

# GPdotNET v3.0 User Guide

bhrnjica  
Nov, 2013

## Software Disclaimer

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program 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 General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

## 1. Project description

GPdotNET is artificial intelligence tool for applying Genetic Programming and Genetic Algorithm in modeling, prediction and optimization of engineering problems, as well as set of linear programming based problems e.g. traveling salesman problem, assignment and transportation problems. The GPdotNET is built on top of .NET and Mono frameworks written in C# programming language which can run both on Windows, Linux based OS, or any OS which supports Mono framework. The Project started in 2006 within postgraduate study of modeling and optimization with evolutionary algorithms. As open source project, the GPdotNET is first published on November 5, 2009 on [gpdotnet.codeplex.com](http://gpdotnet.codeplex.com).

GPdotNET is very easy to use and handling with it. Even if you have no deep knowledge of GP and GA, you can apply those methods in finding solution. The project can be used in modeling any kind of engineering process, which can be described with discrete data, as well as in education during teaching students about evolutionary methods, mainly GP and GA. The project is licensed under GNU Library General Public License (LGPL).

For information about license and other kind of copyright please see <http://gpdotnet.codeplex.com/license>.



Figure1.1: GPdotNET v3 Start Screen

The GPdotNET project is hosted at <http://gpdotnet.codeplex.com> for Windows users, as well as <http://code.google.com/p/gpdotnet> for Linux users.

Main place for all news, documentation and code changes is the blog site at <http://bhrnjica.net/gpdotnet>.

**Note:** If you have never heard about GP and GA, recommendation for getting basic information about GP is [http://en.wikipedia.org/wiki/Genetic\\_programming](http://en.wikipedia.org/wiki/Genetic_programming). The wiki page also contains some links to other web sites about GP. For GA there is wiki page which contains a basic information about GA at this link [http://en.wikipedia.org/wiki/Genetic\\_algorithm](http://en.wikipedia.org/wiki/Genetic_algorithm).

GPdotNET supports the following types of use case scenarios:

1. **Model for Discrete Data** – modeling with/without prediction of discrete data by using Symbolic Regression modeling with GP,
2. **Model & Opt. for Discrete Data** - modeling with/without prediction of discrete data by using Symbolic Regression (SR) with GP and Optimizing calculated GPdotNET model by using GA,
3. **Model for Time Series** - Time Series modeling and prediction data by using SR with GP,
4. **Optimization of Analytic Function** - optimization of analytic defined function by using GA,
5. **TSP** – solver for Traveling Salesman Problem (TSP) by using GA,
6. **Assignment Problem** – solver for Assignment Problem (AP) by using GA,
7. **Transportation problem** – solver for Transportation Problem (TP) by using GA.

Listed GPModel types are shown on the Figure 1.2. To create new GPModel you can click on “**New**” button located in application bar. If you want to open existing one, you can choose “**Open**” button. Among New and Open options GPdotNET contains more than 10 created samples which describe every supported type in GPdotNET. To open one of those models click the hyperlink from the starting panel.

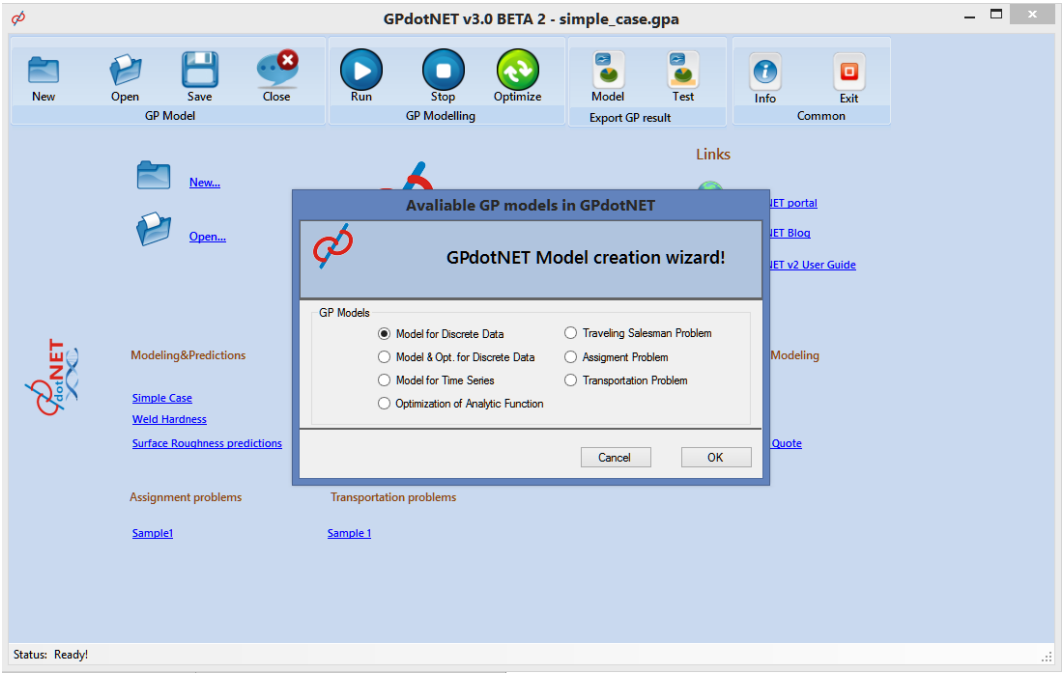


Figure1.2: GPdotNET "New Dialog" options in GPdotNET

## 2. Introduction to GPdotNET

This User Guide is related to GPdotNET version 3.0, the latest release which brings set of new solvers in linear programming. With previously 4 models, and new 3 models, with GPdotNET you can create 7 different GPModels, from modelling, optimization, time series modelling, and optimization of analytic function, TSP, AP and TP. This is huge step forward in comparing to previous version.

In the following text, it is listed the main features in GPdotNET grouped by version 1, 2 and 3.

### Cross OS and Cross platform software (new in v2)

One of the main requirements for GPdotNET is ability to run on multiple OSs, by using .NET and Mono Framework. So GPdotNET can run on all OS where Mono is implemented. During the implementation every piece of code is tested against Mono. During implementation, when code was not compatible with Mono, it was replaced with the code implementation compatible with Mono. It can be said that the whole implementation was done using Visual Studio and Mono Develop, working on Windows and Fedora 17. I didn't have much time to test GPdotNET on OS other that Windows 7 and Fedora 17, so every bug report would be appreciated.



Figure 2.1: GPdotNET in MAC OS environment.

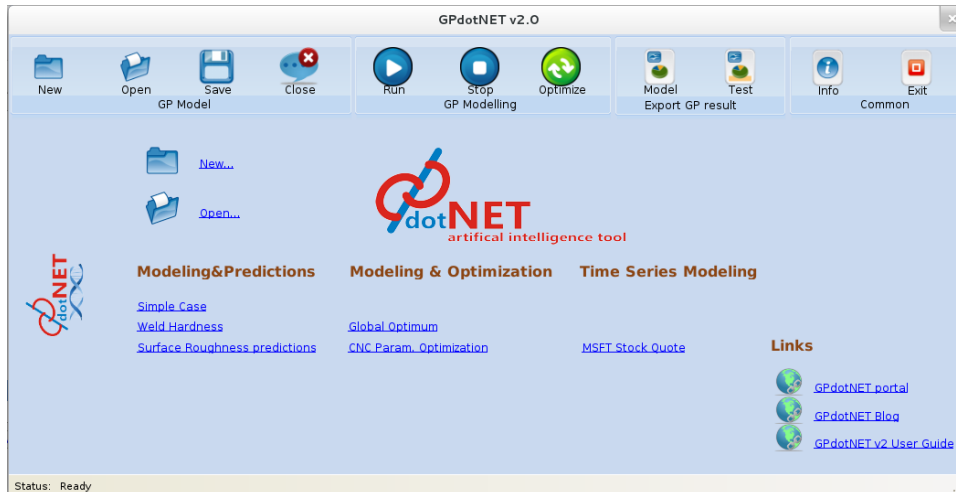


Figure 2.2: GPdotNET in Fedora 17 OS environment

### **New text based file format \*.gpa (new in v2)**

GPdotNET V1.0 supported binary file format, and for large population size the file size was also big. On the other hand, with text file format there is a possibility to modify file outside the GPdotNET. For example you can see whole population chromosomes, and see other data you are interesting in. You can also perform some manual modification if you like by modifying training or testing data as well as parameters. In general, manual modification file is not recommended.

### **Support for Excel and CSV export (new in v2)**

Exporting in GPdotNET is based on open XML file format, but there are some compatibility issues in Mono. In fact you cannot use Excel exporting in Mono. While you running GPdotNET on Mono you can export data in CSV file format, and Wolfram Mathematica. This is only one feature which is not running in both Mono and .NET.

### **Optimization of GPModels (new in v2)**

GPdotNET can run optimization of calculated GPModel. Optimization is very important for any engineering system. You can perform optimization after you perform modeling and get the parameters which your model is optimal. In fact you can run optimization and modeling as much as you want with only one constrains: "You cannot run Optimization and Modeling at the same time".

### **Optimization of analytically defined function (new in v2)**

GPdotNET now supports optimization of any analytically defined function. You can define function in "**Tree Expression Designer**" - TED, define constraints and perform optimization.

### **Traveling Salesman Problem (TSP) (new in v3)**

The solver in GPdotNET version 3 contains set of solvers based on linear programming. TSP problem represents common problem in LP, and GPdotNET supports this kind of problem in unlimited number of cities. Unlimited number of cities can be defined in GA, because of nature how GA is searching for solution.

### **Assignment problem (AP) (new in v3)**

Assignment problem present generalized TSP problem. Similar as previous it is implemented with vector based chromosome type.

### **Transportation problem (TP) (new in v3)**

The most complex implementation in GPdotNET is transportation problem which required matrix based chromosome. With this implementation GPdotNET can search for solution to almost any LP problem.

### **Support \*.csv data file**

GPdotNET support csv file format for loading training, testing and time series data. Common example of data file can be seen on the following picture.

Regardless of user localization floating numbers must be written with decimal point. On this way we skip some complexity and localization issue seen in the previous version. Columns are separated by semicolon, and rows with newline. The last column is always output variable. In case of Time Series, data file can contains only one column.

```
1500;152.4;127;1016.81;1422.4
1500;457.2;25.4;1358.05;3048
1250;152.4;25.4;901.88;1270
1000;609.6;25.4;1171.56;4140.2
1500;152.4;127;1053.35;1422.4
750;304.8;76.2;1278.61;2590.8
1500;609.6;127;1787.36;2794
1250;609.6;76.2;2196.5;2768.6
```

*Figure 2.3: Sample of csv file*



## Info tab in Model (new in v2)

When new GPdotNET model is created a new Info Tab is created as well. Info tab contains rich edit control in which you can paste or load any rich text content from text to picture. On this way, you can attach textual information of you model.

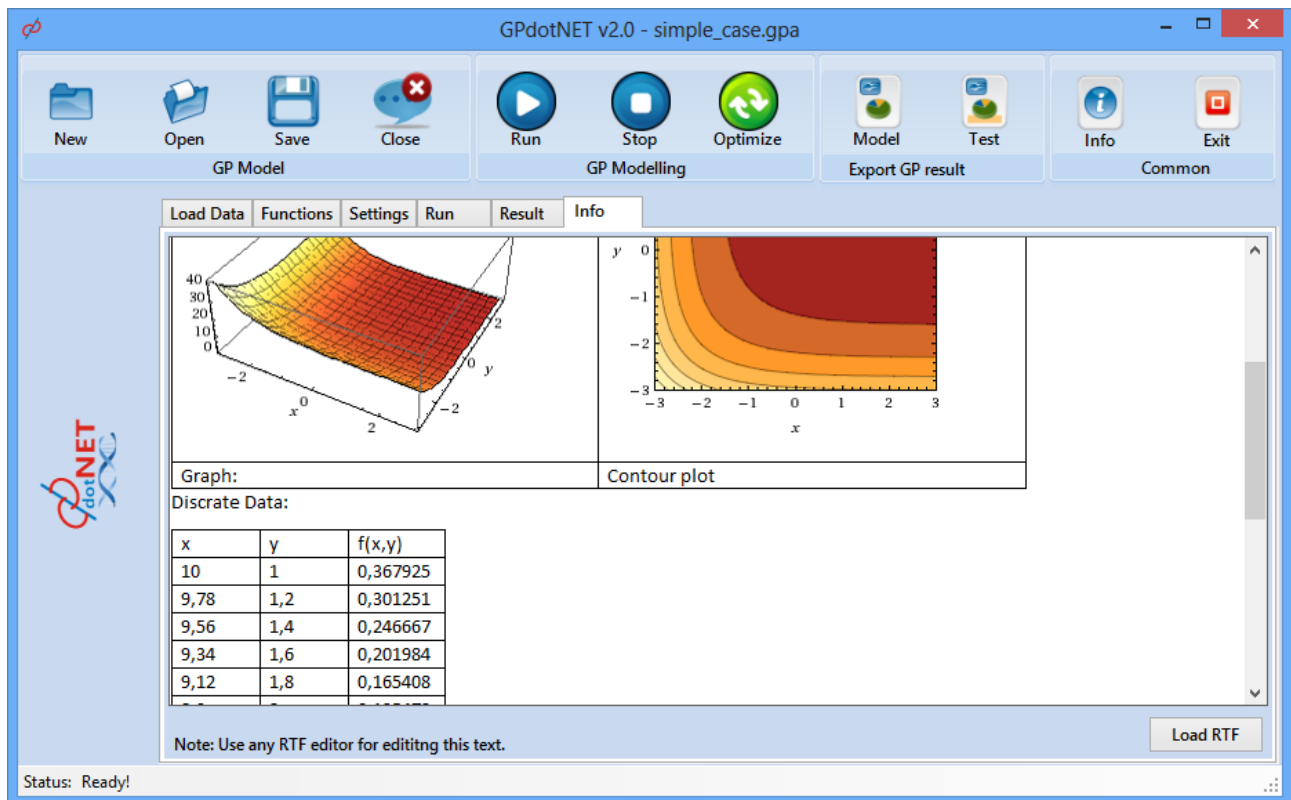
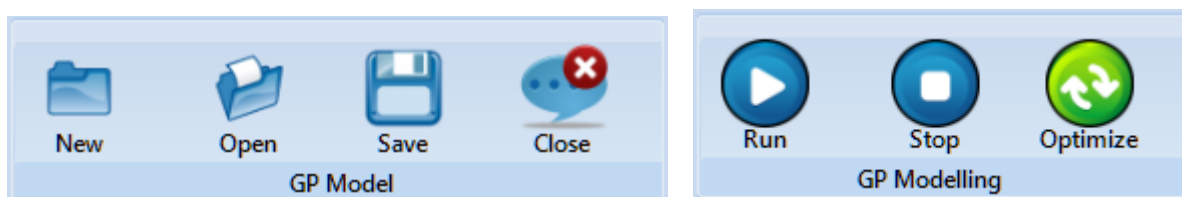


Figure 2.4: Info Tab in GPdotNET

## New Look& Feel (new win V2)

Unlike previous version, GPdotNET has new simplified GUI with only one application bar containing all available options. Commands are split in to 4 major groups: Model, Modeling, Export and Common. It is very simple and gives you all options directly on the screen. Run, Stop and Optimize commands are shifted to main toolbar, in order to give usability to stop or run programs from any tab page, not only from run page.



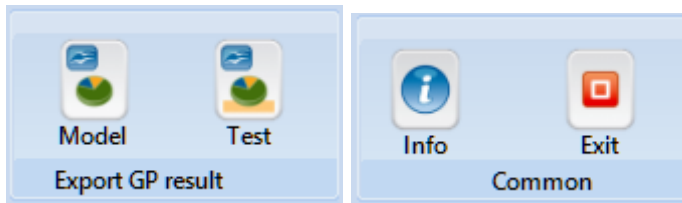


Figure 2.5: Grouped Options in GPdotNET

### 3.1. Working with GPdotNET

Picture below shows typical Start Screen of GPdotNET. Start screen can be divided in to several meaningful parts in order for better understanding UX of the GPdotNET.

