LiU CPgui: A Toolbox for Parameterizing Compressor Models

Xavier Llamas and Lars Eriksson

Vehicular Systems, Department of Electrical Engineering Linköping University, SE-581 33 Linöping, Sweden
E-mail: {xavier.llamas.comellas, lars.eriksson}@liu.se

Technical Report: LiTH-ISY-R-3102

January 15, 2018

Abstract

A toolbox for parameterizing the ellipse model, that is a control-oriented compressor model, to any given measured compressor map is described in detail in this document. The compressor model has been developed in previous publications and shown to be capable of accurately reproducing the measured data obtained from gas stand measurements, for a wide range of compressors, starting from small automotive applications to large compressors used in marine propulsion. In addition, it has been shown that it is possible to extrapolate both mass flow and efficiency to the unmeasured low speed region of the compressor in a physical way. The parameterization algorithm is based on Total Least Squares (TLS), which is shown here and in previous publications to be a fast and reliable approach to fit the compressor model to the map. The toolbox is implemented in a Matlab Graphical User Interface (GUI) in order to make it easy for the user to parameterize the compressor model. To demonstrate the workflow and ease of use, a complete step-by-step example of how to work with the toolbox is provided. To further facilitate the user in applying the model, the package also provides implementations of the ellipse compressor model both as a Matlab function and as a Simulink block. This way, the user can quickly and reliably use the results of the parameterization process in a desired application, e.g. including the compressor model of a given compressor map in a combustion engine simulation model.

Contents

1 Introduction

LiU CPgui is a compressor model parameterization package implemented as a Matlab Graphical User Interface (GUI). The GUI is built with the intention to simplify the parameterization process of the complete compressor model proposed, thus making it easy for any user to input a measured compressor map and obtain the model parameters. This report contains a description of the implemented model and parameterization procedure together with a tutorial about how to work with LiU CPgui. The package work flow is explained with the help of a step-by-step parameterization example using the toolbox.

1.1 Report outline

Section 2 contains an introduction to the measured compressor performance maps and which signals are usually included. Section 3 describes the heat correction method implemented in the LiU CPgui package that enables the user to obtain the compressor adiabatic efficiency if needed. Section 4 introduces the model equations for the compressor flow and efficiency. The parameterization algorithm is described in Section 5. How to work with the toolbox is illustrated in Section 6, together with a step-by-step example.

2 Compressor performance maps

Compressor performance is normally measured together with a turbine in special turbocharger test benches called gas stands. The measurements are done following a standard procedure, see SAE [1]. For the compressor case, the provided data usually include; inlet and outlet total pressures, p_{01} and p_{02}, inlet and outlet total temperatures, T_{01} and T_{02}, mass flow, W_c, and rotational speed ω_c. Figure 1 contains a diagram of the compressor with the location of the surrounding measurements.

To reduce the amount of measurements necessary to determine the compressor performance, the compressor maps are usually provided in corrected variables. This is done to cover different inlet conditions with a single set of measurements. The corrected quantities are defined as in [2]

$$\bar{N}_c = N_c \frac{1}{\sqrt{T_{01}/T_{c,ref}}} \tag{1}$$

$$\bar{W}_c = W_c \frac{\sqrt{T_{01}/T_{c,ref}}}{p_{01}/p_{c,ref}} \tag{2}$$

where $T_{c,ref}$ and $p_{c,ref}$ are the reference values included in the map. The compressor speed is usually given in revolutions per minute. Finally, the included map signals are pressure ratio $\Pi_c = \frac{p_{02}}{p_{01}}$, efficiency η_c, corrected speed $\bar{N}_c$ and corrected mass flow $\bar{W}_c$.

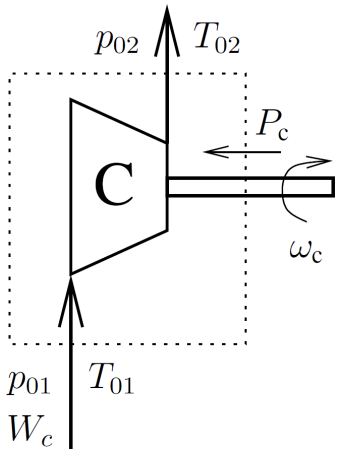

Figure 1: Diagram of a compressor with the surrounding pressures and temperatures at the inlets and outlets.

The compressor performance is usually represented visually in a plot called compressor map that contains the flow characteristics together with the device efficiency. In the map, the operating points with the same speed are drawn together to form Speed Lines (SpLs). Figure 2 shows an example of a compressor map with its distinct operating zones.

3 Compressor efficiency correction

In the compressor map, the total to total efficiency is calculated from pressure and temperature measurements as

$$\eta_c = \frac{\Delta h_{is}}{\Delta h_{act}} = \frac{\frac{p_{02}}{p_{01}}^{\frac{\gamma-1}{\gamma}} - 1}{\frac{T_{02}}{T_{01}} - 1} \tag{3}$$

see Dixon and Hall [3] for more information. In gas stands, the compressor is normally measured with a hot gas turbine side that will influence the compressor temperature measurements, with a greater effect on low compressor speeds and flows. Under these conditions, the measured temperatures will not represent the temperature increase due to the air compression, since they will also contain the temperature rise corresponding to the heat transfer. As a result, the calculated efficiency will be lower than the real one. This issue has received substantial attention in the research literature, see for example [4, 5, 6, 7, 8, 9, 10, 11, 12]. Since the heat transfer in the gas stand is not the same as in the engine operation, having the right adiabatic compressor efficiency is important in order to be able to simulate the correct compressor-turbine power balance. If the right temperature at the compressor outlet has to be achieved during simulation, a model for the heat transfer present in the combustion engine operation can be included afterwards, see [5, 8].

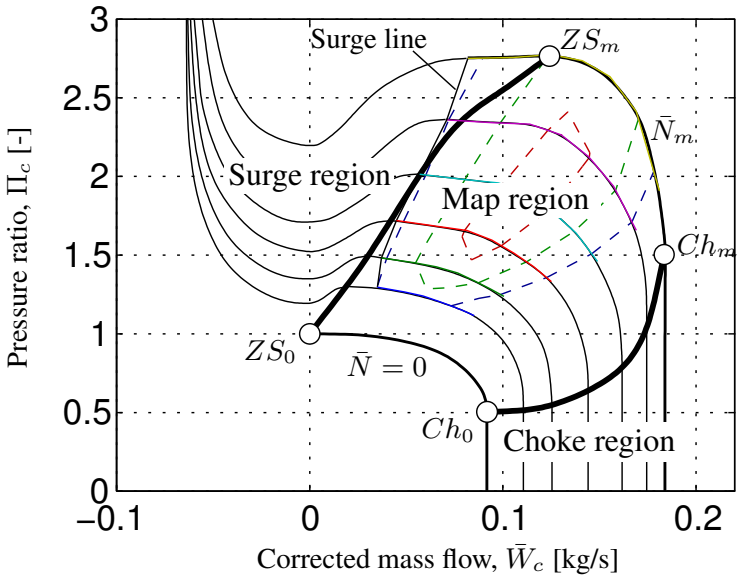

Figure 2: Compressor map sketch with different operating areas indicated. Choke line is denoted with Ch_0 to Ch_m and Zero Slope line with ZS_0 to ZS_m, while the subscripts mean 0–stand still and m–maximum speed.

The LiU CPgui toolbox implements the efficiency correction method used in Llamas and Eriksson [13]. This method is only summarized here, the authors refer to the original publication from Casey and Fesich [9] for more information. The key idea is that the different compressor SpLs should collapse into a single line in the $\lambda_{Euler} - \phi_2$ plane, unless there is heat transfer affecting the efficiency measurements. The compressor work coefficient is computed using the actual enthalpy gain as

$$\lambda = \frac{\Delta h_{act}}{U_2} \tag{4}$$

where $U_2 = \omega_c D_2/2$ is the compressor blade tip speed, with D_2 denoting the impeller diameter. The Euler compressor work coefficient can be computed from the real work coefficient using the disk friction losses factor k_{fric}, see Casey and Schlegel [14].

$$\lambda_{Euler} = \left(1 + \frac{k_{fric}}{\phi_1}\right)^{-1} \lambda \tag{5}$$

where ϕ_1 is the inlet flow coefficient calculated as

$$\phi_1 = \frac{W_c}{\rho_{01} D_2^2 U_2} \tag{6}$$

with ρ_{01} as the inlet density. With the previous definitions, the aforementioned linear relation can be derived from turbomachinery equations as

$$\lambda_{\text{Euler}} = 1 - \frac{c_s}{U_2} + \phi_2 \tan \beta_2 \tag{7}$$

where $\tan\beta_2$ is negative for a back-swept impeller, and c_s/U_2 represents the slip factor which can be considered reasonably constant. The difference between the calculated Euler work input coefficient and the adiabatic, $\lambda_{\mathrm{Euler,ad}}$, can be computed as

$$\lambda_{\mathrm{Euler}} - \lambda_{\mathrm{Euler,ad}} = \frac{k_q}{\phi_2 M_{U_2}^3} \tag{8}$$

where M_{U_2} is the tip-speed Mach number and k_q is the dimensionless constant that depends on the heat transfer rate and has to be adjusted to ensure that all SpLs collapse into a single one in the $\lambda_{Euler} - \phi_2$ plane, see Casey and Fesich [9]. Thus, by adjusting a single parameter, k_q, any compressor map can be potentially corrected. This parameter is an input option of LiU CPgui. Note that the heat correction method is only applicable if the compressor map was measured with constant turbine inlet temperature, which is the usual case with SAE maps, see SAE [1]. If the user is unsure whether or not the map is measured with constant turbine inlet temperature, the parameter k_q can be set to zero in the toolbox, which will avoid applying the proposed correction method.

Unfortunately, in order to calculate the outlet flow coefficient, ϕ_2, geometrical details of the compressor are required to compute the velocity triangles at the impeller outlet. Since these details are usually not provided with standard SAE compressor maps, the iterative method from Casey and Schlegel [14] is used in the LiU CPgui implementation to obtain ϕ_2 for vaneless automotive compressors. This iterative method requires the compressor diameter D_2 and the diffuser channel width, b_2. If the latter one is unknown, a value between $7-10\%$ of the diameter D_2 is a fair approximation as suggested in Llamas and Eriksson [13].

Figure 3 contains the Euler work coefficient plotted against outlet flow coefficient for a compressor map with and without the described correction method applied. As can be seen, the different SpLs collapse on top of each other when the correction is applied in the right side of the figure. The corresponding change in efficiency values for the same correction can be observed in Figure 4. As can be observed, only the lower SpLs are affected by the correction. This is expected since for a small flow going through the compressor, the heat transfer has the largest effect on the temperature.

To sum up, the implemented correction method only requires the user to tune one parameter, k_q. There is no guarantee that the obtained efficiency is indeed the true adiabatic efficiency, but it can be used to obtain efficiency values that are close to it. It is suitable for vaneless compressors measured with constant inlet temperatures, and knowing the impeller diameter and diffuser width is preferable but not necessary.

4 Compressor model

This Section contains a full description of the mathematical formulation of the compressor model parameterized with LiU CPgui. More details about the motiva-

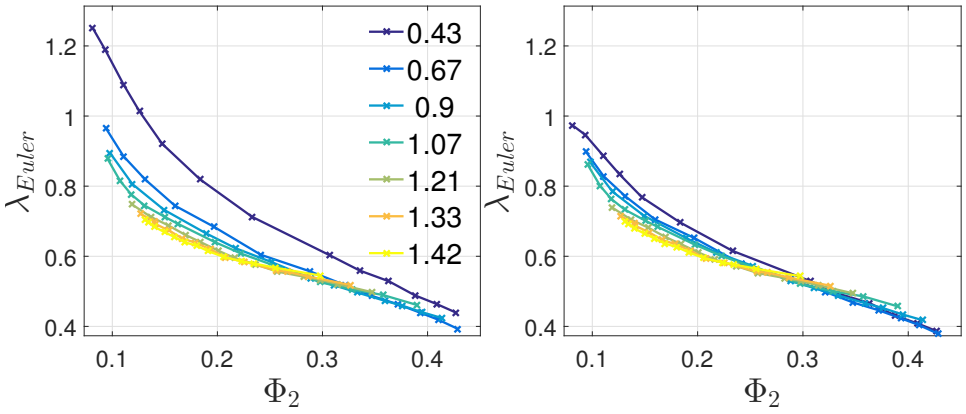

Figure 3: Euler work input coefficient vs outlet flow coefficient. Left; no heat transfer correction. Right; with the correction applied. The values in the legend correspond to different tip-speed Mach numbers.

