A GUI for online presentation of steel and steelmaking ladle temperature data and simulation.

Muhammad Faheem

Master’s Thesis
Electrical Engineering
Nr: E3775E

September 2009
Abstract

To understand, present and work with any important data, proper presentation techniques are needed. Different techniques can be used, most importantly the GUI. This report basically deals with the GUI that was designed to present the results of a model and work with it. Continuous casting is a casting process that produces steel slabs in a continuous manner with steel being poured at the top of the caster and a steel strand emerging from the mould below. Liquid steel is transferred from the AOD converter to the caster in a ladle. The ladle is designed to be strong and insulated. Complete insulation is never achieved. Some of the heat is lost to the refractories by convection and conduction. Heat losses by radiation also occur. For this reason, a model was previously developed to simulate the steel and ladle wall temperatures during the ladle cycle. The model was developed as an ODE based model using grey box modeling technique. The model’s performance was acceptable and needed to be presented in a user friendly way. The aim of this report is basically to elaborate the design and
working of the GUI, that was designed to present steel and ladle wall temperatures calculated by the model. The GUI not only presents the temperatures and other important data about the process, but also allows its user to make changes to the model during the simulation. The user is able to view the plots in any time scale. The GUI also alerts the user with warning and error messages and keeps a log of all those messages. The GUI was designed using Matlab’s GUIDE tool. The report introduces continuous casting process and the ladle. The report presents a literary review of previous models designed for more or less the same purpose. The effects of temperature are also discussed based on the natural behavior of the process and the results achieved by different researchers. The GUI is then completely explained with all its callbacks, variables and algorithm. The report is also aimed at discussing the sensitivity analysis of the different parameters and their effects on different temperature estimations. Only the most significant results for the sensitivity analysis are presented by the use of plots and theory. Some hints on related future work that might be done with the help of the GUI are also presented at the end.
Acknowledgements

I would like to thank my parents for their unconditional love and support. I also owe much to Pär Samuelsson for his support throughout my thesis work. Many thanks to Tahera Jan at Outokumpu Avesta Works, who helped me with testing and waited patiently to see the results of my work. I would also like to thank all my teachers who tried their best to teach me all those confusing concepts and all my friends who made life good and enjoyable for me.
Contents

1. Introduction. .......................................................................................................................... 1
   1.1. The ladle .......................................................................................................................... 2
   1.2. The ladle cycle ............................................................................................................... 3
   1.3. The temperature factor and its effects ......................................................................... 6

2. Ladle temperature modeling .................................................................................................. 9
   2.1. The need for modeling .................................................................................................. 9
   2.2. Literature review ......................................................................................................... 9
   2.3. The modeling methods ............................................................................................... 11
   2.4. Steps in grey box modeling ..................................................................................... 12
   2.5. The process, its inputs, outputs and assumptions ................................................... 12
   2.6. The measurement campaign ..................................................................................... 14
   2.7. The model ................................................................................................................ 14
   2.8. Simulation example ................................................................................................. 15

3. The GUI Features ............................................................................................................... 19
   3.1. The inputs and outputs ............................................................................................ 20
   3.2. Common features ...................................................................................................... 21
   3.3. The “General” tab ....................................................................................................... 27
   3.4. The “Steel temperature” tab ...................................................................................... 28
   3.5. The “Inner ladle-wall temperature” tab .................................................................... 29
   3.6. The “Intermediate ladle-wall temperature” tab ..................................................... 30
   3.7. The “Outer ladle-wall temperature” tab .................................................................. 31
   3.8. The “All temperatures” tab ...................................................................................... 32
   3.9. The “Messages” tab .................................................................................................. 33
   3.10. Running the simulation .......................................................................................... 34
4. The GUI code. ................................................................. 35
  4.1. Callbacks. ................................................................. 35
  4.2. Ladle GUI callbacks .................................................. 36
  4.3. The variable reference. .............................................. 45
  4.4. Reconstruction from saved data .................................. 51

5. Sensitivity analysis. ....................................................... 53
  5.1. Sensitivity for the current steel temperature ................... 53
  5.2. Sensitivity for the inner ladle wall temperature ................ 55
  5.3. Sensitivity for the ladle wall temperature at an intermediate point ........... 56
  5.4. Sensitivity for the outer ladle wall temperature ............... 58

6. Future work. .................................................................. 61

7. Conclusions. .................................................................. 63

8. References. .................................................................. 65

9. Appendices. .................................................................. 67
  9.1. Appendix A: Short description of files related to the GUI & the model. .... 67
  9.2. Appendix B: Code for the GUI ....................................... 71
  9.3. Appendix C: The reconstruction code ............................... 87

10. Glossary ...................................................................... 89
Page left blank intentionally
1. Introduction.

The process of casting molten steel into semi finished shapes in a continuous fashion is called continuous casting. These unfinished shapes are then ready for desired shapes and structures be made from them. The process is called continuous casting because molten steel is cast into slabs continuously without the need for stationary moulds. Before the introduction of continuous casting technique, molten steel was cast into stationary moulds which involved many steps and needed more time and energy. But with the introduction of continuous casting, time consuming steps involving casting of steel into stationary moulds and other stationary casting steps were left out. More time consumed during casting meant more energy required to keep the molten steel at the desired temperature and this in turn meant more cost. Figure 1.1 below illustrates the continuous casting process in a simplified form.

![Continuous Steel Casting Diagram](image)

Figure 1.1: The continuous steel casting.

Molten steel is poured continuously from a ladle into the tundish. From the tundish, the molten steel flows down to a water cooled copper mould where the skeleton of the slab is formed as the steel slab is cooled to a greater extent. Then that semi cooled slab slowly rolls out of the copper mould downwards while more molten steel continues to come into the copper mould. In this way a continuous steel strand is formed which is then cut into pieces once the steel slab is completely solidified.
1.1. The ladle.

Ladle is a basic tool extensively used in the continuous casting cycle. Ladle is a container used to transport molten steel from the furnace to the casting machine. Ladles are designed to be heat resistant and strong. Moreover it is also necessary that a ladle be heat insulated. Proper heat insulation is required so that the molten steel contained in the ladle remains at a proper temperature. Ladles come in different capacities. At Outokumpu Avesta Works, teeming ladles carry 95 tons of molten steel. The ladle is 3.2 meters in height and has a diameter of 3.5 meters [17].

A general construction of the ladle is shown in Figure 1.2 [17]. The ladle structure is multilayered because of the fact that a ladle should be strong and heat insulated. The inner face of the ladle is built from specialized refractory bricks. These bricks are resistant to high temperature, thus making it possible for the ladle to hold molten steel. Two types of brick are used to construct the inner surface. One type of brick is used to construct part of the surface that will interact with the liquid steel while the other type of brick is used to construct the surface that will interact with the slag layer above the molten steel. Then there is a mass layer. Behind the mass layer is a safety layer. Then comes the insulation layer and all these layers are covered by a steel shell on the outermost side. All these layers make the ladle wall about 0.3 meters in thickness. All these layers are to ensure the ladle will be able to withstand and contain high temperatures. A lid is also usually used to cover the top of the ladle.

![Figure 1.2: The ladle [17].](image)

The transportation of steel poured in the ladle is an important step in the casting process and most of the heat is lost during this phase. The melt might remain in the ladle for a long time, which is why the ladle is properly insulated and strengthened.
1.2. The ladle cycle.

The ladle cycle involves a number of steps more or less the same in all corporations. The process can be divided into six steps. They are, ladle maintenance, ladle preheating, converter, transportation, secondary refining and at last the continuous casting [17]. Figure 1.3 below shows the process. These steps will be explained one by one.

1.2.1. Ladle maintenance:

The ladle has to hold molten steel at temperatures around 1600 °C for long durations. These high temperatures mean that even if the ladle is strong enough to hold them, it is natural for the ladle to wear out after some time. Every ladle needs continuous maintenance to keep it in good shape. So the maintenance station is a place where the ladle arrives after going through the cycle. Here the ladle is cleaned from any residuals of the previous heat. After cleaning, sand is poured in the nozzle of the ladle in order to prevent it from getting blocked once the ladle is filled. The sliding gates or nozzle of the ladle is changed if necessary. The sliding gates are replaced after every three heats.
At Outokumpu Avesta Works, a new ladle is usually used for 35 heats before a major maintenance is conducted. After this, the ladle is used for 35 more heats and then the ladle is taken out of service for a relining of the refractory layer. The ladle endures high temperatures when the ladle is charged. Those high temperatures cause the lining to wear. As the lining gets thinner by the time, the temperature performance of the ladle decreases. So this is why relining is needed. Once this relining is done, the ladle is then ready for a rally of 70 charges again.

1.2.2. Ladle preheating:

After the ladle leaves the maintenance station, it arrives at the preheating station. Here a burner is placed in the ladle which is used to push the ladle temperature to a desirable value. It is not necessary that every ladle is brought to the preheating station. Moreover, the heating time for each ladle varies. This is because of the ladle scheduling. If the ladle is scheduled to have a long empty time, then it must be placed under the flame in order to maintain the heat content of the walls. But if the ladle is scheduled for the next heat soon then the ladle will go to the converter station skipping the preheating.

Research has suggested [12] that long empty periods (exceeding 2.5 hours) gives low metal delivery temperature and needs preheating. This is because of the fact that if the ladle is kept idle for long periods, it will lose much of its heat due to radiation and conduction. On the other hand, if the empty period is short, then preheating is counterproductive [12]. This is because the surface of the refractory layer is still hot from the previous exposure to liquid steel and the temperature gradient between the flame and the refractory layer will be small hence giving low heat flux. Preheating in this situation will only maintain the surface temperature and the heat of the subsurface refractory layers will be lost to the outer layers eventually resulting in lower metal delivery temperature [12].

1.2.3. Converter:

Primary refining is carried out at the converter where the desired content of carbon and alloys is obtained in the steel. Different qualities of steel require different proportions of alloy addition. Once primary refining is done, the liquid steel is poured into the ladle. This pouring of liquid steel is called tapping. The ladle comes from the preheating station or it may come directly from the maintenance station depending on the empty period of the ladle. The temperature of the liquid steel is around 1600 °C. The weight of the liquid steel in a filled ladle is usually up to 90 tons.

1.2.4. Ladle transportation:

Once the ladle is filled, it is placed on the rails for the transportation to begin. The molten steel is covered with a layer of slag in order to reduce temperature losses. After this, a lid is used to cover the top of the ladle for further reduction in temperature losses. Now the ladle is ready to be transported. The ladle might stay on the rails, waiting to be taken to the
ladle furnace for secondary refining, depending on the schedule of the cycle at the facility. It is here at the transportation phase, when most of the temperature loss occurs. And to compensate any losses incurred here, the secondary refining stage is required.

1.2.5. Secondary refining:

At Outokumpu Avesta Works, secondary refining is done at the ladle furnace station. It may or may not be present in other steel making facilities elsewhere but where there is no secondary refining phase, the ladle is taken to the casting machine directly from the converter. Secondary refining is important because the carbon content and alloy proportions might not be at the required level. Along with tuning of the alloy proportion in the molten steel, the temperature also needs to be adjusted to achieve any specific quality of steel.

At the ladle furnace, specified amounts of alloys are added to the steel and excessive carbon is removed. Temperature measurements are taken to ensure the steel is at the correct temperature for casting. Apart from any specific steel grade requirements, it is important for the molten steel to be at the correct temperature for other reasons too. If the temperature is too high, a breakout can occur and if it is too low, nozzle clogging might be the result. Heat bursts are given to the steel if the temperature is low and scrap metal is added if it is high. Stirring is performed using gas or electromagnetic purging to reduce stratification hence making the liquid homogeneous.

1.2.6. Continuous casting:

In this phase, the ladle is taken to the upper edge of the casting machine where steel from the ladle is gradually tapped into a tundish via the nozzle. The tundish is used to provide the casting machine with a continuous flow of molten steel during the time when an empty ladle is being replaced with a filled one.

From the tundish, the molten steel flows down to a mould which is water cooled. Liquid steel flows into the mould and a continuous strand of steel slides out of the mould. Water cooling is used so that the steel strand’s outer shell is formed. When the strand slides out of the mould, it has a solid shell but on the inside, the strand is not completely solid and it needs to be cooled further for complete solidification. When the strand is solidified, it is cut into slabs.

From the casting machine, the empty ladle goes back to the maintenance station where it undergoes necessary maintenance in order to prepare it for the next heat.
1.3. The temperature factor and its effects.

The temperature of the molten steel at casting plays an important role in the quality of the steel. To keep the casting temperature at the desired level, the temperature needs to be monitored and controlled throughout the process. The first step in designing a control system for a process is to understand the behavior of the system. In case of secondary steel casting, it is necessary to understand the general behavior of the ladle and the effects of temperature on the ladle process.

During the ladle cycle, some heat is lost to the refractory layers by conduction while some of the heat is lost to the atmosphere by radiation. The radiated heat is reduced by the usage of slag layer and the lid. The addition of ferroalloys also adds to the heat loss. At the ladle furnace, all the lost heat is compensated as the molten steel is reheated to the required level by arcing. The heat loss is directly related to the thermal status of the ladle. If the ladle has high heat content at the instant of tapping, less heat loss from the molten steel will occur but if the ladle has lower heat content, then heat losses from the molten steel will be higher. The amount of heat content a ladle can carry depends upon the construction of the ladle and its thermal properties.

At the preheating station, the inner ladle refractories absorb heat from the gas flame. Some of the heat from the flame is lost to the ambience. Heat absorbed by the inner ladle refractories is gradually passed to the outer refractories by conduction. Heat is also lost to the ambience by radiation from the ladle shell. If the ladle remains empty for longer periods, then preheating will be necessary because the ladle will lose its heat content to the environment. But if the ladle’s empty time is short then the ladle can be utilized without preheating. This is because the ladle has inherited a suitable amount of heat content from the previous cycle that is enough to provide a suitable metal delivery temperature for the next heat. Ladles with short empty times have higher heat content in their lining and relatively low surface temperature while ladles that undergo a long preheating have lower heat content in their lining and a higher surface temperature [5].

A previous study [12] has found that preheating after a short empty period (30 minutes or less) is counterproductive because the temperature difference between the flame and the still hot refractories is small. So preheating in this case will only maintain the surface temperature and during that time, the heat content of the inner refractories will reduce. Its heat will be transferred to the outer layers by conduction and eventually radiated from the outer shell. Similarly, the study also indicates that if the ladle is empty for one hour, then a preheating of only 30 minutes will give the highest metal delivery temperature.