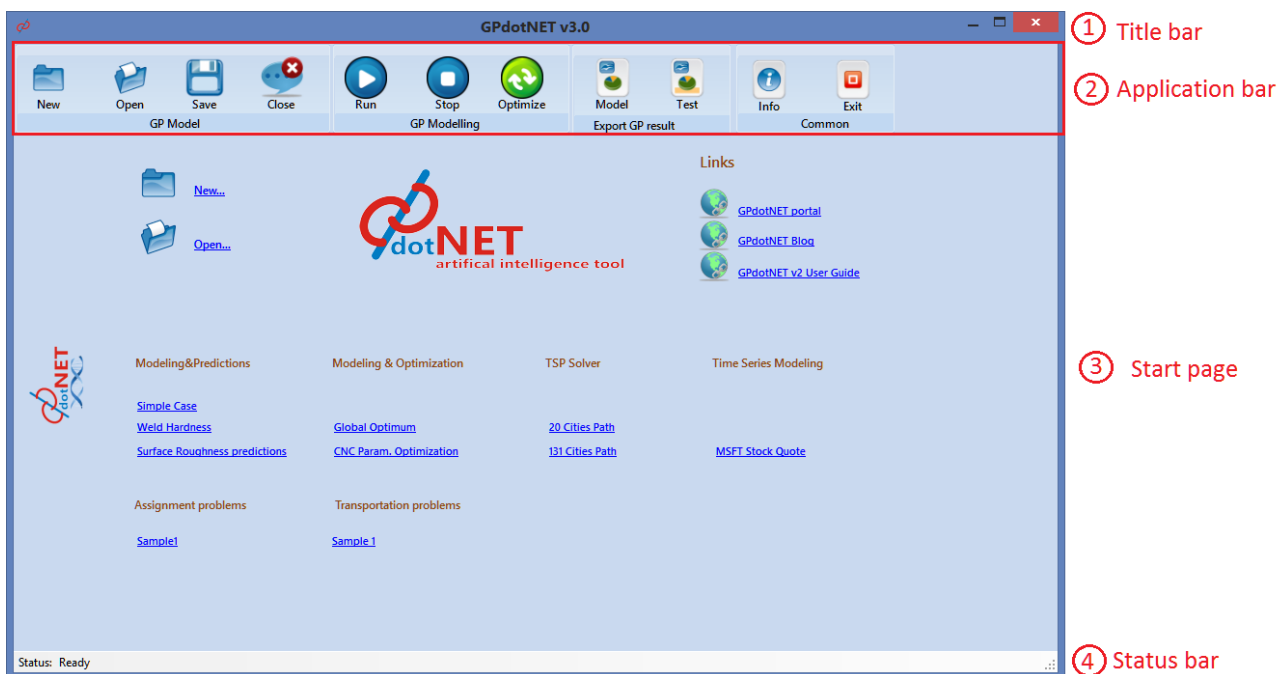


Figure 3.1: GPdotNET Start Screen

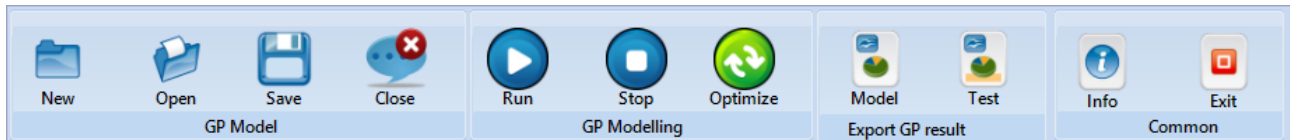
Main parts of the Starts Screen are:

1. Title Bar
2. Application Bar
3. Start Page
4. Status Bar

### 3.1. Title Bar

Title bar contains Icon, application name and system options on the right side. With system options you can Close, Maximize and Minimize application. In fact this is standard windows system options.

### 3.2. Application Bar



Main Toolbar – exposes main commands in GPdotNET. The commands are grouped in to 4 major groups.

1. GPMModel – gathers commands for manipulation of GPdotNET model file. There are options for Create, Open, Save and Save as GPdotNET model file, as well as Close currently opened model. Those commands are self-explained.
2. GP Modeling contains three commands for Run, Stop and Optimize GPMModels. Commands are enabled or disabled automatically, whenever there is a possibility user can achieve logic action. For example while GP is running, user cannot press Run button, because there is no sense to press this button. In case Stop button is enabled. Optimize button is available when GPMModel is ready for optimization.
3. Export GP Result contains options for exporting result to other format: Excel, CSV and Mathematica.
4. Common group contains option for common usage: Info to show basic information and Copyright of the application, and Exit option to close the application.

### 3.3. GPdotNET Start Page



Figure 3.2: Start Page in GPdotNET

If you are new to GPdotNET, there is no better starting point than the Start Page. With the Start Page, you can try one of the predefined and recalculated samples. In fact, the Start Page contains all the information you need to begin using GPdotNET. From recalculated samples to links for documentation, all information about GPdotNET can be found here. The Start Page contains samples which are split into 6 major groups (see Fig. above).

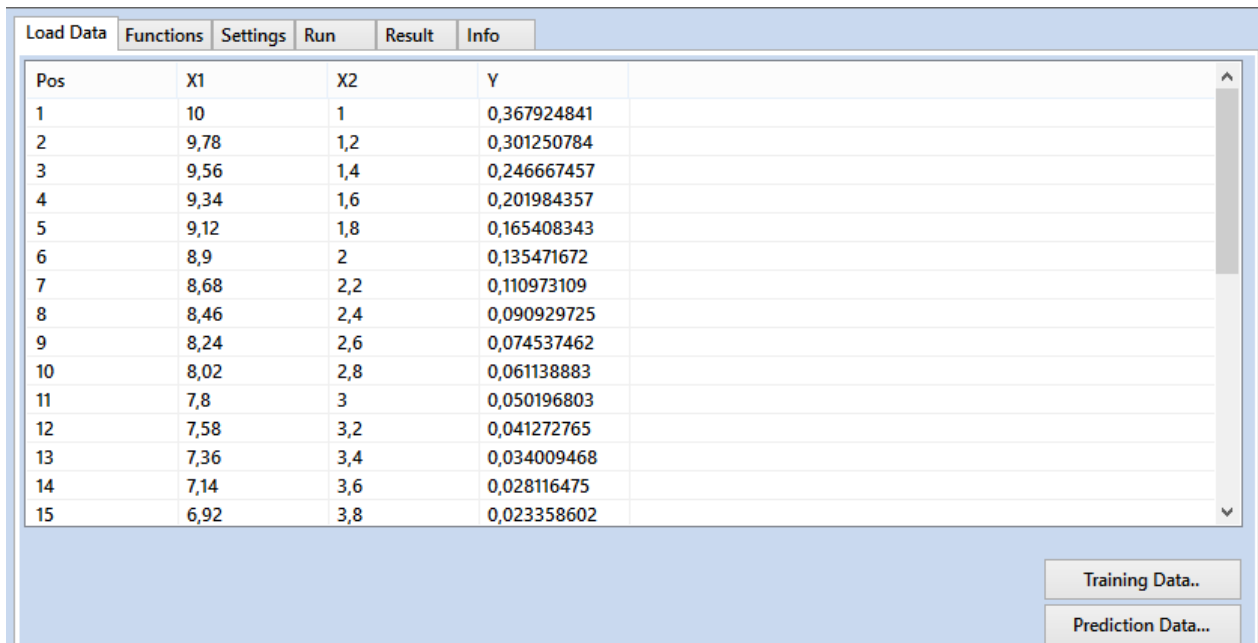
1. Modeling & predictions
2. Modeling & Optimization
3. *TSP Solver*
4. Time Series Modeling
5. *AP Solver*
6. *TP Solver*

**Note:** This version doesn't support persisting Optimization of analytic function, so there are no samples for it.

The last group of links is links for documentation and User Guide.

## 4. Working with Modelling and Prediction

Start working with GPdotNET can be as simply as clicking any of the predefined use case sample. If you click on **Simple Case** link from Start Page. After some time GPdotNET Model is loaded similar picture shows below.



Pos	X1	X2	Y
1	10	1	0,367924841
2	9,78	1,2	0,301250784
3	9,56	1,4	0,246667457
4	9,34	1,6	0,201984357
5	9,12	1,8	0,165408343
6	8,9	2	0,135471672
7	8,68	2,2	0,110973109
8	8,46	2,4	0,090929725
9	8,24	2,6	0,074537462
10	8,02	2,8	0,061138883
11	7,8	3	0,050196803
12	7,58	3,2	0,041272765
13	7,36	3,4	0,034009468
14	7,14	3,6	0,028116475
15	6,92	3,8	0,023358602

Figure 4.1: Load Data Table page in GPdotNET

There are several tab controls which separate information about model. GPdotNET shows several tabs when you load sample from Modeling and Prediction. The same set of tab controls GPdotNET load when you select New, and choose the first option for Modeling.

### 4.1. Load Data Tab

#### 4.1.1 Loading Discrete Data

The Load Data tab control (see Fig. 10) appears in every type of modeling. This is one of the main tabs. For every GPModel you need a data, to train your model. When you want to model common discrete data, Load Data Tab contains two buttons.

1. Button for loading training Data – with training data your GP model will be trained.
2. Button for loading testing data – with testing data your model will be tested, after model is calculated.

**Note:** You don't need to load testing data in order to make a GPdotNET model. Training data is used when you want to test calculated model.

Load Data	Functions	Settings	Run	Result	Info
Pos	X1	X2	Y		
1	10	1	0,367924841		
2	9,78	1,2	0,301250784		
3	9,56	1,4	0,246667457		
4	9,34	1,6	0,201984357		
5	9,12	1,8	0,165408343		
6	8,9	2	0,135471672		
7	8,68	2,2	0,110973109		
8	8,46	2,4	0,090929725		
9	8,24	2,6	0,074537462		
10	8,02	2,8	0,061138883		
11	7,8	3	0,050196803		
12	7,58	3,2	0,041272765		
13	7,36	3,4	0,034009468		
14	7,14	3,6	0,028116475		
15	6,92	3,8	0,023358602		

Training Data..
Prediction Data...

Figure 4.2: Load Data Table page in GPdotNET

### 4.1.2 Loading Time Series Data

With GPdotNET you can perform modeling with Time Series data as well. Run GPdotNET and click on MSFT Stock Quote. You opened Times series model. You can recognize that Load data is different than previous (see fig.)

Load Data	Functions	Settings	Run	Result	Info	Prediction
Pos	Y					
1	30,59					
2	29,19					
3	31,81					
4	32,05					
5	31,53					
6	29,15					
7	25,62					
8	25,25					
9	26,09					
10	24,38					
11	26,06					
12	26,67					

Settings

Nr.Series: 100

Nr. Variables: 10

Nr. Series for test: 5

Set to GP

Load Series...

Figure 4.3: GPdotNET Load Data tab in case of Time Series Modeling

Bottom part of Tab page, you can set number of variables to be as input variables, as well as if you want to define testing data to test calculated model. After you defined variables and Test data, in order to run, you need to press "**Set to GP**" button, that GPdotNET create training and testing data.

## 4.2. Function Tab

Function tab contains all available function in GPdotNET. Function defined here will defined function set, from which model is constructed.

There are two important columns which you can modify. Selected and Weight columns. When you want that certain function will be included in Function Set you need to check it. Weight column describe selection probability. From the picture above, + function has Weight=4, that means it has 3 times greater probability to be chosen, than subtraction function which has weight=1. On this way we can influence on probability of certain function in Function Set.

### 4.2.1. Changing Weight of the function

If you want to change weight of certain function do the following:

1. Click on certain function
2. Enter new weight value in text box on right side
3. Press button **Update row**.

Load Data

Functions

Settings

Run

Result

Info

Selected	Weight	Name	Definition	Arity	P..	Description	ExcelDefinition
<input checked="" type="checkbox"/>	4	+	$x_1+x_2$	2		Addition	$x_1+x_2$
<input checked="" type="checkbox"/>	1	-	$x_1-x_2$	2		Subtraction	$x_1-x_2$
<input checked="" type="checkbox"/>	3	*	$x_1*x_2$	2		Multiplication	$x_1*x_2$
<input checked="" type="checkbox"/>	1	/	$x_1/x_2$	2		Division	IF(ISNUMBER(...
<input type="checkbox"/>	1	Add3	$x_1+x_2+x_3$	3		Addition with with 3 arguments	$x_1+x_2+x_3$
<input type="checkbox"/>	1	Sub3	$x_1-x_2-x_3$	3		Subtraction with 3 arguments	$x_1-x_2-x_3$
<input type="checkbox"/>	1	Mul3	$x_1*x_2*x_3$	3		Multiplication with 3 arguments	$x_1*x_2*x_3$
<input type="checkbox"/>	1	Div3	$x_1/x_2/x_3$	3		Division with 3 arguments	IF(ISNUMBER(...
<input type="checkbox"/>	1	Add4	$x_1+x_2+x_3+x_4$	4		Addition with with 4 arguments	$x_1+x_2+x_3+x_4$
<input type="checkbox"/>	1	Sub4	$x_1-x_2-x_3-x_4$	4		Subtraction with 4 arguments	$x_1-x_2-x_3-x_4$
<input type="checkbox"/>	1	Mul4	$x_1*x_2*x_3*x_4$	4		Multiplication with 4 arguments	$x_1*x_2*x_3*x_4$
<input type="checkbox"/>	1	Div4	$x_1/x_2/x_3/x_4$	4		Division with 4 arguments	IF(ISNUMBER(...
<input type="checkbox"/>	1	$x^2$	$x_1^2$	1		x to the power of 2	power(x1;2)
<input type="checkbox"/>	1	$x^3$	$x_1^3$	1		x to the power of 3	power(x1;3)
<input type="checkbox"/>	1	$x^4$	$x_1^4$	1		x to the power of 4	power(x1;4)
<input type="checkbox"/>	1	$x^5$	$x_1^5$	1		x to the power of 5	power(x1;5)
<input type="checkbox"/>	1	$x^{1/3}$	$x_1^{1/3}$	1		Cube root	power(x1;1/3)
<input type="checkbox"/>	1	$x^{1/4}$	$x_1^{1/4}$	1		Quartic root	IF(ISNUMBER(...
<input type="checkbox"/>	1	$x^{1/5}$	$x_1^{1/5}$	1		Quintic root	POWER(x1;1/5)
<input type="checkbox"/>	1	1/x	$1/x_1$	1		Inverse	IF(x1=0;0;1/x1)
<input type="checkbox"/>	1	abs	$abs(x_1)$	1		Absolute value of x	abs(x1)

Weight:

Update row

Figure 4.4: GPdotNET Function Tab

### 4.3. Settings Tab

With Settings tab you can define parameters of genetic programming. GP parameters are Self-explained. If you don't know how to setup parameter, just live default values. It is suitable for most of problems.

The screenshot shows the 'Settings' tab in the GPdotNET application. The interface includes a top navigation bar with tabs: 'Load Data', 'Functions', 'Settings' (active), 'Run', 'Result', and 'Info'. The main settings area is divided into several sections:

- Population:** Size: 200 (50-5000), Fitness: RMSE-Root mean square error (dropdown), Initialization: HalfHalfInitialization (dropdown).
- Max Tree depth:** Initialize depth: 5 (3-17), Operation depth: 17 (3-17).
- Type of procesors:** Single core (radio button), Multy Core (radio button, selected).
- Selection:** Elitism: 1 (0-PopSize), Method: FitnessProportionateSelection (dropdown).
- Random constants:** From: 0, To: 10, Count: 6, and a 'Generate' button.
- Probability of gp operations:** Crossover: 0,9 (0,0 -1,00), Mutation: 0,05 (0,0 -1,00), Reproduction: 0,2 (0,0 -0,50).

Figure 4.5: Settings Tab in GPdotNET

Settings tab also contains option to enable parallel processing of some certain GP calculation. So if your PC has more than one processor you can enable this kind of processing.



#### 4.4. Run Tab

With Run Tab you control of GP modelling simulation, as well as defining Termination criteria of GP run. Run Tab contains two Chart controls for simulation Fitness values, and GP Model during evolution of the program.

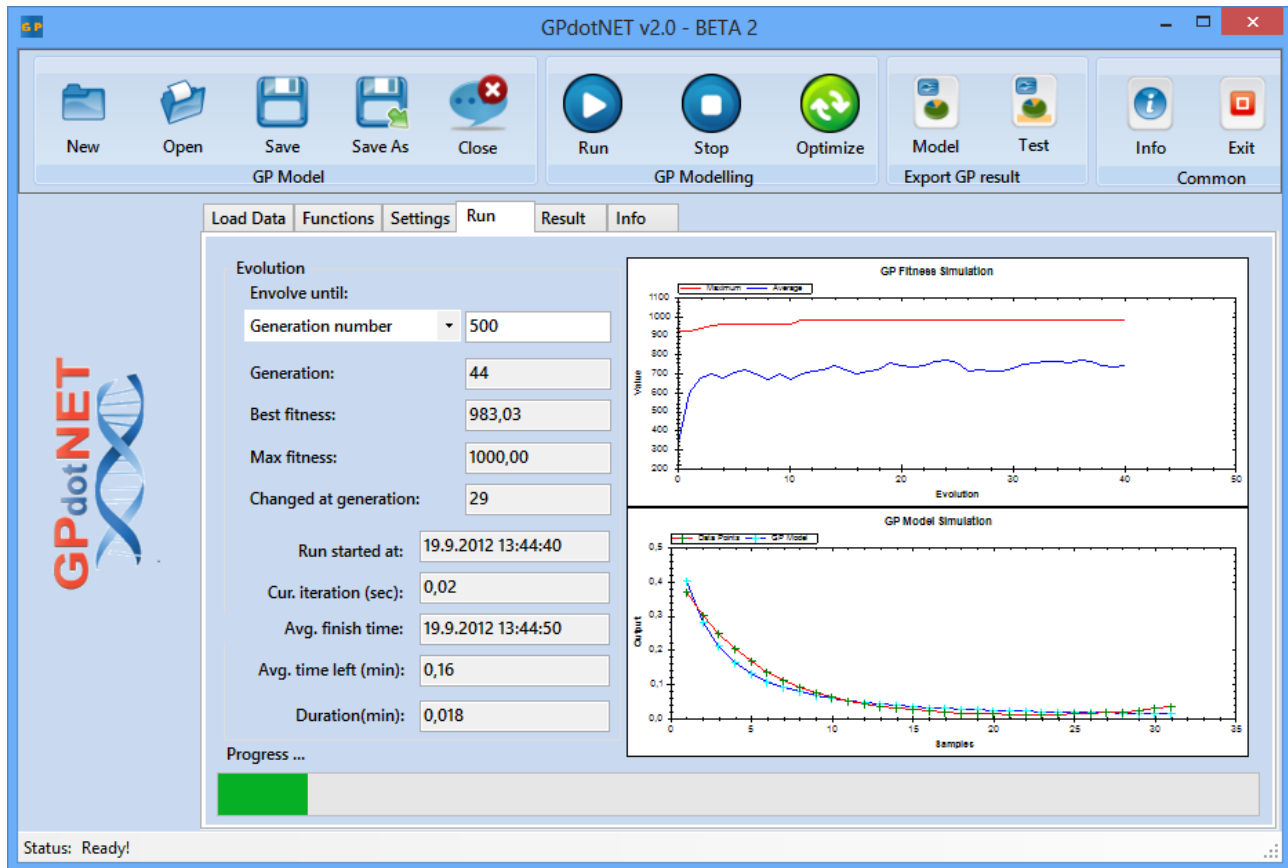


Figure 4.6: GPdotNET v2 Run Page

By using GP Modeling controls from the application toolbar you can control simulation and program running. In general there are 3 way to define termination criteria.

