17
18 k2_hat = K2*abs(m_dot_2)
19 A(6, 2) = -k2_hat
20 A(7, 2) = -k2_hat
21 A(8, 2) = k2_hat
22 A(9, 2) = -k2_hat

```

With the matrix **A** defined, the completed linear system can be constructed. Equation 3.19 presents the complete linear system.

$$\mathbf{A} \begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \\ \hat{m}_3 \\ \hat{m}_4 \\ \hat{m}_5 \\ \hat{m}_6 \\ \hat{m}_7 \\ \hat{m}_8 \\ \hat{m}_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -gs^2\rho A \end{bmatrix} = \mathbf{b} \quad (3.19)$$

The linear system is then inverted and solved. Specifically, this is implemented in FORTRAN using a Gauss-Jordan elimination adapted from source code provided here [34]. The only difference in the implementation is that the largest *absolute* value on the column i is chosen on Line 24.

Listing 3.3: Code used to solve linear system of equations

```

1 ! Update b
2 b(9) = -g*s**2*rho*coef_a
3
4 ! Invert matrix and compute mass flow rates
5 Call matrixinv(A, A_inv, 9)
6 m_dots = matmul(A_inv, b)
7
8 subroutine matrixinv(a,b,n)
9 integer :: i,j,k,l,m,n,irow
10 double precision:: big,a(n,n),b(n,n),dum
11
12 do i = 1,n !build the identity matrix
13     do j = 1,n
14         b(i,j) = 0.0
15     end do
16     b(i,i) = 1.0

```

```

17 end do
18
19 do i = 1,n ! this is the big loop over all the columns of a(n,n)
20     ! in case the entry a(i,i) is zero, we need to find a good pivot; this
        pivot
21     ! is chosen as the largest value on the column i from a(j,i) with j = 1,n
22     big = abs(a(i,i))
23     do j = i,n
24         if (abs(a(j,i)).gt.big) then
25             big = abs(a(j,i))
26             irow = j
27         endif
28     end do
29     ! interchange lines i with irow for both a() and b() matrices
30     if (big.gt.abs(a(i,i))) then
31         do k = 1,n
32             dum = a(i,k) ! matrix a()
33             a(i,k) = a(irow,k)
34             a(irow,k) = dum
35             dum = b(i,k) ! matrix b()
36             b(i,k) = b(irow,k)
37             b(irow,k) = dum
38         end do
39     end if
40     ! divide all entries in line i from a(i,j) by the value a(i,i);
41     ! same operation for the identity matrix
42     dum = a(i,i)
43     do j = 1,n
44         a(i,j) = a(i,j)/dum
45         b(i,j) = b(i,j)/dum
46     end do
47     ! make zero all entries in the column a(j,i); same operation for indent()
48     do j = i+1,n
49         dum = a(j,i)
50         do k = 1,n
51             a(j,k) = a(j,k) - dum*a(i,k)
52             b(j,k) = b(j,k) - dum*b(i,k)
53         end do
54     end do
55 end do

```

```

56
57 ! subtract appropriate multiple of row j from row j-1
58 do i = 1,n-1
59     do j = i+1,n
60         dum = a(i,j)
61         do l = 1,n
62             a(i,l) = a(i,l)-dum*a(j,l)
63             b(i,l) = b(i,l)-dum*b(j,l)
64         end do
65     end do
66 end do
67
68 end

```

Furthermore, for stability reasons, the mass flow rate used in the next iteration is defined as the average between the previous and the new solved flow rates. This is illustrated in Equation 3.20.

$$\dot{\mathbf{m}}_{j+1} = \frac{\dot{\mathbf{m}}_j + \mathbf{A}^{-1}\mathbf{b}}{2} \quad (3.20)$$

The new mass flow rates are sent through the TRNSYS model, and the process repeats itself until convergence is achieved. Lastly, this methodology needs to also reliably compute flow rates through a network with valves that are closed. When a valve is closed, mass flow rate through that line should be very close to zero. Furthermore, for closed valves, the C_v value is very small. Because of this, there can be large perturbations in the pressure drop due to very small changes in mass flow rate. This yields a significant amount of instability to the solving method. To address this, mass flow rates through lines with closed valves are adjusted between iterations on the log scale.

$$\hat{x}_{i,j} = \frac{\log_{10}(\dot{m}_{i,j}) + \log_{10}(\hat{\dot{m}}_{i,j})}{2} \quad (3.21)$$

$$\dot{m}_{i,j+1} = 10^{\hat{x}_{i,j}} \quad (3.22)$$

This log-based approach dramatically improves stability for the methodology and allows the solver to converge to a solution in fewer iterations. An example of how this is implemented in the model is presented below.

Listing 3.4: Update mass flow rates according to learning rate

```

1 ! Learning rate for update
2 LR = 0.5
3
4 ! total mass flow will never be approaching zero
5 m_dot_1 = m_dots(1)*LR + m_dot_1*(1-LR)
6
7 ! If flow through near field is shut off
8 if(m_dots(2)<0.001)Then
9     m_hold = Log10(m_dots(2))*LR + Log10(m_dot_2)*(1-LR)
10    m_dot_2 = 10**m_hold
11 else
12    m_dot_2 = m_dots(2)*LR + m_dot_2*(1-LR)
13 ENDIF

```

3.1.4 Expansion and Overflow System

The minimum HTF pressure for trough systems is determined with reference to the HTF expansion tanks, and this often-overlooked equipment serves several important roles in model solar field dynamic behavior. Expansion tanks are required due to large temperature and volume variation of the HTF between nighttime and daytime operations. For example, the density of Therminol-VP1 varies from $\rho = 1,000[\frac{\text{kg}}{\text{m}^3}]$ at $T = 100[^\circ\text{C}]$ to $\rho = 756[\frac{\text{kg}}{\text{m}^3}]$ at $T = 350[^\circ\text{C}]$. The volume of fluid in the active flow loop must be constant to avoid pump cavitation, fluid hammer, or loss of heat transfer during startup, so the total mass of fluid in the system significantly exceeds the amount needed during normal

operation when the average temperature of HTF in the system is at its highest. As the HTF heats up during a startup sequence, this excess HTF volume is stored in the tall, pressurized expansion tanks. While design of the expansion system varies to some degree, it is generally true that the expansion tanks dictate the HTF pressure at the inlet of the main HTF pumps where the expansion line is connected to the main flow network.

Most trough plants simply have numerous expansion tanks directly connected to the main HTF piping network. The size and number of tanks is selected such that they can account for the volumetric fluctuations of the HTF. However, because pressurized tanks are expensive, Solana elected to reduce cost by way of an “expansion and overflow system” that is comprised of numerous smaller pressurized expansion tanks and two large overflow tanks that are kept near atmospheric pressure. Note, users have the ability to choose whether the model accounts for just expansion tanks or the ‘expansion and overflow system’ by adjusting a parameter in the TRNSYS type. During a startup as the average HTF temperature rises, operators send accumulating HTF from the expansion tanks to the overflow tanks to avoid an over-filling the expansion tanks. During the solar field startup process when the average HTF temperature is sufficiently high, the power block system begins startup, drawing energy from the HTF and causing the average HTF temperature to decrease. Then, the operators must pump HTF from the overflow tanks back to the expansion tanks to keep the main HTF pumps from cavitating. Pump cavitation is an important aspect of operator training and is a significant source of historical plant unavailability. Operators can control flow to or from the expansion tanks utilizing control valves illustrated in Figure 3.9. This figure and corresponding labeled pipe segments should be referenced for generating the file used to define pipe geometries in the expansion system in TRNSYS.

There are three main modes of operation for the expansion system that are utilized daily. First, during start up as the HTF in the solar field heats up, HTF is sent from the

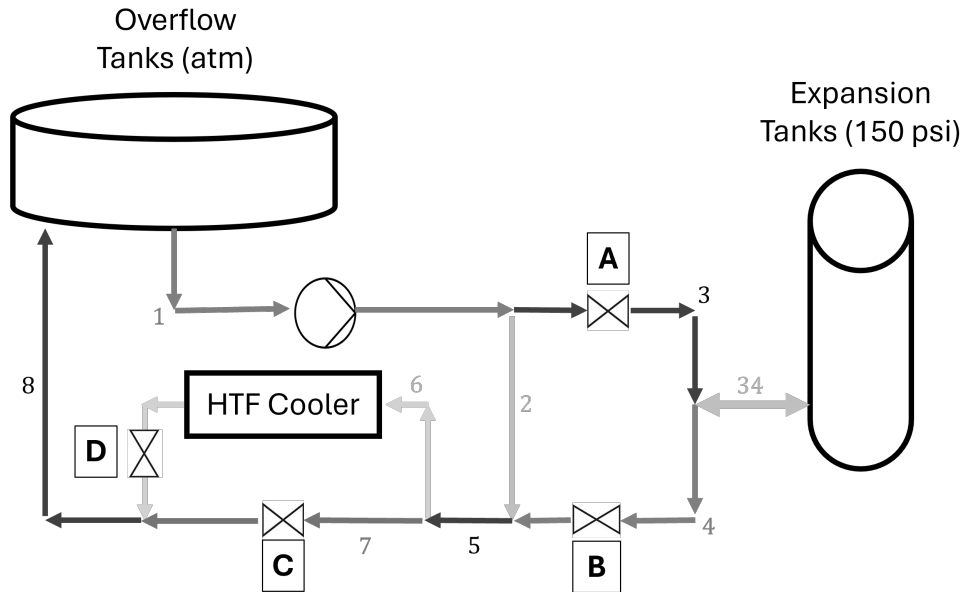


Figure 3.9: Illustration of expansion system comprised of pressurized expansion tanks and overflow tanks

expansion tank to the overflow tank by closing valve A, opening valve B, opening valve C, and closing valve D. Once the plant reaches a pseudo-steady state with constant solar availability, the expansion system transitions into a cooling mode. This HTF cooling is necessary to keep the fluid from vaporizing in the overflow tanks that are not pressurized. For this mode, valves A and B are shut while valves D and C are opened. Lastly, during nighttime operations where the HTF temperature in the solar field dramatically decreases, the level in the expansion tanks becomes extremely low. Thus, the HTF from the overflow tank must be transferred back by opening valve A, closing valve B, opening valve C, and closing valve D. The operating modes are concisely summarized in Table 3.2.

Table 3.2: Operational modes of expansion system

Mode	Valve A	Valve B	Valve C	Valve D	Description
1	Closed	Open	Open	Closed	Expansion to Overflow
2	Closed	Closed	Open/Closed	Open	Cooling HTF
3	Open	Closed	Open	Closed	Overflow to Expansion

The methodology outlined in Section 3.1.3 is repurposed for computing HTF mass flow rates through the expansion system for the various modes of operation. To illustrate

the capabilities of the model, Figure 3.10 presents a representative daily operation.

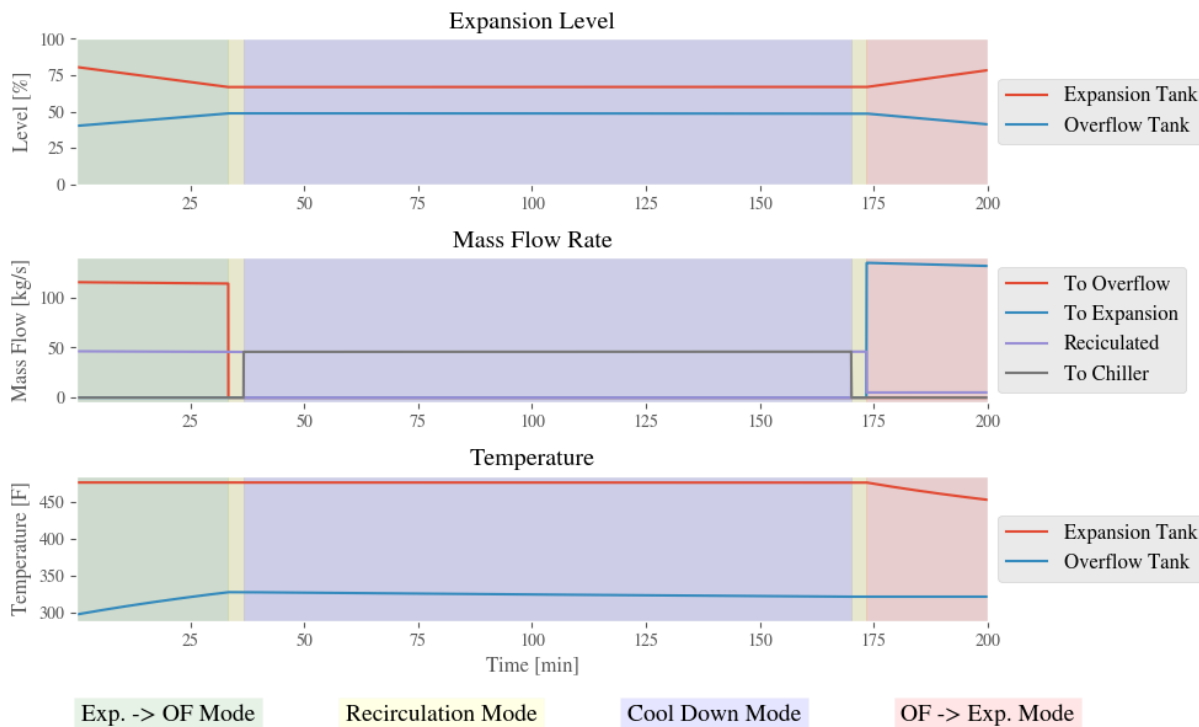


Figure 3.10: Illustration of expansion system comprised of pressurized expansion tanks and overflow tanks

During the first 30 minutes (green), the valves are positioned such that HTF is sent from the expansion tanks to the overflow tanks. This results in the corresponding tank levels to decrease and increase. Furthermore, because hot HTF from the expansion tanks is flowing into the overflow tanks, the temperature of the HTF in the overflow tanks increases. The thermal modeling will be discussed later in Section 3.2.7. In addition to the expansion to overflow mode, the cool down and overflow to expansion modes are also presented in this case study.

Lastly, it is relevant to discuss how the level in the expansion tanks is computed as HTF temperatures throughout the plant change. Before the simulation begins, there are a few parameters that must be defined. These include the initial level in the overflow tank, the initial temperature of the HTF in the overflow tank, and the total mass of HTF in entire plant. Next, at each time step, the mass of HTF in the HTF piping network (excluding the

expansion system) is computed by summing up the mass within each control volume in the model. Formally, the mass in the HTF piping network is computed using Equation 3.23.

$$m_{\text{network}} = \sum_{i=1}^{n_{\text{cv,plant}}} \text{Vol}_i \rho(T_i) \quad (3.23)$$

Vol_i is the volume of the i^{th} control volume, $\rho(T_i)$ is the density of the HTF evaluated at the average temperature of the i^{th} control volume, and $n_{\text{cv,plant}}$ is the number of control volumes in the solar field model (excluding the expansion system). The mass of the HTF in the piping network depends on the temperature of the HTF because changes in temperature affect the fluid's density. For example, as the solar field heats up, the HTF density decreases, and there is less HTF mass in the piping network. Next, the total mass in the expansion tanks is computed using Equation 3.24.

$$m_{\text{expansion}} = m_{\text{total}} - m_{\text{network}} - m_{\text{overflow}} \quad (3.24)$$

Note, the total mass in the overflow tanks, m_{overflow} , is adjusted according to HTF flow rate in the expansion system due to different modes of operation discussed above. Lastly, the level of the expansion tanks is evaluated with Equation 3.25.

$$\text{Level}_{\text{expansion}} = \frac{\frac{m_{\text{expansion}}}{\rho(T_{\text{expansion}})}}{\text{Vol}_{\text{expansion}}} \quad (3.25)$$

The fractional level of the expansion tanks, $\text{Level}_{\text{expansion}}$, is simply the volume of HTF currently occupying the expansion tanks divided by the total volume of the expansion tanks. Lastly, to compute the pressure of the HTF at the expansion tanks outlet (the reference pressure for the entire piping network), the static head due to the level of

HTF is combined with the pressure in the expansion tank.

$$P_{exp} = m_{expansion}g(Level_{expansion}H_{expansion}) + P_{expansion,set} \quad (3.26)$$

$P_{expansion,set}$ is the pressure the expansion tanks are constantly pressurized to, and $H_{expansion}$ is the height of the expansion tanks. Using the pressure computed, P_{exp} , the entire hydraulic model is constrained and defined.

3.2 Thermals

With the hydraulic solver accurately computing HTF mass flow rates, the next and perhaps most interesting modeling phase begins: the thermal modeling of the solar field. The solar field thermal model is essential because it determines the heat input which is effectively the operating currency of the plant. Accurately modeling transient thermal events is crucial and especially relevant for plant procedures that operators require training for such as daily plant start-up. This section will build up the methodology from collector optics and a simple energy balance for a differential collector segment to a fully functional solar field thermal model.

3.2.1 Collector Optics

The optical calculations presented herein are based on SAM's physical trough model [7], and the reader is encouraged to reference it for a detailed optics discussion to supplement information presented in this section. The analysis of collector optics presented here considers the collector itself and does not deal with optical properties of the receiver. Therefore, these calculations are relevant for determining the solar flux incident upon the receiver.

In an ideal scenario, solar irradiation would always be normal to the plane of the collectors. This would maximize the amount of solar irradiation incident upon the receivers, and solar fields would absorb more heat. However, this ideal is rarely achieved in reality. As the sun traverses through the sky, the position almost never perfectly aligned with the collectors. Typically, collectors are aligned in a north/south configuration, so they can rotate about the north/south axis to track the sun as it moves from east to west diurnally (in the northern hemisphere). This concept is illustrated in Figure 3.11 where the collector can adjust ω_{col} , the tracking angle, as the solar position changes. Note, please reference [7] for a discussion and equations used for solar position computations.

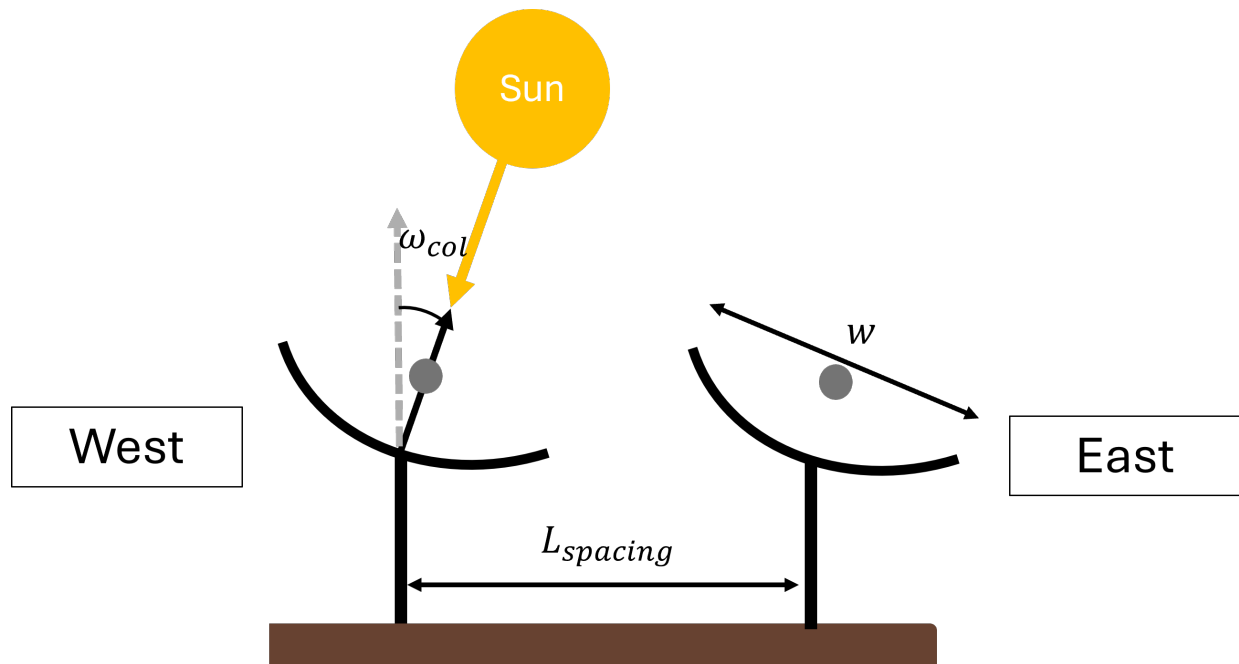


Figure 3.11: Solar tracking to maximize solar irradiation upon receiver as sun traverses through the sky

In the plane presented in 3.11, the solar irradiation unit vector is directly aligned with the unit vector normal to the collector. As the sun moves through the sky, ω_{col} is adjusted to maintain this condition. However, during dusk or dawn, shadowing becomes an issue. Although Figure 3.11 is not drawn to scale, one can imagine that if the sun was just rising in the east that the first collector would shade the other. To compute the optical efficiency due to shadowing, η_{shadow} , Equation 3.27 is used.

$$\eta_{\text{shadow}} = |\cos(\omega_{\text{col}})| \frac{L_{\text{spacing}}}{w} \quad (3.27)$$

The new variables introduced in this equation are L_{spacing} , the distance between rows of collectors, and w , the width of the collector. Both variables are also illustrated in Figure 3.11. This efficiency is calculated for each SCA using the FORTRAN function below.

Listing 3.5: Function used to compute shadowing efficiency

```

1 function Row_shadow(phi, row_distance, w_ap) result(eta_row)
2   double precision :: phi, row_distance, w_ap, eta_row
3
4   ! Compute shadowing efficiency
5   eta_row = Abs(cos(phi))*row_distance / w_ap
6
7   ! Ensure that efficiency is bound between 0 and 1
8   if (eta_row > 1.0) then
9     eta_row = 1.d0
10  else if (eta_row < 0.d0) then
11    eta_row = 0.d0
12  end if
13
14 end function

```

The sun also moves north and south throughout the day. Because the collectors are tracking sun movement in the east/west direction, they are unable to track the movement in the north/south direction. The angle that represents the difference between the solar irradiation vector and the aperture normal vector of the collector is called the incidence angle, θ . This angle is illustrated in Figure 3.12.

The incidence angle, θ , is relevant for several reasons. First, the solar irradiation incident upon a solar field is scaled by the cosine of θ . This is known as the "cosine loss" and it contributes significantly to losses in the solar field. Second, optical losses due to specific receiver-collector combinations are computed using the incident angle modifier,

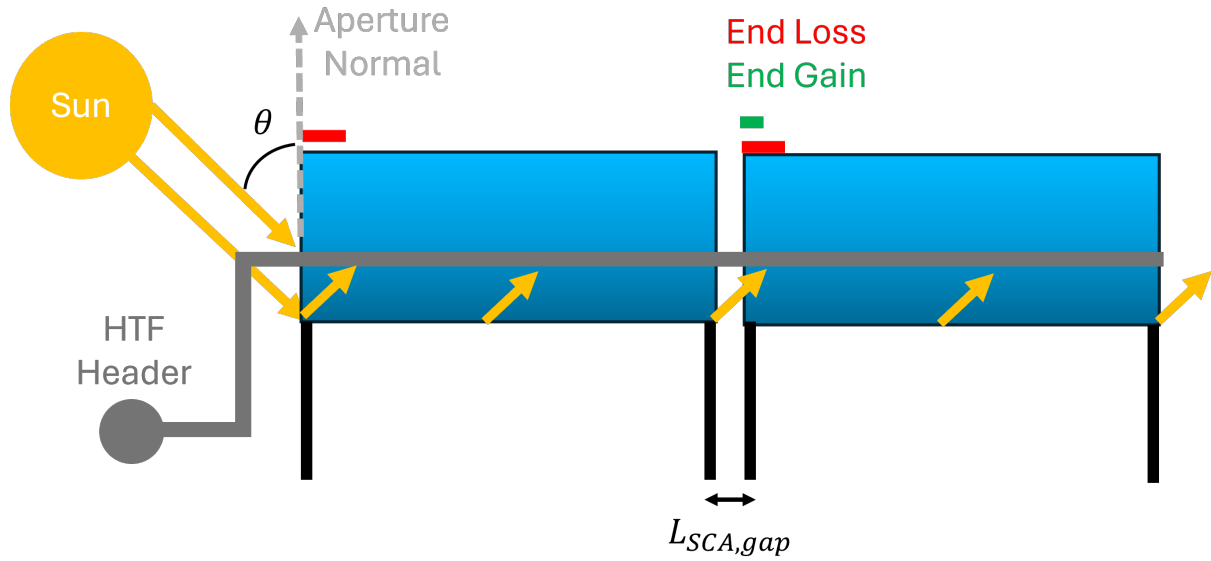


Figure 3.12: Visualization of

η_{IAM} . This efficiency is calculated using Equation 3.28 where the various coefficients are obtained from physical testing of the collectors.

$$\eta_{IAM} = a_0 + a_1 \frac{\theta}{\cos(\theta)} + a_2 \frac{\theta^2}{\cos(\theta)} \quad (3.28)$$

When using this equation, it is critical that θ is in units of radians. Third, when the incidence angle is not zero, some radiation that is reflected off the end of the collector doesn't reach the receiver. This is illustrated in Figure 3.12 where there is a portion of each collector, indicated by the red bar, where the solar irradiation would not hit the receiver. This phenomena is referred to as "End Loss". However, if there is an adjacent collector, this adjacent collector can partially recover some of the solar irradiation reflected from the previous collector, known as "End Gain". This is illustrated in Figure 3.12 with the green bar. While these values are often quite small, Equations are used to compute end losses and end gains respectively for each SCA in a solar field loop.

For $i = 1, N_{SCA}$

$$\eta_{\text{endLoss},i} = 1 - \frac{L_{f,\text{ave}} \tan(\theta)}{L_{SCA}} + \eta_{\text{endGain},i} \quad (3.29)$$

Where:

$$\eta_{\text{endLoss},i} = \frac{L_{f,\text{ave}} \tan(\theta) - L_{SCA,\text{gap}}}{L_{SCA}} \quad (3.30)$$

L_{SCA} is the length of each solar collector assembly, $L_{SCA,\text{gap}}$ is the distance between SCA's as illustrated in Figure 3.12, and $L_{f,\text{ave}}$ is the average surface-to-focus path length. This surface-to-focus path length is not the focal length of the parabola at the vertex. Using the focal length, a , along with other geometric properties of the collector, the path length can be computed.

$$L_{f,\text{ave}} = \sqrt{\frac{(4a^2 + (\frac{w}{2})^2)^2}{a^2}} \cdot \frac{12a^2 + (\frac{w}{2})^2}{12(4a^2 + (\frac{w}{2})^2)} \quad (3.31)$$

Recall w is the width of the collector as illustrated in Figure 3.11. The last important item to note regarding end losses is that $\eta_{\text{endGain},i}$ is zero for some SCA's. For example, in Figure 3.12, the first collector does not have an adjacent collector to absorb reflected heat from. Thus, given these solar conditions, $\eta_{\text{endGain},1}$ is equal to zero and $\eta_{\text{endGain},2}$ is greater than zero. SCA's with $\eta_{\text{endGain},i}$ are defined based upon the orientation of the solar field loop along with the sun's position. For Solana, the plant is located in the northern hemisphere, so "End Gains" are applied to SCA 1 and 4 for loops that are pointed south while SCA 3 and 4 have "End Gains" for loops that are pointed north. The end losses are computed for each SCA of each loop using the code below.

Listing 3.6: Code used to compute end losses for the 4 SCA's in a loop

```
1 ! End Loss Efficiency
2 L_SCA = L_tot/n_SCA
```

```

3 EndGain_pred = Max(Ave_focal_length * tan(theta) - Distance_SCA, 0.d0) / L_SCA
4 EndLoss_pred = 1.d0 - Ave_focal_length * tan(theta) / L_SCA
5 if(MOD(loop, 2) == 0) then ! Loops oriented north
6     eta_endLoss(1) = EndLoss_pred
7     eta_endLoss(2) = EndLoss_pred + EndGain_pred
8     eta_endLoss(3) = EndLoss_pred + EndGain_pred
9     eta_endLoss(4) = EndLoss_pred
10
11 else ! Loops oriented south
12     eta_endLoss(1) = EndLoss_pred + EndGain_pred
13     eta_endLoss(2) = EndLoss_pred
14     eta_endLoss(3) = EndLoss_pred
15     eta_endLoss(4) = EndLoss_pred + EndGain_pred
16 endif

```

In addition to the optical losses due to solar position, there are also constant losses due to collector manufacturing and materials. These losses are tracking error (η_{tracking}), mirror reflectance (ρ_m), and mirror soiling (η_{soil}). Using these values, a general optical efficiency is generated.

Listing 3.7: Baseline optical efficiency

