*GAPI*Lib - A Modelica Library for Model Parameter Identification Using Genetic Algorithms

Miguel A. Rubio+, Alfonso Urquia*, Leandro Gonzalez+, Domingo Guinea+, Sebastian Dormido*

+ Instituto de Automática Industrial (IAI), CSIC
Ctra. Campo Real, Km. 0,200 – La Poveda, 28500 Arganda del Rey, Madrid, Spain
E-mail: {marubio, leandrog, domingo}@iai.csic.es

* Departamento de Informática y Automática, ETS de Ingeniería Informática, UNED
Juan del Rosal 16, 28040 Madrid, Spain
E-mail: {aurquia, sdormido}@dia.uned.es

Abstract

The design, implementation and use of *GAPI*Lib is discussed in this manuscript. *GAPI*Lib is a new Modelica library for parameter identification in Modelica models, using genetic algorithms (GA). *GAPI*Lib can be used for parameter estimation in any Modelica model and the estimation process does not require to perform model modifications. This new library supports simple- and multi-objective optimization. *GAPI*Lib library is composed of a set of functions that can be easily used, modified and extended. The use of *GAPI*Lib is illustrated by means of a case study: the estimation of electrochemical parameters in fuel cell models, which have been composed by using *FuelCell*Lib library. *GAPI*Lib is completely written in Modelica language and it will be freely available soon.

Keywords: Genetic Algorithm, Fuel cell, parameter identification

1 Introduction

Frequently, the modelling process includes the estimation of model parameters from experimental data. The use of genetic algorithms (GA) to perform this task is broadly accepted.

GA are numerical optimisation algorithms inspired by the natural selection processes that take place among the live beings. Individuals evolve thought adaptation to their external environment. Most capable individuals pass on their genetic information to later generations. As a consequence, the population evolution after a number of generations allows to obtain optimum results.

The use of GA for parameter estimation in Modelica models has been previously proposed by Hongesombut et al. [1]. However, these authors programmed and ran the GA using Matlab/Simulink. As a consequence, these authors' approach requires the combined use of Modelica/Dymola and Matlab/Simulink.

The lack of a freely-available Modelica library implementing GA, suited for parameter estimation in Modelica models, has motivated the implementation of *GAPI*Lib library.

The fundamentals of the GA supported by *GAPI*Lib library are briefly explained in Section 2 and the library structure is discussed in Section 3. Finally, the use of *GAPI*Lib is illustrated by means of a case study: the estimation of electrochemical parameters in fuel cell models. This case study is described in Section 4.

2 GA supported by *GAPI*Lib

The GA supported by *GAPI*Lib library is schematically represented in Figure 1. The algorithm consists of the steps described next.

The GA starts with an initial population, which is randomly selected from the search space. Each individual of the population is formed by a group of chromosomes, which represents a solution to the problem.

This initial population is evaluated by using a cost function. This function is used to calculate the validity of the population members, which are ordered according to this criterion. The most valid member is selected for the crossover process, which generates a new population. This new population is evaluated and recombined to obtain a new generation, and so on. These steps of the algorithm are repeated until the stop condition is satisfied (see Figure 1).

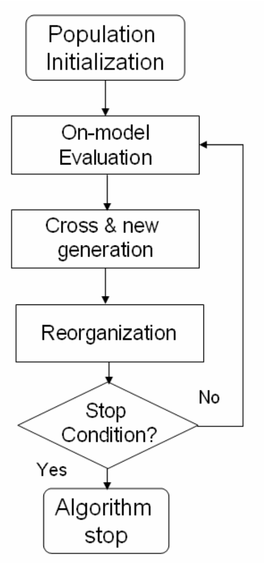


Figure 1: GA supported by *GAPI*Lib

The GA supported by *GAPI*Lib includes several processes intended to improve the algorithm performance, such as:

1. *Elitism*. Most valid individuals are passed on to the next generation without being altered by genetic operators. Using elitism ensures that the best solution is never lost from one generation to the next.
2. *Mutation*. It is a genetic operator that introduces random changes on the individuals, maintaining genetic diversity from one generation of the population of chromosomes to the next. The purpose of mutation is to allow the algorithm to avoid local minima.