Heat loss also occurs by radiation from the inner and outer face of the ladle when the ladle is being brought for tapping. During tapping, the molten steel loses some of its heat to the ambience by radiation and some of its heat is lost by convection to the refractories of the converter’s mouth [3]. If an inadequately heated ladle is brought for tapping, the molten steel poured into it will lose heat at a higher rate. Similarly, a green skin ladle chills the molten steel more than a well-cycled ladle [12]. This is because the heat content of the refractories of a green skin ladle, even after heating, is relatively lower than that of a well-cycled ladle.
Once the ladle is tapped, heat loss from the molten steel then depends upon the thermal status and the thermal properties of the ladle. Ideally, the ladle must have a high heat content and the thermal conductivity of the ladle refractories must be as low as possible. But in practice, the refractories that make the different layers of the ladle have a high heat conductivity. Some heat insulation layers are incorporated into the ladle for this reason. This makes the ladle somewhat insulated but still, the heat continues to leak out of the ladle during the rest of the cycle.

The inner refractories of the ladle start receiving heat from the molten steel by conduction. This heat is further conducted to the outer refractories and eventually, gets radiated to the ambience. The initial heat loss is at a higher rate which then gradually reaches a steady rate. As discussed earlier, the heat lost to the refractories depend upon the status of the ladle. Compared to a green skin ladle that has even been heated at the preheating station, lesser heat is lost if the ladle is well-cycled.

Heat is also lost by radiation directly from the surface of the melt to the ambience. A slag layer is used to reduce the radiated heat loss from the surface of the melt. The reduction in heat loss due to a slag layer depends upon the thickness of the slag layer, its type and its distribution [4]. A thick slag layer provides better insulation. To reduce the radiation losses from the hot face of the ladle and the melt surface further, a lid is used to cover the top of the ladle. A study [12] has indicated that with a thick slag layer, 15 °C more heat was lost without the usage of the lid. The study also indicated that with a thin slag layer, 40 °C more heat was lost without the use of lid. This indicates that both the thickness of the slag layer and the usage of lid contribute to the heat held by the melt.

During the phase when the ladle is being transferred towards the casting machine, the ladle has to wait depending upon the schedule at the facility. The molten steel continues to lose heat to the refractories and to the ambience. Stratification occurs during this phase and increases with holding time. A study [3] shows that a thick slag layer results in lesser heat loss but a higher degree of stratification while a thin slag layer results in more heat loss but less stratification occurs and the melt remains homogeneous. The thin layer causes buoyancy driven convection currents which keeps the melt well mixed.

When the ladle reaches the secondary refining stage, ferroalloys are added to the melt which results in more heat loss. Heat loss due to the addition of ferroalloys depends upon the quantity of the added material and its chill factor. Arc heating is then used to compensate for all the heat losses and bring the melt’s temperature into a desired range. Upon reaching the casting machine, the casting starts and the melt is poured from the ladle to a tundish. More heat is lost during casting by radiation and conduction from both the ladle and the tundish.

A study [3] has estimated that typically 55 to 60 % of the total heat is lost through the ladle wall refractories. 15 to 20 % is lost through the ladle bottom and 25 to 30 % is lost through the slag layer.
Page left blank intentionally
2. Ladle temperature modeling.

Achieving any desired quality of steel depends upon many factors. Controlling the carbon content and the alloy proportions in molten steel is important, but equally important is the control of molten steel temperature. Unlike the alloys and carbon content control, temperature control is difficult. This is because the desired temperature must not only be achieved; it must be achieved at a specific time. The control of such processes is called time-temperature control problem [2].

Models can be offline or online. Initial conditions and assumptions are decided before any model is developed. These decisions play an important role and affect the performance of the model. A lot of online and offline models for the steel temperature prediction exist and some of them will be discussed.

2.1. The need for modeling.

Proper modeling of the temperature can be used to control the continuous casting process. Modeling is done for the whole ladle cycle. Modeling the temperature during the empty state and the preheating state enables us to determine the condition of the ladle. It helps us decide the moment when the ladle is ready for tapping. This will not only result in saving gas at the preheating station, but the time too. Both these resources will add up in the end to reduce the cost of the production. The modeling during the stages when the ladle is filled will help producing better steel quality hence resulting in higher commercial benefits for the organization. If the temperature modeling is accurate, the organization can enjoy huge profits and popularity in the target industry.

With the help of a temperature model, the operator is able to keep the temperature in the desired range. If the temperature is too low, the molten steel homogenization will not be achieved properly and in some cases the nozzle can also get clogged. On the other hand if the molten steel temperature is too high, centerline segregation or a breakout can occur. This will result in huge losses for the company. So the temperature needs to be at a suitable level. In this way, the models, apart from bringing profits to the organization, are also instrumental in saving the organization and the workers at the plant from dangerous accidents.

2.2. Literature review.

Temperature modeling in continuous casting process has been done by a lot of researchers in many different ways. The amount of research done on this time-temperature control problem is a proof of its importance to the steel industry. Here, a brief review from some of the papers will be presented in order to get a bird’s eye view of the work done earlier to model the molten steel temperature.

A model was developed for temperature prediction [3]. A combination of one dimensional heat transfer model and a statistical model was used. The heat transfer model is used to model heat losses up to the ladle furnace stage while the statistical model is used to
model the stages after the secondary refining at the ladle furnace. The model was designed to be used as an online model.

The same overall technique was also found in another model [4], where in the first stage, a mathematical model was developed based on the two dimensional fluid flow and thermal analysis was developed. And in the second stage an online model was developed using the data generated by the mathematical model. The second stage was based on the simplified physics and statistical analysis [4]. This online two stage model gave satisfactory results when analyzed and validated for over 100 heats. The predicted temperatures were within ±5 °K for more than 90% of the heats [4].

Another model was developed by Zoryk and Reid [6]. The model developed is based on finite difference method. The online system developed is based upon two mathematical models. They are called Ladle Thermal Tracking and Flight-path [6]. The Ladle Thermal Tracking model continuously calculates the refractory lining temperatures of all the ladles in operation while the Flight-path model calculates the liquid steel temperature in the ladle and the tundish.

Zabadal, Vilhena and Bogado Leite [7] developed another online model. Here, the two dimensional heat transfer problem is solved numerically with finite differences. The model developed is aided with two auxiliary algorithms. The first auxiliary algorithm is used to bring the accuracy of the model into narrow limits while the second one is used for time extrapolation of the temperature behavior. Both these auxiliary algorithms help improve the efficiency of the model. The model resulted in quality improvement with 80% of the steel ladle batches coming out within the end user specifications [7].

An intelligent ladle furnace control system was designed [8]. It was developed for temperature prediction in the secondary refining stage. This temperature prediction was in turn used for the electrode control in that stage. A combination of artificial neural network and an expert system was used to predict the temperatures. The artificial neural network was used to calculate the normal temperature and the expert system was used to calculate the delta temperature according to the reference temperature. It gave good results resulting in 14% reduction of electrical energy consumption per ton.

Jormalainen and Louhenkilpi [9] also developed a mathematical model for the temperature in the ladle and the tundish. Separate but dependent models were created for each stage in the ladle cycle finally giving the estimated temperatures for the outlet temperature of the melt drained from the ladle and the steel in the ladle. The model can be used both offline and online. The different models were designed using different techniques. The overall correlation between the test and the model data was nearly 0.9, showing the effectiveness of the model.

A two dimensional mathematical model for steelmaking ladles was presented [10]. The model is an offline model. Initially, the heat flow through the refractory was calculated through partial differential equations. This was then assisted with other routines for the calculation of radiation losses, consideration of tapping situations and calculation of steel heat content [10]. It was found that the steel temperature was influenced by the thermal state of the lining of the ladle. The results achieved by the model were found to be good when compared with measurement values.
Another two dimensional model was developed [12]. The system of equations was solved using the ADI (Alternating Direction Implicit) method and adaptive time stepping. The model is calibrated in a stage-wise manner [12]. Each stage is taken and the parameters relevant to that stage are adjusted iteratively until the predicted temperatures matches the measured temperatures. It was found that the use of slag layer and the lid reduced the radiation heat loses. The results of this model were found to be good.

2.3. The modeling methods.

Models are needed for complex processes to be understood and executed properly. Modeling is extensively used in the industry to improve the system and increase its efficiency. With modeling, a process produces good quality products in a cost effective manner. Models can be developed using any of the three techniques depending upon the availability of data. The three techniques are white, black and grey box modeling techniques. A detailed theory about different modeling techniques can be found in [1].

In white box modeling, the model is developed completely from mathematical relations, for example, from difference, differential or algebraic equations. The system is described by inputs, outputs and some state variables. The output is expressed as a relation between the inputs and the states hence completely covering the behavior of the system. To get a good model using white box modeling technique, different parameters involved in the process must be known with accuracy. No measurements from the data are required to develop a white box model. The white box model is completely modeled from mathematical equations.

Black box modeling technique is used when it is difficult to know the parameters involved in the process. As its name implies, the model is unknown and is a black box, but in this case, the inputs and outputs are known. To model a system using black box modeling technique, sufficient amount of measurements are taken from the process. A model from a family of available models is selected and the unknown parameters of that model are then estimated using the measured data. After the parameters are estimated, the model then represents the process. A linear black box model can be an ARX, ARMAX, OE or any other type of black box models available. A nonlinear black box model can be made using fuzzy logic or artificial neural networks. Black box models are in fact identified using the measurement data.

As its name implies, grey box modeling has the characteristics of both white and black box modeling techniques. Grey box modeling is used when knowledge about the process is incomplete. For example in the case of nonlinear processes where there are many parameters involved and knowing all of them is virtually impossible. In grey box modeling, the incomplete knowledge about a process is used to set up a basic structure of the model and then the unknown parameters of the process are estimated using measurement data. In this way the model is calibrated. Because of the fact that grey box modeling has the characteristics of both the white and the black box modeling techniques; it usually results in a better model. In short, grey box modeling involves both modeling and identification.
2.4. Steps in grey box modeling.

Developing a model using the grey box methodology is a step by step process. The process of developing a grey box model is an interactive and iterative one. Depending upon the results achieved after going through the steps, different steps might be repeated to achieve a desired model. The different steps involved in the process are basic modeling, experimentation, estimation, model analysis, model appraisal and model expansion.

In basic modeling step, a basic structure of the model is created based upon incomplete knowledge about the process. This basic model is based on physical relations. The knowledge about the process is based on some assumptions and some facts. Some of the assumptions may come out to be wrong during a later stage which would then require an according change in the basic model.

In the next step, experimentation is done and data is collected from the process. The inputs and outputs of the process are decided and data for those inputs and outputs are collected. This data is then used to estimate or calibrate the basic model. In the calibration procedure, the measurement data is tested with several different available versions of the basic model.

The models are then analyzed by performing simulations and carrying out statistical analysis of the model parameters that were estimated by calibration. An appraisal is performed and a basic model is selected as a tentative model from several possible basic models. The model that is selected works better with the measurement data than all other alternative models. That selected basic model is then, in later stages and iterations, expanded. Different dimensions for the expansion of the basic model are identified.

After this, the tentative model is expanded by estimating more parameters and by changing the influence of different parameters. Such changes in the tentative model that is selected when calibration was done, makes room for the model to be expanded in different directions and so more versions of the tentative model are available for further calibration.

The newly created versions of the tentative model are calibrated and analyzed. An appraisal is performed and the model that represents the process more close than the others is selected as the new tentative model. In this way, the basic model expands to encompass more unknown parameters. The steps in the grey box modeling process are iterated until a model that satisfactorily represents the behavior of the process is achieved.

2.5. The process, its inputs, outputs and assumptions.

Outokumpu Avesta Works is a leading Swedish stainless steel manufacturer producing up to 600,000 tons of stainless steel every year. As discussed earlier, the ladles used in the steelmaking process at the facility have a capacity of 95 tons. The ladles have a diameter of 3.5 meters and a height of 3.2 meters. Up to four ladles operate simultaneously and 20 charges are cast at the facility everyday.
During the ladle process, once the molten steel is poured into the ladle, the temperature of the melt starts to drop gradually. The temperature of the molten steel at casting is required to be at a certain level depending upon the desired grade of steel. So in order to predict the temperature of the steel in the ladle throughout the cycle, a model was required. A model that can predict the temperature with a good accuracy hence providing the control engineer useful data based on which the engineer can take decisions. Such a model for the steelmaking ladle was developed by Samuelsson and Sohlberg [17]. The model was developed using grey box modeling technique. The model will be adequately explained throughout the rest of this chapter.

The temperature at tapping, the steel weight and some other related parameters were provided as inputs for the process model while the outputs of the process model were decided to be the molten steel temperature in the ladle and the temperatures at the measured nodes in the interior of the ladle wall.

In this thesis, a grey box model developed in [17] is used as a basic model in the GUI. Some assumptions were made in order to define the model boundaries. The assumptions are [17]:

- The steel bath is completely mixed.
- Heat loss from the liquid steel occurs through conduction/convection via the refractory layer surface.
- When the ladle is not filled, heat loss occurs from the refractory surface by convection.
- Heat loss from the ladle steel shell is modeled as a radiation process.
- Heat is transported in a radial direction in the ladle walls and in axial direction in the ladle bottom.
- The ladle is assumed to be filled with steel from converter tapping to the end of casting.

Further assumptions were made to model the input signals. The assumptions are [17]:

- The temperature impact on the liquid steel of the alloy and cooling scrap additions are assumed to be completely known. The method to calculate the chill factors for different alloy additions can be found in [20].
- The heating effect from the ladle furnace on the liquid steel was assumed to be constant. A signal was used to indicate the start and stop of the heating at the ladle furnace.
- At the preheating stage, the heating impact of the burner on empty ladle’s refractory surface was assumed to be constant. A signal was used to indicate the start and stop of the preheating.
2.6. The measurement campaign.

Many unknown parameters were involved in the process. So the grey box modeling technique was chosen. And as in grey box modeling, apart from the partial knowledge about the process, measurements were also required. In order to model the process [17], temperature measurements were taken and those temperature measurements were then used in the calibration of the model.

For temperature measurements of the ladle, thermocouples were inserted into the ladle wall at an intermediate height (1500 mm from the ladle top). Thermocouples of type K were used at that height to measure the temperature of the outer boundary of the refractory layer and the outer boundary of the mass layer. The temperature measurements of the ladle’s steel shell were taken using an IR measurement device. Initial temperature of the steel was measured at the converter before tapping. Later in the process, steel temperature measurements were taken at the ladle furnace station. The ladle wall temperatures were measured at the ladle maintenance station and at the ladle furnace station.

Two measurement campaigns were undertaken at the facility. The first campaign was carried out in May 2007 and the second one was in May 2008. The data from May 2008 campaign was used for calibration while that of May 2007 was used for validation. In depth information about the measurement campaign can be found in [17].