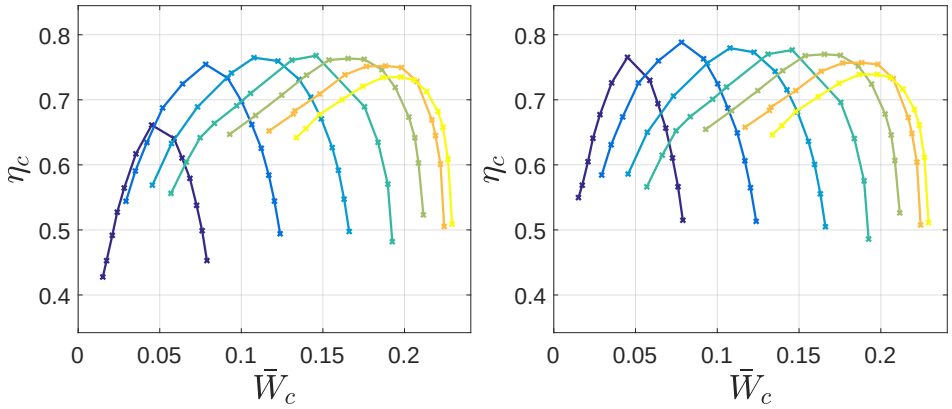

Figure 4: Efficiency vs corrected mass flow. Left; no heat transfer correction. Right; with the correction applied.

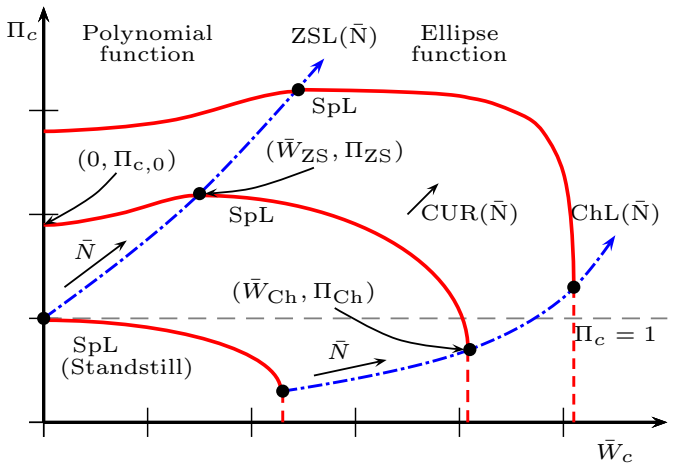

Figure 5: A sketch of the Ellipse flow model. The zero slope and choking lines are plotted in dashed-dotted blue lines. The speed lines (SpL) are drawn using red lines.

tions for the model formulation can be found in the original work [15, 16], together with the later papers [13, 17] and the provided references therein.

4.1 Ellipse mass flow model

The Ellipse flow model defines a mathematical relation between mass flow, pressure ratio and rotational speed. The main characteristics of the model are shown in the sketch of Figure 5. The Zero Slope (ZSL) and Choke lines (ChL) are defined in the sketch, together with the compressor Speed Line (SpL) at standstill. The ZSL is not necessary equal to the surge line, even if both lines can coincide at low speeds, they tend to differ at higher speeds. This can be observed in the general diagram of Figure 2. Four different equations for four distinct zones are required to completely define the Ellipse mass flow model.

4.1.1 Between the ZSL and the ChL

The main compressor operation zone is between the ZSL and the ChL, ($\bar{W}_{ZS} \leq \bar{W}_c < \bar{W}_{ch}$). In this zone, the compressor SpLs are modeled using an ellipse function, which gives the name to the mass flow model.

$$\left(\frac{\bar{W}_c - \bar{W}_{ZS}}{\bar{W}_{Ch} - \bar{W}_{ZS}}\right)^{CUR} + \left(\frac{\Pi_c - \Pi_{Ch}}{\Pi_{ZS} - \Pi_{Ch}}\right)^{CUR} = 1 \tag{9}$$

where $\bar{W}_{Ch}$, Π_{Ch}, $\bar{W}_{ZS}$, Π_{ZS} and CUR are base parameters that define the position of the modeled SpLs in the compressor maps. Note that in the notation its dependency of compressor corrected speed is dropped for simplicity. These base

functions depend on compressor corrected speed and the 14 model parameters, $C_{-,i}$.

$$\bar{W}_{Ch}(\bar{N}_{c,n}) = \bar{W}_{c,max}(C_{Wch,1} + C_{Wch,2}\arctan(C_{Wch,3}\bar{N}_{c,n} - C_{Wch,4})) \tag{10a}$$

$$\Pi_{Ch}(\bar{N}_{c,n}) = \Pi_{c,max}(C_{\Pi ch,1} + C_{\Pi ch,2}\bar{N}_{c,n}^{C_{\Pi ch,3}}) \tag{10b}$$

$$\bar{W}_{ZS}(\bar{N}_{c,n}) = \bar{W}_{c,max}(C_{Wzs,1}\bar{N}_{c,n}^{C_{Wzs,2}}) \tag{10c}$$

$$\Pi_{ZS}(\bar{N}_{c,n}) = 1 + (\Pi_{c,max} - 1)C_{\Pi zs,1}\bar{N}_{c,n}^{C_{\Pi zs,2}} \tag{10d}$$

$$CUR(\bar{N}_{c,n}) = C_{cur,1} + C_{cur,2}\bar{N}_{c,n}^{C_{cur,3}} \tag{10e}$$

In the definitions, the base functions use the normalized corrected compressor speed with the maximum measured speed of the compressor map as

$$\bar{N}_{c,n} = \bar{N}_c/\bar{N}_{c,max} \tag{11}$$

Moreover, the base functions are also normalized with the maximum measured corrected mass flow and pressure ratio in the map, $\bar{W}_{c,max}$ and $\Pi_{c,max}$.

The package LiU CPgui has the option to use another base function for the Choke mass flow ($\bar{W}_{Ch}$). This alternative base function is used in [16, 17] and is defined as a piecewise function

$$\bar{W}_{Ch}(\bar{N}_{c,n} \leq \bar{N}_{c,s}) = C_{Wch,1} + C_{Wch,2} \cdot \bar{N}_{c,n}^{C_{Wch,3}} \tag{12a}$$

$$\bar{W}_{Ch}(\bar{N}_{c,n} > \bar{N}_{c,s}) = C_{Wch,4} + C_{Wch,5} \cdot \bar{N}_{c,n} \tag{12b}$$

where the switching $\bar{N}_{c,s}$ is a parameter to be estimated. To ensure the continuity of both equations (12) at the switching speed ($\bar{N}_{c,s}$), a condition is introduced between the two equations. This continuity condition reduces the number of parameters from six to five, since the parameter $C_{Wch,4}$ has to fulfill the following equation

$$C_{Wch,4} = C_{Wch,1} + C_{Wch,2}\bar{N}_{c,s}^{C_{Wch,3}} - C_{Wch,5}\bar{N}_{c,s} \tag{13}$$

Note that the arctan base function from (10a) has one parameter less than this one.

The implicit equation (9) can be solved for either mass flow or pressure ratio. Solving for mass flow, the following equation is obtained as function of the surrounding pressures, the inlet temperature and the compressor speed

$$\bar{W}_c = \bar{W}_{ZS} + (\bar{W}_{Ch} - \bar{W}_{ZS})\left[1 - \left(\frac{\Pi_c - \Pi_{Ch}}{\Pi_{ZS} - \Pi_{Ch}}\right)^{CUR}\right]^{\frac{1}{CUR}} \tag{14}$$

On the other hand, equation (9) can also be solved for pressure ratio, in this case the explicit equation becomes

$$\Pi_c = \Pi_{Ch} + (\Pi_{ZS} - \Pi_{Ch})\left[1 - \left(\frac{\bar{W}_c - \bar{W}_{Ch}}{\bar{W}_{Ch} - \bar{W}_{ZS}}\right)^{CUR}\right]^{\frac{1}{CUR}} \tag{15}$$

4.1.2 Below the ChL

For pressure ratios smaller than the choking point, ($\Pi_c < \Pi_{Ch}$), the model is assumed to be vertical. Other approaches for this region are proposed in Leufvén and Eriksson [16], and can be useful in case the mass flow saturation gives problems during simulation.

4.1.3 Between zero flow and the ZSL

A third-order polynomial is used to model the pressure ratio for mass flows smaller than the ZS flow, ($0 < \bar{W}_c < \bar{W}_{ZS}$).

$$\Pi_c = \Pi_{c,0} + 3\frac{\Pi_{ZS} - \Pi_{c,0}}{\bar{W}_{ZS}^2}\bar{W}_{el}^2 - 2\frac{\Pi_{ZS} - \Pi_{c,0}}{\bar{W}_{ZS}^3}\bar{W}_{el}^3 \tag{16}$$

where W_{el} is an artificial mass flow, introduced to give more flexibility to the shape of the third order polynomial. More details are included in Llamas and Eriksson [13]. This artificial mass flow is defined as

$$\bar{W}_{el} = \bar{W}_{ZS}\left(1 - \left(1 - \frac{\bar{W}_c}{\bar{W}_{ZS}}\right)^{C_s}\right)^{1/C_s} \tag{17}$$

where the parameter C_s is estimated from the data. For the case $C_s = 1$, both flows are identical ($W_{el} = \bar{W}_c$). The pressure ratio at zero flow $\Pi_{c,0}$ is defined as

$$\Pi_{c,0}(\bar{N}_{c,n}) = \Pi_{ZS}(\bar{N}_{c,n}) - \Gamma_{\Pi_{cs}}(\Pi_{ZS}(\bar{N}_{c,n}) - 1) \tag{18}$$

where, if surge measurements are available, the constant $\Gamma_{\Pi_{cs}}$ can be adjusted to the desired map. Unfortunately, surge measurements are not usually available, so the suggested value from Eriksson et al. [18], $\Gamma_{\Pi_{cs}} = 1/2$, is used as a fixed value in LiU CPgui.

4.1.4 Negative mass flow

The usual assumption for negative flows is that the compressor is operating as a poorly designed turbine, see Figure 2 for a qualitative representation. Hence, a turbine mass flow model can be use to extend the compressor model in this region, see [15, 19] for specific models. Inverting the flow model to obtain pressure ratio yields

$$\Pi_c = (\Pi_{c,0} - 1) + \left(1 - \left(\frac{-\bar{W}_c}{K_0\bar{W}_{c,max}}\right)^2\right)^{\frac{-1}{K_t}} \tag{19}$$

where K_0 and K_t are model parameters, and $\Pi_{c,0}$ is function of compressor speed according to (18). Since this area is not measured, these parameters are not estimated during the model parameterization described in Section 5. Instead the user has to find the values that best represent the compressor operation during surge, see

Eriksson et al. [20] for an example. As a rule of thumb, the following parameters can be used: $K_t \approx 2$ and $K_0 \approx 0.3$. Note that the product $K_0 \cdot \bar{W}_{c,max}$ sets the location of the vertical asymptote in the negative flow area, see Figures 24 and 25 for graphical examples.

4.2 Enthalpy-based efficiency model

The Enthalpy-based efficiency model is capable of extrapolating the efficiency to low speed regions. It complements the Ellipse mass flow model and thus provides a complete representation of the compressor performance. The model is developed by starting from the physical definition of the compressor isentropic efficiency

$$\eta_c = \frac{\text{isentropic work}}{\text{actual work}} = \frac{h_{02_s} - h_{01}}{h_{02} - h_{01}} = \frac{\Delta h_{0s}}{\Delta h_{act}} \tag{20}$$

with the isentropic work defined as in [3]

$$\Delta h_{0s} = c_p T_{01} \left[\left(\frac{p_{02}}{p_{01}} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right] \tag{21}$$

where γ is the ratio of specific heats and c_p is the specific heat at constant pressure.

To carry out the compression of the gas, the compressor consumes a shaft work that can be computed using Euler's equation on the compressor speed triangles, see Dixon and Hall [3]. Rearranging the terms and taking assumptions about the flow direction, the following expression is achieved

$$\Delta h_{act} = \left(1 + k_{loss}(\bar{N}_c, \bar{W}_c)\right) \left(b(\bar{N}_{c,n}) - a(\bar{N}_{c,n})\bar{W}_c\right) \tag{22}$$

see [13] for more details. The base functions $a(\bar{N}_{c,n})$ and $b(\bar{N}_{c,n})$ are defined as

$$b(\bar{N}_{c,n}) = \Delta h_{act,max}(C_{b,1} \cdot \bar{N}_{c,n}^2 + C_{b,2} \cdot \bar{N}_{c,n}^3) \tag{23a}$$

$$a(\bar{N}_{c,n}) = \frac{\Delta h_{act,max}}{\bar{W}_{c,max}} \frac{C_{a,1}\bar{N}_{c,n}}{[1 + C_{a,2}\bar{N}_{c,n}^2]^{C_{a,3}}} \tag{23b}$$

where $\Delta h_{act,max}$ is the map's maximum actual work value, calculated using (20), (21), and the measured efficiency. The model parameters C are determined with the parameterization algorithm.