A candidate problem solution is a value selection of the parameters under estimation. These parameter values are included in the model and this is simulated. A "fitness function" is applied to the obtained model response in order to evaluate the candidate solution. The next candidate solution is obtained by applying different genetic operators. The new values of the parameters are included in the model, which is simulated, and so on.

3 *GAPI*Lib structure

*GAPI*Lib has been programmed by combining the use of the scripting Modelica language (see Figure 2a) and of functions written in Modelica language (see Figure 2b). The overall library structure is schematically represented in Figure 2.

The GA execution starts by running the script file `GAPILib.mos`. This file only contains the sentences required to execute the following two script files: `GAPILIb_INI.mos` and `GAPILib_CYCLE` (see Figure 2a). The purpose of these files can be summarized as follows:

- The script file `GAPILIb_INI.mos` carries out the initialization of the *GAPI*Lib parameters and generates the initial population.
- The script file `GAPILib_CYCLE` performs the operations required to obtain the next generations. Its execution finishes when the stop condition is satisfied. The stop condition shown in Figure 2a is of the type: "N_Cycle generations have been obtained". Other stop conditions are possible, e.g., "the calculated fitness value is smaller than a given value".

Further details about these two script files are provided next.

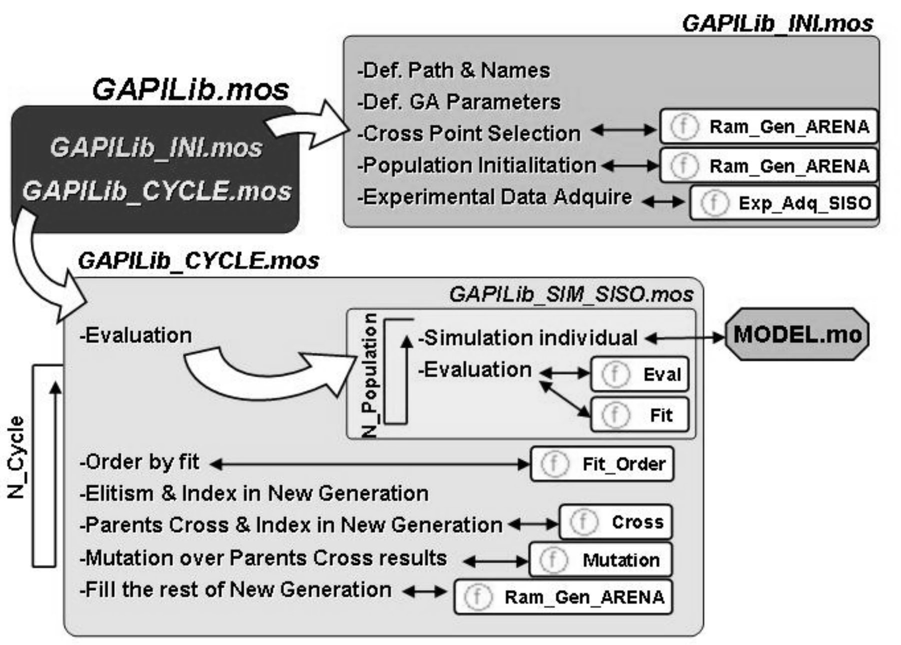
3.1 Script file `GAPILIb_INI.mos`

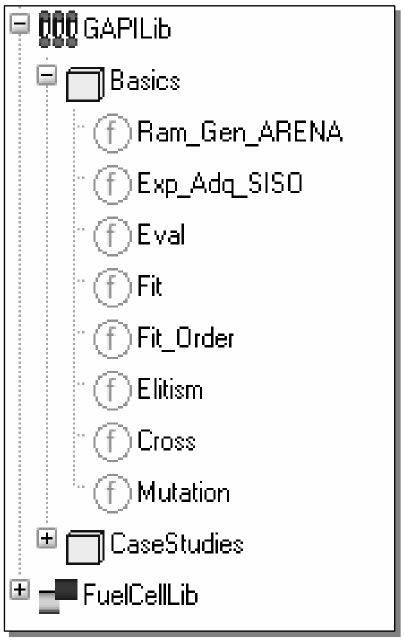
The script file `GAPILIb_INI.mos` contains the required function calls to perform the following tasks (see Figure 2a):

