Documentation for using HeatEstim graphical user interface in MATLAB environment

1. Description of HeatEstim interface

HeatEstim is a software application for numerical estimation of the specific heat capacity *c*, [*c*] = J/(kg.K), of certain chemical electrolytes. The calculations needed are based on solving the non-stationery ordinary differential heat equation and consecutive parametric identification. This equation is treated in general as a non-linear differential equation, hence higher-order numerical methods are being applied for numerical integration. Here the Runge-Kutta method of fourth order is preferred [2], [4].

HeatEstim graphical user interface is created entirely in the integrated software environment of MATLAB. It depends on the current version of MATLAB and cannot be run outside of its scope. Besides, HeatEstim cannot be run as an *.exe file* stored in a separate directory or a file repository.

2. HeatEstim interface functionalities

HeatEstim application finds an approximation for the specific heat capacity *c* of the most common non-linear heat equation of first order

$$(1)\ [c(T)a(t,T)]T'_t = f(t,T),\ t>t_0$$

at a given initial condition

$$(2)\ T(t = t_0) = T_0.$$

In equation (1) we consider the heat capacity a function of the temperature $T(t)$. Additionally we assume that the coefficient *a* and the right-hand side *f* are functions of time *t* and the temperature *T* simultaneously.

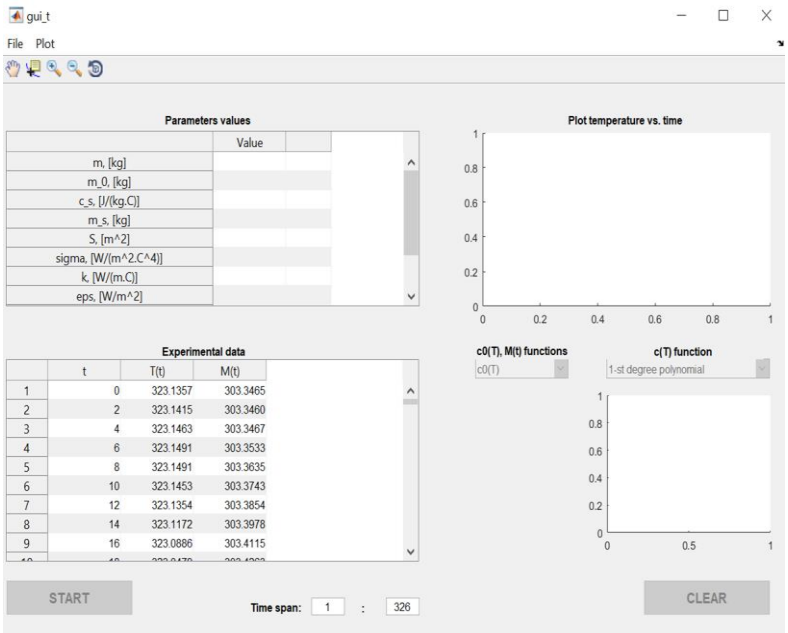


Fig. 1. *HeatEstim application graphical layout*

In fig. 1 the initial layout of HeatEstim is shown. The application is created using the GUIDE environment of MATLAB [3], [5], [6] and is composed of the following main components.

a) A table with a title **Parameters values** that contains the values of all parameters included in equation (1). These values need to be typed interactively by the user. More detailed information about the meaning of the mentioned parameters can be found in the project report.

b) A table with a title **Experimental data** that contains the experimental values of time t, the

temperature T and the environment temperature M(t). The data is loaded by default but it can be dynamically set-up by the user via MATLAB workspace variables.

c) An empty axes titled **Plot temperature vs. time** which will handle both the plots of the experimental temperature data vs. time and the numerical temperature data vs. time. The second plot is visualized only after finding an approximation for the heat capacity coefficient.

d) A second empty axes which will handle the plot of the heat capacity approximation as a function of the temperature.

e) A popup menu for typing the analytical formulae of the functions $c_0(T)$ and $M(t)$. Its initial state is set up to `'Enable' = 'off'`. More detailed information about the meaning of both functions can be found in the project report.

f) A popup menu for choosing the degree of the heat capacity polynomial approximation, first, second, third and fourth respectively. Its initial state is set up to `'Enable' = 'off'`.

g) Two edit text fields for typing time indices corresponding to the initial and final time values respectively. The application performs preliminary verification for correct input values. In case of incorrect values a dialogue box describing the error appears.

h) A **START** button for running the application. Initially this button is set up to be inactive and it will be activated only after both tables **Parameters values** and **Experimental data** and both edit text fields are filled with correct numerical values. The activation is performed by right-clicking on whichever cell of the **Parameters values** table.