Compressor wheel friction and other losses like flow recirculation at low flows and leakage are represented in the term $k_{loss}(\bar{N}_c, \bar{W}_c)$. This loss term is calculated as

$$k_{loss}(\bar{N}_c, \bar{W}_c) = \frac{C_{loss}\rho_{01}D_2^3\pi\bar{N}_c}{60\bar{W}_c} \tag{24}$$

where C_{loss} is a model parameter, D_2 is the compressor impeller outlet diameter and ρ_{01} is the inlet density. The value of D_2 might not be provided with the compressor map, however, it is not problematic to guess it since during the parameterization the model parameters will compensate for errors done in the guessing.

The LiU CPgui has alternatives implemented for the base functions of the Enthalpy-based efficiency model. The previously described base functions correspond to the ones from Llamas and Eriksson [13], and are named SAE base functions in the toolbox. The alternative base functions from Llamas and Eriksson [17], are named ASME base functions in the toolbox and are defined as

$$b(\bar{N}_{c,n}) = \Delta h_{act,max}(C_{b,1} \cdot \bar{N}_{c,n} + C_{b,2} \cdot \bar{N}_{c,n}^2) \tag{25}$$

$$a(\bar{N}_{c,n}) = \frac{\Delta h_{act,max}}{\bar{W}_{c,max}}(C_{a,1} \cdot \bar{N}_{c,n} - C_{a,2} \cdot \bar{N}_{c,n}^2) \tag{26}$$

For this case, the losses are not considered, and thus the k_{loss} function is defined as

$$k_{loss}(\bar{N}_c, \bar{W}_c) = 0 \tag{27}$$

which results in the expression for Δh_{act} from Llamas and Eriksson [17]. The complete derivation of the base functions for the two shown cases can be found in the original publications [13, 17], together with an analysis of the model results.

4.3 Compressor model structure

The compressor model is structured to agree with a component-based modeling framework. There are two main formulations, named forward and backward implementation depending on the intended use. The forward implementation is the most common way to represent the compressor, and it provides as outputs; mass flow, efficiency, temperature and consumed power as function of the surrounding pressures, inlet temperature and speed. On the other hand, the backward implementation is suitable if the model is to be used for surge phenomena simulation in a More-Greitzer formulation [21].

4.3.1 Forward implementation

In the forward implementation the model is structured in four main functions of the surrounding pressures, temperatures and compressor speed. The four equations are defined as

$$W_c = f_{W_c}(p_{01}, p_{02}, T_{01}, \omega_c) \tag{28a}$$

$$\eta_c = f_{\eta_c}(p_{01}, p_{02}, T_{01}, \omega_c, W_c) \tag{28b}$$

$$T_c = T_{01} + \frac{T_{01}}{\eta_c}\left\{\left(\frac{p_{02}}{p_{01}}\right)^{\frac{\gamma-1}{\gamma}} - 1\right\} \tag{28c}$$

$$P_c = W_c\, c_{p,c}\, (T_c - T_{01}) \tag{28d}$$

A distinct model equation has to be defined for the zone $(0 < \bar{W}_c < \bar{W}_{ZS})$ described in Section 4.1.3 if the model is to be used in forward mode. This is because if the model is used with pressure ratio as input, two different outputs

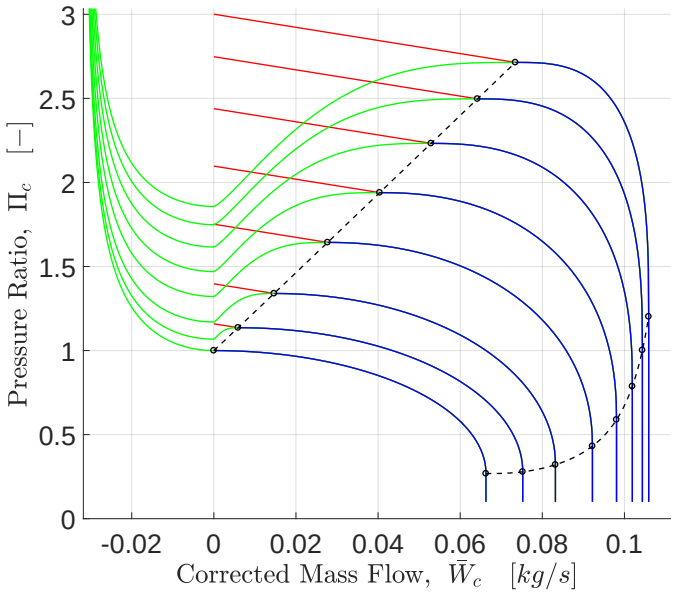

Figure 6: Compressor model with the two different implementations for the unstable area. The forward implementation is plotted in red and the backward implementation with reverse flow is drawn in green.

cannot exist for the same input value, thus a linear model is used instead in the area between zero flow and the ZSL. The mass flow in this area is computed as

$$\bar{W}_c = \bar{W}_{ZS} - \frac{\Pi_c - \Pi_{ZS}}{\alpha_{k\Pi W_0}} \tag{29}$$

where $\alpha_{k\Pi W_0}$ defines the slope of the linear function and it is defined using the maximum measured mass flow and pressure ratios in the map as

$$\alpha_{k\Pi W_0} = 0.15 \frac{\Pi_{c,max}}{\bar{W}_{c,max}} \tag{30}$$

Graphically, this modifies the SpLs in this area as shown in Figure 6, with linear SpLs that have the same slope. This parameter is not included in the estimation algorithm, and any value can be used in order to affect the slope of the SpLs in this area as desired.

Hence, for the forward mode the Ellipse mass flow model in implicit form is calculated as

$$\bar{W}_c = \begin{cases} \text{equation (29)} & \text{if} \quad \Pi_c > \Pi_{ZSL} \\ \text{equation (14)} & \text{if} \quad \Pi_{ZSL} \leq \Pi_c < \Pi_{Ch} \\ \bar{W}_{Ch} & \text{if} \quad \Pi_c < \Pi_{Ch} \end{cases} \tag{31}$$

which is inserted in (28a), after uncorrecting the mass flow. The Enthalpy-based efficiency model described in (20) with the selected subfunctions, (21) to (27), is

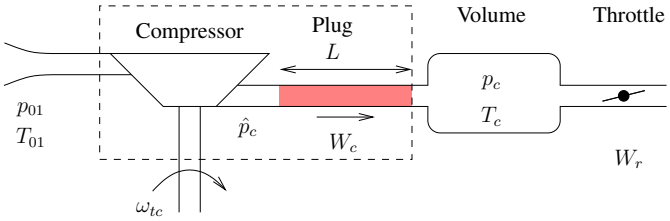

Figure 7: More-Greitzer model approach to simulate surge. A model of a plug with mass together with a control volume and a throttle are required.

inserted in (28b). The forward Simulink implementation provided with the LiU CPgui package agrees to this equation structure. It is important to mention that assuming linear SpLs in this area has no physical meaning, it is only done to have a fully defined function. During simulations, it should be checked that the model only enters this region during fast transients to later return to the ellipse equation zone. If there is interest in obtaining a more realistic performance prediction, the surge capable compressor model from the following section should be used.

4.3.2 Surge-capable backward implementation

The capacity of the ellipse equation to be inverted, enables the model to be used in for surge simulation. Surge is simulated using the More-Greitzer formulation described in [21]. Figure 7 contains a diagram of the compressor model in the More-Greitzer setup, and Figure 6 shows the unstable area speed lines in the backward implementation. The extra components are the plug with mass that models the acceleration of mass flow though the outlet pipe, and the control volume before the throttle.

The model structure then becomes

$$\frac{\mathrm{d}W_c}{\mathrm{d}t} = \frac{\pi\, D^2}{4\, L}(\hat{p}_c - p_c) \tag{32a}$$

$$\hat{p}_c = p_{01} \cdot f_{\hat{\Pi}_c}(W_c, T_{01}, \omega_c) \tag{32b}$$

$$\eta_c = f_{\eta_c}(p_{01}, p_{02}, T_{01}, \omega_c, W_c) \tag{32c}$$

$$T_c = T_{01} + \frac{T_{01}}{\eta_c}\left\{ \left(\frac{p_{02}}{p_{01}}\right)^{\frac{\gamma-1}{\gamma}} - 1 \right\} \tag{32d}$$

$$P_c = W_c\, c_{p,c}\, (T_c - T_{01}) \tag{32e}$$

where the mass flow becomes a dynamic state of the model that has to be integrated. And p_c is the dynamic pressure of the control volume downstream of the plug. The

compressor operating compression ratio is calculated as

$$f_{\hat{\Pi}_c}(W_c, T_{01}, \omega_c) = \begin{cases} \text{equation (19)} & \text{if} \quad \bar{W}_c < 0 \\ \text{equation (16)} & \text{if} \quad 0 \leq \bar{W}_c < \bar{W}_{ZSL} \\ \text{equation (15)} & \text{if} \quad \bar{W}_{ZSL} \leq \bar{W}_c < \bar{W}_{Ch} \end{cases} \tag{33}$$

with the corresponding unit changes and speed and mass flow corrections. The backward Simulink implementation provided with the LiU CPgui package follows this model structure.

5 Parameterization algorithm

With the compressor model already described, the next step is to parameterize the model to a measured compressor map. This is not a trivial task because of the nonlinearities of the proposed model, as well as the asymptotic shape of the compressor map SpLs close to the choke and surge areas, see Pachner et al. [22]. Due to this, parameterizing the model using a regular least-squares approach tends to either fail or produce unsatisfactory results. To remedy this and provide satisfactory parameterization results, a Total Least Squares (TLS) method is used. The TLS method implementation in the LiU CPgui toolbox is described in detail in this section.

5.1 Ellipse mass flow model parameterization

The parameterization process first focuses on the Ellipse mass flow model, in the $\Pi_c - \bar{W}_c$ plane. Figure 8 shows a representation of the model errors to be minimized when applying a regular least-squares approach with the corrected mass flow as independent variable. As can be seen in the figure, the errors close to the choke region become very large even if the measured points are not that far away from the modeled SpL, which numerically complicates the optimization problem. Note that if the errors are computed with the corrected mass flow value and using pressure ratio as independent variable, this issue will be located close to the Zero Slope line insead.

This issue, is addressed by minimizing instead the orthogonal distance between the modeled SpL and the measured points. This approach was shown to be a suitable way to deal with this problem in Llamas and Eriksson [17]. However, the orthogonal projections needed to solve a system of equations each iteration which slowed down the algorithm. In Llamas and Eriksson [13], a faster approach was developed to compute the orthogonal distance based on TLS, which is the algorithm implemented in LiU CPgui. More details about TLS can be found in Nocedal and Wright [23]. Figure 9 contains a diagram of the TLS algorithm applied to the Ellipse flow model for a single SpL. The main difference is that TLS allows deviations (δ) in the independent variable, in this case mass flow. Then the model errors

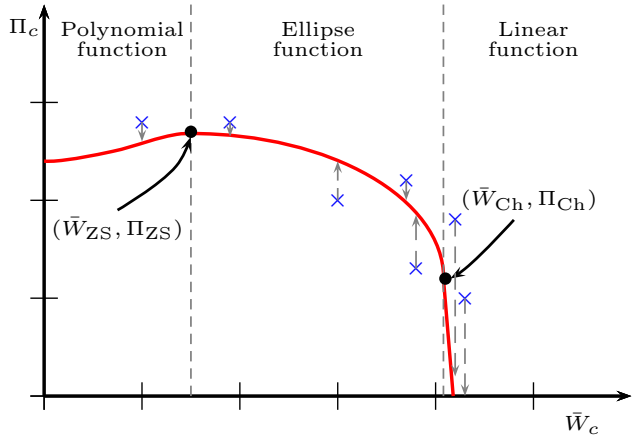

Figure 8: Regular least-squares errors of the Ellipse mass flow model. The modeled curve is seen in solid red. The blue crosses represent measurement points with the computed distance to the model as gray arrows.

are computed at the deviated measured point. In the TLS algorithm, the deviations (δ) are included in the parameter vector and also in the residual vector to be minimized. As can be seen in Figure 9, the model errors in gray are substantially reduced in the choke region.