2.7. The model.

As discussed earlier, some assumptions were made. Based on those assumptions, an ordinary differential equation (ODE) can be approximated. That ODE based model was calibrated for steel and ladle wall temperatures of the ladle using grey box approach [17]. A detailed synthesis of the temperature model can be seen in the work done by Samuelsson and Sohlberg [17]. The complete temperature model is made up of two different models. One of the models is for the case when the ladle is filled and the other one is for the case when the ladle is empty. The difference between the two is the fact that no steel temperature equation is needed when the ladle is empty. First of all, a basic model was created. The temperature in the ladle wall can be modeled as a distributed parameter system as [17],

$$\frac{\partial T}{\partial t} = \frac{\lambda}{\rho C_p} \left( \frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right)$$

where $T$ is the temperature in the node [K], $\rho$ is the material density [kg/m$^3$], $C_p$ is the heat capacity [J/kg/K], $\lambda$ is the heat conductivity [W/m/K] and $r$ is the radial position [17]. The partial differential equation (Equation 2.1) can be approximated as an ODE using a standard procedure elaborated by Sohlberg [19].

The model was then calibrated from the data that was achieved in the May 2008 campaign. The data of May 2007 campaign was used for validation of the model. A set of
parameters for the model was eventually achieved that made the model represent the process to a good degree of accuracy, making it suitable for online prediction. A non-commercial software MoCoVa was used for the grey box model calibration.

2.8. Simulation example.

As an example, Figure 2.1 shows the results achieved by the steel temperature model that is implemented in the GUI. In this trial, the ladle was followed for two charges over the duration of 10 hours. Both the measured and simulated temperatures are shown. The estimated temperature of the melt is shown using a blue line and the measured temperatures of the melt at any instant are shown using green dots. The results from this example show that steel temperature model has a good accuracy for the first heat. But for the second heat, the accuracy is not good. However this simulation example is included only to illustrate the model outputs. For a more thorough simulation study, see [17]. Errors in this simulation example may be due to, for example, initialization or they might just be measurement errors.

![Figure 2.1: Steel temperature achieved by the simulation example.](image)

The wall temperatures estimated by the model in this simulation example are also shown. There are no measured wall temperatures for this example. For a more thorough simulation study, that shows the measured temperatures for the walls too, see [17]. Figure 2.2 shows the temperature at the outer surface of the refractory brick layer. Figure 2.3 shows the temperature at the outer boundary of the mass layer inside the wall while Figure 2.4 shows the temperature of the outer shell of the ladle.
Figure 2.2: Inner ladle-wall temperature achieved by the simulation example.

Figure 2.3: Intermediate point ladle-wall temperature achieved by the simulation example.
Figure 2.4: Outer ladle-wall temperature achieved by the simulation example.
Page left blank intentionally
3. The GUI Features.

Accurate calculations are necessary for any scientific project to be successful. But only accurate information is never enough. That accurate information needs to be presented in a form that is easy to understand and manipulate. There are many ways to present the data. The data can be put in the form of tables, plots, pie charts etc., but to be able to present the data as well as manipulate it, a good and an effective GUI is the solution. A GUI must be easy to use and easy to understand.

A GUI for the ladle was designed so that the model for the steel and ladle temperatures can be presented and controlled. The GUI is also capable of taking inputs from the user during the simulation. It presents online data about the ladle during all stages in the ladle process. All the initial inputs are acquired from the online monitoring system installed at the facility. These inputs from the online monitoring system are used by the ladle and steel temperature model to estimate the temperatures of the steel and the ladle wall. The results from the model are read by the GUI and presented accordingly. The user can make corrections to the ladle model during the simulation by adjusting the steel temperatures to a measured temperature. A view of the main interface of the GUI can be seen in Figure 3.1 below.

![Figure 3.1: The Ladle GUI; main interface.](image)

The GUI’s main interface shows all related information about the ladle. The values of these fields are displayed once the simulation starts. Different tabs are available to show different outputs and data in the form of plots and messages. The user can switch between different tabs at ease during the simulation by clicking the desired tab name available
in the tab field on the right. Moreover an easy and safe control of the simulation and plot display is provided in the bottom and bottom-right corner of the GUI window.

### 3.1. The inputs and outputs.

There are some inputs and outputs of the GUI. The inputs to the GUI are obtained from the online monitoring system and the user while the outputs are presented in the form of plots, message log and numerical data display. The data from the online monitoring system at the facility is available in a markup language file. The structured data is extracted from that file using some specialized files created in Matlab. That data is then fed to the model which in turn, is embedded in the GUI.

The ladle status is extracted. The ladle status shows us whether the ladle is empty, filled or is at the preheating station. This data is converted from numbers to words and presented as an output of the GUI. The GUI presents them as “Empty”, “Filled” and “Preheating”. The heat number and the steel weight are extracted and displayed as an output.

All the temperatures, the current steel, the inner ladle wall, the intermediate ladle wall, the outer ladle wall and the tapping temperature are extracted. They are converted from Kelvin to Celsius and presented as both numerical data and as plots.

Heating time in the ladle furnace is extracted and presented. The alloy addition data is extracted and used to calculate and present the cooling effects of alloy addition. In the same way, the update time is extracted from the online monitoring system, converted to a presentable form and then displayed.

Measured steel temperatures are also extracted from the system and then used to calculate and display some of the outputs like the RMSE overall, the RMSE current. The measured steel temperature is used to calculate the adjust factor by which the user can adjusts the model.

The RMSE is calculated as,

\[
RMSE(y) = \sqrt{\frac{1}{N-1} \sum_{k=1}^{N} (y(k) - \hat{y}(k))^2} \quad \text{............... (3.1)}
\]

where \(y(k)\) is the measured temperature and \(\hat{y}(k)\) is the modeled temperature.

The adjust factor at any instant is the amount of error \(e\) and is given as,

\[
e(k) = y(k) - \hat{y}(k) \quad \text{............... (3.2)}
\]

where \(y(k)\) is the measured temperature and \(\hat{y}(k)\) is the modeled temperature.

The inputs from the user are switching the X/Y gridlines ON/OFF and setting the plot limits. The user also has the ability to control the model by being able to adjust the model to a certain measured temperature.
3.2. Common features.

There are some common features of the GUI, which, whatever tab is active, remain visible. These features are common because mostly they are crucial for the functioning of the GUI. Figure 3.2 below shows the common features of the GUI.

![Common features](image)

Figure 3.2: The common features.

The Outokumpu logo is one of the common features, although it is functionally of no use, but it serves a purpose as it represents the company for which the GUI is designed.

An important common feature is the tab panel. This is a panel that has seven different tabs. The user can easily switch between the tabs to observe the status of different parameters. These tabs will be discussed in next sections of this report.

Then there are buttons. Buttons provided for control purposes are Start, Stop, Save, Close, Set plot limits and Adjust. Pressing the Start button starts the simulation. Data is acquired from the online monitoring system and the model then estimates the temperature values accordingly. The temperature values calculated by the model are then displayed by the GUI. When the simulation is not running the Stop button is inactive as there is nothing to stop and the Start button is active. But once the simulation is started, the Stop button becomes active and the Start button turns inactive. Figure 3.3 shows the situation where the simulation has started.
Disabling the Start button helps prevent an accidental restart of the simulation. This disabling is necessary since the simulation can run for as long as 10 days and 10 hours (250 hours in total) and any accidental restart would destroy the simulation data in the GUI. An autosave function has been implemented which saves important temperature data after every 10 minutes but that saved data cannot be loaded in the GUI again. It can only be manipulated manually in Matlab.

Pressing the Stop button pops up a dialog box which asks “Are you sure you want to stop?” Pressing Yes will stop the simulation while pressing No will resume the simulation. This dialog box is also a safety measure to avoid accidental stop since an accidental stop will leave the user with an incomplete data of the process. The pop up dialog box for the Stop button is shown in Figure 3.4 below.

The Save button saves some important data of the process. It doesn't matter whether the Save button is pressed after or before stopping. It will save the data in either case. When the Save button is pressed, a pop-up window appears asking for the name and location of the .mat file to save. This helps the operator to give proper names to the data files and save the data to a desired location. These data files can then be manipulated in Matlab whenever required. The data vectors saved by pressing the Save button are,
• The steel temperatures
• The inner ladle wall temperatures
• The ladle wall temperature at an intermediate point
• The outer ladle wall temperature
• Heating time in the ladle furnace
• Cooling effects of alloy additions
• History of steel presence in the ladle
• The temperature at tapping for each heat
• Steel weight during each heat
• History of preheating
• Heat numbers
• Update time
• The messages and warning log
• The measured heat temperatures
• The update time of each sample

and

• The points where the model temperature was adjusted to the measured temperatures

In addition to the Save button, there is also an autosave functionality used in the GUI which autosaves the data vectors mentioned above after every 10 minutes. This data is saved in the current directory of Matlab with the name “autosave.mat”.

The “Close” button, when pressed brings up a pop up window which asks “Are you sure you want to close?” Pressing Yes will close the GUI window while pressing No will maintain the GUI window. This dialog box is also necessary because closing the GUI window will destroy the plots displayed in the GUI and the user will have to run the Matlab code again in order to bring up the GUI window again. This dialog box helps avoiding accidental closure. Also note that if the simulation is running and the user chooses to close the GUI window, the simulation will stop and the window will be closed. The pop up dialog box for the Stop button is shown in Figure 3.5 below.

![Figure 3.5: The pop up dialog box for Close.](image)

The close sign on the title bar of the GUI window, if pressed, will also bring up the same pop up dialog box. So the control of the GUI is made safe enough to avoid undesired and accidental stops and closures.
Common features of the GUI also include some plot-viewing tools. The “Set plot limits” button is part of those plot viewing tools and works in conjunction with the editboxes that set the plot limits. These editboxes help observing the temperatures during desired time intervals. The editboxes are integrated in the statement at the bottom of the GUI, just above the space where the current message is displayed. The statement goes like “Plot data between value_{th} and value_{th} hour.” By providing the limits for viewing the plots (in hours), the temperature plots during the desired interval can be seen. Values of the desired plotting limits are set at the editboxes and to make those limits work, the “Set plot limits” button needs to be pressed. All plots will be displayed between the requested plot limits. The default plot-view setting is set to show the previous five hours of the simulation.

The ladle simulation is very sensitive and in order to achieve meaningful results, it needs to run without encountering an error. To ensure the GUI would not encounter an error and eventually stop due to user input, coding for the plot limits was done such that if the user enters anything other than a number as the plot limit, the default plot limits will take effect. Unlike shifting between the tabs, all new plot limits take effect on the next sample. The option of setting the plot limits also serves as the zoom tool for the GUI. The user can view the plots from minutes to any number of hours and at any interval.

Another plot-viewing tool is the grid checkboxes. Checking the “X Grid” and “Y grid” checkbox will result in the display of the respective gridlines in the plots. These gridlines appear as soon as the checkboxes are checked. An example of the GUI interface with the both the grids on is shown in Figure 3.6 below.

![Image of GUI interface with both grids on]

**Figure 3.6:** The GUI with the both the grids switched on.

Different types of steel qualities have different requirements for the addition of alloy type and amounts. The liquid steel temperature also needs to be kept at a specific level. The alloys are mixed in the steel and arc heating is performed to maintain the steel
temperature at the required level. This mixing and arc heating occurs at the ladle furnace stage. Temperature readings are taken so that the operator may know if the temperature is at the required level or not. The measured steel temperature is plotted in the same plot with the model steel temperature. Usually, the model steel temperature should be close to the measured steel temperature, but sometimes the operator might note that the model steel temperature is not correct and he/she might decide to adjust the model steel temperature to the measured steel temperature. The Adjust button does this operation.

When the Adjust button is pressed and confirmed, the model steel temperature will adjust to the measured steel temperature. This can be seen as some sort of correction. The operator may decide not to adjust the temperature. If the operator does not press the Adjust button, the measured temperature will be plotted but it will not have any effect on the model temperature.

If a temperature measurement is taken and the operator does not decide to adjust the temperatures, the variable holding the value of the adjustment factor of the steel temperature will hold that value until the next measurement is taken. When the next temperature measurement is taken, the new value of the temperature adjustment factor will replace the old adjustment factor.

The adjustment factor is calculated as,

\[ \text{Adjust factor} = \text{measured temperature} - \text{model temperature at that instant} \]

As this temperature adjustment is also a very important operation, the GUI has been designed such that a pop up dialog box will appear when the Adjust button is pressed. The dialog box asks “Are you sure you want to adjust the simulation to the latest measured temperature?”. Pressing Yes will adjust the model temperature by the calculated adjustment factor. Pressing No will leave the model unchanged. The pop up dialog box is shown in Figure 3.7 below.

![Confirm Adjustment](image)

**Figure 3.7:** The pop up dialog box for Adjust.

The Adjust button is inactive when there is no temperature measurement. When a temperature measurement is taken, the Adjust button turns active representing that adjustment can be made. Once the Adjust button is clicked and an adjustment is made, the Adjust button will turn inactive again till the next steel temperature measurement is taken.

Another common feature is the “Current message” which shows any message or warning at any time during the simulation. An error or warning message is displayed when the model fails to acquire required data from the online monitoring system or some data is not available.
The latest error message is displayed at the bottom of the GUI window in red bold font. The complete error log is maintained and is available at the “Messages” tab. There can be more than one error or warning message at an instant during the simulation. The “Current message” will display only the latest message and the rest of the messages will not be shown. So it is necessary to visit the “Messages” tab whenever a message appears at the bottom of the GUI to check if there are more error messages at that minute. A typical error message appearing at the bottom of the GUI window is shown in Figure 3.8 below.

Moreover failure to read a variable from the online monitoring system can result in multiple types of errors. This is because some variables are dependent on other variables and failure to read a master variable can result in failure to read the dependent variables too.

The list of warning and error messages is given below.

- Ladle temperature inappropriate
- Steel in ladle for a long time
- Errors in the file
- Steel detection failed
- Heat number failed
- Preheating status failed
- Temperature at tapping failed
- Steel weight failed
- No steel measurements
- No alloy addition data available
- No furnace heating data available
3.3. The “General” tab.

The General tab displays the current values of different parameters. A preview of the “General” window is shown in Figure 3.9 below. The “Ladle status” is displayed. It has three possible values. “Filled”, “Preheating” or “Empty”. If the ladle is filled with molten steel, it will display “Filled”. If the ladle is empty then it will display “Empty” and if it is at the preheating station, it will display “Preheating”.

The “Heat Number” shows the heat number of the ladle. Then the current, inner ladle wall, wall temperature at medium depth and the outer wall temperature are displayed. After that follows the initial steel temperature that the steel had at tapping. All temperatures are displayed in degree Celsius. Heating time in the ladle furnace shows the amount of time in seconds for which the filled ladle was given bursts of heating once it reached the secondary refining stage. Then the cooling effect due to the addition of different alloys is displayed.