i) A **CLEAR** button for clearing all the information for the current computer simulation. On starting the application this button is set up to be inactive. It returns the application into its initial state.

j) Five initially invisible edit text fields for output on the screen the estimated coefficients of the heat capacity approximation.

k) A toolbar with five components: a pan button, a data cursor button, a zoom in button, a zoom out button and a rotate button.

l) A standard menu bar with two menus, **File** and **Plot**. The **File** menu contains two components, **Export ODE solution for estimated parameters values** and **Export heat capacity coefficients values**. The **Plot** menu contains a single component, **Plot analytic heat capacity**. The sub-items of the menus are set to be inactive. They become active during the running of the application.

3. Usage of HeatEstim interface

3.1. Input parameters values

The table titled **Parameters values** stores the values of all model parameters. The values must belong to some of the numerical types (`int`, `float` or `double`). Every other data type different than the mentioned ones is considered as an input error and is indicated by a dialogue error popup window. In fig. 2 a segment of the already filled-in table is shown.

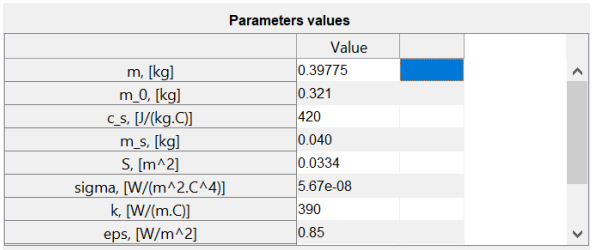


Fig. 2. *Input parameters values*

3.2. Setting up experimental data values

On running the application, the **Experimental data** table stores already loaded data corresponding to measurements for pure water electrolyte. The user might use a set of experimental data different from the loaded one and with accordance to the parameters of another electrolyte. In this case one needs to perform the next actions in order to change the initial experimental data set.

a) Navigating the computer system to a file that contains the desired experimental data. The structure

of the file must correspond to the table structure, i.e. a matrix with a finite number of rows and three columns.
b) Retrieving and loading the file data in the MATLAB workspace using for instance the built-in `uigetfile()` function.
c) Storing the loaded data in a local variable.
d) Setting up the **Experimental data** table content to the values stored in the local variable. In GUI editor, by right-clicking on the area of the Experimental data table, we choose **Table property editor → Data** and set up the table data to `Exp_Data`. Here `Exp_Data` is the local variable storing the experimental data extracted from the file (see fig. 3).

