# Instruction for CADLIVE toolbox

Update Nov 29, 2013

## Table of contents

Environments ........ 1

* Install of CADLIVE GUI Network Constructor and CADLIVE Text Editor ........ 1
* Install of CADLIVE toolbox ........ 1
* Path for Java ........ 1
* Path on MATLAB ........ 2
* Make a folder for a model ........ 2

0 Overview ........ 3

1 Conversion into dynamic model and simulation ........ 6

1.1 Start ........ 6
1.2 Load of a regulator-reaction equation data file ........ 7
1.3 Selection of conversion methods ........ 8
1.4 Edition of mathematical model data ........ 9
1.5 Selection of analysis types and input control data ........ 10
1.6 Input of parameters ........ 12
1.6.1 CADLIVE_initial.m ........ 19
1.6.2 CADLIVE_param.m ........ 21
1.7 Results ........ 24
1.7.1 Dynamic analysis ........ 24
1.7.2 Steady-state analysis ........ 30
1.7.3 S-system ........ 32

2 GA ........ 33

2.1 Start ........ 34
2.2 Search parameter setting ........ 36
2.3 Fitness function setting ........ 37
2.4 Execution ........ 40

3 TPS ........ 42

3.1 Start ........ 43
3.2 QMPS on TPS ........ 44

3.3 Execution ........................................................................................................ 45

4 Execution on command line ..................................................................................... 49

4.1 CADLIVE_simulator ........................................................................................... 49
4.2 CADLIVE_myGAcommand ................................................................................. 52
4.3 CADLIVE_myTPScommand ................................................................................ 53
4.4 CADLIVE_DispFigure ......................................................................................... 55
4.5CADLIVE_DispCumulFreqQMPS, CADLIVE_DispHistFreqQMPS,
CADLIVE_DispHistCumulFreqQMPS ..................................................................... 56
4.6 CADLIVE_optimtool ........................................................................................... 58

References ................................................................................................................ 59

# Environments

Developmental environment

| Software | Version |
| --- | --- |
| OS | Windows 7 (64 bit) |
| MATLAB | Ver.7.8.0（R2009a） |
| JAVA | Java SE 6 (**32bit**)<br>Xerces Java Parser-2.7.0 |

Path

| Application | Path |
| --- | --- |
| Java | C:¥Program Files (x86)¥Java¥jre6 |
| CADLIVE Text Editor | C:¥CADLIVE¥ |
| CADLIVE MATLAB (this application) | C:¥CADLIVE_v1 |

## * Install of CADLIVE GUI Network Constructor and CADLIVE Text Editor

These are described at http://www.cadlive.jp/cadlive/editor/download.html

## * Install of CADLIVE toolbox

CADLIVE_v1.zip (http://www.cadlive.jp) is unzipped, the folder CADLIVE_v1 is copied to "C:¥".

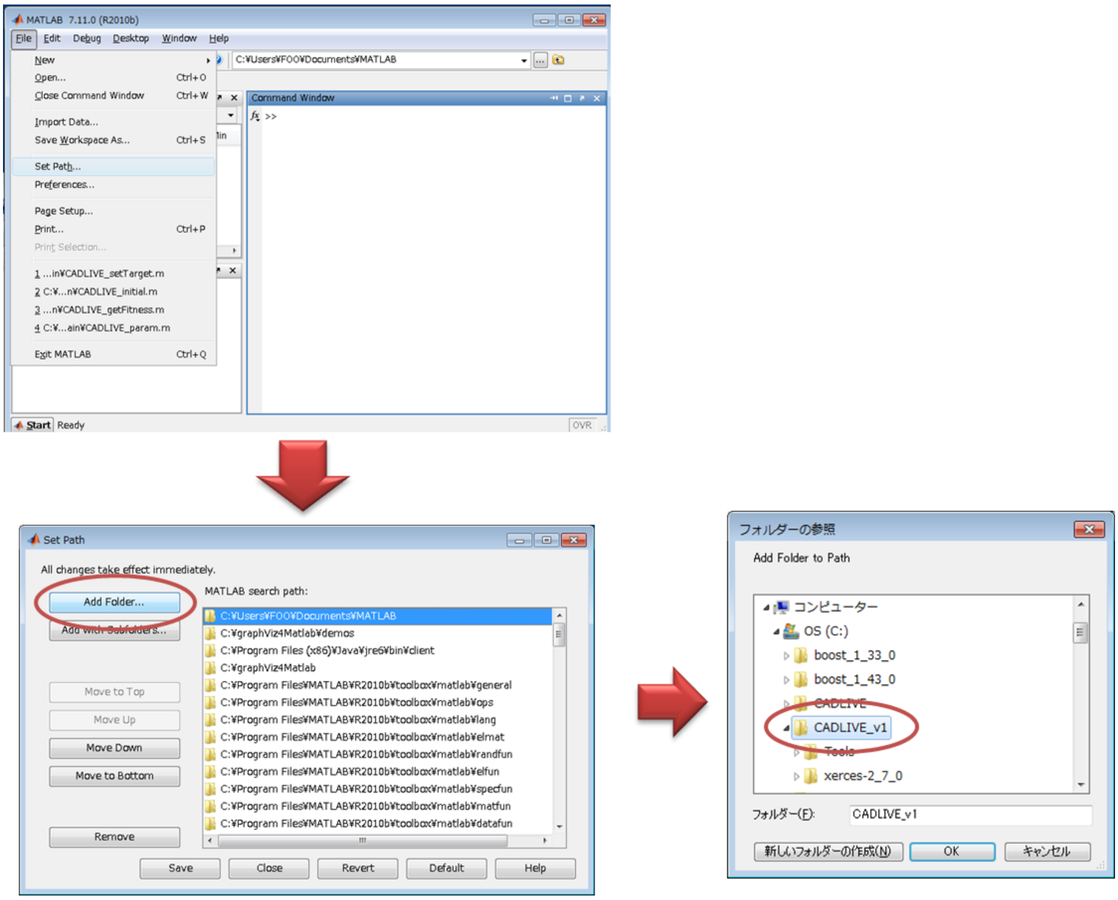
## * Path for Java

Open the file C:¥CADLIVE_v1¥CADLIVE_setenv.m, and change the path as follows if users use Windows 32 bit.

%setenv('JAVA_HOME32','C:¥Program Files (x86)¥Java¥jre6');
setenv('JAVA_HOME32','C:¥Program Files¥Java¥jre6');

## * Path on MATLAB

Run MATLAB, and the folder, C:¥CADLIVE_v1, is added to the path of MATLAB.




## * Make a folder for a model

The files for a model are made in the current folder. Users should make a folder for the model and use this application in the folder.

# 0 Overview

A goal of systems biology is to construct biological systems at the molecular interaction levels and to understand some design principles underlying the molecular processes. Biochemical networks are the sound bases for pathway analysis and dynamic modeling. The CADLIVE system implements a variety of application modules to perform the network analysis and the dynamic simulations based on biochemical network maps (http://www.cadlive.jp) [1-5]. As an extension of CADLIVE, this standalone application is developed for constructing mathematical models which work on MATLAB.

This application has functions for conversion into dynamic model and simulation, parameter optimization, and system analysis (**Fig.1**). First, the conversion and simulation module automatically converts a biochemical map into a mathematical model and subsequently simulates the dynamic behaviors. Here, the mathematical model is made in MATLAB. Second, the parameter optimization module employs a genetic algorithm (GA) and two-phase search (TPS) method [6] to seek out a global minimum and to estimate many plausible values of the kinetic parameters that determine the dynamic behavior of systems, respectively. The employed GA is derived from the CADLIVE Optimizer [5]. On the other hand, the TPS smoothly combines a random search with an evolutionary algorithm to achieve both nonbiased and high-speed searches for a large parameter space. Finally, system analysis module includes the sensitivity analysis with respect to a single parameter and qausi-multiparameter sensitivity (QMPS) [7]. QMPS measures a robust property of the model to the uncertainty of all kinetic parameters and provides a theoretical or quantitative insight to an understanding of how specific network structures are related to robustness.

These algorithms in CADLIVE greatly facilitate simulating and analyzing a biological system, enhancing the efficiency for the research in systems biology.

This application has mainly three parts; conversion into dynamic model and simulation, and simulations for local parameter optimization by GA and global parameter optimization by TPS.

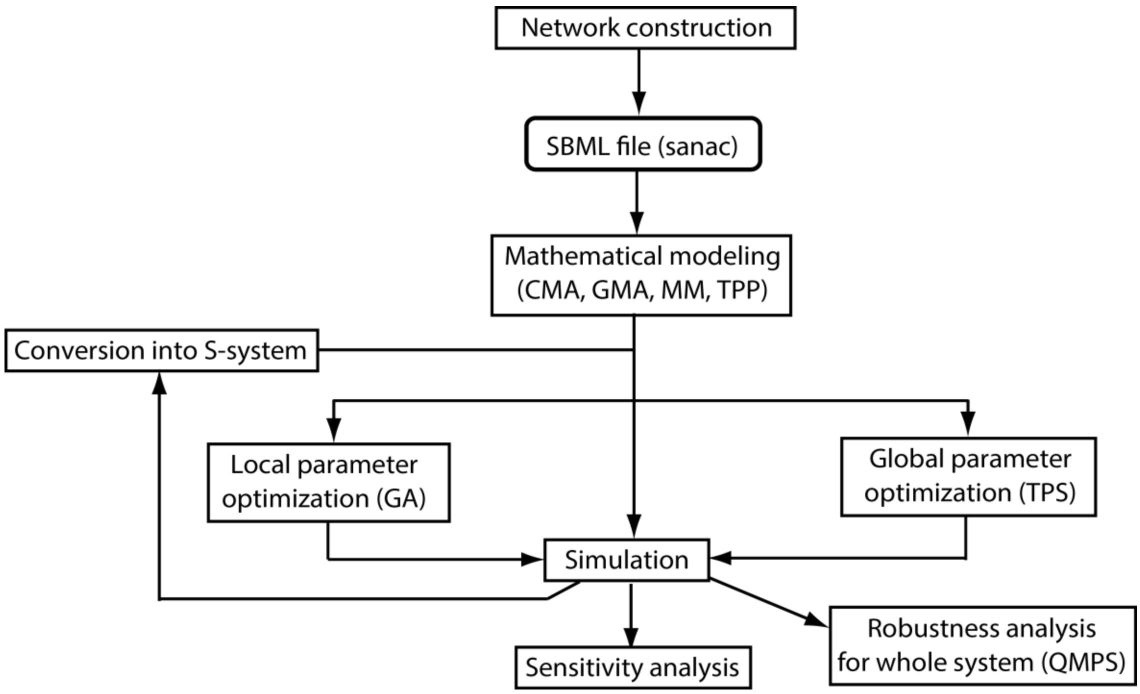



Fig.1 Overview of functions for CADLIVE toolbox

Here, we illustrate the simulation of a straight reaction chain model. The map of the model is made by the CADLIVE GUI Network Constructor (**Fig.2**). X0 is the constant. X1, X2, X3 and X4 are the time-dependent variables. X1, X2 and X3 don’t decompose and X4 decompose. All the reactions occur in the metabolic layer. StraightChainModel.xml is written in the CADLIVE format (**Fig.3**).

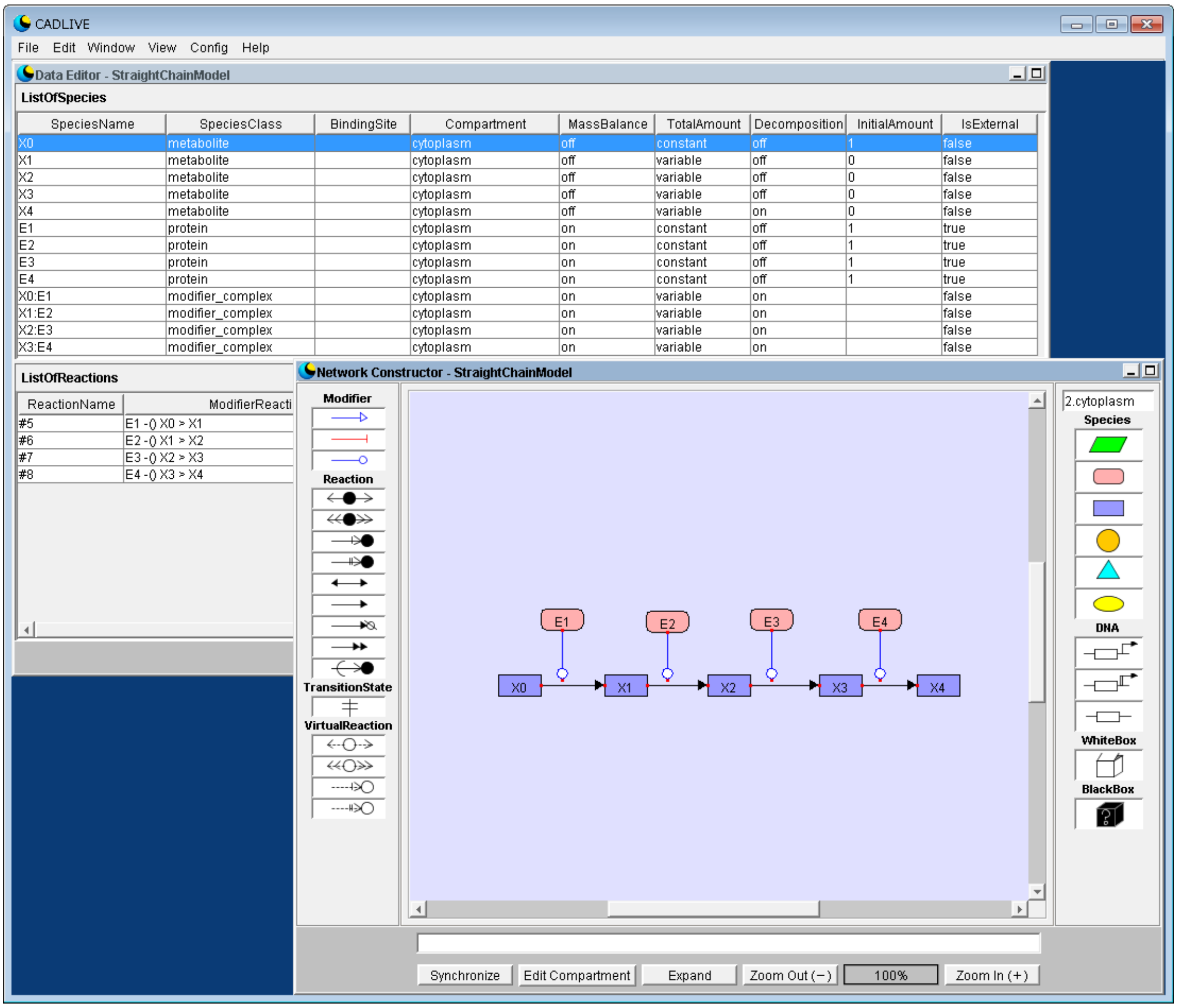



Fig.2 A straight reaction chain model in the CADLIVE GUI Network Constructor

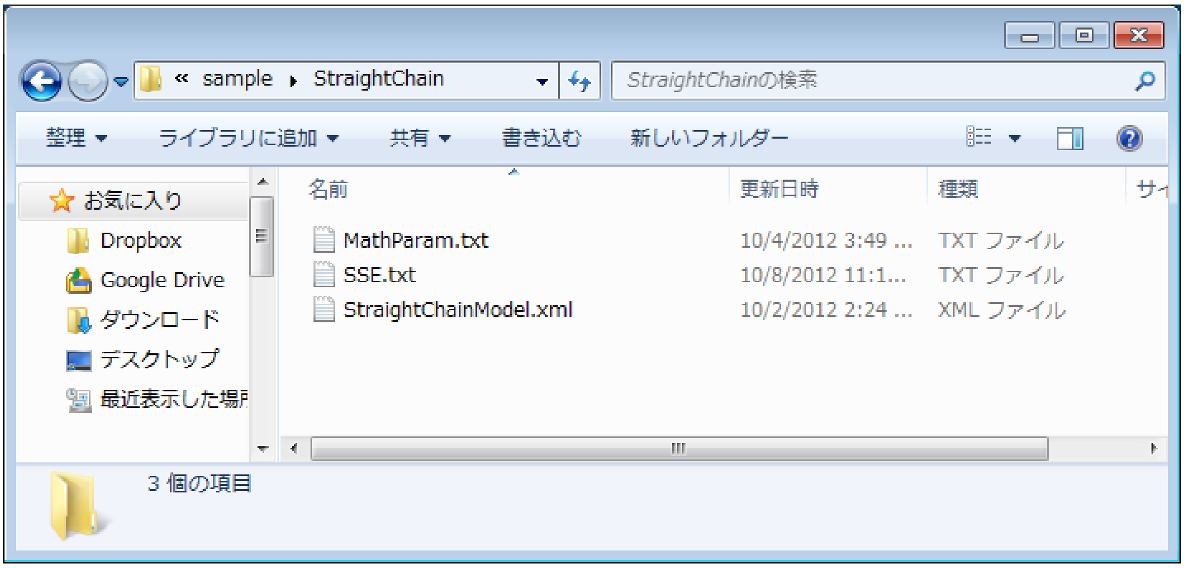



Fig.3 Folder prepared for the dynamic model for this application

# 1 Conversion into dynamic model and simulation

This part is an application program for automatically converting biochemical networks into mathematical models, simulating the model and performing sensitivity analysis.

## 1.1 Start

Execute “CADLIVE_start” on the MATLAB command window to start the CADLIVE toolbox (**Fig.4**).

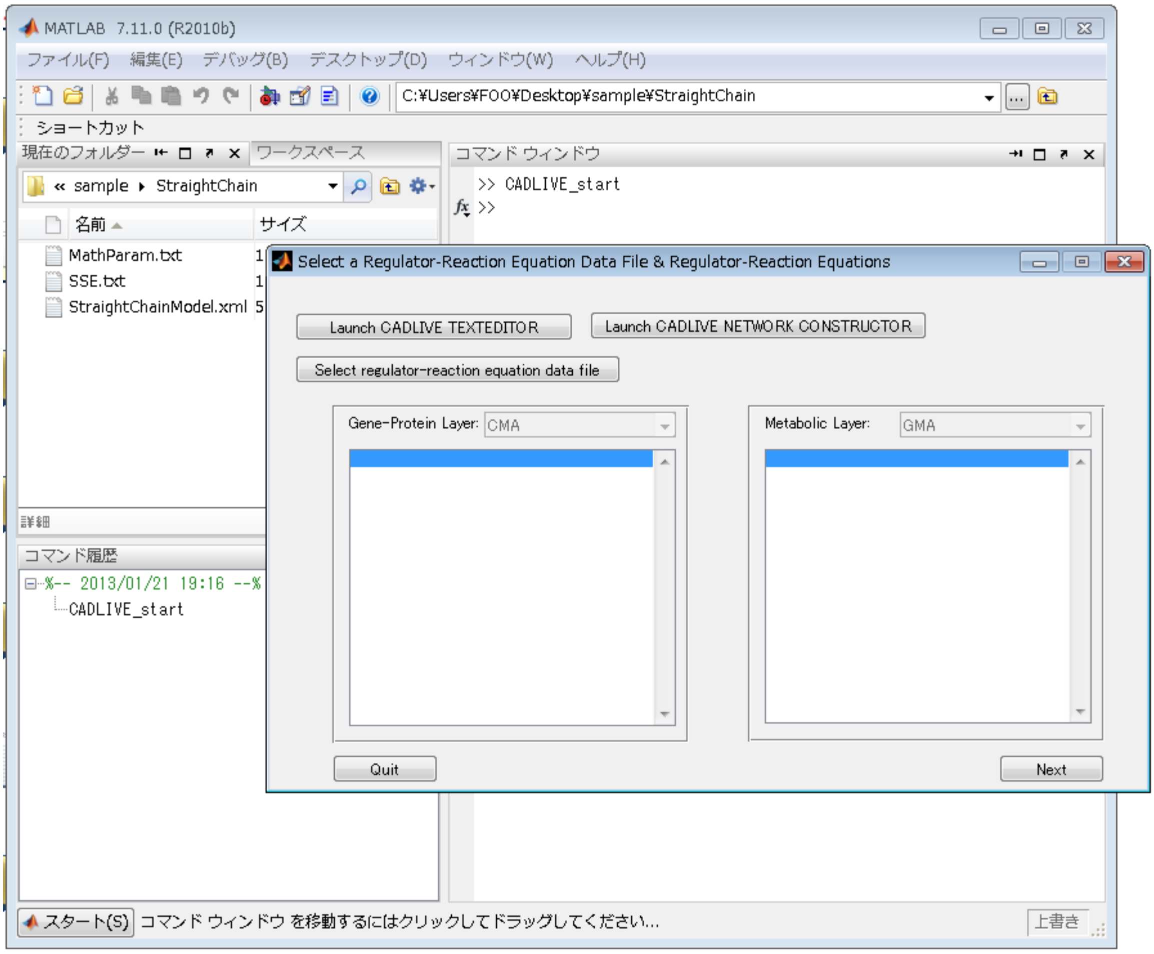



Fig.4 Start of the CADLIVE toolbox

## 1.2 Load of a regulator-reaction equation data file

By clicking the "Launch CADLIVE TEXTEDITOR" button, the CADLIVE Text Editor is opened. The CADLIVE Text Editor helps users describing a network model. The manual of the CADLIVE Text Editor is downloaded at http://www.cadlive.jp/cadlive/editor/download.html.