1. Number of evolution – when you chose the Generation number from the Combo Box
2. Fitness Exceed Value – when you choose **Fitness**  $\geq$  from Combo Box.
3. If you click on Stop toolbar button during program execution

#### 4.5. Setting Termination Criteria

Termination criteria can be set, when you choose one of the two predefined option in Combo Box. After you choose Combo Box option, you have to specify value in edit box.

You can change termination criteria whenever you want except during the program execution while the controls are disabled for editing.

## 4.6. Result Tab

Result tab show current best solution GPdotNET calculated. The main space of Result tab occupy Tree Draw expression which show best solution in expression tree. Below tree expression you can find solution in analytical form.

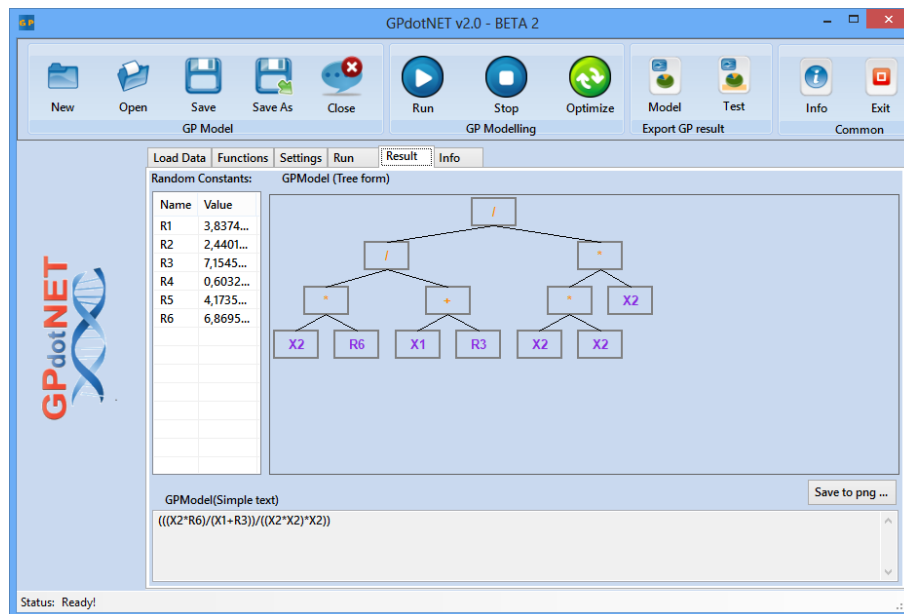


Figure 4.7: GPdotNET v2 Result Tab

Expression tree can also be saved in png image format.

## 4.7. Prediction Tab

Prediction Tab is shown when you load Testing Data. In any other case Prediction Tab will not be shown in GPdotNET environment. Picture 9 shows Prediction Tab, which contains Table and Chart of predicted data calculated with the current solution.

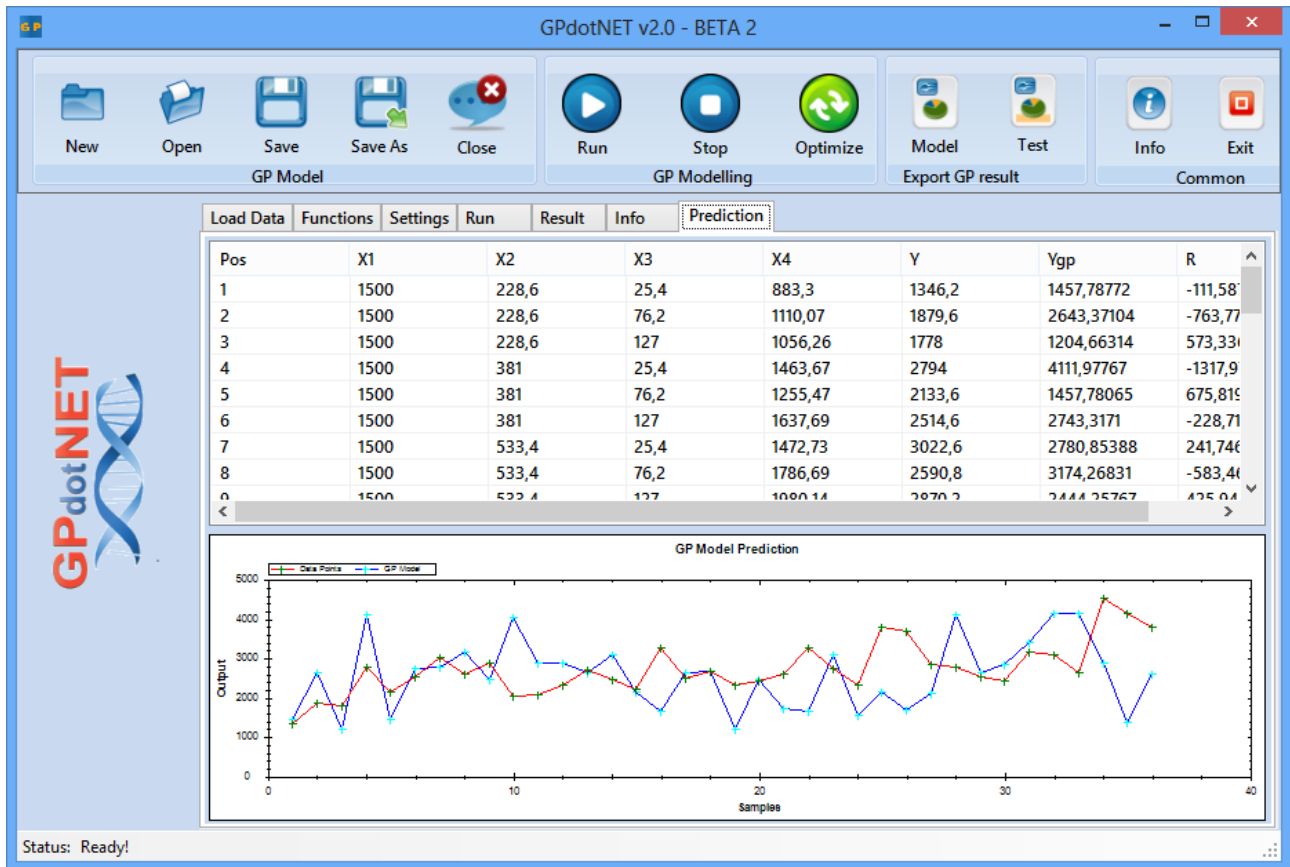


Figure 4.8: GPdotNET Prediction Tab

## 4.8. Info Tab

Info Tab is useful when you want to attach some information about problem you want to solve with GPdotNET. Info Tab contains Rich Edit control which you can load any rtf file format. By choosing "**Load RTF**" button command, you can load rtf file from disk.

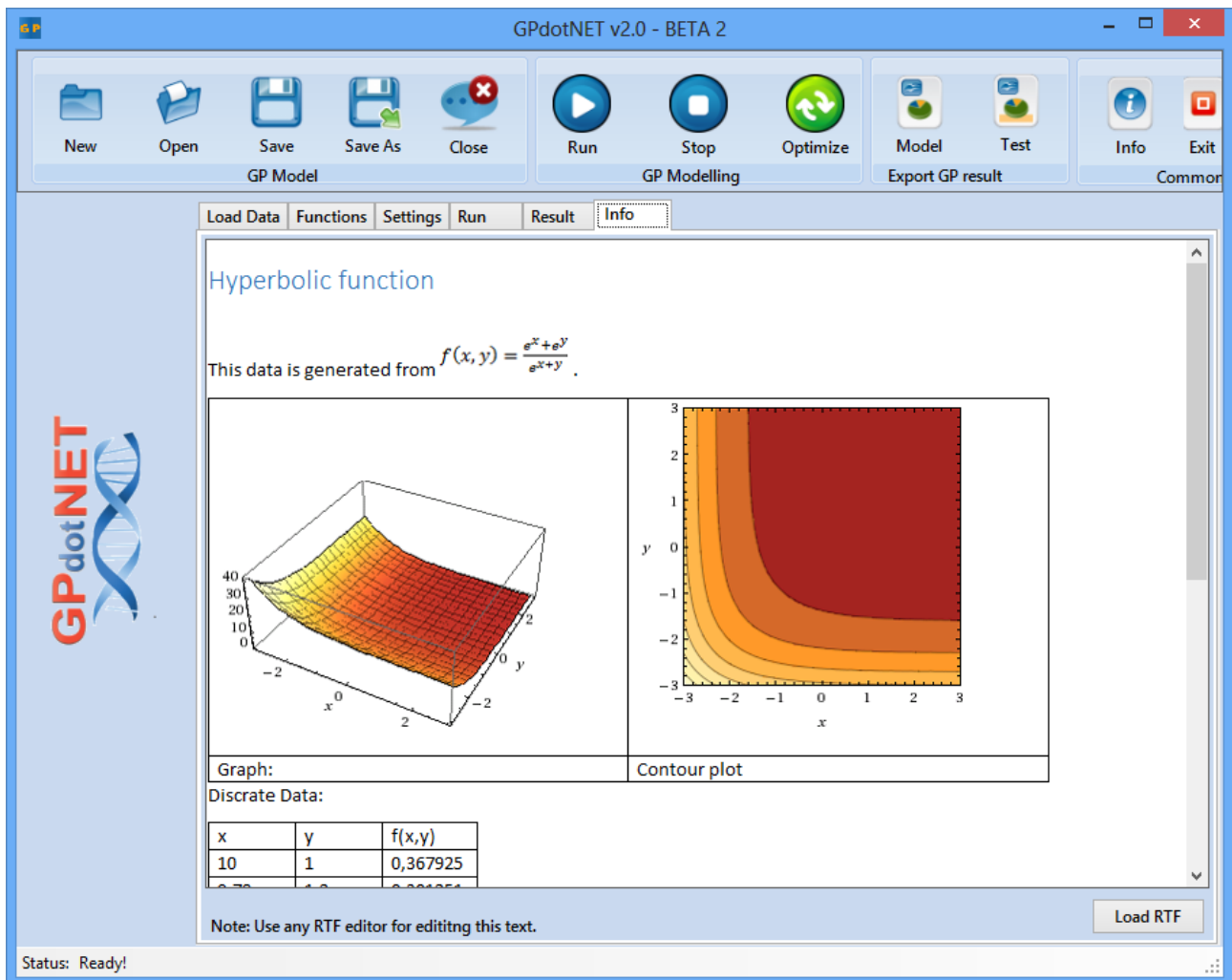


Figure 4.9: GPdotNET Info Tab

## 5. Working with Optimization

When you choose new use case called **Modeling&Optimization** additional tab will be shown called Optimization, in comparison of **Modelling/Prediction**.

Optimization is always performed after you got a good GP model. After GP Modeling is finished, you have to set boundaries of input variables, set termination criteria (similar as in previous Run Tab), and check Minimum check box if you want to find minimum value of the model. Unchecked means you are finding Maximum value.

### 5.1. How to Set Min/Max value of Variables

1. Select variable form bottom grid in Optimize Model Tab, by clicking left mouse button.
2. In Min and Max text box input values
3. Press Update button.
4. Select another variable and perform previous steps.

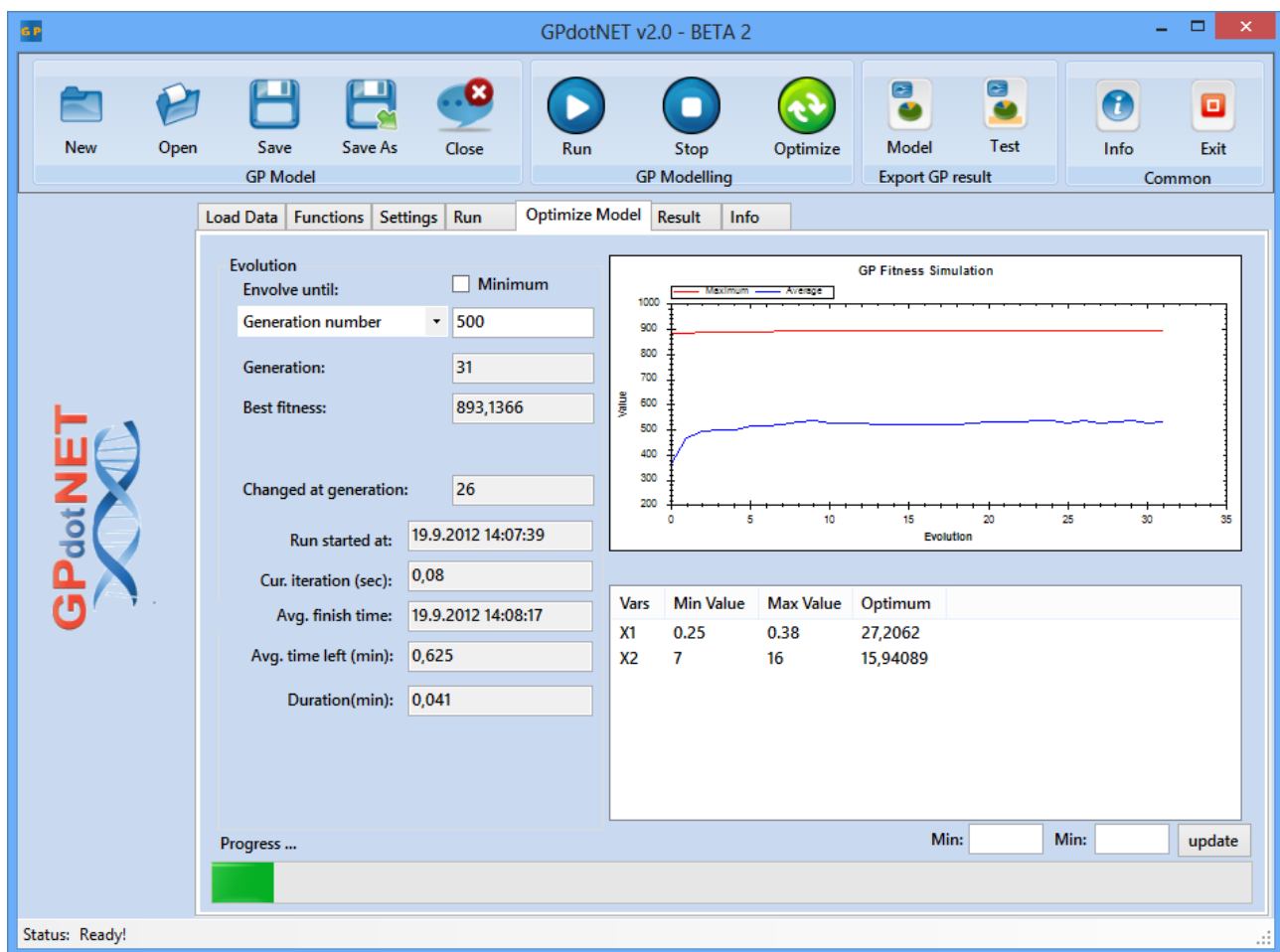


Figure 5.1: GPdotNET Optimize Model Tab

## 6. GPdotNET Optimization of analytic function

The main new feature among several other is Optimization based on Genetic Algorithm. You can either optimize GPModel calculated previously by using GP modeling, or also optimize analytically defined function defined with analytic function editor. The picture below shows sample of defining  $f(x) = x^3 - 6x^2 + 4x + 12$  in the analytic function editor. During construction of the function, right table is filled automatically with variables and constants. End of process of defining analytic function is finished when the Finish button is pressed to transfer variables and constants in to Optimization panel.