```

1 ! User defined optical errors
2 eta_tot = eta_tracking*eta_soiling*eta_reflectance

```

Lastly, during certain periods of operation, HTF in the solar field can over heat. To limit the overheating of the HTF, SCA's are defocused to reduce the heat solar irradiation upon the receivers. With reference to Figure 3.11, ω_{col} is adjusted to $\omega_{\text{col}} + \omega_{\text{defocus}}$ such that the unit vector normal to the plane of the collector and the unit vector of the solar irradiation are no longer aligned. As ω_{defocus} increases, the efficiency of the collector, η_{SCA} , decreases. η_{SCA} is often referred to as an intercept factor, and it represents the percentage of irradiation incident upon a pristine collector that should reach the receiver. These intercept factors are unique to collector geometries and magnitude of defocusing, ω_{defocus} . To compute the intercept factors, SolTrace is used [19]. SolTrace

is a Monte-Carlo ray tracing software built specifically for CSP applications. There are many important factors to consider when utilizing this tool such as sun position, ray angular distributions, optical assumptions, and convergence requirements which will be discussed.

First, in SolTrace, adjusting the orientation of a parabolic trough mirror and its associated absorber directly can be challenging. Instead of rotating these components to explore different defocusing angles, a python script that calls a SolTrace API modifies the apparent position of the sun. By altering the sun's position in the model, the simulation effectively evaluates the collector performance at different defocusing angles. This is illustrated in Figure 3.13.

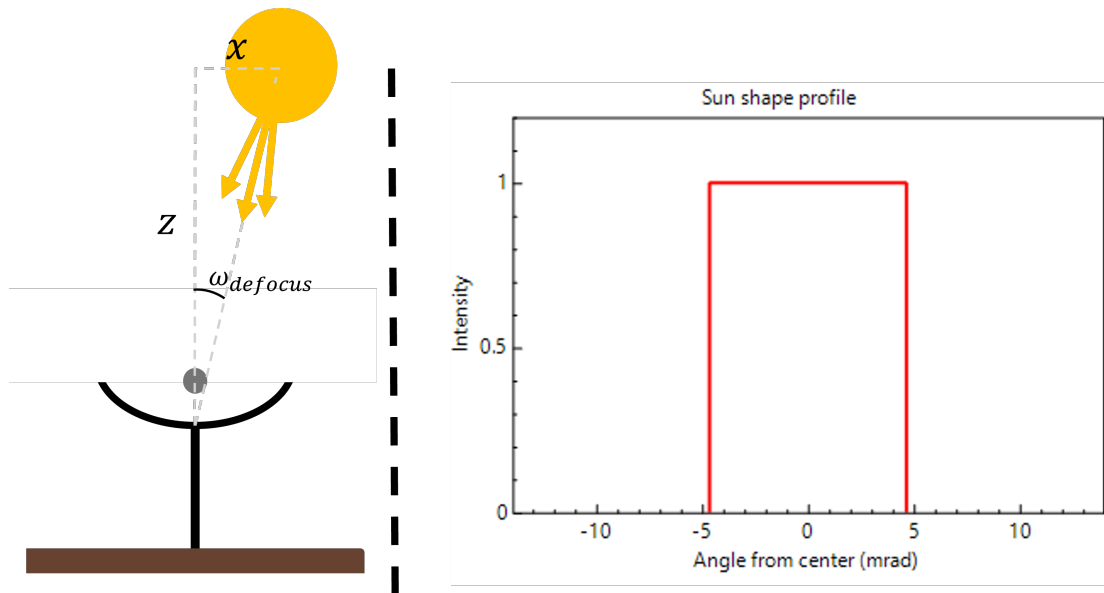


Figure 3.13: Visualization of sun definition and distribution of angular positions used

There are a few important items to note. By assuming that the defocusing angle, ω_{defocus} , is small, the small angle approximation can be applied. This approximation states that $\tan(\omega_{\text{defocus}}) \approx \omega_{\text{defocus}} = \frac{x}{z}$. While maintaining a constant z , the value of x is adjusted to modify ω_{defocus} .

Additionally, it is important to note that the sun's emission is not perfectly uniform; the

rays do not all travel in exactly the same direction. To account for this variation in SolTrace, rays are sampled from what is termed a "pillbox distribution," which is illustrated on the right side of Figure 3.13. While this distribution does not perfectly mimic natural solar dispersion, it is sufficiently accurate for scenarios where mirror surface imperfections dominate the optical outcomes. Such imperfections typically have a more significant impact on the system's performance than the slight variations in the sun's ray directions. For applications like power towers, where the heliostats (large mirrors that track the sun) exhibit fewer surface imperfections, the accuracy of the sun's angle distribution becomes more critical. In these cases, it is advisable for users to select a sun angle distribution that aligns with their specific geographic location and atmospheric conditions to achieve more precise simulation results.

In SolTrace, the impact of surface imperfections on the optical performance of a mirror is accounted for through the use of user-defined slope errors. These errors represent deviations in the angle of reflected rays which arise from imperfections in the mirror surface often caused by manufacturing processes or weathering. Users can specify the extent of these imperfections by defining the standard deviation of the slope error which assigns the variability in the angle of reflection. In a theoretical scenario where a mirror is manufactured perfectly, this standard deviation would be zero indicating that rays would be perfectly reflected. However, for parabolic troughs, the slope errors are generally on the higher side. One significant factor contributing to this is the curved shape of these large mirrors which makes it susceptible to surface imperfections during the manufacturing process. To illustrate the implications of various slope errors, consider Figure 3.14. This figure presents the intercept factor as a function of degrees defocused for the SCA configuration used at Solana with different amounts of slope error.

When the slope errors are below 2 milliradians, the optical efficiency of the collector remains nearly optimal. However, as the slope error exceeds the 2 milliradian threshold,

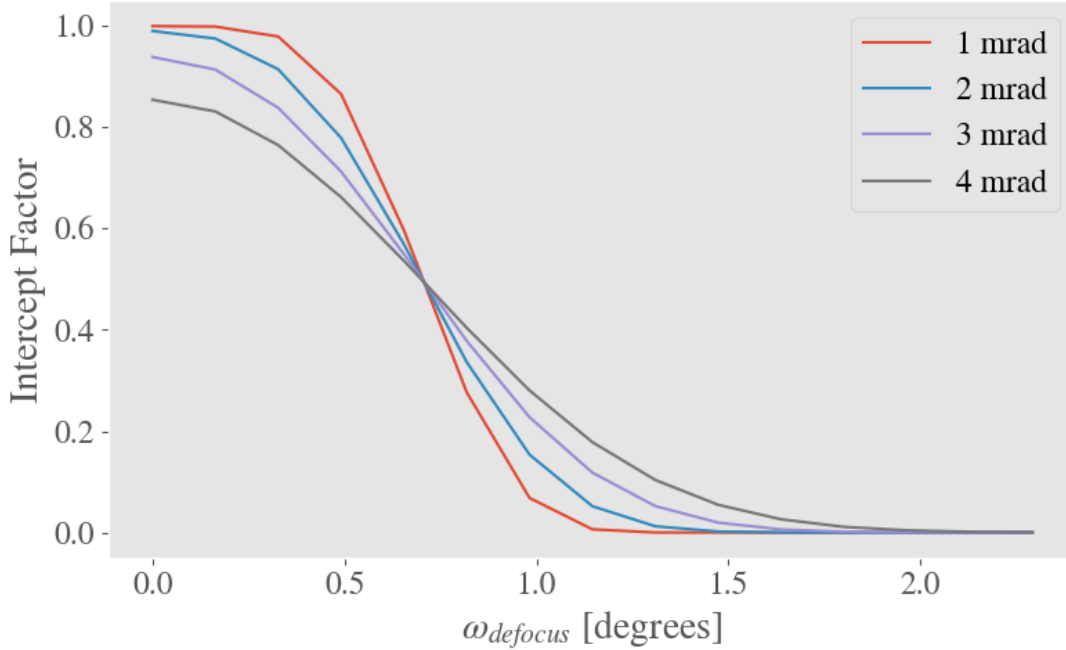


Figure 3.14: Intercept factors computed using various intercept factors

a noticeable decline in optical efficiency is observed. This decrease is attributed to the increased scatter of reflected rays which are less likely to converge at the focal point of the collector. For this work, an assumed slope error of 3 milliradians is utilized. This higher value was chosen to represent the surface imperfections that were noted during numerous site visits.

Since SolTrace employs a Monte-Carlo method, addressing convergence is crucial to ensure the model's reliability. In Monte-Carlo simulations, convergence refers to the point at which increasing the number of trials (or rays, in the case of SolTrace) does not significantly change the outcome thereby indicating that the simulation results are stable and can be trusted. In this work, the convergence was tested by performing the same simulation multiple times where the number of rays varied from 5×10^2 to 5×10^6 . To quantitatively assess the convergence, the Euclidean distance between the intercept factor curve generated with the maximum number of rays (5×10^6) and the curves from each of the other simulations was calculated. This error metric provides a straightforward

measure of difference between the results of the two simulations. A smaller Euclidean distance indicates closer alignment between curves suggesting that the simulation is approaching convergence. The results of this analysis are illustrated in Figure 3.15.

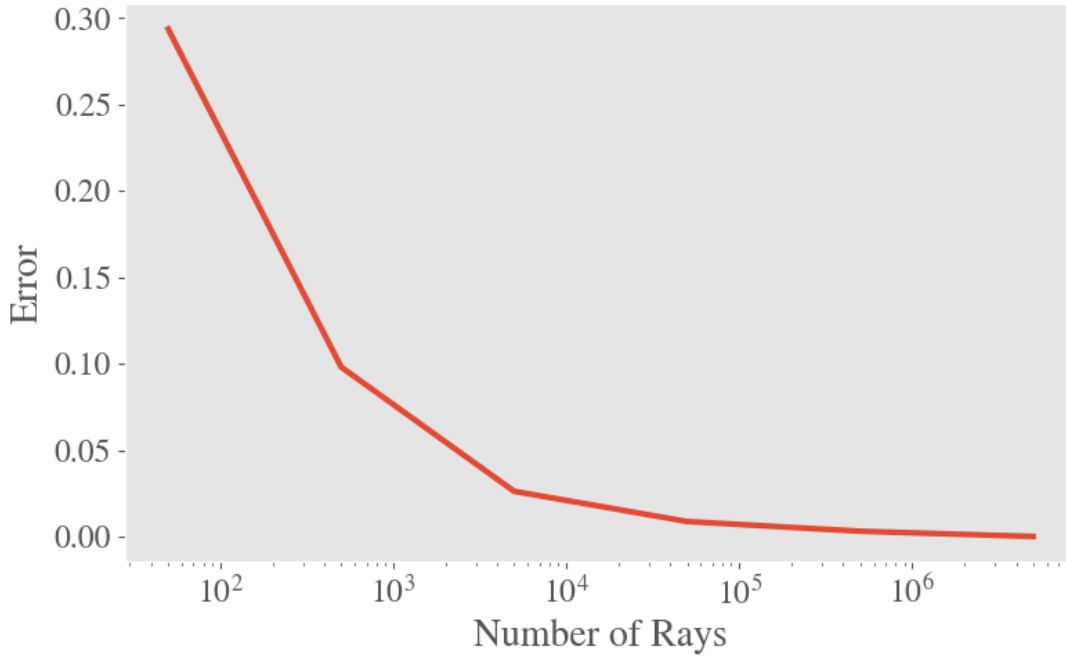


Figure 3.15: Error between intercept factor curves as a function of rays used during the simulation

As demonstrated in Figure 3.15, the simulation results using 5×10^5 rays closely align with those obtained when 5×10^6 rays are used. Specifically, the Euclidean distance between the various intercept factor curves are very small. This close agreement indicates that increasing the ray count beyond 5×10^5 yields little improvement in the accuracy of the results and suggests diminishing returns on computational resources.

With the intercept factors defined by the results from the SolTrace simulation, the next step is to incorporate these results into the TRNSYS model. In this work, it is assumed that each SCA can defocus only to a discrete set of angles. These defocus angles and corresponding intercept factors obtained from SolTrace are defined by the user. The mapping between defocusing angles and intercept factors is illustrated in the code below.

Listing 3.8: Assign intercept values for various defocusing values

```

1 ! Array defining number of degrees defocused
2 omega_defocus = (/0.0, 0.8, 1.2, 2.0)
3 ! Array defining intercept factor for corresponding defocusing angle
4 eta_defocus = (/eta_defocus_1, eta_defocus_2, eta_defocus_3, eta_defocus_4/)

```

With all of the optical losses now defined, the total radiation energy incident upon a solar collector assembly can be computed. This is accomplished using Equation 3.32.

$$\dot{q}_{\text{solar}} = I_{\text{bn}} A_{\text{ap,SCA}} \cos(\theta) \eta_{\text{opt}}(\theta, \omega_{\text{col}}, \omega_{\text{defocus}}) \quad (3.32)$$

Where:

$$\eta_{\text{opt}}(\theta, \omega_{\text{col}}, \omega_{\text{defocus}}) = \eta_{\text{endLoss}}(\theta) \eta_{\text{shadow}}(\omega_{\text{col}}) \eta_{\text{IAM}}(\theta) \eta_{\text{SCA}}(\omega_{\text{defocus}}) \eta_{\text{track}} \rho_{\text{m}} \eta_{\text{soil}} \quad (3.33)$$

I_{bn} is the direct normal irradiation incident upon the field, and $A_{\text{ap,SCA}}$ is the total aperture area of collectors that make up an SCA.

3.2.2 Solar Field Loop Thermal Model

A finite volume approach is utilized to compute the temperatures of the HTF in a solar field loop. This is accomplished by first discretizing the solar field loop into a user defined number of control volumes as illustrated in Figure 3.16.

However, unlike traditional finite volume approaches, the temperatures computed are at the interface of the control volumes instead of the average temperature of each individual control volume. This subtle but important difference in approach has advantages for control volumes in which there is a linear temperature gradient between the inlet and outlet. By tracking the boundary temperatures instead of the mean control volume temperature, the simulation maintains good stability for coarser discretizations. Formally,

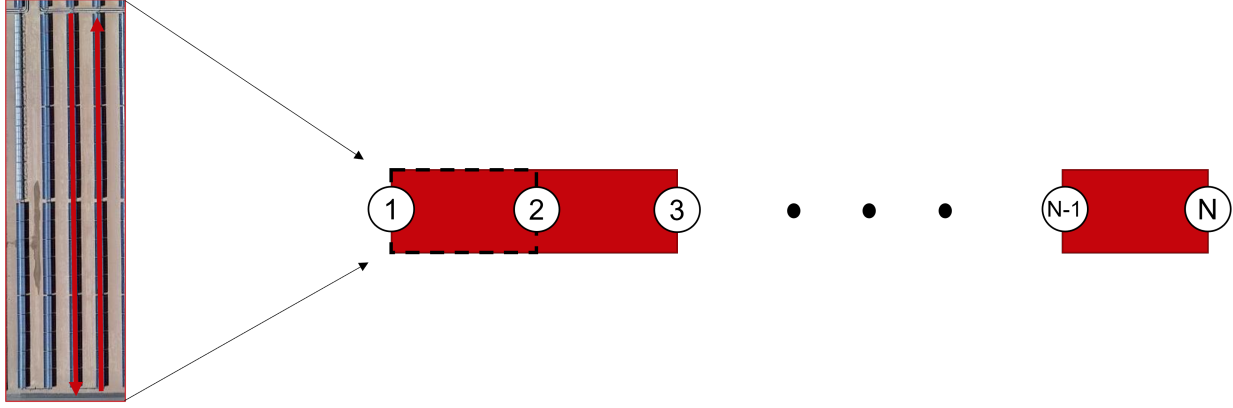


Figure 3.16: Discretization of solar field loop into $N - 1$ control volumes

the energy balance is expressed as Equation 3.34.

$$\dot{m}h_i + \dot{q}_{in} = C_{pipe}(Vol)\rho\bar{c}\frac{\partial\bar{T}}{\partial t} + \dot{m}h_{i+1} \quad (3.34)$$

Here, $\dot{q}_{in}$ represents the thermal energy absorbed by the HTF, $\bar{c}$ is the specific heat of the HTF evaluated at the average temperature of nodes i and $i + 1$, h_i is the enthalpy of the HTF evaluated at the temperature of the i^{th} node, and $\frac{\partial\bar{T}}{\partial t}$ is the average temperature rate of change of the control volume.

Furthermore, C_{pipe} is a multiplier used to represent the thermal capacitance of the piping in the solar field loops. This unitless multiplier should be determined by attempting to fit operational data. The importance of selecting an appropriate value for this parameter is quite high. Anecdotal evidence suggests that setting this value to 1.5 is a good place to start. When fine tuning the value, attention should be focused on matching transient events such as HTF heating up in the solar field following sunrise or HTF cooling off in the solar field as a sunset approaches. For example, if the HTF heats up too quickly during the morning start-up phase in the simulation, it indicates that the thermal capacitance is underestimated. In this case, increasing the value of C_{pipe} would be appropriate as it slows down the rate of temperature rise. Conversely, if the HTF heats up more slowly in

the simulation than what is observed in the field, this suggests that the thermal capacitance is overestimated. Reducing the value of C_{pipe} will allow the HTF to heat up faster. In essence, a higher C_{pipe} value results in a slower response of the HTF temperature to changes in solar heating, whereas a lower value leads to a quicker response.

After some algebra, the control volume temperature rate of change can be computed.

$$\frac{\partial \bar{T}}{\partial t} = \frac{1}{C_{pipe}(Vol)\rho\bar{c}}(\dot{m}\bar{c}(T_i - T_{i+1}) + \dot{q}_{in}) \quad (3.35)$$

Note, the difference between inlet and outlet enthalpies is approximated with $\bar{c}\Delta T$. The choice of placing temperature nodes at the boundary of the control volumes means that the time rate of change of a tracked node is dependent on the mean temperature of both of the adjacent control volumes. Therefore, the nodal temperature rates of change are computed by averaging the control volume temperature rates to the left and right of the node of interest, as illustrated in Equation 3.36.

$$\frac{\partial T_i}{\partial t} = \frac{1}{2} \left[\left(\frac{\partial \bar{T}}{\partial t} \right)_{LHS} + \left(\frac{\partial \bar{T}}{\partial t} \right)_{RHS} \right] \quad (3.36)$$

With the nodal temperature derivatives, the temperatures of each node are stepped through time using the Runge-Kutta 4th order method.

To validate the strategy of positioning nodes on the edges of the control volume, a case study was developed. This study was designed to be simple enough such that it had an analytical solution to be used as a baseline for comparison. In this study, HTF flows at a constant rate through a solar field loop and maintains a constant inlet temperature over time. Also, a uniform heat flux is applied along the loop's length, and to simplify the problem further, the HTF material properties are assumed to be constant regardless

of temperature. This case study is illustrated in Figure 3.17.

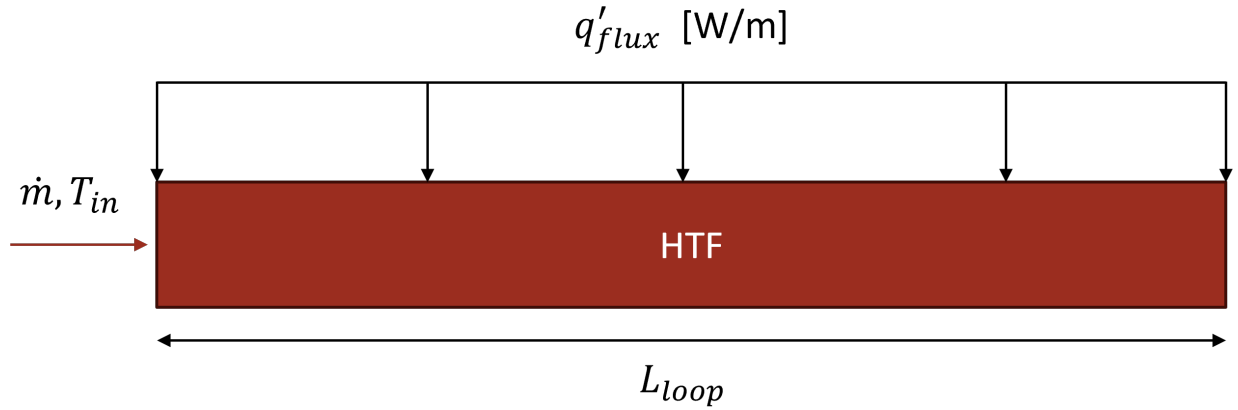


Figure 3.17: Simple case study with constant flux applied to HTF in a solar field loop

Conducting an energy balance on a differential control volume of the HTF yields a partial differential equation that varies with position and time. Next, boundary conditions are defined such that all the HTF is originally T_{in} and that the inlet temperature is always T_{in} . Using these boundary conditions, a closed form solution for HTF temperature spatially and temporally was obtained.

$$T_{HTF}(x, t) = \begin{cases} \frac{\phi x}{v_{fluid}} + T_{in} + \frac{\phi(tv_{fluid} - x)}{v_{fluid}} & \text{for } x > v_{fluid}t \\ \frac{\phi x}{v_{fluid}} + T_{in} & \text{for } x \leq v_{fluid}t \end{cases} \quad (3.37)$$

Where:

$$\phi = \frac{q'_{flux}}{A_{cross-section}\rho c} \quad (3.38)$$

The analytical solution served as the baseline for comparisons with the numerical approach which also simulated the same system. To investigate convergence, the simulation was performed multiple times with different number of nodes used for discretization. Figure 3.18 illustrates this comparison by showing the analytical solution as a line and

the numerical model solution as dots across different node counts.

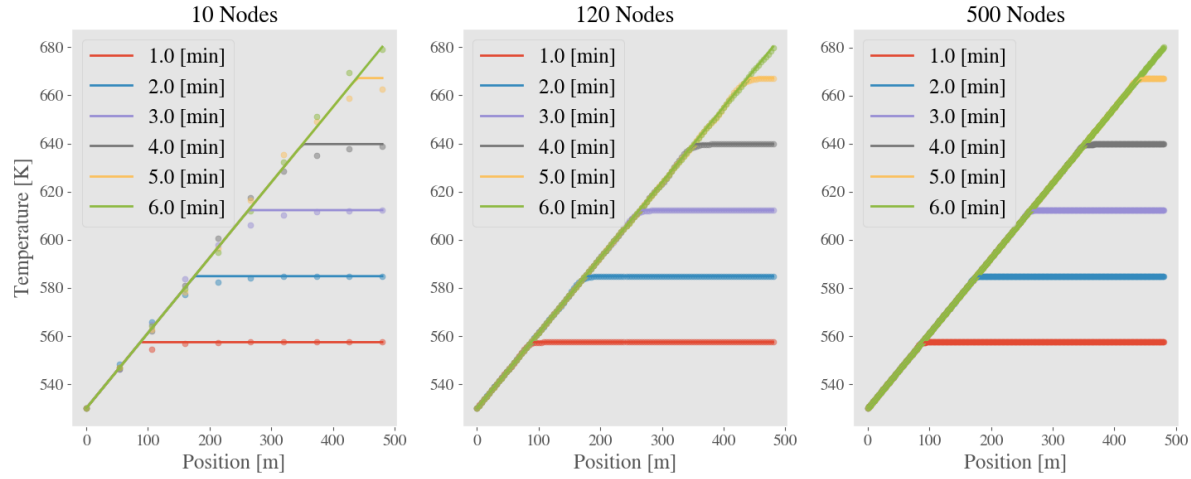


Figure 3.18: Comparing analytical solution (line) with numerical model (dots) using different amount of nodes

Using 10 nodes, significant deviations are observed between the numerical model and the analytical solution. However, these deviations diminish substantially with 120 nodes resulting in excellent agreement between the two models. With 500 nodes, the differences are essentially eliminated which indicates that the numerical approach converges accurately to the analytical solution. The case with 120 nodes is particularly relevant as it matches the physical lengths of individual receivers at Solana, correlating each control volume directly with a physical receiver. This alignment adds meaningful context to the selected number of control volumes.

Lastly, this case study was also used to compare a numerical model with nodes located at the center of a control volume. Using a control volume energy balance, the temperature rate of change for each node is defined below.

$$\frac{\partial T}{\partial t} = \frac{1}{(\text{Vol})\rho c} (\dot{m}c(T_{\text{in}} - T_{\text{out}}) + \dot{q}_{\text{in}}) \quad (3.39)$$

The primary challenge with this modeling approach stems from the assumption of a

linear temperature across each control volume. The outlet temperature for each control volume is calculated using Equation 3.40.

$$T_{\text{out},i} = 2T_i - T_{\text{in},i} \quad (3.40)$$

This linear assumption increases the model's sensitivity to variations in input temperature which leads to significant instability. As a result, the model struggles to converge accurately to the analytical solution even with a large number of nodes. This issue is demonstrated in Figure 3.19.

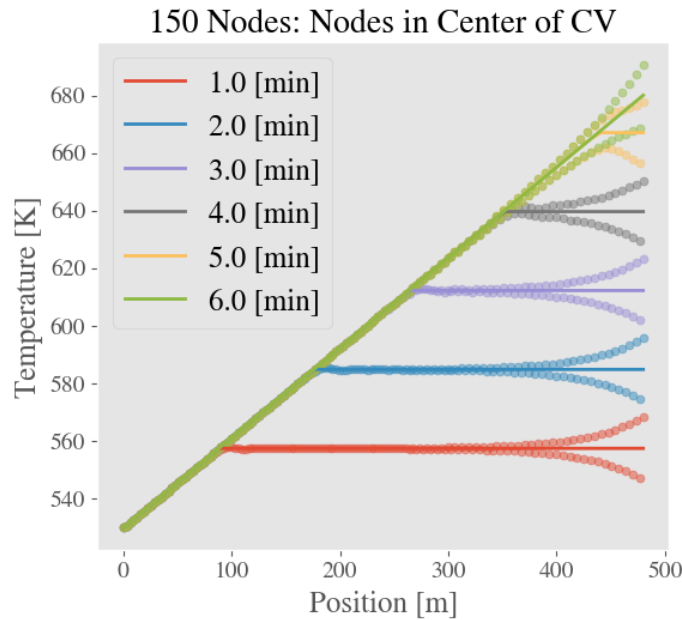


Figure 3.19: Instability with numerical model that has nodes in the center of each control volume

3.2.3 Receiver Heat Loss Model Overview

The solar field loop thermal model relies heavily on the accurate computation of the heat absorbed by the HTF, $\dot{q}_{\text{in}}$. This computation is performed using the Forristall model [8] energy balance which serves as the basis for the SAM physical trough thermal model [7]. This model utilizes a detailed 1-D energy balance to compute the heat lost to the

environment and subsequently the heat absorbed by the HTF. While the Forristall model has been validated and is considered accurate, it is relatively computationally expensive when applied to each individual control volume. Thus, to decrease the time it takes to solve the energy balance, a neural network is utilized to compute the heat absorbed by the HTF ($\dot{q}_{in}$) as a function of HTF temperature (T_{HTF}), ambient conditions (T_{amb} , T_{sky} , v_{wind}), HTF mass flow ($\dot{m}_{HTF}$), gas pressure in the annulus (P_a), and solar energy incident upon the absorber ($\dot{q}_{solar}$). While neural networks often have a sense of mystique for engineers, this approach can simply be thought of as developing a large, non-linear, multivariate curve-fit to the Forristall model. The design of the neural network utilized is illustrated in Figure 3.20.

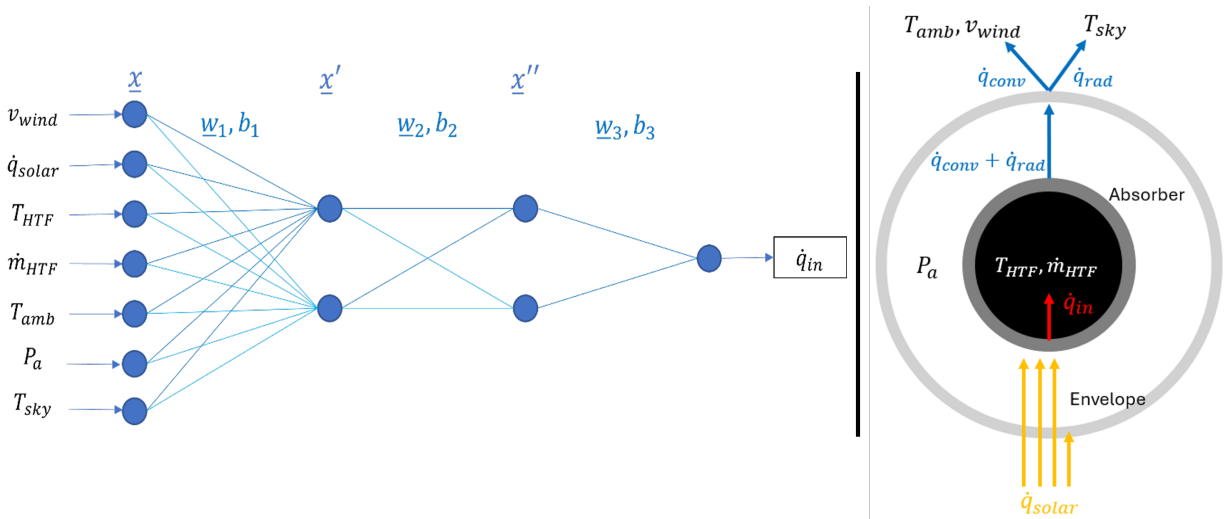


Figure 3.20: Visual representation (not to scale) of neural network used to predict HTF heat absorption (left) and Forristall model energy balance (right)

It is important to note that this visualization is true in that it takes the given inputs and uses them to predict heat absorption. However, the architecture used in this work is different from Figure 3.20 as there are seven hidden layers with varying number of nodes ranging from seven to two. In Figure 3.20, there are two hidden layers each with two nodes. Furthermore, each hidden layer is fully connected and utilizes the ReLU activation function to introduce non-linearity [35].

There are two significant aspects to consider when implementing this neural network approach. Firstly, due to the significant performance variations based on the collector's different states (such as pristine, broken glass, lost vacuum, and hydrogen permeation), a separate neural network needs to be trained for each state, resulting in a total of four trained neural networks. Additionally, the effectiveness of the neural network relies on its ability to accurately predict heat absorption across a wide range of simulation conditions. Thus, developing a reliable and effective method for training the neural network is of paramount importance.

3.2.4 Neural Network Architecture and Training Protocol

As mentioned in the previous section, the receiver heat loss neural network contains seven hidden layers with varying number of nodes. While numerous neural network architectures were tested, the one that yielded the best accuracy is described in Table 3.3.

Table 3.3: Neural network architecture utilized to compute heat absorption

Layer	1	2	3	4	5	6	7	8
Nodes	5	6	4	5	4	3	2	1
Act. Function	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	Linear

All 7 hidden layers utilize the ReLU activation function to introduce non-linearity. However, the last layer, which computes the prediction for heat absorption, uses a linear activation function. This is standard practice in neural network regression applications.

The choice to use the ReLU activation function is due to its superior performance for use in deep learning applications [36]. The ReLU activation function along with other commonly used functions are presented in Figure 3.21.

There are a handful of beliefs why ReLU functions are more useful for deep neural networks. One of the most prominent is the idea of the vanishing gradient. Common activation functions like sigmoid and tanh suffer from this problem where gradients

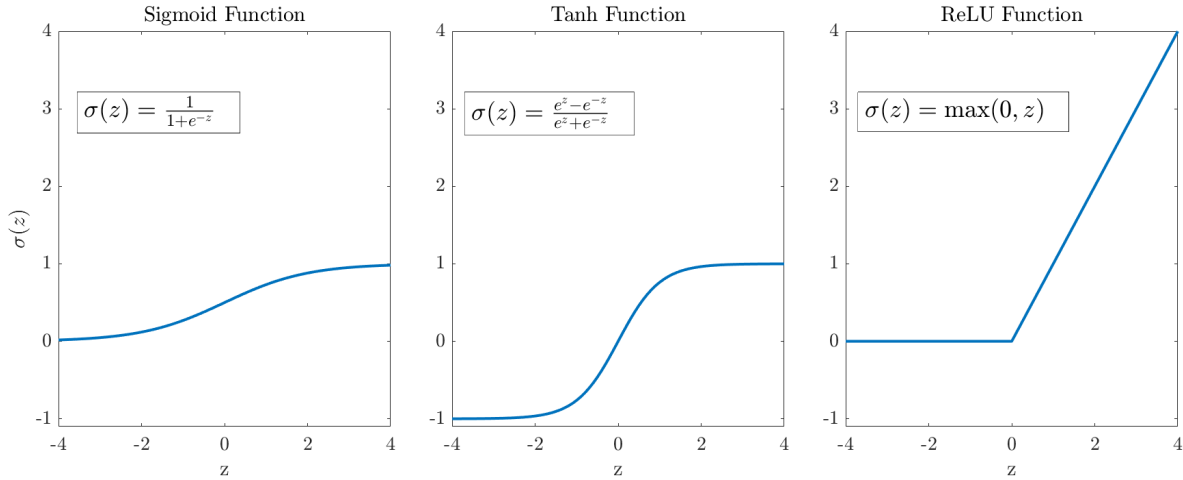


Figure 3.21: Various activation functions commonly used in neural networks

become small and insignificant as they are propagated back through the network during training. When gradients approach zero, it is difficult to update the weights and learn from the training data. Because the ReLU has a constant gradient for all positive values, as opposed to a near zero gradient for positive values greater than one, there is a smaller probability that the vanishing gradient phenomenon will occur.

The training process of the neural network involves finding optimal weight values using gradient descent, which is implemented using the TensorFlow library in Python. However, before training the neural network, a training dataset must be established. The dataset is comprised of combinations of input values within specified minimum and maximum ranges where, for each combination, a python implementation of the Forristall model is utilized to compute $\dot{q}_{in}$. Table 3.4 presents the bounds on the inputs used for training data generation.

Table 3.4: Limits for neural network training dataset

	$v_{wind} \left[\frac{m}{s} \right]$	$DNI \left[\frac{W}{m^2} \right]$	$T_{HTF} [K]$	$\dot{m}_{HTF} \left[\frac{kg}{s} \right]$	$T_{amb} [K]$	$P_a [torr]$	$T_{sky} [K]$
Min.	0.5	0	470	0.5	273	0.001	263
Max.	5	1100	675	5	323	1	323

The bounds defined in Table 3.4 are derived from historical weather data and physical limitations of the universe and of the CSP plant. For example, due to Earth's position in

the solar system, there is a maximum amount of direct normal irradiance (DNI) that can be incident upon Earth's surface. Furthermore, all trough plants have a wind speed limit where collectors will defocus to a stow mode that protects the collectors from the stresses due to aerodynamic effects, so the maximum wind speed used to generate training data can be defined as the wind speed corresponding to this stow limit. Thus, the bounds on the input data used to generate training data accurately reflect and capture the wide range of operational conditions experienced in a solar field.

With the bounds defined, it is now necessary to define the values of the input variables within those bounds. This is accomplished using linear spacing and a input specific number of discretizations. For example, the code snippet below defines an arrays containing the various input values used for the training dataset generation where the bounds are those previously defined and the number of samples per feature is defined by `n_variable`.

Listing 3.9: Define arrays used to generate combinations of input features