1. To define the directory path and the name of the Modelica models. This is the only place where the user has to provide this information.
2. To set the value of the GA parameters, which are listed in Table 1.

Table 1: *GAPI*Lib's GA parameters

`N_Population`	Number of individuals in the population.
`N_Parameters`	Number of parameters to identify in the model (i.e., in `Model.mo`).
`N_Parents`	Number of parents of the population selected to cross.
`N_Elitism`	Number of elite individuals. If `N_Elitism=0`, then the Elitism function is no applied.
`N_Cross_Point`	Number of crossing points.
`N_Cycle`	Number of generations calculated. This value sets the stop condition.
`F_Mut`	Probability of random modification of an individual due to mutation.





a) b)

Figure 2: Schematic representation of *GAPI*Lib library architecture

3. To carry out the random selection of the crossing points used for the crossover process. The function `Ram_Gen_ARENA` is used to generate the pseudo-random numbers. This function implements the pseudo-random number generator used by Arena 7.0 simulation environment.
4. To select the initial population, which is composed of random elements. Again, the `Ram_Gen_ARENA` function is used for pseudo-random number generation. The user is allowed to select the range of each parameter under estimation. This capability allows to reduce the search space.
5. To read the experimental values used as a reference to fit the model. The `Exp_Adq_SISO` function is used to perform this task.

3.2 Script file `GAPILIb_CYCLE.mos`

The script file `GAPILIb_CYCLE.mos` contains the required function calls to perform the following tasks (see Figure 2a):

1. To execute the script file `GAPILib_SIM_SISO`, that performs the simulation of the model `Model.mo` with the parameter values corresponding to each of the individuals of the population. The model is simulated as many times as individuals are in the population. The simulation results are stored and compared with the experimental data. The `Eval` and `Fit` functions are used. All the population individuals are evaluated.
2. To sort the population individuals according to the fitness values previously calculated. The `Fit_Order` function is used.
3. To pass on the elite individuals to the next generation. These individuals are not altered by crossover and mutation.