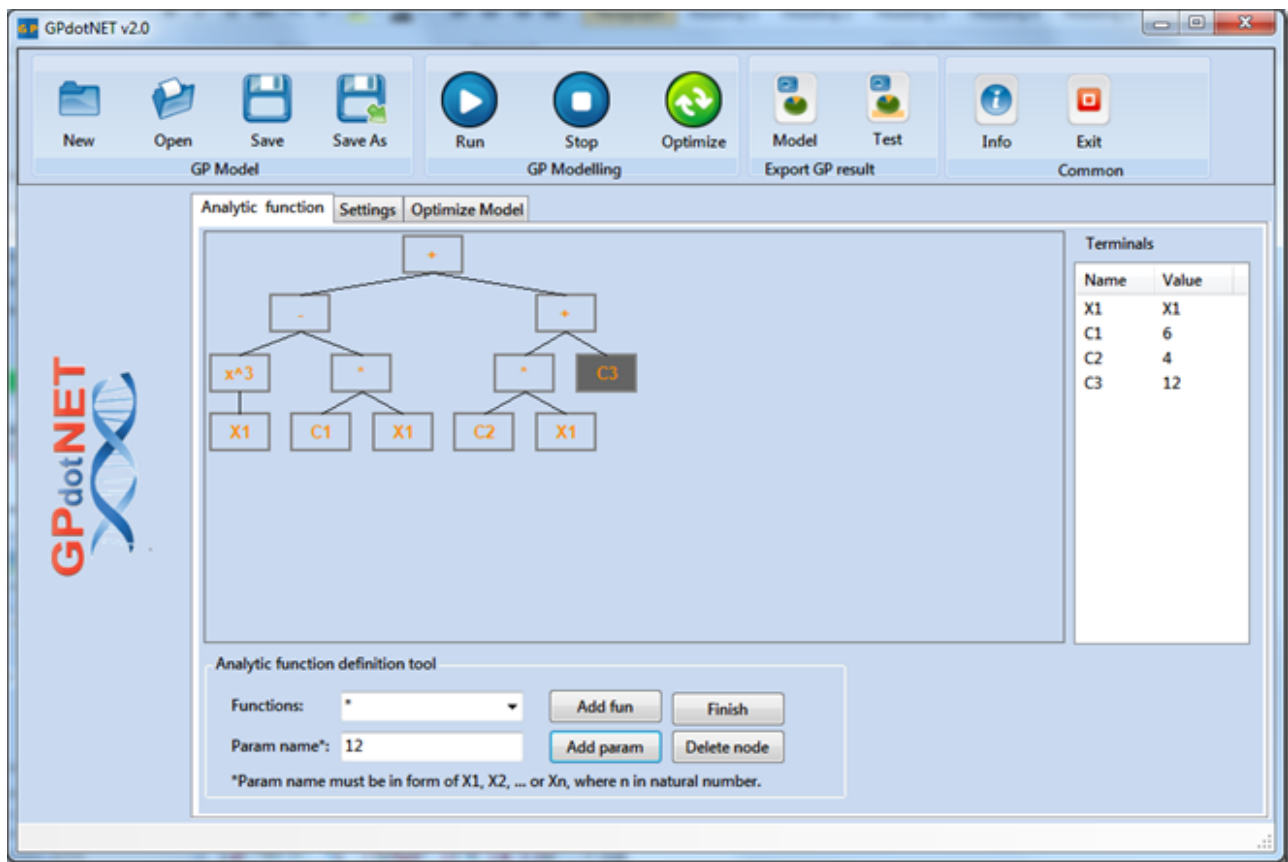


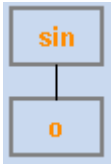
Figure 6.1: Analytic function editor in GPdotNET

After the Finish button is pressed, switch to Optimize model tab and define maximum and minimum values for input variables. The remaining process is the same as we have seen in previous chapter.

### 6.1. Working with Analytic function definition tool

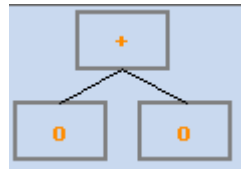
In addition with GPdotNET it will be possible to find global maximum or minimum of the function defined in analytic term as well. The picture below shows analytic tool definition. You can also see a brand new GUI for the GPdotNET v2.0.

Within Group Box (see Fig. 12) you can see several buttons and combo box for selection basic math function. For example if you select SIN function from the combo box and click on Add function button, in the central window two rectangles will appear similar like this:



The picture represents sin function with one argument. O argument means that it is not defined yet, and you will not go further until you specified the name of the argument, or define another function in chain. The tool automatically knows how many arguments need every defined function in GPdotNET.

If you select + or \* function you will get three node, one with function name and two for arguments. Similar like this picture:



### 6.1.1. Defining function arguments

Every function must define its argument in order to works correctly. Letter small o in the second rectangle means that the argument of function sin is not defined yet. To define argument select node with left mouse click, enter the name of the argument, and click on Add Param button. Whole process is depicted on picture below.

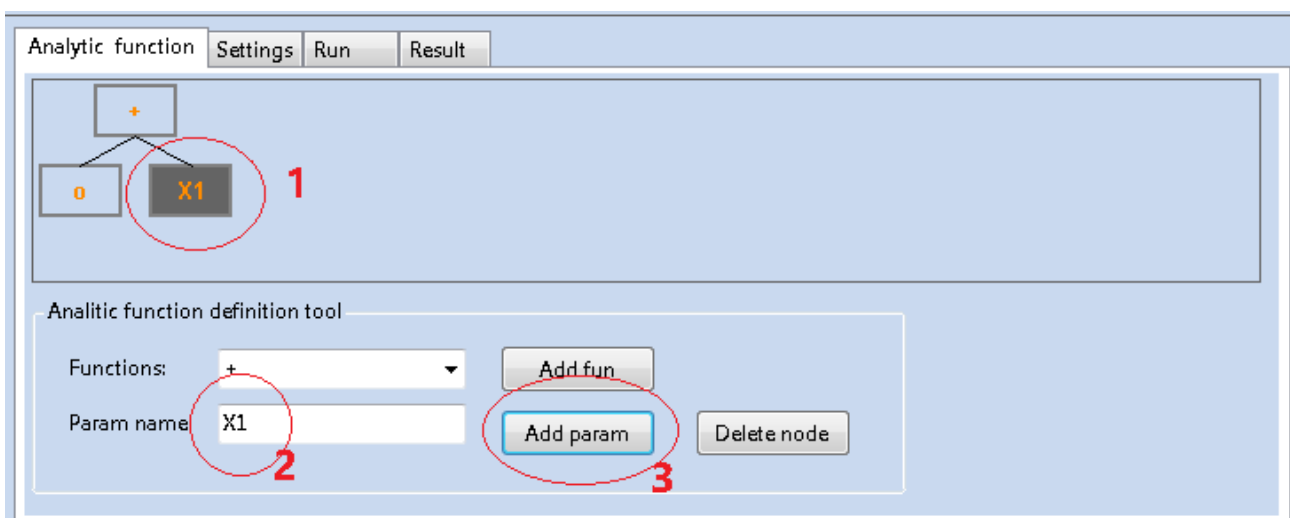


Figure 6.2: Adding function argument

### 6.1.2. Deleting the nodes in function

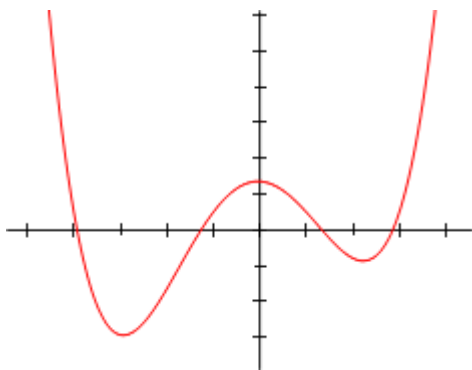
With Delete button you can delete nodes. In fact you can delete only leaf node. If you have bunch of nodes, and want to delete node in the middle, you need to delete all leaf node below it, in order to delete it. So select the leaf node with mouse and click on Delete node button.

After you finish the function definition, you must be sure that you defined all function nodes correctly with proper number of argument, otherwise GPdotNET will analyses it and cannot pass further until you correct the function argument issue.

### 6.1.3. How to optimize analytic function in GPdotNET

This is short tutorial how to use GPdotNET in order to find global optimum of analytically defined function.

Let's take an example of not so simple function. The picture below shows graph of the function. The function has two local minimum in interval from -5 to 5.



*Figure 6.3: Graph of polinomal function*

Analytic term of the function is:  $y = \frac{(x+4)(x+1)(x-1)(x-3)}{14} + 0.5$ .

To be sure that we are dealing with correct result, first find optimum from Wolfram Alpha. The picture below shows global optimum of our function.



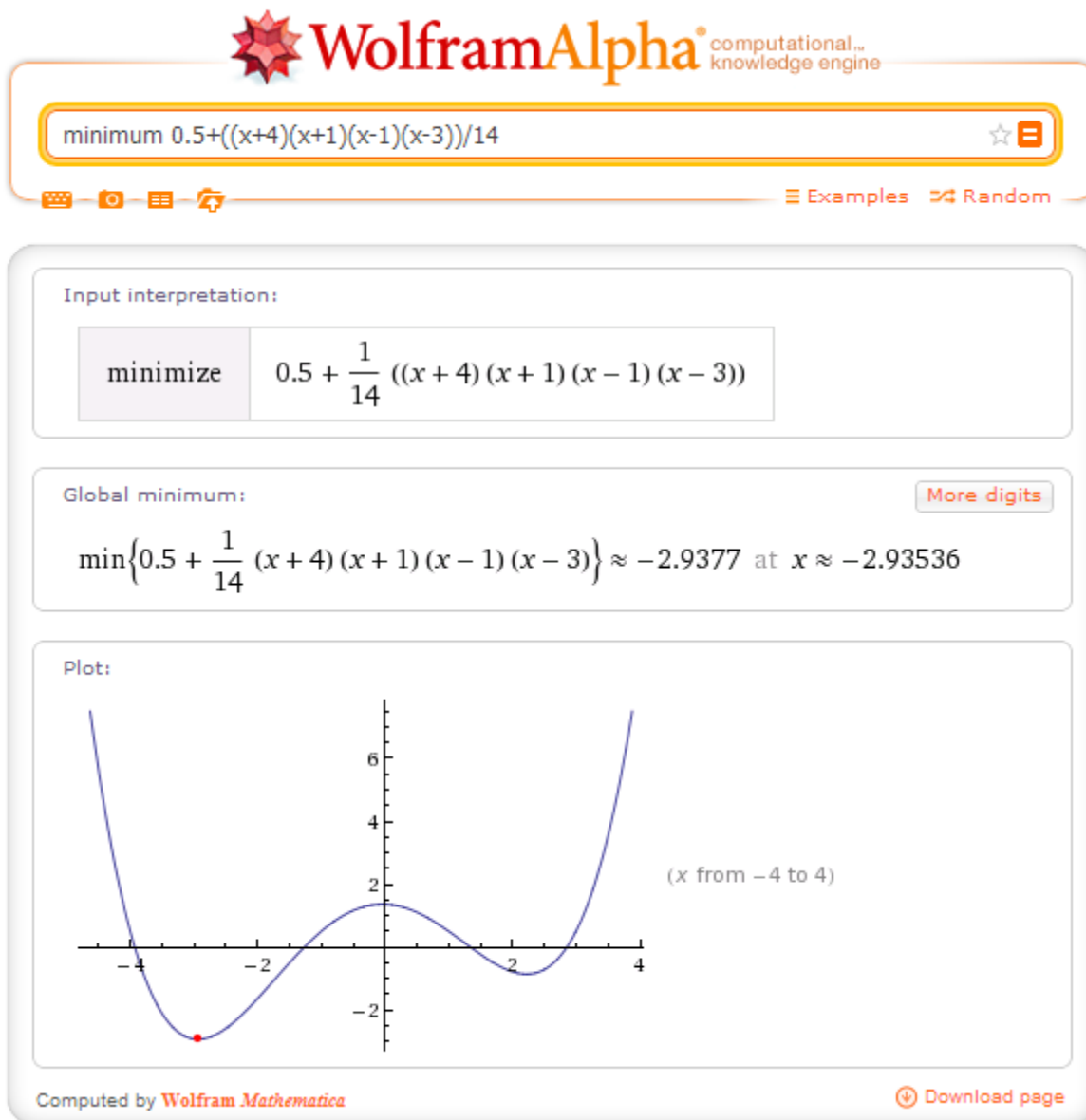
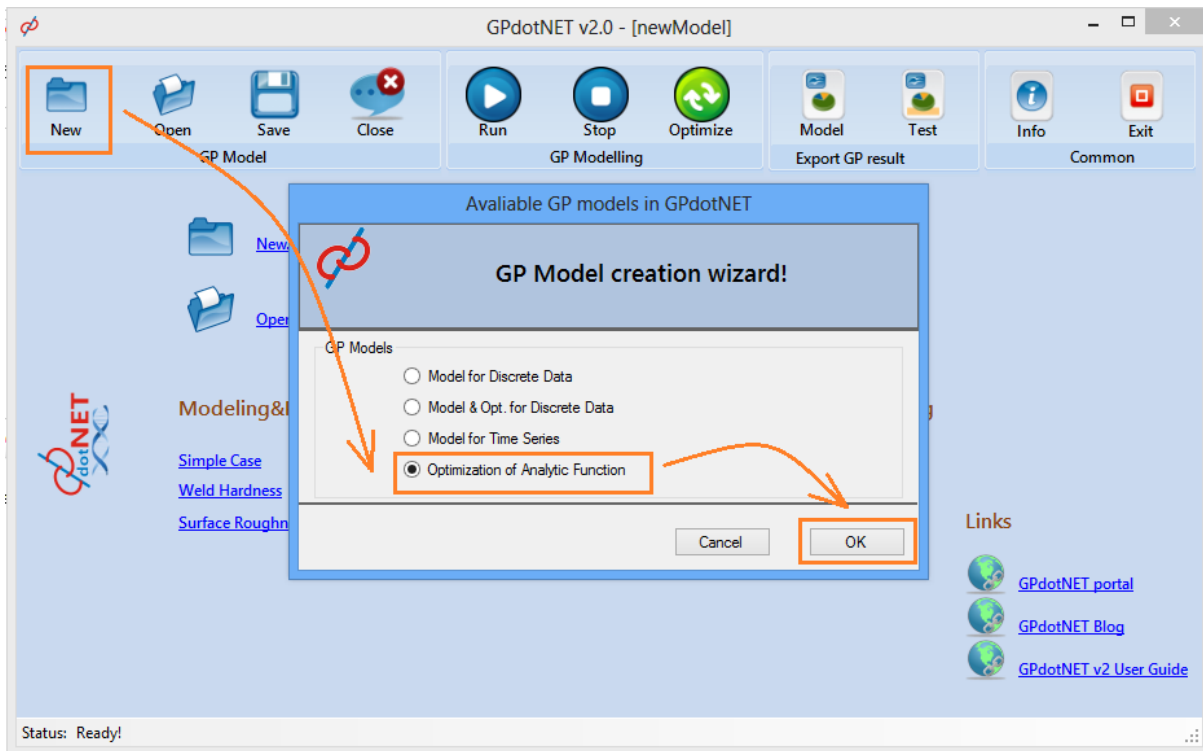


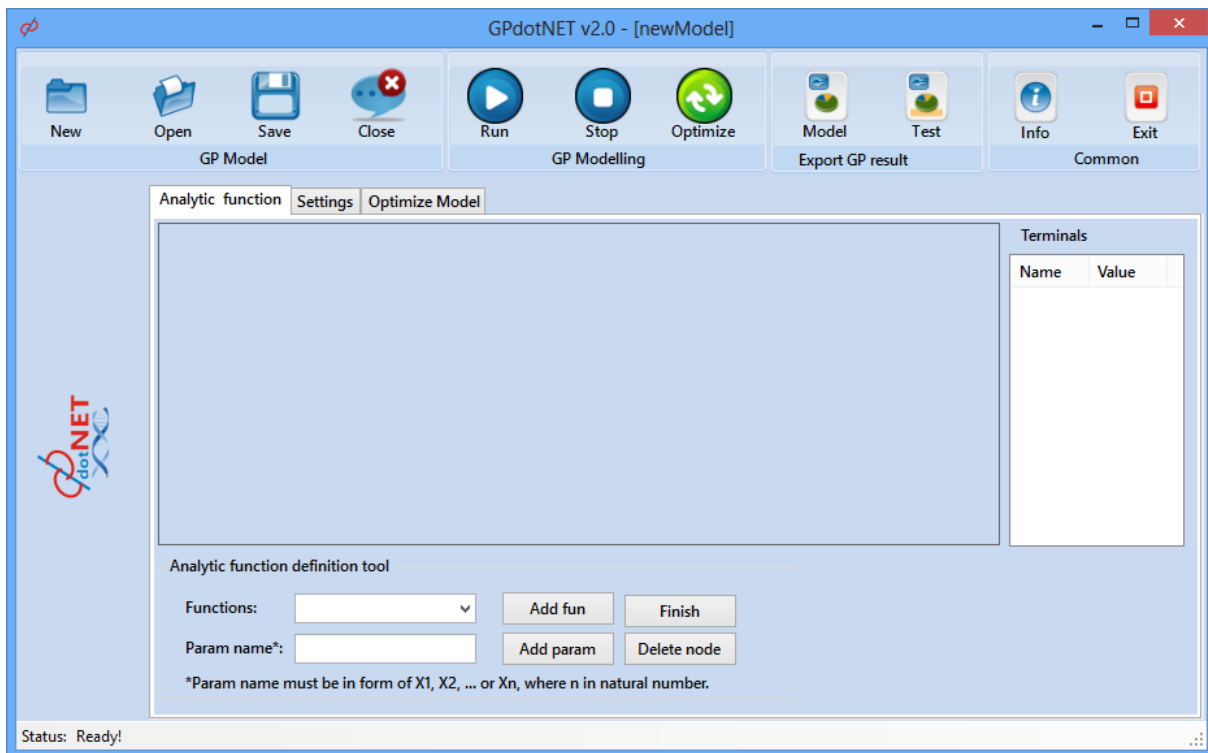
Figure 6.4: Wolfram Alpha info about function minimum of the function

From the above we see that global minimum can be found for  $X = -2.93536$ , and corresponded optimal value is  $y = -2.9377$ .

So let's try to find this optimum with GPdotNET v2.