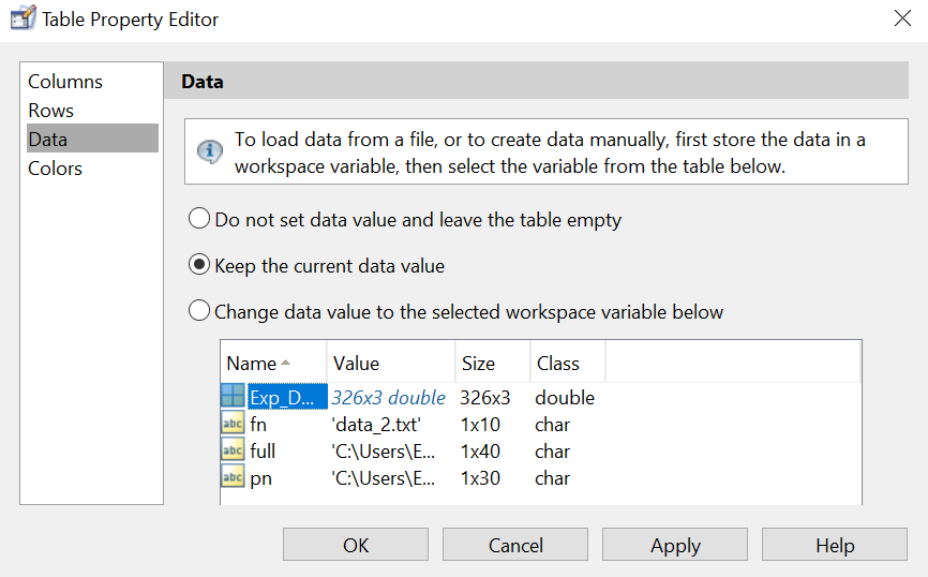


Fig. 3. *Setting up the experimental data*

3.3. Setting up time spanning

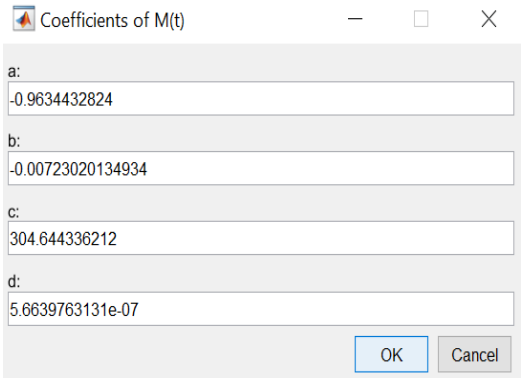
The initial time spanning is according to the initial experimental data set. The starting index must be at least 1 and at most the total number of elements in the time vector column. The final index must be at least 2 and at most the total number of elements in the time vector column. In all other cases (including typing non-numeric values, Not-A-Number values, infinite values or empty array values) the MATLAB interpreter will raise an error and will indicate by printing an appropriate error message on a popup dialogue window.

3.4. Activating START button

After all required data is once set up, the START button is ready to be activated. Right-clicking on whichever cell of the Experimental data table enables the button.

3.5. Input functions definitions

Besides filling up the tables in a proper way, the user must provide information about the functions $c_0(T)$ and $M(t)$ as well. In general the function $c_0(T)$ is defined as a rational fraction with both temperature-dependent first-degree numerator and denominator. The function $M(t)$ corresponds to the environment temperature. The lab experiments unequivocally show slowly increasing in its values and asymptotic approaching to a horizontal line. Besides, the function is known over a discrete region. The user must perform a preliminary data fitting using the guess $M(t) \approx ae^{bt} + ce^{dt}$. When the START button is pressed, the popup menu for defining $c_0(T)$ and $M(t)$ becomes active and is ready for user input. If $c_0(T)$ and $M(t)$ functions are typed incorrectly (i. e. some of their coefficients are non-numeric values, Not-A-Number values, infinite values or empty array values), the MATLAB interpreter raises an error.



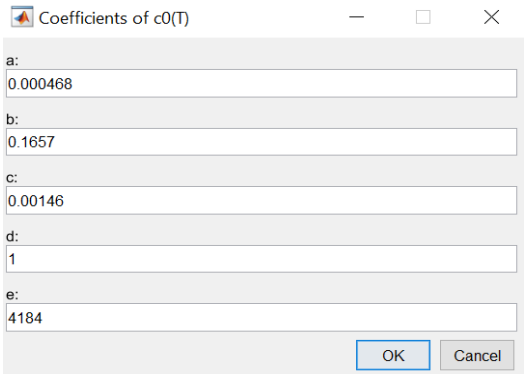


Fig. 4a) **Fig. 4b)**

Fig. 4. *Defining $c_0(T)$ and M(t) functions: 4a) – M(t), 4b) – $c_0(T)$*

3.6. Polynomial approximation for the specific heat capacity

In Physics, a global theoretical formula for evaluating specific heat capacity has not been established yet. Such a formula is known only for a tight class of substances, for instance pure water and lithium chloride solution with a certain concentration of the active component. For each electrolyte we search for a local lower-degree approximating polynomial formula (at least 1-st and at most 4-th degree). The identification of the unknown coefficients of the approximating formula is performed using Parameter Estimation Method. The application implicitly invokes the built-in procedure `lsqcurvefit()` whose syntax is the following:

```
lsqcurvefit(fun,x0,xdata,ydata,lb,ub).
```

Here `fun` is a handle to the objective function to be fitted in least squares sense, `x0` is an initial guess vector, `xdata` and `ydata` are the x- and y-coordinates respectively of the experimental data for the fitting, `lb` and `ub` are vectors containing lower and upper bounds for the estimated parameters [1]. The procedure attempts to find the values of the estimated parameters so that the solution of the ODE is as close as possible to the experimental data stored in the **Experimental data** table. It is important to underline that if lower and/or upper bounds are not specified, then return reasonable estimated values is not guaranteed.

In dialogue mode the user performs an input of the lower and the upper bounds. The values must be typed as a list whose elements are separated with whitespace (no comma or semicolon) and are not framed by any brackets (neither square nor round ones). If other separator is used, the MATLAB interpreter indicates for error.

The initial guess requires dialogue input of each coordinate of the vector with the estimated parameters. This starting point must refer to some numeric type (`int`, `float` or `double`) otherwise the interpreter raises an error and the application running gets corrupted.

In fig. 5 an output of a HeatEstim simulation is shown. The upper axes handles the graphical comparison between the experimental data stored in the **Experimental data** table and the ODE solution corresponding to the estimated parameters. The lower axes handles the graphical representation of specific heat capacity vs. temperature dependence and provides a context menu for certain line specifications (color, line width and line style; see fig. 6). On finishing `lsqcurvefit()` algorithm a dialogue window brings to the screen a text message for successful estimation process and on the left from the lower axes the estimated parameters values are being printed.