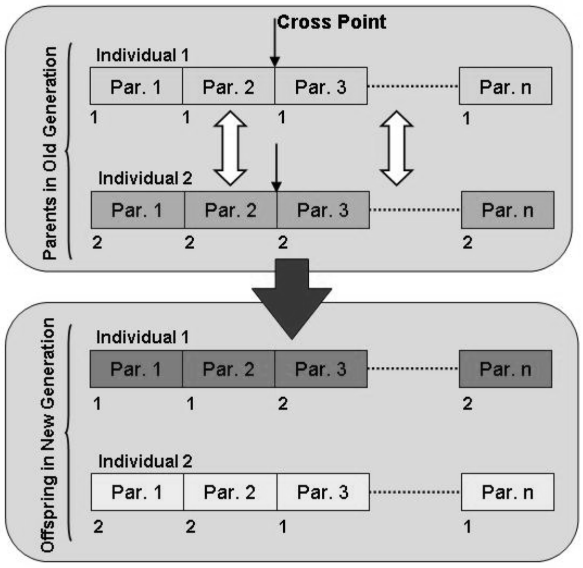


Figure 3: New generation obtained by crossover

Table 2: Input and output variables of *GAPI*Lib functions

Function name	Input variables	Output variables
Ram_Gen_ARENA	• Seed [1,6]: seed of ARENA algorithm • N_Ram: number of pseudo-random numbers to generate	• Ram_list:[1,N_Ram]: Array of N_Ram pseudo-random numbers
Exp_Adq_SISO	• namefile: name of .mat file where the experimental data is stored • Xmatrixname: name of X variable matrix • Ymatrixname: name of Y variable matrix	• State_Exp_Adq: state of experimental data • SizeMatrix [2]: size of the Xmatrixname and Ymatrixname matrixes • Xexp: array of experimental X data • Yexp: array of experimental Y data
Eval	• SimuPath: path of the model Model.mo • SimuCaseStudyName: name of the model to simulate • Xexp: Array of experimental X data • New_Generation: complete set of population values	• DATASim_Int: interpolated values with Xexp of model simulate result
Fit	• DATASim_Int: interpolated values with Xexp of model simulate result • Yexp: array of experimental Y data	• Eval_Mod: fitness evaluation result
Fit_Order	• Pop: population to be ordered • Fit: fitness of all population • Population: number of population individuals • Parents: number of population parents • Elitism: number of elitism individual	• Pop_Ordered: population sorted by fitness value
Cross	• Parents: number of population parents • Elitism: number of elitism individual. • OldGeneration: old generation of population ordered by fitness	• NewGeneration_Cross: population obtained from elitism and crossover
Mutation	• Nparameters: complete population parameters of NewGeneration_Cross • F_Mut: mutation factor • RamMut: pseudo-random number generated using Ram_Gen_ARENA. This number is used to decide whether a parameter is mutated and its value	• NewGeneration_Mut: population obtained after elitism, crossover and mutation

4. To cross the selected parents (the most capable individuals), using the crossing point calculated from GAPILIb_INI. The Cross function is used. The algorithm implemented is shown in Figure 3.
5. To apply the Mutation function. The mutation factor is the probability used to mutate any parameter of an individual.
6. The new population is completed with random elements. The Ram_Gen_ARENA function is used.

The input and output variables of *GAPI*Lib functions are shown in Table 2. These functions are stored within the GAPILib.Basics package (see Figure 2b).

4 Case study

*GAPI*Lib has been successfully applied to the estimation of electrochemical parameters in fuel cell models composed by using *FuelCell*Lib. The packages and models of *FuelCell*Lib are shown in Figure 4. Further information about this free Modelica library can be found in [2].

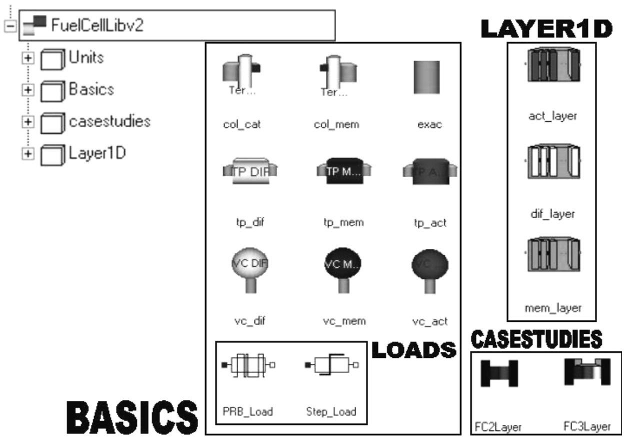


Figure 4: Packages and models of *FuelCell*Lib

The obtained models can be used to simulate the steady-state and the dynamic behavior [3,4,5] of the fuel cells along their complete range of operation. For instance:

- The experimental and simulated polarization curve (I-V) of a fuel cell is shown in Figure 5.
- The experimental and simulated fuel cell voltage, obtained in response to step changes in the load, is shown in Figure 6a.
- The simulated vs. experimental data of the water long-term effect is shown in Figure 6b. The simulation reproduces: (1) the slow voltage rise due to the membrane hydrate; and (2) the voltage fall due to the water flooding of the cathode.

Next, the these three fitness processes are discussed.