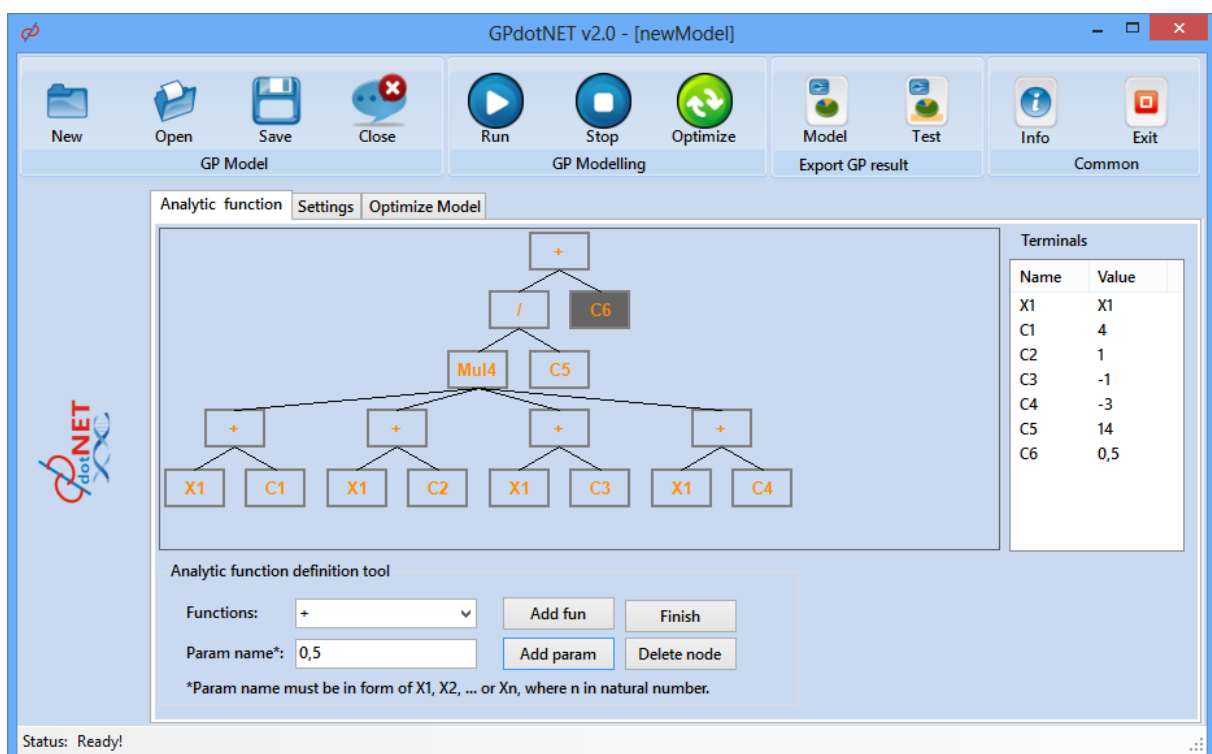
1. Open the GPdotNET and choose **New**.
2. Select **Optimization of Analytic Function** radio box
3. Click OK button.
4. Optimization module appears on the screen.



Now we need to define function in analytic form. As we have previously seen in blog post:

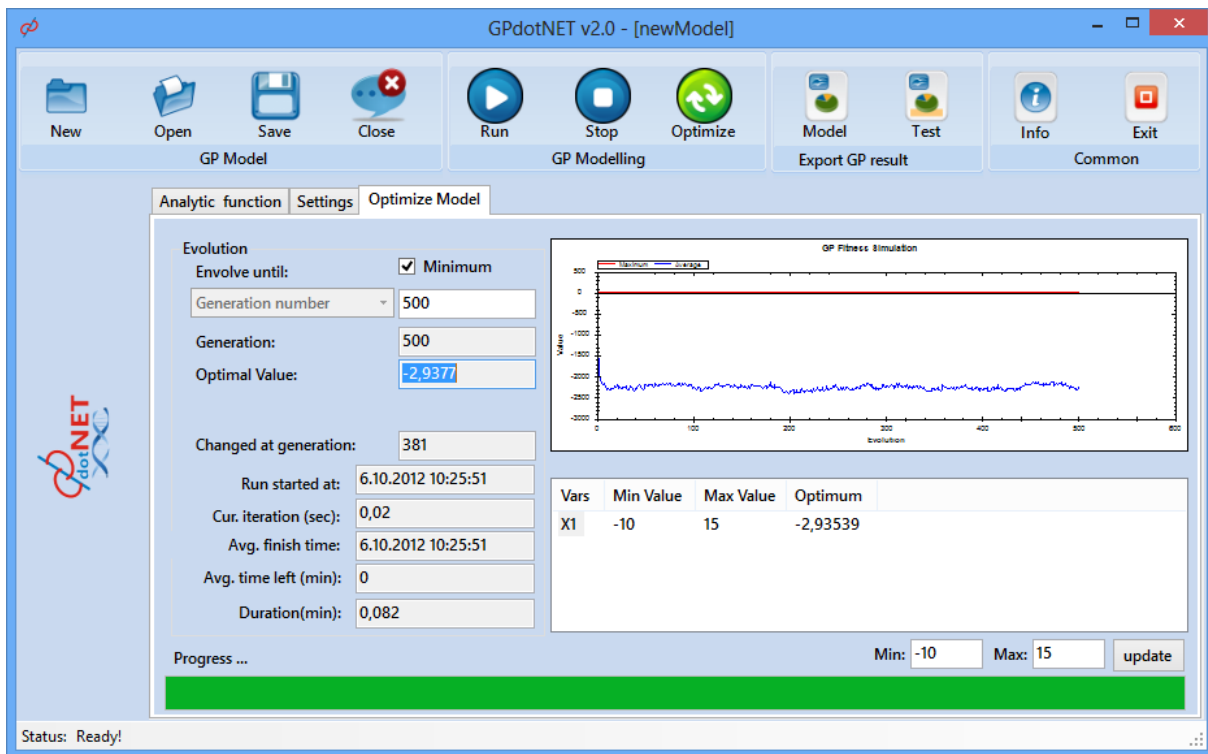
1. Select "+" function from Function combo box and press Add Function button.
2. Click on left outer node
3. Select "/" function and press Add Function button.
4. Select left outer node
5. Select **Mul4** function and press Add Function button.
6. For each outer node of Mult4 function add + function, similar like previous.
7. For each left outer node of + function, add Param Name **X1** and press **Add Param** button.
8. For right node add 4, 1,-1,-3 param from left to right.
9. For right node of Mul4 function add 14 param.
10. For right node of / function add 0, 5 param.

After you finish you get the following picture:



11. Press Finish button and select Optimize Tab page.
12. Select X1 variable in List View control for defining variable range

13. Enter -10 for min and 15 for maximum and press **update** button.
14. Select Minimize check box, for minimum.
15. Now we can perform optimization by press **Optimize** toolbar icon.
16. After very short period of time you get the following picture which



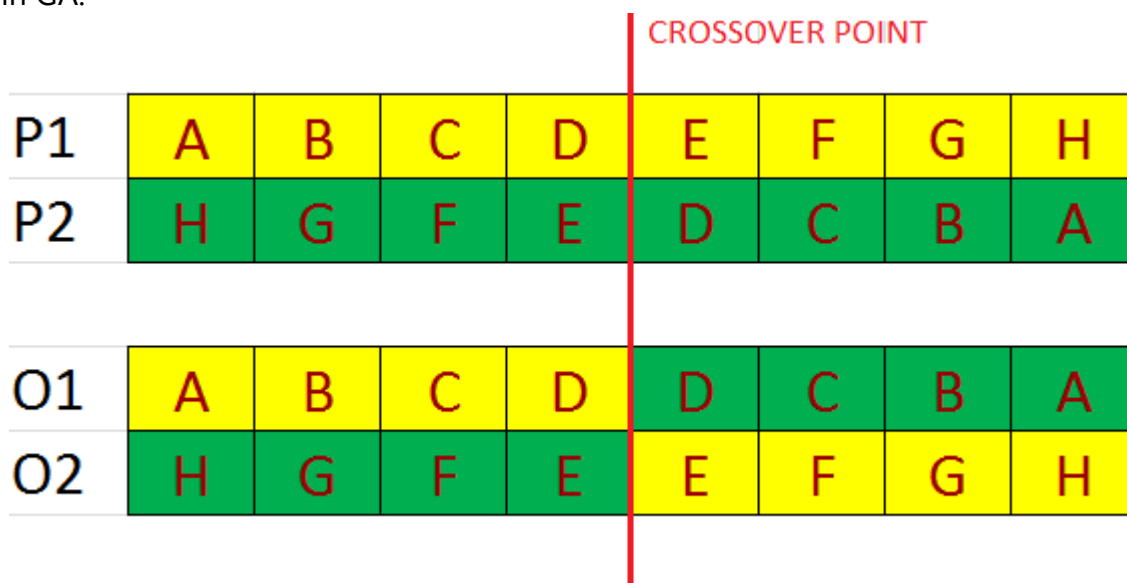
From picture above we can see that we got  $X_{\min} = -2,935$  and  $Y_{\min} = -2,9377$ , exactly as we got when we have performed optimization by Wolfram Alpha.

## 7. Solving TSP problem with GPdotNET

### 7.1. TSPChromosome class implementation

Instead of binary value, TSP chromosome must be array of integers by representing path around cities. The main problem with this chromosome type is genetic operations, because you cannot implement it easily.

For example standard Crossover in GA takes random point and split parents in two parts, see picture below. One part goes to first offspring, the second one goes to second offspring. The same action is performed for the second parent. The picture below shows standard crossover in GA.



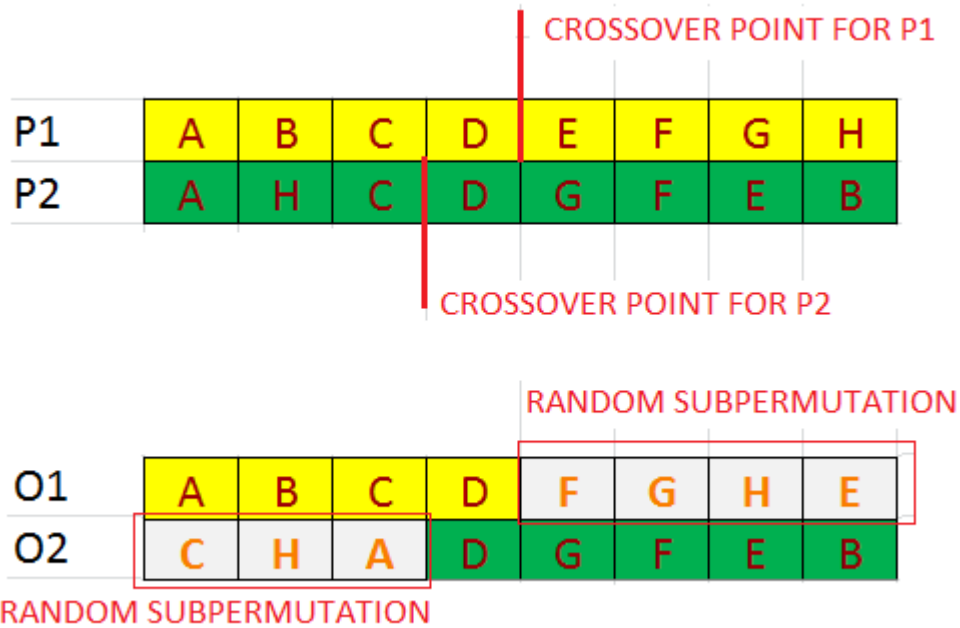
Unfortunately this operation cannot be applied to TSP, because every city can be visited only once. But how can we perform crossover at all? There are several methods can be found on internet. The method applied in GPdotNET is the following.

Choose different random points for Parent 1 and Parent 2.

**Offspring 1** is created based on the left part of Parent 1, and sub permutation of the rest of cities.

**Offspring 2** is created based on the right part of Parent 2 and sub permutation of rest of the cities.

Picture below show crossover used in GPdotNET.



The problem with this crossover is that offspring are not created based on genetic material from both parents.

## 7.2. Loading City Map

TSP Solver needs City data of points in (X;Y) format. It must be loaded by CSV file format, the same as GPdotNET uses for loading data for modelling and optimization. The following data represent correct format:

CityMap data format:

```

tsp_133.csv
1 0;13
2 0;26
3 0;27
4 0;39
5 2;0
6 5;13
7 5;19
8 5;25
9 5;31
10 5;37
11 5;43
12 5;8
13 8;0
14 9;10
15 10;10
16 11;10
17 12;10
18 12;5
19 15;13
20 15;19
21 15;25
22 15;31

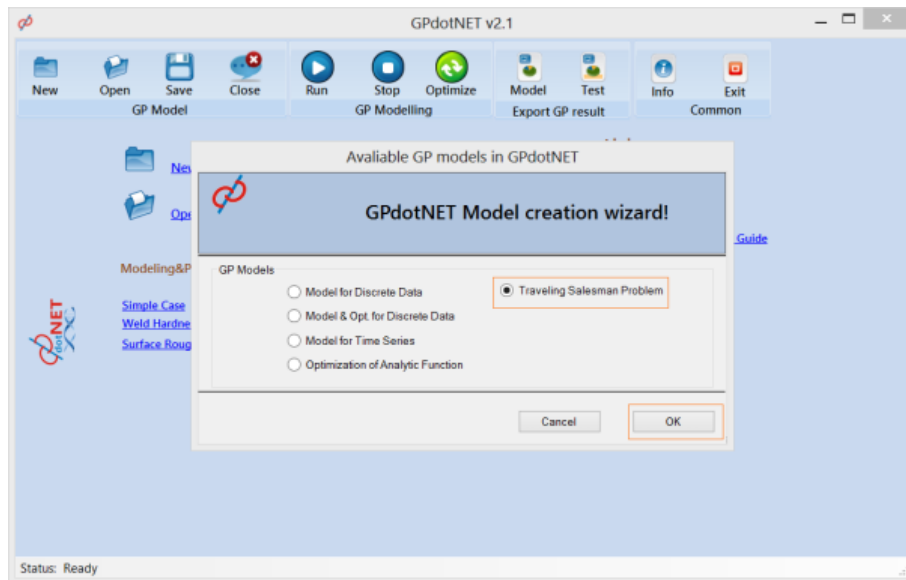
```

Active Server Pages script file length: 892 lines: 132 Ln: 58 Col: 6 Sel: 0 | 0 Dos/Windows ANSI as UTF-8 INS

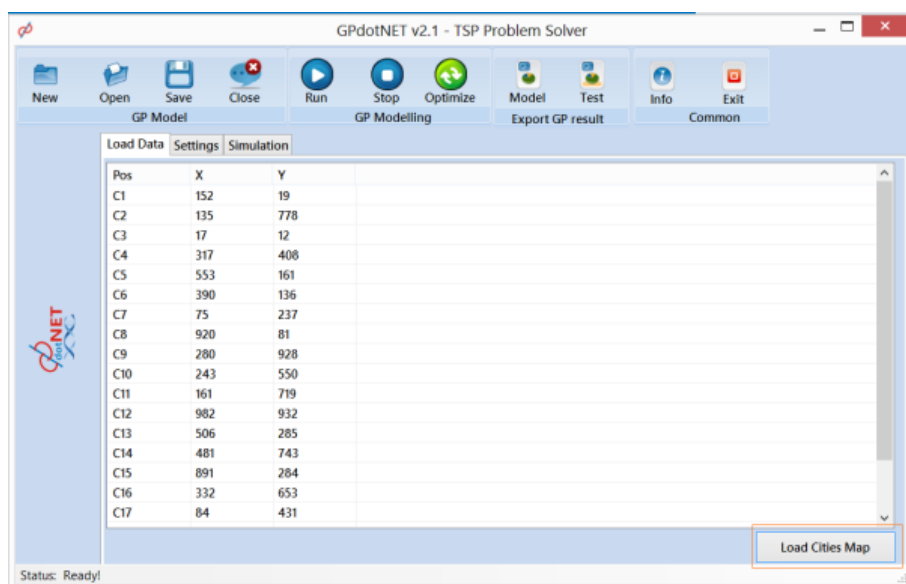
### 7.3. How to run TSP Solver on GPdotNET

Running TSP solver with GPdotNET is relatively easy. The procedure follows global convention for creating new model or solver.

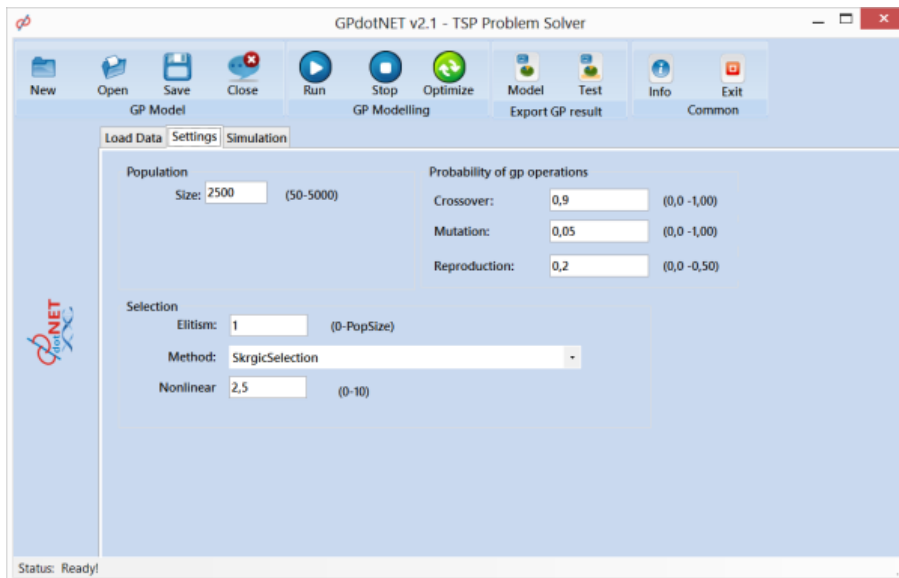
1. Choose **New** from Application Bar.