The GUI also shows the update time and the steel weight. Update time is the time when the online monitoring system updated its values. It not only shows the current time, but also the date. Root mean square error (RMSE) is displayed for the current heat and for the overall duration of 250 hours. When the ladle is empty, the current steel temperature and the temperature at tapping displays N/A. The “RMSE current heat” also displays N/A. The steel temperature displays 0.


3.4. The “Steel temperature” tab.

This tab shows the current temperature of the steel. Steel temperatures are plotted on the scale with hours on the x-axis and temperature in degree Celsius on the y-axis. Although the time scale is in hours, but the plot is updated every minute in an online manner. The plot is a strip chart type plot where the old data moves out of the scale to the left while new data is added to the right. When any measurement of steel is acquired at the ladle furnace, those measurements appear as green dots in the plot. And if the ladle model is adjusted to that measured temperature, a red circle will appear at the instant of adjustment.

When the ladle is empty, there is no steel temperature available. Nothing is plotted in that case. But the old data is maintained throughout the duration of the simulation. Figure 3.10 below shows the “Steel temperature” tab. This specific example shows the results achieved from online simulation of steel temperatures. The simulation was done for two heats over the duration of 10 hours. No model adjustments were made. The figure shows the steel temperature activity for the whole duration of the simulation.

The X and Y gridlines can be switched on/off from the checkboxes on the right while the resolution of the plot can be set from the editboxes integrated in a sentence, provided below the plot. As described earlier, any duration of time on the x-axis can be set to view the plot. By default, the last five hours of activity are plotted.

Displaying steel temperature is important because it helps the operator maintain the required amount of heat content in the molten steel and it also shows the behavior of the liquid steel during its time in the ladle. Moreover, the operator can adjust the model to a correct steel temperature and have an accurate model of the process.

Figure 3.10: The “Steel temperature” tab.

Muhammad Faheem  28  Master's thesis E3775E
3.5. The “Inner ladle-wall temperature” tab.

This tab shows the temperature of the inner wall (temperature of the outer surface of the refractory brick layer) of the ladle. This plot too, is updated in an online manner. Whether the ladle is filled, empty or is at the preheating station, this temperature will be plotted. The temperature of the inner wall of the ladle is plotted on the scale with hours on the x-axis and temperature in degree Celsius on the y-axis. Same as in the steel temperature tab, the time scale is in hours, but the plot is updated every minute. The plot is a strip chart type plot where new data is added to the right pushing the old data out to the left.

This plot is necessary to track the heat content of the inner ladle wall. The ladle wall temperature must be high so that the molten steel doesn’t lose much of its heat content to the ladle wall. If the plot shows a decrease in the ladle wall temperature before tapping, the operator will be able to decide the exact amount of time the ladle must be kept under the flame for preheating. The plot is useful because it not only shows the current temperature of the inner wall but it also shows whether the trend is upwards or downwards. The inner ladle wall temperature also helps control the steel temperature. If the inner ladle wall temperature is below 760 degree Celsius, then a warning will be displayed telling the operator that the ladle is not ready to be tapped and hence heating at the ladle preheating station will be carried out. Figure 3.11 below shows this tab.

![Figure 3.11: The “Inner ladle-wall temperature” tab.](image)

In this particular case, the ladle was charged at the beginning of the simulation as the temperature was suitable for the molten steel to be poured into the ladle. The steel remained in the ladle for the next three hours and the temperature of the wall increased. After that, the ladle remained empty for about two hours. During this time, the ladle temperature decreased. Since the temperature of the ladle was suitable even after the empty time, the ladle...
was charged again without preheating. The ladle remained filled for the next three hours and in the process, increased the heat content of the ladle wall. During the last two hours of the simulation, the ladle remained empty and again a downward trend was noticed in the ladle wall temperature during that time.


This tab shows the temperature of the ladle wall at the outer boundary of the mass layer inside the wall. Whether the ladle is filled, empty or is at the preheating station, this temperature will be plotted. Same as with other previously described temperature plotting tabs, the temperature of the ladle wall at the outer boundary of the mass layer is plotted on the scale with hours on the x-axis and temperature in degree Celsius on the y-axis. The time scale is in hours, but the plot is updated every minute.

As for all plots, the X and Y gridlines can be switched on/off from the checkboxes on the right while the resolution of the plot can be set from the editboxes integrated in a sentence, provided below the plot.

For the purposes of not losing the heat content of the molten steel and strength to sustain high temperatures, the ladle wall is made up of several layers. This plot has its own importance because it shows the operator the heat lost from the molten steel and it also shows how hot or cold the overall ladle structure is becoming.

Figure 3.12. The “Intermediate ladle-wall temperature” tab.

Figure 3.12 above shows this tab. In this particular case, the ladle was filled for the first three hours, and then it was empty for the next two hours. Then the ladle was filled again and remained so for the next three hours. During the last two hours, the ladle remained
empty. It takes a little time for the heat to be transferred to the mass layer of the ladle wall. This effect can be seen prominently at the start of the simulation. The ladle was filled at the very start of the simulation but the decrease in the temperature of the mass layer continued for about half an hour.

3.7. The “Outer ladle-wall temperature” tab.

This tab shows the temperature of the outer shell of the ladle wall. Like previous temperature plots, this plot too, is updated in an online manner every minute. Whether the ladle is filled, empty or is at the preheating station, this temperature will be plotted. This temperature is plotted on the scale with hours on the x-axis and temperature in degree Celsius on the y-axis.

Figure 3.13 below shows this tab. In this particular case, the ladle was filled for the first three hours and then it was empty for the next two hours. Then the ladle was filled again and remained so for the next three hours. During the last two hours, the ladle remained empty. When the ladle is filled, the temperature of the outer wall raises implying that the heat gained by conduction from the inner wall is greater than the heat radiated to the surroundings and more heat is lost from the inner layers. The delay effect can be seen here too. When the ladle is filled, the increase in the shell temperature is not seen until a little time is passed. This is because it takes time for the heat to be transferred from the inner layers to the outer layers. The same is true when the ladle turns empty. The temperature didn’t start decreasing at the very moment when the ladle turned empty. It took some time for the downward trend to be seen when the ladle was empty.

![Image of the “Outer ladle-wall temperature” tab.

Figure 3.13: The “Outer ladle-wall temperature” tab.
3.8. The “All temperatures” tab.

This tab shows all temperatures, the current steel temperature, the inner ladle wall temperature, the intermediate ladle wall temperature and the outer ladle wall temperature. The temperatures are plotted on the scale with hours on the x-axis and temperature in degree Celsius on the y-axis. The plot is updated every minute. Like all temperature plots, this plot is a strip chart type plot where the old data moves out of the scale to the left while the new data is added to the right. Figure 3.14 below shows a snapshot of this tab.

Although all of the temperatures shown in this plot are also displayed independently in tabs described previously, it was difficult to get a comparison of the different temperatures at a certain instant. This plot provides us with that comparison. The operator can view all these temperatures and analyze the behavior of the process for any period of time in any resolution. The differences in magnitude of the temperatures can be easily viewed. The purpose of this tab is to provide a bird’s eye view of the process temperatures. If any detailed and clearer view of a specific temperature activity is required, then that can be viewed in that temperature’s specific tab.

Figure 3.14: The “All temperatures” tab.
3.9. The “Messages” tab.

This tab shows a list of any error and warning messages that have appeared throughout the process. An error or warning message can occur when the model fails to acquire required data from the online monitoring system or some data is not available. A message displayed cannot be only an error or a warning. It can be a general message just informing the operator that a certain data is not available. Immediate messages can be seen at the bottom of the GUI but as explained earlier, there can be more than one message at a time and the space available at the bottom of the GUI allows only for the latest message to be shown. So to confirm whether the displayed message is the only message at that particular instant, the operator has to check this “Messages” tab. The messages displayed in this tab are also saved in a cell array. A typical “Messages” tab is displayed in the Figure 3.15 below.

![Figure 3.15: The “Messages” tab.](image)

The structure of the message log is such that it displays the date and time at which the message was displayed and then the message follows. This helps the operator to relate a message to some time scale in the process. As discussed earlier, a failure to read a variable from the online monitoring system can result in multiple types of errors. This is because some variables are dependent on other variables and failure to read a master variable can result in failure to read dependent variables too.

A warning message of “Steel in ladle for a long time” will be displayed when the steel remains in the ladle for 180 minutes. Another such warning “Ladle temperature inappropriate” will be displayed if the temperature of the inner ladle wall is less than 760 degree Celsius.
The complete list of warning and error messages is given below.

- Ladle temperature inappropriate
- Steel in ladle for a long time
- Errors in the file
- Steel detection failed
- Heat number failed
- Preheating status failed
- Temperature at tapping failed
- Steel weight failed
- No steel measurements
- No alloy addition data available
- No furnace heating data available

All of the messages above except the first two are generated by the file that is tasked to acquire the data from the online monitoring system. No message will cause the simulation to stop.

**3.10. Running the simulation.**

Running the simulation requires the user to set the current directory as the directory where all the ladle files are stored. Then the user has just to type the name of the required model’s GUI on the command prompt of Matlab in order to run the simulation.

Type `ladle_GUI1` on the command prompt to run the simulation for model 1.

Type `ladle_GUI2a` on the command prompt to run the simulation for model 2 variant a.

Type `ladle_GUI2b` on the command prompt to run the simulation for model 2 variant b.

These different models were formed when some parameters of the model were changed in order to get the best model. A short description of different models and the variants is given in Appendix A. The tests on the simulation GUI were performed and presented in the previous sections when different tabs of the GUI were explained.
4. The GUI code.

GUIDEs have never been front-end only applications. Every GUI has back-end programming that makes them work hence providing a user friendly interface for some otherwise difficult to use functions. Matlab is no exception. A Matlab tool with the name GUIDE is available that helps build a GUI for scientific functions. With the GUIDE tool, the front end of the Matlab GUI is designed. Adequate changes are then made to the code file generated automatically for the GUI to work properly.

Without a GUI, many scientific functions and procedures would have been “Scientist only”. So, once designed by an expert, in order to enable a layman to use the same scientific functions and do the same scientific calculations, GUIDEs in Matlab are made using the GUIDE tool.

The purpose of this chapter is to discuss the general algorithms and functions that made up the GUI for the steel ladle process. The code for the ladle GUI is provided in Appendix B.

4.1. Callbacks.

There are many functions used by the GUI. The general structure of these functions is created automatically when the designer works with the GUIDE tool to create the GUI. Then that skeleton is properly programmed and connected to scientific functions in order make the simulations GUI work properly.

The functions are called “callbacks” in general. They are called “callbacks” because the programmer associates a specific component of the GUI to it and callbacks control GUI or component behavior by performing some action in response to an event for its component [18]. The event can be a mouse click on a pushbutton or a selection on a listbox etc. When an event occurs for the component, Matlab invokes the component callback that is associated with that event [18]. As an example, suppose a GUI has a pushbutton that triggers the plotting of some data. When the user clicks the button, the software calls the callback associated with the clicking of that button, and the callback which has been programmed, gets the data and plots it [18].

GUIDE provides three arguments to a callback, always named the same [18]. The first one is hObject. It is the handle of the object for which the callback was triggered. It is used to obtain relevant properties that the callback code uses and changes them as necessary [18]. The second one is eventdata. It is a stream of data describing user gestures such as key presses, scroll wheel movements and mouse drags [18]. The third one is handles. This is a Matlab struct that contains the handles of all the objects in the GUI [18].

Apart from these default arguments, more input arguments can also be defined by the designer.
4.2. Ladle GUI callbacks.

The list of functions that created the Ladle GUI and work on its back-end is given below. Pieces of code were added to some of these functions in order to make the features of the GUI work properly.

- ladle_GUI2_OpeningFcn
- ladle_GUI2_OutputFcn
- selection_Callback
- selection_CreateFcn
- start_test_Callback
- close_GUI_Callback
- figure1_CloseRequestFcn
- stop_sim_Callback
- save_data_Callback
- message_box_Callback
- message_box_CreateFcn
- grid_x_Callback
- grid_y_Callback
- adjust_Callback
- lim1_Callback
- lim1_CreateFcn
- lim2_Callback
- lim2_CreateFcn
- set_limit_Callback

All of these functions will be explained one by one in the following sections.
4.2.1. ladle_GUI2_OpeningFcn:

```matlab
function ladle_GUI2_OpeningFcn(hObject, eventdata, handles, varargin)
```

It is the first callback in every GUI M-file [18]. It is executed just before the GUI is made visible to the user but after all the components have been created [18]. An opening function OpeningFcn, in general, performs the initialization tasks before the user has access to the GUI [18]. Changes were made to this function. The tabs were properly initialized and assigned in this function here. The positions of the tabs and their visibility were set. The logo of Outokumpu was loaded from an array and displayed in the GUI. The Stop button was also disabled here.

4.2.2. ladle_GUI2_OutputFcn:

```matlab
function varargout = ladle_GUI2_OutputFcn(hObject, eventdata, handles)
```

OutputFcn returns the outputs that are generated during its execution, to the command prompt. No changes were necessary here so it was left as it was created.

4.2.3. selection_Callback:

```matlab
function selection_Callback(hObject, eventdata, handles)
```

Callback is used for control action. Changes were required according to the ladle GUI. Control of the tab selection was defined in this function. The tab which is clicked will be visible only.

4.2.4. selection_CreateFcn:

```matlab
function selection_CreateFcn(hObject, eventdata, handles)
```

A CreateFcn initializes the component when it is created. Here it initializes the listbox that contains the tabs. No changes were made to this function.

4.2.5. start_test_Callback:

```matlab
function start_test_Callback(hObject, eventdata, handles)
```

This is the major callback in the GUI. This callback is associated with the Start button. The whole simulation code was added here. This is because when the Start button is pressed, the whole simulation is required to start. The algorithm of the code included in this callback function will be presented on the following pages.
Start of code

Declared some global variables here.

(Theses variables were necessary as its data was needed to be shared with the file that acquires data from the online monitoring system and the save and stop functions.)

```matlab
if simulation started
    Disable the Start button.
    Enable the Stop button.
end
```

Initialize the messages for the Ladle Status.

Initialize the warning messages.