```

1 # Wind Velocity
2 v_wind_array = np.linspace(v_wind_min, v_wind_max, n_wind)
3
4 # DNI
5 DNI_array = np.linspace(DNI_min, DNI_max, n_DNI)
6
7 # HTF Temperature
8 T_HTF_array = np.linspace(T_HTF_min, T_HTF_max, n_T_HTF)
9
10 # HTF Mass Flow Rate
11 m_dot_array = np.linspace(m_dot_min, m_dot_max, n_m_dot)
12
13 # Ambient Temperature
14 T_amb_array = np.linspace(T_amb_min, T_amb_max, n_T_amb)
15
16 # Annulus Pressure
17 P_a_array = np.linspace(P_a_min, P_a_max, n_P_a)
18
19 # Sky Temperature

```

```
20 T_sky_array = np.linspace(T_sky_min, T_sky_max, n_T_sky)
```

Each input variable has a different number of values, and this number is correlated to its impact on the computation of q_{in} . This is especially relevant for different receiver states. To illustrate this idea, consider the effect ambient conditions has on heat loss for a receiver that is pristine and a receiver that has its glass shattered off. For the pristine collector, heat loss is incredibly low, so the variability of $\dot{q}_{in}$ due to variability in ambient conditions is quite low. On the other hand, for a receiver with broken glass, heat loss is high and depends significantly on ambient conditions. This results in a high degree of variability of $\dot{q}_{in}$ due to variability in ambient conditions. Thus, the training dataset corresponding to the neural network with broken glass must have more ambient condition data points to accurately compute heat loss across a wide range of operating conditions in comparison to the pristine receiver. That is, n_{wind} for the broken glass training dataset would be higher than n_{wind} for the pristine training dataset. This is reflected in Table 3.5 which presents the number of samples within the training datasets for the four receiver states.

Table 3.5: Number of samples within the training dataset for each neural network

Receiver State	Number of Training Samples
Pristine	72,000
Lost Vacuum	72,000
Broken Glass	1,536,640
Hydrogen Permeation	1,693,440

This table leads to a few items for discussion. First, the heat loss effects due to ambient conditions are less for pristine and lost vacuum receivers compared to receivers with broken glass or hydrogen permeation. This results in less data points for the ambient condition input variables and subsequently less total training samples. Second, heat loss for receivers with hydrogen in the annulus space is the most difficult to capture resulting in the highest number of training samples. This is due to the fact that the pressure of hydrogen within the annulus is not constant as it is for the other receiver states, and as

the pressure of hydrogen increases, ambient conditions have a greater impact on heat loss. Lastly, it is advantageous to have as few training samples as possible. Smaller training datasets not only take less time to generate, but they also result in shorter training time for the neural network. Therefore, the number of training samples utilized was derived through trial and error and has been used to train neural networks that can adequately predict heat loss over the wide range of operating conditions CSP plants experience.

With the training datasets generated, the last step before training the neural networks is data pre-processing. This often overlooked step for new machine learning practitioners is perhaps the most important. Because the inputs have different units, it is important to utilize some form of scaling. While there are numerous options for scaling such as min-max scaling (normalization), z-score normalization (standardization), and logarithmic scaling, the scaling chosen should reflect the distribution of the input values. For example, if the training dataset is comprised of inputs that largely follow normal distributions, z-score normalization should be used. For this work, min-max scaling is the most appropriate as the training data is linearly distributed across defined upper and lower bounds. Scaling is applied to both inputs and outputs of the neural network. The normalization used is provided in Equation 3.41.

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} + 0.1 \quad (3.41)$$

This equation bounds all inputs and outputs used for training between 0.1 and 1.1. Note, this is slightly different from the typical normalization bounds of 0 and 1. There are a few reasons for this deviation. First, although it is extremely unlikely, during operator simulations it is possible to have input values that are less than the minimum bounds set; specifically, this could be the case for ambient temperature and HTF temperature. Input values less than the minimum would then be normalized to a negative value. Propagating

a negative a through a neural network that is only trained with positive input values would almost certainly yield extremely poor results. Thus, the padding of 0.1 attempts to minimize the chances of this occurring. Additionally, when a 0 is propagated through a neural network, it provides no useful information for training. To validate this claim, consider Equation which is a simplified form of backpropagation used for training neural networks.

$$w_j^{t+1} = w_j^t - \beta \delta x_j \quad (3.42)$$

Here, the j^{th} weight, w_j is optimized iteratively using gradient descent. β is a learning rate, δ is a representative error value, and x_j is the input value. When, x_j is 0, the gradient value becomes zero, and the weight cannot be optimized. Thus, having a training dataset with numerous input values set to 0 results in many samples that cannot be used to optimize the neural network weights.

Lastly, the neural networks trained are dependent on the geometry and thermal material properties of the receiver. While the simulation tool will come prepackaged with default receiver neural networks corresponding to receivers in use today, it is within the scope of this project to allow the user to define their own receiver geometries and properties. Thus, a set of scripts were developed to automatically create and train the neural networks for a new receiver specification. Safeguards were included to ensure convergence during the training period and reliability for use in the simulation. The total time it takes to create and train all four of the neural networks is about 30 minutes.

3.2.5 Solar Field Sector Model

The solar field sector model simulates numerous solar field loops connected by inlet and return headers. Similar to the loop thermal model, a finite volume approach is

utilized to compute the temperatures of HTF in the inlet header and return header. An illustration of the coupled header/loop model is presented in Figure 3.22 where the various dots represent nodes where temperatures are tracked. Note, this illustration is not representative of the number of control volumes used.

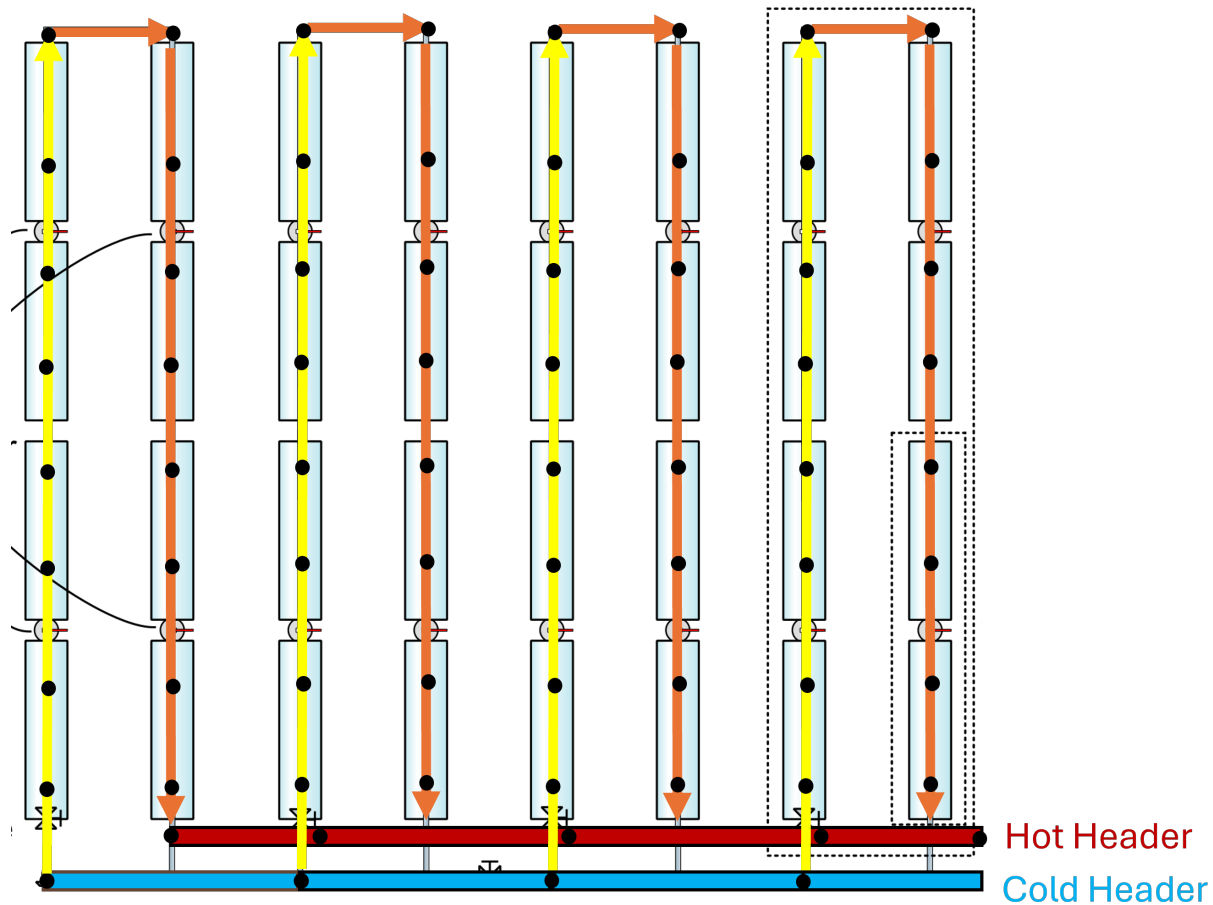


Figure 3.22: Solar field sector model illustrating nodes in finite volume approach used to compute temperatures

When creating sectors in the model, users have the ability to define the number of loops. Although Figure 3.22 only presents 4 loops, the solar field sectors are defined such that loops are always symmetric about the headers. Thus, for this solar field sector, the model would have an additional 4 loops below the cold/hot headers that are not pictured. It's important to highlight that all solar field loops in each sector (up to 104 at Solana) are modeled. Simulating all loops faster than real-time is made possible solely by the

computational efficiency of the neural network approach.

One of the key advantages of this solar field sector model is that users have the ability to define various imperfections that typically aren't captured in detail. For example, with the neural network approach outlined in Section 3.2.3, there are four receiver states that can be simulated: pristine, broken glass, lost vacuum, and hydrogen. With each solar field sector, users have the ability to define the parameters outlined in Table 3.6.

Table 3.6: User-defined parameters utilized to generate solar field loops in a sector

Parameter	Description
Prob_pristine	Fraction of receivers in pristine condition [0-1]
Prob_brokenGlass	Fraction of receivers with shattered glass [0-1]
Prob_lostVacuum	Fraction of receivers with the vacuum lost [0-1]
SF_avail	Fraction of SCA's in sector that are operational [0-1]
m_dot_std	Standard deviation of mass flow rates through solar field loops
n_H2	Number of receivers per loop with hydrogen in the annulus space
H2_std-i	Standard deviation of hydrogen pressure in annulus space in the i^{th} receiver [10^x torr]
H2_mean-i	Average hydrogen pressure in annulus space in the i^{th} receiver [10^x torr]

First, users can define the number of receivers with hydrogen in the annulus space. Surveying and communication with industry partners has indicated that hydrogen permeation occurs in receivers where HTF is the hottest as this is where the HTF degrades. Therefore, the receivers with hydrogen in the annulus space are generally located at the end of the solar field loops. Because of this observed phenomena, users have the ability to define the number of individual receivers that have hydrogen in them starting at the last receiver in the loop. Then, for each receiver, the user can specify a hydrogen pressure distribution representative of the sector. This concept is illustrated in Figure 3.23 where a user has specified that the last three receivers in a loop have hydrogen.

This figure illustrates a representative solar field loop indicating which receivers have hydrogen, and it also presents the user defined distribution of hydrogen pressure in

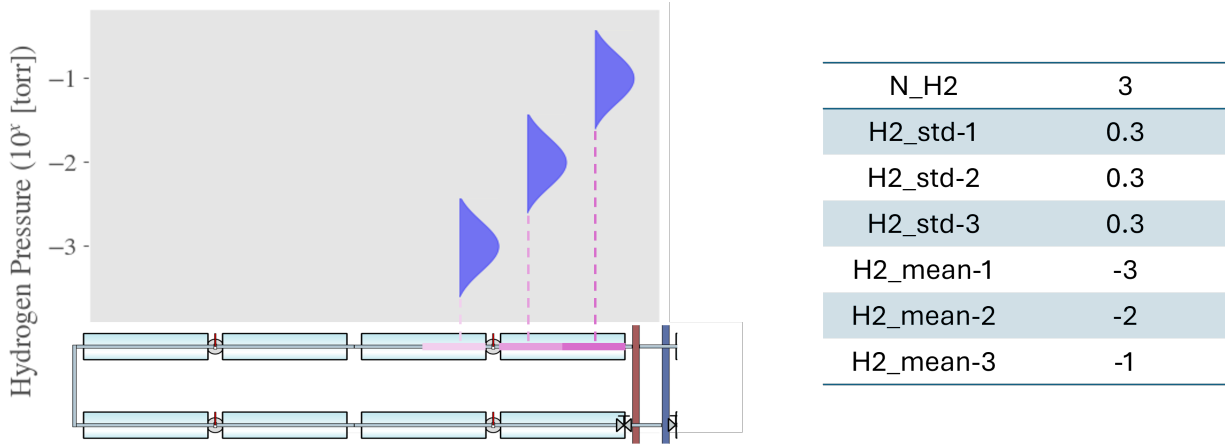


Figure 3.23: User defined hydrogen permeation in the last three receivers with various pressure distributions

those receivers. The solar field loop, In this example, the user has specified that the last three receivers in a loop have accumulated hydrogen in the annulus space. Furthermore, the user has also determined that there are higher average hydrogen pressures in receivers closer to the loop outlet. With this information specified, loop definitions will be generated for every loop in the sector where the pressure in the last three receivers will be randomly sampled from the normal distributions illustrated in Figure 3.23. This introduces realistic variability in heat loss due to more or less hydrogen in the annulus space according to the defined distribution.

The next item to note from Table 3.6 are the receiver probabilities. For the remaining receivers that don't have hydrogen, the state of those receivers are determined based on user input values of Prob_pristine, Prob_brokenGlass, and Prob_lostVacuum which are the respective probabilities of the state's existence in the sector. Using these probabilities, a domain for choosing receiver states is produced. For example, Figure 3.24 illustrates the domain used for selecting receiver states for a sector with Prob_pristine = 0.9, Prob_lostVacuum = 0.07, and Prob_brokenGlass = 0.03. Note, it is crucial that these three probabilities sum to 1.

For each non-hydrogen receiver in each loop, a random number is generated from

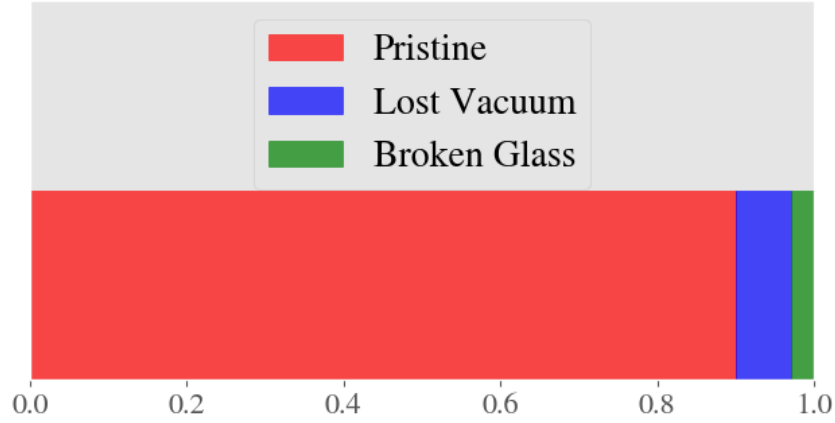


Figure 3.24: Receiver state probabilities

a uniform distribution between 0 and 1. This random number is then compared to the domains presented in Figure 3.24, and the domain it falls into will be the state of that receiver. For example, if the random number generated is 0.7, the receiver will be defined as pristine. However, if a random number is 0.96, the receiver is modeled in the lost vacuum state.

The last two relevant parameters to discuss are SF_{avail} and m_{dot_std} . SF_{avail} allows the user to define the percentage of solar collector assemblies that are in operation. Potentially, SCA's could be out of service due to mirror cleaning or broken tracking drives. Next, m_{dot_std} allows the user to introduce variability to mass flow rates of HTF through the solar field loops. The average flow rate to each loop is defined as the total mass flow into the sector divided by the number of loops in the sector. Flow rate variability is then introduced for each loop using Equation 3.43.

$$\dot{m}_i = \dot{m}_{ave}(1 + \mathcal{N}(0, (m_{dot_std})^2)) \quad (3.43)$$

The flow rate fore each loop, $\dot{m}_i$, is determined based on the average flow rate through the sector, $\dot{m}_{ave}$, plus an adjustment factor. For each loop at the beginning of the simulation, this factor is generated from a random sampling of a zero-mean normal distribution

with a standard deviation of m_dot_std . This results in the flow rates for the loops to follow a normal distribution as well. Because the distribution is normal, 95% of the loops in the sector will have a mass flow rate within $\pm$ two standard deviations of the average. Therefore, if a user is confident that loop control valves are set such that 95% of the loops have mass flow rates that are within $\pm$ 5% of the average, the user should set m_dot_std equal to 0.025.

3.2.6 SCA Defocusing

Parabolic trough HTF temperatures at the outlet of a solar field loop may exceed a temperature limit that leads to HTF degradation. To prevent this degradation, heat absorption in solar field loops is managed by defocusing control strategies. Recall, each solar field loop is comprised of solar collector assemblies (SCA's) which are simply collectors that all share the same tracking drive. For Solana, there are four SCA's per loop with temperature sensors at the midpoint of each SCA. The defocusing algorithm implemented is loosely based on the original Solana defocusing controller (no longer in use). The pseudo-code below outlines the defocusing strategy used in the solar field model.

- If $T_{outlet} > T_{limit}$, Then the last two SCA's will defocus 0.8° ($\eta_{SCA} = 0.4$)
- If $T_{outlet} > T_{limit} + 2.0[K]$, Then the last two SCA's will defocus 1.2° ($\eta_{SCA} = 0.1$)
- If $T_{outlet} > T_{limit} + 4.0[K]$, Then the last two SCA's will defocus 2.0° ($\eta_{SCA} = 0.0$)
- If $T_{outlet} > T_{limit} + 6.0[K]$, Then all SCA's will defocus 2.0° ($\eta_{SCA} = 0.0$)

Here, T_{outlet} is the loop outlet temperature, T_{limit} is the HTF temperature limit defined by the user, and η_{SCA} is the optical efficiency of the SCA given the current amount of defocusing. These efficiencies are computed with SolTrace as described in Section 3.2.1.

3.2.7 Expansion System

The expansion system, as described in Section 3.1.4, is comprised of a piping network connecting expansion and overflow tanks. Tracking temperatures of the HTF tanks is relevant because of the potential for HTF flashing in the overflow tank. Thus, the following simple thermal model is implemented to compute the temperatures of HTF in these tanks. The simple tank model is presented in Figure 3.25.

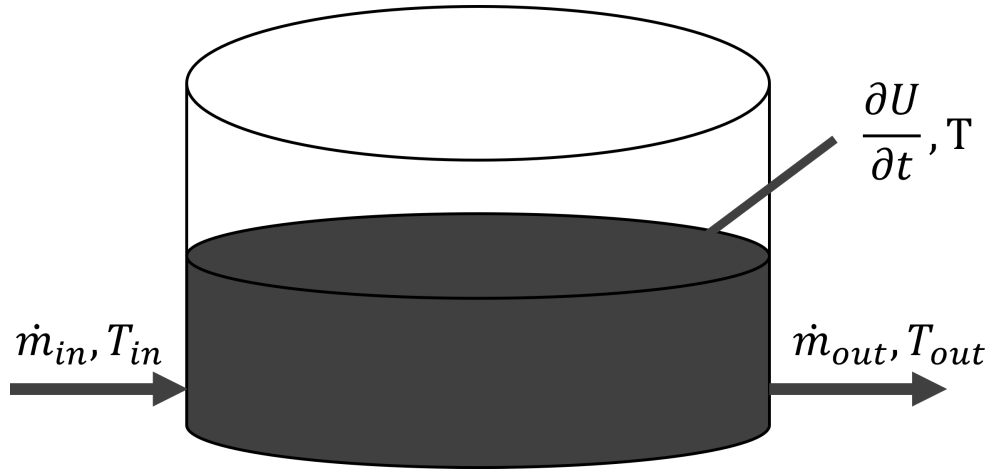


Figure 3.25: Illustration of tank thermal model

The derivation begins with an energy balance on the control volume encompassing the fluid in a tank. The energy balance is provided in Equation 3.44.

$$\dot{m}_{in} h_{in} = \frac{\partial U}{\partial t} + \dot{m}_{out} h_{out} \quad (3.44)$$

$\dot{m}$ is the flow into or out of the tank, h is the enthalpy evaluated at the temperature of its corresponding mass flow, and $\frac{\partial U}{\partial t}$ is the change in internal energy of the control volume with reference to time. This internal energy rate can be expanded by substituting in specific internal energy and applying the chain rule.

$$\frac{\partial U}{\partial t} = \frac{\partial m u}{\partial t} = m \frac{\partial u}{\partial t} + u \frac{\partial m}{\partial t} \quad (3.45)$$

Internal energy, U , is rewritten as the product of the mass of the fluid, m , and the specific internal energy, u , evaluated at the temperature of the HTF in the tank. Using the chain rule, definition of mass rate of change, and thermodynamic relationships, Equation 3.45 can be further simplified to its final form in Equation 3.46.

$$\frac{\partial U}{\partial t} = mc \frac{\partial T}{\partial t} + h(\dot{m}_{in} - \dot{m}_{out}) \quad (3.46)$$

A key assumption is that the pressure-volume term in enthalpy is negligible. Recall, specific enthalpy is computed with Equation 3.47.

$$h = u + Pv \quad (3.47)$$

At relatively high HTF temperatures, as this the case here, specific internal energy contributes significantly more than the pressure volume term. This simplification allows a substitution for specific internal energy and specific enthalpy. The original energy balance from Equation 3.44 can be used to solve for a time-rate change of fluid temperature within the expansion tank.

$$\frac{\partial T}{\partial t} = \frac{1}{mc} (\dot{m}_{in}(h_{in} - h) + \dot{m}_{out}(h - h_{out})) \quad (3.48)$$

Note the specific enthalpy values are as a function of temperature using a quadratic fit to liquid enthalpy data from a specification sheet. Using this temperature rate equation with the mass flow values computed with the hydraulic model, the temperature of the HTF in each tank is stepped through time using the simple Euler technique.