By clicking the "Select regulator-reaction equation data file" button, users select the data file (CADLIVE format file) for the regulator-reaction equation model from their PC (**Fig.5**). The data file written in the XML format is built by the CADLIVE Text Editor [2], GUI Network Constructor [2,3] or Converter [4].

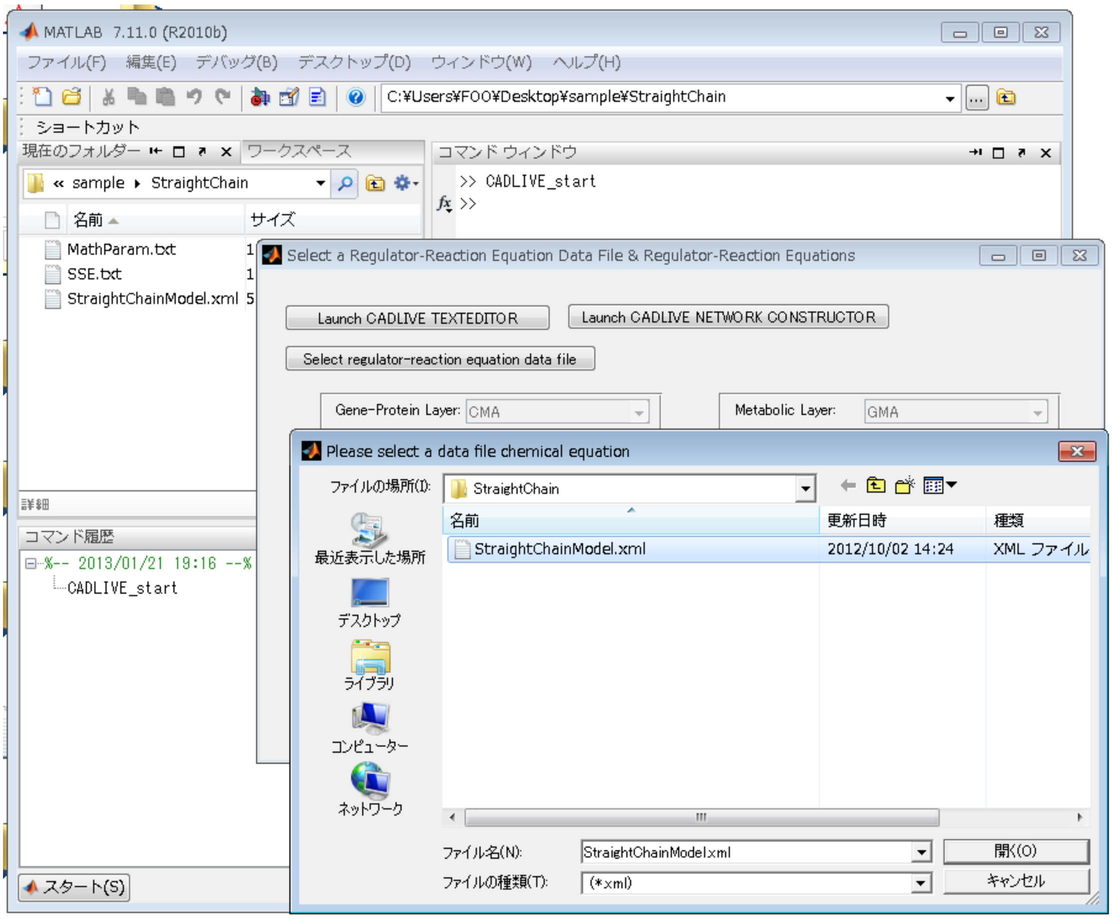



Fig.5 Selection of the regulator-reaction equation data file

*Once users convert an xml file, the DAE file with a mathematical model can only be loaded if users use the same model in the next.

### 1.3 Selection of conversion methods

The selected regulator-reaction equation data file is displayed in the window (**Fig.6**), where users choose the conversion methods with respect to gene-protein layer and metabolic layer, respectively. The conversion method can be selected out of the following methods:

Gene-Protein layer:

・CMA

・TPP_STEADYSTATE_1

・TPP_STEADYSTATE_2

・TPP_RAPID

Metabolic layer:

・GMA

・MM

・SAME_AS_GENE-PROTEIN

The details of description for them are in
http://www.cadlive.jp/cadlive/simulator/Suppl_method_1.pdf.

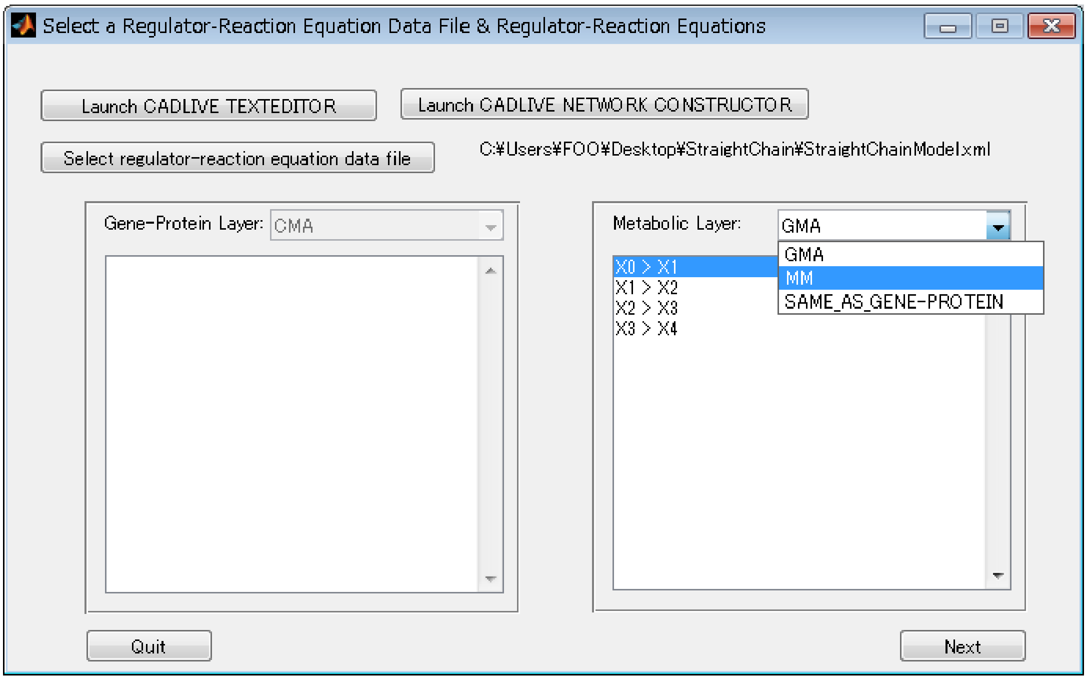



Fig.6 Selection of conversion methods for the Gene-Protein Layer and Metabolic Layer

## 1.4 Edition of mathematical model data

By clicking the “Next” button on the “Select a Regulator-Reaction Equation Data File & Regulator-Reaction Equations” window, the “Edit Math Model Data” window is displayed. By clicking the “Edit Parameter” button on the window, the file for a math model (DAE file) is opened on the MATLAB editor so that users can change the model and its parameters (**Fig.7**). This file is saved as MathDAE.txt in the current folder.

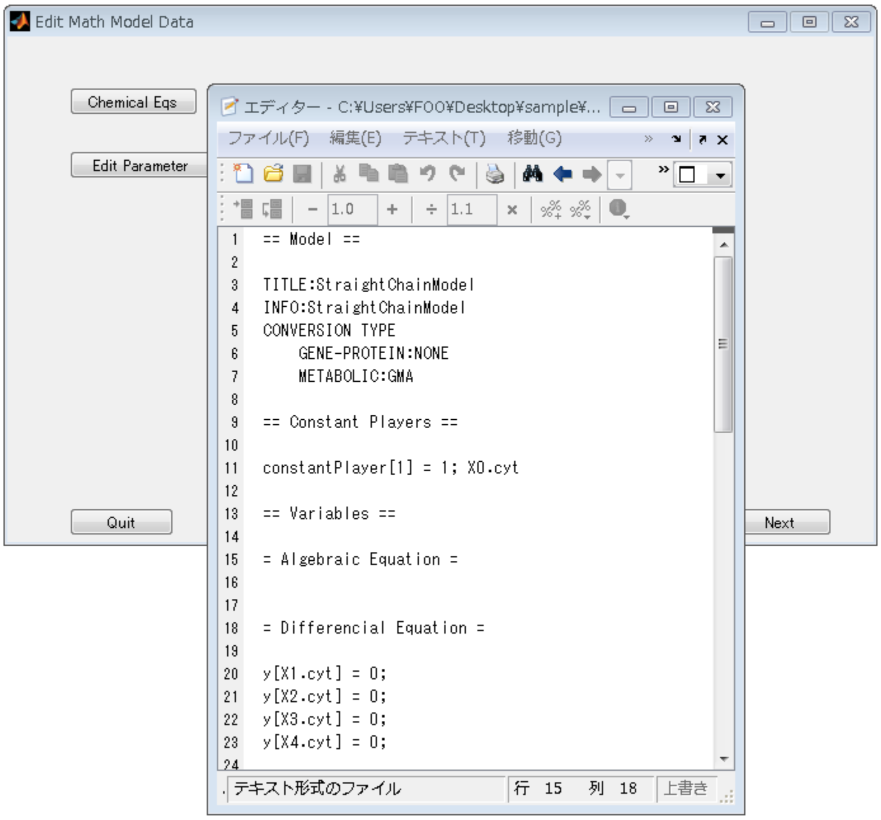



Fig.7 Edition for a mathematical model

* The DAE file must be saved before clicking the “Next” button, “Back” button or “Quit” button if users change its content.

* If users manually make the mathematical model from scratch, users can load a dummy xml file and edit the DAE file in this control or the edited DAE file can be loaded in Section 1.2.

## 1.5 Selection of analysis types and input control data

By clicking the “Next” button on the “Edit Math Model Data” window, the “Select Analysis Type & Set Control Data for Simulation” window is displayed, where users select the analytical type for a mathematical model and input conditions of the type (**Fig.8**).

Analysis type

Users can choose either “Dynamic Analysis” or “Steady-state Analysis”. “Dynamic Analysis” simulates the time evolution of the values by calculating differential and algebraic equations, and “Steady-state Analysis” calculates the values at steady state by solving algebraic equations. The checkbox of "Parameter survey" determines if the simulation surveys the parameter space. The checkbox of "Use S-system", which employs S-system differential equations, appears only under the condition that the simulation has been solved before. It never appears when the TPP is selected as the conversion method.

Control data for simulation

In “Dynamic Analysis”, users set “Solver Type (tolerance)” and “Set time span and time step-size”. “relative” is a relative tolerance and “absolute” is an absolute tolerance.
In “Steady-state Analysis”, users set “Set values for Newton-Raphson Method”. Sensitivities with respect to a change in each parameter are simultaneously calculated when “Steady-state Analysis”.

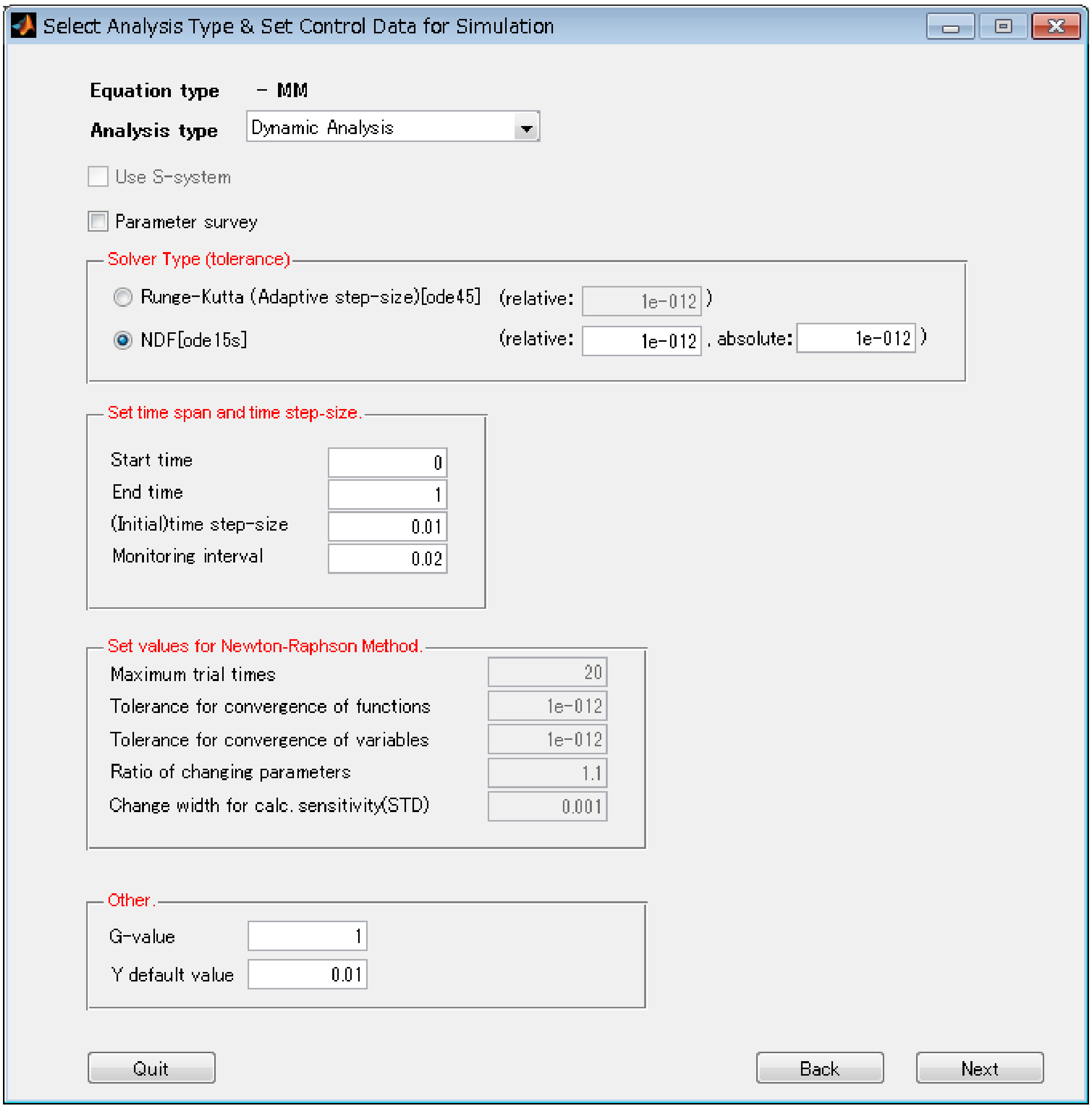


Fig.8 Selection of Analysis Type and Set of Control Data for Simulation

## 1.6 Input of parameters

By clicking the “Next” button on the “Select Analysis Type & Set Control Data for Simulation” window, the “Set Parameters and Initial Values” window is displayed (**Fig.9**), where users input the (initial) values of kinetic parameters and the variables for the mathematical model.

User Functions

Users select “usr_fvec” or “usr_fjac” button. “usr_fvec” (CADLIVE_usr_fvec.m) is the function for the differential equations (**Fig.10**). “usr_fjac” (CADLIVE_usr_fjac.m) is for the Jacobian function (**Fig.11**). By clicking the “Edit User Function” button, the file selected is opened and can be edited on the MATLAB editor.

Initial Values

Users select the “Initial Value” or “Parameters” button. “Initial Value” (CADLIVE_initial.m) has information of the control data and dependent variables (**Fig.12**). “Parameters” (CADLIVE_param.m) has information of the (kinetic) constant parameters (**Fig.13**). By clicking the “Edit Initial Values” button, the file selected is opened on the MATLAB editor.

Merge File

The Merge File helps users setting the parameters by copying or merging the existing data, which greatly reduces laborious parameter setting. By clicking the “Execute Merge” button, the existing data of a parameter file is copied to CADLIVE_initial.m and CADLIVE_param.m (**Fig.14**). The parameter file can be downloaded as “MathParam.txt” (**Fig.15**). By checking the “Update Blank Only” button, the only blank data of initial values and parameters are added to CADLIVE_initial.m and CADLIVE_param.m as uploaded data. By checking the “Update All” button, all the data are input.

Here, users mainly edit CADLIVE_initial.m and CADLIVE_param.m. The meanings for each variable are described in section 1.6.1 and 1.6.2. If users edit the differential equations in “usr_fvec” and/or add new variables, the variables need to be added to CADLIVE_initial.m and CADLIVE_param.m. The entire edition needs to be saved before next operation.