Initialize the rest of the variables.

```matlab
for = start from 1 to 15000 samples
    if Stop button pressed.
        Stop the simulation.
    end

    Wait for 60 seconds.

    Clear the "Current message" box at the bottom of the GUI.

    Acquire the data from the online monitoring system using a defined function.

    if steel is in the ladle
        Call the model defined for filled ladle.
    else
        Call the model defined for empty ladle.
    end

    Put the values achieved in a vector.

    if the ladle is empty
        Set steel temperature values to NaN so that the plot turns blank at these samples.
        Set the tapping temperature and the steel weight to zero.
    end
```
if steel temperature measurement sample taken
    Update the vector and put the temperature values in it.
end

if steel is in the ladle and the heat number is the same
    Put the measured steel temperature value in a vector.
    Put the corresponding model steel temperature in another vector.
    Calculate the current and overall Root Mean Square Error values.
    Enable the Adjust button.
    Calculate the Adjust factor value.
else
    Set the current Root Mean Square Error vector to zero.
    Set the vectors holding the model and measurement temperatures to zero.
end

if the model is adjusted
    Put the instant of adjustment in a vector so that it can be plotted.
end

if the ladle is empty
    Display “N/A” at temperature at tapping.
    Display “N/A” at current temperature.
else
    Get their values and display.
end

Display update time.
Display steel weight.
Display heat number.
Display cooling effects of alloy additions.
Display heating time in the ladle furnace.
Display inner ladle wall temperature.
Display intermediate point ladle wall temperature.
Display outer ladle wall temperature.
if steel is in the ladle and ladle not at preheating
   Display ladle status as “Filled”.
else if ladle is empty and ladle is not at preheating
   Display ladle status as “Empty”.
else if ladle is empty and ladle is at preheating
   Display ladle status as “Preheating”.
end

if steel is in the ladle
   Display RMSE for current heat.
else
   Display N/A at RMSE current heat.
end

Display overall RMSE.
Plot the current steel temperature.
Plot the inner ladle wall temperature.
Plot the intermediate point ladle wall temperature.
Plot the outer ladle wall temperature.
Plot all the above four temperatures in a single axes.
Update cell array of Error messages.

if 10 more minutes passed
   Save some important vectors.
end

if the global variable for error message 1 is not empty
   Display the error at the bottom of the GUI.
   Append it with current time and also display it at the messages tab.
end
if the global variable for error message 2 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 3 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 4 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 5 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 6 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 7 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 8 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

if the global variable for error message 9 is not empty
  Display the error at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
if steel is in the ladle
  if 3 hours passed
    Display warning at the bottom of the GUI.
    Append it with current time and also display it at the messages tab.
  end
end

if inner ladle wall temperature is less than 760 degree Celius
  Display warning at the bottom of the GUI.
  Append it with current time and also display it at the messages tab.
end

display of for loop

4.2.6. close_GUI_Callback:

function close_GUI_Callback(hObject, eventdata, handles)

This Callback is associated with the Close button. Changes were required according to the ladle GUI. So the Yes/No dialog box for Close was defined under this callback. If Yes is pressed, the GUI closes else the GUI is maintained.

4.2.7. figure1_CloseRequestFcn:

function figure1_CloseRequestFcn(hObject, eventdata, handles)

A CloseRequestFcn executes when the figure closes. This function is associated with the close sign on the title bar (top right) of the GUI window. It was required that this close must not be direct. So the same Yes/No dialog box for Close was defined under this callback. If Yes is pressed, the GUI closes else the GUI is maintained.

4.2.8. stop_sim_Callback:

function stop_sim_Callback(hObject, eventdata, handles)

This callback is associated with the Stop button. It was required that this stop must not be direct. So the same Yes/No dialog box for Stop was defined under this callback. If Yes is pressed, the simulation stops, the Start button becomes active and the Stop inactive. But if No is pressed the simulation resumes.
4.2.9. save_data_Callback:

function save_data_Callback(hObject, eventdata, handles)

This callback is associated with the Save button. A uisave command with some global variables was added to this callback. The uisave command is used to prompt the user for a filename and location. So when the Save button is pressed, a pop-up window appears asking for the name and location of the .mat file to save. The global variables saves the following data upon pressing the Save button.

- The steel temperatures
- The inner ladle wall temperatures
- The ladle wall temperature at an intermediate point
- The outer ladle wall temperature
- Heating time in the ladle furnace
- Cooling effects of alloy additions
- History of steel presence in the ladle
- The temperature at tapping for each heat
- Steel weight during each heat
- History of preheating
- Heat numbers
- Update time
- The messages and warning log
- The measured heat temperatures
- The update time of each sample

and

- The points where the model temperature was adjusted to the measured temperatures

4.2.10. message_box_Callback:

function message_box_Callback(hObject, eventdata, handles)

This is a callback for the listbox in which the error and warning messages are logged. No changes were required as the listbox is meant for display only.

4.2.11. message_box_CreateFcn:

function message_box_CreateFcn(hObject, eventdata, handles)

This callback is used to initialize the error and warning message listbox. No changes were required.
4.2.12. grid_x_Callback:

function grid_x_Callback(hObject, eventdata, handles)

This is a callback for the checkbox used for switching the X-grid ON or OFF. An if statement was added to this callback which switches the X-grid ON for all plots when the checkbox is checked and OFF otherwise.

4.2.13. grid_y_Callback:

function grid_y_Callback(hObject, eventdata, handles)

This is a callback for the checkbox used for switching the Y-grid ON or OFF. An if statement was added to this callback which switches the Y-grid ON for all plots when the checkbox is checked and OFF otherwise.

4.2.14. adjust_Callback:

function adjust_Callback(hObject, eventdata, handles)

This callback is related to the Adjust button. A piece of code was added to this callback which brings up a pop up dialog box when then the Adjust button is pressed. The pop up dialog box asks the user if he really wants to adjust the model or not. If the user confirms the adjust operation, the model is adjusted else the old model will continue to estimate the temperatures.

4.2.15. lim1_Callback:

function lim1_Callback(hObject, eventdata, handles)

This is a callback for the first editbox used in the plot-limits setting. No changes were required.

4.2.16. lim1_CreateFcn:

function lim1_CreateFcn(hObject, eventdata, handles)

This callback is used to initialize the first editbox for setting the plot limits. No changes were required.
4.2.17. **lim2_Callback:**

```matlab
function lim2_Callback(hObject, eventdata, handles)

This is a callback for the second editbox used in the plot-limits setting. No changes were required.
```

4.2.18. **lim2_CreateFcn:**

```matlab
function lim2_CreateFcn(hObject, eventdata, handles)

This callback is used to initialize the second editbox for setting the plot limits. No changes were required.
```

4.2.19. **set_limit_Callback:**

```matlab
function set_limit_Callback(hObject, eventdata, handles)

This callback is used to set the limits that are provided by the user in the two editboxes. A piece of code was added to this callback that gets the user input from the editboxes. Then it arranges the limits and assigns them to the global variables to be used in the plotting.
```

4.3. **The variable reference.**

Many variables were created ranging from strings, character arrays to double arrays and single value variables for the GUI to work properly. This section does not help in the general understanding of the working of the GUI, rather it can be used as a reference to understand the functionality of the variables. The variables are arranged in order of their occurrence in the code. The variables used in the M-file for the GUI of the ladle will be explained below.

**E_message1:** A global string to store the message “Errors in the file”. If a failure to read data from the file occurs, E_message1 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message2:** A global string to store the message “Steel detection failed”. If a failure to read the steel presence from the online monitoring system occurs, E_message2 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.
**E_message3**: A global string to store the message “*Heat number failed*”. If a failure to read the heat number from the online monitoring system occurs, E_message3 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message4**: A global string to store the message “*Preheating status failed*”. If a failure to read the preheating status from the online monitoring system occurs, E_message4 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message5**: A global string to store the message “*Temperature at tapping failed*”. If a failure to read the steel temperature at tapping from the online monitoring system occurs, E_message5 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message6**: A global string to store the message “*Steel weight failed*”. If a failure to read the steel weight from the online monitoring system occurs, E_message6 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message7**: A global string to store the message “*No steel measurements*”. If a failure to read any available steel measurement from the online monitoring system occurs, E_message7 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message8**: A global string to store the message “*No alloy addition data available*”. If a failure to read the alloy additions data from the online monitoring system occurs, E_message8 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**E_message9**: A global string to store the message “*No furnace heating data available*”. If a failure to read furnace heating data from the online monitoring system occurs, E_message9 will store this string, otherwise it will remain empty. It is declared global because it is also used by the file that acquires the data from the online monitoring system.

**Stopper**: A global flag variable used in stopping the simulation. It is declared global because it is used by both the Start and Stop buttons.

**X0**: A global variable that holds the temperature values from the previous instance used in the temperature estimation at the next instance. It is basically used in the Start button callback but it is declared global because it is also used by the Adjust button callback as the adjust factor is added to its first element when the Adjust button is pressed.
**X:** This vector stores most of the model data. The vector stores the steel temperature, the inner ladle wall temperature, the ladle wall temperature at an intermediate point, the outer ladle wall temperature, heating time in the ladle furnace, cooling effects of alloy additions, history of steel presence in the ladle, the temperature at tapping for each heat, steel weight during each heat, history of preheating and the heat number. It is declared global because it is used by the Save and Start buttons.

It has 12 rows. Each of those rows is described below.

Row 1: This row holds the model steel temperature in degree Celsius.

Row 2: This row holds some intermediate data.

Row 3: This row holds the inner ladle wall temperature in degree Celsius.

Row 4: This row holds the intermediate point ladle wall temperature in degree Celsius.

Row 5: This row holds the outer ladle wall temperature in degree Celsius.

Row 6: This row holds the time (in seconds) for which heat bursts were given to the steel at the ladle furnace. Usually this row has zero value. It only stores the value at the instant when a heat burst is given to the steel.

Row 7: This row holds the value of the cooling effects of alloy additions at any instant. Usually this row has zero value. It only stores the value at the instant when alloys are added to the steel.

Row 8: This is the steel flag. The steel flag is 1 when steel is present in the ladle and 0 otherwise.

Row 9: This row holds the temperature at tapping for a heat.

Row 10: This row holds the steel weight for a heat.

Row 11: This is the preheating flag. The preheating flag is 1 when the ladle is under the flame for preheating and 0 otherwise.

Row 12: This row holds the heat number for a heat.

**uptime_vec:** A vector that saves update times. It is declared global because it is used by the Save and Start button.

**mesg_cell:** A cell array that stores all the error and warning messages. It is declared global because it is used by the Save and Start button.
vec measured: A vector that holds the measured steel temperature values. This vector is used for plotting the measured steel temperature. It is declared as a global variable because it is used by both the Start and the Save button.

start_flag: A flag variable used to enable and disable the Start and Stop button. This flag is declared global because it is used by both Start and Stop buttons. This flag is 1 when the Start button is pressed and 0 when the Stop button is pressed.

adjuster: A flag which is 0 when there is no temperature measurement taken and 1 if there is any temperature measurement. This flag is declared global because it is used by the Start button as well as the Adjust button.

adjust_value: This variable holds the value of the adjust factor. The adjust factor is calculated as,

\[
\text{Adjust factor} = \text{measured temperature} - \text{model temperature at that instant}
\]

This variable is declared global because it is used by both the Start and the Adjust button.

indicator_flag: This flag is used in plotting of the adjust points. When the flag is 1, a circle is plotted at that instant in the plot. It is a global variable because it is used both by the Start and the Adjust button.

limit1: This variable holds the lower limit of the plot scale. It is a global variable because it is used by the Start and the Set plot limits button.

limit2: This variable holds the upper limit of the plot scale. It is a global variable because it is used by the Start and the Set plot limits button.

adjust_points: This is a vector that holds the adjust points throughout the simulations. It is used in plotting the adjust points. It is declared global because it is used by the Start as well as the Save button.

s1: A local variable to hold the string “Filled”.

s2: A local variable to hold the string “Empty”.

s3: A local variable to hold the string “Preheating”. 
warning1: A local variable to hold the string “Ladle temperature inappropriate”.

warning2: A local variable to hold the string “Steel in ladle for a long time”.

warning_counter: A counter used to activate warning2 when it exceeds 180.

string_new: A string that holds all the messages as one string. This is used as a temporary variable during displaying the error messages.

RMSE_current: Variable to hold the current RMSE.

RMSE_overall: Variable to hold the overall RMSE.

Ym1_vec: Vector used to hold model temperature values for the samples where measured temperature is available. It is used in overall RMSE calculations.

Ym2_vec: Vector used to hold model temperature values for the samples where measured temperature is available. It is used in current RMSE calculations.

Y1_vec: Vector used to hold measured temperature values. It is used in overall RMSE calculations.

Y2_vec: Vector used to hold measured temperature values. It is used in current RMSE calculations.

temp_at_tapping_temp: A temporary variable that holds the temperature at tapping just before displaying once it is converted to a string.

current_temp_temp: A temporary variable that holds the current steel temperature just before displaying once it is converted to a string.

update_time_temp: A temporary variable that holds the update time just before displaying once it is converted to a string.

steel_weight_temp: A temporary variable that holds the steel weight just before displaying once it is converted to a string.
**heat_number_temp:** A temporary variable that holds the heat number just before displaying once it is converted to a string.

**cooling_effects_temp:** A temporary variable that holds the value of cooling effects parameter just before displaying once it is converted to a string.

**ladle_heating_temp:** A temporary variable that holds heating time at the ladle furnace just before displaying once it is converted to a string.

**inner_wall_temp:** A temporary variable that holds the value of inner ladle wall temperature just before displaying once it is converted to a string.

**mid_point_temp:** A temporary variable that holds the value of the ladle wall temperature at an intermediate point just before displaying once it is converted to a string.

**outer_wall_temp:** A temporary variable that holds the value of outer ladle wall temperature just before displaying once it is converted to a string.

**rmse_curr_temp:** A temporary variable that holds the value of current RMSE just before displaying once it is converted to a string.

**rmse_ovral_temp:** A temporary variable that holds the value of overall RMSE just before displaying once it is converted to a string.

**plot_index:** A vector that stores the index for plotting. This vector is only used during plotting.

**string_old:** A temporary variable used to hold the contents of the cell array mesg_cell.

**string_new:** A temporary variable that holds the vertically concatenated contents of the cell array mesg_cell.

**E_message1_temp:** A temporary variable used to hold the error message concatenated to the date and time.

**E_message2_temp:** A temporary variable used to hold the error message concatenated to the date and time.
**E_message3_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**E_message4_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**E_message5_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**E_message6_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**E_message7_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**E_message8_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**E_message9_temp**: A temporary variable used to hold the error message concatenated to the date and time.

**Warning_time**: A temporary variable used to hold the warning message “Steel in ladle for a long time” concatenated to the date and time.

**Warning_cold**: A temporary variable used to hold the warning message “Ladle temperature inappropriate” concatenated to the date and time.