4 RESULTS

This chapter presents interesting outcomes from various applications of the solar field model. The first section of the chapter is dedicated to providing validation of the methodologies that were established in the previous chapter. This validation process is crucial as it not only reinforces the credibility of the model but also sets the stage for its application in operator training scenarios. Following this validation section, the chapter transitions into a series of case studies selected to demonstrate the model's capabilities. These case studies not only illustrate the model's capabilities for a training simulator, but also highlight its potential to optimize the performance of solar fields used in parabolic trough solar thermal power plants in the future.

4.1 Model Validation

The validation phase is important to confirm that the assumptions made in the modeling process are in fact accurate. The thermal model contains novel methodologies and numerous assumptions, so validation efforts are focused here. This section first validates the neural network approach that is used to dramatically reduce the runtime of the simulation. Then, the thermal behavior of a solar field sector is validated with operational data from Solana.

4.1.1 Neural Network

To investigate the predictive ability of the neural networks, solar field loops, comprised of the four relevant collector states, were simulated using weather data from three separate days representing a sunny spring day, sunny summer day, and an intermittent cloudy fall day. For example, one loop is comprised of receivers that are all pristine, and another loop has receivers that all have 0.1 [torr] of hydrogen in the annulus space. For each loop

and day, a simulation was performed where the heat absorbed by the HTF was computed by the neural network and another simulation was performed where the Forristall model implementation was used to compute the heat absorption. The latter serves as the basis for comparison as this is what the neural network is trying to predict. Parameters for the simulations run are presented in Table 4.1

Table 4.1: Parameters used in neural network validation simulation

Parameter	Value
Time Step	1 [s]
Loop Length	480 [m]
Control Volume Size	4 [m]

Note, the simulation runs in real time with 1-second timesteps, and the control volume size was selected to match the length of a receiver at Solana. After running all the simulations, Equation 4.1 quantified the results by computing a normalized root mean squared difference.

$$\text{error} = \left(\frac{\sum_{i=1}^{n_{\text{time}}} \left(\frac{\dot{q}_{\text{NN,tot},i} - \dot{q}_{\text{F,tot},i}}{\max(\dot{q}_{\text{F,tot}})} \right)^2}{n_{\text{time}}} \right)^{0.5} \cdot 100 \quad (4.1)$$

Here, $\dot{q}_{\text{F,tot},i}$ is the total heat absorbed by the solar field loop at time step i computed using the Forristall model, $\dot{q}_{\text{NN,tot},i}$ is the heat absorbed by the solar field loop at time step i predicted by the neural network, and n_{time} is the number of time steps in the simulation. Essentially, this metric communicates the average loop heat absorption error throughout the day as a percentage of the maximum heat absorbed during that day. The results from this study are summarized in Table 4.2 which presents the error for the various loops/states for each of the three days simulated.

Table 4.2: Error between Forristall model and neural network approach for various receiver states

State	Day 1 Error [%]	Day 2 Error [%]	Day 3 Error [%]
Pristine	0.07	0.04	0.06
Broken Glass	0.26	0.76	1.2
Lost Vacuum	0.16	0.15	0.28
H2 (1 torr)	0.3	0.53	0.84
H2 (0.1 torr)	0.27	0.3	0.3
H2 (0.01 torr)	0.16	0.1	0.1
H2 (0.001 torr)	0.09	0.12	0.19

Of the simulations run, the largest RMS difference recorded was 1.2% and the average difference for the 21 simulations performed was 0.3% indicating that the neural network approach is accurate with reference to the Forristall model. To bring more context to these error values, Figure 4.1 presents the difference between heat absorbed by a solar field loop using the neural network and the Forristall approach. Three days are presented along with the different solar conditions. For reference, in these simulations the loop is comprised of receivers with 0.1 [torr] of hydrogen in the annulus space.

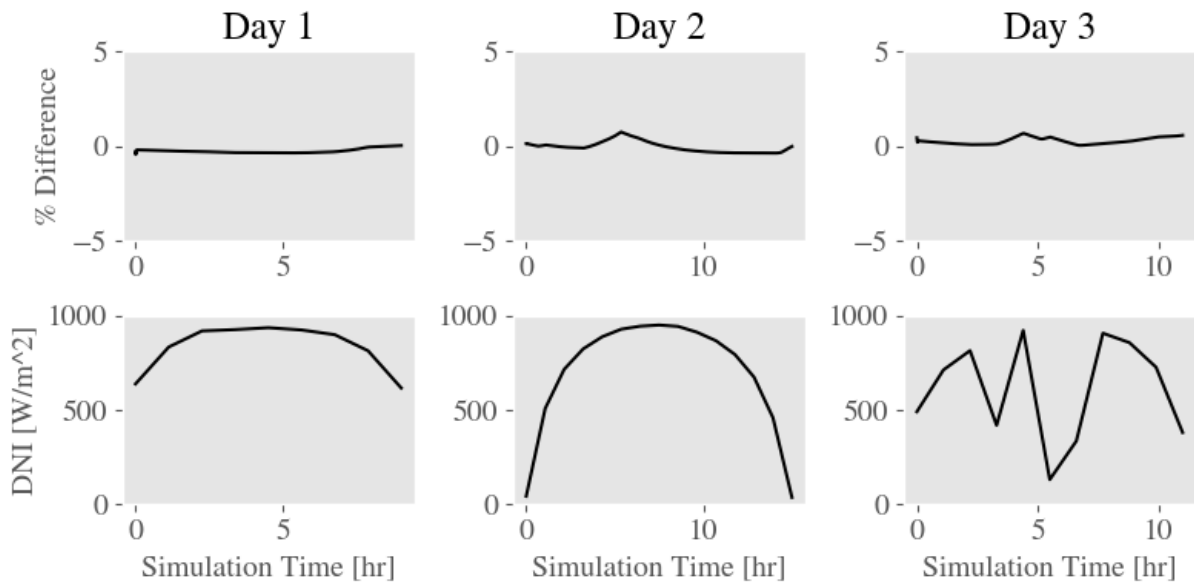


Figure 4.1: Neural network validation study illustrating the difference in the computed heat absorbed by an entire loop using the Forristall model and the neural network

In Figure 4.1, the columns correspond to the day simulated. The top row presents

the percent difference between heat absorbed using the Forristall model and the neural network using Equation 4.1. Next, the bottom row illustrates the solar conditions by plotting direct normal irradiance (DNI). Day 3, representative of a cloudy day, illustrates the neural network's capability to provide performance predictions consistent with the Forristall model over a wide range of operating conditions. With reference to Table 4.2, all three of these simulations have differences near 0.3% thereby depicting the average difference of the simulations performed. Thus, these simulation case studies suggest that the neural network approach can predict heat absorption accurately (relative to the Forristall model) and can be used to predict heat absorption for various receiver states that are more dependent upon ambient conditions.

To further assess the capability of the neural network in predicting the Forristall model output, a study was conducted using a set of baseline input values defined in Table 4.3. The objective of this study was to investigate how changes in each input influence the predicted heat absorption by both the neural network and the Forristall model. These baseline values were chosen based on standard operational ranges for each value.

Table 4.3: Input values used in case study

Input	Value
$\dot{m}_{\text{HTF}}$	5 [kg/s]
T_{HTF}	650 [F]
P_a	0.01 [torr]
T_{amb}	80 [F]
v_{wind}	5 [m/s]
DNI	800 [W/m ²]

Each input was then systematically varied individually from its minimum to its maximum value. To visualize the influence of the various inputs on heat absorption, as well as the potential dependence between prediction error and the various inputs, Figure 4.2 presents the normalized heat absorption predictions for each parameter sweep. Heat absorption and the various inputs are normalized to ensure the results are bounded

between 0 and 1.

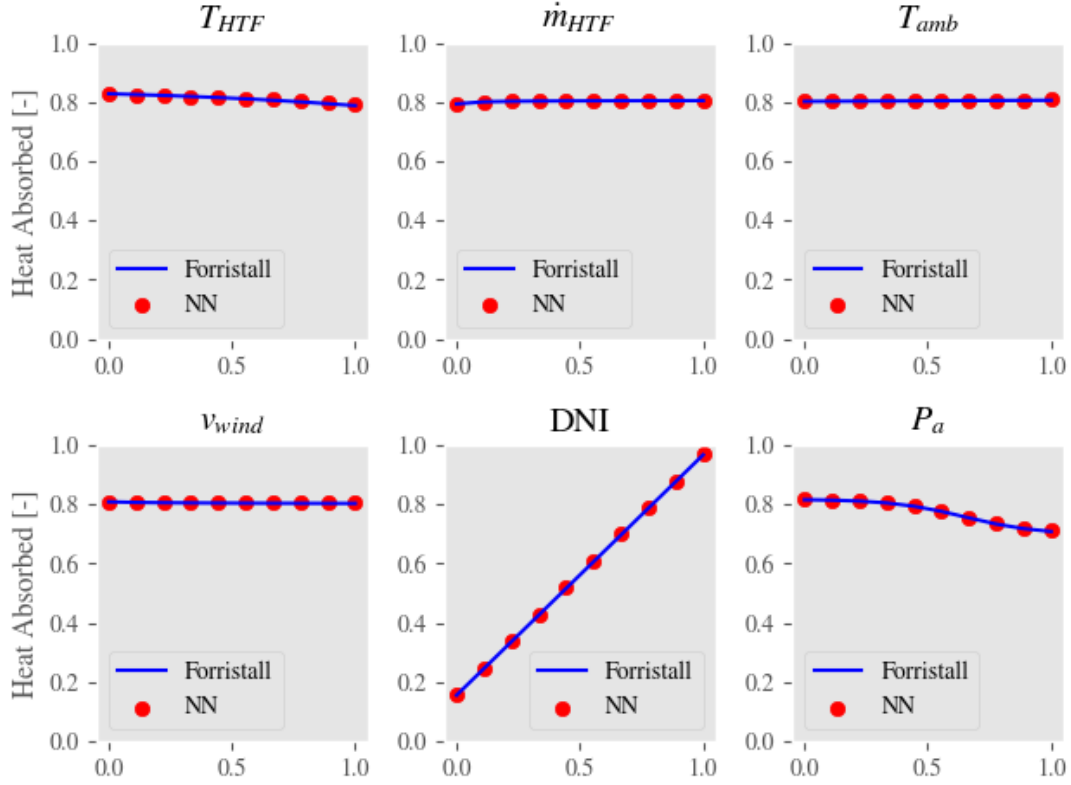


Figure 4.2: Parameter sweep investigating heat absorption computed with neural network and Forristall model

Figure 4.2 illustrates that no single input significantly skews the accuracy of the model. Consistent with the previous study, substantial agreement between the Forristall model and the neural network exists for each parameter sweep. As expected by the laws of physics, DNI and heat absorbed by the HTF are highly correlated. However, besides DNI, the two variables that most significantly affect heat absorption are the HTF temperature (T_{HTF}) and the pressure of hydrogen in the annulus (P_a). This outcome is expected since both the temperature of the HTF and pressure in the annulus space are important factors for determining the extent of heat loss to the environment. As these variables increase, more heat is lost to the surroundings thereby reducing the heat absorbed by the HTF. For a more detailed visualization of this behavior, Figure 4.3 depicts the heat absorption as a function of the HTF temperature and hydrogen pressure in the annulus using the

baseline inputs from Table 4.3.

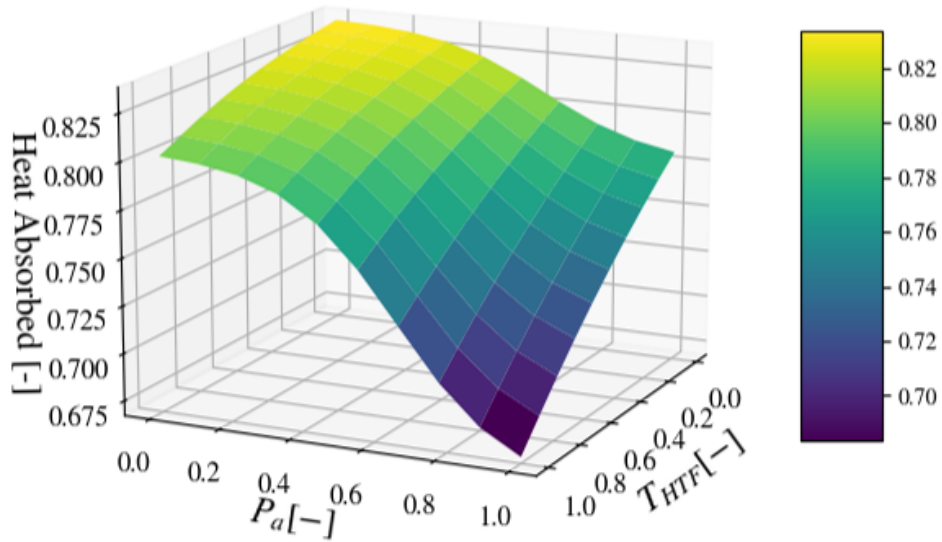


Figure 4.3: Heat absorbed by the HTF as a function of HTF temperature and hydrogen pressure in the annulus

While Figure 4.3 illustrates the heat absorbed by the HTF, it effectively communicates heat loss dynamics. Since the same amount of energy is incident upon the system, less heat absorbed implies more heat is lost to the environment. It is important to note that as the hydrogen pressure in the annulus increases, the heat loss also increases. However, for the region of normalized hydrogen pressures from 0.0 to 0.3, increasing hydrogen pressure does not have a significant impact on heat loss. Within this same range, the HTF temperature also is less impactful in regard to heat loss. Interestingly, above the normalized hydrogen pressure of 0.3, heat loss increases exponentially, and HTF temperature is increasingly influential. For example, at the maximum hydrogen pressure in the annulus, HTF temperatures cause about a 10% variation in heat loss. This interesting behavior suggests that plants may be accumulating hydrogen in their receivers long before they notice it.

The complex dynamics between the HTF temperature/hydrogen pressure and the heat absorption are well captured by the neural network. Figure 4.4 presents the absolute

value of the difference in the normalized heat absorbed using the Forristall model and the neural network approach over the same surface illustrated in Figure 4.3.

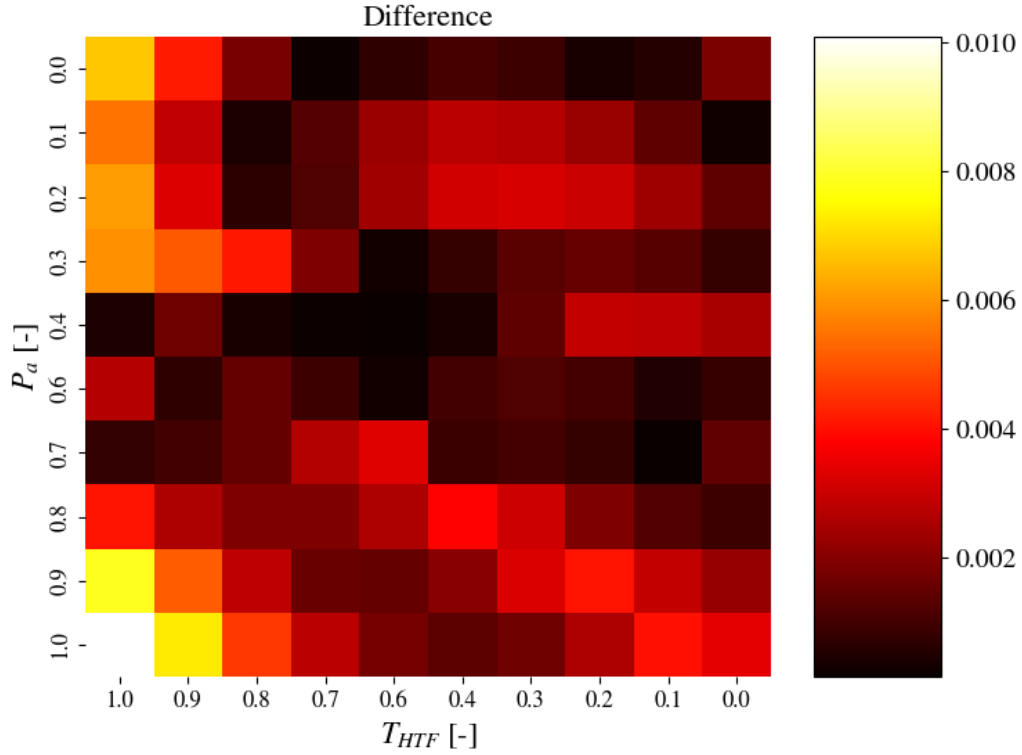


Figure 4.4: Absolute difference in heat absorbed by the HTF using the Forristall model and the neural network approach while varying HTF temperature and hydrogen pressure in the annulus

The agreement between the neural network predictions and the Forristall model is again excellent with the worst discrepancy being only 1% of the maximum heat absorption. This slight deviation is likely due to the training approach which used a linear distribution of data points across the entire parameter space. In regions with high variability in heat absorption, particularly at high HTF temperatures and annulus pressures in this case, increasing the density of training data points could improve accuracy. This approach would mirror the use of adaptive time-stepping in simulations where smaller steps are taken when the rate of change increases in magnitude.

Lastly, it is important to mention that this endeavor was driven by the desire to reduce

the runtime of the solar field loop thermal model. A loop, discretized into 120 control volumes, was simulated in FORTRAN using an implementation of the Forristall model and simulated again with the neural network approach. Per timestep, the Forristall model approach took 8.0 milliseconds, and the neural network approach took 0.059 milliseconds. Thus, the neural network approach is more than 100x faster per loop while maintaining the capability to simulate various operating conditions accurately. This runtime reduction is incredibly important for scaling up to a large solar field with hundreds of loops.

4.1.2 Solar Field Sector

To validate the solar field sector model, 15-minute time-averaged operational data from Solana is utilized. The measurements include HTF flow rate through the sector, inlet HTF temperature to the sector, outlet HTF temperature of the sector, and weather/solar conditions. As a reminder, Figure 4.5 presents a satellite image of a solar field sector at Solana. The HTF data, such as flow rate and temperature, were measured at the inlet/outlet of the sector indicated by the black circle, and DNI was measured in the corner of the solar field near the yellow circle. The sector under consideration is comprised of 104 loops connected by inlet and return headers. Thus, for the model to be accurate, it not only needs to capture the thermal behavior in all the loops but it also must capture the transient behavior of the HTF as it flows through the headers.

Using the data provided, three different days of operation were simulated. The simulation utilized 2-second time steps, and discretized each solar field loop into control volumes 4 meters in length. At each time step in the simulation, the weather data, sector flow rate, and inlet temperature to the sector from the operation data were used as inputs for the model. As a function of these inputs, the model then computed the outlet temperature of the solar field sector throughout the day. The three days used for model validation included a cloudy March day, a cloudy December day, and a sunny April day.



Figure 4.5: Sector visualization with DNI measured at the yellow dot and HTF measurements at the black dot

Optical properties, thermal capacitance multipliers, and receiver states in the field were tuned to match the response on the sunny April day. This sunny day was chosen for parameter tuning because it introduces no uncertainty in solar availability across the sector. The tuned values of these parameters are presented in Table 4.4.

Table 4.4: Parameters obtained from fine tuning the simulation results

Parameter	Value
η_{track}	0.97
η_{soil}	0.84
ρ_m	0.92
C_{loop}	2.0
C_{header}	1.25
Prob_pristine	0.98
Prob_brokenGlass	0.005
Prob_lostVacuum	0.015
n_H2	3
H2_std-i	0.25
H2_mean-i	-3

Recall from Section 3.2.2, C is a thermal capacitance multiplier that accounts for the thermal inertia of the piping. Thus, C_{loop} is the multiplier used in the solar field loop thermal model, and C_{header} is the multiplier used for the header thermal model.

Changing these values can dramatically alter the sector's thermal response during highly transient periods such as start up after sunrise and transition to TES before sunset.

Figure 4.6 presents the results from this model validation case study where HTF temperatures at the sector outlet computed by the model (black) and measured during operation (red) are compared for the three separate days.

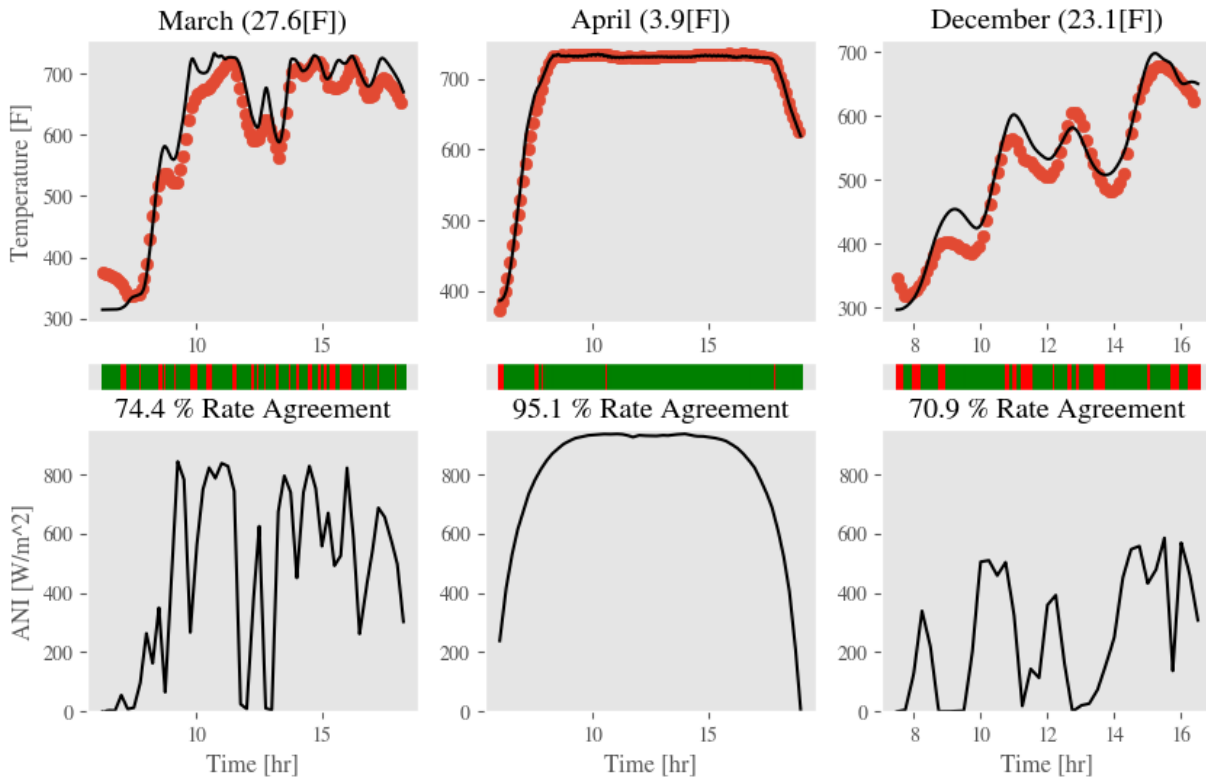


Figure 4.6: Solar field sector responses for three separate days comparing simulated sector outlet temperatures (black) and measured outlet temperatures (red dots)

The labels of each column of Figure 4.6 indicate the month corresponding to the day simulated along with the mean absolute error between simulated and measured sector outlet temperatures. Temperatures are provided in the top plot, and aperture normal irradiance (ANI), the DNI normal to collector, is presented in the bottom plot. Furthermore, the green/red indicator below the temperature plots displays agreement between simulated and measured temperature derivatives. Green indicates that the simulation and measured data have temperature rates (increasing, decreasing, or near

zero) that are aligned. However, a red mark indicates where the respective rate directions differ. For example, near 9:00 AM on the December day, the simulation's temperature increases while the measured temperature decreases. This mismatch in temperature rate is then indicated with red bars. The purpose of this added dimension is to visualize and quantitatively assess whether the simulation matches the temperature trends observed during operation. "Rate Agreement" is defined as the number of green marks divided by the total number of marks. With all days having rate agreement greater than 70%, this case study suggests that the solar field sector model predicts the transient temperature response well even on challenging cloudy days where the DNI likely has high spatial variability across the sector. It is important to note that the model has could theoretically simulate spatially resolved DNI, but this feature has not been implemented in the code yet.

In addition to sector temperature data, Solana also provided defocusing measurements in a sector. Specifically, the defocusing degree for each SCA averaged across all loops in a sector was provided. Because the defocusing control strategy at Solana differs from the strategy used in the model, comparisons were made via total optical efficiency of the loop as opposed to directly comparing defocusing degree measurements. The loop optical efficiency is based on intercept factors. Recall, the intercept value is the same as η_{SCA} from Section 3.2.1. These intercept factors are unique to collector geometries and were computed using SolTrace. The intercept factor curve presented in Figure 3.14 is utilized to map collector defocusing in degrees to a loop optical efficiency value. Loop optical efficiency is computed using Equation 4.2.

$$\eta_{opt} = \frac{1}{n_{SCA}} \sum_{i=1}^{n_{SCA}} f_{IC}(\theta_i) \quad (4.2)$$

In this equation, η_{opt} is total optical efficiency of the loop, θ_i is the average number of

degrees the i^{th} SCA is defocused, and $f_{\text{IC}}()$ is the function that maps degrees defocused to the intercept factors computed using SolTrace. The optical efficiency has a minimum of 0% when all of the SCA's are defocused and has a maximum of 100% when zero SCA's are defocused and the collectors have perfect optical properties. In the model, and in reality, geometric imperfections of the collector ensure η_{opt} will never equal 100%. Figure 4.7 illustrates the loop optical efficiency computed from the operational data (red) and simulation (black) for the sunny April day originally presented in Figure 4.6.

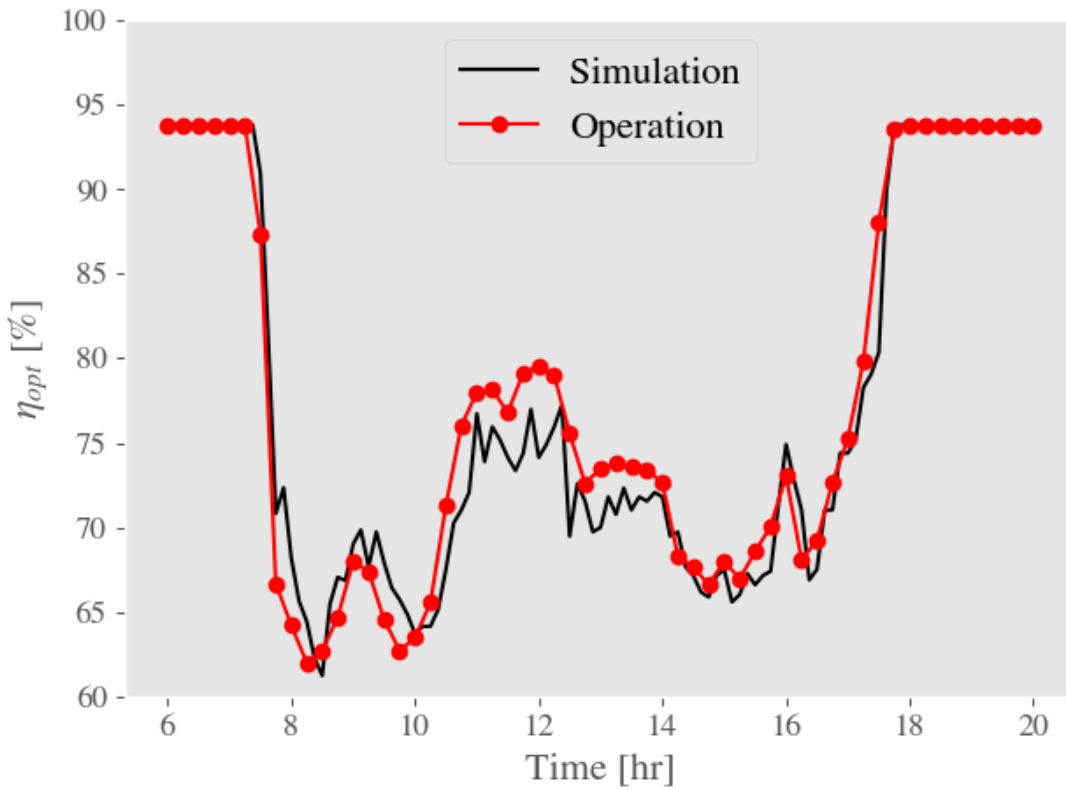


Figure 4.7: Intercept metric comparing heat rejection via defocusing for the model (black) and operation (red)