The experimental data used in this work have been obtained in the laboratory of renewable energy of the IAI, CSIC.

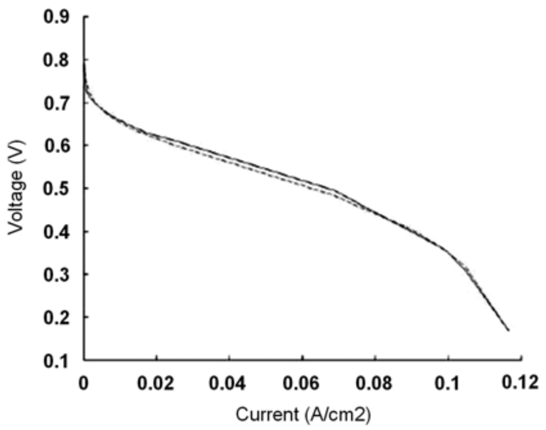
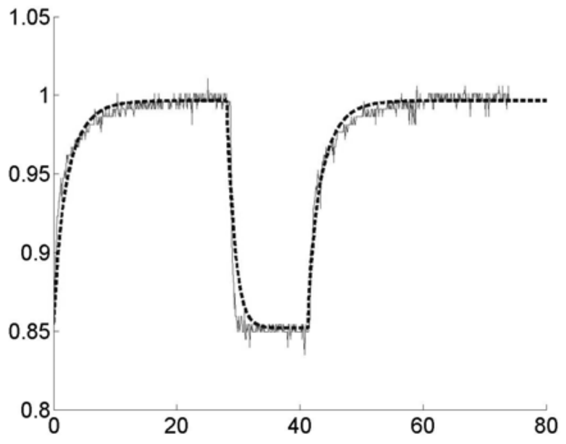
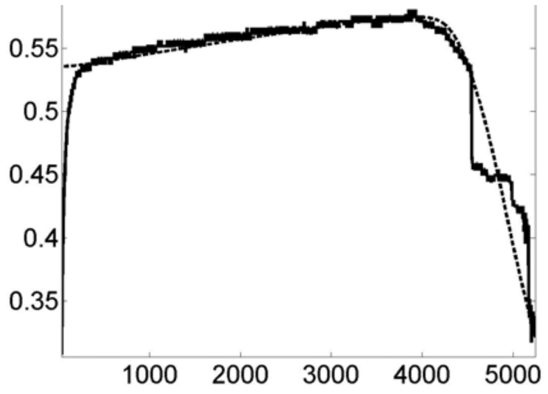


Figure 5: Fuel-cell polarization curve: (o) experimental; (--) simulated using *FuelCell*Lib



a)



b)

Figure 6: Experimental (__) and simulated using *FuelCell*Lib (- -). Time [s], X axis. Voltage [V], Y axis. a) Fuel cell voltage in response to step changes in the load. b) Long-term effect of the water, with constant load.

4.1 Fitness of the polarization curve

In order to obtain the polarization curve, the model has to be simulated, for each of the operation points composing the curve, until the steady-state is reached.

The parameters identified to fit the polarization curve are shown in Table 3. In all experiments, one crossing point was used. The fitness function is defined as the sum of the quadratic differences between the experimental and the simulated values of the variable. I.e.:

$$\mathrm{Fit} = \sum(Y_i-\hat{Y}_i)^2 \qquad (1)$$

The GA parameters are set to the following values:

- The stop condition is satisfied after 5000 generations.
- Each population was composed of 100 individuals.
- The mutation factor was 0.25.
- 70 parents and one elite element were used in each generation.

The values of a parameter obtained with best fitness value are shown in the Table 4.