### 4.4. Reconstruction from saved data.

The data saved by the GUI during the simulation can be of great importance. It can be used for academic and research purposes as well as for an investigation in case of any malfunction. It is possible to work with the data in Matlab. All plots, the steel and the wall temperatures can be rebuild using Matlab.

A Matlab code is provided with the ladle GUI which can be used for reconstruction purposes. To reconstruct, the user has to type reconstruction on the command prompt. The file when executed will bring up a popup window. The user will load the saved data file using that window. Then the code will reconstruct all temperature plots associated with the ladle. The file will also display all error & warning messages encountered during the ladle simulation. All plots will appear in separate windows and the error messages will be displayed in the command window. Its code can be studied in Appendix C.
5. Sensitivity analysis.

When a system is being designed, proper attention needs to be given to all the parameters involved. Usually when systems or models are designed, they are designed under specific conditions. For example, an amplifier may work well indoors in normal conditions. But if the same amplifier is used outdoors, say, at a temperature of 45 degree Celsius, it may not meet the standards of performance. So the change in temperature caused the amplifier to deviate from its predicted performance. So this means there is a need to have an analysis that can help us examine the factors that affect the performance of the system. This type of analysis is called Sensitivity analysis.

In sensitivity analysis, engineers test the system or equipment for parameter values that are deviated from the normal mode values. In this way, a change in the system model or a change in parameter values is simulated. Sensitivity analysis is of great importance in the evaluation of system behavior. If system sensitivities are known, the system operator will know the system’s limitations and will arrange to avoid any conditions that lead to a worst case scenario.

Sensitivity analysis for the ladle process is also necessary. The operator must know which parameter can affect the steel temperature and to what extent. For this purpose, a sensitivity analysis for the ladle process was done. There are some very important parameters that are involved in the ladle process. For example, the conduction parameter, the radiation parameter, and the steel temperature drop parameter. Values of these parameters were varied by ±10% to check their effects. Offline simulation was performed where the ladle was kept empty for one hour. Then the ladle was kept under preheating for two hours. After that, the ladle was charged for the next three hours. The results showed that some of the parameters had little while the others have a higher influence on the performance of the ladle model. These results are for a six hours simulation only but it represents the general behavior of different parameters through the whole simulation period. Detailed simulation results are shown in the following sections.

5.1. Sensitivity for the current steel temperature.

The steel temperature is, to a great degree, influenced by the conduction parameter. Looking at the results of the six hours of offline simulation we can deduce that, for a percent increase in the conduction parameter, the steel temperature is lower than normal. And for a percent increase in decrease in the conduction parameter, the steel temperature is higher than normal. The results can be verified by looking at the RMSE values. The RMSE value for a 10% increase is 8.1076 degree Celsius and that for a 10% decrease is 9.0969 degree Celsius. So, whether the variation in the conduction parameter was positive or negative, the effect of reducing or increasing remained nearly the same in terms of magnitude. Figure 5.1 shows the effects of conduction parameter on the steel temperature. The plot is only for the last three hours of the simulation because the ladle was empty and then preheated in the first three hours.

The parameter that affects the steel temperature to the least degree is the radiation parameter. A 10% increase in the radiation parameter resulted in an RMS error value
of 0.2127 degree Celsius. On the other hand when the radiation parameter was decreased by 10%, the RMS error has a magnitude of 0.2344 degree Celsius. Figure 5.2 shows the effects of radiation parameter on the steel temperature. The steel temperature plot is only for the last three hours of the simulation because the ladle was empty and then preheated in the first three hours.

![Figure 5.1: Variation in steel temperature due to conduction parameter](image1)

![Figure 5.2: Variation in steel temperature due to radiation parameter](image2)
The conduction parameter has a large effect on the steel temperature because the main source of change in the steel temperature is conduction. The steel loses most of its lost temperature through inner ladle wall layers by conduction. So if the conduction parameter is reduced due to any environmental or good engineering procedure, a higher steel temperature can be maintained while if the conduction parameter increases due to any environmental or bad engineering procedure, a much lower temperature than usual can be the result. In the same way, the steel temperature is least affected by the radiation parameter because of the fact that there is small surface area available for radiation (on the top). Moreover, that upper surface of the liquid steel is also covered with slag hence reducing the radiation loses further.

### 5.2. Sensitivity for the inner ladle wall temperature

The refractory brick layer is sandwiched between the molten hot steel and other ladle wall layers. So in the case of inner ladle wall temperature, the main source of variation is the conduction parameter. During the simulation, an RMS error value of 22.6938 degree Celsius was recorded for a 10% increase in the conduction parameter. A 10% decrease in the conduction parameter resulted in an RMS error value of 24.9861 degree Celsius. **Figure 5.3** below show the results. When the conduction parameter is decreased by some value, the temperature loses are reduced and less heat content will flow in and then out of the inner ladle wall. This less trans-layer activity will help in reducing the steel temperature loses and a higher steel temperature can be maintained for a longer period of time. **Figure 5.1** also supports this statement. But if the conduction parameter is increased, the trans-layer activity will increase and a large amount of heat will flow into and then out of the refractory brick layer hence resulting in a large decrease in the steel temperature. The inner ladle wall temperature depends largely upon the conduction parameter because on the inner side it has contact with the molten steel and on the outer side it has contact with other ladle wall layers.

![Variation in inner ladle wall temperature due to conduction parameter](image-url)
The inner ladle wall temperature is least affected by the steel temperature drop parameter. For a six hour simulation, the RMS error value for a 10% increase in the steel temperature drop parameter is 1.1508 degree Celsius while that for a 10% decrease is 1.1733 degree Celsius. The Figure 5.4 below shows the results. It can be observed that initially the change in the parameter value has no effect but once the steel was poured into the ladle, the parameter variation starts affecting the inner ladle wall temperature. A substantial variation of this parameter for a longer duration can result in the GUI giving us false messages of “Ladle temperature inappropriate”.

Figure 5.4: Variation in inner ladle wall temperature due to steel temperature drop parameter.

5.3. Sensitivity for the ladle wall temperature at an intermediate point.

This temperature is recorded for the mass layer which is at a certain depth in the ladle wall. The mass layer lies between the refractory brick layer and the insulation layer. So in the case too, the main source of variation is the conduction parameter. During the simulation, an RMS error value of 16.8691 degree Celsius was recorded for a 10% increase in the conduction parameter. A 10% decrease in the conduction parameter resulted in an RMS error value of 18.5428 degree Celsius. When the conduction parameter is decreased by some value, the temperature loses are reduced and less heat content will flow in and then out at this point in the ladle just like the inner ladle wall. This less trans-layer activity will help increase the insulation of the steel from the outer layers hence keeping the steel at a higher temperature. But if the conduction parameter is increased, the trans-layer activity will increase and a large amount of heat will flow in and then out at this point hence resulting in a large decrease in the steel temperature. The molten steel present in the ladle, the inner ladle wall temperature and this temperature at an intermediate point in the ladle are substantially affected by the conduction parameter. The results of the variation are displayed in Figure 5.5.
In this case too, the temperature is least affected by the steel temperature drop parameter. For the simulation duration, the RMS error value for a 10% increase in the steel temperature drop parameter is 0.5868 degree Celsius while that for a 10% decrease is 0.5972 degree Celsius. The Figure 5.6 below shows the results.

Figure 5.5: Variation in temperature at an intermediate point in the ladle wall due to conduction parameter.

Figure 5.6: Variation in temperature at an intermediate point due to steel temperature drop parameter.
5.4. Sensitivity for the outer ladle wall temperature.

This temperature represents the temperature of the outer shell of the ladle. The inner face of this shell is covering the whole inner layers of the ladle wall while the outer face of this shell is in contact with the atmosphere. So because of this, the outer ladle wall temperature is affected mainly by the radiation parameter. The plot of Figure 5.7 shows the results. For the simulation duration, a 10% increase in the radiation parameter resulted in an RMS error value of 8.8162 degree Celsius while a 10% decrease resulted in a RMS value 9.8155 degree Celsius. The plot shows that if the value of the radiation parameter is increased by some amount, the radiation losses will increase. On the other hand, if the value of the radiation parameter is decreased by some amount, the radiation losses will decrease. This can be verified from the plot of Figure 5.7 below.

![Figure 5.7: Variation in outer ladle wall temperature due to radiation parameter.](image)

Again, the outer ladle wall temperature is least affected by the steel temperature drop parameter. For the simulation duration, the RMS error value for a 10% increase in the steel temperature drop parameter is 0.0849 degree Celsius while that for a 10% decrease is 0.0863 degree Celsius. The Figure 5.8 shows the results. The steel temperature drop parameter is most effective against the steel temperature but even in that case, the conduction parameter is more effective than the steel temperature drop parameter.

Moreover, the conduction parameter also affects the outer ladle wall temperature, but its effect is not greater than the effect of the radiation parameter.
The sensitivity analysis described above provides a better insight about the effects of different parameters. All the parameters involved in the process were tested but only the most interesting results were included in this report.

Figure 5.8: Variation in outer ladle wall temperature due to steel temperature drop parameter.
Page left blank intentionally
6. Future work.

The GUI for the ladle was designed with all the required functions. The functions of the GUI are enough for the status of the research going on at Outokumpu Avesta Works at the moment. But nevertheless, there is a huge space for improvement in the GUI. The improvements in the GUI can be helpful in improving the model and can also be helpful in introducing some changes to the system.

One such future work is to extend the GUI so that it can work for multiple ladles at a time. Currently, the GUI works for a single ladle. Some changes to the GUI needs to be done if this is desired at a later stage. A new tab selection menu will be required to be added to the current GUI. That tab selection menu will contain the names of individual ladles that are being followed. Following multiple ladles will also require a powerful computer that is able to withstand the computational load. Some changes might also be required to the online monitoring system.

Standard buttons to the GUI can be added, for example, for zoom in and zoom out. But that is possible only in the latest version of Matlab GUIDE tool. Nevertheless, the GUI still has informal zoom capability. The temperature plots of any number of samples can be viewed by setting desired time interval in the editboxes in the GUI.

Another future work might be to use the predicted temperature of the steel and the refractory brick layer to develop a model that will calculate the exact number of heats after which the refractory brick layer must be changed. This can save useful resources and can result in longer lives of the ladles. Currently, the refractory brick layer is changed after a total of 70 heats.

There might be many other possible improvements that can be done to the GUI or can be done to the system due to the GUI.
7. Conclusions.

This report mainly dealt with the GUI of the ladle’s temperature simulation. A GUI was designed that presented the different temperature plots. The GUI has different tabs and each of them presents temperature estimation from the steel and the walls. The GUI designed is quite secure and every step was taken to ensure no queries are made to the GUI unintentionally. This was made sure by the use of popup confirmation windows on important functions like Start, Stop, Adjust and Close. It was assured that the GUI will not stop functioning in any case. Even if the user mistypes the limits of the plot-viewing range, the GUI will still run the simulation without an error. This was necessary as this simulation is important and it has to run for long periods.

The reconstruction of the plots is also a feature that will be helpful in further research or analyzing any malfunction of the steel process. The GUI is quite interactive as it can change the model parameters even during the simulation by pressing the “Adjust” button. This feature gives the operator the chance to make the model and the real process as close to each other as possible. This feature has resulted in an increased human interaction of the model and has hence increased accuracy of the system.

A study of temperature and its effects was made in the first chapter. The study of the effects of temperature was carried out with the help of previous researches and personal observations. It can be concluded from the study that most of the heat was lost to the ladle walls and bottom. Another study was made and complied in the form of a literature review about different models other researchers worked on. Most of them involved partial differential equations and were solved by finite difference method.

The ODE-based model, on the basis of which this GUI was created, was studied. It was presented briefly, highlighting all the important points in the model. The results achieved by the model were also presented, picked from the paper (paper [17]) that presented the model. The results were found good enough for the model to work as an online model.

The whole GUI was explained, its functionality and its code, variables involved and the algorithm. All the tabs were presented and any function involved was presented. As explained in the “Future work” section, this GUI can be helpful in pushing the research further and can also be enhanced further to involve more than one ladle at a time.

The sensitivity analysis provided interesting results. All the parameters of the model were analyzed but only the parameters that had significant effects on the model were discussed. The discussion is aided with relevant plots. The analysis reveals that the steel temperature drop parameter was the least sensitive while the conduction parameter was the most sensitive parameter that would shift the temperature estimates by large degrees even for changes less than ±10%.

All the relevant theoretical background was studied and presented in this report. The objectives of the GUI design were achieved and its implementation successful.
Page left blank intentionally
8. References.


9. Appendices.

9.1. Appendix A: Short description of files related to the GUI & the model.

Many Matlab files were created for the model to perform. These files were used to read data from the online monitoring system and use it in the temperature model. The GUI has also some of its own files that it needs to perform perfectly. All these files are necessary and if any of these files is missing, errors will occur and the GUI is not expected to work.

There are three sub-models provided. They are designed to model the same temperature loss. But there is a difference in the parameters of the model. These models were calibrated using different calibrations.

The file "ladle_GUI1.m" is the model 1. The model is a sixth order model and has the initial parameters,

\[ x_0 = [1816 \ 1303 \ 1101 \ 1046 \ 985 \ 555] \]

The file "ladle_GUI2a.m" is the model 2 variant a. The model is a fifth order model and has the initial parameters,

\[ x_0 = [1816 \ 1303 \ 1046 \ 985 \ 555] \]

The file "ladle_GUI2b.m" is the model 2 variant b. The model is a fifth order model and has the initial parameters,

\[ x_0 = [1816 \ 1101 \ 1046 \ 985 \ 555] \]

Following are the filenames and a short description of what they are used for.

1. autosave.mat:

This file is created automatically after every 10 samples and it saves the following variables.

- \( X \) (A vector that stores inputs and outputs)
- \( \text{updtime_vec} \) (A vector that stores the update times throughout the process)
- \( \text{mesg_cell} \) (A cell array that stores all the messages)
- \( \text{vec_measured} \) (A vector that stores the measured steel temperatures)
2. **emptyLadle1.m:**
   The model for the process when the ladle is empty (model 1).

3. **filledLadle1.m:**
   The model for the process when the ladle is filled (model 1).

4. **emptyLadle2.m:**
   The model for the process when the ladle is empty (model 2, variant a & b).

5. **filledLadle2.m:**
   The model for the process when the ladle is filled (model 2, variant a & b).

6. **Ladle_GUI1.fig:**
   The .fig file of the GUI for the model (model 1).

7. **Ladle_GUI1.m:**
   The .m file of the GUI for the model (model 1). It has initial parameters as, x0=[1816 1303 1101 1046 985 555].

8. **Ladle_GUI2a.fig:**
   The .fig file of the GUI for the model (model 2 variant a).