On this sunny day, consistent agreement between the loop optical efficiency values computed with operational defocusing and simulation defocusing is observed. Coupling this result with the accuracy of sector outlet temperature from Figure 4.6 suggests that the thermal behavior of a sector is remarkably captured by the model. Thus, not only can the model be used to predict sector outlet temperatures for training purposes, but it can

also be utilized as a tool to optimize solar field control strategies.

4.2 Case Studies

This section presents case studies that illustrate new capabilities of this solar field model. First, a study focuses on a user-controlled solar field start up. This study demonstrates the pump cavitation prediction along with the transient, coupled behavior operators can expect to experience at a plant. The next case study leverages the model's feature that allows users to account for hydrogen permeation in receivers. This study highlights the detrimental impact of hydrogen-induced heat loss. Lastly, a study presents the utility of this model for enhancing plant design and operational strategies by looking at plant energy production as a function of HTF flow rate through the solar field.

4.2.1 Solar Field Start Up

A solar field TRNSYS model, representing a plant similar to Solana, couples the hydraulic solver with the solar field sector thermal models presented in the previous chapter. The complete solar field model is comprised of eight solar field sectors, ten HTF pumps operating in parallel, runner pipes to deliver and return HTF from the sectors, sector control valves, bypass control valves, central plant control valves, and an expansion system. The eight solar field sectors are grouped into the near and far field sectors where the far field sectors are the four sectors furthest away from the plant. To demonstrate the ability of the model, a simplified interactive graphical user interface (GUI) was developed to allow users the ability to try starting up the plant after sunrise. The GUI is presented in Figure 4.8.

The challenge operators face each day is to make the correct set of decisions that result in hot HTF being sent to the central plant to be used for steam generation as quickly as possible while taking care to not exceed the HTF fluid temperature limit. With this model,

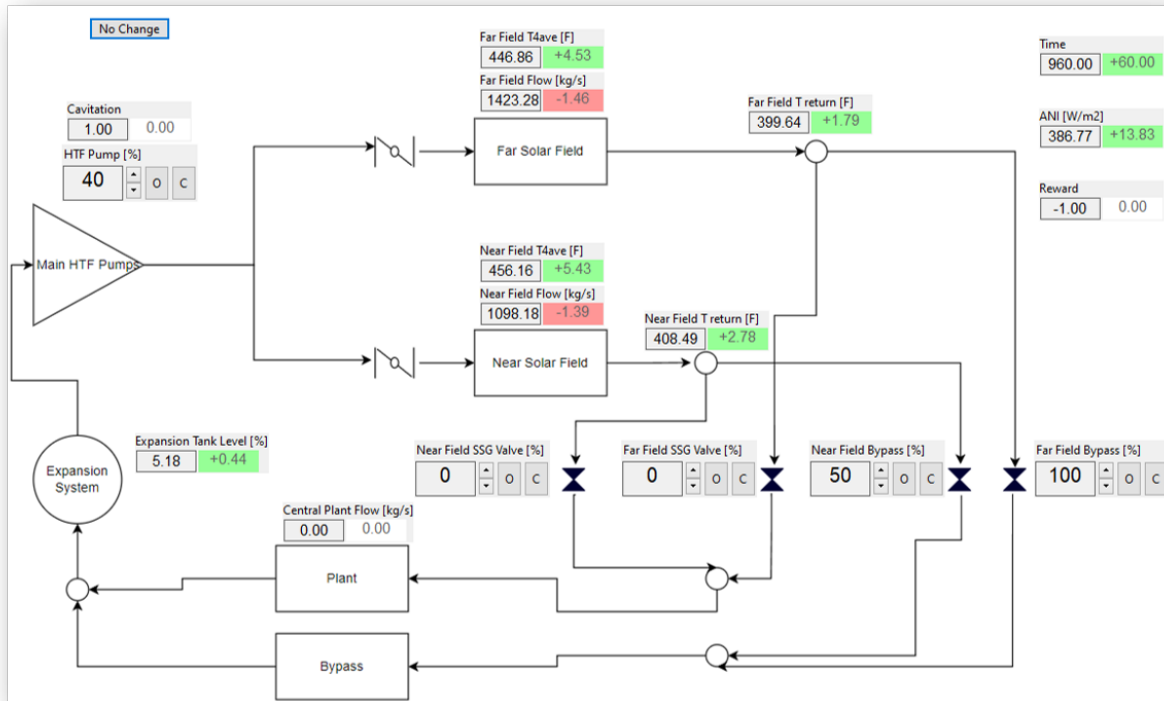


Figure 4.8: Preliminary interface for an operator-controlled daily start-up

the user has the ability to control the plant by adjusting pump speeds and valve positions. While this tool has illuminated some of the difficulties associated with starting up the solar field portion of a CSP plant, it is also useful for illustrating the realistic capabilities of the solar field model. For example, at the moment in the simulation captured in Figure 4.8, the control valve from the near field sectors to the bypass is closed to 50% while the valve from the far fields to the bypass is 100% open. Due to this configuration of valve positions, the flow to the near fields is lower than the far fields. Because the near fields have less flow, the average temperature of the HTF at the 4th SCA and the rate of temperature increase in the near fields is greater than the far fields. Furthermore, because HTF from the far field physically has a longer distance to return to the central plant, the temperature of the HTF arriving at the central plant from the near fields is greater than the HTF returning from the far fields. Lastly, although the level in the expansion tank is at 5.18% and increasing, the pump speed is sufficiently high such that the pumps are

cavitating (indicated by 1 in the Cavitation box). In this scenario, the operator should immediately decrease the pump speeds. By reducing the pump speed, the total flow delivered by the pump will decrease, and the corresponding net positive suction head required will also decrease to a level required to stop pump cavitation.

While the user-friendly graphical interface might initially give an impression of simplicity, it hides the intricate details that operate behind the scenes. This model simulates the entire solar field, divided into eight sectors, with each sector being simulated individually. Within each sector, every loop is also accounted for. As an example, consider the case that occurs during most start ups where a "cold slug" of HTF is sent through the solar field. Figure 4.9 presents the response to this cold slug for one of the eight sectors.

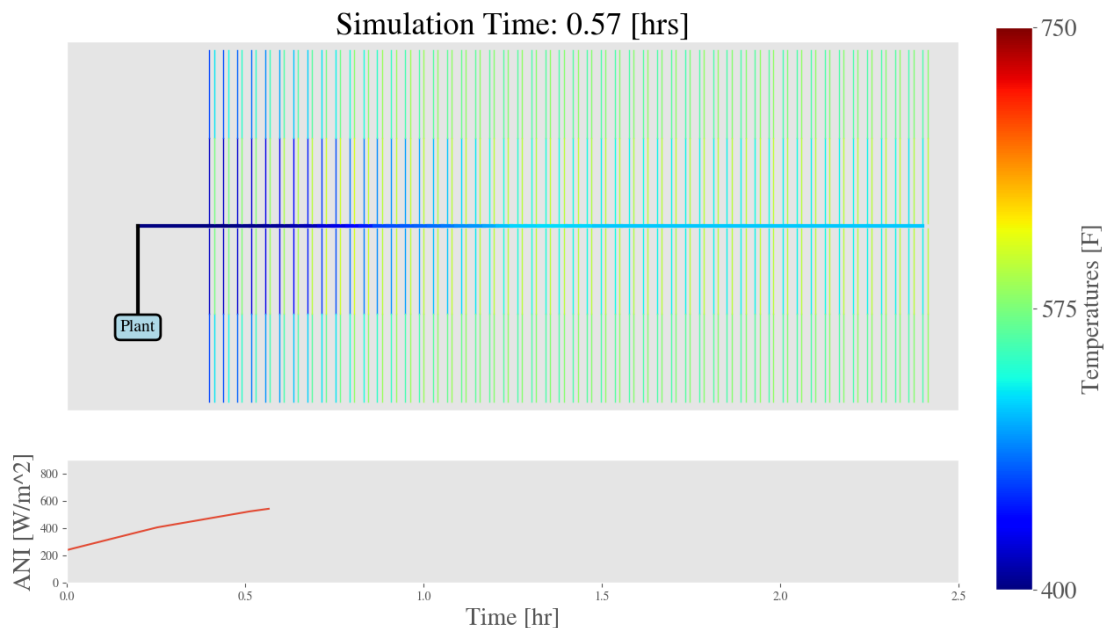


Figure 4.9: Solar field sector temperature distribution during start up with cold slug moving through it

This figure illustrates the inlet header and all solar field loops within a specific sector. The temperature of the HTF throughout the sector is represented through color coding as indicated by the color bar. At this moment in the simulation, a cold slug is gradually

advancing through the sector. Loops further from the sector inlet remain unaffected and exhibit higher temperatures. However, loops nearer to the inlet experience a flow of colder fluid. This visualization highlights the complex dynamics within the sector that cannot be adequately captured by a single numerical value. Commonly, sector temperatures are summarized by temperature at the 4th SCA averaged across all loops. However, this simplification overlooks the actual temperature distribution.

As mentioned previously, the model captures the complex interplay between the temperature of the HTF in the field, expansion tank level, and pump speeds to predict pump cavitation. Operators often want to increase pump speeds during startup to decrease the time it takes for HTF to return to the central plant from the solar field. However, increasing pump speeds prematurely can lead to pump cavitation. Particularly, this occurs due to inadequate level in the expansion tank which results in low net positive suction head available. Figure 4.10 illustrates a sequence at the beginning of a startup where a user attempts to increase pump speeds and consequentially experiences cavitation.

Cavitation is indicated in the top plot with a value of 1. Initially, the pump speed increased from 10% to 20% without inducing cavitation. However, at the four-minute mark, the speed is increased to 30% which causes the pumps to cavitate. Once the speed is reduced to 20%, the cavitation resolves. Subsequently, the user waits for the HTF temperature in the solar field to increase which results in HTF expansion. This volumetric expansion increases the level in the expansion tanks thereby raising the net positive suction head available to the pumps. At ten minutes, the user again increases the pump speed to 30%, but cavitation still occurs. Only after the expansion tank level reaches about 9%, providing sufficient inlet pressure, does the cavitation stop. This scenario highlights an important lesson during operator training and could be useful for preventing potential damage to the pumps during actual plant operations.

Furthermore, additional complex behavior is observed at plants like Solana that have

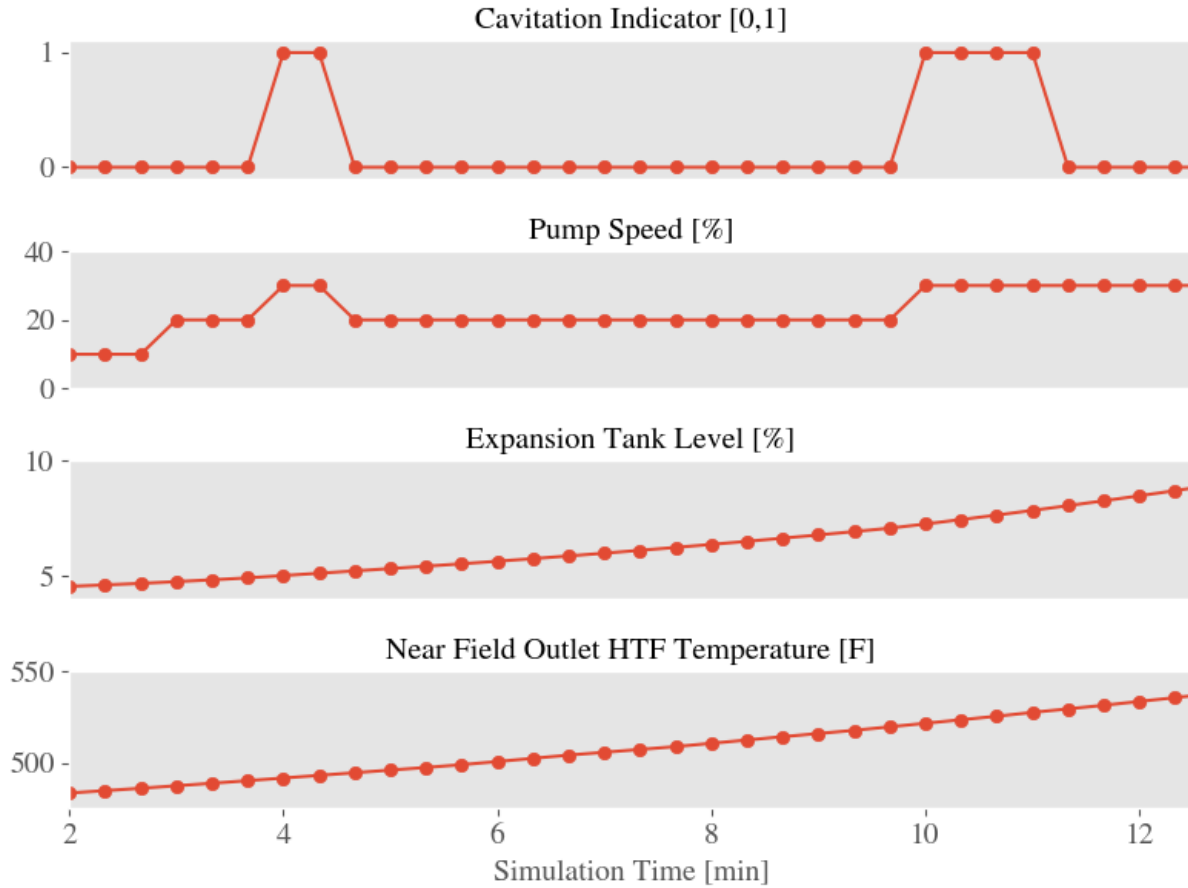


Figure 4.10: User-induced pump cavitation due to insufficient level in the expansion tanks during a startup sequence

very large solar fields. Specifically, a time lag between the temperatures of HTF recorded in the solar field and those at the central plant exists. This lag is due to the physical distance the HTF must travel which in the case of the far field sectors can be miles. Figure 4.11 presents the average temperature of the 4th SCA in the far and near sectors along with the temperatures of the HTF that has returned back from these sectors to the central plant.

The top plot illustrates that the temperatures of HTF in the solar field are nearly the same in the far and near fields due to similar inlet temperatures during the startup period and identical solar resource. However, the temperature of the HTF returning from the near field to the central plant is consistently hotter than that from the far field due to the

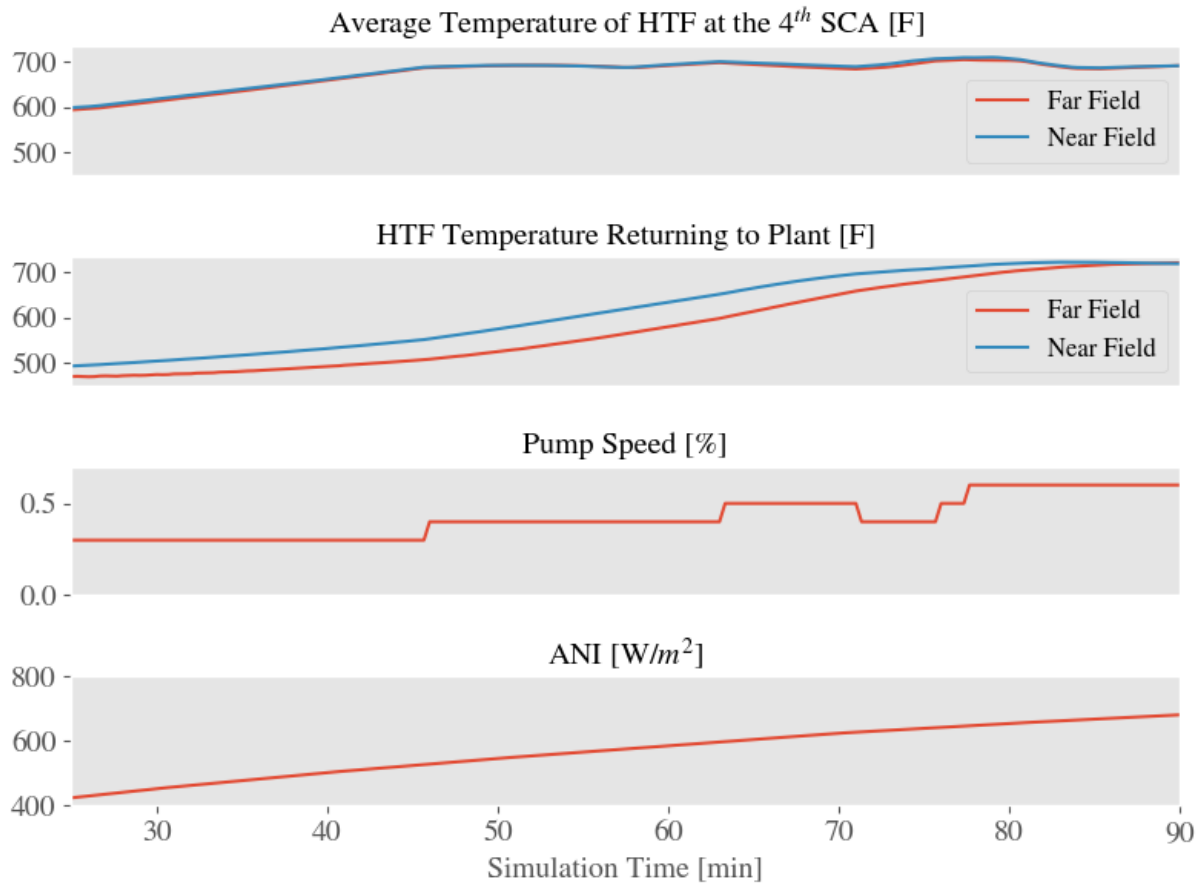


Figure 4.11: Time lag of HTF returning from the solar field back to the central plant

extra distance the HTF must travel. This lag is an interesting phenomenon that operators at a large CSP plant must contend with. However, after about 90 minutes, the solar field consistently returns HTF at the temperature limit, and the lag disappears. In addition to the time lag, the model realistically responds to mass flow rate changes. At about 65 minutes, increasing the pump speed to 50% causes the temperatures of the HTF in the solar field (top plot) to decrease. Then, decreasing the pump speed to 40% at 70 minutes causes the HTF temperature to increase. This rise and fall in temperature is due to the residence time the HTF has in the solar field loops. With lower flow rates, the HTF takes longer to travel through the loop which allows for more heat absorption and results in higher temperatures. Conversely, higher flow rates has the opposite effect. This demonstrates that a plant's main control mechanism for achieving optimal HTF outlet

temperatures is the HTF flow rate.

As a final note, while this complete solar field model has a high level of detail and captures realistic transient effects observed at Solana, it takes around 50 milliseconds to solve per time step (2 second time steps). If 1 second time steps were used, the model would run about 20 times faster than real time.

4.2.2 Hydrogen Variation

The purpose of this study is to utilize the model's ability to alter receiver conditions for each loop of a solar field sector in order to evaluate daily sector performance as a function of hydrogen permeation. In this study, the operational data from the April day from the validation study was repeatedly simulated while varying amounts of hydrogen in the receivers throughout the solar field. Specifically, two variables regarding hydrogen permeation are altered: the percentage of receivers in the fourth/last SCA with hydrogen and the pressure of hydrogen in the annulus space. To quantify the performance of the sector, total heat loss was computed using a sector energy balance resulting in Equation 4.3.

$$\dot{W}_{\text{loss}} = \sum_{i=1}^{n_{\text{loop}}} w_{\text{ap}} L_{\text{loop}} (\text{DNI}_{\text{avail},i} - \text{DNI}_{\text{defocus},i}) + \dot{m}(h(T_{\text{inlet}}) - h(T_{\text{outlet}})) \quad (4.3)$$

Here, n_{loop} , w_{ap} and L_{loop} are the number of loops in the sector, collector width aperture and length of a loop respectively. Furthermore, $\text{DNI}_{\text{avail},i}$ is the incident energy upon loop i accounting for optical effects while $\text{DNI}_{\text{defocus},i}$ is the solar energy rejected due to defocusing of loop i . Lastly, $\dot{m}$ is the mass flow through the sector and $h()$ is the enthalpy of the HTF evaluated at the inlet of the sector and the outlet of the sector. Equation 4.3 is evaluated at each timestep in the simulation. To obtain a daily heat loss

metric, the heat loss is integrated over the day.

$$\text{Daily Heat Loss} = \int_{t_0}^{t_{\text{end}}} \dot{W}_{\text{loss}} dt \quad (4.4)$$

Finally, the daily heat loss values are normalized by subtracting the daily heat loss of a sector that has zero hydrogen in any of its receivers. Thus, this final heat loss value reports the additional heat loss due to hydrogen in the receivers. The additional daily heat loss values of a solar field sector for various levels of hydrogen are presented in Figure 4.12.

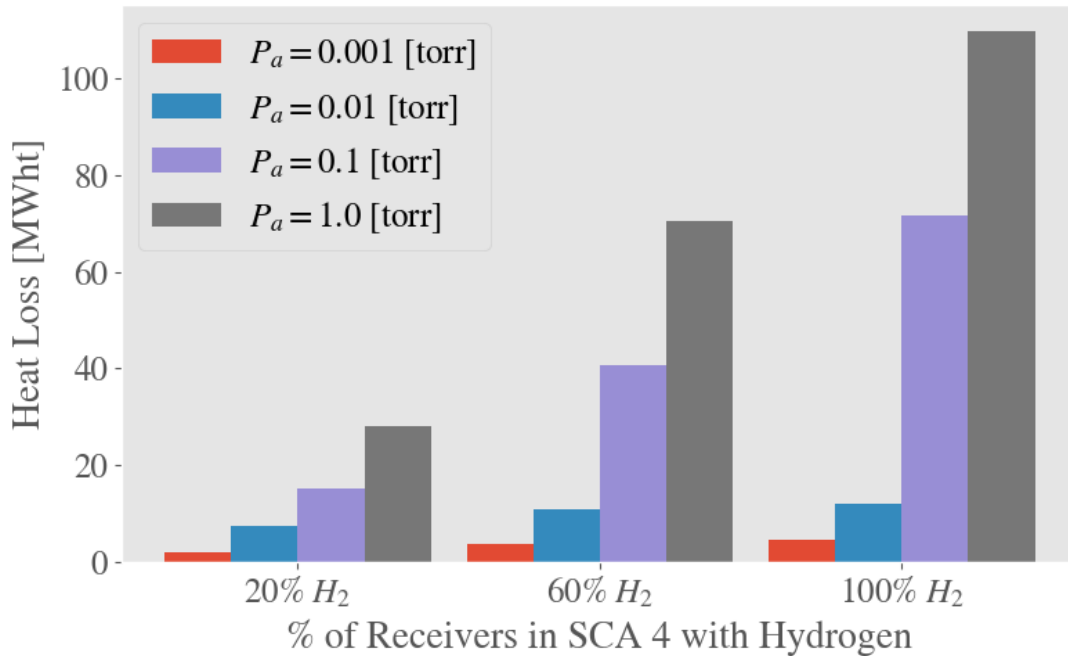


Figure 4.12: Additional heat loss of a solar field sector as a function of hydrogen pressure in the annulus (see legend) and percentage of receivers in the last SCA with hydrogen permeation

As expected, the additional daily heat loss differs quite significantly with the pressure of hydrogen in the annulus and with the number of receivers that have hydrogen. Interestingly, receivers with hydrogen a pressure greater than 0.1 [torr] appear to have a considerable increase in heat loss. This could be due to the the heat transfer mechanism through the annulus space changing from free molecular heat loss to a transition region

consisting of free molecular and continuum heat losses [37]. Furthermore, considering the scenario where 60% of the last SCA contains hydrogen pressurized to 0.1 [torr], an additional heat loss of 40 [MWht] is observed. Recall, this heat loss is only for one sector. If this amount of hydrogen permeation was representative of all sectors in the solar field, the plant would be losing 320 [MWht] of heat each day solely due to hydrogen. This corresponds to just under 10% of the total electric power during this April day, thereby emphasizing the devastating effects of hydrogen permeation in receivers.

4.2.3 Sector Flow Rate Variability

This final case study investigates a sector's performance where the primary objective is to explore the tradeoff between HTF outlet temperature and defocusing losses by varying mass flow rates from a baseline operational case. This study uses the operational data from the April day once again. Multiple simulations were repeated where the only variable changing was the mass flow rate into the sector. Flow rates were adjusted with Equation 4.5.

$$\dot{m}_{\text{new}} = (1 + Z) \cdot \dot{m}_{\text{operation}} \quad (4.5)$$

At each timestep, the sector model takes in the operational mass flow rate adjusted by the parameter Z . Thus, the operational mass flow rate profile for the day is modified uniformly by Z . Three new flow rates were originally tested with Z equal to -0.1, 0.1, and 0.3. Therefore, in the new simulations the flow rates were reduced by 10%, increased by 10%, and increased by 30% respectively. Two key optical results from these simulations merit attention. The first is the average loop optical efficiency, η_{opt} , computed using Equation 4.2, and the second is the total number of loops defocused within the sector. These two metrics are reported in Figure 4.13 as a function of time.

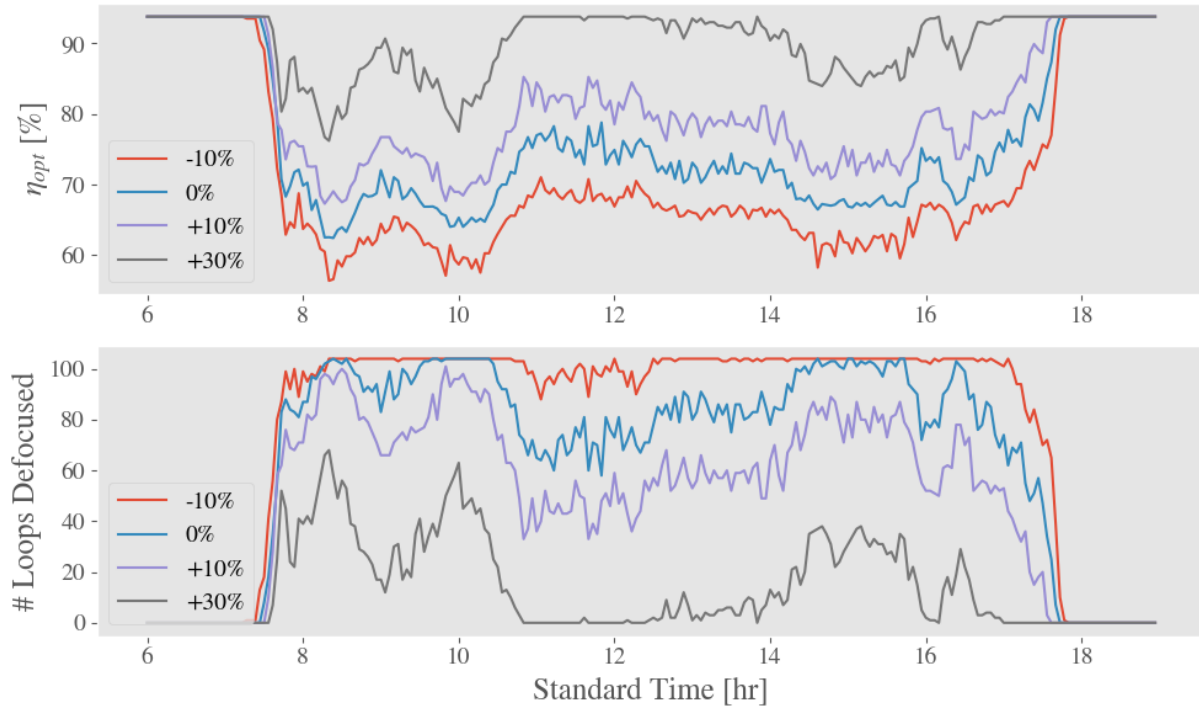


Figure 4.13: Optical efficiency (top plot) and number of loops defocused in the sector (bottom plot) for simulations with varying mass flow rates

In this figure, the red line corresponds to the simulation with mass flow reduced by 10%, the blue line corresponds to the simulation with the baseline mass flow from operation, the purple line corresponds to the simulation with mass flow increased by 10%, and the grey line corresponds to the simulation with the mass flow increased by 30%. The top plot presents the average loop optical efficiency throughout the day, and the bottom plot presents the number of loops in the sector that are defocused. Recall, the maximum loop optical efficiency is about 93% for these specific SCA's, and the sector has 104 loops that can defocus. As illustrated in Figure 4.13, these two metrics are inversely correlated. The correlation is expected because the average loop optical efficiency inherently decreases as the number of loops with defocusing increases.

Two noteworthy observations are drawn from this figure. First, the overall loop optical efficiency increases when the HTF mass flow rate increases. This happens because, at higher flow rates, the HTF in the solar field loops has less residence time in the loop to

absorb heat resulting in a lower outlet temperature. However, even with the flow rate increased by 30%, some defocusing still occurs throughout the day indicating that the outlet HTF temperature exceeds its limit.

Second, observing a sector where all solar field loops defocus simultaneously is rare. If mass flow rates were equally distributed through the loops, this observation would likely be quite common. However, in this study, the flow rates are distributed such that 95% of loops have flow rates within $\pm 3\%$ of the average flow. This flow variability leads to some loops overheating due to lower flow rates and others having reduced outlet temperatures because of higher flow rates. Ultimately, this realistic distribution in flow rates produces variations of HTF outlet temperatures. This phenomena is illustrated in Figure 4.14.

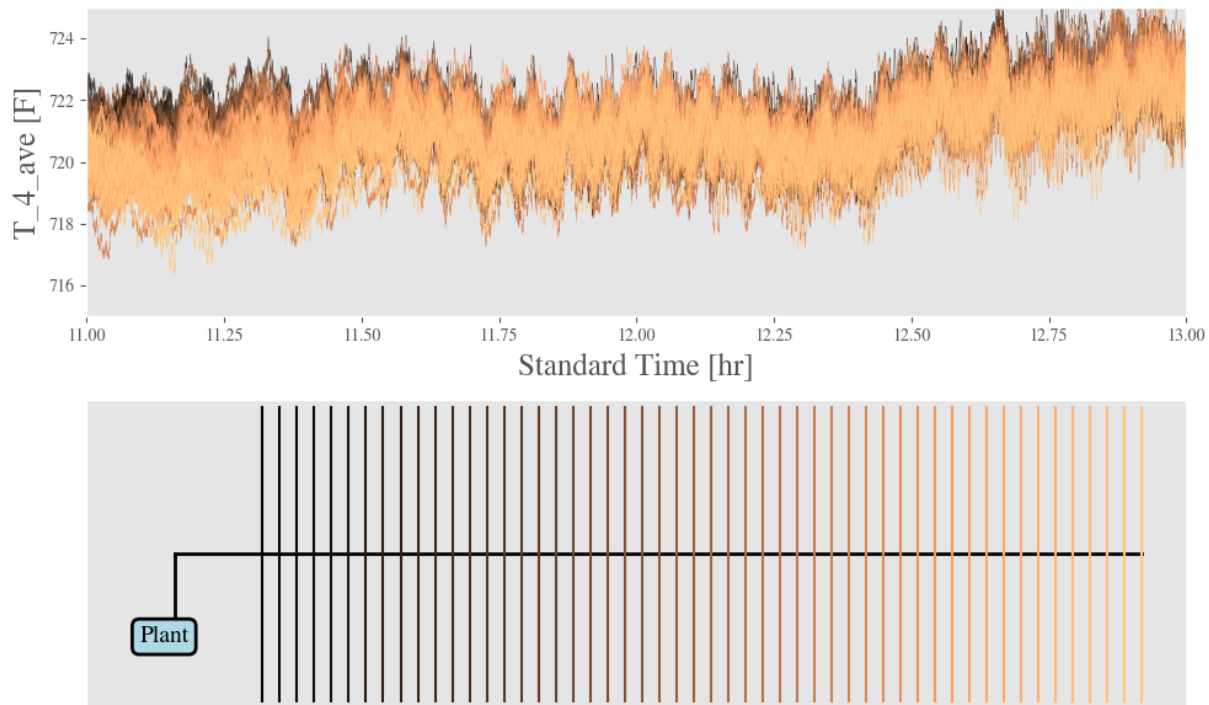


Figure 4.14: Temperature of 4th SCA for all loops in a sector (top plot) with colors in reference to loops drawn (bottom plot)

Here, the top plot presents the HTF temperature at the center of the 4th SCA of each loop in the sector. The color coding corresponds to the loop location shown in the bottom plot. The temperature distribution has a range of about 6 degrees Fahrenheit. Ultimately,

this range of outlet temperatures, resulting from the mass flow rate distribution, plays a crucial role in generating some of the realistic defocusing effects depicted in Figure 4.13.