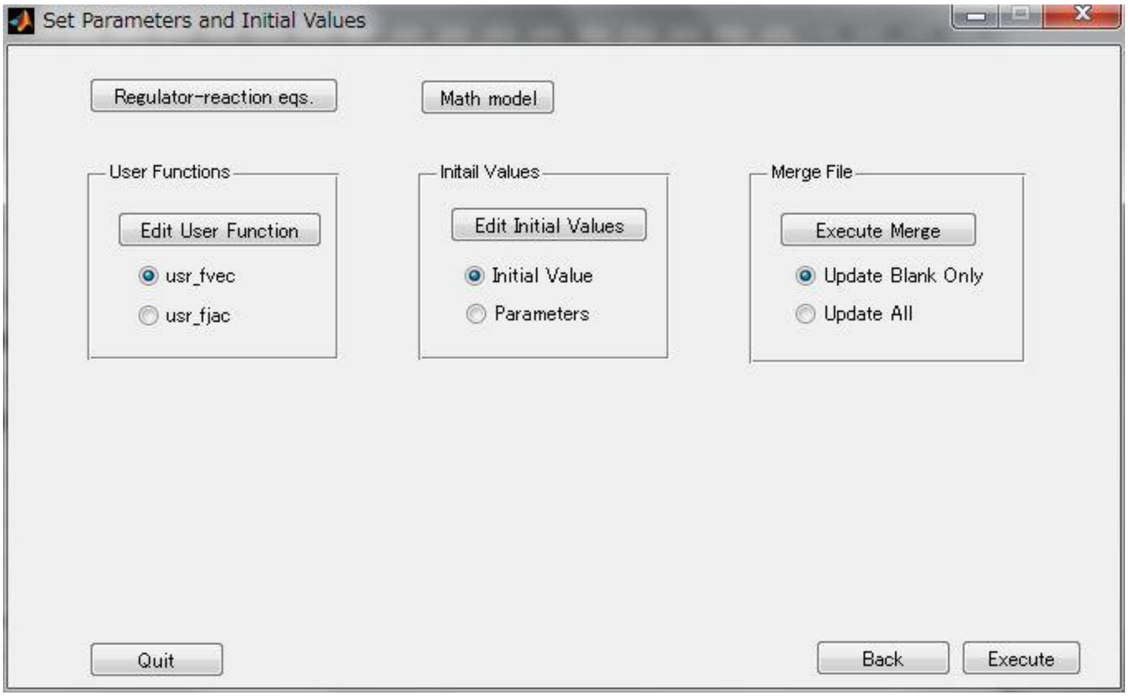



Fig.9 Control of setting parameters

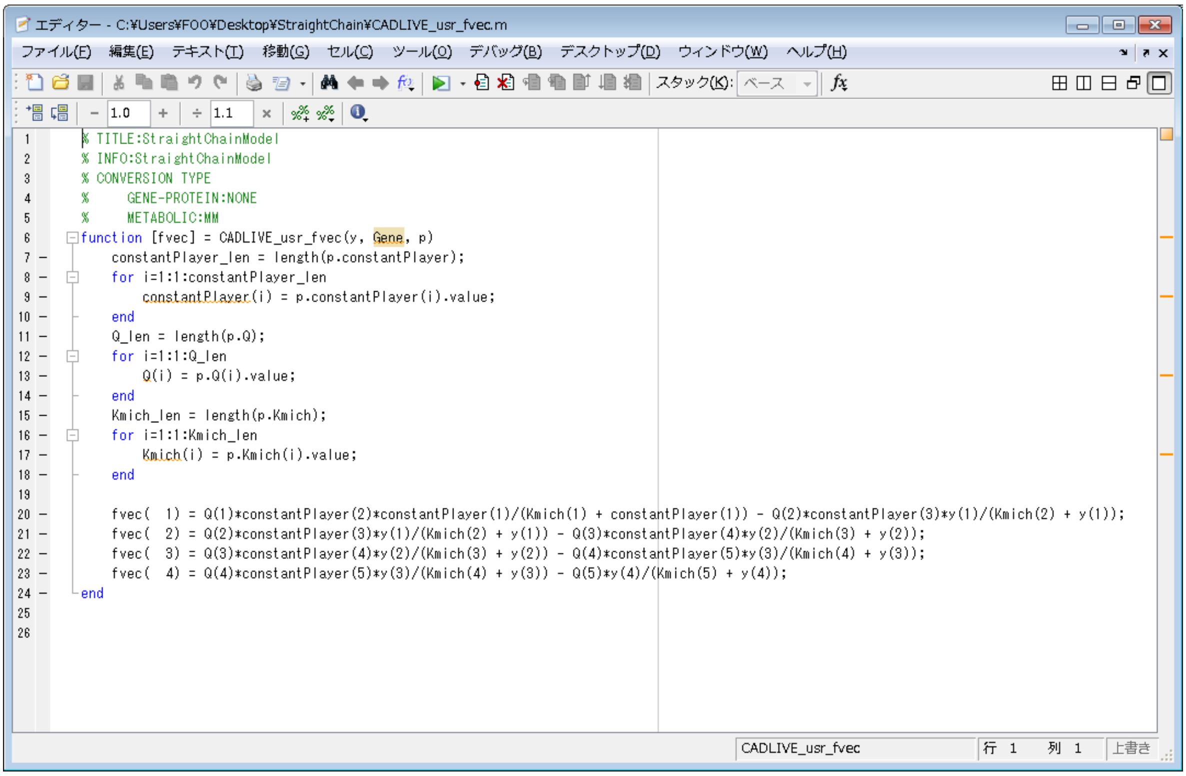
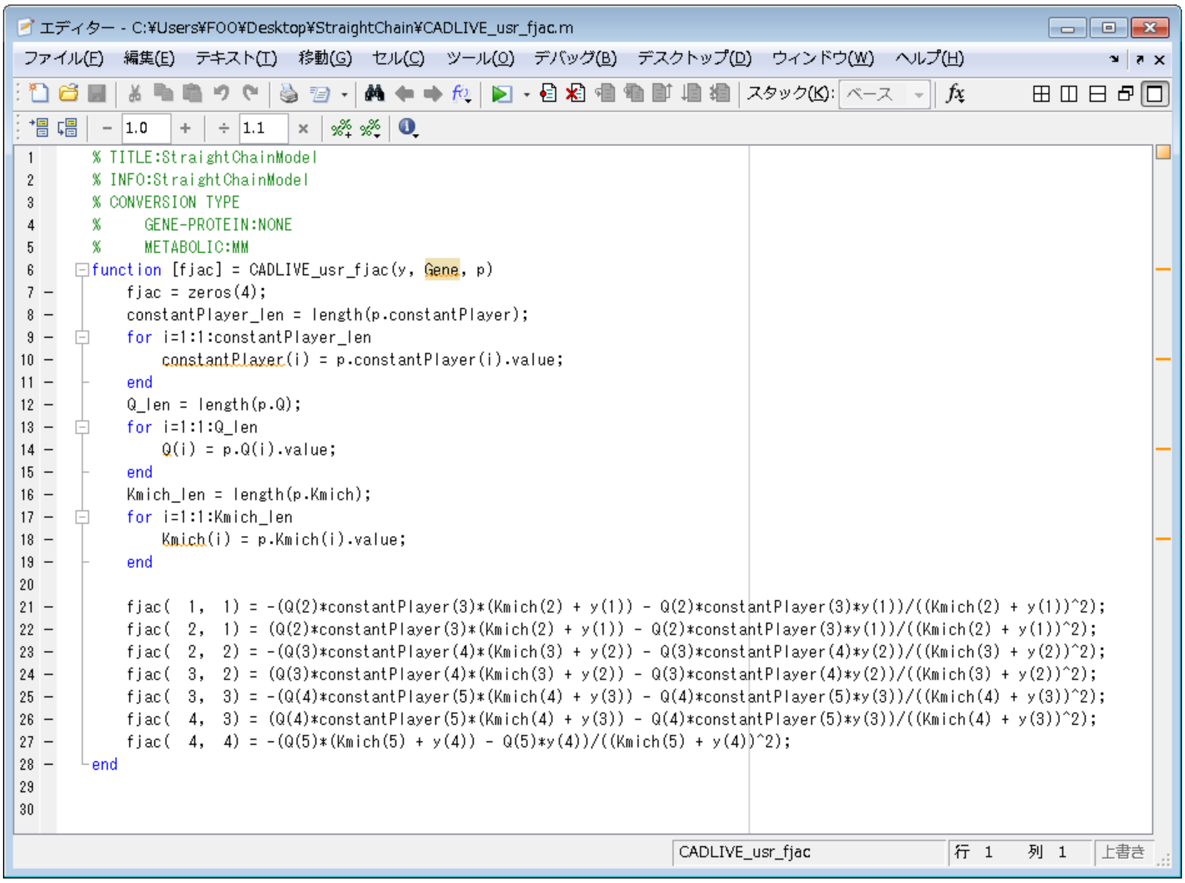


Fig.10 CADLIVE_usr_fvec.m

The “fvec” indicates the differential equation for a dependent variable y.

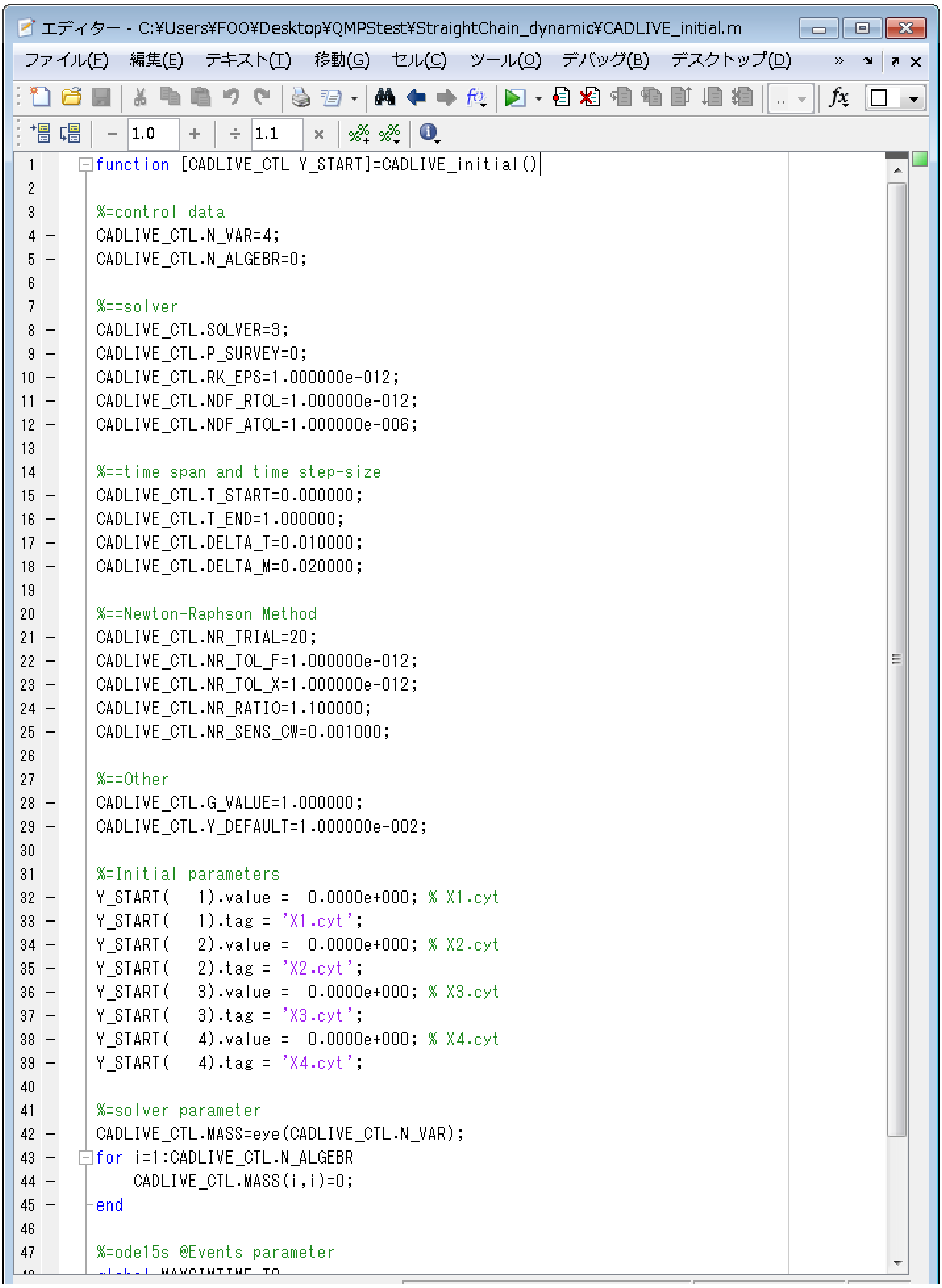



```matlab
% TITLE:StraightChainModel
% INFO:StraightChainModel
% CONVERSION TYPE
%     GENE-PROTEIN:NONE
%     METABOLIC:MM
function [fjac] = CADLIVE_usr_fjac(y, Gene, p)
    fjac = zeros(4);
    constantPlayer_len = length(p.constantPlayer);
    for i=1:1:constantPlayer_len
        constantPlayer(i) = p.constantPlayer(i).value;
    end
    Q_len = length(p.Q);
    for i=1:1:Q_len
        Q(i) = p.Q(i).value;
    end
    Kmich_len = length(p.Kmich);
    for i=1:1:Kmich_len
        Kmich(i) = p.Kmich(i).value;
    end

    fjac(  1,  1) = -(Q(2)*constantPlayer(3)*(Kmich(2) + y(1)) - Q(2)*constantPlayer(3)*y(1))/((Kmich(2) + y(1))^2);
    fjac(  2,  1) = (Q(2)*constantPlayer(3)*(Kmich(2) + y(1)) - Q(2)*constantPlayer(3)*y(1))/((Kmich(2) + y(1))^2);
    fjac(  2,  2) = -(Q(3)*constantPlayer(4)*(Kmich(3) + y(2)) - Q(3)*constantPlayer(4)*y(2))/((Kmich(3) + y(2))^2);
    fjac(  3,  2) = (Q(3)*constantPlayer(4)*(Kmich(3) + y(2)) - Q(3)*constantPlayer(4)*y(2))/((Kmich(3) + y(2))^2);
    fjac(  3,  3) = -(Q(4)*constantPlayer(5)*(Kmich(4) + y(3)) - Q(4)*constantPlayer(5)*y(3))/((Kmich(4) + y(3))^2);
    fjac(  4,  3) = (Q(4)*constantPlayer(5)*(Kmich(4) + y(3)) - Q(4)*constantPlayer(5)*y(3))/((Kmich(4) + y(3))^2);
    fjac(  4,  4) = -(Q(5)*(Kmich(5) + y(4)) - Q(5)*y(4))/((Kmich(5) + y(4))^2);
end
```

Fig.11 CADLIVE_usr_fjac.m

The “fjac” indicates the partial derivative for “fvec”.




```
function [CADLIVE_CTL Y_START]=CADLIVE_initial()

%=control data
CADLIVE_CTL.N_VAR=4;
CADLIVE_CTL.N_ALGEBR=0;

%==solver
CADLIVE_CTL.SOLVER=3;
CADLIVE_CTL.P_SURVEY=0;
CADLIVE_CTL.RK_EPS=1.000000e-012;
CADLIVE_CTL.NDF_RTOL=1.000000e-012;
CADLIVE_CTL.NDF_ATOL=1.000000e-006;

%==time span and time step-size
CADLIVE_CTL.T_START=0.000000;
CADLIVE_CTL.T_END=1.000000;
CADLIVE_CTL.DELTA_T=0.010000;
CADLIVE_CTL.DELTA_M=0.020000;

%==Newton-Raphson Method
CADLIVE_CTL.NR_TRIAL=20;
CADLIVE_CTL.NR_TOL_F=1.000000e-012;
CADLIVE_CTL.NR_TOL_X=1.000000e-012;
CADLIVE_CTL.NR_RATIO=1.100000;
CADLIVE_CTL.NR_SENS_CW=0.001000;

%==Other
CADLIVE_CTL.G_VALUE=1.000000;
CADLIVE_CTL.Y_DEFAULT=1.000000e-002;

%=Initial parameters
Y_START(   1).value =  0.0000e+000; % X1.cyt
Y_START(   1).tag = 'X1.cyt';
Y_START(   2).value =  0.0000e+000; % X2.cyt
Y_START(   2).tag = 'X2.cyt';
Y_START(   3).value =  0.0000e+000; % X3.cyt
Y_START(   3).tag = 'X3.cyt';
Y_START(   4).value =  0.0000e+000; % X4.cyt
Y_START(   4).tag = 'X4.cyt';

%=solver parameter
CADLIVE_CTL.MASS=eye(CADLIVE_CTL.N_VAR);
for i=1:CADLIVE_CTL.N_ALGEBR
    CADLIVE_CTL.MASS(i,i)=0;
end

%=ode15s @Events parameter
```

Fig.12 CADLIVE_initial.m

The control data for simulation (section 1.5) and initial values of dependent variables are set.

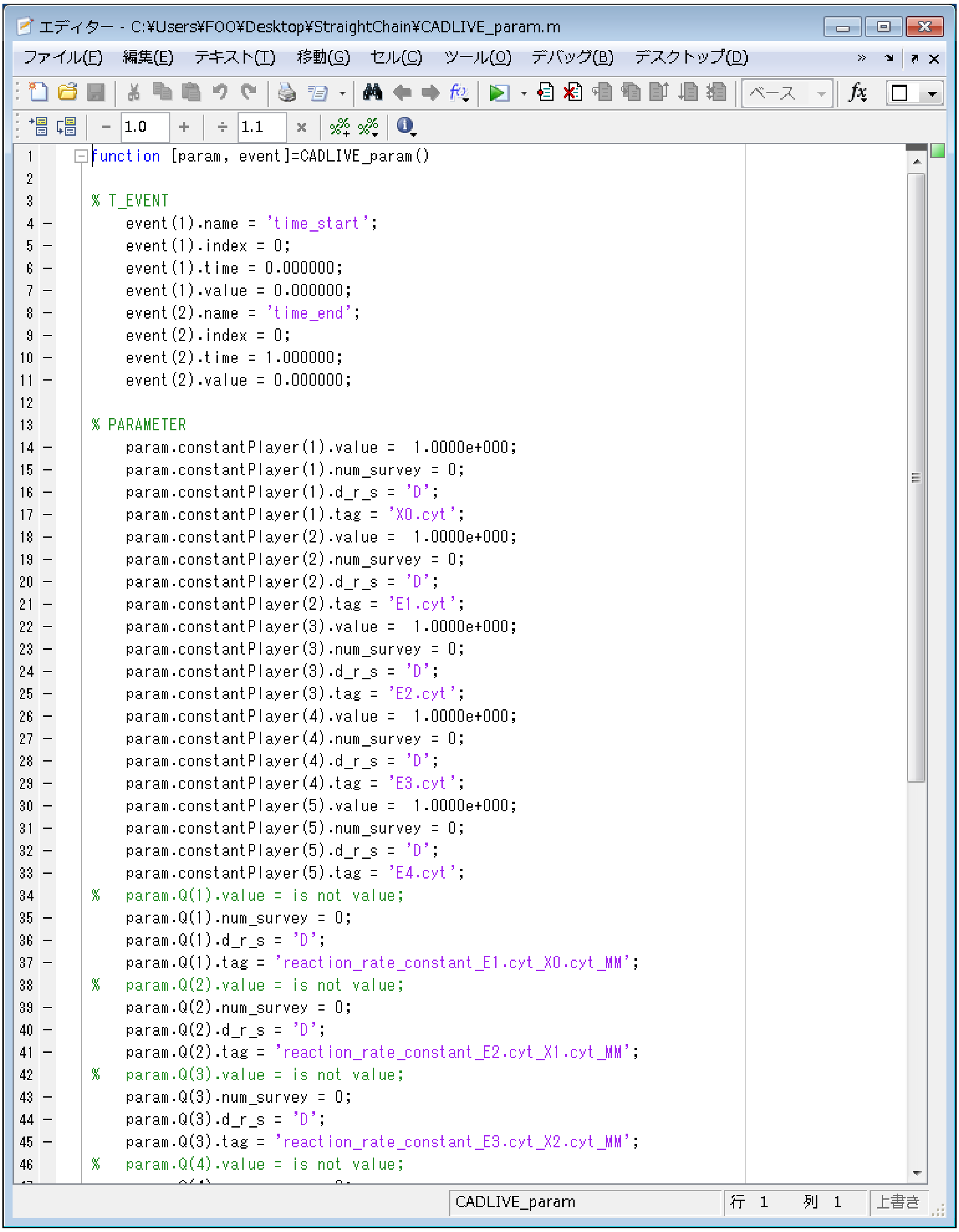



Fig.13 CADLIVE_param.m

The kinetic parameters and the events that change the parameter values in a given time are set.