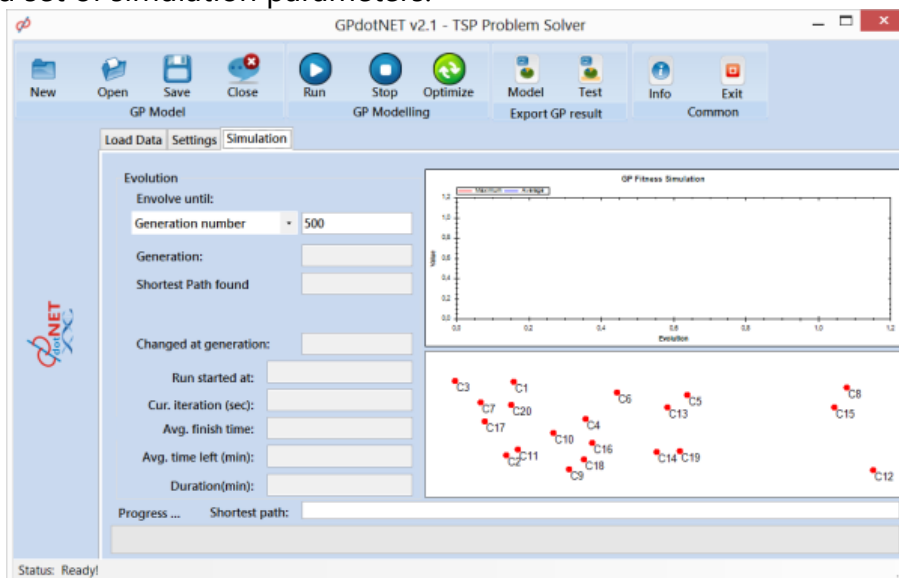
2. Choose **Traveling Salesman Problem**, and click OK button.
3. Open **Load Data** tab, (if it not opened), press **Load Cities Map** button, choose **CityMap.CSV**, and press OK. Data is presented on picture below.



4. Set GA parameters on Setting Tab panel:

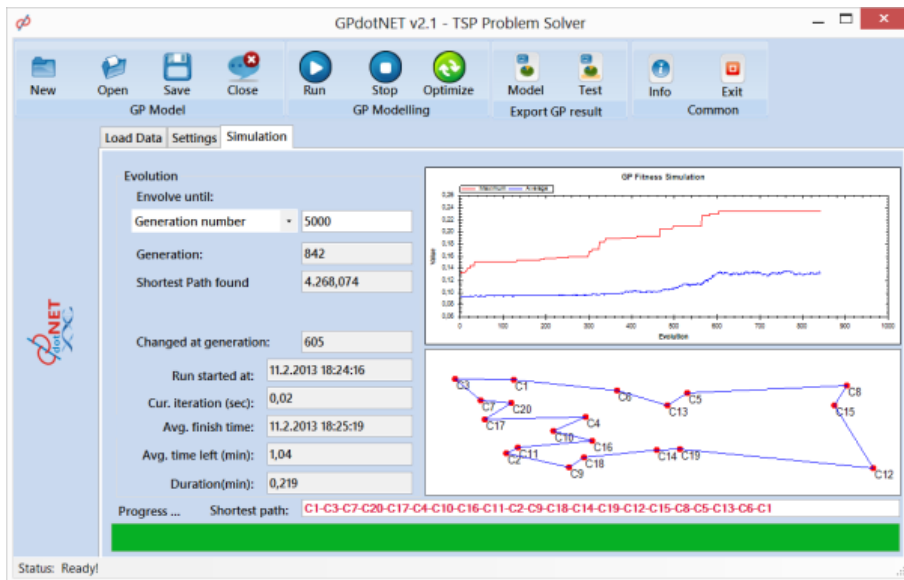


5. Open Simulation Tab panel. You can see graphical representation of loaded map, and set of simulation parameters.



6. Press **Run Application** button, and Solver will start. During simulation you can see that the path length is becoming shorter and shorter
7. As final result, Solver found the path which is the shortest or very close to shorter one.





### 8. Exit and Save your solution.

We have seen how easy to implement such a problem based on existing foundation of the GPdotNET.

## 8. Solving Assignment Problems (AP) in GPdotNET

### 8.1. Introduction to AP

AP deal with the question how to assign  $n$  objects to  $m$  other objects in the best possible way. Best possible way can be represent optimal (with minimum or maximum) value of the objective function. Most of the APs have  $m > n$  which means that assignment operation deals with not equally numbers of objects. All such a problems can be transformed to  $m = n$ , by adding fake assignment operations.

Ona example of AP can be formulate as follow: *Suppose that a taxi firm has three taxis (the agents) available, and three customers (the tasks) wishing to be picked up as soon as possible. The firm prides itself on speedy pickups, so for each taxi the "cost" of picking up a particular customer will depend on the time taken for the taxi to reach the pickup point. The solution to the assignment problem will be whichever combination of taxis and customers results in the least total cost.*

*However, the assignment problem can be made rather more flexible than it first appears. In the above example, suppose that there are four taxis available, but still only three customers. Then a fourth dummy task can be invented, perhaps called "sitting still doing nothing", with a cost of 0 for the taxi assigned to it. The assignment problem can then be solved in the usual way and still give the best solution to the problem.*

*Similar tricks can be played in order to allow more tasks than agents, tasks to which multiple agents must be assigned (for instance, a group of more customers than will fit in one taxi), or maximizing profit rather than minimizing cost.*

The formal definition of the assignment problem (or linear assignment problem) is

*Given two sets,  $A$  and  $T$ , of equal size, together with a weight function  $C: A \times T \rightarrow R$ . Find a bijection  $f: A \rightarrow T$  such that the cost function:*

$$\sum_{a \in A} C(a, f(a))$$

*is minimized.*

Usually the weight function is viewed as a square real-valued matrix  $C$ , so that the cost function is written down as:

$$\sum_{i \in A} \sum_{j \in T} C(i, j) x_{ij},$$

*Subject to the constants*

$$\sum_{j \in T} x_{ij} = 1, \text{ for } i \in A$$

$$\sum_{i \in A} x_{ij} = 1, \text{ for } j \in T$$

$$x_{ij} \geq 0, \text{ for } i, j \in A, T$$

The variable  $x_{ij}$  represents the assignment of agent  $i$  to task  $j$ , taking value 1 if the assignment is done and 0 otherwise. This formulation allows also fractional variable values, but there is always an optimal solution where the variables take integer values. This is because the constraint matrix is totally unimodular. The first constraint requires that every agent is assigned to exactly one task, and the second constraint requires that every task is assigned exactly one agent.

## 8.2 AP in GPdotNET

Solving AP with GPdotNET begins by clicking “**New**” button from Application Bar, and selecting AP Solver from New GPMModel dialog (see pic. below).

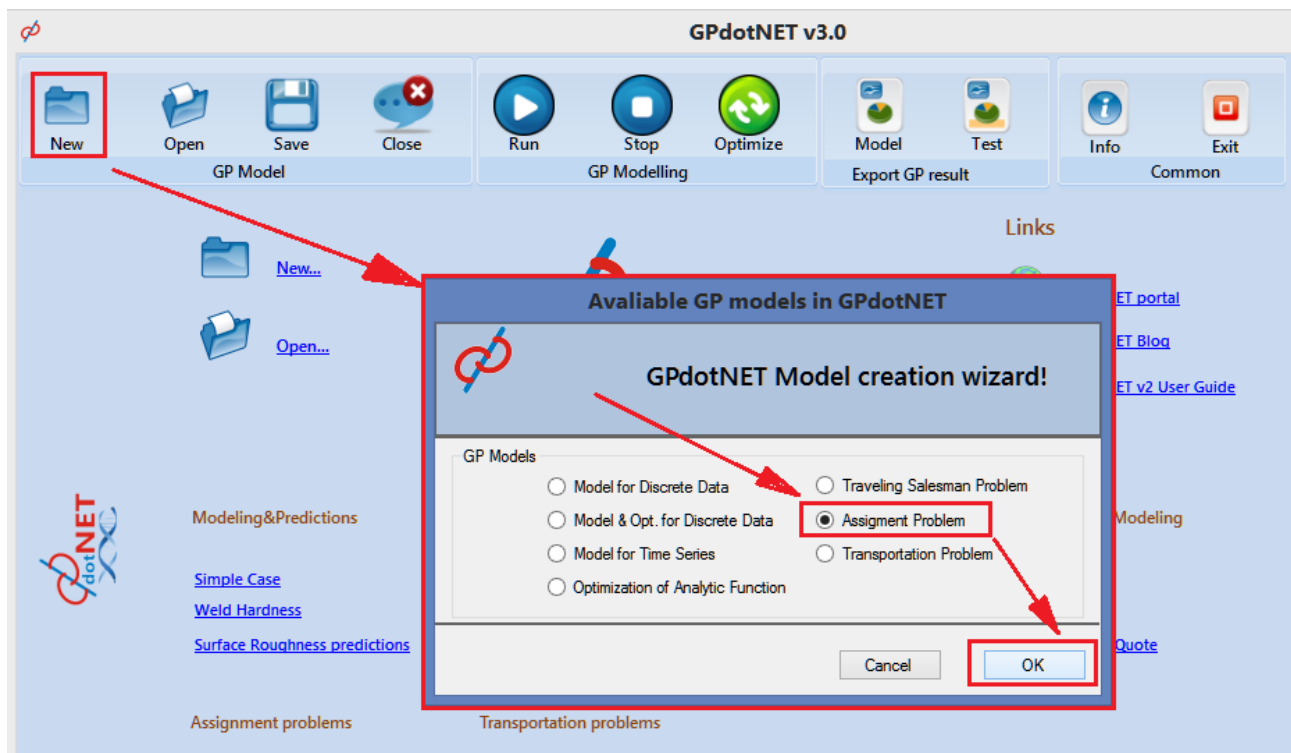


Figure 8.1: Creating new AP Model in GPdotNET

After clicking OK button, GPdotNET shows several tab pages similar on the following picture.

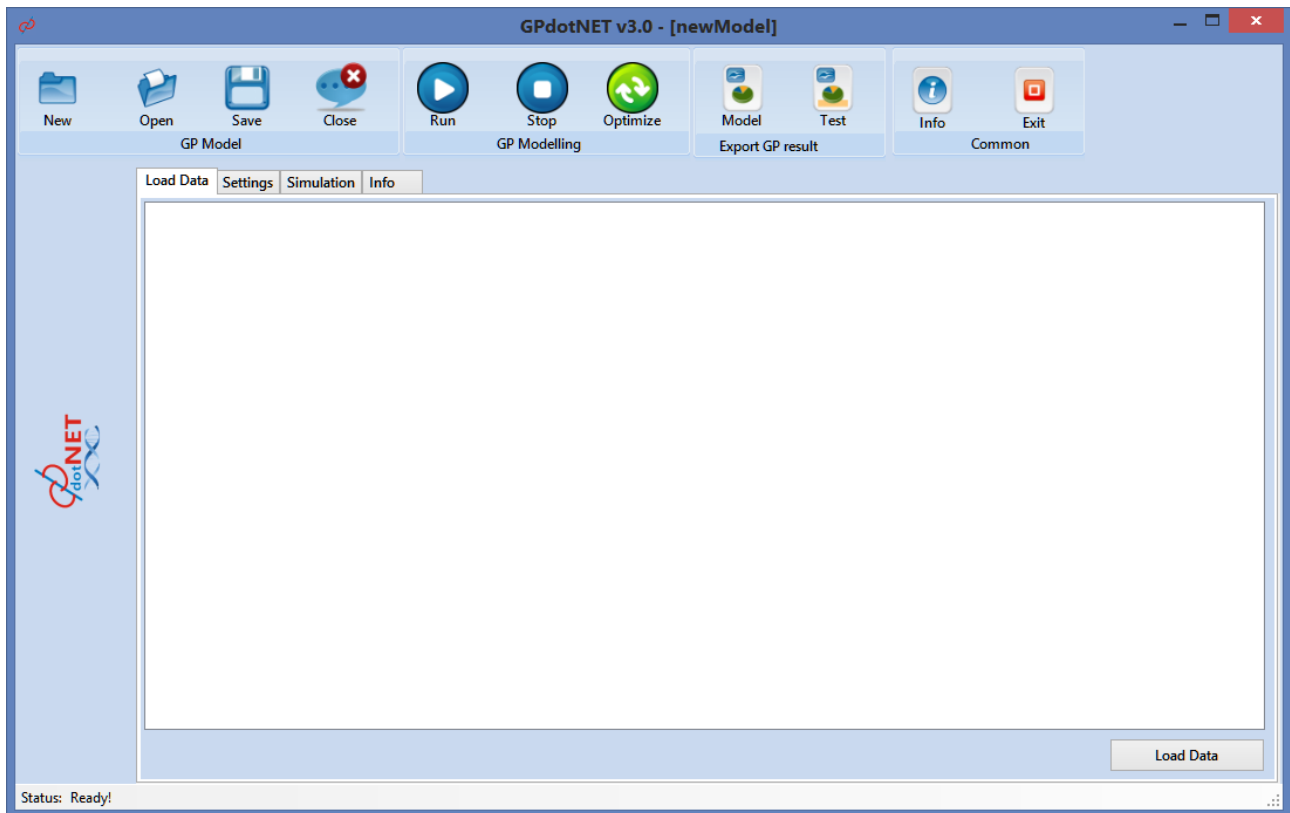


Figure 8.1 Load Data Tab in when modeing AP in GPdotNET

The first step in defining AP in GPdotNET is loading data of the problem.

### 8.2.1 Preparing and loading data in AP

As we can see from introduction data for AP problem must be in matrix form, so the following picture can represent proper format of AP data.

```
!4 locations
!4 cars
!W F1 F2 F3 F4
26;28;34;34;0
28;M;36;34;0
20;28;30;28;0
30;32;36;30;0
24;30;32;32;0
```

In GPdotNET v3 and late you can put comments in data file. The picture above contains three lines of comments which will not be loaded. Then we have 5 lines represent rows in AP. Columns are divided by semicolon. From the picture above it can be seen "M" letter in the middle of the data. "M" represent maximum number (infinity number). Other symbol which can be seen in data is "L" which represents the minimum value (negative infinity). Decision when to put M or L letter depends of the solution.

1. If we want to find minimum cost of AP and want to put fake row in order to define closed solution, we can put **M** letter.
2. If we want to find maximum cost of AP and want to put fake row in order to define closed solution, we can put **L** letter.

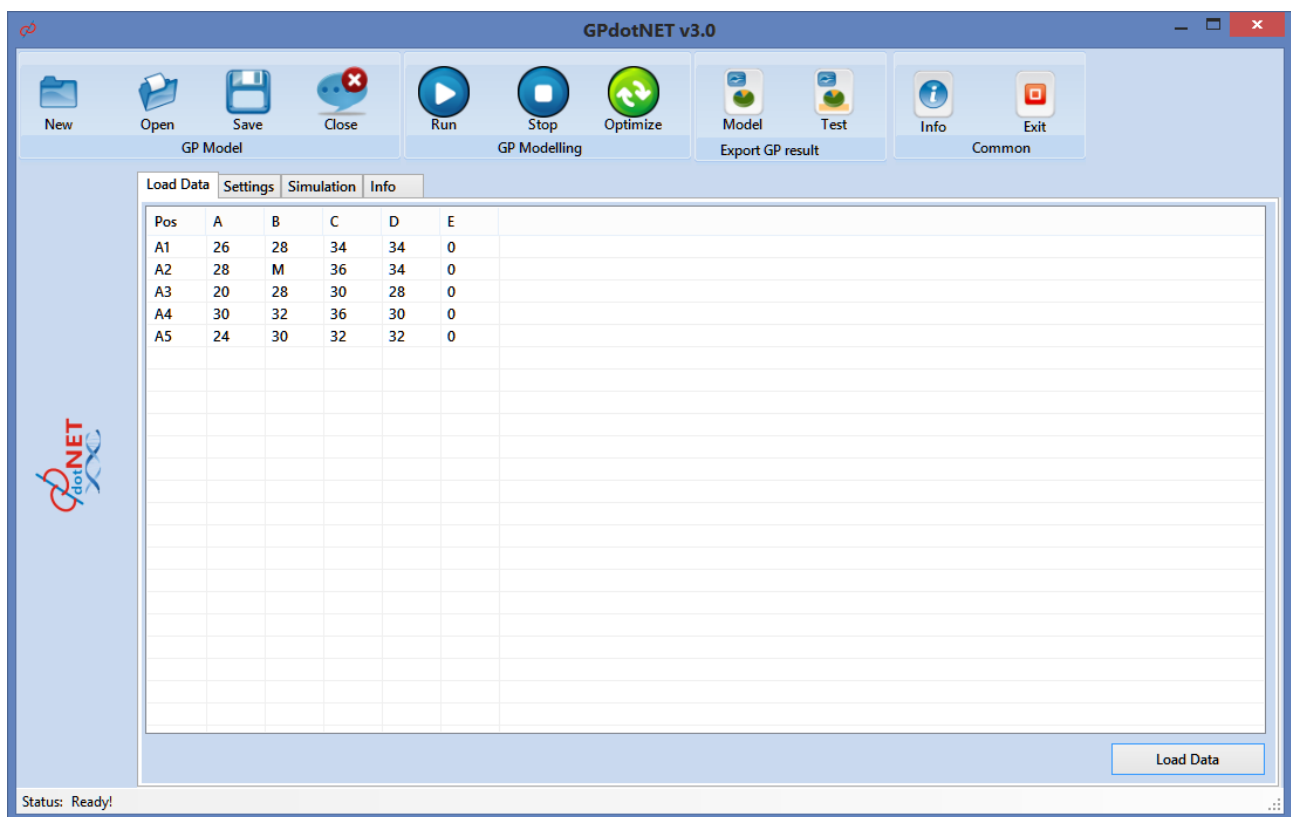


Figure 8.2: Loaded data in AP model

### 8.3. Running AP problem with GPdotNET

When the data is loaded we can start by defining GP parameters and start with finding solution. The process is identical like other, so this step we will not be describe. After we start finding solution by clicking START button, we can see that GPdotNET find solution very quickly. The picture below show solution for problem we have described above.