The mathematical expression of the TLS algorithm applied to the Ellipse mass flow model is as follows

$$\min_{\theta,\delta} \sum_{k=1}^{m} (\epsilon_k^2 + \delta_k^2) \tag{34a}$$

s.t.

$$\epsilon_k = \Pi_{c,k} - f(\theta; \bar{W}_{c,k} + \delta_k), \quad \text{for } k = 1, 2, ..., m, \tag{34b}$$

$$\theta_l \leq \theta \leq \theta_u \tag{34c}$$

where m is the number of measured points and θ is the parameter vector

$$\theta = [C_{Wch}, C_{\Pi ch}, C_{Wzs}, C_{\Pi zs}, C_{cur}, C_s]^T \tag{35}$$

where each base function correspond to a set of parameters, e.g $C_{\Pi ch}$ has 3 parameters see (10b). Therefore, depending on the choice of base functions, the parameter vector is of different length. The parameter vector is restricted by fixed upper and lower limits, respectively θ_u and θ_l. The objective function is the sum of squared perpendicular distances, ϵ and δ, from the model to the measured points. Which geometrically correspond to the orthogonal distance between the SpL and measurements squared.

Since it is required to have a properly defined function for all map regions, the vertical line in the choke region is changed for a very steep linear function during

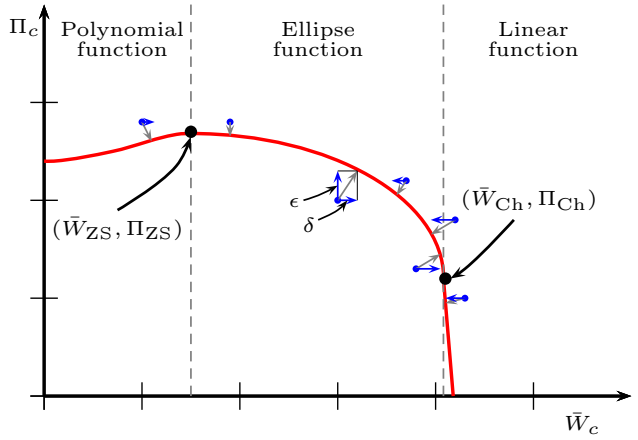

Figure 9: TLS algorithm with the deviations δ depicted as blue arrows. The blue dots represent measurement points and the red curve is the modeled SpL. The computed distance to the model is drawn as gray arrows.

the parameterization

$$\Pi_c = \frac{(1 + k_{lin})\Pi_{Ch}}{k_{lin}} - \frac{\Pi_{Ch}}{\bar{W}_{Ch} k_{lin}} \bar{W}_c \tag{36}$$

where k_{lin} is a parameter that defines the slope, here it is fixed to 0.01. Note also that the values of Π_{Ch} and $\bar{W}_{Ch}$ are given by the corresponding base functions and its dependency on the corrected compressor speed and the model parameters is omitted in the notation. The function f calculates the model pressure ratio and it is defined as a piecewise function

$$f(\theta; \bar{W}_c) = \begin{cases} \text{equation (16)} & \bar{W}_c < \bar{W}_{ZSL} \\ \text{equation (9)} & \bar{W}_{ZSL} \leq \bar{W}_c < \bar{W}_{Ch} \\ \text{equation (36)} & \bar{W}_c > \bar{W}_{Ch} \end{cases} \tag{37}$$

where its compressor speed dependency is omitted in the notation. In practice, the measured corrected compressor speed for the considered SpL is used as input in f.

5.2 Including the enthalpy-based efficiency model

Ideally, we would like to parameterize the enthalpy-based efficiency model alone. However, as discussed in Section 4.2, the efficiency model requires the flow prediction from the Ellipse mass flow model as input together with the pressure ratio to compute the efficiency values. This issue implies that, when used in real simulations, the errors of the Ellipse mass flow model will affect the efficiency predictions. Hence, this issue has to be properly addressed during the parameterization in order

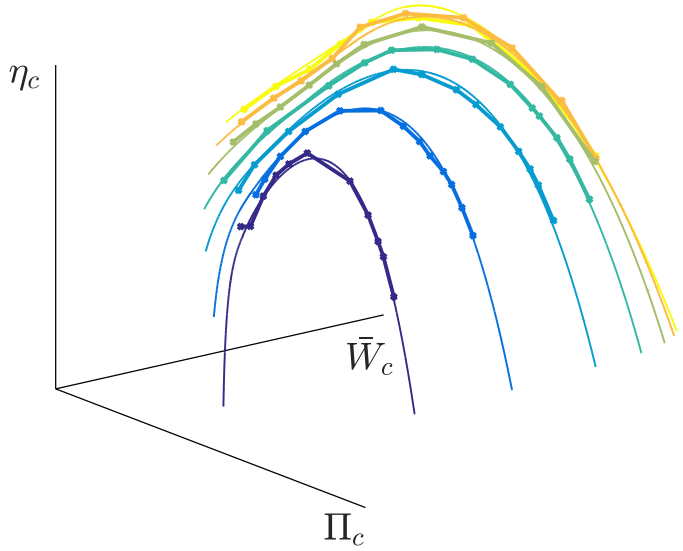

Figure 10: Three dimensional diagram of the modeled SpL in solid lines vs measured SpL in dot connected lines. Yellow corresponds to the highest speed and blue to the lowest.

to achieve the best model fit. This is done by solving a complete model parameterization of both mass flow and efficiency submodels at the same time to gave a good overall fit. Which in practice means minimizing the orthogonal distance between SpL and measured points in the three dimensional space defined by $(\bar{W}_c, \Pi_c, \eta_c)$. Figure 10 shows a sketch of the modeled and measured SpLs in the considered three dimensional space.

Mathematically, the formulation of the three dimensional case in TLS is very similar to the previous case (34). The difference is that the efficiency errors have to be computed composing the Ellipse mass flow model equation inside the Enthalpy-based efficiency model. By doing so, it results in

$$\min_{\tilde{\theta},\delta} \sum_{k=1}^{m} (\kappa_k^2 + \epsilon_k^2 + \delta_k^2) \tag{38a}$$

s.t.,

$$\kappa_k = \eta_{c,k} - g(\tilde{\theta}; \bar{W}_{c,k} + \delta_k, f(\tilde{\theta}; \bar{W}_{c,k} + \delta_k)), \tag{38b}$$

$$\epsilon_k = \Pi_{c,k} - f(\tilde{\theta}; \bar{W}_{c,k} + \delta_k), \tag{38c}$$

$$\text{for} \quad k = 1, 2, ..., m,$$

$$\tilde{\theta}_l \leq \tilde{\theta} \leq \tilde{\theta}_u \tag{38d}$$

where the parameter vector $\tilde{\theta}$ now contains all model parameters

$$\tilde{\theta} = [C_{Wch}, C_{\Pi ch}, C_{Wzs}, C_{\Pi zs}, C_{cur}, C_s, C_b, C_a, C_{loss}]^T \tag{39}$$

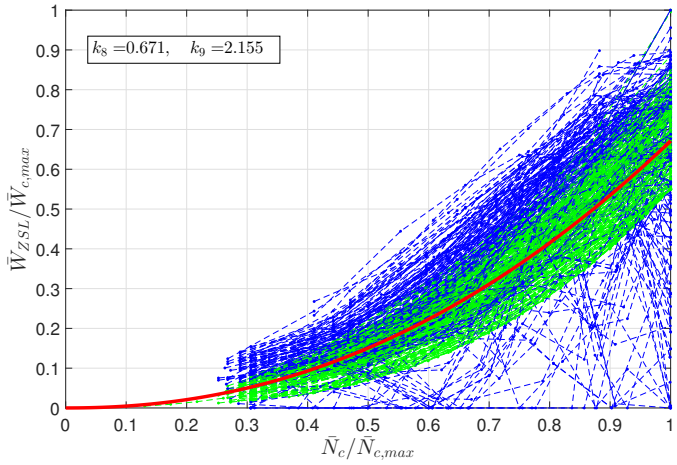

Figure 11: Zero slope line mass flow initialization results on 234 different compressor maps. Top left corner parameters correspond to the red curve fit. More information in the original publication Llamas and Eriksson [13].

The efficiency function g is defined by combining (20) (21) and (22) as follows

$$g(\tilde{\theta}; \bar{W}_c, \Pi_c) = \frac{c_p T_{01} \left[\Pi_c^{\frac{\gamma_c - 1}{\gamma_c}} - 1 \right]}{\Delta h_{act}} \tag{40}$$

The corrected compressor speed dependency is also omitted in the notation for simplicity. More details and results of the proposed algorithm can be found in Llamas and Eriksson [13].

5.3 Algorithm initialization

The initial parameters define the starting point of the nonlinear optimization problems defined in the previous section. To obtain good results, and even sometimes to be able to solve the problem, good initial values are essential. The automatic generation of suitable initial values is done in LiU CPgui by using the normalized compressor database results from Llamas and Eriksson [13]. An example of the initialization functions from [13] is shown in Figure 11 for the Zero Slope Mass flow base function. A separate set of initialization parameters is defined for large marine-size maps, since those have a slightly different shape as can be seen by comparing Figures 24 and 25. Table 1 contains the initialization values of the different base functions used in the toolbox. The distinction between marine-size or automotive-size values is done based on the maximum mass flow value measured in the compressor map.

In LiU CPgui, the model shape with this automatic initialization parameters can be compared to the current map and modified if necessary with a convenient graphical drag and drop method described in Section 6.5.

Table 1: Initialization values for the different base functions parameters.

Parameters (eq.)	Automotive-size	Marine-size
$C_{Wch,0}$ (10a)	$[0.795, 0.278, 2.491, 1.441]^T$	$[0.769, 0.354, 3.726, 2.929]^T$
$C_{Wch,0}$ (12)	$[0.491, 0.617, 1.274, 0.314, 0.804]^T$	$[0.336, 0.758, 2.371, 0.856, 0.896]^T$
$C_{\Pi ch,0}$	$[0.109, 0.387, 3.493]^T$	$[0.066, 0.641, 2.931]^T$
$C_{Wzs,0}$	$[0.671, 2.155]^T$	$[0.874, 2.149]^T$
$C_{\Pi zs,0}$	$[0.995, 2.274]^T$	$[1.020, 2.620]^T$
$C_{cur,0}$	$[2.092, 0.984, 5.001]^T$	$[2.453, 1.887, 2.421]^T$
$C_{s,0}$	1	1
$C_{b,0}$ (23a)	$[1.022, 0.0979]^T$	$[0.988, 0.086]^T$
$C_{b,0}$ (25)	$[0.091, 1.249]^T$	$[0, 1.382]^T$
$C_{a,0}$ (23b)	$[0.403, 0.0177, 2.568]^T$	$[0.311, 0.071, 5.209]^T$
$C_{a,0}$ (26)	$[0.757, 0.216]^T$	$[0.635, 0.216]^T$
$C_{loss,0}$	0.0114	0.0161

6 How to work with LiU CPgui

This section provides a complete tutorial about the different steps to follow to parameterize a given compressor map using LiU CPgui. Every step is described in detail using a compressor map parameterization example.

Figure 12 contains a flow diagram of the different steps to be followed from inputing a compressor map to getting the model parameters. Four main parameterization steps have to be performed, which can be repeated if the results are not satisfactory. Figure 12 represents this repetition after the blocks "is shape ok?" and "model fit ok?" going back to the previous parameterization block. Note that in the toolbox, it is possible to repeat any previous parameterization step, not only the latest executed, which is represented in Figure 12 by a dashed line. When repeating a parameterization step, the initial parameters to be used and the parameter constraints can be chosen from the available ones to modify the result. The names of the steps coincide with the titles used in the different button groups of the LiU CPgui, see Figure 13.

6.1 Software requirements

LiU CPgui requires Matlab 2013b or newer. If **`lsqnonlin`** is the chosen least-squares solver, the Matlab Optimization Toolbox is required, otherwise a basic Matlab installation is sufficient. When LiU CPgui is started, the toolbox checks automatically whether or not the Optimization Toolbox is installed and enables or disables the **`lsqnonlin`** solver appropriately.