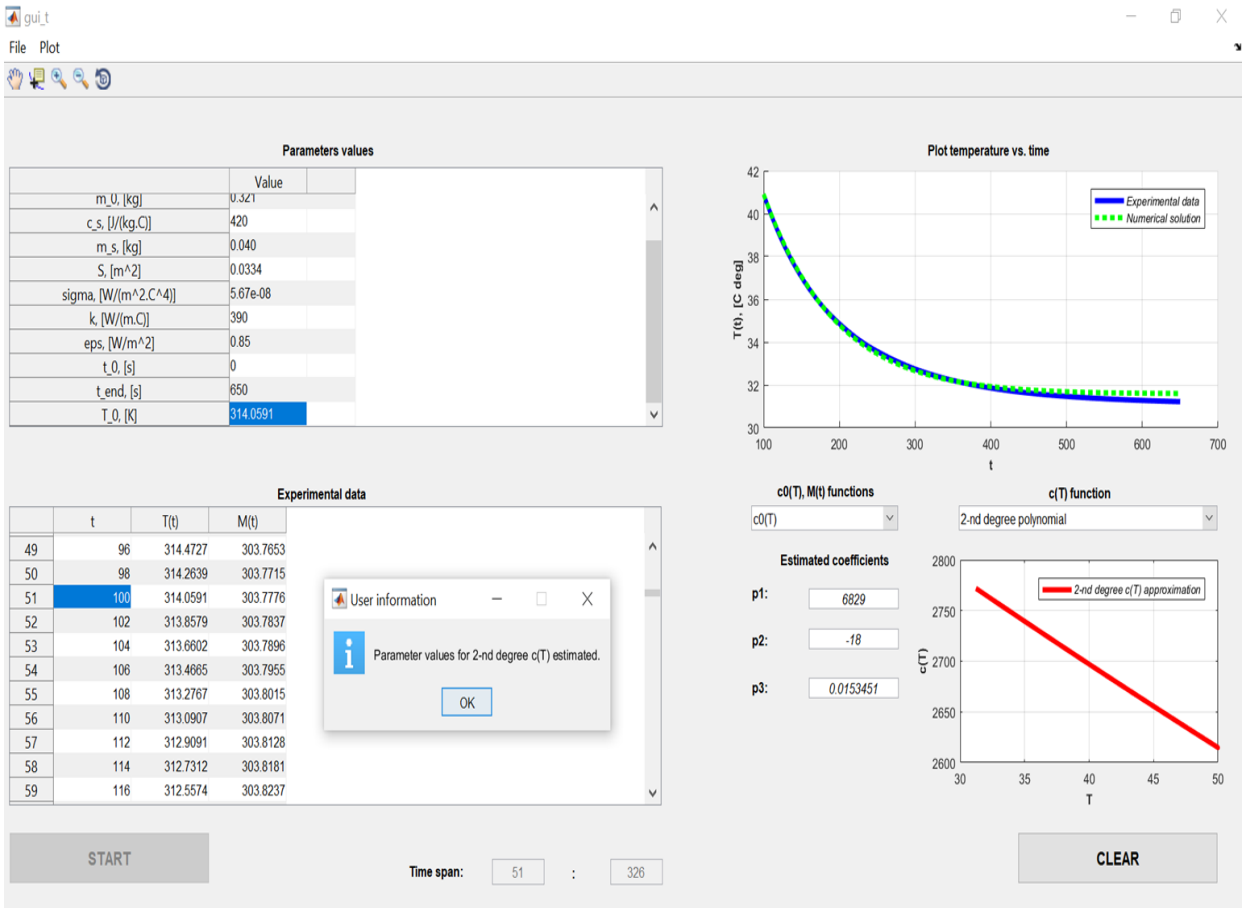


Fig. 5. *HeatEstim graphical output*

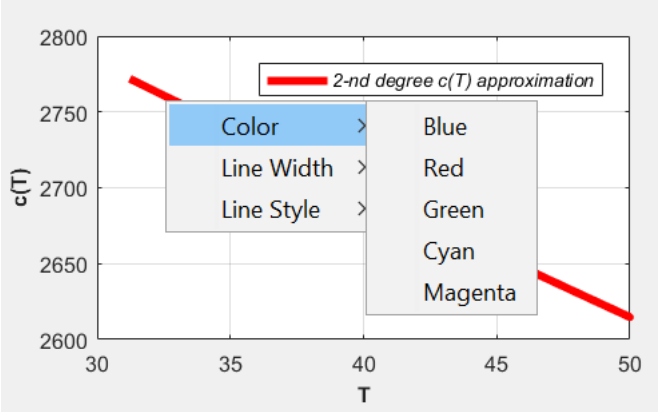


Fig. 6. *Context menu for line object*

3.7. Graphical toolbar

The application provides a toolbar with five components for interacting with the graphics in the axes: a pan button, a data cursor button, a zoom in button, a zoom out button and a rotate button. These components can be used as in any other graphical editor.