Next, attempting to achieve high HTF outlet temperatures affects energy collection due to defocusing and increased heat loss from the receivers. Both issues ultimately lead to reduced power production but do so through different mechanism. This model is well positioned to investigate the tradeoff between the two losses in the context of net power production. Therefore, the final step of this analysis is to determine the HTF flow rate that maximizes net electricity generation during this particular day of operation. Note, this study evaluates the potential increase in net power production by balancing defocusing losses with exergetic losses. However, it is important to remember that practical operational concerns influenced the selection of the original mass flow profile obtained from the plant. This analysis quantifies the performance of several hypothetical approaches to enhance both current plant operations and the design of future solar fields.

To begin this analysis, Equation 4.6 computes the total heat absorbed by the sector. The absorbed heat, $\dot{W}_{\text{sector}}$, depends on the HTF mass flow rate, $\dot{m}_{\text{HTF}}$, and the difference of HTF temperatures at the inlet and outlet of the sector.

$$\dot{W}_{\text{sector}} = c\dot{m}_{\text{HTF}}(T_{\text{HTF,out}} - T_{\text{HTF,inlet}}) \quad (4.6)$$

Here, c represents the specific heat of the HTF at the mean temperature within the sector. While total heat absorbed is highly relevant, the ultimate goal is to maximize the net electricity generation which can be calculated using Equation 4.7. $\dot{W}_e$ is the net electrical power output, and it accounts for the efficiency of the power cycle, η_{plant} , along with the power consumption of the pumps, $\dot{W}_{\text{pump}}$.

$$\dot{W}_e = \dot{W}_{\text{sector}} \cdot \eta_{\text{plant}} - \dot{W}_{\text{pump}} \quad (4.7)$$

The efficiency of the power plant, η_{plant} , is estimated using the Carnot efficiency modified by a factor b which reflects real-world deviations from ideal performance.

$$\eta_{\text{plant}} = b \cdot \left(1 - \frac{T_{\text{cold}}}{T_{\text{hot}}}\right) = \left(\eta_{\text{plant,ref}} \cdot \left(1 - \frac{T_{\text{cold,ref}}}{T_{\text{hot,ref}}}\right)^{(-1)}\right) \cdot \left(1 - \frac{T_{\text{cold}}}{T_{\text{hot}}}\right) \quad (4.8)$$

The Carnot efficiency is a function of the hot and cold temperatures that are inputs to the power cycle. In this case, the cold temperature is assumed to be constant, and the hot temperature corresponds to the temperature of the HTF at the sector outlet which is subject to change. Furthermore, the multiplier factor, b , is computed by assuming the power cycle has an efficiency of about 35.6% ($\eta_{\text{plant,ref}}$) when the input to the cycle is HTF at 730 [F] ($T_{\text{hot,ref}}$). Using this assumption, the multiplier factor can be computed. Then, for future calculations, the multiplier remains constant and can be used to evaluate the efficiency of the power cycle for various sector outlet temperatures.

Additionally, the power required for the pump operation, computed with Equation 4.9, is a function of the pressure drop across the sector, ΔP_{sector} , the volumetric flow rate of HTF, $\dot{Q}_{\text{HTF}}$, and the pump efficiency, η_{pump} .

$$\dot{W}_{\text{pump}} = \frac{\Delta P_{\text{sector}} \dot{Q}_{\text{HTF}}}{\eta_{\text{pump}}} \quad (4.9)$$

The pump efficiency, defined as the fraction of mechanical power converted from electrical power, is assumed to be constant despite the fact that pump efficiency varies with different pump speeds. This assumption is based on the fact that plants would select

pumps that have a high efficiency in the expected range of operational speeds. Again, the results primarily serve as a reference for future plant designs rather than for immediate operational strategies.

The simulation of the same day in April was repeated with various flow rates ranging from -10% to $+80\%$ of the operational profile. Figure 4.15 illustrates the sector outlet temperatures of the HTF throughout the day for these varying flow rates.

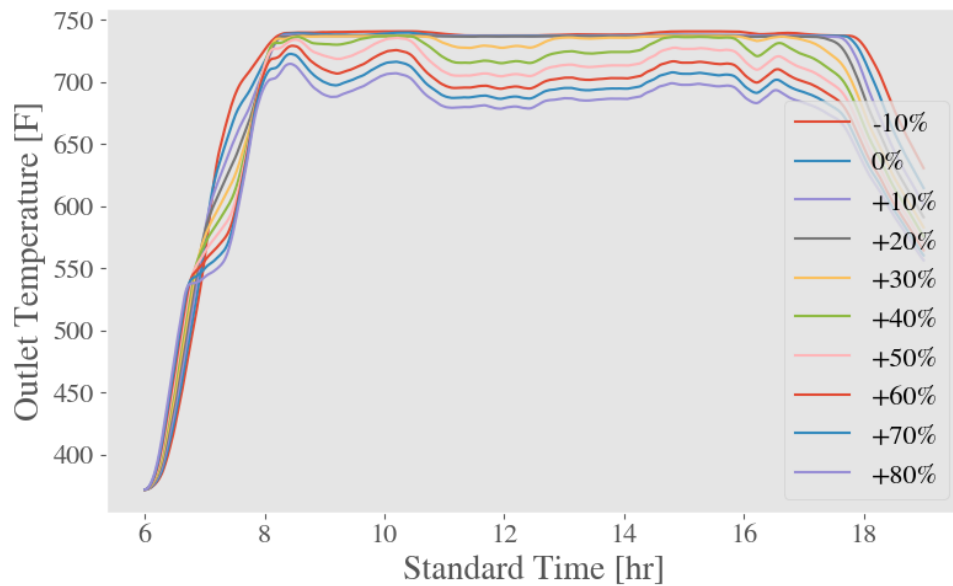


Figure 4.15: Outlet temperatures of HTF at different flow rates

Notably, flow rates between -10% and 30% exhibit similar temperature profiles during the midday period. This supplements findings from Figure 4.13, which shows that even at 30% above the operational flow rate, loops in the sector experience defocusing. Due to this defocusing, temperature is controlled to the temperature setpoint and excess energy is dumped. However, for flow rate profiles greater than 30% , the HTF rarely reaches the temperature setpoint. In fact, for the flow profile that is 80% higher than operation, the HTF temperature rarely exceeds 700 [F].

To assess which flow rate strategy is more advantageous, Figure 4.16 presents various metrics related to net electricity generation for the different flow rates simulated.

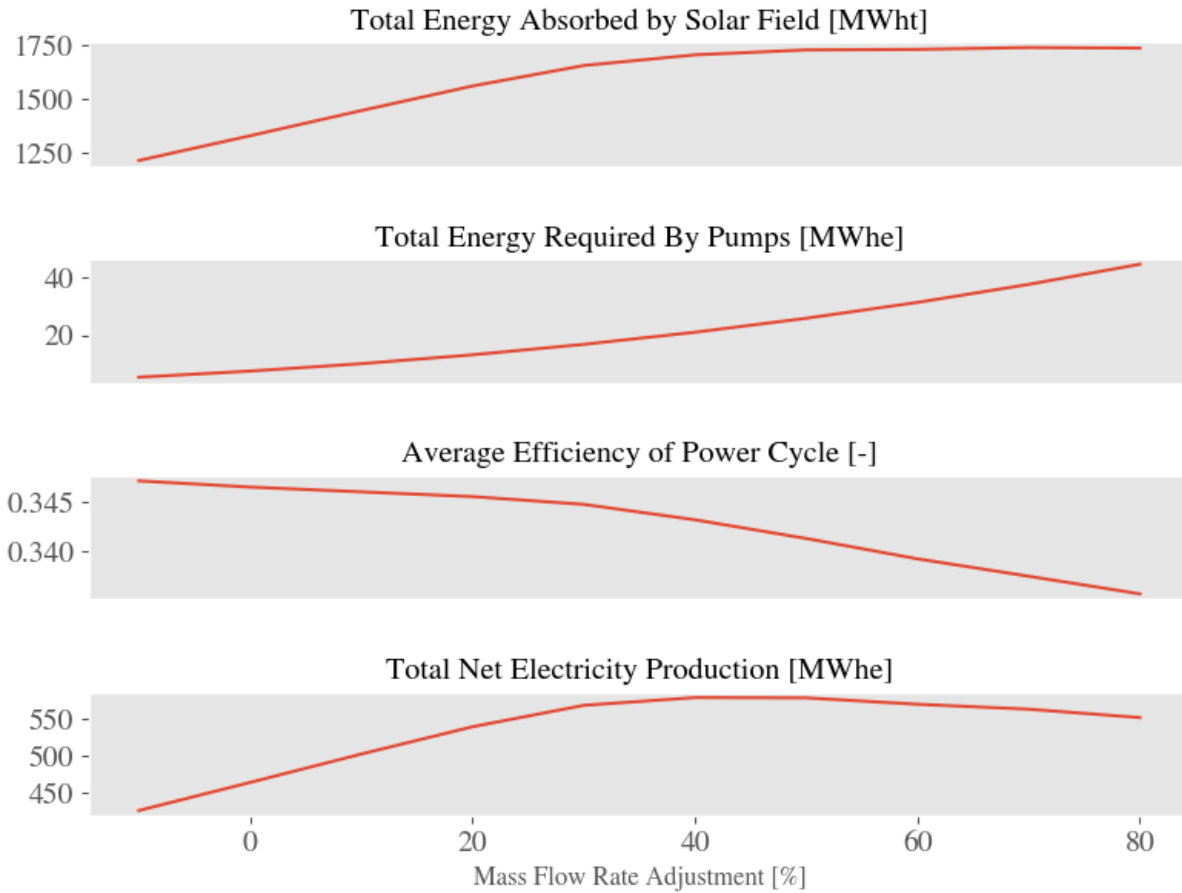


Figure 4.16: Impact of varying flow rates on solar field performance parameters

There are many interesting items to note here. Starting with the top plot, as the flow rate increases to 30%, heat absorbed by the solar field increases significantly. This observation aligns with the outlet temperatures presented in Figure 4.15 and Equation 4.6. In this region of flow rates, the outlet temperature remains constant due to the defocusing. Because of this, the temperature differential remains the same in Equation 4.6. However, the mass flow through the sector is increasing. This increase in flow rate then results in an increase in heat absorption since the two are proportional given an unchanging temperature differential.

On the other hand, for flow rate profiles above 30%, the heat absorption begins to taper off. While the mass flow continues to increase, the decreasing outlet temperature reduces

the magnitude of the temperature differential between the inlet and outlet. These two factors effectively cancel each other out, and the heat absorption remains nearly constant for flow rate profiles greater than 50%.

Next, the net electricity generation peaks when using a flow rate 40% above the operational profile. With this flow rate profile, an optimal balance exists between the heat absorbed by the solar field, the efficiency of the power cycle, and the power consumed by the pumps. Although flow rate profiles above 40% do slightly increase the heat absorbed by the solar field, these gains are offset by the power required to operate the pumps. Notably, the primary inefficiency stems from pump power rather than a decrease in power cycle efficiency due to reduced sector outlet temperatures.

A critical insight from this study is that higher flow rates through the solar field are advantageous for net electricity production even when outlet temperatures are below the HTF temperature setpoint. At flow rates 40% and 50% above the profile used during operation on this day, the HTF temperature seldom reaches the defined target of 730 [F]. Note, the specific values of 40% and 50% are not particularly relevant in this finding as there are numerous reasons why Solana may have been running HTF slower during this specific day. However, the conclusion that running higher flow rates to achieve lower HTF temperatures is relevant. Even with the lower temperatures, the net electricity generated is substantially greater than that generated with lower flow rates where HTF temperatures meet the target. Moreover, targeting a lower HTF temperature by increasing HTF flow, approximately 715 [F] instead of 730 [F], can significantly reduce defocusing. As a potential benefit, lowering the HTF temperature setpoint could also decrease the generation of hydrogen in the receivers. Less hydrogen in the receivers would improve an aging solar field's efficiency and could potentially extend the plant's operational lifetime. Also, in the framework of this study, the power cycle efficiency varies within a narrow range of 1.2%. This variation represents 21 MWht of the 1750 MWht absorbed by the

solar field which is a relatively small amount compared to the 420 MWht lost due to defocusing at the baseline flow rate. Therefore, designing thermal oil trough plants to achieve lower outlet temperatures through the mechanism of higher HTF flow rates can decrease defocusing and potentially hydrogen permeation, increase heat absorption by the solar field, and ultimately boost the net electricity output of the plant which would lead to increased profitability.

5 SOLAR TRANSFORMER ASSISTANT NEURAL NETWORK (STANN)

The primary motivation of this work is to develop a sophisticated tool to aid in training operators. Following the use of the plant start-up GUI in the previous chapter, there was interest in creating an artificial intelligence (AI) tool. Inspired by the 'Hint' button found in many chess applications, which suggests optimal moves, the vision was to create a similar feature. This tool would recommend actions believed to yield beneficial outcomes in the plant's operations. Rather than creating a simple rule-based AI, the goal was to develop a dynamic AI. This AI would be capable of learning the intricacies of any plant's operations over time through interaction and observation. This concept led to the creation of the Solar Transformer Assistant Neural Network (STANN). The architecture of STANN is presented in Figure 5.1.

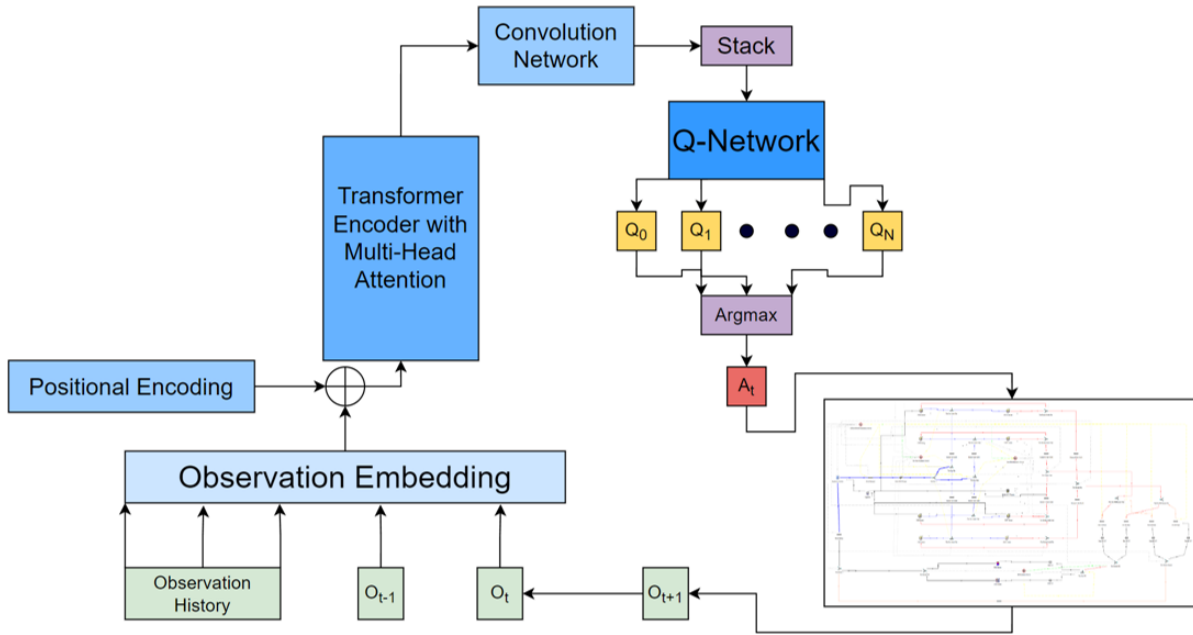


Figure 5.1: Architecture of the Solar Transformer Assistant Neural Network

STANN is categorized within the reinforcement learning (RL) modeling framework. In this context, STANN engages with a simulated environment, specifically the solar field model, in order to learn an optimal policy. This policy is comprised of a set of actions

aimed at maximizing cumulative rewards. The reward function should be designed such that it promotes favorable outcomes for the plant. Essentially, STANN is designed to identify the most beneficial action based on the current and historical states of the plant. Unlike previous CSP applications of RL that focused on single state decisions [25] [24], this work incorporates sequences of both current and past observations. This sequential approach is advantageous for several reasons. First, CSP plants are transient by nature; decisions such as increasing pump speed can affect HTF temperatures with a delay due to the time required for the HTF to circulate from the solar field back to the plant. Many other operational decisions exhibit similar delayed effects. Secondly, this method mirrors human learning more closely by including memory in its system management. For example, it would be incredibly challenging for a human to start up a plant without recalling recent events.

Furthermore, STANN utilizes the transformer encoder architecture [26]. This architecture is a powerful framework implemented in advanced tools such as ChatGPT [27]. Transformers are known for their exceptional ability to learn dependencies within sequences which has led to breakthroughs in understanding complex sequences such as the English language. It is anticipated that this architecture will similarly decode the relationship between past and present measurements of CSP plants. Additionally, a key factor in selecting the transformer architecture is its scalability. They can be stacked on top of one another where each new layer enhances the model's comprehension.

This research is still in its early stages and is not yet ready for deployment, but it does exhibit significant potential. This chapter will dive into the architecture of STANN in detail and present preliminary results from a case study that explores the model's ability to aid operators with training.

5.1 Methodology

For many mechanical engineers, machine learning may initially appear to be a foreign discipline. However, at its core, machine learning extensively uses linear algebra and calculus. These are math areas that are fundamental to any engineering research. This methodology section is designed to demystify machine learning by detailing each step of the STANN architecture.

5.1.1 Q-Learning

Before diving into the architecture used to develop STANN, it is important to provide a brief introduction to Q-learning, the reinforcement learning algorithm framework employed in this research. Q-learning is a model-free reinforcement learning algorithm that directly interacts with an environment to learn an optimal policy. It is particularly suited for modeling Markov Decision Processes (MDPs) which are defined by states (s), actions (a), and rewards (r). The primary objective of Q-learning is to derive a policy that maximizes the expected value of the total reward over any and all successive steps. This objective is mathematically captured by the Bellman equation. It formulates the expected value of the total reward as a recursive function which allows the algorithm to update its estimates based on immediate rewards and future potential benefits.

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(s', a') \quad (5.1)$$

The Q value, $Q(s, a)$, represents the expected cumulative reward for choosing an action a in state s and following an optimal policy after that action. It is calculated as the sum of the immediate reward, $r(s, a)$, received upon taking action a in state s and the maximum expected future rewards. The future rewards are estimated as the Q-value of the new state, s' , associated with the action, a' , that maximizes the Q-value. The recursive

definition of the Q-value essentially allows the agent to project future rewards out to the end of the simulation. This future reward is discounted by a factor, γ , which allows the model to weigh the importance of immediate versus future rewards. Essentially, the Q-value indicates the total expected reward an agent can receive starting from state s and taking action a .

In this work, the state is defined by a historical sequence of measurements. For example, Equation 5.2 presents a representative state consisting of a current observation and two preceding ones.

$$s = [m_2^{i-2}, m_1^{i-2}, m_2^{i-1}, m_1^{i-1}, m_2^i, m_1^i] \quad (5.2)$$

In this example, m_j^z represents the j^{th} measurement at time step z . Measurements could be temperatures, flow rates, valve positions, or other relevant variables that characterize the state of a CSP plant. Traditionally, Q-learning uses a tabular approach that requires discrete actions and discrete states. For example, in a simplified model, an environment might have 100 discrete states with an agent capable of making one of 5 possible actions. This would result in the Q-values filling 100×5 matrix where rows represent states and the columns represent actions. However, many real-world environments are described by continuous state values which make this tabular approach intractable. To manage these continuous states, deep-Q learning was developed. This algorithm uses neural networks to map the continuous state inputs to outputs that are Q-values for potential actions. This technique was first applied by researchers to achieve super-human levels of performance in various Atari video games[38].

Similar to [30], the approach utilized in this work enhances traditional deep-Q learning by integrating the the transformer encoder into the model's architecture. This adaptation leverages the transformer's ability to learn dependencies within sequential data that

results in an improved understanding of the state. However, the core objective is the same in that the architecture concludes with n_{action} (the number of possible actions) output nodes where each node represents the Q-values corresponding to the possible actions for the given state input to the model.

5.1.2 Preprocessing

Preprocessing is a crucial step in all machine learning applications, and it serves as the foundation for effective model training and performance. In the transformer architecture, preprocessing typically involves transforming input data from its original form into tokens. For instance, in language applications, tokenization involves mapping each word to an integer that serves as input to the model. In the case of this specific application, continuous measurements are similarly mapped to discrete tokens. To illustrate this concept more clearly, consider the following example. The system obtains two measurements at each observation. The state representation in this scenario is a series including two past observations along with the current observations. This sequence is demonstrated in Figure 5.2.

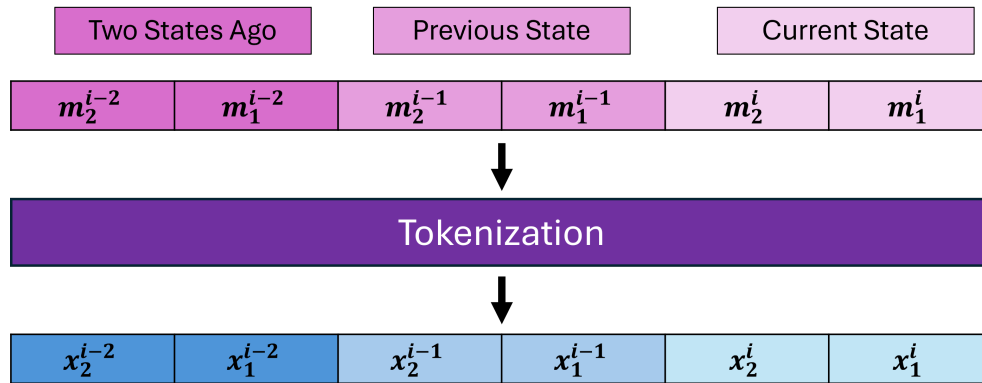


Figure 5.2: Example of taking continuous measurements m and mapping them to tokens x

Each observation in this example includes measurements of variables m_2 and m_1 which, for this example's sake, correspond to HTF mass flow rate and solar field outlet temperature respectively. These measurements are continuous values and must be

converted to integers. This conversion is accomplished by creating a grid for each measurement and determining the grid point nearest to the continuous value. For instance, the Python code snippet below outlines the grid definitions for these measurements along with their corresponding tokens.

Listing 5.1: Mapping from continuous measurement to tokens