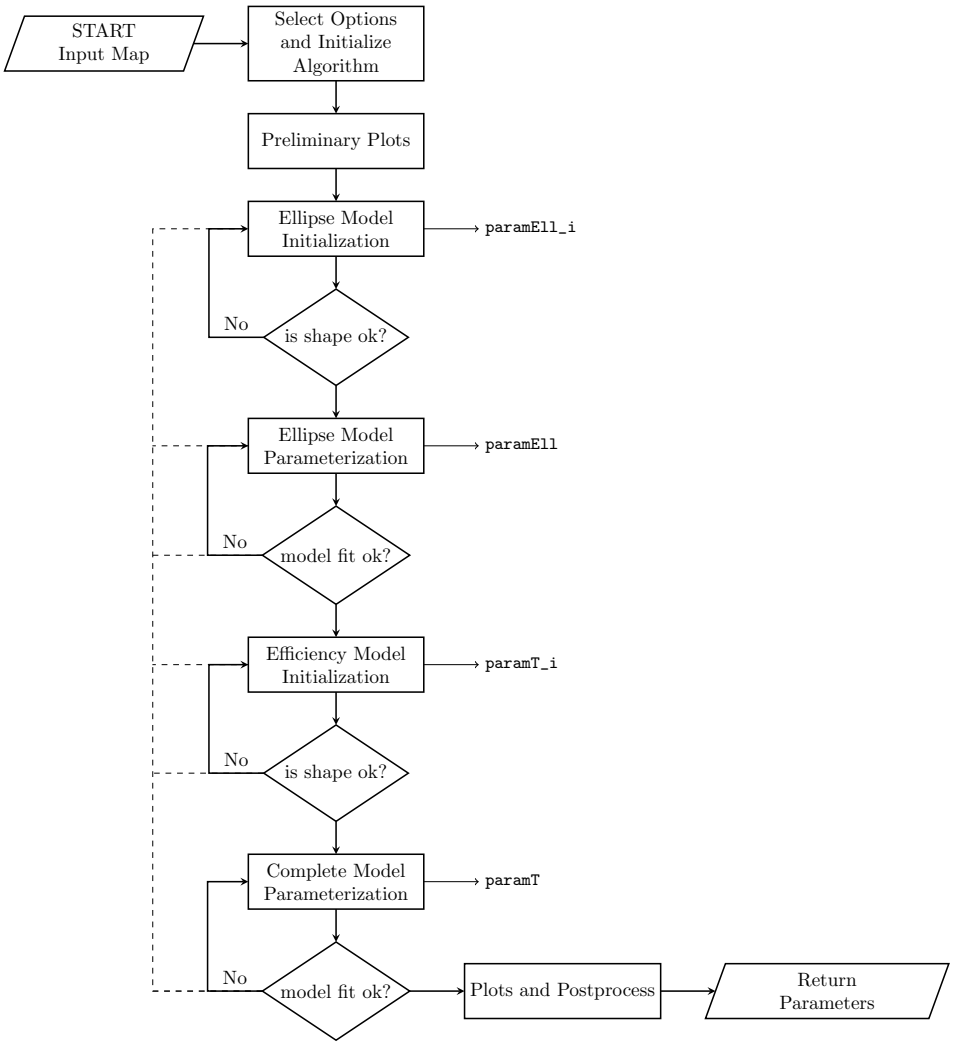

Figure 12: Flow chart of the parameterization steps with LiU CPgui. The four parameterization steps contain the name of the parameter variable created/updated when executed. The dashed line represents the possibility to go back and repeat any previous parameterization step if the results are not satisfactory.

6.2 Starting LiU CPgui with a compressor map

First of all, LiU CPgui needs to be initialized. This is done by calling the function **`init_LiU_CPgui`** from the Matlab prompt, which adds the necessary folders to the Matlab path.

The required compressor map measurements and its format are listed in Table 2. The impeller diameter is optional and it does not need to be specified if the user does not have the value available. If the diameter is unknown, the numerical rule depicted in Figure 8 from [18] is used to get an estimate of the compressor diameter from the maximum speed measured in the map.

Table 2: Required compressor map data inputs to LiU CPgui.

Variable	Units	Description	Type
`NcCorr`	[rpm]	Corrected compressor speed	Column vector
`WcCorr`	[kg/s]	Corrected compressor mass flow	Column vector
`PiC`	[-]	Compressor pressure ratio	Column vector
`etaC`	[-]	Compressor efficiency	Column vector
`TCref`	[K]	Compressor map reference temperature	Scalar
`pCref`	[Pa]	Compressor map reference pressure	Scalar
`D_2`	[m]	Compressor outer impeller diameter	Scalar

Once the required data is available, the compressor map can be structured in the required format using the provided function **`get_mapStruct`**. This function requires the map variables as inputs and returns the compressor map in a Matlab structure array. Running this command in the matlab prompt results in the following structure array with a field for each map variable:

```
map = get_mapStruct(NcCorr,WcCorr,PiC,etaC,TCref,pCref,D_2)
map =

struct with fields:

NcCorr: [25x1 double]
WcCorr: [25x1 double]
PiC: [25x1 double]
etaC: [25x1 double]
TCref: 298
pCref: 100000
D_2: 0.0300
```

Once the map is in the required format, the GUI can be started by running the command **`LiU_CPgui(map)`**.

The file **`example_LiU_CPgui.m`** contains an example call of the GUI with a representative but fictive compressor map (note that the provided map is not a real

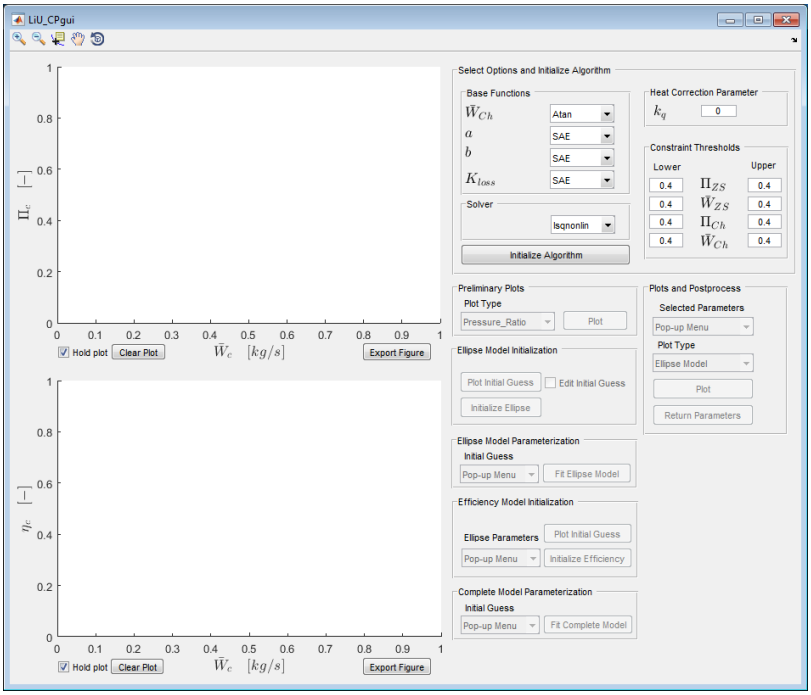

Figure 13: LiU CPgui after being loaded with most of the buttons disabled initially.

measured compressor map). A step-by-step parameterization process of the representative compressor map is described in the following sections, the user can later use this file as a template to load any desired compressor map. Start by running the **`example_LiU_CPgui.m`** command in the Matlab prompt. This will set the required folders to the path and load the GUI with the fabricated compressor map. Figure 13 shows the GUI after being loaded with most of the buttons disabled at the step displayed.

6.3 Select options and initialize algorithm

As is in Figure 13, the top right part of the LiU CPgui is where all the parameterization options are selected before initializing the algorithm. The following subsections provide a description of each of them. Once all options are selected, the lower left button, Initialize Algorithm, has to be pressed so all the required variables are created. Once the Algorithm is initialized, more GUI buttons are enabled. To follow the **`example_LiU_CPgui.m`** parameterization steps, taken here, keep the default options and press Initialize Algorithm.

6.3.1 Base functions

Four of the model base functions have to be chosen from two different alternatives per base function. These options are listed in Table 3.

Table 3: Base function options.

Base Function	Option 1	Option 2
$\bar{W}_{Ch}$	Atan: defined as (10a)	Switch: defined as (12)
a	SAE: defined as (23b)	ASME: defined as (26)
b	SAE: defined as (23a)	ASME: defined as (25)
K_{loss}	SAE: defined as (24)	ASME: defined as (27)

The base function selection is performed using the pop-up menus. When the button Initialize Algorithm is pressed, the selected base functions are loaded for the parameterization steps. If the user decides to change the base functions after the initialization is performed, the LiU CPgui buttons are disabled until the button Initialize Algorithm is pressed again.

6.3.2 Solver

Two solvers can be selected with the corresponding pop-up menu; **`lsqnonlin`** and **`LSOptim`**. The default solver is **`lsqnonlin`**, which is a least-squares solver provided with the Matlab Optimization Toolbox. Thus, the user needs to have this toolbox available if the solver **`lsqnonlin`** is selected. On the other hand, the alternative solver **`LSOptim`** is provided in case the user does not have access to the Optimization Toolbox. **`LSOptim`** is a custom built least-squares solver by Professor Lars Eriksson. More information about it can be obtained by contacting the author.

6.3.3 Heat correction parameter

The heat correction method implemented in LiU CPgui is described in Section 3. In the top right corner of the LiU CPgui, a numeric value can be specified for the correction parameter k_q, by default this value is zero. After Initialize Algorithm button is pressed, the equation (8) is calculated again. The authors suggest to start with a small number, e.g. a value around $k_q \approx 0.002$. The general criterion to know if the heat correction is suitable is to check that all SpLs collapse on top of each other in a work coefficient vs flow coefficient plot. This plot can be performed by selecting the work coefficient option in the preliminary plots menu. This process can be repeated until a suitable k_q value is found.

6.3.4 Constraint thresholds

The constraints of the different base functions can be modified by changing the value box corresponding to the upper and the lower bound constraints for each of the base function parameters. In this case, a value of $x_{thr} = 0.1$ corresponds to a relative 10% margin (upper or lower) of the current parameter value. The

parameter threshold limits can be written, for upper and lower case as

$$\theta_{lim} = \theta_o \pm |\theta_o \cdot x_{thr}| \tag{41}$$

where θ_o is the initial vector of parameters. Hence, increasing x_{thr} will result in wider bound constraints, so more space to adjust the shape of the base functions. On the other hand, decreasing x_{thr} will make sure that the solver has little room to modify the current base function parameters. The thresholds can be modified at any time before the parameterization, it is not necessary to press the Initialize Algorithm button for the changes to take place.

6.4 Preliminary Plots

Once the initialization has been done, the Preliminary Plots menu is enabled. This menu allows the user to do four different plots of the current compressor map. By selecting the type of plot in the pop-up menu and then pressing the Plot button, the desired figure is displayed at one of the two GUI axes. Figure 14 shows the pressure ratio and efficiency plots as function of mass flow, while Figure 15 shows the other two available plots, Actual Enthalpy and Work Coefficient. The Legend corresponds to the blade Mach number M_{U_2}. Under the GUI axes, there is a check box that enables or disables the hold plot option together with a Clear Plot button. At the opposite side under the axes there is an Export Figure button, its function is to export the current plot from the corresponding axes to a new figure outside the GUI.

6.5 Ellipse model initialization

Starting with a good initial guess for the Ellipse model is crucial in order to obtain good results. Hence, a point and click with drag and place function is implemented to modify the default initial guess is implemented in the package to help the user define a suitable initial guess for the given compressor map. Pressing the Plot Initial Guess button shows what will be send to the solver as initial guess, see Figure 16. The check box Edit Initial Guess lets the user drag and drop the different points in the Choke and Zero Slope lines, see Figure 17, where the red color indicates that the lines are currently modifiable. The drag and drop functionality works with the following sequence: mouse click on the desired point and hold the mouse button pressed, move the cursor to the desired position and release the button, the point is moved accordingly. Once the Zero Slope and Choke lines are at the desired positions, uncheck the checkbox and the lines become black again, see Figure 18.

Now that the Ellipse model initial guess is suitable, the model is initialized by pressing Initialize Ellipse. Essentially what the algorithm does is to solve separated least squares problems for each compressor speed line, and then use the results to initialize the base function parameters. This will create a Matlab struct

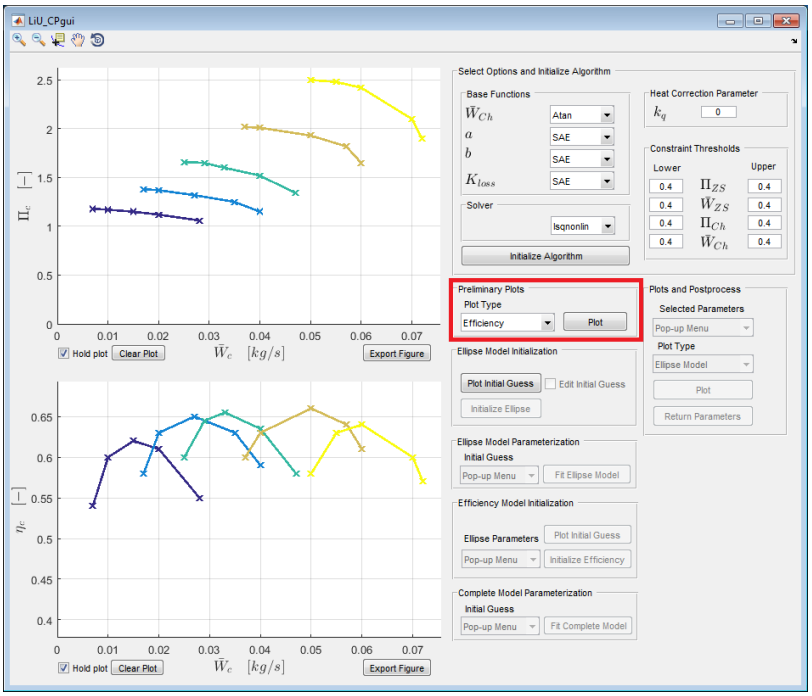

Figure 14: LiU CPgui after plotting Pressure ratio and efficiency. The corresponding panel is highlighted in red.