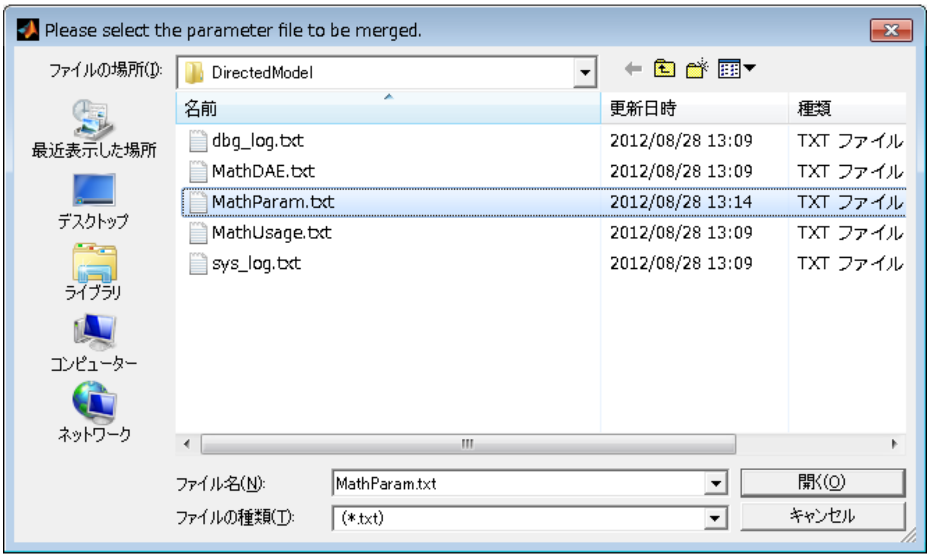



Fig.14 Select “MathParam.txt”

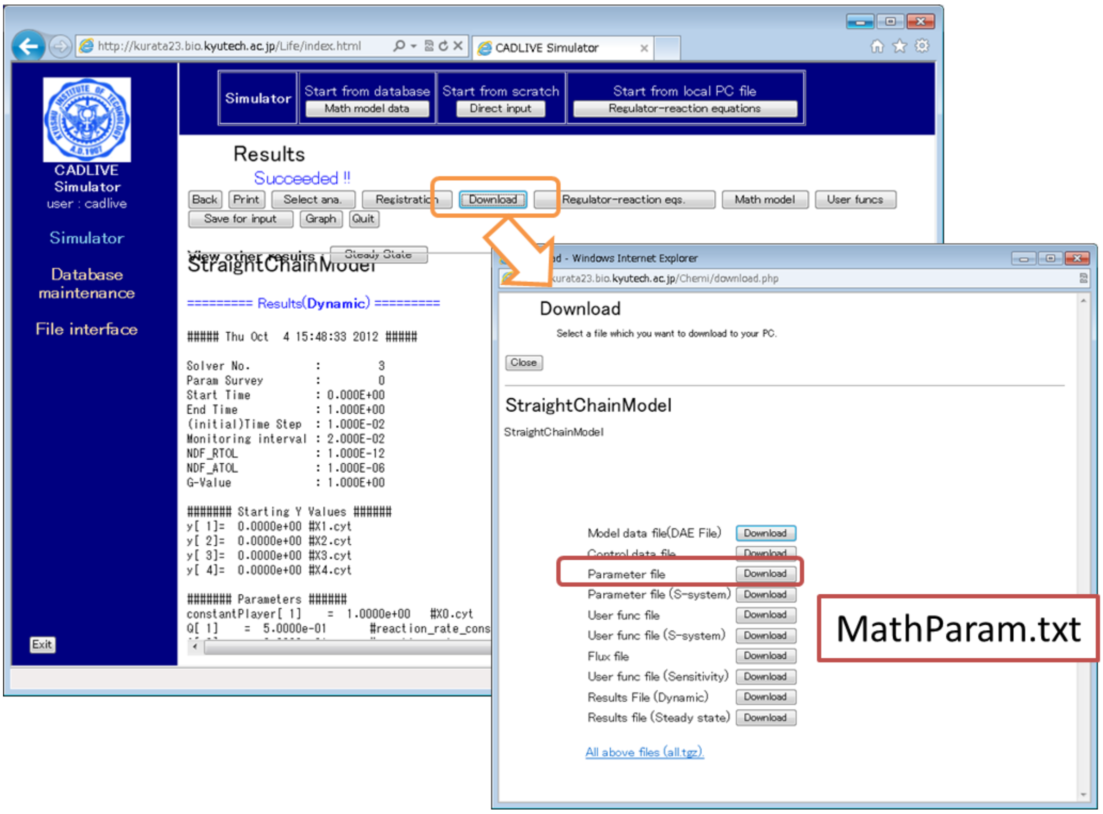



Fig.15 “MathParam.txt” in the CADLIVE Dynamic Simulator (Web application)

### 1.6.1 CADLIVE_initial.m

This file is written as control data ("CADLIVE_CTL") for simulation, and information ("Y_START") for the dependent variables. "CADLIVE_CTL" is automatically input by editing parameters in section 1.5. Users usually edit the "value" of "Y_START" in this file.

[Structure variables "CADLIVE_CTL"]

| Field | Meanings |
|---|---|
| N_VAR | Number of variables (all) ; Integer value > 0 |
| N_ALGEBR | Number of variables (algebraic equations) ; Integer value ≥ 0 |
| SOLVER | Solver number; 2: ode45, 3: ode15, 11: steady-state, 12: S-system |
| P_SURVEY | Parameter survey; 1: yes, 0: no |
| RK_EPS | Relative error tolerance for Runge-Kutta; Real value > 0 |
| NDF_RTOL | Relative error tolerance for NDF; Real value > 0 |
| NDF_ATOL | Absolute error tolerance for NDF; Real value > 0 |
| T_START | Start time; Real value ≥ 0 |
| T_END | End time; Real value > 0 |
| DELTA_T | Initial time step size; Real value > 0 |
| DELTA_M | Monitoring interval; Real value > 0 |
| NR_TRIAL | Maximum trial times for "Steady-state analysis" (Newton-Raphson) ; Integer value > 0 |
| NR_TOL_F | Tolerance for convergence of functions for "Steady-state analysis" (Newton-Raphson) ; Real value > 0 |
| NR_TOL_X | Tolerance for convergence of variables for "Steady-state analysis" (Newton-Raphson) ; Real value > 0 |
| NR_RATIO | Ratio of changing parameters for "Steady-state analysis" (Newton-Raphson) ; Real value > 0 |
| NR_SENS_CW | Change width calculation sensitivity for "Steady-state analysis" ; Real value > 0 |
| G_VALUE | Value for one molecule concentration in a cell;<br>Real value > 0 |
| Y_DEFAULT | Default values for y (molecular concentrations) ;<br>Real value > 0 |
| MASS | Mass matrix for "Dynamic analysis" ;<br>0: differential equation, 1: algebraic equation |

[Structure variables “Y_START”]

| Field | Meanings |
|---|---|
| value | Initial value; Real value $\geq$ 0 |
| tag | Name of variable distinguished location;<br>(name).(location) |

| variable | Meanings |
|---|---|
| MAXSIMTIME | Limit actual time (second); default: 5 minutes |
| T0 | Start time for ode15s options (Events) |

If the simulation is stiff, the calculation takes a long time or doesn’t finish. When the actual time of the calculation is MAXSIMTIME, the calculation forcibly finishes even if stiff simulation.

### 1.6.2 CADLIVE_param.m

This file is written as information for events (“event”) and (kinetic) constant parameters (“param”). The events indicate changes in parameter values in a given time. That can indicate, for example, environmental changes to stimuli.

[Structure variables “event”]

| Field | Meanings |
|---|---|
| name | Field name of “param”, i.e. constantPlayer, Kb, Q etc., for events.<br>time_start and time_end have to be fixed at first and last index, respectively. |
| index | Index for “param.(name)” |
| time | Time occurred an event; Real value $\geq$ 0 |
| value | Value for “param.(name)” on the time;<br>Real value $\geq$ 0 |

*Events need to be registered in the ascending order of time between time_start and time_end.

Example for events:

```
% T_EVENT
    event(1).name = 'time_start';
    event(1).index = 0;
    event(1).time = 0.000000;
    event(1).value = 0.000000;
    event(2).name = 'kx';
    event(2).index = 2;
    event(2).time = 60.000000;
    event(2).value = 150.000000;
    event(3).name = 'kp';
    event(3).index = 1;
    event(3).time = 60.000000;
    event(3).value = 80.000000;
    event(4).name = 'time_end';
    event(4).index = 0;
    event(4).time = 100.000000;
    event(4).value = 0.000000;
```

[Structure variable “param”]

| Field | Meanings |
|---|---|
| constantPlayer | Independent variable |
| Ka | Binding association rate constant |
| Kx | Reaction rate constant |
| Kb | Binding constant |
| Kp | Translation rate constant |
| Kpd | Decomposition rate constant |
| Km | Transcription rate constant |
| Kmd | Decomposition rate constant |
| Ktr | Transport rate constant |
| Kxg | Reaction rate constant |
| F | Power coefficient (Target variable) |
| Q | Reaction rate constant |
| Kmich | Michaelis constant |

*These fields are defined by structure variables. For CMA and TPP, ka, kd, kx, Kb, kp, kpd, km, kmd and ktr are utilized. For GMA, kxg and f are utilized. For MM, Q and Kmich are utilized.

[Structure variable “param.(name)”]

| Field | Meanings |
|---|---|
| value | Value for the constant; Real value ≥ 0 |
| num_survey | Number of parameter survey; Integer ≥ 0 |
| d_r_s | D: parameter survey by arithmetic series<br>R: parameter survey by geometric series<br>S: parameter search for GA or TPS |
| tag | Name of the constant |
| change_val | Change for parameter survey; Real value ≥ 0 |
| upperBound | Upper bound of parameter search for GA in section 2 or TPS in section 3; Real value ≥ 0 |
| lowerBound | Lower bound of parameter search for GA in section 2 or TPS in section 3; Real value ≥ 0 |

*The Fields, change_val, upperBound and lowerBound, are written by users.

¥ Parameter survey ¥

When checking the checkbox of parameter survey on the “Select Analysis Type & Set Control Data for Simulation” window, “num_survey” and “change_val” need to be more than zero at least one parameter. For example, parameter survey is executed as follows.

1) Parameter survey by arithmetic series

When the fields of “param.(name)” are r_d_s=‘D’; num_survey=3; change_val = 1-e3; value=1e-3; , the three cases for {1e-3, 2e-3, 3e-3} are calculated.

2) Parameter survey by geometric series

When the fields of “param.(name)” are r_d_s=‘R’; num_survey=3; change_val = 1-e3; value=1e-3; , the three cases for {1e-3, 1e-6, 1e-9} are calculated.

## 1.7 Results

By clicking the "Execute" button on the "Set Parameters and Initial Values" window, the simulation starts and the "Results" window is displayed if the simulation is successful. If the simulation fails, the error message is displayed on the MATLAB command window.

The window for the result depends on analysis types; dynamic analysis, steady-state analysis and S-system.

### 1.7.1 Dynamic analysis

The window for "Dynamic analysis" is displayed as shown in **Fig.16**. The table indicates the temporal data of the simulation. By clicking the "Show graph" button, the graphs of the simulation are displayed (**Fig.17**). The variables to be displayed at the graph can be set at CADLIVE_DispFigure.m, where users input the indices of variables and graph type. By clicking the "setting" button, CADLIVE_DispFigure.m is opened (**Fig.18**). The "Reset" button sets the setting for the graph to default. CADLIVE_DispFigure.m is copied in the current folder. By clicking the "Export CSV" button, the result of the simulation is saved as a CSV file (**Fig.19**). By clicking the "Export MATLAB" button, the result of the simulation is saved as a mat file. By clicking the "Save for input" button, the concentrations at the final time of the simulation are saved as "CADLIVE_saveInitial.m", which can be used as the initial values for the next simulation and is written the same format as "CADLIVE_initail.m". By clicking the "QMPS" button, "SetQMPS" is opened (**Fig.20**), where users can calculate QMPS using the simulation results.

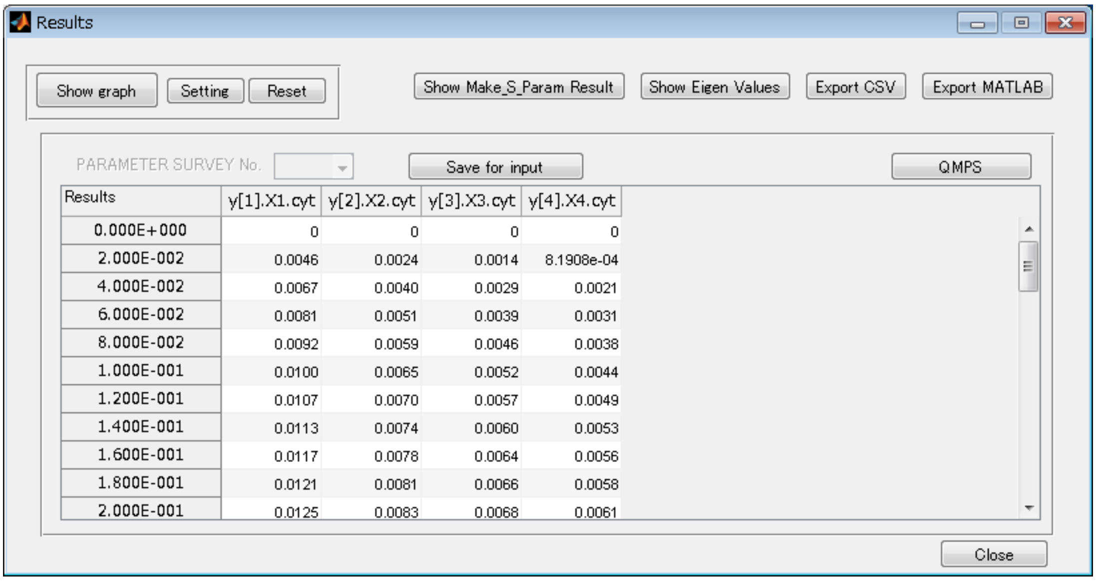
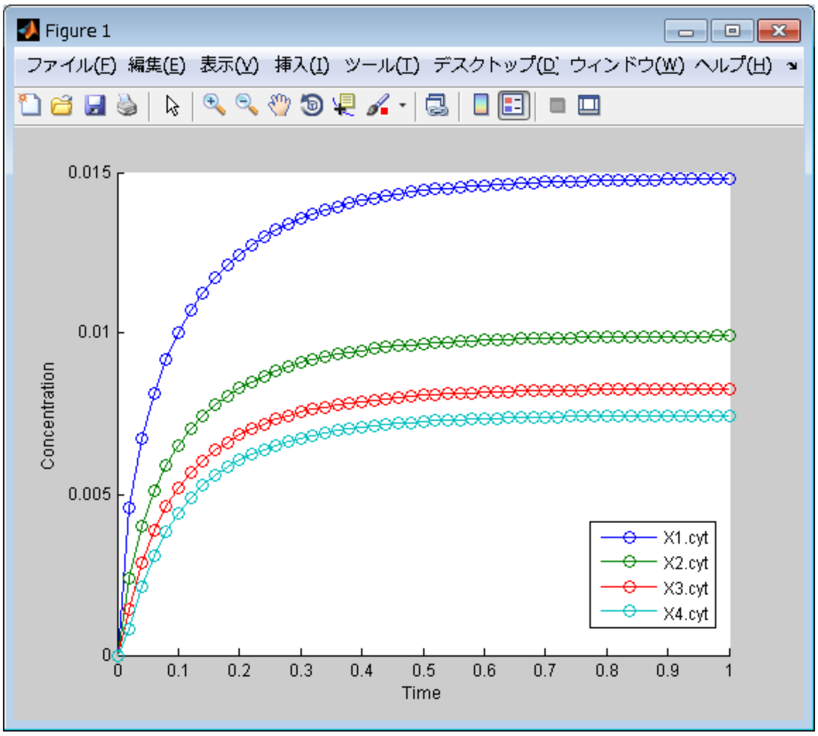