9. **Ladle_GUI2a.m:**
   The .m file of the GUI for the variant “a” of model 2. It has initial parameters as x0=[1816 1303 1046 985 555].

10. **Ladle_GUI2b.fig:**
    The .fig file of the GUI for the model (model 2 variant b).

11. **Ladle_GUI2b.m:**
    The .m file of the GUI for the variant “b” of model 2. It has initial parameters as x0=[1816 1101 1046 985 555].
12. **LadleData_TeemingLadle.xml:**

   The file from which this process model receives data. The data from this file is used in the model. This file is accessed at each sample.

13. **makeStructFromNode.m:**

   File used for acquiring data from the **LadleData_TeemingLadle.xml** file.

14. **parseAttributes:**

   File used for acquiring data from the **LadleData_TeemingLadle.xml** file.

15. **parseChildNodes:**

   File used for acquiring data from the **LadleData_TeemingLadle.xml** file.

16. **timecast.m:**

   File used for acquiring update time from the **LadleData_TeemingLadle.xml** file.

17. **readData.m:**

   The main file that reads the data from the **LadleData_TeemingLadle.xml** file.

18. **OutoKumpu.mat:**

   The file that stores the Outokumpu logo as a matrix and this file is loaded when the file **Ladle_GUI** is simulated. The logo is displayed in the GUI window.

19. **modalclose.fig:**

   The .fig file of the dialog box that appears when the Close button is pressed or the close is clicked on the title bar (top right).

20. **modalclose.m:**

   The .m file of the dialog box that appears when the Close button is pressed or the close is clicked on the title bar (top right).
21. modalstop.fig:

The .fig file of the dialog box that appears when the Stop button is pressed.

22. modalstop.m:

The .m file of the dialog box that appears when the Stop button is pressed.

23. modaladjust.fig:

The .fig file of the dialog box that appears when the Adjust button is pressed.

24. modaladjust.m:

The .m file of the dialog box that appears when the Adjust button is pressed.

25. reconstruction.m:

The .m file used to reconstruct the all the plots from saved data file.
9.2. Appendix B: Code for the GUI.

```matlab
function varargout = ladle_GUI2b(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @ladle_GUI2b_OpeningFcn, ...
    'gui_OutputFcn', @ladle_GUI2b_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function ladle_GUI2b_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Defining listbox panels.
panels={handles.general_tab handles.steel_temperature_tab
    handles.inner_wall_tab handles.mid_point_tab handles.outer_wall_tab
    handles.all_temperatures handles.messages};
handles.currPanel=1;

% Setting the position and visibility of the panels.
set(panels{1},'Visible','on','Position',[5 8 135 40]);
set(panels{2},'visible','off','Position',[5 8 135 40]);
set(panels{3},'visible','off','Position',[5 8 135 40]);
set(panels{4},'visible','off','Position',[5 8 135 40]);
set(panels{5},'visible','off','Position',[5 8 135 40]);
set(panels{6},'visible','off','Position',[5 8 135 40]);
set(panels{7},'visible','off','Position',[5 8 135 40]);

handles.panels=panels;
set(handles.stop_sim,'Enable','off'); % Disable the stop button.
set(handles.adjust,'Enable','off'); % Disable the temperature adjuster.

guidata(hObject, handles);

% Loading OutoKumpu logo in the GUI.
load OutoKumpu;
axes(handles.logo_ok);
image(logodata);
axis off;
```
function varargout = ladle_GUI2b_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.output;

function selection_Callback(hObject, eventdata, handles)
    % Controlling the listbox panels.
    set(handles.panels(handles.currPanel), 'Visible', 'off');
    handles.currPanel = get(hObject, 'Value');
    set(handles.panels(handles.currPanel), 'Visible', 'on');
    guidata(hObject, handles);

function selection_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject, 'BackgroundColor'),
                       get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', 'white');
    end

function start_test_Callback(hObject, eventdata, handles)

    % Declaring the Error messages as global variables in order to recieve
    % error messages from the readData file.
    global E_message1;
    global E_message2;
    global E_message3;
    global E_message4;
    global E_message5;
    global E_message6;
    global E_message7;
    global E_message8;
    global E_message9;

    global x0; % Vector with initial temperature values.

    % Variable for the Stop button to work.
    global stopper;
% Variables to saved in .mat files. Declared global as it is also used in % the save button.
global X;
global updtme_vec;
global mesg_cell;
global vec_measured;

% Disabling and Enabling the Start and Stop buttons.
global start_flag;

start_flag=1; % The simulation started.

if start_flag == 1
    set(handles.start_test,'Enable','off'); %Disable the start button.
    set(handles.stop_sim,'Enable','on'); % Enable the stop button.
end

global adjuster; % Adjusts the simulation steel temperature to measured.
global adjust_value; % Records the value for adjust.
global indicator_flag; % Used in displaying the adjust instant in the plot.

adjuster=0;
indicator_flag=0;

global limit1; % Holds lower limit of the plot scale.
global limit2; % Holds upper limit of the plot scale.

global adjust_points;
adjust_points=[]; % Records the temperature values for instant of adjustment.

% Getting the plot limits.
limit1=abs(str2num(get(handles.lim1,'String')));
limit2=abs(str2num(get(handles.lim2,'String')));

% Messages for the Ladle Status.
s1='Filled';
s2='Empty';
s3='Preheating';

% Warning messages.
warning1='Ladle temperature inappropriate';
warning2='Steel in ladle for a long time';

warning_counter=0; % Used to activate warning2.

stopper=1; % Setting the variable to 1 for the Start button to work.

string_new=[]; % Variable used in handling the error messages.

% Variables for calculating the current and overall RMSE.
RMSE_current=0;
RMSE_overall=0;
Ym1_vec=[];
Ym2_vec=[];
Y1_vec=[];
Y2_vec=[];
vec_measured=[]; % Measured steel samples for plotting.
updtme_vec=[]; % Vector containing the update times.

tsamp=60;
tic;
format long;
DT=clock;

InitialTime=datenum(DT(1),DT(2),DT(3),DT(4),DT(5),DT(6));
updtmeold=0;
ubrannold=0;
x0=[1816 1101 1046 985 555]'; % Initial steel and wall temperatures.
T=zeros(1,15000);
X=zeros(12,15000);
SteelData=[];
steelDataold=[];
TT=0;
tapptempold=0;
chNrold=0;
uvarmold=0;
ulegold=0;
Stalflagold=0;
Msold=1;
tic;
Newtime=toc+tsamp;
format long;

for i=1:15000 % 250 hours 15000 samples.
    if stopper==0 % true if the stop button is pressed.
        break % Stops the simulation.
    end

    while Newtime>=toc
        drawnow; % Keeps the GUI refreshed.
    end

    % Clearing the "Current message" box at the bottom of the GUI.
    set(handles.alert,'string','');

    [steelDatanew,Ms,uleg,tapptemp,Stalflag,ubrann,uvarm,updtme,chNr]=readData
    (updtmeold,ubrannold,tapptempold,chNrold,uvarmold,ulegold,Stalflagold,Msold,
    steelDataold);

    if chNr==chNrold
        Dataold=[];
        uvarm=0;
        uleg=0;
    end

    if chNr==chNrold
        ulegold=0;
        uvarmold=0;
if uleg ~= ulegold
    ulegdiff = uleg - ulegold;
else
    ulegdiff = 0;
end

if uvarm ~= uvarmold
    uvarmdiff = uvarm - uvarmold;
else
    uvarmdiff = 0;
end

if uvarmdiff < 0
    uvarmdiff = 0;
end

if chNr ~= chNrold
    ulegdiff = 0;
    uvarmdiff = 0;
end

if (Stalflag == 0) && (Stalflagold == 1)
    ulegdiff = 0;
    uvarmdiff = 0;
end

if Stalflag == 1
    if (tapptemp ~= tapptempold) || (chNr ~= chNrold)
        x0(1) = tapptemp + 273.15;
        %tapptempold = 0;
    end
    [t, x] = ode45(@filledLadle2, [0 1/60], x0, [], uvarmdiff, ulegdiff, Ms);
else
    [t, x] = ode45(@emptyLadle2, [0 1/60], x0, [], ubrann);
end

x0 = x(end, :)

% Putting the data into X vector and converting the temperatures from
% Kelvin to Celsius.
X(1:5, i) = x(end, :)
X(1:5, i) = X(1:5, i) - 273.15;

if Stalflag == 0 % Setting the values in case no steel in the ladle.
    X(1:5, i) = NaN;
    tapptemp = 0;
    Ms = 0;
end

% If ladle is empty then the steel temperature vector is padded with NaN
% just in order to make breaks in the plot when the ladle is empty. And
% the steel weight made zero.
X(6:12, i) = [uvarmdiff ulegdiff Stalflag tapptemp Ms ubrann chNr]'};
\begin{verbatim}
T(i)=TT;
if (length(steelDataold)<(length(steelDatanew)))
    if((length(steelDatanew)==0)\&(length(steelDataold)==0))
        SteelData=[SteelData steelDatanew'];
    end
    if((length(steelDatanew)==0)\&(length(steelDataold)==0))
        steelDatanew(length(steelDataold)+1:end,:)';
    end
end

% The vector with measured steel temperatures.
vec_measured=[vec_measured NaN];
if X(8,i)==1
    if chNr==chNrold
        if (length(steelDataold)<(length(steelDatanew)))
            vec_measured(i)=SteelData(2,end);
            % Vectors for calculating the Overall and Current RMSE.
            Ym1_vec=[Ym1_vec X(1,i)];
            Ym2_vec=[Ym2_vec X(1,i)];
            Y1_vec=[Y1_vec SteelData(2,end)];
            Y2_vec=[Y2_vec SteelData(2,end)];
            % Calculating the RMSE values.
            RMSE_overall = sqrt(sum((Y1_vec - Ym1_vec).^2)/length(Y1_vec));
            RMSE_current = sqrt(sum((Y2_vec - Ym2_vec).^2)/length(Y2_vec));
            adjuster=1;
            set(handles.adjust,'Enable','on'); % Enable the temperature adjuster.
            adjust_value=(SteelData(2,end)+273.15)-x0(1); % Recording the value to adjust.
        end
    else
        % Emptying the current RMSE related vectors for next heat.
        Ym2_vec=[{}];
        Y2_vec=[{}];
        RMSE_current=0;
    end
end
\end{verbatim}
% Making X and Y vectors for the adjustment points to be indicated in
% the plot.
if indicator_flag==1
    adjust_points=[adjust_points X(1,i)];
    indicator_flag=0;
else
    adjust_points=[adjust_points NaN];
end

if Stalflag==0  % Displaying the tapping and current temperature.
    set(handles.temp_at_tapping,'string','N/A');
    set(handles.current_temp,'string','N/A');
else
    temp_at_tapping_temp=num2str(X(9,i));
    set(handles.temp_at_tapping,'string',temp_at_tapping_temp);
    current_temp_temp=num2str(X(1,i));
    set(handles.current_temp,'string',current_temp_temp);
end

update_time_temp=datestr(updtime);  % Update time.
set(handles.update_time,'string',update_time_temp);

steel_weight_temp=num2str(X(10,i));  % Steel weight.
set(handles.steel_weight,'string',steel_weight_temp);

heat_number_temp=num2str(X(12,i));  % Heat number.
set(handles.heat_number,'string',heat_number_temp);

cooling_effects_temp=num2str(X(7,i));  % Cooling effects of alloys.
set(handles.cooling_effects,'string',cooling_effects_temp);

ladle_heating_temp=num2str(X(6,i));  % Heating time in the ladle furnace.
set(handles.ladle_heating,'string',ladle_heating_temp);

inner_wall_temp=num2str(X(3,i));  % Inner wall temperature.
set(handles.inner_wall,'string',inner_wall_temp);

mid_point_temp=num2str(X(4,i));  % Intermediate point wall temperature.
set(handles.mid_point,'string',mid_point_temp);
outer_wall_temp=num2str(X(5,i)); % Outer wall temperature.
set(handles.outer_wall,'string',outer_wall_temp);

if X(8,i)==1 && X(11,i)==0 % Displaying the ladle status.
    set(handles.ladle_status,'string',s1)
elseif X(8,i)==0 && X(11,i)==0
    set(handles.ladle_status,'string',s2)
elseif X(8,i)==0 && X(11,i)==1
    set(handles.ladle_status,'string',s3)
end

% Displaying the RMSE for the current heat.
if X(8,i)==1 % stalflag and X(8,i) can be used interchangeably.
    rmse_curr_temp=num2str(RMSE_current);
    set(handles.rmse_curr,'string',rmse_curr_temp);
else
    set(handles.rmse_curr,'string','N/A');
end

rmse_ovral_temp=num2str(RMSE_overall); % The overall RMSE.
set(handles.rmse_overall,'string',rmse_ovral_temp);

plot_index=1:i;

% Plotting the current temperature from both the model and the measured
% steel temperature samples.
axes(handles.steel_temperature);
plot(plot_index,X(1,i:-1:1),plot_index,vec_measured(1,i:-1:1),'.');
hold on;
scatter(plot_index,adjust_points(1,i:-1:1),10^2,'r');
hold off;

if (get(handles.x_grid,'Value') == get(handles.x_grid,'Max'))
    set(handles.steel_temperature,'XGrid','on');
end
if (get(handles.y_grid,'Value') == get(handles.y_grid,'Max'))
    set(handles.steel_temperature,'YGrid','on');
end

set(gca, 'XDir', 'reverse');
xlim([limit1*60-5 limit2*60]);
set(gca, 'xTick',(0:60:15000));
set(gca, 'xTickLabel',{'Now',1:250});
drawnow;
% Plotting the inner ladle wall temperature.
axes(handles.inner_wall_plot);
plot(X(3,i:-1:1));
if (get(handles.x_grid,'Value') == get(handles.x_grid,'Max'))
    set(handles.inner_wall_plot,'XGrid','on');
end
if (get(handles.y_grid,'Value') == get(handles.y_grid,'Max'))
    set(handles.inner_wall_plot,'YGrid','on');
end
set(gca, 'XDir', 'reverse');
xlim([limit1*60-5 limit2*60]);
set(gca,'xTick',(0:60:15000));
set(gca,'xTickLabel',{'Now',1:250});
drawnow;

% Plotting the ladle wall temperature at an intermediate point.
axes(handles.mid_point_plot);
plot(X(4,i:-1:1));
if (get(handles.x_grid,'Value') == get(handles.x_grid,'Max'))
    set(handles.mid_point_plot,'XGrid','on');
end
if (get(handles.y_grid,'Value') == get(handles.y_grid,'Max'))
    set(handles.mid_point_plot,'YGrid','on');
end
set(gca,'XDir', 'reverse');
xlim([limit1*60-5 limit2*60]);
set(gca,'xTick',(0:60:15000));
set(gca,'xTickLabel',{'Now',1:250});
drawnow;