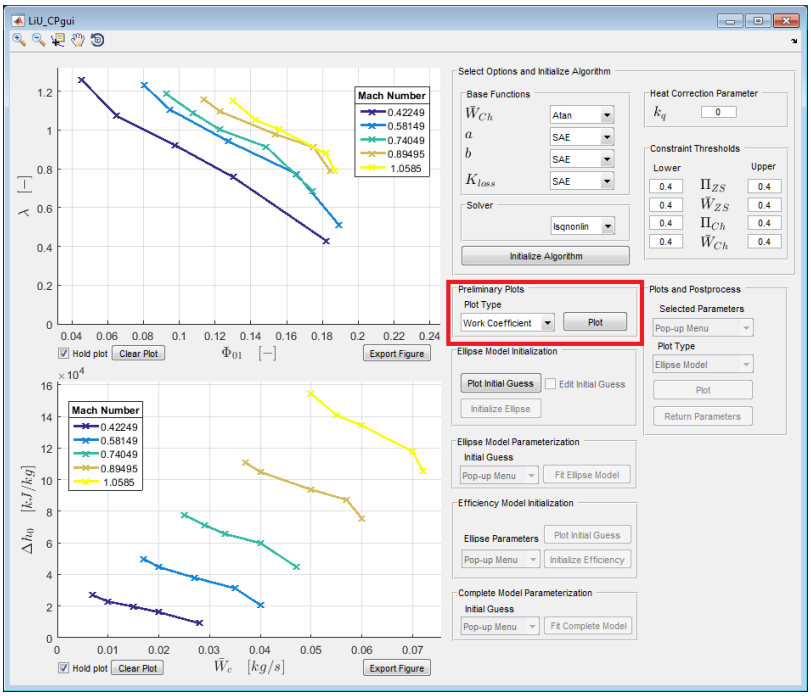

Figure 15: LiU CPgui after plotting Enthalpy and Work Coefficient. The corresponding panel is highlighted in red

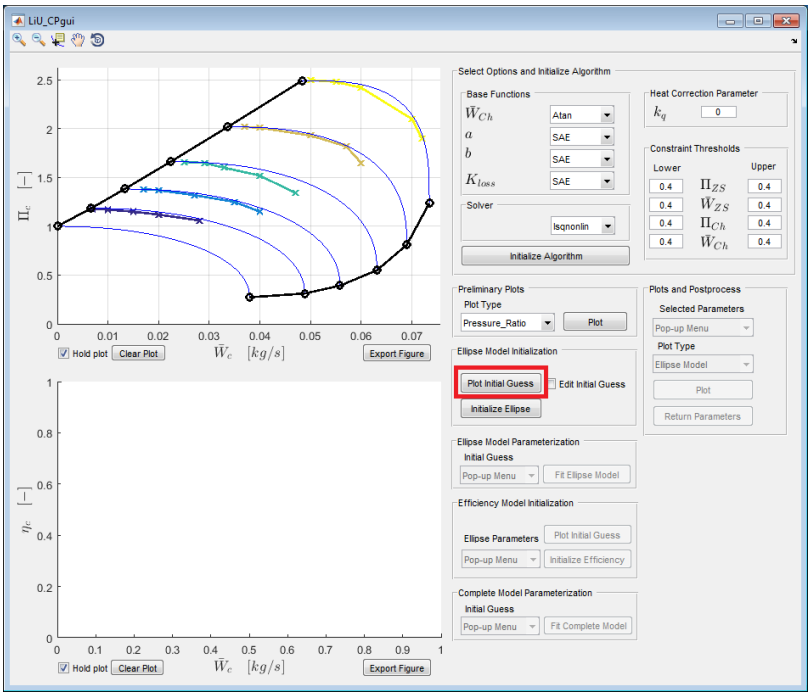

Figure 16: LiU CPgui with the default Ellipse model initial guess. The corresponding button is highlighted in red.

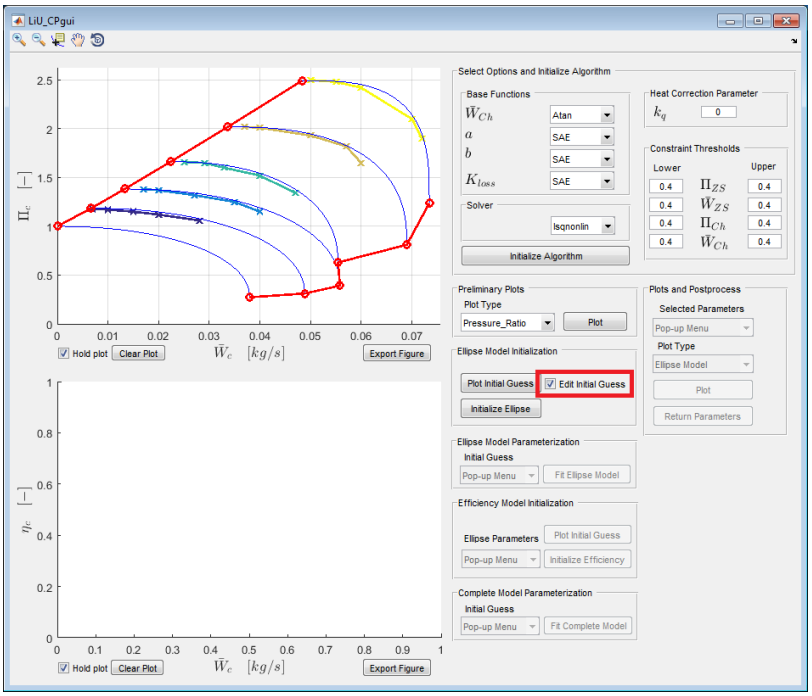

Figure 17: LiU CPgui with the Ellipse model initial guess being edited in red. The corresponding button is highlighted in red.

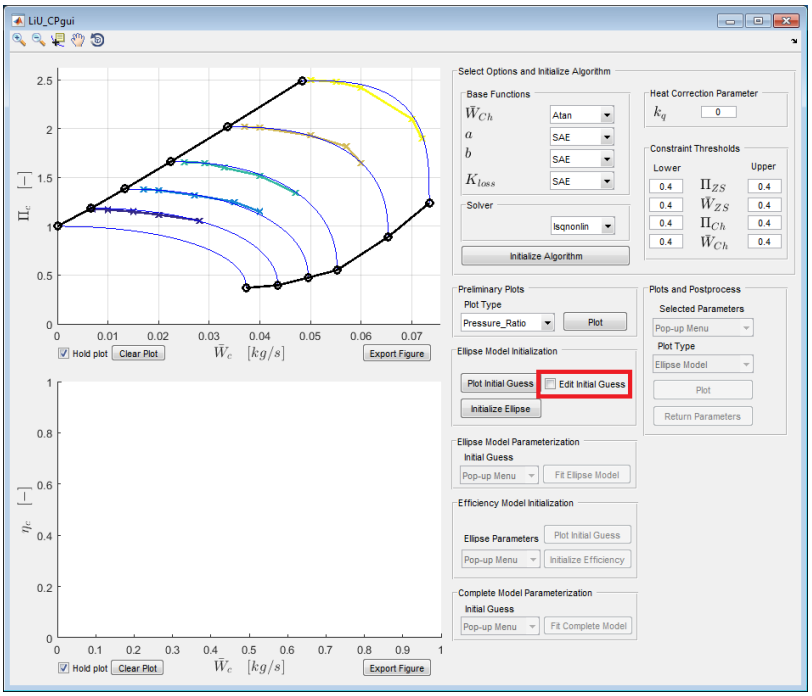

Figure 18: LiU CPgui with the Ellipse model initial guess after the drag and drop process. The corresponding button is highlighted in red.

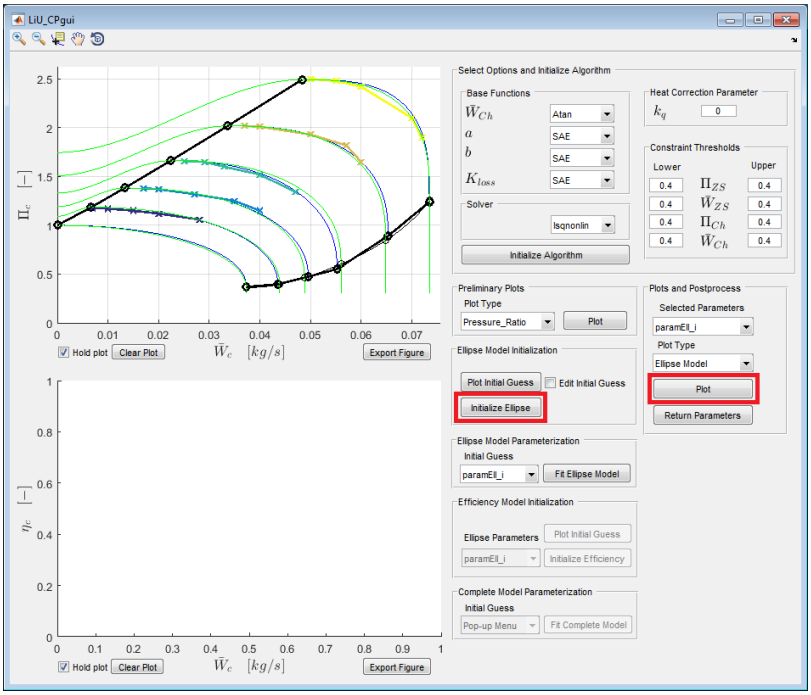

Figure 19: LiU CPgui with the initialized Ellipse model plot in green. Note that the plots are being added on top of each other because the Hold plot option is enabled. The corresponding buttons are highlighted in red.

variable named **`paramEll_i`** that contains the model parameters. **`paramEll_i`** can be selected in the Plots and Postprocess>Selected Parameters pop-up menu in order to plot the ellipse model in green on top of the compressor map, see Figure 19 to see how it looks like in the GUI. If the results are not satisfactory, the initial guess can be further edited with the drag and drop functionality and the constraint thresholds can be adjusted accordingly.

6.6 Ellipse model parameterization

After initializing the Ellipse model and obtaining a suitable model shape, the Ellpise Model Parameterization block in the LiU CPgui is enabled. In the Initial Guess pop-up menu the initial guess parameters can be selected (currently only **`paramEll_i`** are available). Pressing the button Fit Ellipse Model will adjust the Ellipse model parameters to best represent the given compressor map in the Mass flow - Pressure Ratio plane. The resulting parameters are stored in a struct named **`paramEll`**, that becomes available in the pop-up menus for plotting in red or for being selected as initial guess for other parameterization steps. Figure 20 shows the LiU CPgui after plotting the Ellipse model on top of the previous parameterization. Note also that the parameterization process writes the solver iterations in the

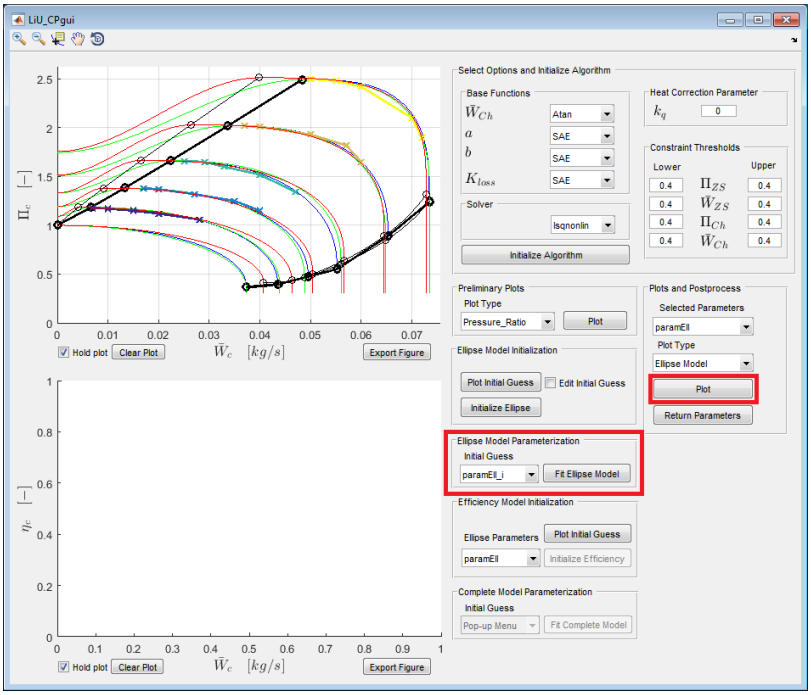

Figure 20: LiU CPgui with the parameterized Ellipse model in red. Note that the plots are being added on top of each other because the Hold plot option is enabled. The corresponding buttons are highlighted in red.