Fig.16 Results window for dynamic analysis



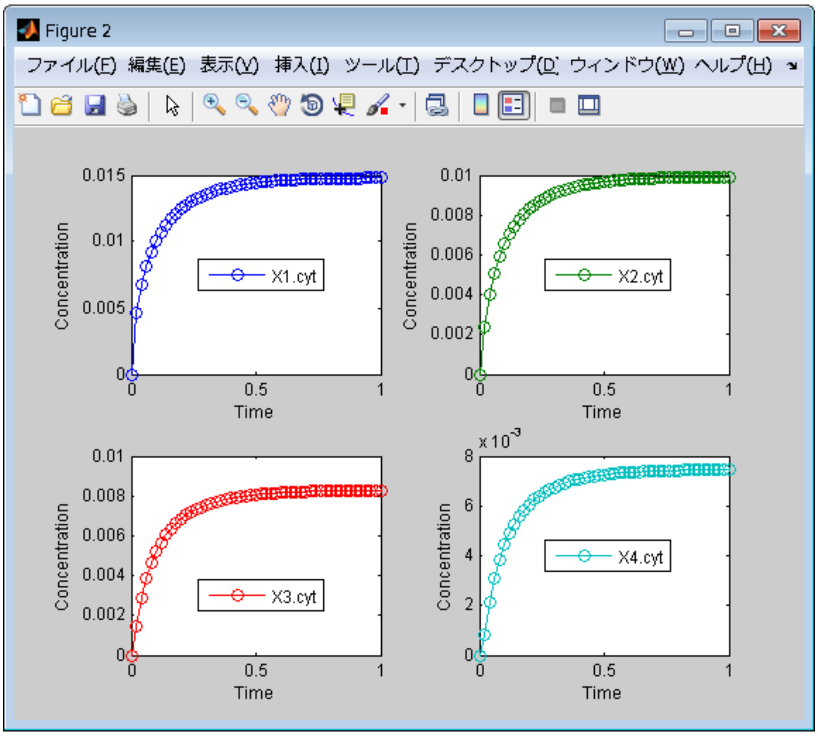




Fig.17 Graphs for simulations

Top: graph for type 1, Bottom: graph for type 2

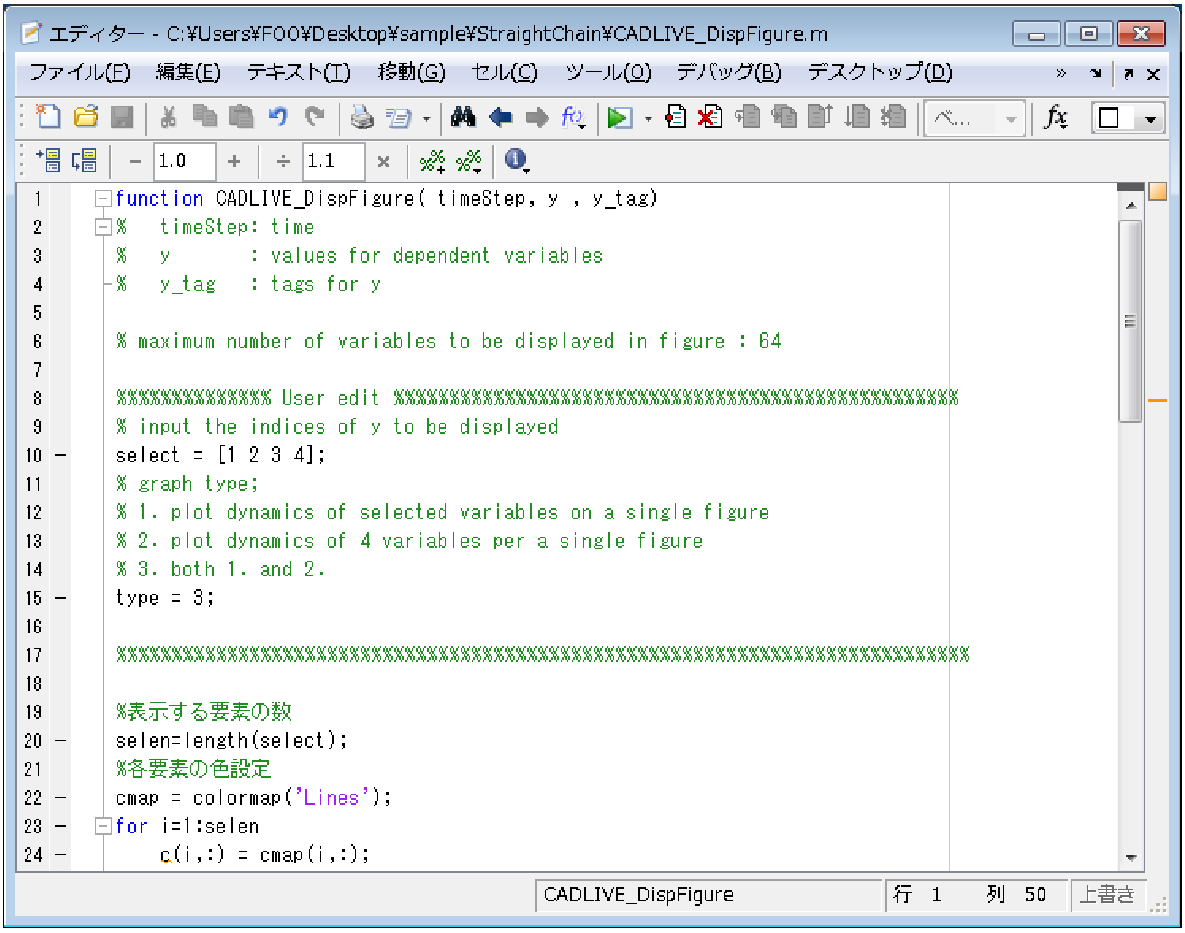
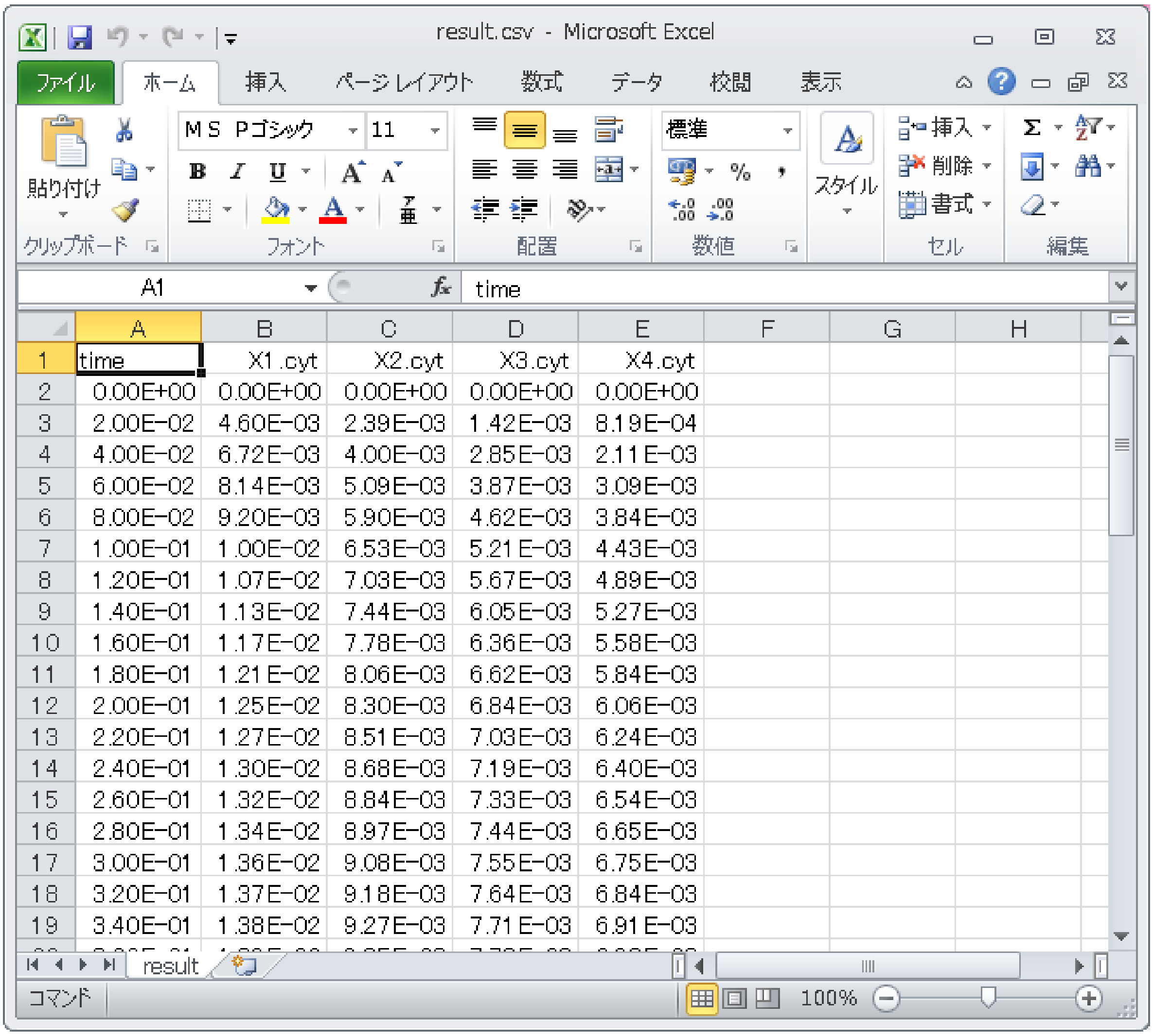



Fig.18 CADLIVE_DispFigure.m

Users edit the “select” and “type” below the “User edit”. The “select” indicates the indices of y to be displayed. The “type” indicates the number of graph type. 1 and 2 of “type” indicate the top and bottom figures in **Fig.17**, respectively.



| time | X1.cyt | X2.cyt | X3.cyt | X4.cyt |
|---|---|---|---|---|
| 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| 2.00E-02 | 4.60E-03 | 2.39E-03 | 1.42E-03 | 8.19E-04 |
| 4.00E-02 | 6.72E-03 | 4.00E-03 | 2.85E-03 | 2.11E-03 |
| 6.00E-02 | 8.14E-03 | 5.09E-03 | 3.87E-03 | 3.09E-03 |
| 8.00E-02 | 9.20E-03 | 5.90E-03 | 4.62E-03 | 3.84E-03 |
| 1.00E-01 | 1.00E-02 | 6.53E-03 | 5.21E-03 | 4.43E-03 |
| 1.20E-01 | 1.07E-02 | 7.03E-03 | 5.67E-03 | 4.89E-03 |
| 1.40E-01 | 1.13E-02 | 7.44E-03 | 6.05E-03 | 5.27E-03 |
| 1.60E-01 | 1.17E-02 | 7.78E-03 | 6.36E-03 | 5.58E-03 |
| 1.80E-01 | 1.21E-02 | 8.06E-03 | 6.62E-03 | 5.84E-03 |
| 2.00E-01 | 1.25E-02 | 8.30E-03 | 6.84E-03 | 6.06E-03 |
| 2.20E-01 | 1.27E-02 | 8.51E-03 | 7.03E-03 | 6.24E-03 |
| 2.40E-01 | 1.30E-02 | 8.68E-03 | 7.19E-03 | 6.40E-03 |
| 2.60E-01 | 1.32E-02 | 8.84E-03 | 7.33E-03 | 6.54E-03 |
| 2.80E-01 | 1.34E-02 | 8.97E-03 | 7.44E-03 | 6.65E-03 |
| 3.00E-01 | 1.36E-02 | 9.08E-03 | 7.55E-03 | 6.75E-03 |
| 3.20E-01 | 1.37E-02 | 9.18E-03 | 7.64E-03 | 6.84E-03 |
| 3.40E-01 | 1.38E-02 | 9.27E-03 | 7.71E-03 | 6.91E-03 |

Fig.19 Excel data exported

[Variables exported by “Export MATLAB”]

| Variable | Meanings |
|---|---|
| Eigen | Eigen values |
| TimeStep | Time |
| tag | Name of the variables |
| y | Dynamics of each dependent variable (each column indicates each dependent variable) |

QMPS

The “Delta” indicates the change in parameters when calculating the QMPS. By clicking the “Target function” button, the file CADLIVE_setTarget.m is opened (**Fig.21**) if the file exists in the current folder. If the file does not exist, the file is newly made and opened. The target function is defined as a particular function such as oscillation, for sensitivity in the biological system. Users write a target function below “%input a target function for QMPS below” according to the MATLAB language. By clicking the “Execute” button, the QMPS is calculated. The result is displayed in “QMPS” of the “SetQMPS” window (**Fig.22)**. If some target functions are set, users can select the target number of “Target No.” and display its QMPS.

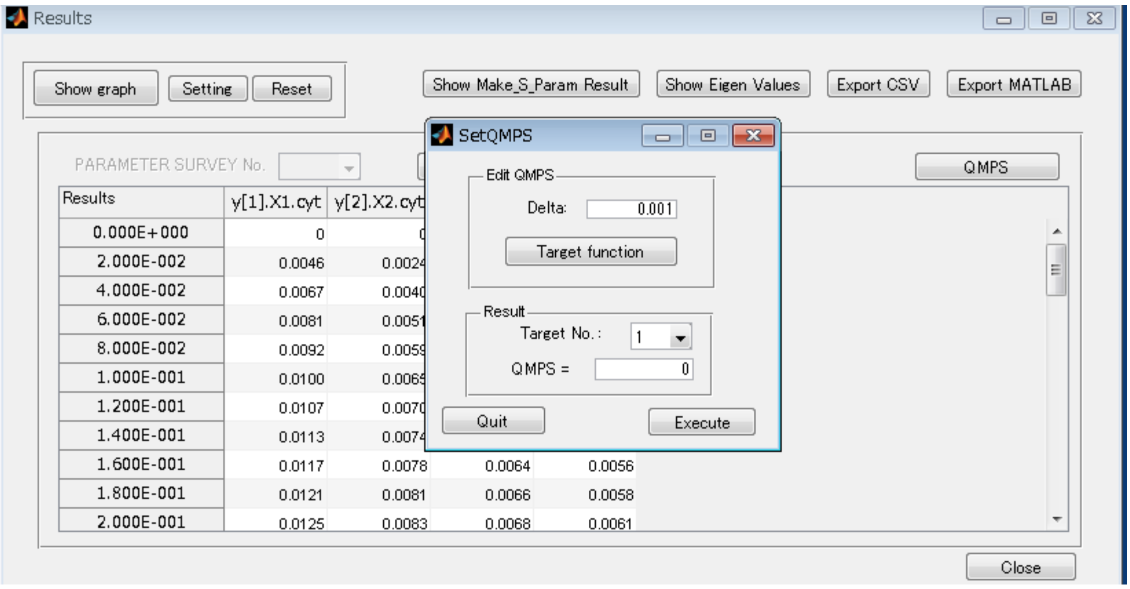



Fig.20 SetQMPS window

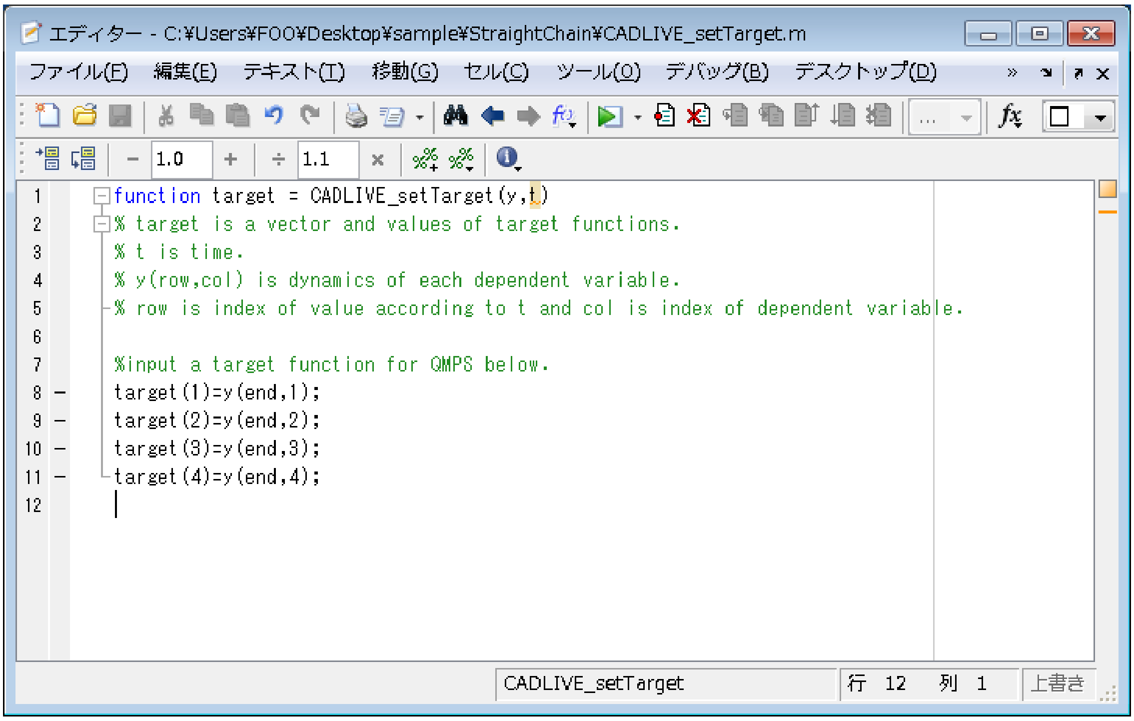



Fig.21 CADLIVE_setTarget.m

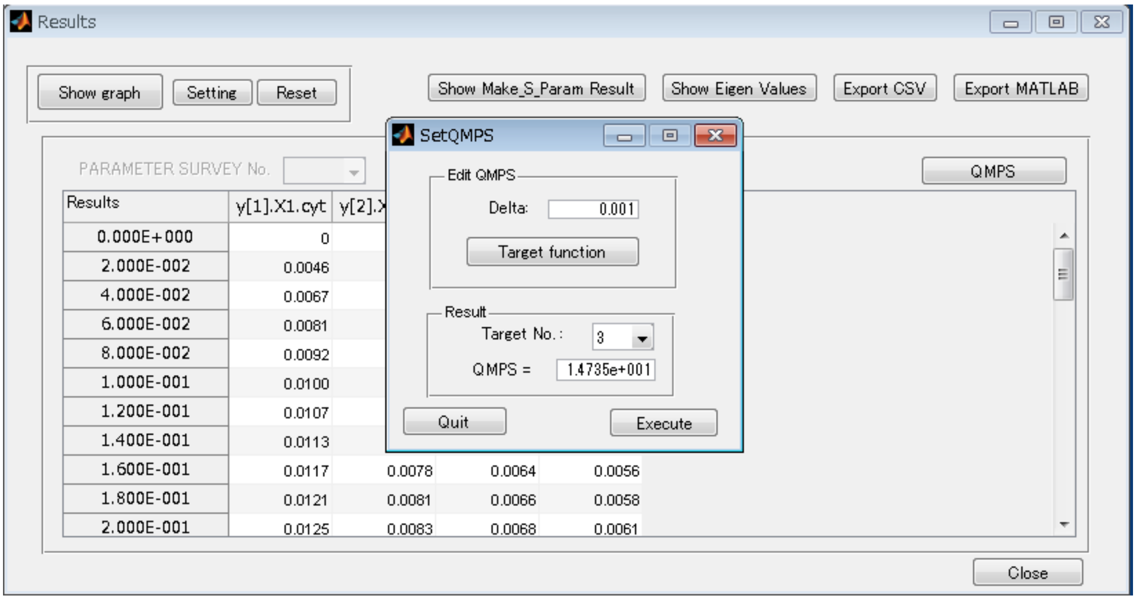



Fig.22 Result of QMPS in dynamic analysis

### 1.7.2 Steady-state analysis

The window for “Steady-state analysis” is displayed as shown in **Fig.23**. The “Results” on the window show the steady-state values. The “Sensitivity” on the window indicates the sensitivities with respect to each parameter and QMPSs. In this case, the target functions of QMPS are the steady-state values of each dependent variable y. By clicking the “Show Make_S_Param Result” button, the parameters for S-system are displayed (**Fig.24**). By clicking the “Show Eigen Values” button, the eigen values are displayed (**Fig.25**).