% Plotting the ladle wall temperature at the outer surface.
axes(handles.outer_wall_plot);
plot(X(5,i:-1:1));
if (get(handles.x_grid,'Value') == get(handles.x_grid,'Max'))
    set(handles.outer_wall_plot,'XGrid','on');
end
if (get(handles.y_grid,'Value') == get(handles.y_grid,'Max'))
    set(handles.outer_wall_plot,'YGrid','on');
end
set(gca,'XDir', 'reverse');
xlim([limit1*60-5 limit2*60]);
set(gca,'xTick',(0:60:15000));
set(gca,'xTickLabel',{'Now',1:250});
drawnow;

% plotting all the temperatures just in order to get a comparison.
axes(handles.all_plots);
plot(plot_index,X(1,i:-1:1),plot_index,X(3,i:-1:1),plot_index,X(4,i:-1:1),plot_index,X(5,i:-1:1));
set(gca,'XDir', 'reverse');
legend('Steel temperature', 'Inner ladle-wall temperature', ...
'Intermediate ladle-wall temperature', 'Outer ladle-wall temperature', 'Location', 'NorthWest');
xlim([limit1*60-5 limit2*60]);
set(gca,'xTick',{0:60:15000});
set(gca,'xTickLabel',{'No
w',1:250});
drawnow;

TT=TT+tsamp;
updatetimeold=updatetime;
ubrannold=ubrann;
Newtime=Newtime+tsamp;
chNrold=chNr;
tapptempold=tapptemp;
uvarmold=uvarm;
Msold=Ms;
steelDataold=steelDatanew;
ulegold=uleg;
Stalflagold=Stalflag;
updatetime_vec(i)=updatetime; % updating the update time vector.

mesg_cell=cellstr(string_new); % Updating cell array of Error messages.

if mod(i,10)==0 % Saving some important vectors.
    save autosave X updatetime_vec measg_cell vec_measured adjust_points;
end

if ~isempty(E_message1) % File error.
    string_old = get(handles.message_box,'string');
    E_message1_temp=strcat(datestr(now),E_message1);
    string_new = strvcat(E_message1_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message1);
end

if ~isempty(E_message2) % Stalflag data read error.
    string_old = get(handles.message_box,'string');
    E_message2_temp=strcat(datestr(now),E_message2);
    string_new = strvcat(E_message2_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message2);
end

if ~isempty(E_message3) && X(8,i)==1 % chNr data read error.
    string_old = get(handles.message_box,'string');
    E_message3_temp=strcat(datestr(now),E_message3);
    string_new = strvcat(E_message3_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message3);
end

if ~isempty(E_message4) % ubrann data read error.
    string_old = get(handles.message_box,'string');
    E_message4_temp=strcat(datestr(now),E_message4);
    string_new = strvcat(E_message4_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message4);
end

if ~isempty(E_message5) % tapptemp data read error.
    string_old = get(handles.message_box,'string');
    E_message5_temp=strcat(datestr(now),E_message5);
    string_new = strvcat(E_message5_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message5);
end

if ~isempty(E_message6) % Ms data read error.
    string_old = get(handles.message_box,'string');
    E_message6_temp=strcat(datestr(now),E_message6);
    string_new = strvcat(E_message6_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message6);
end

if ~isempty(E_message7) % SteelData not available.
    string_old = get(handles.message_box,'string');
    E_message7_temp=strcat(datestr(now),E_message7);
    string_new = strvcat(E_message7_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message7);
end

if ~isempty(E_message8) % uleg data not available.
    string_old = get(handles.message_box,'string');
    E_message8_temp=strcat(datestr(now),E_message8);
    string_new = strvcat(E_message8_temp,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',E_message8);
end

if ~isempty(E_message9) % uvarm data not available.
    string_old = get(handles.message_box,'string');
    E_message9_temp=strcat(datestr(now),E_message9);
    string_new = strvcat(E_message9_temp,string_old);
    set(handles.message_box,'string',string_new);
if X(8,i)==1 % The warnings.

    warning_counter=warning_counter+1;

    % If the steel remains in the ladle for three hours.
    if warning_counter>180
        string_old = get(handles.message_box, 'string');
        Warning_time=strcat(datestr(now),warning2);
        string_new = strvcat(Warning_time,string_old);
        set(handles.message_box,'string',string_new);
        set(handles.alert,'string',warning2);
    end

else
    warning_counter=0;
end

if X(3,i)<760 && X(8,i)==0 % If the steel temperature dips below the threshold.
    string_old = get(handles.message_box, 'string');
    Warning_cold=strcat(datestr(now),warning1);
    string_new = strvcat(Warning_cold,string_old);
    set(handles.message_box,'string',string_new);
    set(handles.alert,'string',warning1);
end

guidata(hObject, handles);

% The close button (GUI button).
function close_GUI_Callback(hObject, eventdata, handles)

    pos_size = get(handles.figure1, 'Position');
    % Call modaldlg with the argument 'Position'.
    user_response = modalclose('Title','Confirm Close');
    switch user_response
        case {'No'} % take no action
        case 'Yes'
% Prepare to close GUI application window
delete(handles.figure1);
end

% The title bar close button (top right).
function figure1_CloseRequestFcn(hObject, eventdata, handles)

% Get the current position of the GUI from the handles structure
% to pass to the modal dialog.
pos_size = get(handles.figure1,'Position');
% Call modaldlg with the argument 'Position'.
user_response = modalclose('Title','Confirm Close');

switch user_response
    case {'No'}
        % take no action
    case 'Yes'
        % Prepare to close GUI application window
        delete(handles.figure1);
end

% Stopping the simulation.
function stop_sim_Callback(hObject, eventdata, handles)

global stopper;
global start_flag;

% Get the current position of the GUI from the handles structure
% to pass to the modal dialog.
pos_size = get(handles.figure1,'Position');
% Call modaldlg with the argument 'Position'.
user_response = modalstop('Title','Confirm Stop');
switch user_response
    case {'No'}
        % take no action
    case 'Yes'
        % Prepare to stop GUI application window
        stopper=0;
        start_flag=0;
        set(handles.start_test,'Enable','on'); % Enable the start button.
        set(handles.stop_sim,'Enable','off'); % Disable the stop button.
end

guidata(hObject, handles);
% The save button.
function save_data_Callback(hObject, eventdata, handles)

    global X;
    global updtime_vec;
    global mesg_cell;
    global vec_measured;
    global adjust_points;

    uisave ({'X','updtime_vec','mesg_cell','vec_measured','adjust_points'});

function message_box_Callback(hObject, eventdata, handles)

function message_box_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function grid_x_Callback(hObject, eventdata, handles)

    if (get(hObject,'Value') == get(hObject,'Max'))
    % Checkbox is checked-take appropriate action.
    set(handles.steel_temperature,'XGrid','on');
    set(handles.inner_wall_plot,'XGrid','on');
    set(handles.mid_point_plot,'XGrid','on');
    set(handles.outer_wall_plot,'XGrid','on');
    else
    % Checkbox is not checked-take appropriate action.
    set(handles.steel_temperature,'XGrid','off');
    set(handles.inner_wall_plot,'XGrid','off');
    set(handles.mid_point_plot,'XGrid','off');
    set(handles.outer_wall_plot,'XGrid','off');
end

function grid_y_Callback(hObject, eventdata, handles)

    if (get(hObject,'Value') == get(hObject,'Max'))
function adjust_Callback(hObject, eventdata, handles)
    global adjuster; % Adjusts the simulation steel temperature to measured.
    global adjust_value; % Records the value for adjust.
    global x0; % Vector with initial temperature values.
    global indicator_flag; % Used in displaying the adjust instant in the plot.

    % Get the current position of the GUI from the handles structure
    % to pass to the modal dialog.
    pos_size = get(handles.figure1,'Position');
    % Call modaldlg with the argument 'Position'.
    user_response = modaladjust('Title','Confirm Adjustment');
    switch user_response
        case {'No'}
            % take no action
        case 'Yes'
            % Adjust the simulation to the latest measured temperature.
            x0(1)=x0(1)+adjust_value; % Temperature value adjusted.
            adjuster=0;
            set(handles.adjust,'Enable','off'); % Disable the temperature adjuster.
            indicator_flag=1;
    end

function lim1_Callback(hObject, eventdata, handles)

function lim1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
            get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function lim2_Callback(hObject, eventdata, handles)

end

function lim2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function set_limit_Callback(hObject, eventdata, handles)

global limit1; % Holds lower limit of the plot scale.
global limit2; % Holds upper limit of the plot scale.

limit1=abs(str2num(get(handles.lim1,'String')));
limit2=abs(str2num(get(handles.lim2,'String')));

if ~isempty(limit1) && ~isempty(limit2)
    if limit1(1)>limit2(1)
        flip_temp=limit1(1);
        limit1=limit2(1);
        limit2=flip_temp;
    elseif limit1(1)<limit2(1)
        limit1=limit1(1);
        limit2=limit2(1);
    else
        limit1=-0.08333;
        limit2=5;
    end
else
    limit1=-0.08333;
    limit2=5;
end
9.3. Appendix C: The reconstruction code.

%% Title: reconstruction.m
%%
%% Description:
%% This file is used to reconstruct all the temperature plots
%% associated with the ladle. The file will also display the error & warning
%% messages encountered during the ladle simulation.
%%
%% When this file is executed, a popup window will appear
%% asking the user to load the data file that was saved by the ladle
%% simulation GUI. When the data file is loaded, all the plots will appear
%% in separate windows and the error messages will be displayed in the
%% command window.

clear;
clc;

uiload; % Popup window to load the data

lim=length(vec_measured); % Setting the maximum value to plot.
plot_index=1:lim;

% Plotting the estimated & measured steel temperature.
figure(1);
plot(plot_index,X(1,1:lim),plot_index,vec_measured(1,1:lim),'.');
hold on;
scatter(plot_index,adjust_points(1,1:lim),10^2,'r');

% Setting the xtick & xlabel to hours.
xlim([0 lim]);
set(gca,'xTick',(0:60:lim));
set(gca,'xTickLabel',0:250);
grid on;
xlabel('Elapsed time (hours)');
ylabel('Temperature (degree Celsius)');
legend('estimated','measured');
title('Steel temperature');

% Plotting the estimated inner ladle-wall temperature.
figure(2);
plot(plot_index,X(3,1:lim));

% Setting the xtick & xlabel to hours.
xlim([0 lim]);
set(gca,'xTick',(0:60:lim));
set(gca,'xTickLabel',0:250);
grid on;
xlabel('Elapsed time (hours)');
ylabel('Temperature (degree Celsius)');
title('Inner ladle-wall temperature');
% Plotting the estimated intermediate point ladle-wall temperature.
figure(3);
plot(plot_index,X(4,1:lim));

% Setting the xtick & xlabel to hours.
xlim([0 lim]);
set(gca,'xTick',(0:60:lim));
set(gca,'xTickLabel',0:250);

grid on;
xlabel('Elapsed time (hours)');
ylabel('Temperature (degree Celsius)');
title('Intermediate point ladle-wall temperature');

% Plotting the estimated outer ladle-wall temperature.
figure(4);
plot(plot_index,X(5,1:lim));

% Setting the xtick & xlabel to hours.
xlim([0 lim]);
set(gca,'xTick',(0:60:lim));
set(gca,'xTickLabel',0:250);

grid on;
xlabel('Elapsed time (hours)');
ylabel('Temperature (degree Celsius)');
title('Outer ladle-wall temperature');

% Plotting all the temperatures.
figure(5);
plot(plot_index,X(1,1:lim),plot_index,X(3,1:lim),plot_index,X(4,1:lim),...%
plot_index,X(5,1:lim));

% Setting the xtick & xlabel to hours.
xlim([0 lim]);
set(gca,'xTick',(0:60:lim));
set(gca,'xTickLabel',0:250);

grid on;
xlabel('Elapsed time (hours)');
ylabel('Temperature (degree Celsius)');
legend('Steel','Inner ladle-wall','Intermediate ladle-wall',...%
'Outer ladle-wall','Location','NorthWest');
title('All temperatures');

% Displaying the error and warning messages.
FLIPUD(mesg_cell)
display('      ABOVE IS THE LIST OF ALL ERROR AND WARNING MESSAGES.');

% End of reconstruction code.

**Alternating Direction Implicit (ADI) method**: A finite difference method for solving parabolic and elliptical partial differential equations.

**AOD converter furnace**: A furnace used to purify the steel from carbon by using Oxygen and Argon gas. AOD stands for Argon Oxygen Decarburization.

**Artificial Neural Network (ANN)**: A computational model based on the idea of biological neural network. It used to model complex I/O relationships and find patterns in data.

**ARX model**: It is a black box model used for system identification. ARX stands for Autoregressive Exogenous.

**ARMAX model**: It is a black box model used for system identification. ARMAX stands for Autoregressive Moving Average Exogenous.

**Callback**: An executable code that is associated with a GUI component.

**Centerline segregation**: A line of impurities in the central zone of a steel slab caused due to non-uniformity of chemical composition in molten steel.

**Conduction**: The transfer of thermal energy between neighboring molecules in a substance due to temperature gradient.

**Convection**: The transfer of heat energy between a solid surface and a nearby liquid or gas in motion.

**Empty time**: The time for which the ladle remains empty.

**Finite difference method**: A method for approximating solutions to differential equations.

**Fuzzy logic**: A form of multi-valued logic dealing with problem solving in an approximate rather than precise way.

**Green skin ladle**: New ladle or the ladle that has just been brought to service after relining.

**Holding time**: The time for which a filled ladle awaits casting.

**Hot face**: The inner surface of the ladle that comes in contact with the molten steel.

**Ladle**: A container used to transfer molten metal in a steelmaking facility.

**Melt delivery temperature**: The steel temperature at the time of casting.

**OE model**: It is a black box model used for system identification. OE stands for Output Error.
**Ordinary Differential Equation (ODE):** A relation that contains functions of only one independent variable, and one or more of its derivatives with respect to that variable.

**Partial Differential Equation (PDE):** A relation involving an unknown function of several independent variables and its partial derivatives with respect to those variables.

**Refractory:** A material that retains strength at high temperatures.

**Radiation:** The transfer of heat energy through empty space.

**Sensitivity analysis:** The quantification of the sensitivity of a model to a parameter.

**Temperature gradient:** A physical quantity that describes in which direction and at what rate the temperature is changing most rapidly around a particular location.

**Thermal stratification:** The phenomena in which different temperature layers are formed in a substance.

**Tundish:** A container used to hold molten metal in a steelmaking facility.