```

1
2 # Solar Field Outlet Temperature Grid
3 T_HTF_grid = np.array([500, 550, 600, 650, 700]) #[F]
4
5 # Solar Field Outlet Temperature Tokens
6 T_HTF_token = np.array([0, 1, 2, 3, 4, 5])
7
8 # Mass Flow Grid
9 M_dot_grid = np.array([100, 200, 300, 400]) #[kg/s]
10
11 # Mass Flow Tokens
12 M_dot_token = np.array([6, 7, 8, 9])

```

As an example, if the solar field outlet temperature were 610 [F], the closest grid point would be 600 [F], located at index 2 in the grid array. The corresponding token is then retrieved by accessing the temperature token array at index 2, resulting in a token of 2. Similarly, a mass flow rate of 330 [kg/s] would correspond to a grid point of 300 [kg/s], yielding a token of 8. Note, the user has the ability to define the discretization of these grid spaces in order to add more or less refinement to the preprocessing.

5.1.3 Observational Embedding

With the tokens now defining a state that captures a history of measurements, these tokens are fed into the model. The initial transformation within the model is known as observational embedding. In this process, each token is uniquely transformed into a vector of length n_{dim} . This increase in dimensionality allows the model to capture

and learn more detailed dependencies among the data. The process of observational embedding with n_{dim} equal to 4 is illustrated in Figure 5.3.

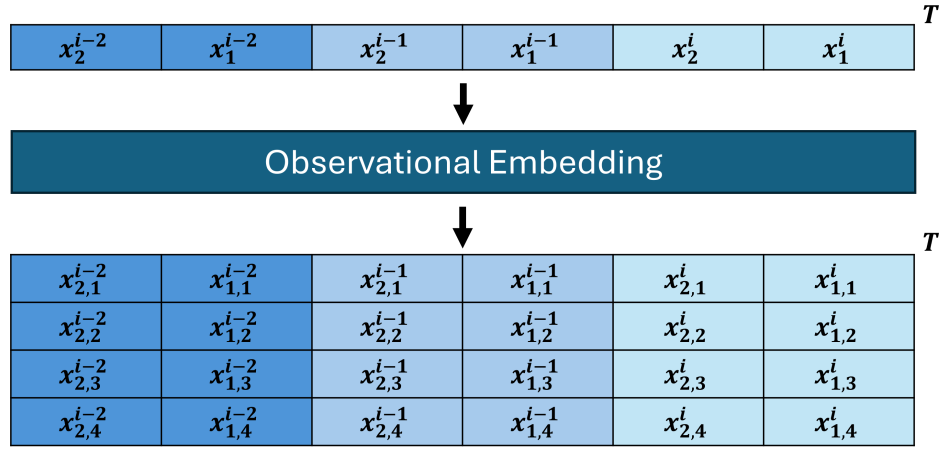


Figure 5.3: Transforming tokens to unique vectors of length $n_{\text{dim}} = 4$ with observation embedding

This transformation results in a matrix of dimensions $n_{\text{sequence}} \times n_{\text{dim}}$. For clarity in presentation, the matrix and vector shown in Figure 5.3 are transposed. A logical question now arises: How are these unique vectors corresponding to each token generated? These vectors are learned through the weights of a linear feed-forward neural network. The input to this network is an array of zeros, except for the index corresponding to the token of interest. The neural network processes this sparse input to produce a vector of length n_{dim} . For instance, Figure 5.4 visualizes how a neural network generates an n_{dim} vector for a token corresponding to an HTF temperature of 700 [F].

Given that the sparse vector has only one non-zero entry, the vector produced by the neural network is composed of the weights that correspond to the token of interest's index. This occurs because the zero entries effectively nullify any contribution from their corresponding weights. Like all neural network weights in this model,

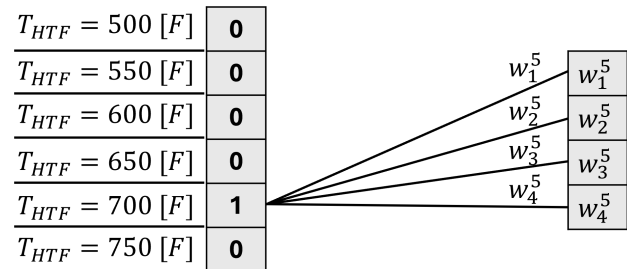


Figure 5.4: Generation of unique vectors for individual tokens using feed-forward neural network

these particular weights are subject to optimization and can be adjusted through the process of gradient descent to improve the model's performance.

5.1.4 Positional Encoding

Now that we have unique vectors representing each token, the next challenge is managing a sequence where the same token may appear multiple times. Without additional information, the model would not be able to discern the temporal position of each token. In other words, the model would not be able to tell whether a token is more recent or occurred earlier in the sequence. Positional encoding addresses this issue by overlaying the observational embedding with a function that assigns a unique signature to each token based on its position in the sequence. Once this positional encoding is applied, even identical tokens are distinguishable based on when they appear in the sequence.

While positional encoding can also be learned, this work employs sinusoidal waves to generate unique encoding for each token in a sequence. This is the same method that was used in the original transformer paper [26]. The positional encoding is uniquely computed for each position in the sequence, pos , and for each dimension, i , in n_{dim} using Equations 5.3 and 5.4.

if i is even:

$$PE(pos, i) = \sin \left(\frac{pos}{1000^{\frac{2i}{n_{dim}}}} \right) \quad (5.3)$$

if i is odd:

$$PE(pos, i) = \cos \left(\frac{pos}{1000^{\frac{2i}{n_{dim}}}} \right) \quad (5.4)$$

Using these equations, a matrix of dimensions $n_{\text{sequence}} \times n_{\text{dim}}$ is generated. This matrix contains the positional encoding, and it is added to the observation embedding matrix. This step effectively combines the unique positional data with the embedded data resulting in an encoded observations matrix with temporal context. This process is visualized in Figure 5.5.

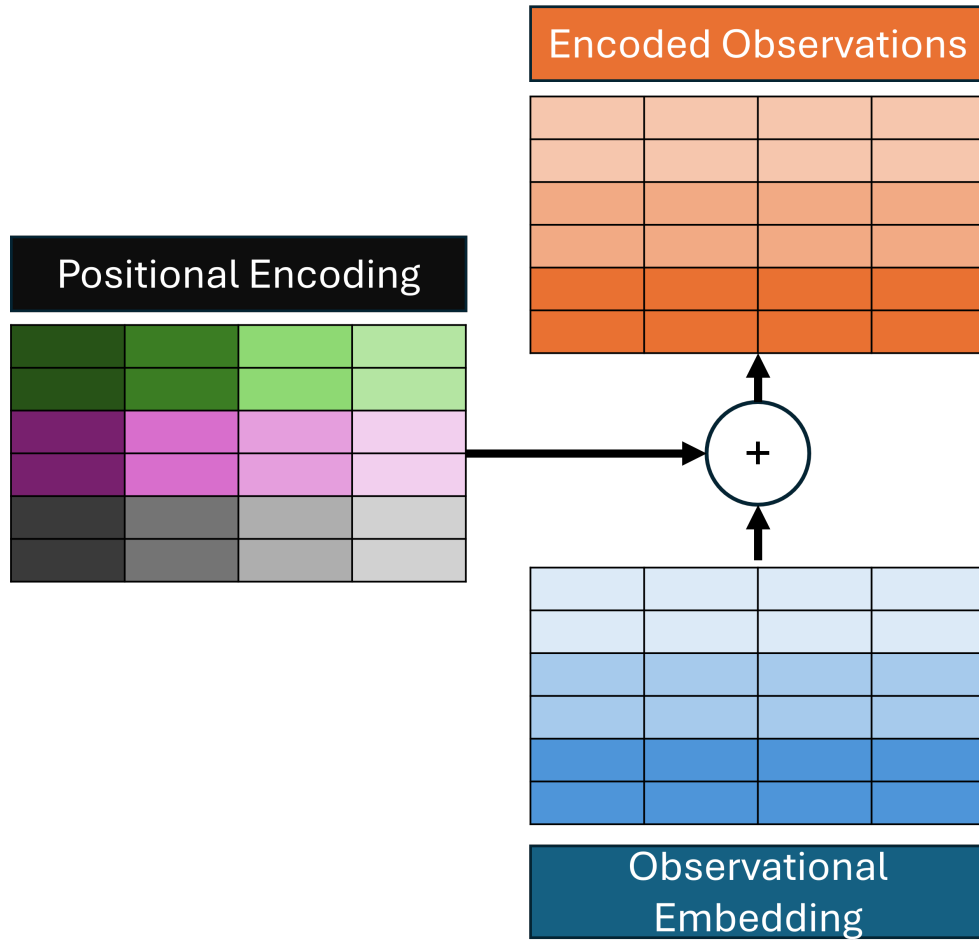


Figure 5.5: Applying positional encoding to the observational embedding matrix

Note that the process outlined here differs slightly from typical positional encoding methods. In this work, each observation in the sequence contains multiple measurements which corresponding to a distinct component of the plant. As a result, these measurements are inherently unique and cannot yield identical tokens. Consequently, the positional encoding is applied uniquely to each observation in the sequence. This ensures that the

model accurately captures the distinct context and timing of each set of measurements.

5.1.5 Transformer Encoder

After all of this preparation, we are finally ready to feed the input into the transformer. The first operation within the transformer is the construction of the Query (Q), Key (K), and Value (V) matrices from the input matrix. These matrices are generated by distinct linear feed-forward networks corresponding to each of the matrices. However, a linear feed-forward network essentially acts as a matrix of weights, meaning that the outputs, Q, K, and V, can be directly obtained through matrix multiplication. This crucial transformation is visualized in Figure 5.6.

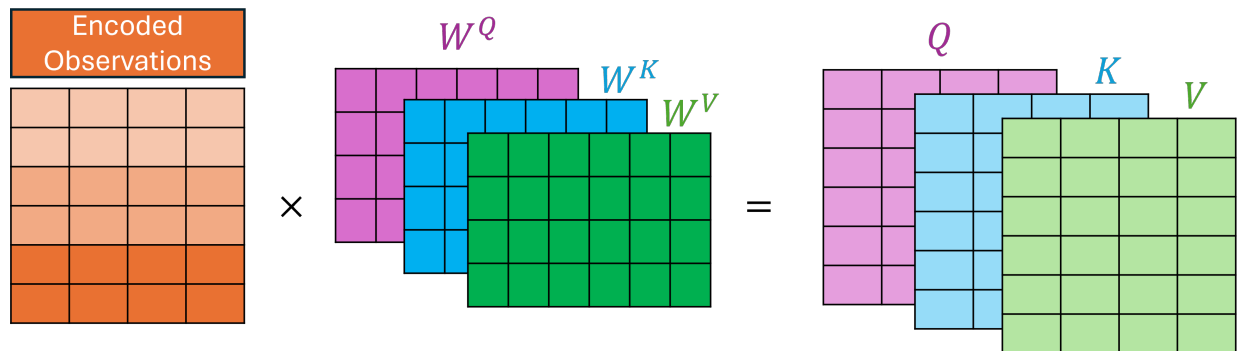


Figure 5.6: Creating Query, Key, and Value matrices from the transformer input using linear feed forward networks

These matrices are foundational to the attention mechanism that will be discussed shortly. Though the concept is abstract (and feels hand-wavy), the intuition behind the Query, Key, and Value matrices can be understood through their roles. The Query matrix contains information about the specific part of the input sequence currently in focus. This is similar to a search query in a database lookup. The Key matrix can be thought of as providing unique identifiers for each element within the input sequence. Lastly, the Value matrix contains the content associated with each input element.

To effectively process different aspects of the information contained in the Query, Key, and Value matrices, these matrices are subdivided according to the number of attention

heads, n_{head} , as specified by the model's architecture. In this work, $n_{\text{dim}}/n_{\text{head}}$ must always be greater than one and be an integer. For example, if n_{head} is set to two, each of the matrices will be split into two new matrices, each with dimensions $n_{\text{sequence}} \times 2$ as illustrated in Figure 5.7.

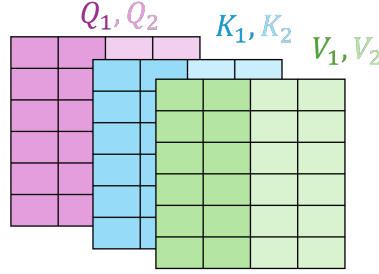


Figure 5.7: Dividing the Query, Key, and Value matrices according to $n_{\text{head}} = 2$

With these subdivided matrices generated, the model applies the self-attention mechanism. Self-attention is a fundamental component for understanding the interdependencies within the sequence. This mechanism operates by multiplying the Query matrix by the transpose of the Key matrix. The product of this multiplication is the self-attention matrix. Figure 5.8 illustrates how the self-attention matrix, S_i , is generated for the i^{th} head.

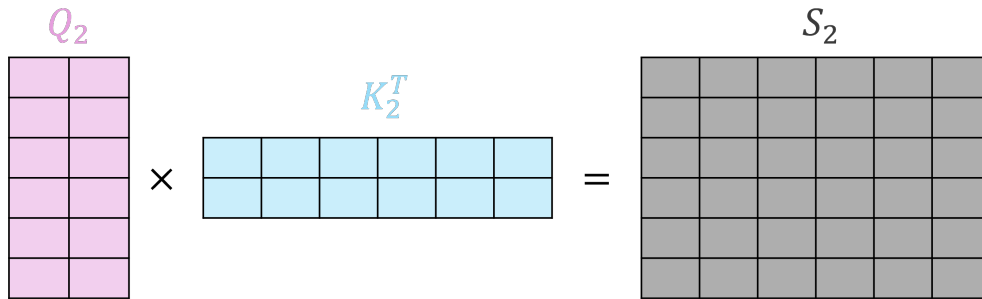


Figure 5.8: Self-attention matrix generated from Query/Key combination

The elements in this matrix, referred to as 'attention scores', indicate the extent to which each part of the sequence impacts other parts. For instance, a high value in the self-attention matrix at row 1, column 6 suggests a strong relationship between the solar field outlet temperature from the current measurement and the mass flow rate measured two observations ago. This matrix of attention scores is crucial as it guides the model to focus

in on the most relevant parts of the input sequence. This process is analogous to human behavior as we continually decide what to focus on in order to make informed decisions. The multi-headed attention mechanism generates multiple self-attention matrices where each of these matrices provide a different interpretation on the input sequence. This allows the model to simultaneously consider various aspects of the sequence which leads to a more comprehensive understanding.

The next step is to convert the attention scores into probabilities. Specifically, the softmax function is applied to each row in the self-attention matrix which results in all values ranging from 0 to 1 with the values across rows summing to 1. Before applying the softmax function, the attention scores are divided by $\sqrt{n_{dim}}$. This is necessary because the magnitude of the multiplication of the Query and Key matrices scales with the dimensionality of the model. Large values are problematic when applying the softmax function because it can lead to small gradients during backpropagation. With the new probability distribution in the self-attention matrix, it is multiplied by the Value matrix for the i^{th} head.

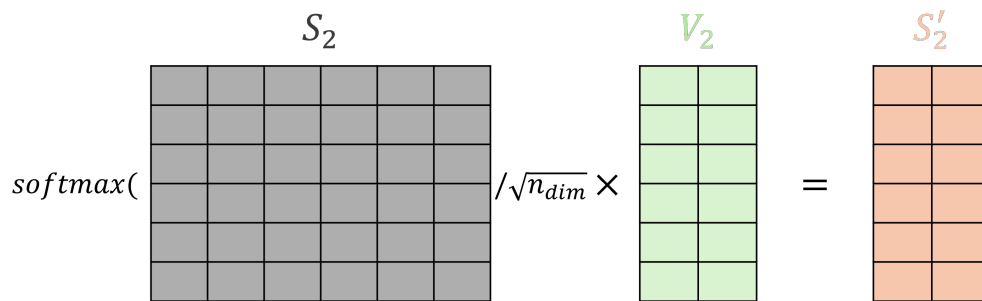


Figure 5.9: Generation of final attention matrix for the i^{th} head

This transformation reevaluates the information provided in the sequence based on the relevance indicated by the probability weights from the self-attention matrix. Essentially, this process emphasizes the key pieces of data in the sequence that the model should focus on which ensures that the most pertinent information is considered in the subsequent steps. Following this, the attention matrices from the various heads are concatenated

to integrate the perspectives they represent. The combined data is further integrated through an additional feed-forward network. This step is presented in Figure 5.10.

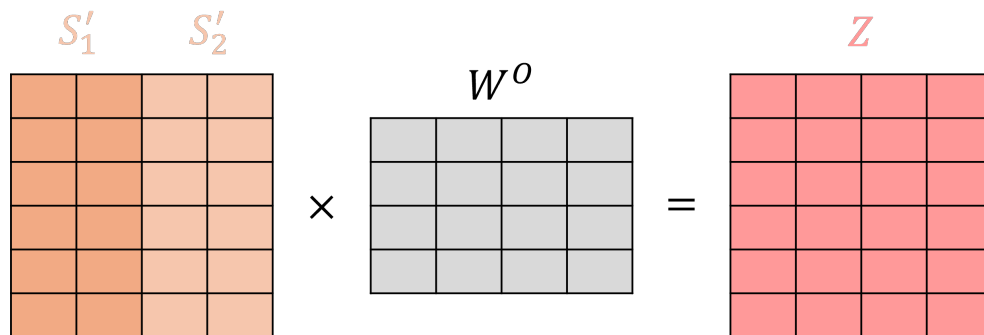


Figure 5.10: Integrating key information learned from each attention head via concatenation and feed forward neural network

We're nearly through with detailing the architecture, but the creators of this model included one more essential step. This final step incorporates a non-linear feed-forward neural network, layer normalization, and residual connections. These components are key to ensuring the model's output is both stable and accurate. Figure 5.11 demonstrates this complex yet vital stage in the transformer structure.

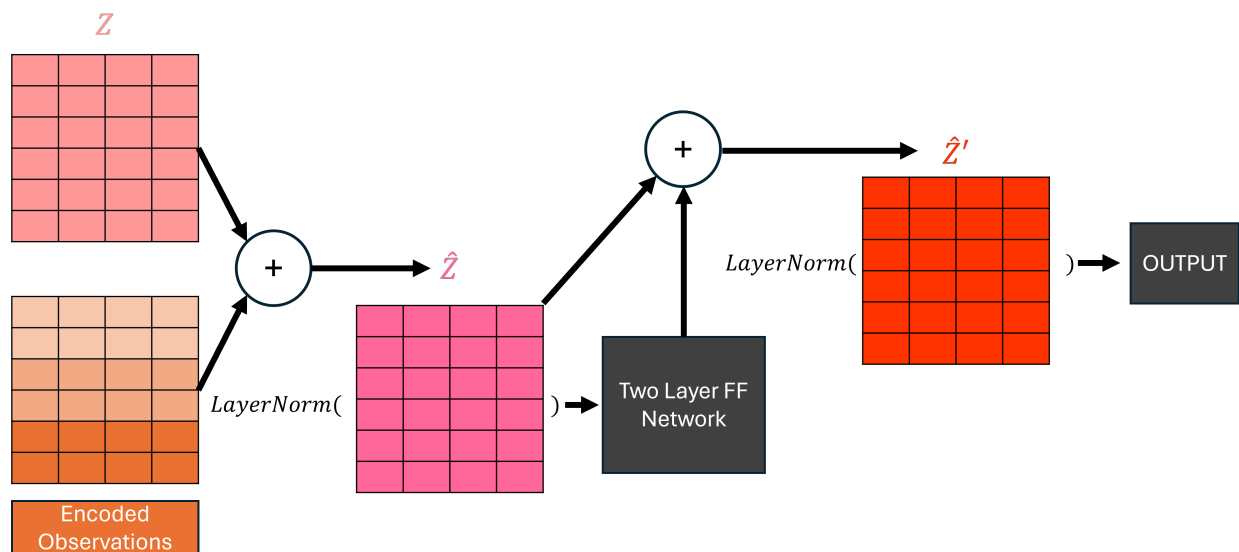


Figure 5.11: INSERT CAPTION

First, the output of the attention mechanism, Z , is combined with the original input that includes the encoded observations. This sum is then subjected to layer normalization

which standardizes each row by subtracting the mean and dividing by the square root of the variance. This normalization step, combined with adding the original input data, is crucial as it has been found to mitigate the vanishing gradient problem and improve learning in deep networks. Following normalization, the matrix is fed through a two-layer feed forward neural network. The first layer utilizes the ReLU activation function to introduce non-linearity to the model. Surprisingly, everything up to this point has used only linear operators. This non-linearity allows the model to apply a complex, learned function to the information that is provided by the attention mechanism. Lastly, the output of this network is added back to its input and normalized again. The resulting matrix represents an attention-based understanding of the sequence history.

5.1.6 Convolutional Neural Network

The integration of a convolutional neural network (CNN) into this model was unplanned, yet anecdotal evidence suggest that it significantly enhances the model's convergence during training. While CNNs are traditionally employed in image processing tasks due to their proficiency in recognizing patterns, in this work, they serve a different purpose. Here, the CNN is essentially used to compress the data output from the transformer encoder.

As the feature dimensions, n_{dim} , and sequence length, n_{sequence} , increase, the number of elements in the transformer output matrix grows exponentially in size. This expansion leads to a corresponding increase in the number of trainable parameters in the Q-learning feed forward network which can hinder effective training due to model complexity. To manage this issue, the CNN compresses the transformer data by applying a kernel. The mechanism of this compression through convolution is illustrated in Figure 5.12.

In CNNs, a kernel slides across the input matrix to perform an element-wise multiplication followed by a sum which simplifies the data into a single output value for each

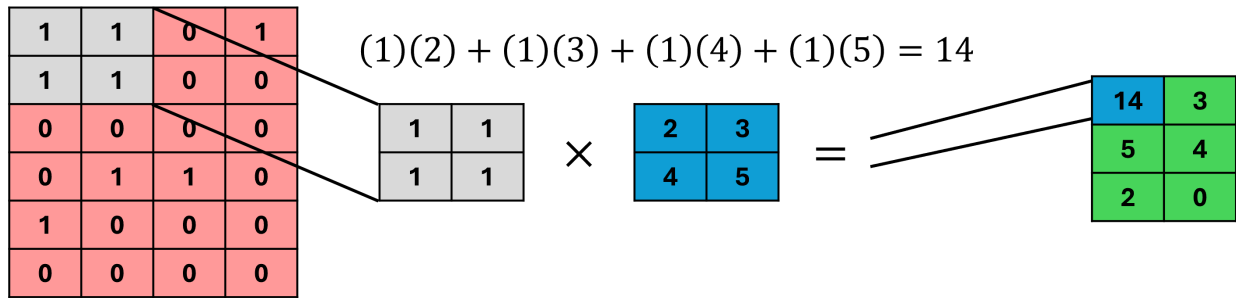


Figure 5.12: Convolutional neural network used to compress data from transformer encoder

position of the kernel. The movement pattern of the kernel is defined by its 'stride'. The stride determines how many rows or columns the kernel skips each time it moves. In this example, the kernel has a stride of two both horizontally and vertically. This means that the kernel shifts two spaces to the right when moving horizontally and two spaces down/up when moving vertically. As a result, the output matrix generated will be smaller and hopefully focused on capturing the most dominant features present in the input matrix.

5.1.7 Final Q-Learning Network

With the information from the transformer now compressed by a CNN, it is time for the decision-making step. The compressed matrix is converted into a one-dimensional vector by stacking its columns vertically. This vector is then fed into a final linear feed-forward network which produces the outputs corresponding to the Q- values for the possible actions that the agent can take. This process is illustrated in Figure 5.13.

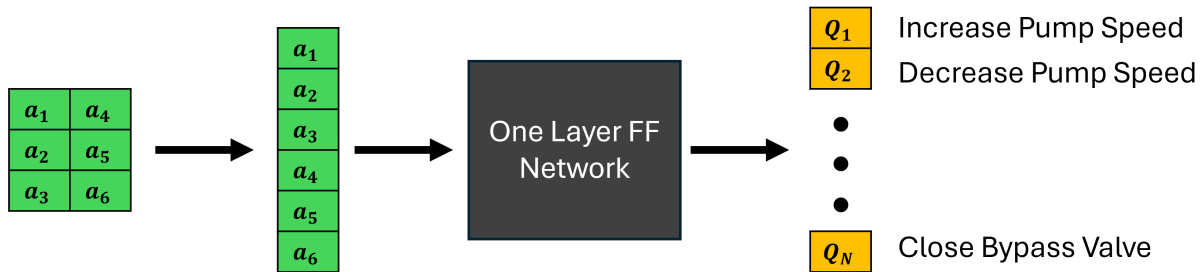


Figure 5.13: Mapping the compressed data from the CNN to Q-values using a linear feed-forward neural network

Recall, each Q -value quantifies the expected cumulative reward associated with selecting a particular action given the current state. Thus, the optimal policy is determined by choosing the action corresponding to the highest Q -value. For example, if the highest Q -value is Q_2 , this would indicate that the best action the agent can take in this scenario is to decrease the pump speed.

5.2 Plant Start Up

To illustrate the capabilities of this architecture and explore training methodologies, consider this case study. In this example, STANN is tasked with learning the optimal procedure for starting up the plant from sunrise on a sunny day. This scenario is similar to the task operators are faced with daily where STANN must make a series of decisions to safely and efficiently send hot HTF to the central plant in order to bring the turbines online. For reference, Figure 5.14 presents the environment that STANN can interact with.

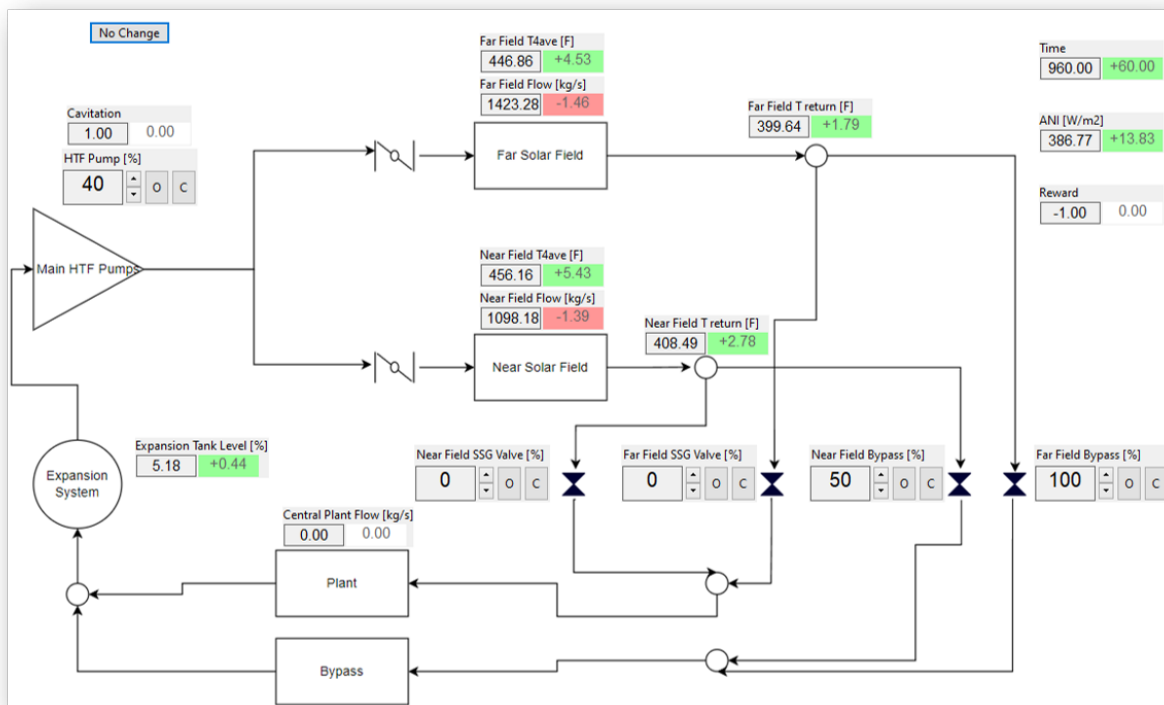


Figure 5.14: GUI for operator-controlled daily start-up

Note although STANN does not interact directly with this GUI, it provides context to the various measurements and actions that make up this problem. Given this environment, it is important to define the various states, actions, and rewards. The state is defined by a historical sequence of measurements. Details about the specific measurements recorded and the number of tokens used to represent each are outlined in Table 5.1.

Table 5.1: Various measurements used to represent the environment's state along with the number of tokens and bounds associated with each measurement

Measurement	Units	Token Lower Bound	Token Upper Bound	# of Tokens
Pump Speed	-	0	1	11
CV Far SF to CP	-	0	1	11
CV Near SF to CP	-	0	1	11
CV Far SF to BP	-	0	1	11
CV Near SF to BP	-	0	1	11
Expansion Tank Level	-	0	1	11
$T_{4,ave}$ Near SF	K	475	667	20
$T_{4,ave}$ Far SF	K	475	667	20
T_{HTF} Return Far SF	K	475	667	20
T_{HTF} Return Near SF	K	475	667	20
ANI	$\frac{W}{m^2}$	0	1000	20

"CV Far SF to CP" represents the position of the control valve that is used to control flow from the far solar field sectors to the central plant for electricity production. Similarly, "CV Far SF to BP" represents the control valve that is used to control flow from the far solar field sectors to the central plant bypass. Next, " $T_{4,ave}$ Near SF" corresponds to the average temperature of the 4th SCA across all the loops in the near solar field sectors. Lastly, " T_{HTF} Return Near SF" represents the temperature of the HTF from the near solar field sectors that has made it back to the plant. Thus, $T_{4,ave}$ and T_{HTF} are correlated, but there can be a significant time delay between the two.

Next, the agent is capable of performing 19 distinct actions. One inherent limitation of this approach is the restriction to discrete actions. However, the ability to act frequently allows the agent to execute complex decisions despite this constraint. The various actions

are summarized in Table 5.2.

Table 5.2: Possible actions that an agent can make

Component	# of Actions	Possible Actions
-	1	No Action
Main HTF Pumps	2	-10%, +10%
CV Far SF to CP	4	-10%, +10%, Fully Close, Fully Open
CV Near SF to CP	4	-10%, +10%, Fully Close, Fully Open
CV Far SF to BP	4	-10%, +10%, Fully Close, Fully Open
CV Near SF to BP	4	-10%, +10%, Fully Close, Fully Open

All of the actions, with exception of 'No Action', are related to the manipulation of HTF flow. This highlights the importance of having an accurate hydraulic model and validates the considerable effort and detail invested in its development. Also, it is important to note that these discrete actions can be adjusted. For example, if it proved useful, additional actions allowing valves to be adjusted by 5% could be included.

Lastly, and perhaps most importantly, is the development of the reward function. Properly defining the reward function is essential as it guides the algorithm in learning behaviors that yield positive outcomes for the plant. Although this area requires further research, the following description outlines the reward function used in this case study. Recall that Q-learning achieves its optimal policy by identifying the maximum Q-values. Consequently, the algorithm administers punishments through negative reward values and encourages good behavior with positive rewards.

For example, before the turbines are operational, the solar field should not defocus. Doing so implies that the plant is wasting energy that is required to get the turbines online. This is undesirable behavior. Similarly, sending 'cold' HTF through the power block is discouraged as it cools down the heat exchangers which can lead to higher thermal stresses and extended startup times. Additional penalties are imposed for actions that could damage the plant or are logically unsound. For instance, operating pumps at a speed that yields pump cavitation results in a penalty. Also, attempts to further open or

close a valve that is already fully opened or closed are also penalized.

On the other hand, desirable behavior is heavily rewarded. Sending 'hot' HTF, specifically HTF above a certain temperature threshold, to the power block helps warm up the heat exchangers and initiate the heating of water in the steam generation train. Furthermore, HTF above 700 [F] is particularly beneficial as it facilitates the generation of superheated steam which enables the turbines to start generating electricity.

The reward function incorporates many conditional rewards and penalties. With more time, a more robust function could likely be developed.

5.2.1 Model Parameters and Training

In this case study, the agent interacts with the environment and records an observation every 2 minutes. With the historical sequence capturing 5 observations, the model effectively retains a perfect recollection of the past 10 minutes. The decision to set the interaction frequency at 2 minutes was made to balance the length of the sequence input into STANN while ensuring that the agent could make informed decisions based on a substantial historical context. Currently, the sequence input contains 55 values. Reducing the interaction frequency to 1 minute would double the sequence length to 110 items (assuming the agent still needs to access data from the past 10 minutes). Such an increase would significantly expand the number of parameters that STANN needs to learn which could potentially complicate the model without inducing proportionate gains in performance.

The training process utilizes an epsilon greedy approach to explore the state/action space. For each simulation, a new epsilon value, bounded between 0 and 1, is prescribed. Then, at each decision point, a random number from a uniform distribution between 0 and 1 is generated. If the random number is greater than epsilon, the agent will select a

random action. However, if the random number is less than epsilon, the agent will select the action that corresponds to the maximum Q-value. Here, an epsilon value of 0 results in purely exploratory behavior where the agent always chooses a random action. On the other hand, an epsilon value of 1 leads the agent to use its learned policy by making decisions based on the highest Q-values. As epsilon increases, the agent becomes less exploratory and more dependent on its knowledge of the environment.

In this work, the model undergoes sequential training involving multiple simulations. Specifically, six simulations are conducted with the epsilon values linearly increasing from 0.6 to 1. The results from these simulations are stored in memory for subsequent training. Upon completion of the six simulations, the model is trained on the accumulated data for 200 epochs. To improve convergence, the training utilizes both a target STANN model and a policy STANN model where the target model is updated every 10 epochs. Following this phase, the model undergoes another set of six simulations using its updated policy. The cycle is repeated 49 times resulting in 294 simulations in total. The training process is outlined in the Python code below.

Listing 5.2: pseudo-code to represent the training process