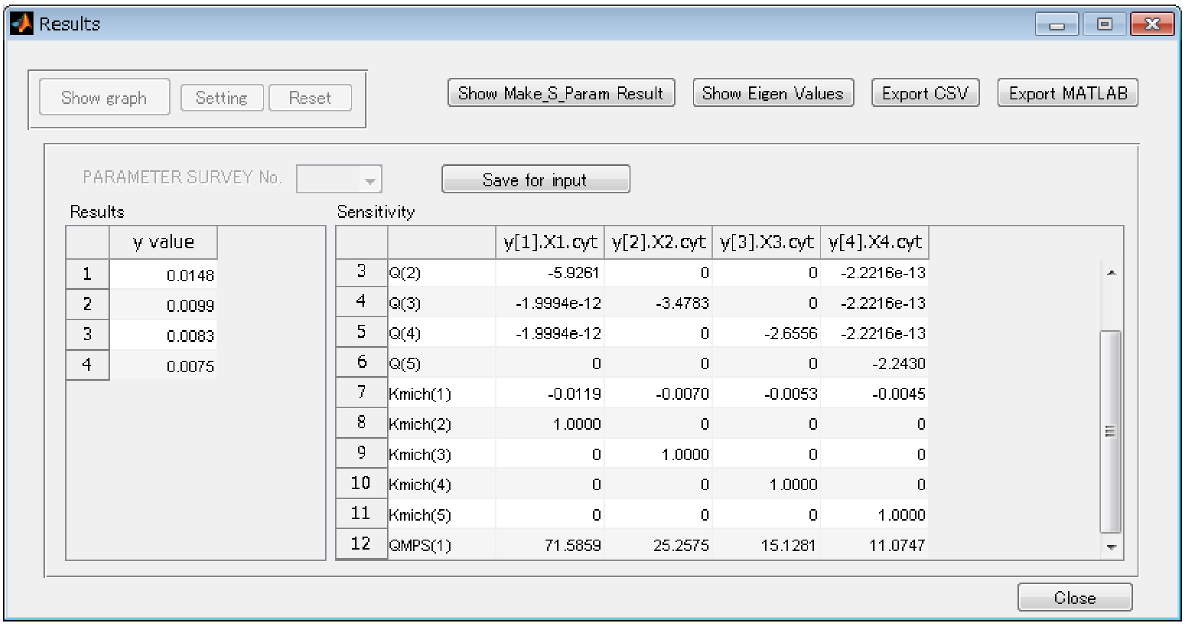



Fig.23 Results window for steady-state analysis

[Variables exported by “Export MATLAB”]

| Variable | Meanings |
|---|---|
| Eigen | Eigen values |
| Sensitivity | Sensitivities with respect to each parameter and QMPSs |
| tag | Name of the variables |
| y | Steady state values |

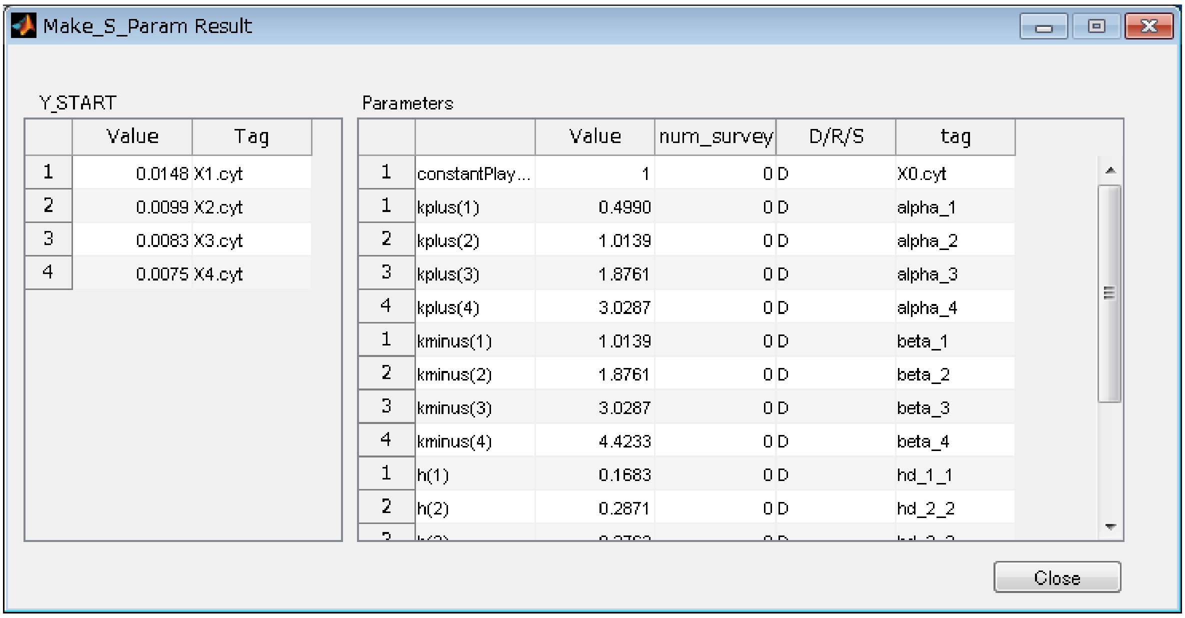



Fig.24 Paramters of the s-system converted

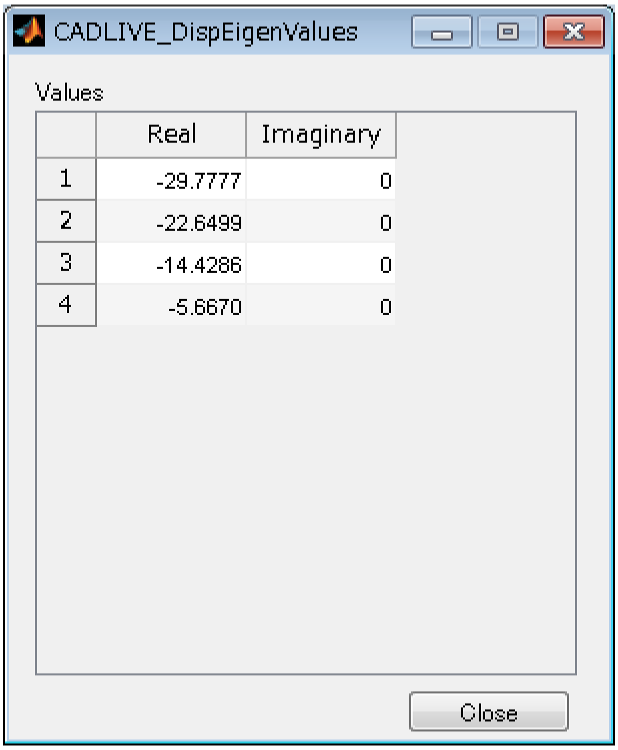



Fig.25 Eigen values

### 1.7.3 S-system

For the sensitivity analysis in S-system, dynamic analysis is first performed as described above (**1.7.1**). By clicking the “Show Make_S_Param Result” button in the results window for dynamic analysis (**Fig.16**), the parameters for S-system are displayed (**Fig.24**). By clicking ‘Back’ on the “Set Parameters and Initial Values” window (**Fig.9**), “Select Analysis Type & Set Control Data for Simulation” window is re-displayed (**Fig.8**). Users check the ‘Use S-system’ in the check box. By clicking the “Next” button, users can execute the simulation. The window for “S-system” is displayed as shown in **Fig.26**, where the sensitivities or logarithmic gains with respect to each parameter are displayed.

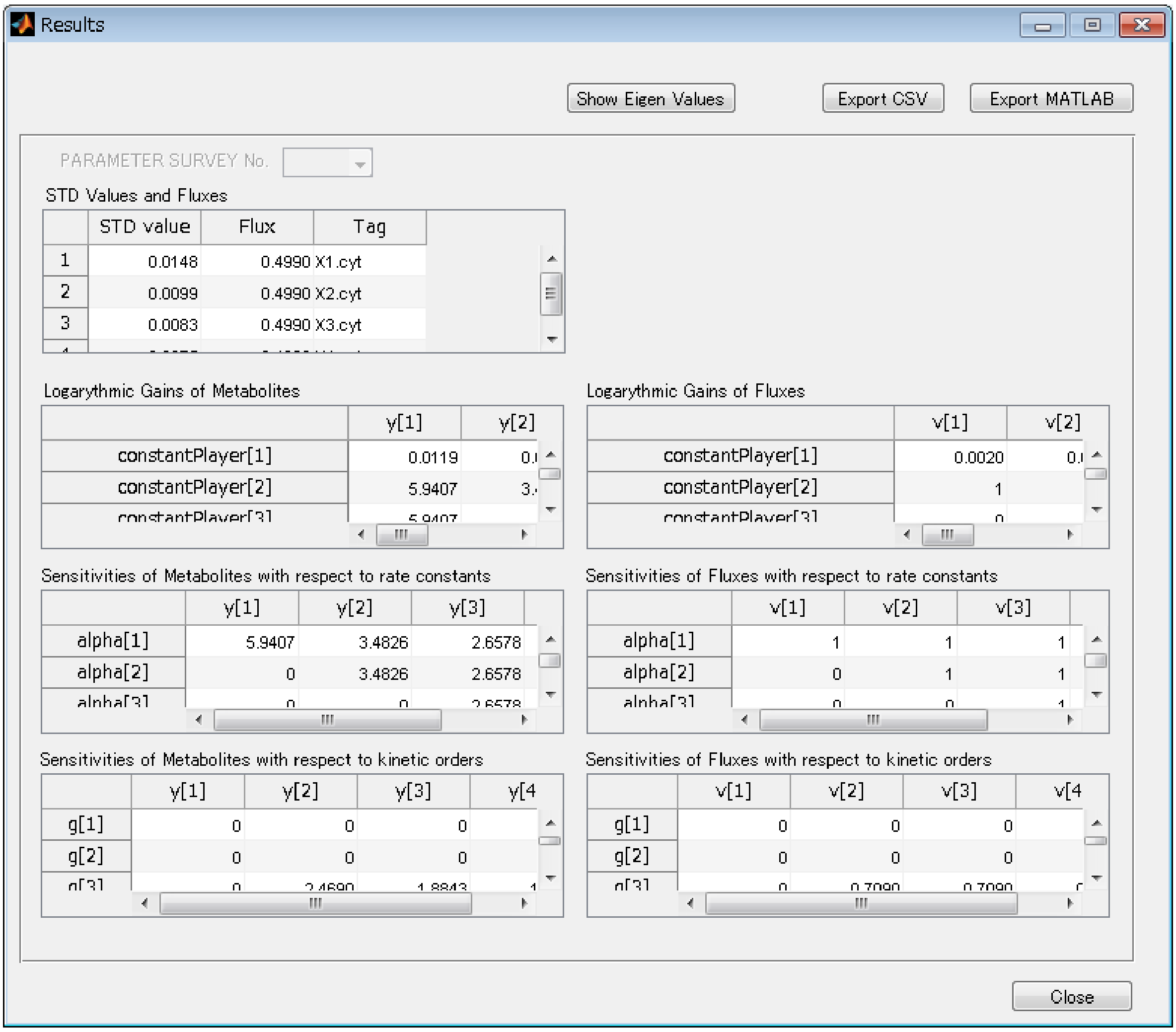



Fig.26 Results window for S-system

## 2 GA

Genetic algorithms (GAs) are known as one of the algorithms that can seek out the global minimum, based on the heuristic assumptions that the best solutions will be found in the regions of the parameter space containing a relatively high proportion of good solutions and that these regions can be explored by the genetic operators of selection, crossover, and mutation. For GA of this application (**Fig.27**), the unimodal normal distribution crossover (UNDX) and minimal generation gap (MGG) are employed as crossover and selection, respectively. The mutation is not employed.

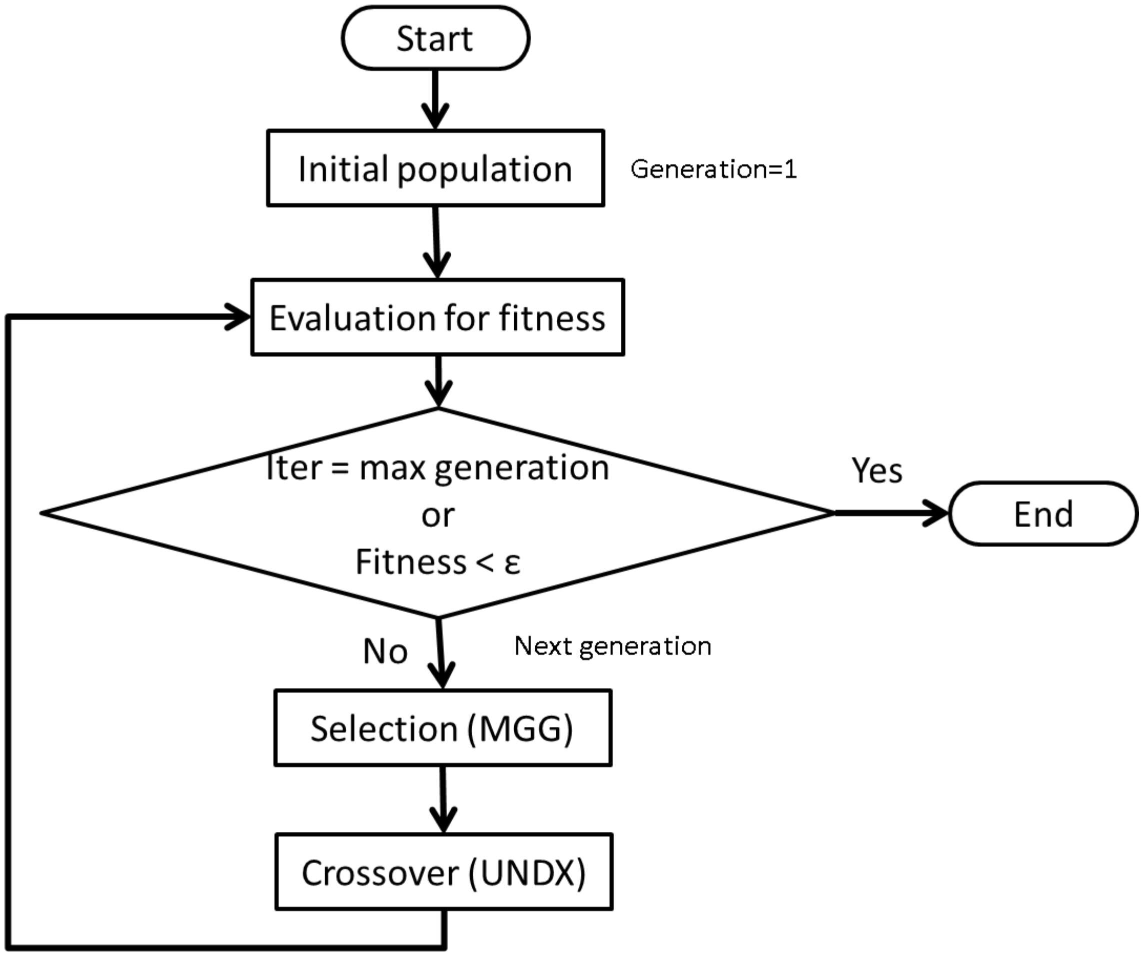



Fig.27 Flowchart for GA

## 2.1 Start

Execute "CADLIVE_SetGA" on the MATLAB command window to start the GA, the "CADLIVE_SetGA" window is displayed (**Fig.28**).

Users edit number of maximum generation, termination condition, number of population, number of children generated, search type, and alpha and beta in UNDX in "Edit GA condition". Analysis type indicates the analysis type for model of the current folder, "Steady-state analysis", "Dynamic Analysis" or "S-system". If users change the analysis type, they need to change the analysis type in section 1.5.

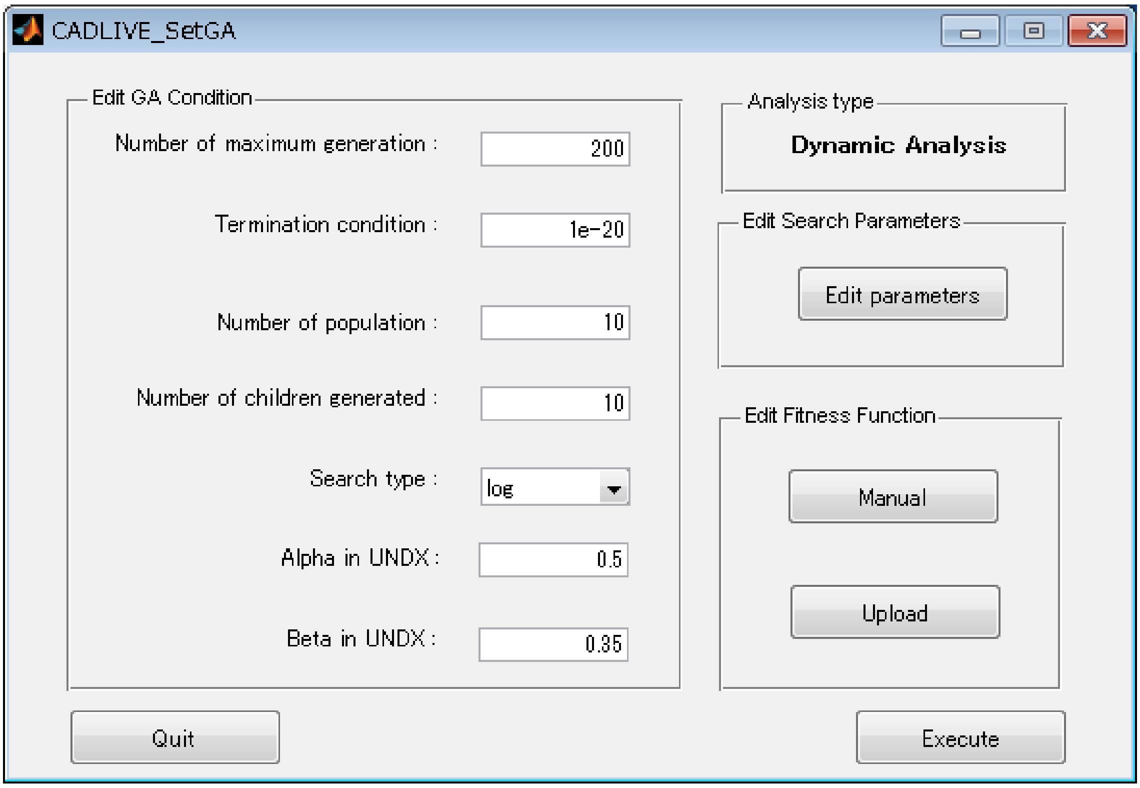



Fig.28 CADLIVE_SetGA window

*To execute the search by GAs, users need to input the initial values and parameters in section 1.5. The parameters can be edited by clicking the "Edit parameters" button.

Table.1 Values that users are allowed to set with respect to each parameter.

| Parameter | Values |
|---|---|
| Number of maximum generation | Integer $\geq$ 1 |
| Termination condition | Real value $\geq$ 0 |
| Number of population | Integer $\geq$ 1 |
| Number of children generated | Integer $\geq$ 1 |
| Search type | log, real |
| Alpha in UNDX | Real value > 0 |
| Beta in UNDX | Real value > 0 |

## 2.2 Search parameter setting

By clicking the “Edit parameters” button, CADLIVE_param.m is opened in the MATLAB editor (**Fig.29**). Users select the search parameters, which are optimized by GA. The search parameters are written as follows;

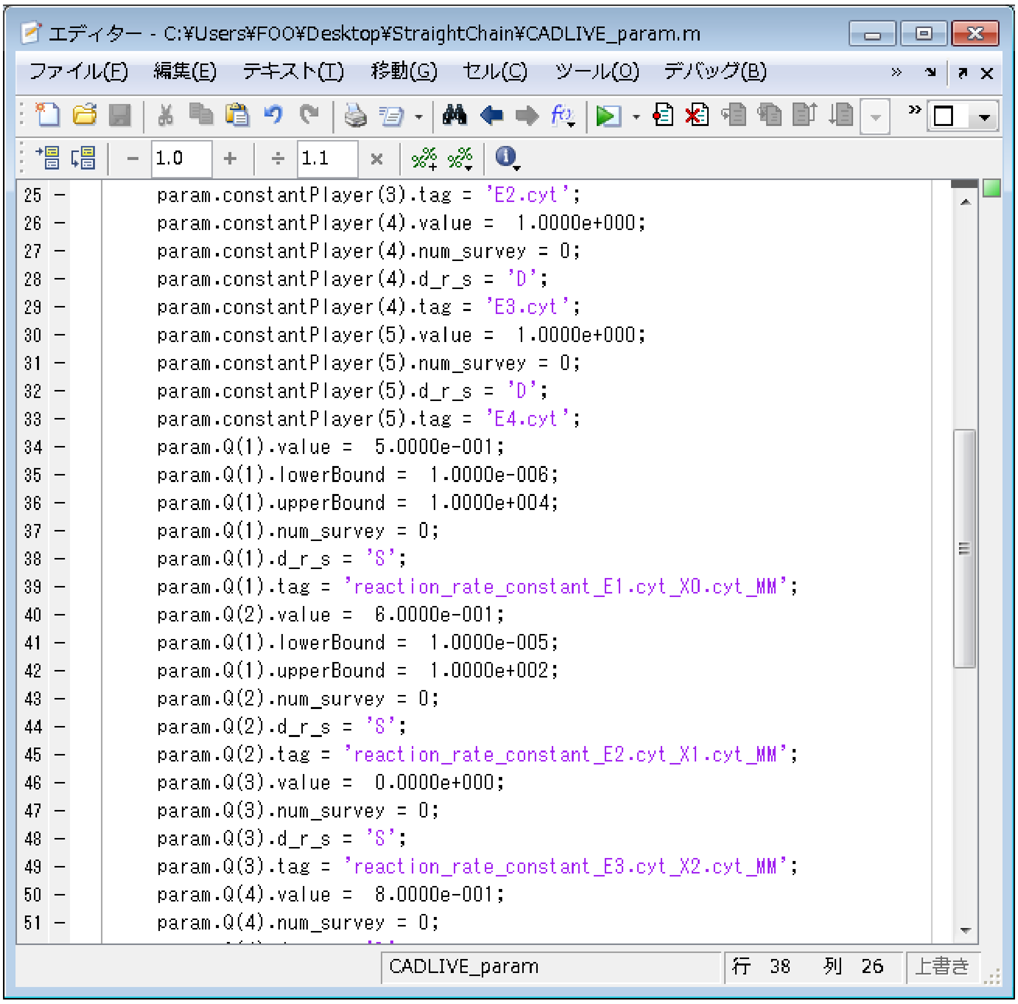
param.(name)(index).d_r_s = ‘S’;