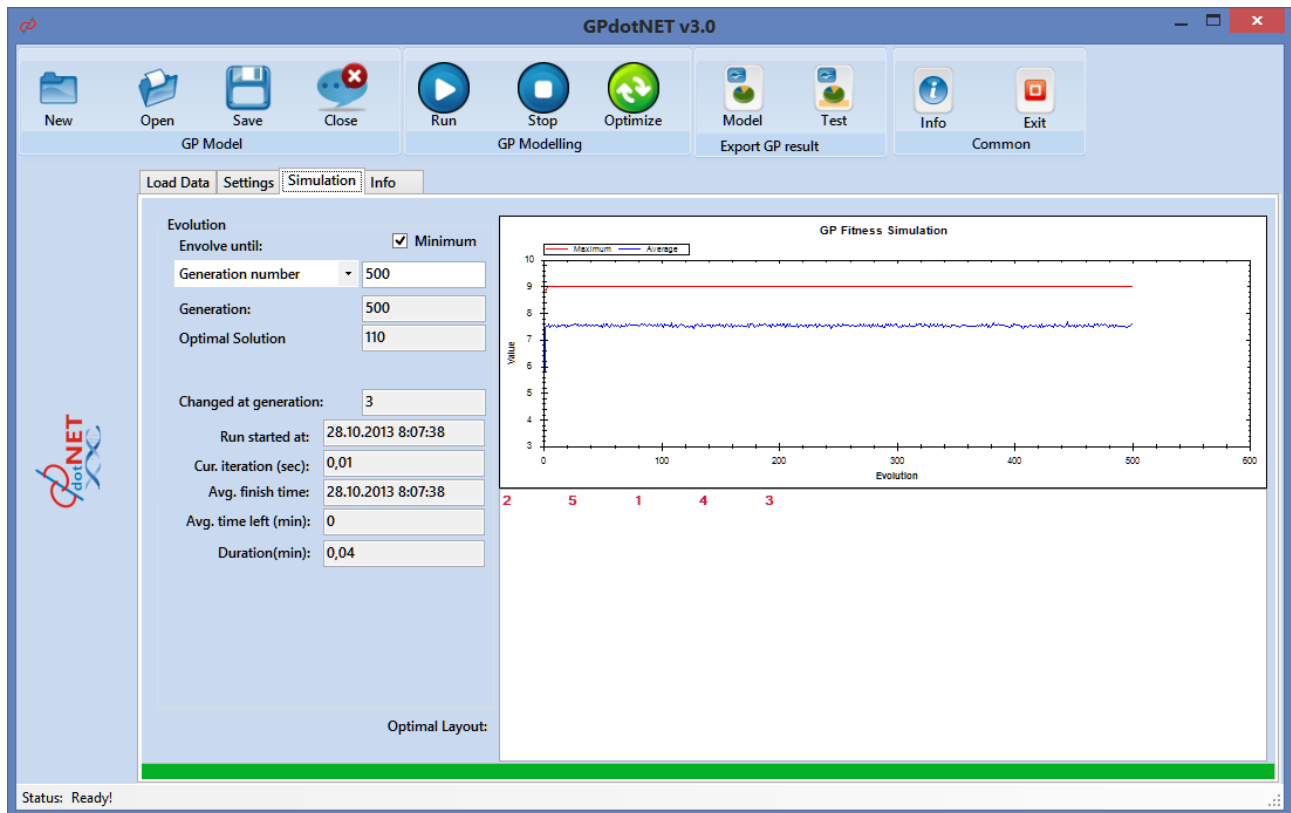


Figure 8.3: Empty TP Model in GPdotNET

As we can see from the picture above, result is shown as array of number. Actually, result is presented as vector, which position of number represent column, and the value represent the row of started problem. From the picture above result is **[2 5 1 4 3]** which means the following results:

```
!4 locations
!4 cars
!W F1 F2 F3 F4
26;28;34;34;0
28;M;36;34;0
20;28;30;28;0
30;32;36;30;0
24;30;32;32;0
```

Vector number represent the row index, and position means column index.

## 9. Solving Transportation problems in GPdotNET

Solving TP with GPdotNET begins by clicking “**New**” button from Application Bar, and selecting TP Solver from New GPMModel dialog (see pic. below).

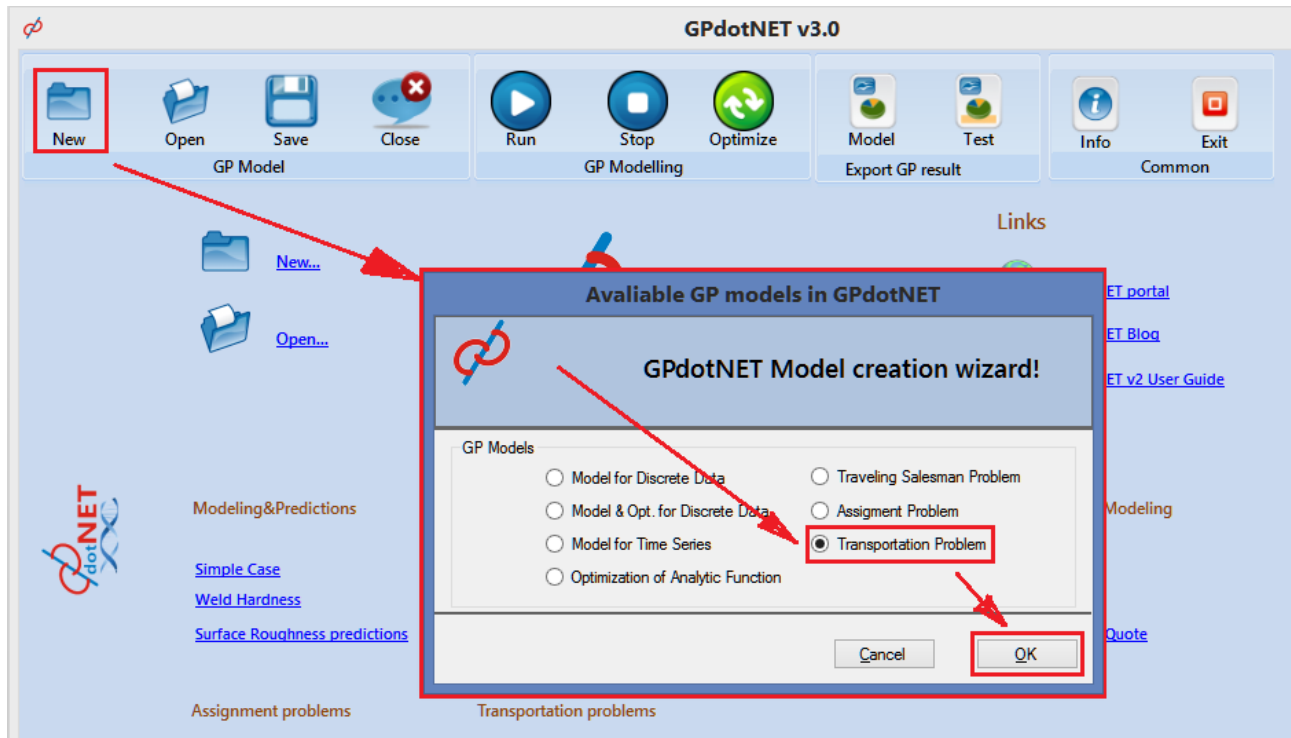


Figure 9.1: Creating new TP Model in GPdotNET

The process of defining TP model is like previous one. So first step is define training data. Training data can be presented as table of values imilar like picture shows below.

```
!4 locations
!4 cars
!W F1 F2 F3 F4
26;28;34;34;0
28;M;36;34;0
20;28;30;28;0
30;32;36;30;0
24;30;32;32;0
```

The AP problem can contains different number of cols and rows, which leads to define

additional col or rows in order to define layout GPdotNET expects. With the same procedure we can define fake col or rows n AP to define compatible problem GPdotNET supports. The symbols M and L represent plus infinity or minus infinity the same s in AP.

The Settings Tab page is the same as in previous problems so no need to additional information.

Simulation tab simulate problem searching and shows result. Unlike from AP problem here we have result in Matrix form.

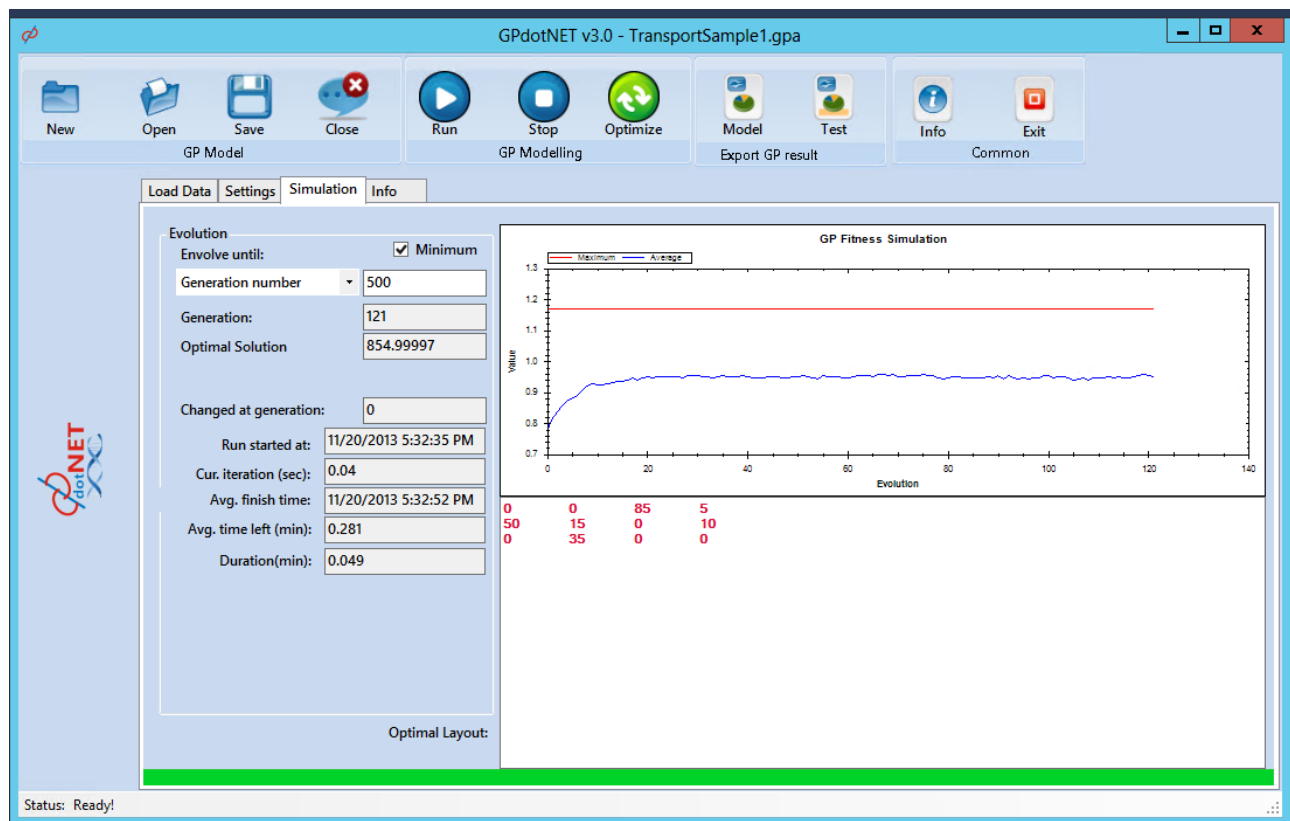


Figure 9.2: Result preentatiom in Simulstion Tab Page

The result represent the optimal transportation solution.



## 10. GPdotNET File format

GPdotNET v2.0 introduce new textual file format for persisting GP and GA models and information during the program run. The file format is simple and easy, so sometimes you can edit in order to correct some minor changes. For example if you create GPModel without optimization, but later on you need to optimize this mode, you can easily change the type of model to enable optimization. The picture below shows sample GPdotNET "gpa" file format opened in Notepad++ with syntax highlighting.

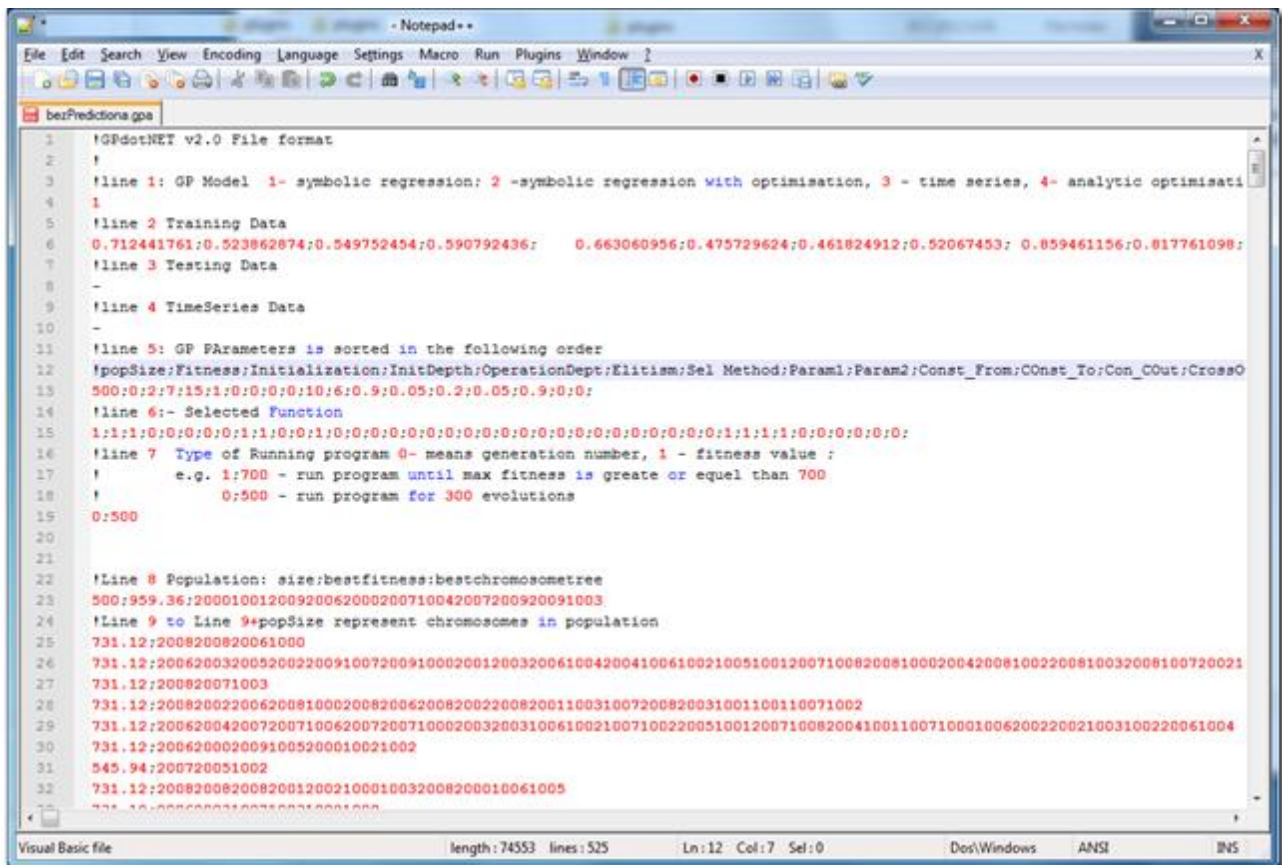


Figure 10.1: GPdotNET file format highlighted in Notepad ++

## 11. Exporting Results in GPdotNET

### 11.1. Exporting to Excel and

When the model is calculated, you can export training and testing data in to Excel (only for Windows user) and csv file format.

When you export to Excel you can export GPModel in form of Excel formula, for further analysis.

Choose model icon if you want for export training data and GPModel. Otherwise choose Test icon for exporting testing data.

After you choose right export icon, export dialog appears:

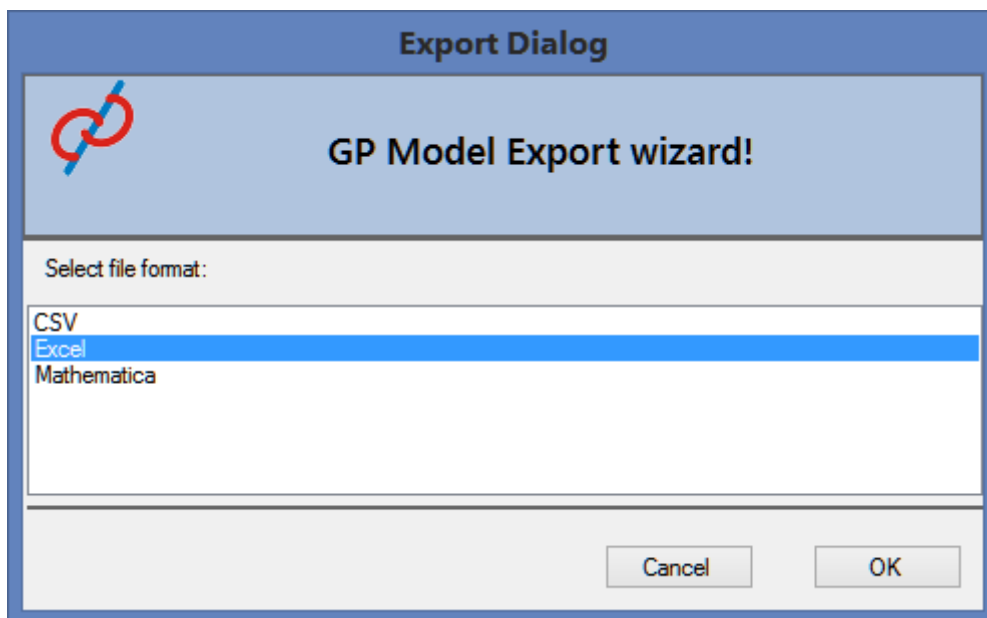


Figure 11.1: Export dialog

After you select right export format, "Save dialog" appears, to choosing right file name and path.