Matlab prompt together with the calculated model relative errors in percentage.

If the results are not satisfactory, this parameterization step can be repeated changing the parameter threshold constraints. Furthermore, the initial guess can be changed using the pop-up menu, the initialization parameters **`paramEll_i`** can be used again, or the results from the current parameterization step, **`paramEll`**, can be chosen. Remember that the parameter struct variables are updated every time the corresponding parameterization button is pressed, see Figure 12. The parameter struct variables can be exported to the workspace using the button Return Parameters in case the user wants to save them before they are updated in a parameterization step. More details about exporting the parameters are given in Section 6.9.1.

6.7 Efficiency model initialization

Once **`paramEll`** becomes available, the block Efficiency Model Initialization is enabled. The first thing to do is to plot the default initial guess, shown in Figure 21. Once this is done, the option to initialize the Efficiency model becomes available. By default the previous Ellipse parameters **`paramEll`** are selected to initialize the Efficiency parameters. Note that there is a link between the efficiency and the mass flow submodels, as described in Sections 4 and 5. This initialization process creates

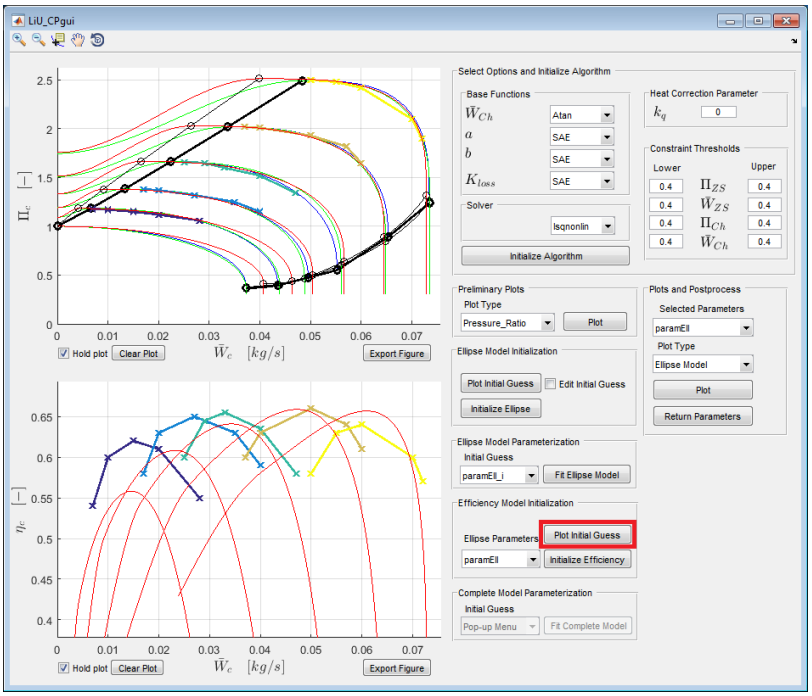

Figure 21: LiU CPgui with the Efficiency model initial guess plotted in red. The corresponding button is highlighted in red.

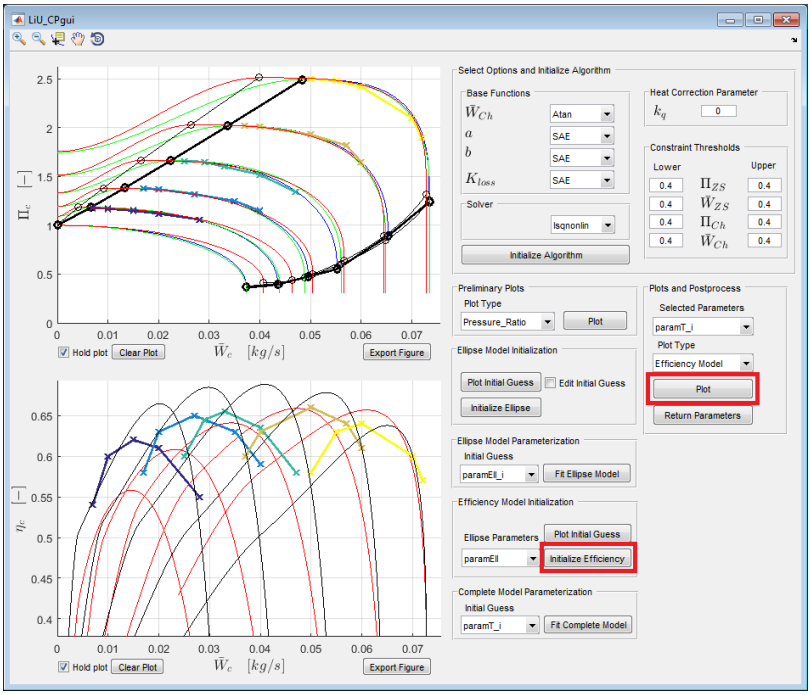

Figure 22: LiU CPgui with the initialized Efficiency model plotted in black. Note that the plots are being added on top of each other because the Hold plot option is enabled. The corresponding buttons are highlighted in red.

the parameter struct **`paramT_i`** that becomes available for plots in the Plots and Postprocess block of the GUI. For this example, the plotted initialized efficiency model is shown in Figure 22, plotted using black color. As can be seen in the figure, the efficiency curves are not very close together to the measured values, nevertheless, this issue is not very problematic because the complete model will be parameterized in the next step. However, if the results are too far away from the measured values, this step and the previous ones can be repeated until the efficiency model initialization produces a model that is close to the measured values.

6.8 Complete model parameterization

With the parameters **`paramT_i`**, the complete parameterization of the ellipse flow and the efficiency models can be executed. By pressing Fit Complete Model, the complete model parameters, named **`paramT`**, are created. Figure 23 shows the model results after the complete parameterization in the three dimensional space Mass Flow - Pressure Ratio - Efficiency. For the **`paramT`** parameters, the plotting color is blue. The solver iterations are written in the Matlab prompt together with the calculated model relative errors in percentage for the three compressor map dimensions.

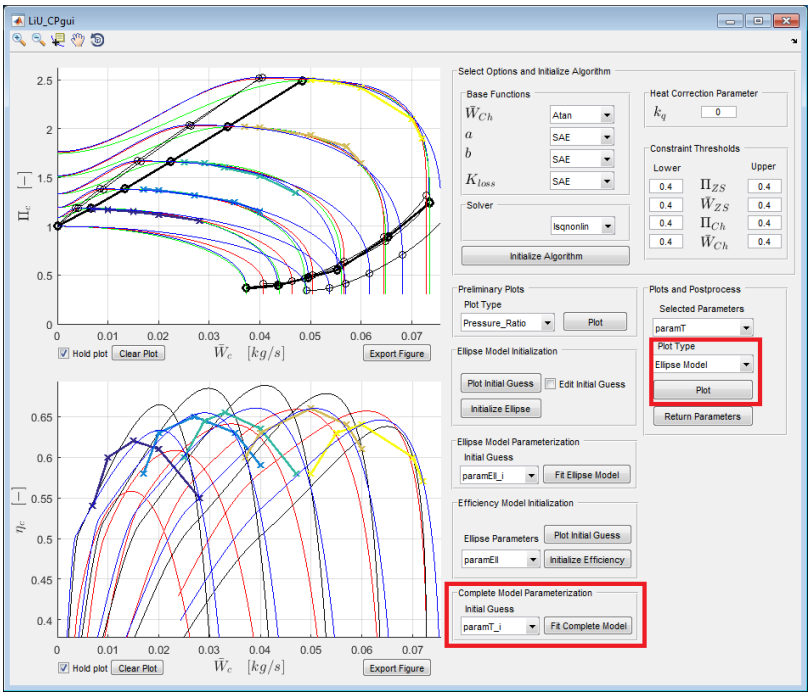

Figure 23: LiU CPgui with the final model parameterization plotted in blue. Note that the plots are being added on top of each other because the Hold plot option is active. The corresponding buttons are highlighted in red.

If the results are not satisfactory, this step is repeated using for example the current final parameters **paramT** and changing the model parameter Constraint Thresholds in the menu.

6.9 Plots and postprocess

The Plots and Postprocess menu works in a similar way as the Preliminary Plots menu. Once the Selected Parameters and the Plot Type are chosen from the Pop-up Menus, pressing the Plot button will execute the plot combination. Note that those buttons become enabled once the algorithm has some model parameters available. The different plotting options are summarized in Table 4.

Table 4: Different plot type options available.

Plot Type	Description
Ellipse Model	Ellipse mass flow model in the top axes.
Efficiency Model	Efficiency model in the bottom axes.
Base Functions	Base functions in a different figure.
Eff. Iso Lines	Ellipse model together with the model efficiency isovalue lines in the top axes.
3D plot	Model in the mass flow-pressure ratio-efficiency space in the top axes. Use the rotate option in the top left bar of LiU CPgui to rotate the obtained figure.
Enthalpy gain	Modeled actual enthalpy vs mass flow in the bottom axes.
Work Coefficient	Modeled work vs inlet flow coefficients in the top axes.

6.9.1 Return Parameters

Pressing the button Return Parameters in the Plots and Postprocess menu sends the current model parameters chosen in the Selected Parameters Pop-up menu to the Matlab workspace. Once this is executed, a struct with the same name is created to be used outside of the LiU CPgui environment. The parameters can then be used together with the provided model implementations in Matlab code and Simulink. The **paramT** struct contains the fields described in Table 5

6.10 Model implementations

An implementation of the compressor model in Matlab code is provided and named **CompressorModel.m**. The implementation follows the forward formulation from (28). Two different Simulink implementations of the compressor model are provided in the Simulink library **LiU_Compressor_Model_Lib.slx**. A forward implementation following the formulation in (28) is provided and named **Forward_CompressorModel**. A backward implementation capable to simulate

Table 5: **paramT** struct field names and description

Field name	Description
vector	Column vector with the model parameters; θ or $\tilde{\theta}$.
delta_Wc	Column vector with the δ deviations, see (34).
C	Vectors with the corresponding model parameters C (Using same subscripts as this text).
F_X	Names of the base functions used in the estimation.
T_ref	Compressor map reference temperature.
p_ref	Compressor map reference pressure.
Nc_max_map	Compressor map maximum corrected speed.
Wc_max_map	Compressor map maximum mass flow.
PI_max_map	Compressor map maximum pressure ratio.
Delta_h_max	Compressor map maximum actual enthalpy gain.
gamma_air	Ratio of specific heats of the compressed air.
rho1	Density at the compressor inlet.
D_2	Compressor outer impeller diameter.
Cp_air	Specific heat at constant pressure of the compressed air.
Gamma_PIcs	Surge parameter $\Gamma_{\Pi_{cs}}$ from (18).
alpha_kPiW0	Linear function parameter $\alpha_{k\Pi W0}$ from (29).

surge phenomena (32) is provided in the library and named **Surge_CompressorModel** . Note that for the surge model some extra parameters need to be specified; Model plug length L and diameter D, and the compressor reverse operation parameters K_0 and K_t for which the values from Section 4 can be used.

The Simulink library also contains implementations of the different compressor model base functions. In case the user wants to change the base functions when parameterizing the model in the GUI, later the Simulink implementation can be modified accordingly. The Simulink implementations require that the parameter struct variable obtained from LiU CPgui is renamed as **ParamComp**, the structure field names are the same. Moreover, the Simulink library is saved in different Matlab versions to minimize compatibility problems.

7 Parameterization examples

This section provides examples of the model parameterization to two very different types of compressors. Figure 24 contains the results for a standard small automotive-size vaneless compressor. As can be seen the agreement of the model and the measured values is very good for both mass flow and efficiency values.

The proposed model and parameterization method is able to deal with different

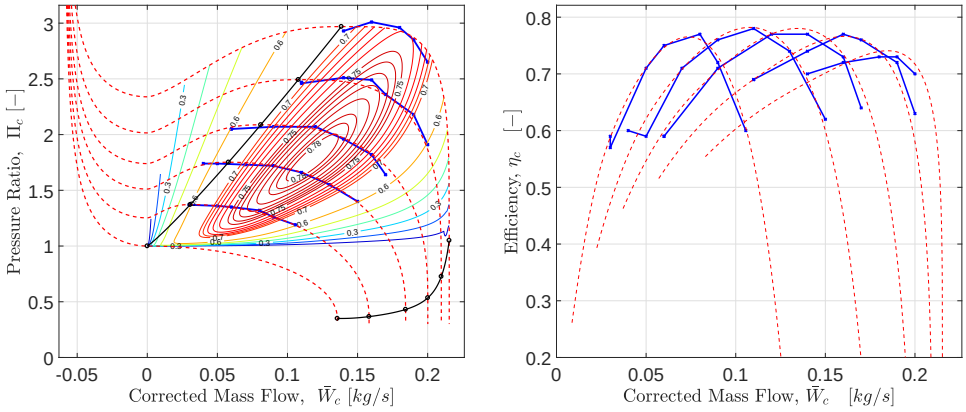

Figure 24: Parameterized automotive compressor map. Measured SpL drawn as solid blue lines. Left: Pressure ratio vs corrected mass flow with efficiency as contour lines. Right: Efficiency vs corrected mass flow. Note that the first two speed lines efficiency values are higher than the following measured point due to measurement errors.

compressor sizes, Figure 25 shows the parameterization results on a big marine vaned compressor map. As can be observed, the shape of the compressor map is significantly different compared to the automotive-size compressor from Figure 24. Despite the differences, the model is also able to adapt to the measured map using the toolbox parameterization method.