‘S’ is a search parameter. Then, users set the search ranges (the lower and upper bounds) of the search parameters.

param.(name)(index).lowerBound = 1e-6;

param.(name)(index).upperBound = 1e+4;

When the search ranges are not set, they are automatically set param.(name)(index).vaule×0.1 and param.(name)(index).vaule×10 as the lower and upper bounds, respectively. When param.(name)(index).value = 0, the search range is $10^{-5}$-$10^{5}$.




```
param.constantPlayer(3).tag = 'E2.cyt';
param.constantPlayer(4).value =  1.0000e+000;
param.constantPlayer(4).num_survey = 0;
param.constantPlayer(4).d_r_s = 'D';
param.constantPlayer(4).tag = 'E3.cyt';
param.constantPlayer(5).value =  1.0000e+000;
param.constantPlayer(5).num_survey = 0;
param.constantPlayer(5).d_r_s = 'D';
param.constantPlayer(5).tag = 'E4.cyt';
param.Q(1).value =  5.0000e-001;
param.Q(1).lowerBound =  1.0000e-006;
param.Q(1).upperBound =  1.0000e+004;
param.Q(1).num_survey = 0;
param.Q(1).d_r_s = 'S';
param.Q(1).tag = 'reaction_rate_constant_E1.cyt_X0.cyt_MM';
param.Q(2).value =  6.0000e-001;
param.Q(1).lowerBound =  1.0000e-005;
param.Q(1).upperBound =  1.0000e+002;
param.Q(2).num_survey = 0;
param.Q(2).d_r_s = 'S';
param.Q(2).tag = 'reaction_rate_constant_E2.cyt_X1.cyt_MM';
param.Q(3).value =  0.0000e+000;
param.Q(3).num_survey = 0;
param.Q(3).d_r_s = 'S';
param.Q(3).tag = 'reaction_rate_constant_E3.cyt_X2.cyt_MM';
param.Q(4).value =  8.0000e-001;
param.Q(4).num_survey = 0;
```

Fig.29 CADLIVE_param.m for set of search conditions

## 2.3 Fitness function setting

By clicking the “Manual” button, the file CADLIVE_getFitness.m is opened if the file exists in the current folder. If the file does not exist, the file is newly made and opened (**Fig.30**). CADLIVE_getFitness.m is a file for setting a fitness function. Users write a fitness function below “%input a fitness function below” according to the MATLAB language. In “Steady-state analysis”, “y(col)” is steady-state values. In “Dynamic Analysis”, “y(row,col)” indicates the time-dependent variable, where “row” is the monitoring index and “col” is index of the dependent variable. An index value in y corresponds to the index of “Y_START” in CADLIVE_initial.m. “t” is time. The fitness function is defined as the minimization problem of $fitness \geq 0$. $fitness = 0$ indicates that a set of the parameters is completely optimized with respect to the fitness function.

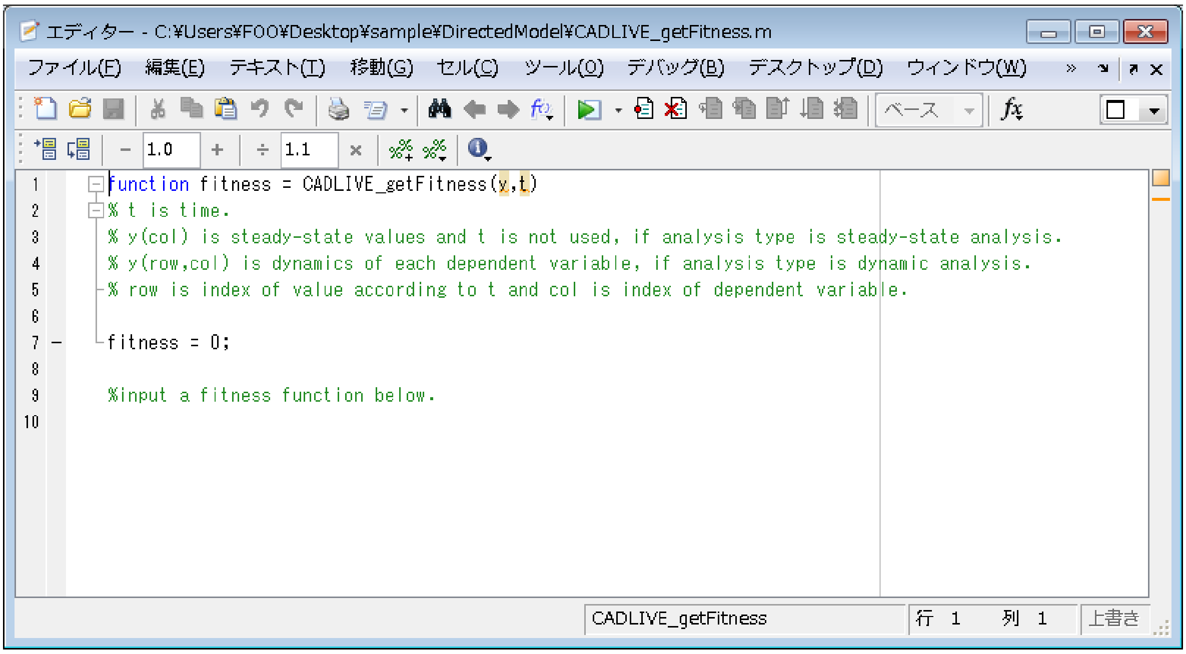


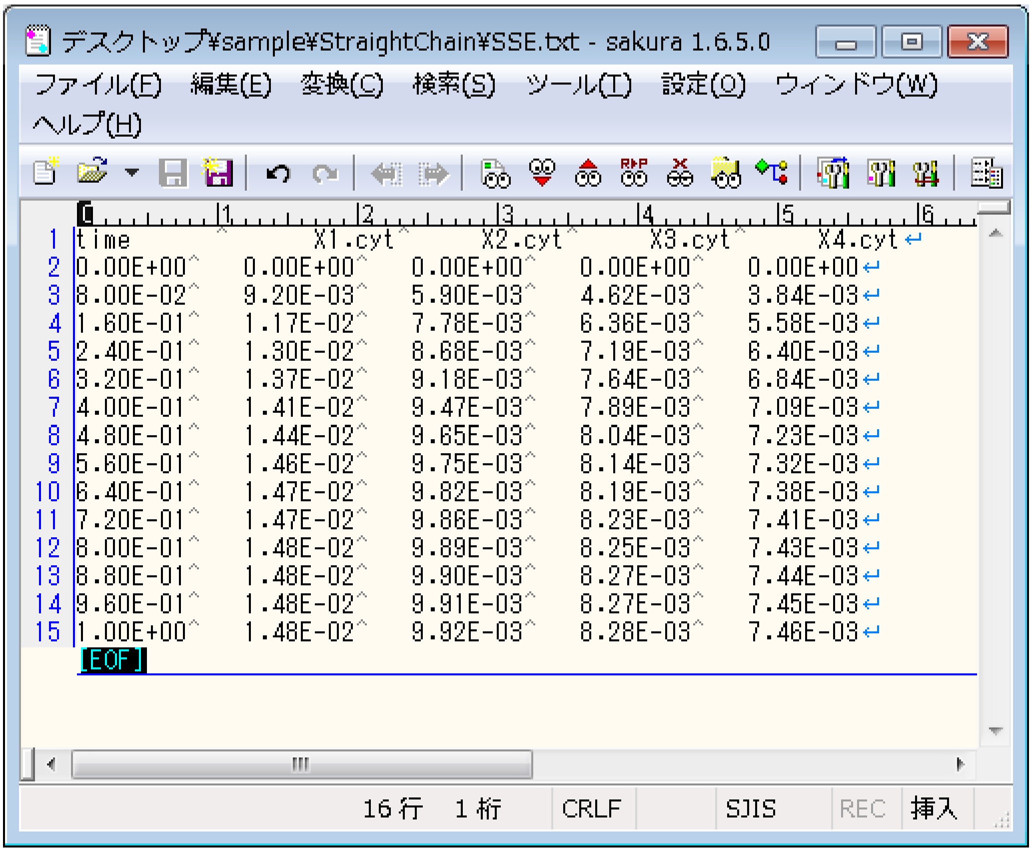

Fig.30 CADLIVE_getFitness.m

For “Dynamic Analysis”, users can set the fitness function as the sum of squared errors (SSE). Users make a file with time-course data (**Fig.31**) such as experimental data. The fitness function is set as the SSE (**Fig.32**). When clicking the “upload” button the file is uploaded. The SSE should be minimized for a parameter set $P$.

$$SSE(P) = \sum_{i=1}^{N}\sum_{j=1}^{k}\left(\frac{x_{ij}(P) - y_{ij}}{y_{ij}}\right)^2,$$

where $x_{ij}(P)$ are the simulated data corresponding to the experimental or reference data $y_{ij}$. $N$ is the number of molecules for optimization. $k$ is the number of the experimental data.

Here, the fitness function is defined as the SSE using the data (**Fig.32**).




| time | X1.cyt | X2.cyt | X3.cyt | X4.cyt |
|---|---|---|---|---|
| 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| 8.00E-02 | 9.20E-03 | 5.90E-03 | 4.62E-03 | 3.84E-03 |
| 1.60E-01 | 1.17E-02 | 7.78E-03 | 6.36E-03 | 5.58E-03 |
| 2.40E-01 | 1.30E-02 | 8.68E-03 | 7.19E-03 | 6.40E-03 |
| 3.20E-01 | 1.37E-02 | 9.18E-03 | 7.64E-03 | 6.84E-03 |
| 4.00E-01 | 1.41E-02 | 9.47E-03 | 7.89E-03 | 7.09E-03 |
| 4.80E-01 | 1.44E-02 | 9.65E-03 | 8.04E-03 | 7.23E-03 |
| 5.60E-01 | 1.46E-02 | 9.75E-03 | 8.14E-03 | 7.32E-03 |
| 6.40E-01 | 1.47E-02 | 9.82E-03 | 8.19E-03 | 7.38E-03 |
| 7.20E-01 | 1.47E-02 | 9.86E-03 | 8.23E-03 | 7.41E-03 |
| 8.00E-01 | 1.48E-02 | 9.89E-03 | 8.25E-03 | 7.43E-03 |
| 8.80E-01 | 1.48E-02 | 9.90E-03 | 8.27E-03 | 7.44E-03 |
| 9.60E-01 | 1.48E-02 | 9.91E-03 | 8.27E-03 | 7.45E-03 |
| 1.00E+00 | 1.48E-02 | 9.92E-03 | 8.28E-03 | 7.46E-03 |

[EOF]

Fig.31 File for SSE

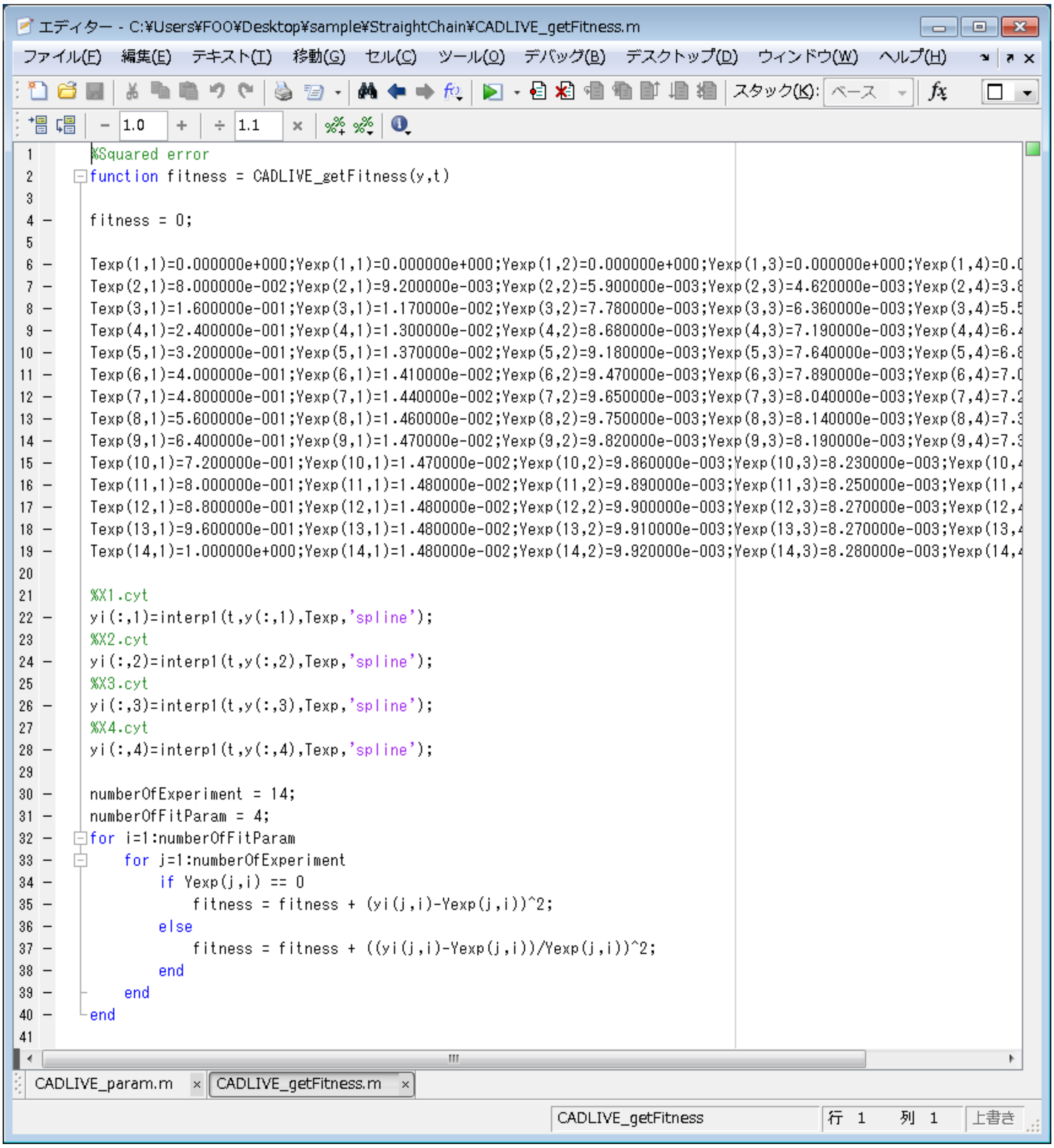



Fig.32 CADLIVE_getFitness.m input as SSE

## 2.4 Execution

By clicking the “Execute” button, the GA search is executed. During the optimization, the progress bar is displayed (**Fig.33**). The number indicates (number of generation calculated) / (maximum number of generation).

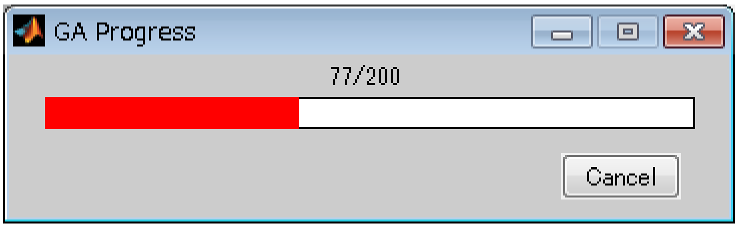



Fig.33 Progress of the GA search

When the optimization finishes, the graph for changes in the fitness with respect to generation (**Fig.34**), the value of best fitness, and the simulation result with the optimized parameters are displayed (**Fig.35**), and OptimizedParameter.mat and CADLIVE_OptParam.m are output. OptimizedParameter.mat is the data of the parameters optimized by GA, and CADLIVE_OptParam.m is the file for calling OptimizedParameter.mat. When the content in CADLIVE_OptParam.m is copied into CADLIVE_param.m, users can simulate the model with the optimized parameters.

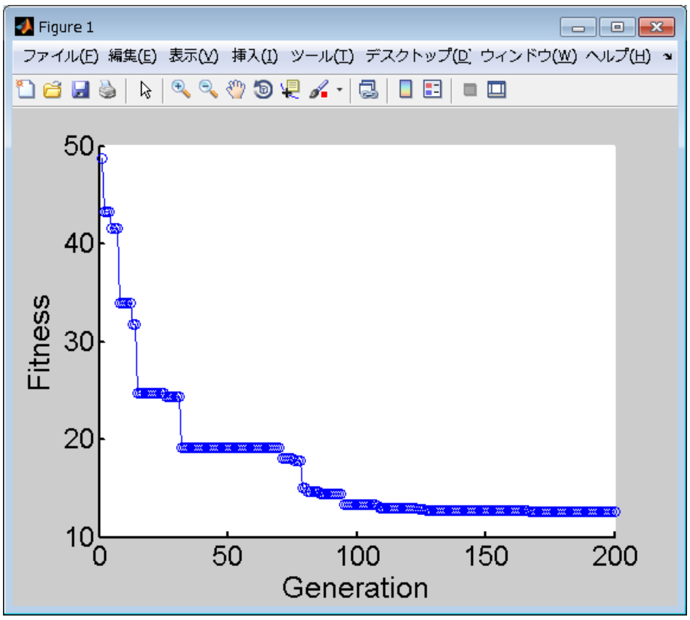
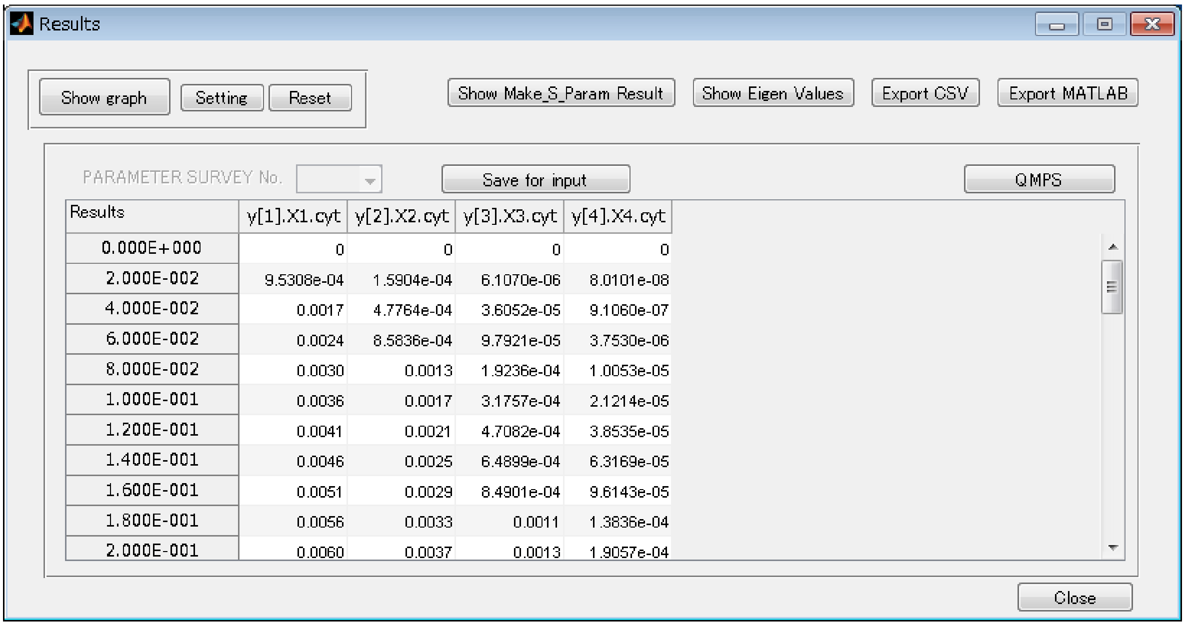



Fig.34 Fitness graph




| Results | y[1].X1.cyt | y[2].X2.cyt | y[3].X3.cyt | y[4].X4.cyt |
|---|---|---|---|---|
| 0.000E+000 | 0 | 0 | 0 | 0 |
| 2.000E-002 | 9.5308e-04 | 1.5904e-04 | 6.1070e-06 | 8.0101e-08 |
| 4.000E-002 | 0.0017 | 4.7764e-04 | 3.6052e-05 | 9.1060e-07 |
| 6.000E-002 | 0.0024 | 8.5836e-04 | 9.7921e-05 | 3.7530e-06 |
| 8.000E-002 | 0.0030 | 0.0013 | 1.9236e-04 | 1.0053e-05 |
| 1.000E-001 | 0.0036 | 0.0017 | 3.1757e-04 | 2.1214e-05 |
| 1.200E-001 | 0.0041 | 0.0021 | 4.7082e-04 | 3.8535e-05 |
| 1.400E-001 | 0.0046 | 0.0025 | 6.4899e-04 | 6.3169e-05 |
| 1.600E-001 | 0.0051 | 0.0029 | 8.4901e-04 | 9.6143e-05 |
| 1.800E-001 | 0.0056 | 0.0033 | 0.0011 | 1.3836e-04 |
| 2.000E-001 | 0.0060 | 0.0037 | 0.0013 | 1.9057e-04 |

Fig.35 Result for the simulation with optimized parameters

# 3 TPS

Two-Phase Search (TPS) smoothly combines a random search with an evolutionary algorithm to achieve both nonbiased and high-speed searches for a large parameter space (**Fig.36**). Use of QMPS with the TPS reveals the mechanism of how a particular architecture is related to robustness in complex regulations.