In case of Excel exporting GP Model column is as text showing formula. You need to put equal sign in front of content in order that model be calculated. The picture below show similar case.

The reason why you need to put = sign is that sometime formula is too long, and cannot be pasted in to cell.

## GPdotNET v3.0 User Guide

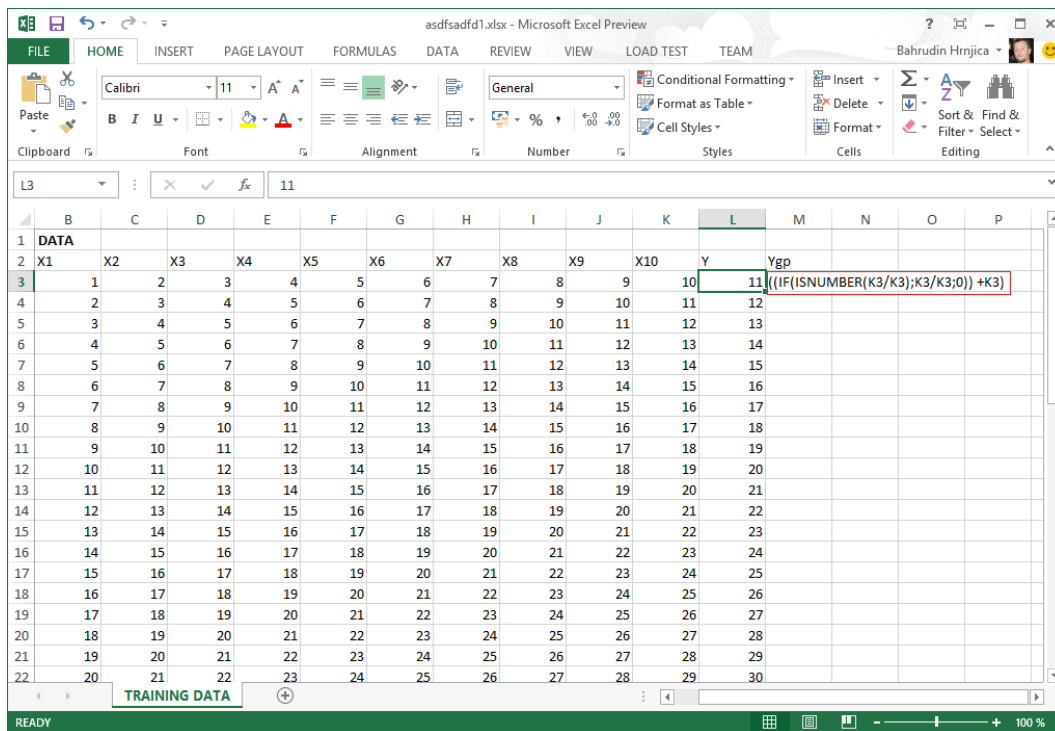
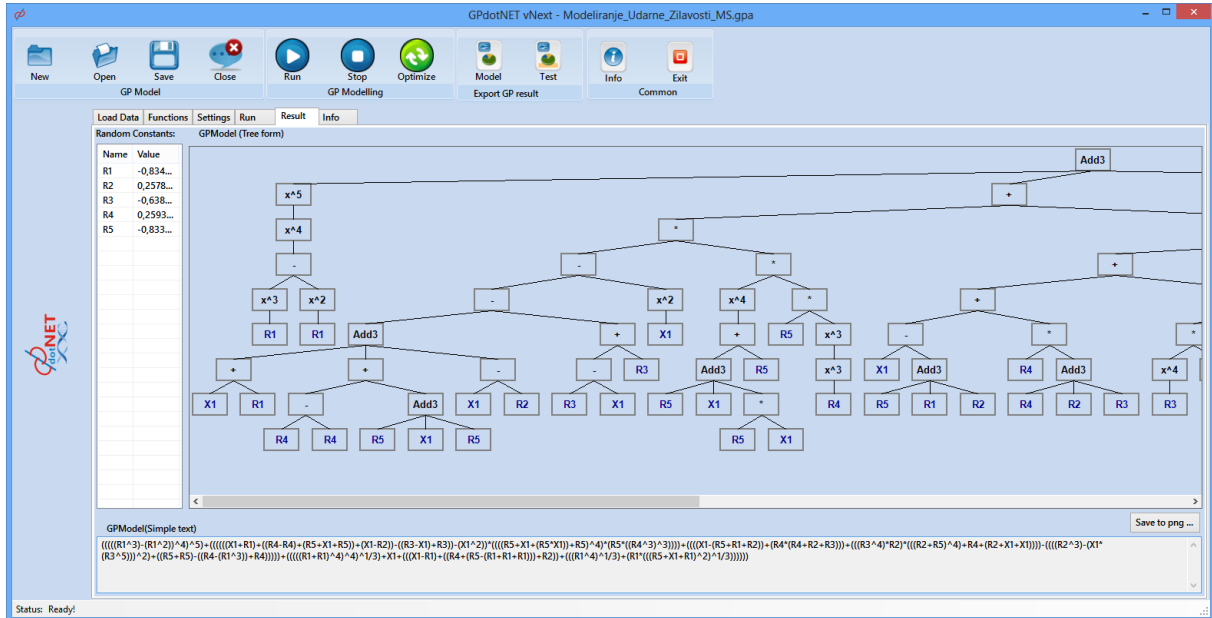


Figure 11.2:: Excel showing GPModel as excel formula. You need to type = in order to evaluate right formula.

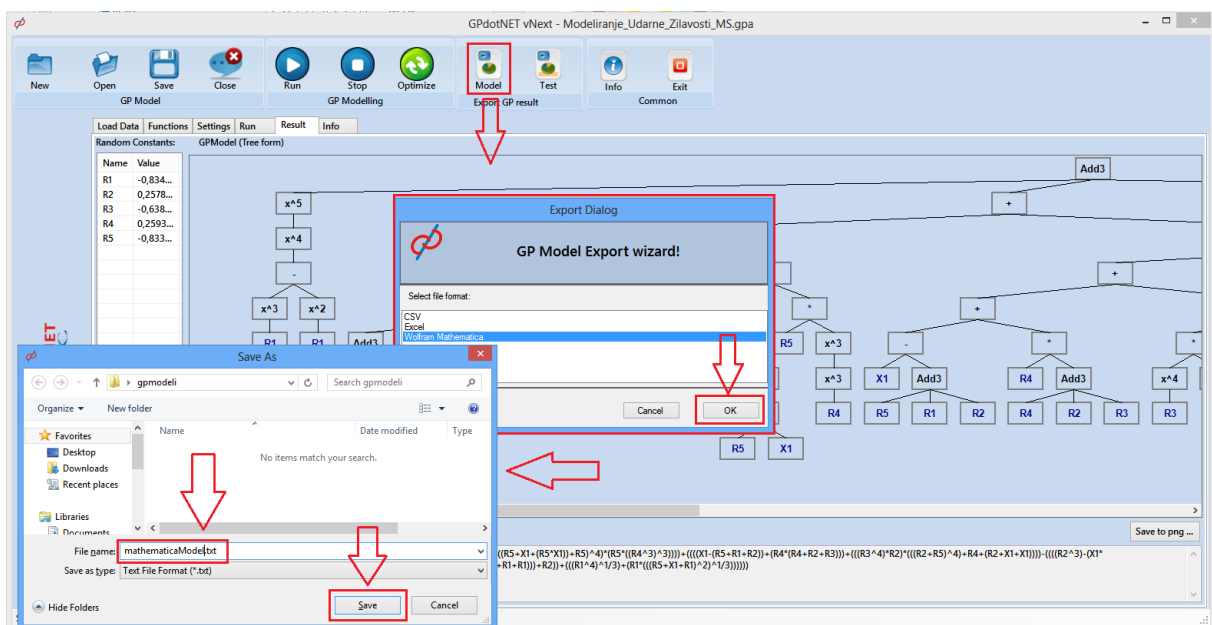
## 11.2. Export GPdotNET Model in to Mathematica

Suppose we have calculated the model and want to export it to Mathematica, see picture below.



The bottom text box from the picture above shows GP model in analytic form. This is pretty much terms and operations. Actually the analytic term of GP Model will be converted in to Mathematica syntax, and exported to txt file. On this way you can copy exported text and paste to Mathematica.

From the Export GP Result group controls choose GP Model button, select Mathematica, click OK, name the txt file and press OK button. All operations are showed on picture below.



## GPdotNET v3.0 User Guide

Now we have Training data set and GP model in Mathematica language, so we can copy them and paste in to Mathematica notebook.

Open saved txt file and you will see something similar showed on the picture below.



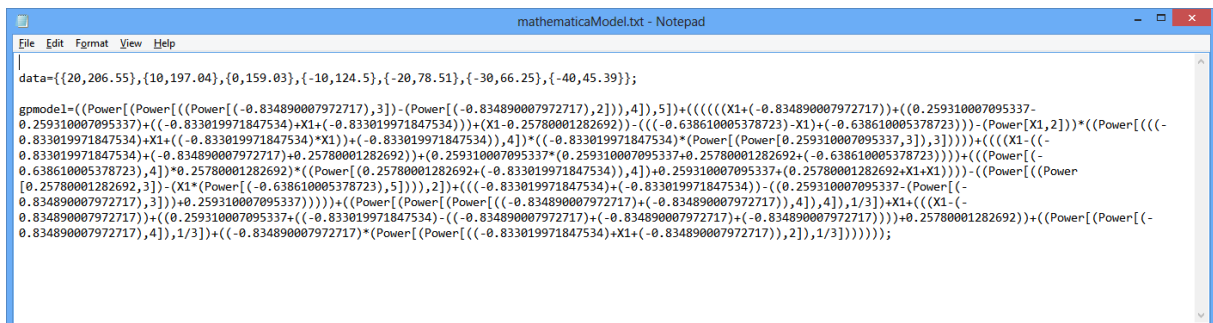
```
data = {{20, 206.55}, {10, 197.04}, {0, 159.03}, {-10, 124.5}, {-20, 78.51}, {-30, 66.25}, {-40, 45.39}};

gpmodel =
((Power[(Power[(Power[(-0.834890007972717), 3]) - (Power[(-0.834890007972717), 2]), 4]), 5]) +
((((((X1 + (-0.834890007972717)) + (0.259310007095337 - (0.259310007095337) * ((-0.833019971847534) - X1 - (-0.833019971847534)) * (X1 - (-0.25780001282692)) - ((-0.638610005378723) - X1) + (-0.638610005378723)) - (Power[X1, 2])) *
(Power[(Power[(Power[(-0.833019971847534) * X1] + (-0.833019971847534) * X1) + (-0.833019971847534) * X1) * (Power[(Power[0.259310007095337, 3]), 3])])) +
(((X1 - (-0.833019971847534) + (-0.834890007972717) + 0.25780001282692) * (0.259310007095337 * (0.259310007095337 - (0.25780001282692 - (-0.638610005378723))))) +
((Power[(Power[(-0.638610005378723), 4]) * 0.25780001282692] * (Power[(0.25780001282692 - (-0.833019971847534) * X1) + 0.259310007095337 * (0.25780001282692 - (-0.638610005378723) * X1))]) -
(Power[(Power[0.25780001282692, 3]) * (X1 + (Power[(-0.638610005378723), 5]), 2)) * ((-0.833019971847534) + (-0.833019971847534) * X1) - (0.259310007095337 - (Power[(-0.834890007972717), 3]) + 0.259310007095337))))) +
(Power[(Power[(Power[(-0.834890007972717), 4]), 4]), 1/3]) * X1 +
(((X1 - (-0.834890007972717)) * (0.259310007095337 - (-0.833019971847534) * X1) - (-0.834890007972717) * (-0.834890007972717)) * 0.25780001282692) +
(Power[(Power[(-0.834890007972717), 4]), 1/3]) * ((-0.834890007972717) * (Power[(Power[(-0.833019971847534) * X1 + (-0.834890007972717), 2]), 1/3]))))];
```

The file contain two things:

- Training data model represented as Mathematica list collection, and
- GP Model translated in to Mathematica notation.

Copy the first group of text and paste it to Mathematica notebook, then copy the second text group and paste to Mathematica below the first one. The flowing picture shows training data set and GP model in Mathematica.



```
data = {{20, 206.55}, {10, 197.04}, {0, 159.03}, {-10, 124.5}, {-20, 78.51}, {-30, 66.25}, {-40, 45.39}};

gpmodel = ((Power[(Power[(Power[(-0.834890007972717), 3]) - (Power[(-0.834890007972717), 2]), 4]), 5]) +
((((((X1 + (-0.834890007972717)) + (0.259310007095337 - (0.259310007095337) * ((-0.833019971847534) - X1 - (-0.638610005378723)) - (Power[X1, 2])) *
(Power[(Power[(Power[(-0.833019971847534) * X1] + (-0.833019971847534) * X1) + (-0.833019971847534) * X1) * (Power[(Power[0.259310007095337, 3]), 3])])) +
(((X1 - (-0.833019971847534) + (-0.834890007972717) + 0.25780001282692) * (0.259310007095337 * (0.259310007095337 - (0.25780001282692 - (-0.638610005378723))))) +
((Power[(Power[(-0.638610005378723), 4]) * 0.25780001282692] * (Power[(0.25780001282692 - (-0.833019971847534) * X1) + 0.259310007095337 * (0.25780001282692 - (-0.638610005378723) * X1))]) -
(Power[(Power[0.25780001282692, 3]) * (X1 + (Power[(-0.638610005378723), 5]), 2)) * ((-0.833019971847534) + (-0.833019971847534) * X1) - (0.259310007095337 - (Power[(-0.834890007972717), 3]) + 0.259310007095337))))) +
(Power[(Power[(Power[(-0.834890007972717), 4]), 4]), 1/3]) * X1 +
(((X1 - (-0.834890007972717)) * (0.259310007095337 - (-0.833019971847534) * X1) - (-0.834890007972717) * (-0.834890007972717)) * 0.25780001282692) +
(Power[(Power[(-0.834890007972717), 4]), 1/3]) * ((-0.834890007972717) * (Power[(Power[(-0.833019971847534) * X1 + (-0.834890007972717), 2]), 1/3]))))];
```

Now that we have GP model in Mathematica, we can do lot of things.

Simplifying GP analytic model: By typing **Simplify[gpmodel]**, Mathematica will try to simplify your model as simple as possible.

Look what happens when we execute **Simplify[gpmodel]** against the model we are currently dealing:

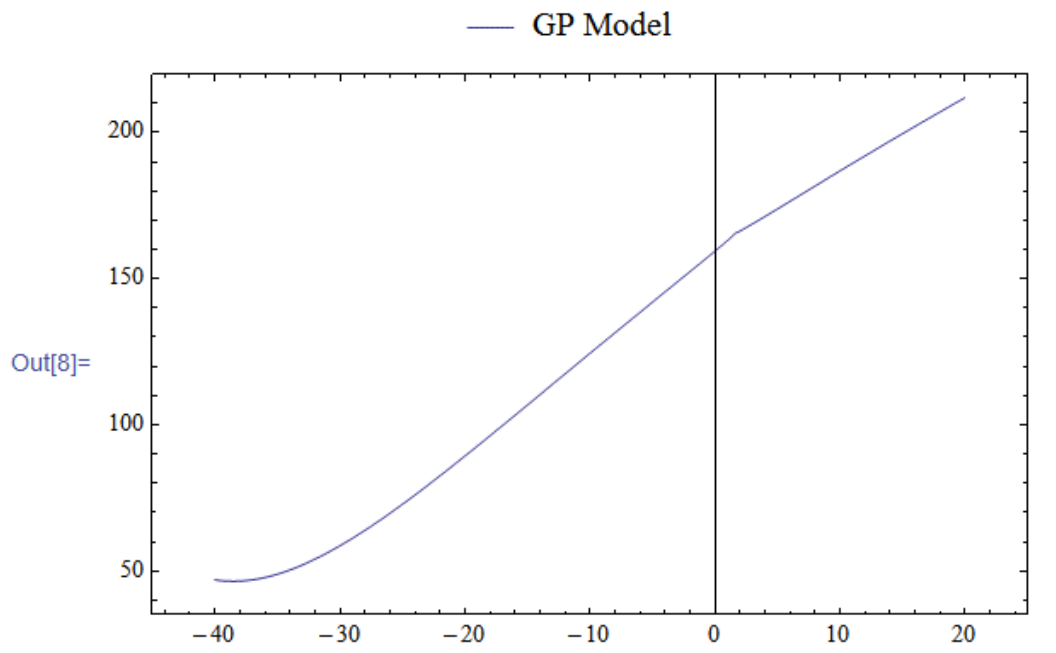
```
In[4]:= Simplify[gpmodel]
Out[4]= -0.83489 (-192.345 + 1. ((-1.66791 + X1)^2)^(1/3) - 3.69145 X1 + 0.0134025 X1^2 + 0.0000264049 X1^3 - 3.11866 × 10^-6 X1^4 + 1.8056 × 10^-7 X1^5 - 4.11204 × 10^-9 X1^6)
```

This is awesome, and nobody can simplify such a complex term better than Mathematica.

If your model contains 3 or less independent variables you can plot it. For example execute this command:

```
funpl = Plot[{gpmodel}, {X1, -40, 20}, PlotRange -> {{-45, 25}, {35, 220}}, Frame -> True ,  
PlotLegends -> Placed[{"GP Model"}, Above]]
```

The picture below shows the result from the command above:



That was just a few options you can execute against GP model when you export it to Mathematica.