```

1 # Define epsilon values
2 eps_array = np.linspace(0.6, 1.0, 6)
3 # Loop through cycle
4 for cycle in range(49):
5     # Loop through simulations
6     for sim in range(6):
7         # Define new epsilon value
8         eps = eps_array[sim]
9         # Loop through simulation timesteps
10        for step in simulation:
11            # Determine action with epsilon greedy approach
12            r_number = random(1)
13            if r_number < eps:
14                action = argmax(STANN_policy(state))
15            else:

```

```

16         action = random(19, dtype = int)
17         # Step through simulation
18         next_state, reward = environment_step(state, action)
19         # save to memory
20         save_to_memory(next_state, reward, state, action)
21         # update state and repeat
22         state = next_state
23     # Train STANN
24     for epoch in range(200):
25         # Loop through all data in memory
26         for state, reward, next_state in memory:
27             # Compute loss and gradient
28             loss += STANN_policy(state)[action] - (reward + gamma*max(
                STANN_target(next_state))
29             # Update policy model with gradient
30             STANN_policy.update()
31         # Update target model
32         if epoch % 10 == 0:
33             STANN_target = STANN_policy

```

5.2.2 Results

Unlike typical training applications where epsilon values start off low and increase to encourage less exploration over time, this work adopts an alternative approach where, during each training cycle, one simulation is intentionally performed with epsilon set to one. This allows the model to act entirely based on its current optimal policy which provides interesting insights into its decision making process given a certain amount of training data. Additionally, this approach enables the ability to visualize how the model's behavior evolves as it gains more experience from an increasing number of simulations. Figure 5.15 displays the cumulative rewards earned by the model during each on-policy simulation. For comparison, the highest cumulative reward that I was able to achieve manually in the same startup scenario is also presented which serves as a benchmark for evaluating the model's performance.

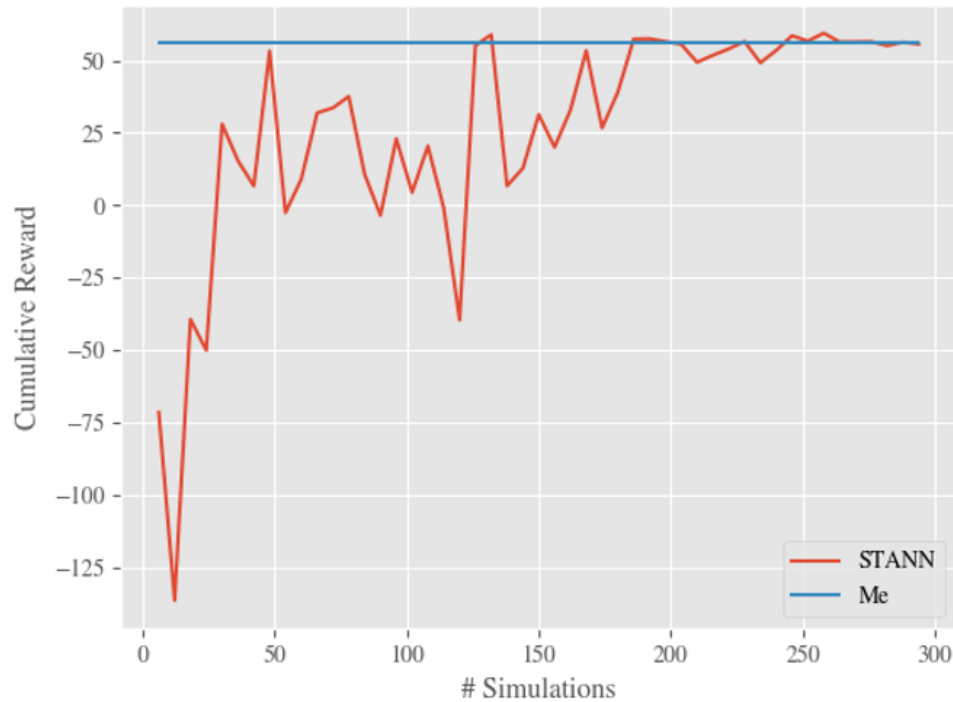


Figure 5.15: Cumulative rewards obtained with optimal policy as a function of number of simulations included in the training memory

This figure clearly illustrates STANN’s evolving understanding of the system as it accumulates experience. Initially, for the first 125 simulations, STANN struggles to consistently achieve positive rewards. However, after 125 simulations, the model appears to stabilize. It seems to focus in on a policy, and it continually refines it to achieve performance levels comparable to, or exceeding, my own.

A crucial aspect to address is the management of memory during the training process. Although typical deep-Q learning approaches cap memory size to include only the most recent simulations, which allows the model to learn from an increasingly effective policy, this approach proves to be inadequate for this environment. Due to the prevalence of many ‘bad’ actions and the scarcity of ‘good’ actions in this system, limiting memory can mislead the model into overestimating the number of actions that could yield positive results. By retaining a full history that includes many of the early ‘bad’ actions, the model avoids repeating past mistakes and can improve its policy more efficiently. Ultimately,

training STANN on its complete experience history has proven crucial.

Initially, the vision for STANN was to create a tool that could assist operators when they encounter challenges during their training with the simulator. To assess its effectiveness in this role, an experiment was conducted where I chose actions that subjected the plant to undesirable states. Subsequently, STANN was tasked with recommending actions that it believed would remedy the problem. The results of this experiment are presented in 5.16 which presents the reward value as a function of time throughout the simulation. Here, negative reward values indicate undesirable plant states which provide a visual representation of STANN's performance in navigating the plant back to optimal operation.

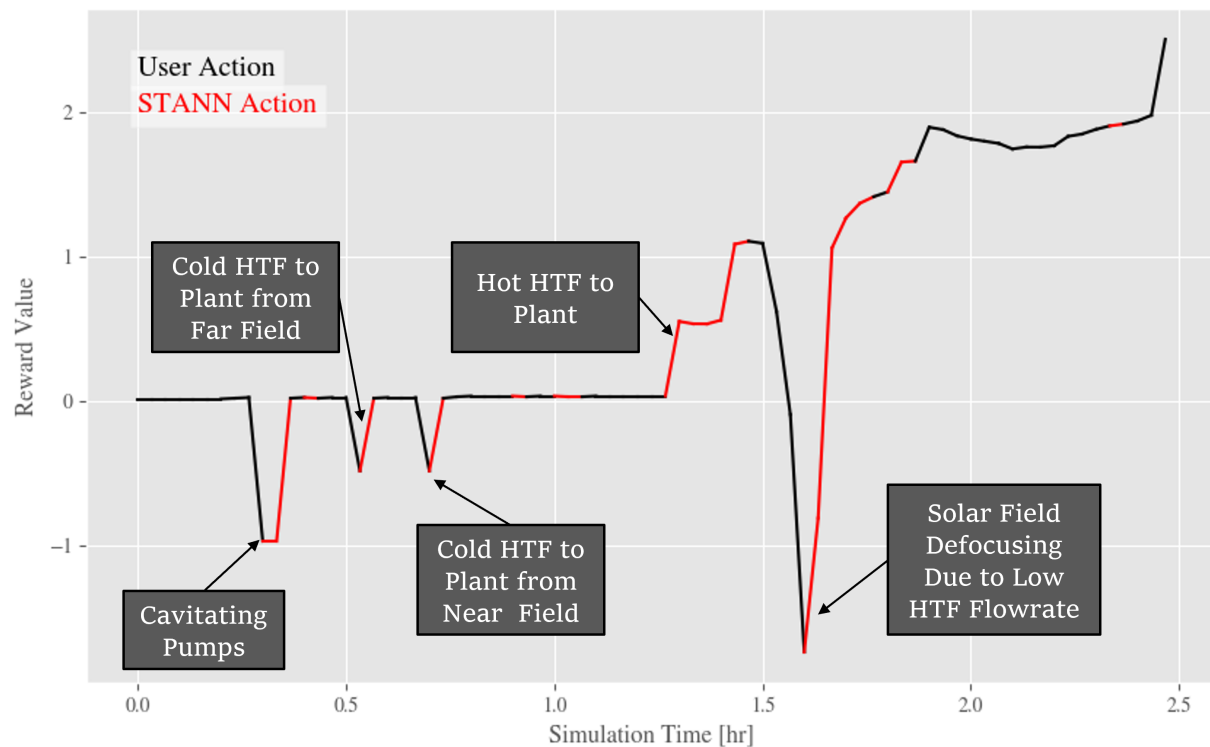


Figure 5.16: Visualization of STANN providing helpful recommendations that are used to get out of undesirable plant states

Remarkably, STANN effectively assists the operator in navigating out of each undesirable state. First, when the pumps cavitate due to high speed with inadequate level in the expansion tanks, STANN waits briefly to see if the level increases to resolve the issue.

Upon realizing the level is not rising sufficiently, it reduces the pump speed to stop the cavitation. Furthermore, in two instances where control valves are incorrectly opened thereby directing cold HTF to the central plant, STANN immediately advises to close the valve of concern.

Moving on, approximately 1.25 hours into the simulation, the HTF from the near solar field sectors becomes hot enough such that it can be sent through the plant and generate a positive reward. Upon realizing this, STANN opens the relevant control valve. It then waits for the HTF from the far solar field sectors, which takes long to reach the plant due to its distance, to warm up before directing it through the plant as well. Lastly, around 1.5 hours into the simulation, the user intentionally reduces pump speeds which results in defocusing in the solar field. STANN quickly responds to this by increasing the pump speeds which brings the field back to nominal optical efficiency. Ultimately, this case study demonstrates the remarkable capabilities of the model for helping operators learn during training.

6 CONCLUSIONS

This research aimed to develop an advanced solar field model that can accurately represent the dynamics of a concentrating solar power plant in real time. The work presented a new solar field model that utilizes a neural network to efficiently compute heat absorbed by the HTF for various receiver conditions. The neural network approach was validated through numerous simulations resulting in average error of 0.3%. Furthermore, the solar field model was validated by simulating an entire sector under conditions of varying solar availability. For the sunny day tested, the model achieves a sector outlet temperature mean absolute error of 3.9° [F]. Additionally, on a cloudy day, the model displayed an impressive 70.9% temperature rate agreement indicating its ability to accurately respond to varying solar conditions.

This work's primary contribution is the creation of an integrated hydraulic and thermal model for parabolic trough solar fields. This model is responsive to operator input such as changes in HTF pump speeds or HTF control valve positions. Furthermore, a key feature of the model is its inclusion of often overlooked components like the expansion system which is essential for operational sequences such as plant start-up. This model differentiates itself from previous performance models in that it enables simultaneous simulation of hundreds of uniquely defined solar field loops. It offers a high degree of user flexibility allowing for the customization of various solar field parameters including hydrogen permeation, receiver states, collector optical properties, and variations in loop mass flow rates. However, most importantly, the model is fast and efficient. The complete HTF piping network, containing the solar field, expansion system, main HTF pumps, bypass lines, and control valves, solves 20 times faster than real-time. This makes it a valuable tool for both operator training and design/control optimizations.

In addition to the solar field model, another novel aspect of this research is the devel-

opment of the Solar Transformer Assistant Neural Network (STANN). Although STANN is in the early stages of development, it has already shown potential in understanding key dynamics of the plant. It has successfully learned a policy for navigating a plant start-up in a way that optimizes a reward function. Moreover, STANN has developed the ability to navigate the plant out of problematic states. This tool could be extremely beneficial for operators. If operators ask for it, STANN provides immediate feedback by suggesting actions that it believes will improve the state of the plant. Such real-time guidance could be invaluable for enhancing the efficiency of operator training.

6.1 Future Work

Through the development of these tools, numerous avenues for future work have been identified. While the solar field model is already detailed and effective for the training simulator application, adding further granularity could expand its utility. Refining the model could enable it to serve as a more useful diagnostic tool or to provide more guidance in plant operation and maintenance. Moreover, the research presented on STANN represents the tip of the iceberg. There are extensive opportunities for further exploration in areas such as architectural improvements, reward function formulation, and training protocols. These potential research directions are discussed in greater detail in the subsections that follow

6.1.1 Solar Field Model

Hydrogen permeation in the receivers is a significant issue that aging PTC CSP plants often face. A common method to mitigate the associated heat loss is to pump argon into the annulus space. Argon acts effectively as a heat transfer buffer against the energetic hydrogen atoms thereby substantially reducing heat loss to the environment. Given this solution, it seems advantageous to develop a neural network model that can predict

heat loss in receivers where the annulus space contains mixtures of argon and hydrogen. This could help plants develop improved control strategies or conduct diagnostic studies. However, the existing Forristall model does not accommodate gas mixtures in the annulus space. Future research could focus on integrating a new model for gas mixture heat loss. A great place to start would be the Burkholder thesis [37] which investigated heat loss in receivers with argon-hydrogen mixtures. Once an updated model is established, the methodologies for developing and training neural networks presented in this work could be applied to this new model.

Next, the hydraulic model developed in this work is designed to calculate mass flow rates that operators can adjust by manipulating valve positions and pump speeds. Since operators do not control individual solar field loop valves, the model does not account for flow rate through each loop. This design choice significantly reduces the runtime of the model by eliminating the need to solve for an additional 808 unknown flow rates. However, feedback from industry partners has highlighted an interest in developing a more detailed hydraulic sector model that simulates all flow rates within the loop. Such a model would include details about piping components and loop control valves. Its primary purposes would be to assist in determining valve positions that ensure balanced flow across the sector. Concerns have been raised that the valve positions do not maintain balanced flow throughout the year as variation in solar availability affect the total flow rates which leads to uneven distribution through the loops. Developing an advanced hydraulic/thermal model of a sector would provide industry partners with a valuable tool to address these challenges, and all of the methodology required to build it resides within this thesis.

Furthermore, one area within the solar field that lacks detailed modeling is the heat loss from header and runner pipes. Currently, the model assumes these pipes are adiabatic, but it also allows users to specify a general heat loss per unit length of piping. There

is significant potential for development in this area. Industry partners have expressed an interest in developing detailed heat loss models particularly for uninsulated sections of these pipes. Significant heat loss may occur through these uninsulated sections, such as supports, ball joints, and other connections, which may impact overall plant efficiency. Additionally, heat loss could also be a result of insulation degradation over time. Developing sophisticated models to accurately quantify heat loss in these often overlooked sections of the plant could provide insights for improving operational efficiency of existing plants. With such detailed models, a non-linear fit could be applied and integrated within the solar field model to compute heat loss more accurately.

Lastly, future research could also focus on developing the model's capability to incorporate spatially resolved solar irradiation data. Currently, each control volume in the solar field can accept an individual value for the solar irradiation incident upon the receiver. Implementing this feature would require developing a prereprocessing system for generating or organizing the spatially resolved data. Furthermore, someone would have to go into the model and ensure that the correct irradiance value is applied to each control volume. As weather forecasting technology improves and cloud imaging becomes more integrated with operational plants, there appears to be an opportunity to develop control strategies to optimize plant output on cloudy days. By enabling the model to accept spatially resolved DNI, the model could simulate the realistic effects caused by cloud coverage. With this capability, it could be used as a tool to develop and evaluate various control strategies.

6.1.2 STANN

As STANN is still in its preliminary stages of development, there are numerous potential research avenues to explore. However, I have identified a few paths that I believe are particularly important and intriguing.

First, the convolutional neural network (CNN) enabled the model to converge in this work. However, moving forward, I suggest revising the architecture by removing the CNN component. While the transformer architecture effectively captures the complexity within the sequence, the subsequent CNN may inadvertently discard valuable information. This represents a significant flaw that needs addressing. Given the extensive body of research on modeling with transformers, it is highly likely that an effective solution already exists.

Second, STANN's training has been limited to plant start up on a specific day. An essential next step is to evaluate whether its learned policy is applicable to varying days in the year and varying weather conditions. If the model lacks generalizability, research should focus on determining how many different weather scenarios STANN needs to be exposed to in order to achieve robust performance.

In conjunction with this, there should be an exploration into varying the architecture of STANN by adjusting the number of transformer encoder layers. Currently, the model employs just one transformer encoder, and this yielded positive results. However, considering that large language models like GPT-3 utilize up to 96 transformer encoders, increasing the number of layers could potentially enhance the model's ability to understand plant dynamics.

Once the model achieves generalizability, the outputs of the self-attention mechanism should be examined. The self-attention mechanism highlights important elements in the input sequence by assigning high attention values to relevant and correlated data points. This feature could provide valuable insights into what the model deems significant. Therefore, in a training scenario, STANN could not only recommend actions to the operator but also clarify which measurements it is focusing on to inform these decisions. By linking recommended actions with the reasoning behind them, this approach could significantly enhance operator training.

REFERENCES

- [1] California ISO, <https://www.caiso.com/todaysoutlook/Pages/index.html>. 2024.
- [2] <https://www.michael-smith-engineers.co.uk/resources/useful-info/centrifugal-pumps>. 2024.
- [3] Mehos, Mark, Hank Price, Robert Cable, David Kearney, Bruce Kelly, Gregory Kolb, and Frederick Morse. 2020. Concentrating solar power best practices study. Tech. Rep., National Renewable Energy Laboratory.
- [4] <https://www.energy.gov/lpo/solana>. 2017.
- [5] Price, Hank. 2003. A parabolic trough solar power plant simulation model. In *Proc. international solar energy conference, hawaii island, hawaii*.
- [6] Blair, Nate, Mark Mehos, and Craig Christensen. 2008. Sensitivity of concentrating solar power trough performance, cost, and financing with the solar advisor model. In *Proc. solarpaces symposium, las vegas, nevada*.
- [7] Wagner, Michael, and Paul Gilman. 2011. Technical manual for the sam physical trough model. Tech. Rep., National Renewable Energy Laboratory.
- [8] Forristall, Russell. 2003. Heat transfer analysis and modeling of a parabolic trough solar receiver implemented in engineering equation solver. Tech. Rep., National Renewable Energy Laboratory.
- [9] Garcia, Isabel Llorente, Jose Luis Alvarez, and Daniel Blanco. 2011. Performance model for parabolic trough solar thermal power plants with thermal storage: Comparison to operating plant data. *Solar Energy*.
- [10] Burkholder, Frank, and Chuck Kutscher. 2009. Heat loss testing of schott's 2008 ptr70 parabolic trough receiver. Tech. Rep., National Renewable Energy Laboratory.

- [11] Carballo, Jose, Javier Bonilla, Manuel Berenguel, and Patricia Palenzuela. 2018. Parabolic trough collector field dynamic model: Validation, energetic and exergetic analyses. *Solar Energy*.
- [12] Dickes, Remi, Adriano Desideri, Vincent Lemort, and Sylvain Quoilin. 2015. Model reduction for simulating the dynamic behavior of parabolic troughs and a thermocline energy storage in a micro-solar power unit. In *Proc. international conference on efficiency cost optimization simulation and environmental impact of energy systems, pau, france*.
- [13] Buch, Q Altes, R. Dickes, A. Desideri, V. Lemort, and S. Quoilin. 2015. Dynamic modeling of thermal systems using a semi-empirical approach and the thermocycle modelica library. In *Proc. international conference on efficiency cost optimization simulation and environmental impact of energy systems, pau, france*.
- [14] Desideri, Adriano, Remi Dickes, Javier Bonilla, Loreto Valenzuela, Sylvain Quoilin, and Vincent Lemort. 2018. Steady-state and dynamic validation of a parabolic trough collector model using the thermocycle modelica library. *Solar Energy*.
- [15] Quoilin, Sylvain, Adriano Desideri, Jorrit Wronski, Ian Bell, and Vicent Lemort. 2014. Thermocycle: A modelica library for the simulation of thermodynamic systems. In *Proc. international modelica conference, lund, sweden*.
- [16] Noureldin, Kareem. 2018. Modelling and control of transients in parabolic trough power plants with single-phase heat transfer fluids. Ph.D. thesis, North Rhine-Westphalia Technical University of Aachen.
- [17] Noureldin, Kareem, T. Hirsch, and R. Pitz-Pall. 2017. Virtual solar field - validation of a detailed transient simulation tool for line focus ste fields with single phase heat transfer fluid. *Solar Energy*.

- [18] Hoegemann, Daniel, Kareem Noureldin, Florian Bamberger, and Tobias Hirsch. 2018. Development of training simulator software for molten salt parabolic trough test platform. In *Proc. solarpaces symposium, casablanca, morocco*.
- [19] Wendelin, Tim, Aron Dobos, and Allan Lewandowski. 2013. Soltrace: A ray-tracing code for complex solar optical systems. Tech. Rep., National Renewable Energy Laboratory.
- [20] Elsheikh, Ammar, Swellam Sharshir, Mohamed Elaziz, AE Kabeel, Wang Guilan, and Zhang Haiou. 2019. Modeling of solar energy systems using artificial neural network: A comprehensive review. *Solar Energy*.
- [21] Farkas, I., and P. Geczy-Vig. 2003. Neural network modelling of flat-plate solar collectors. *Computers and electronics in agriculture*.
- [22] Martinek, Janna, and Michael J. Wagner. 2021. Efficient prediction of concentrating solar power plant productivity using data clustering. *Solar Energy*.
- [23] Tuman, Matthew J., and Michael J. Wagner. 2022. Informed feature selection for data clustering of csp plant production. In *Proc. solarpaces symposium, albuquerque, new mexico*.
- [24] Jullian, Camila Correa, Enrique Lopez Droguett, and Jose Miguel Cardemil. 2020. Operation scheduling in a solar thermal system: A reinforcement learningbased framework. *Applied Energy*.
- [25] Tilanhun, Fitsum Bekele. 2024. Fuzzy-based predictive deep reinforcement learning for robust and constrained optimal control of industrial solar thermal plants. *Applied Soft Computing*.
- [26] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Poloukhin. 2017. Attention is all you need. In *Proc. conference on neural information processing systems, long beach, ca*.

- [27] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Illya Sutskever. 2018. Improved language understanding by generative pre-training. *OpenAI*.
- [28] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformer for language understanding. *Google AI Language*.
- [29] Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeed, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. In *Proc. conference on neural information processing systems*.
- [30] Esslinger, Kevin, Robert Platt, and Christopher Amato. 2022. Deep transformer q-networks for partially observable reinforcement learning. In *Proc. foundation models for decision making workshop at neural information processing systems*.
- [31] Jain, Sarthak, and Byron C. Wallace. 2019. Attention is not explanation. In *North american chapter of the association for computational linguistics*.
- [32] Guo, Wei, Josage Chathura Perera, Daryl Cox, Sachin Nimbalkar, Thomas Wenning, Kiran Thirumaran, and Eli Levine. 2020. Variable-speed pump efficiency calculation for fluid flow systems with and without static head. *International Journal of Energy Management*.
- [33] Jeppson, Roland. 1974. *Steady flow analysis of pipe networks - an instructional manual*. Utah Water Research Laboratory.
- [34] <https://www.webpages.uidaho.edu/gabrielp/ME549-CE546/matrix-inverse.pdf>. 2024.
- [35] Nair, Vinod, and Geoffrey Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. international conference on machine learning, haifa, israel*.

- [36] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectified neural networks. In *Proc. international conference on artificial intelligence and statistics, fort lauderdale, fl.*
- [37] Burkholder, Frank. 2011. Transition regime heat conduction of argon/hydrogen and xenon/hydrogen mixtures in a parabolic trough receiver. Ph.D. thesis, University of Colorado.
- [38] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *Nips deep learning workshop.*