Table 3: Parameters to estimate

Parameter	
A	Tafel slope
I_n	Inner current density
Io	Exchange current density
B	Mass transfer slope
R	Inner area specific resistance
I_{lim}	Limiting transport current density

Table 4: Result of the fitness

Parameter	Value
A (V)	0.0390
I_n (A cm^{-2})	1.4×10^{-3}
Io (A cm^{-2})	1.5856×10^{-6}
B (V)	0.0918
R (Ω cm^{2})	7.2860×10^{-4}
I_{lim} (A cm^{-2})	0.2265

4.2 Fitness of the fuel cell voltage in response to step changes in the load

It is this case, *GAPI*Lib is used to estimate the values of the four parameters shown in Table 5. The GA parameters are set to the following values:

- The stop condition is satisfied after 200 generations.
- 150 individuals were used in each population.
- A factor of mutation of 0.25 was applied.
- 100 parents were used in each generation.
- One elite element was used.

The obtained values of the parameters are shown in the Table 6.

Table 5: Parameters to estimate

Parameter	
R_{inf}	Down resistance value
R_{sup}	High resistance value
C_{dl}	Doble layer capacitance
k_s	Electrical conductivity od the solid

Table 6: Results of the fitness

Parameter	Value
R_{inf} (Ω m^{-2})	0.03315
R_{sup} (Ω m^{-2})	5.1
C_{dl} (F m^{-2})	10.12
k_s (S m^{-1})	0.01

4.3 Fitness of the long term effect of water on the fuel cell voltage with constant resistance load

To carry out this experiment, the fuel cell model was modified in order to reproduce the variation of the membrane conductivity. The parameters used to fit the model are shown in Table 7.

The GA parameters are set to the following values:

- The stop condition is satisfied after 700 generations.
- 70 individuals were used in each population.
- A factor of mutation of 0.15 was applied.

- 50 parents were used in each generation.
- One elite element was used.

The obtained values of the parameters are shown in Table 8.

Table 7: Parameters to estimate

Parameter	
da(Act_Layer)	Width of active layer
X_s	Percentage of pore volume of solid
D_{12}	Binary difusion coefficient
d_a(Mem_Layer)	Width of membrane layer
R_{mem}	Resistance of membrane layer

Table 8: Result of the fitness

Parameter	Value
da(Act_Layer) (m)	6×10^{-8}
X_s	0.05
D_{12} (m^2/s)	5.01×10^{-9}
d_a(Mem_Layer) (m)	1.6×10^{-5}
R_{mem} ($\Omega\ m^{-2}$)	1.419×10^{-3}

5 Future work

Next version of *GAPI*Lib will support the solution of multi-objective problems. Also, it will include additional fitness functions and the user will be allowed to select among them the fitness function best suited to each particular problem.

In addition, a complete guide of *GAPI*Lib use will be developed.

6 Conclusions

The design, implementation and use of *GAPI*Lib has been discussed. *GAPI*Lib library is an effective tool for parameter identification in Modelica models using GA. It is completely written in Modelica language, which facilitates its use, modification and extension. *GAPI*Lib can be used for parameter identification in any Modelica model and the estimation process does not require to perform model modifications. *GAPI*Lib has been successfully applied to the estimation of electrochemical parameters in fuel cell models, which have been composed by using *FuelCell*Lib library.

References

[1] Hongesombut K, Mitani Y, Tsuji K. An Incorporated Use of Genetic Algorithm and a Modelica Library for Simultaneous Tuning of Power System Stabilizers. In: Proceedings of the 2nd International Modelica Conference, 2002, pp. 89-98.

[2] Rubio M.A, Urquia A, Gonzalez L, Guinea D, Dormido S. *FuelCell*Lib- A Modelica Library for Modeling of Fuel Cells. In: Proceedings of the 4th International Modelica Conference, 2005, pp. 75-82.

[3] Larminie J, Dicks A. Fuel Cell Systems Explained, Wiley 2000.

[4] Bevers D, Wöhr M, Yasuda K, Oguro K. Simulation of polymer electrolyte fuel cell electrode. J. Appl. Electrochem. 27 (1997).

[5] Broka K, Ekdunge P. Modelling the PEM fuel cell cathode, J. Appl. Electrochem. 27 (1997).