3.8. Standard menu bar

HeatEstim provides as well a menu bar with two menus, **File** and **Plot**.

The **File** menu contains two components: **Export ODE solution for estimated parameters values** and **Export heat capacity coefficients values**. Left-clicking on them leads to storing either the ODE solution corresponding to the estimated parameters or the heat capacity coefficients values in files with extension *.txt* located in the current directory. The successful data saving is being notified to the user via a dialogue message window (see fig. 7a), 7b)).

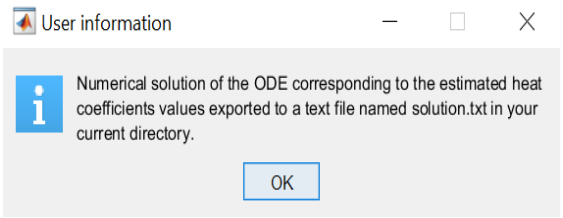


Fig. 7a). *Saving ODE solution*

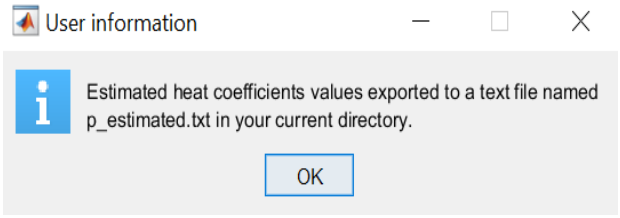


Fig. 7b). *Saving heat capacity coefficients*

The **Plot** menu provides the option to graphically compare the estimated heat capacity and the analytic one if such is known. A question dialogue window asks the user for the existence of analytic heat capacity (see fig. 8). If yes, then the application proceeds with plotting the curves. If no, a message dialogue window prints a short user information (see fig. 9).

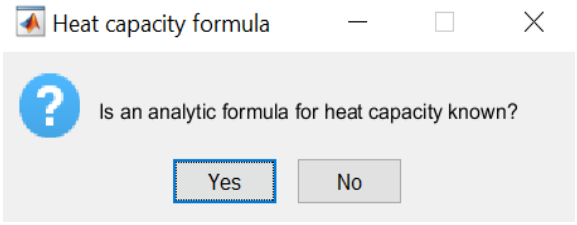


Fig. 8. *Question dialogue window*

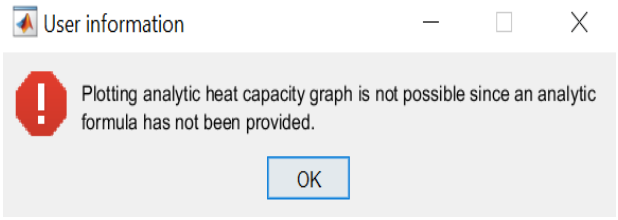


Fig. 9. *User information dialogue window*

3.8. Clear the simulation

Left-clicking on the CLEAR button causes all the numeric and visual information from the current simulation to be cleared. All buttons and menus become inactive. The **Parameters values** table content becomes empty. The table **Experimental data** holds a default data set that can be modified at any moment.

References

[1] *Documentation of the built-in MATLAB procedure lsqcurvefit()*. https://www.mathworks.com/help/optim/ug/lsqcurvefit.html
[2] *Runge-Kutta methods*. https://www.en.wikipedia.org/wiki/Runge-Kutta_methods
[3] **Y. Tonchev.** *Matlab 7. Conversions, computations, visualization, I, II, III parts* (in Bulgarian). Tehnika Publishing House. Sofia, 2009
[4] **St. Dimova, T. Chernogorova, A. Yotova.** *Numerical methods for differential equations*, pp. 21 – 24. St. Kliment Ohridski University Publishing House. Sofia, 2010
[5] **The MathWorks, Inc.** *Creating Graphical User Interface with MatlabR2015b*. Natick, MA, 2015
[6] **The MathWorks, Inc.** *MATLAB. The Language of Technical Computing. Computation. Visualization. Programming*. Natick, MA, 2005