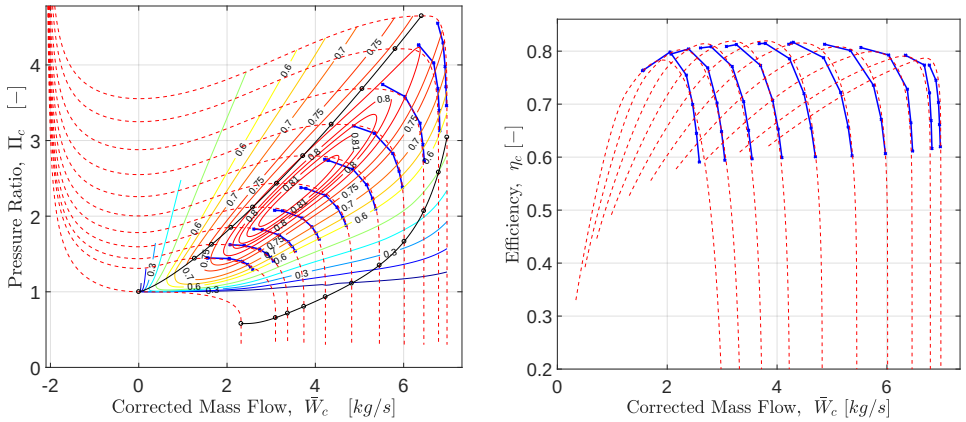

Figure 25: Parameterized marine compressor map. Measured SpL drawn as solid blue lines. Left: Pressure ratio vs corrected mass flow with efficiency as contour lines. Right: Efficiency vs corrected mass flow.

8 Summary

The LiU CPgui toolbox functionality has been described together with all mathematical equations of the compressor model that is parameterized with the toolbox. How to work with the toolbox has been explained in detail by following a step-by-step compressor map parameterization procedure.

Acknowledgments

This work was supported by Vinnova's Industry Excellence Center LINK-SIC. The authors would like to thank Viktor Leek for proofreading this report and for providing valuable comments to improve the text and the LiU CPgui toolbox. Kristoffer Ekberg is also acknowledged for testing the functionality of an earlier version of the parameterization toolbox and for providing useful feedback.

Bibliography

[1] SAE J1826 – Turbocharger Gas Stand Test Code. SAE standard, 1995.

[2] SAE J922 – Turbocharger Nomenclature and Terminology. SAE standard, 1995.

[3] Sydney Lawrence Dixon and Cesare A Hall. *Fluid Mechanics and Thermodynamics of Turbomachinery*. Butterworth-Heinemann, Oxford, UK, 7th edition, 2013. ISBN 9780124159549.

[4] Silvia Marelli, Giulio Marmorato, Massimo Capobianco, and Andrea Rinaldi. Heat transfer effects on performance map of a turbocharger compressor for automotive application. In *SAE Technical Paper 2015-01-1287*, 04 2015. doi: 10.4271/2015-01-1287.

[5] Karl D. Wygant Nick Baines and Antonis Dris. The analysis of heat transfer in automotive turbochargers. *ASME J Gas Turb Pwr*, 4(132), January 2010. doi: 10.1115/1.3204586.

[6] M. Cormerais, J. F. Hetet, P. Chesse, and A. Maiboom. Heat transfer analysis in a turbocharger compressor: Modeling and experiments. In *SAE Technical Paper*. SAE International, 04 2006. doi: 10.4271/2006-01-0023.

[7] S. Shaaban and J.R. Seume. Analysis of turbocharger non-adiabatic performance. In *8th International Conference on Turbochargers and Turbocharging*, pages 119 – 130. Woodhead Publishing, 2006. doi: 10.1016/B978-1-84569-174-5.50012-9.

[8] Jose Serrano, Pablo Olmeda, Francisco Arnau, Miguel Reyes-Belmonte, and Alain Lefebvre. Importance of heat transfer phenomena in small turbochargers for passenger car applications. *SAE Int. J. Engines*, 6:716–728, 04 2013. doi: 10.4271/2013-01-0576.

[9] Michael V. Casey and Thomas M. Fesich. The efficiency of turbocharger compressors with diabatic flows. *ASME J Gas Turb Pwr*, 7(132), April 2010.

[10] Borislav Sirakov and Michael Casey. Evaluation of heat transfer effects on turbocharger performance. In *Proceedings of ASME Turbo Expo*, Vancouver, BC, Canada, June 2011. ASME.

[11] Michael Casey and Chris Robinson. A method to estimate the performance map of a centrifugal compressor stage. *ASME J Turbomach*, 2(135), 2012. doi: 10.1115/1.4006590.

[12] Mehdi Nakhjiri, Peter Pelz, Berthold Matyschok, Lorenz Däubler, and Andreas Horn. Apparent and real efficiency of turbochargers under influence of heat flow. In *14th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery (ISROMAC-14)*, Honolulu, HI, February 2012.

[13] Xavier Llamas and Lars Eriksson. Control-oriented compressor model with adiabatic efficiency extrapolation. *SAE International Journal of Engines*, 10 (4), 2017. doi: 10.4271/2017-01-1032.

[14] M.V. Casey and M. Schlegel. Estimation of the performance of turbocharger compressors at extremely low pressure ratios. *Proc Inst Mech Eng A: Journal of Power and Energy*, 2(224):239–250, November 2009. doi: 10.1243/09576509JPE810.

[15] Oskar Leufvén and Lars Eriksson. A surge and choke capable compressor flow model - validation and extrapolation capability. *Control Eng. Pract.*, 21 (12):1871–1883, December 2013. doi: 10.1016/j.conengprac.2013.07.005.

[16] Oskar Leufvén and Lars Eriksson. Measurement, analysis and modeling of centrifugal compressor flow for low pressure ratios. *Int. J. Engine Res.*, 17 (2):153–168, February 2016.

[17] Xavier Llamas and Lars Eriksson. Parameterizing compact and extensible compressor models using orthogonal distance minimization. *ASME J Gas Turb Pwr*, 139(1), 2017. doi: 10.1115/1.4034152.

[18] Lars Eriksson, Vaheed Nezhadali, and Conny Andersson. Compressor flow extrapolation and library design for the modelica vehicle propulsion library - vehprolib. In *SAE Technical Paper 2016-01-1037*. SAE International, April 2016. doi: 10.4271/2016-01-1037.

[19] Lars Eriksson and Lars Nielsen. *Modeling and Control of Engines and Drivelines*. John Wiley & Sons, Hoboken, NJ, 2014.

[20] Lars Eriksson, Xavier Llamas, Kristoffer Ekberg, and Viktor Leek. *Dynamic Modeling, Simulation and Control of Turbochargers*, pages 176–206. Nova Science Publishers, Inc., 2017.

[21] E. M. Greitzer. The stability of pumping systems - the 1980 Freeman scholar lecture. *Journal of Fluids Engineering*, 103:193–242, 1981.

[22] Daniel Pachner, Lukas Lansky, David Germann, and Markus Eigenmann. Fitting turbocharger maps with multidimensional rational functions. In

SAE Technical Paper 2015-01-1719. SAE International, April 2015. doi: 10.4271/2015-01-1719.

[23] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

A License

```
Version 1.2: 2017-12-20
Copyright (C) 2017, Xavier Llamas

LiU CPgui is free software: you can redistribute it and/or
    modify it under the terms of the GNU Lesser General
    Public License as published by the Free Software
    Foundation, version 3 of the License.

This package is distributed in the hope that it will be
    useful, but WITHOUT ANY WARRANTY; without even the
    implied warranty of MERCHANTABILITY or FITNESS FOR A
    PARTICULAR PURPOSE. See the GNU Lesser General Public
    License for more details.

You should have received a copy of the GNU Lesser General
    Public License along with LiU CPgui. If not, see <http
    ://www.gnu.org/licenses/>.
```

B List of files in LiU CPgui

```
============================================================
Package Contents
============================================================
init_LiU_CPgui - Initializes the compressor parameterization
     package and sets the required path.
example_LiU_CPgui - Example call of the CP GUI with a
    example compressor map, created to resemble a measured
    one, the user is supposed to load the compressor map in
    the provided fields.
ExampleMap - mat file which contains the example compressor
    map (not measured from any real compressor).
LiU_CPgui - Compressor parameterization Graphical User
    Interface.


Simulink Implementations
============================================================
CompressorModel - m-function implementation of the
    compressor model in the forward implementation.
CompModel - Simulink implementation of the compressor model
    in usual forward mode.
CompModel_Surge - Simulink implementation of the compressor
    model for surge simulation in the More-Greitzer
```

```
    framework.
BaseFunc_Blocks - Simulink library of the different model
    base functions, implemented to be used in case the user
    selects different base functions in the LiU CPgui tool.


Subfunctions used inside the GUI
=========================================================
lsoptim - Solves an uncostrained non-linear least squares
    problem using a Levenberg-Marquardt like method.
initialize_algorithm - Creates all initial signals required
    for the parameterization process depending on the user
    inputs
get_signals - Creates all initial signals required for the
    given compressor map, in thegiven map struct.
get_vector - Returns a vector of the mean of each column in
    the Matrix M
get_mapStruct - Creates the map structure required by the
    parameterization GUI as input.
Ellipse_Initialization - Returns the Ellipse parameters
    given the Initial guess, the Constraints and solver
    options.
Ellipse_Parameterization - Solves a Total Least Squares
    problem to optimize the Ellipse parameters given the
    Initial guess, the Constraintsand solver options.
Efficiency_Initialization - Provides an initial guess of the
     efficiency parameters by fitting each SpL independently
     and then parameterizing the base functions.
Complete_Parameterization - Solves a Total Least Squares
    problem to optimize the complete compressor model
    parameters given the Initial guess, the Constraintsand
    solver options.


Plotting Subfunctions
=========================================================
plot_modelMassFlow - Plots the model pressure ratio vs mass
    flow in the given axes handle.
plot_modelEfficiency - Plots the model efficiency vs mass
    flow in the given axes handle.
plot_modelContour - Plots the model pressure ratio vs mass
    flow in the given axes handle with the efficiency
    extrapolation as contour levels.
plot_model3D - Plots a number of speed lines in the 3D space
     formed by mass flow, pressure ratio and efficiency.
```

plot_lambda - Plots the work coefficient vs flow coefficient
or the enthalpy vs mass flow for the given map and
parameter sets
plot_baseFunctions - Plots each model base function vs
corrected speed.
compute_grid - Creates the compressor mass flow grid area
given the model parameters and the options.
compute_PiC - Computes the pressure ratio given the gridded
compressor mass flow area and the model parameters.
compute_etaC - Computes the efficiency given the gridded
compressor mass flow and pressure ratio areas and the
model parameters.
compute_basefunctions - Computes the base functions in a
dense grid given the model parameters and the gridding
options.

Base functions
==========
F_WZSL - Computes the Zero Slope Line Mass Flow given the
parameters and normalized corrected speed vector.
F_WChL_switch - Computes the choke line mass flow given the
parameters and normalized corrected speed vector.
F_WChL_atan - Computes the choke line mass flow given the
parameters and normalized corrected speed vector.
F_PIZSL - Computes the Zero Slope Line pressure ratio given
the parameters and normalized corrected speed vector.
F_PIChL - Computes the choking pressure ratio given the
parameters and normalized corrected speed vector.
F_PI0 - Computes the pressure ratio at zero flow given the
parameters and normalized corrected speed vector.
F_kloss_none - Computes k_loss values given the parameters
and the corrected speed and mass flow values.
F_kloss - Computes k_loss values given the parameters and
the corrected speed and mass flow values.
F_CUR - Computes CUR values given the parameters and the
normalized corrected speed vector.
F_b_SAE - Computes B value given the parameters and the
normalized corrected speed vector.
F_b_ASME - Computes B value given the parameters and the
normalized corrected speed vector.
F_a_SAE - Computes A value given the parameters and the
normalized corrected speed vector.
F_a_ASME - Computes A value given the parameters and the
normalized corrected speed vector.