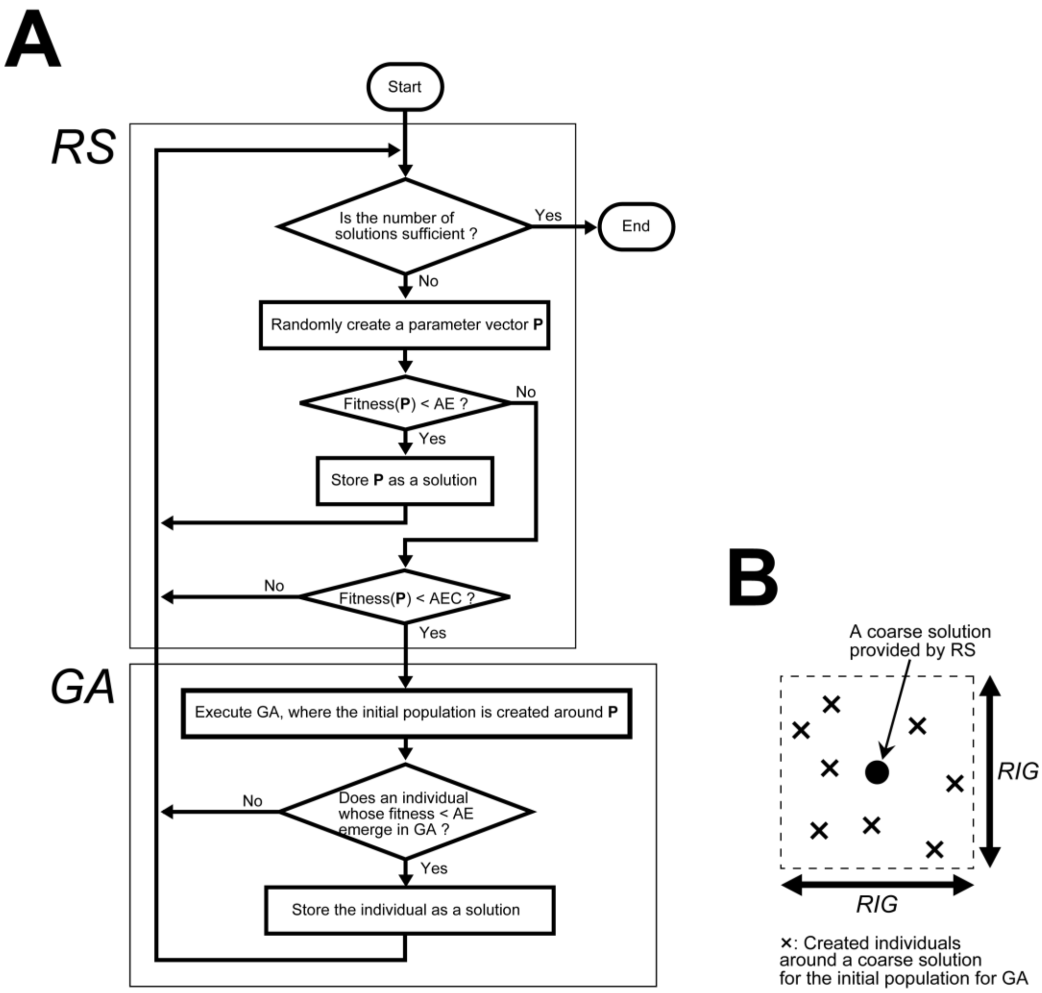



Fig.36 Schematic diagrams of the TPS method (Fig.1 in [6])
A: A flow chart for TPS that consists of a random search (the first phase) and a search by GAs (the second phase). B: How to create the initial populations for the second phase search by GAs. AE, AEC and RIG indicate the allowable error, the allowable error for the coarse solution, and the region of the initial population for the search by GAs, respectively.

## 3.1 Start

Execute "CADLIVE_SetTPS" on the MATLAB command window to start the TPS, the "CADLIVE_SetTPS" window is displayed (**Fig.37**). In "Dynamic Analysis", users can select whether QMPS is executed because QMPS may require a long calculation time. In "Steady-state Analysis", QMPS is always calculated.

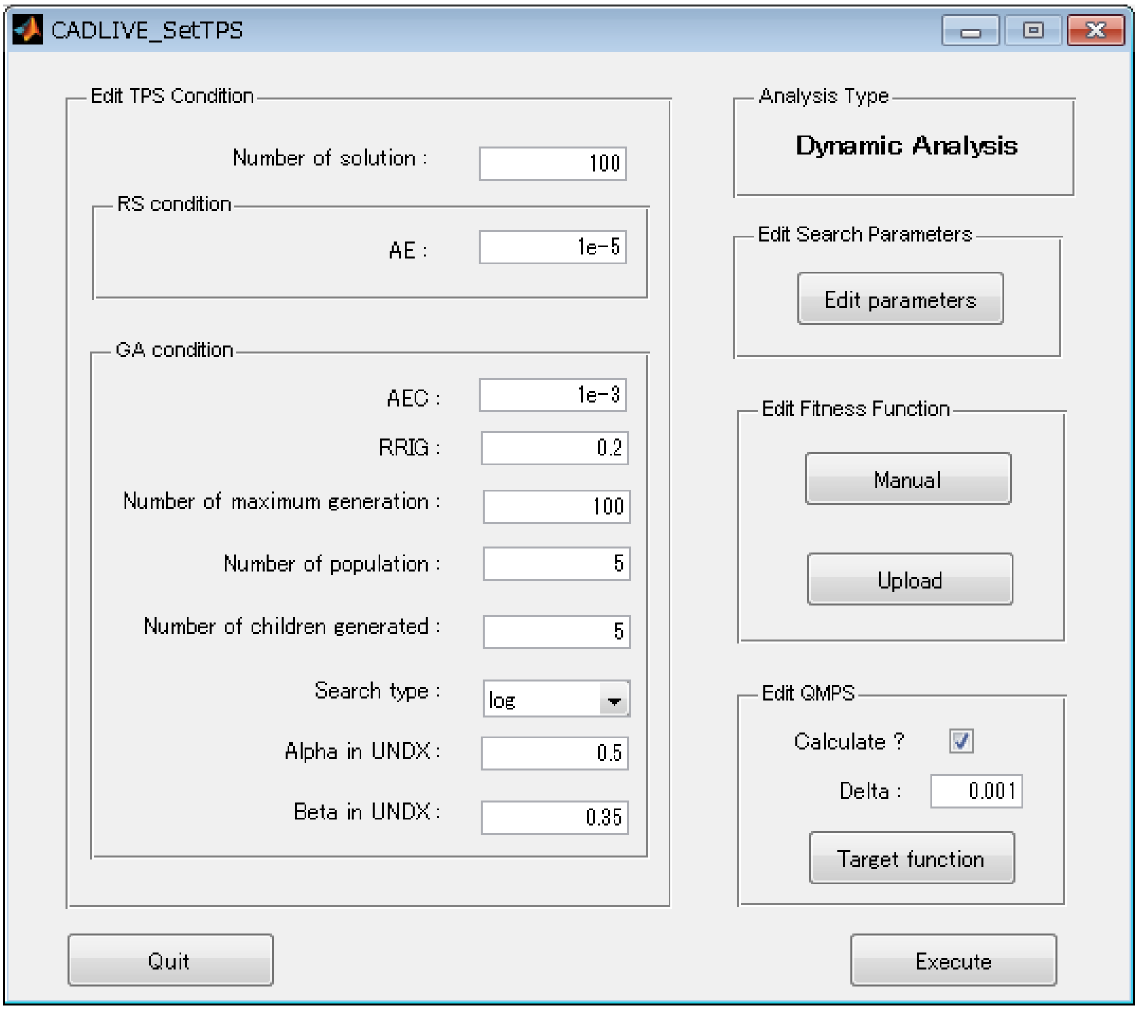



Fig.37 CADLIVE_SetTPS window

AE: Allowable error, AEC: Allowable error for the coarse solution, RRIG: Relative value of the region of the initial population for the search by GAs (RIG) to the search region of each parameter.

* To execute the TPS, users need to input the initial values and parameters in section 1.6. The parameters can be edited at the "Edit parameters" button.

Table2 Values that users are allowed to set with respect to each parameter.

| Parameter | Values |
|---|---|
| Number of solution | Integer $\geq$ 1 |
| AE | Real value $\geq$ 0 |
| AEC | Real value $\geq$ AE |
| RRIG | Real value > 0 |
| Number of maximum generation | Integer $\geq$ 1 |
| Termination condition | Real value $\geq$ 0 |
| Number of population | Integer $\geq$ 1 |
| Number of children generated | Integer $\geq$ 1 |
| Search type | log, real |
| Alpha in UNDX | Real value > 0 |
| Beta in UNDX | Real value > 0 |

Here, we set the objective function as follows:

$$\text{objective function} = \left( \frac{X4(t) - 5\times 10^{-3}}{5\times 10^{-3}} \right)^2,$$

where X4(t) is the value of X4 on the end time. The target functions are the values of X1, X2, X3 and X4 on the end time, shown as **Fig.21**.

## 3.2 QMPS on TPS

QMPS on TPS may require a long calculation time because it calculates QMPS with respect to many plausible solutions obtained by TPS. When QMPS is calculated, users check the checkbox beside “Calculate ?”. The “Delta” and “Target function” are described in section 1.7.1. In “Steady-state Analysis”, QMPS on TPS is always calculated.

### 3.3 Execution

By clicking the "Execute" button, TPS is executed. During the process, the progress bar is displayed (**Fig.38**). The number indicates (number of solution calculated) / (number of solution). When the calculation for TPS finishes, two files (TPSresultinfo.dat (**Fig.39**), TPSresult.mat) are output. When QMPS is calculated, the progress bar for QMPS is displayed (**Fig.40**). The number indicates (number of solution calculated) / (number of solution) and (number of single parameter sensitivity calculated) / (number of parameter) for QMPS. When the calculation for QMPS finishes, the file TPStoQMPS.mat is output and the cumulative frequency distribution for QMPS is displayed (**Fig.41**).

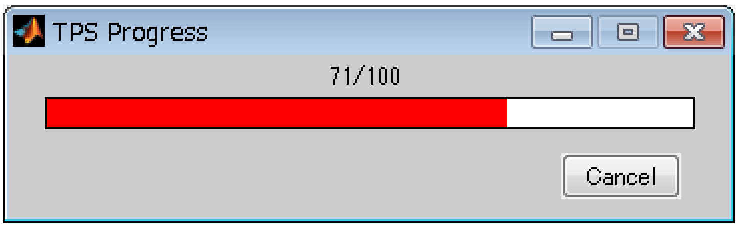
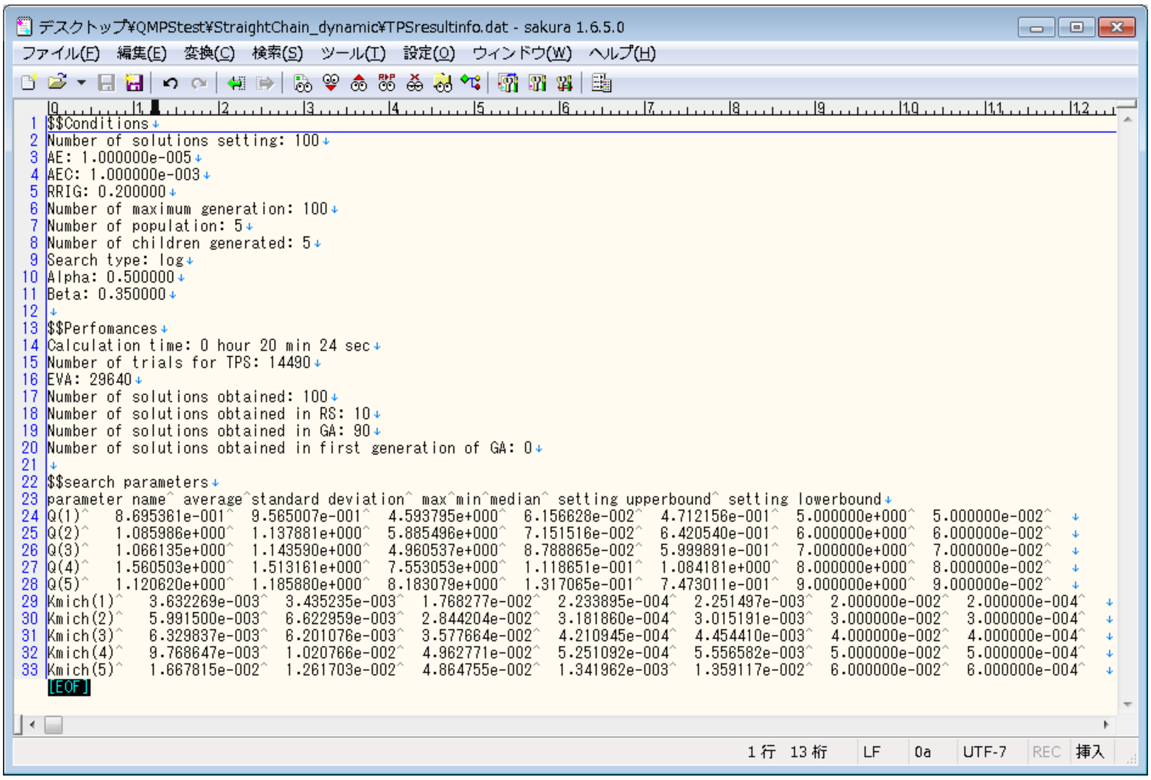



Fig.38 Progress of TPS

* If suitable parameters with respect to the fitness function are not searched or the TPS fails, users can check the fitness on the MATLAB command window, and can change the values of AE and AEC.



デスクトップ¥QMPStest¥StraightChain_dynamic¥TPSresultinfo.dat - sakura 1.6.5.0

```
$$Conditions
Number of solutions setting: 100
AE: 1.000000e-005
AEC: 1.000000e-003
RRIG: 0.200000
Number of maximum generation: 100
Number of population: 5
Number of children generated: 5
Search type: log
Alpha: 0.500000
Beta: 0.350000

$$Perfomances
Calculation time: 0 hour 20 min 24 sec
Number of trials for TPS: 14490
EVA: 29640
Number of solutions obtained: 100
Number of solutions obtained in RS: 10
Number of solutions obtained in GA: 90
Number of solutions obtained in first generation of GA: 0

$$search parameters
parameter name^ average^standard deviation^ max^min^median^ setting upperbound^ setting lowerbound
Q(1)^    8.695361e-001^  9.565007e-001^  4.593795e+000^  6.156628e-002^  4.712156e-001^  5.000000e+000^  5.000000e-002^
Q(2)^    1.085986e+000^  1.137881e+000^  5.885496e+000^  7.151516e-002^  6.420540e-001^  6.000000e+000^  6.000000e-002^
Q(3)^    1.066135e+000^  1.143590e+000^  4.960537e+000^  8.788865e-002^  5.999891e-001^  7.000000e+000^  7.000000e-002^
Q(4)^    1.560503e+000^  1.513161e+000^  7.553053e+000^  1.118651e-001^  1.084181e+000^  8.000000e+000^  8.000000e-002^
Q(5)^    1.120620e+000^  1.185880e+000^  8.183079e+000^  1.317065e-001^  7.473011e-001^  9.000000e+000^  9.000000e-002^
Kmich(1)^    3.632269e-003^  3.435235e-003^  1.768277e-002^  2.233895e-004^  2.251497e-003^  2.000000e-002^  2.000000e-004^
Kmich(2)^    5.991500e-003^  6.622959e-003^  2.844204e-002^  3.181860e-004^  3.015191e-003^  3.000000e-002^  3.000000e-004^
Kmich(3)^    6.329837e-003^  6.201076e-003^  3.577664e-002^  4.210945e-004^  4.454410e-003^  4.000000e-002^  4.000000e-004^
Kmich(4)^    9.768647e-003^  1.020766e-002^  4.962771e-002^  5.251092e-004^  5.556582e-003^  5.000000e-002^  5.000000e-004^
Kmich(5)^    1.667815e-002^  1.261703e-002^  4.864755e-002^  1.341962e-003^  1.359117e-002^  6.000000e-002^  6.000000e-004^
[EOF]
```

Fig.39 Result for TPS

The file TPSresultinfo.dat is written condition and performances for TPS, and statistics with regard to search parameters.

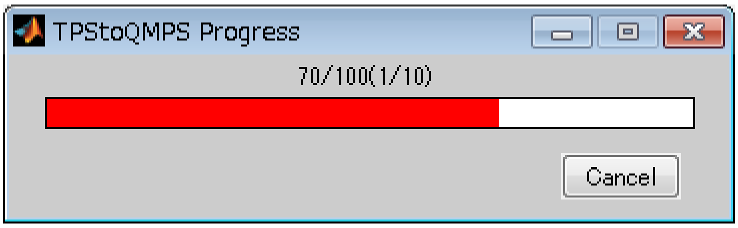
TPSresult.mat

[Structure variable “searchParam”]

| Field | Meanings |
|---|---|
| paramName | Parameter name |
| paramIndex | Index of paramName |
| upperBound | Upper bound of parameter search |
| lowerBound | Lower bound of parameter search |
| newValue | Ignore |

[Structure variable “solution”]

| Field | Meanings |
|---|---|
| fitness | Value of fitness |
| paramValue | Vector of the parameter with respect to index of searchParam |
| type | Method when the result is obtained<br>‘RS’: Random search<br>‘GA’: Genetic algorithm<br>‘GAinit’: First generation of GA |




Fug.40 Progress of QMPS on TPS

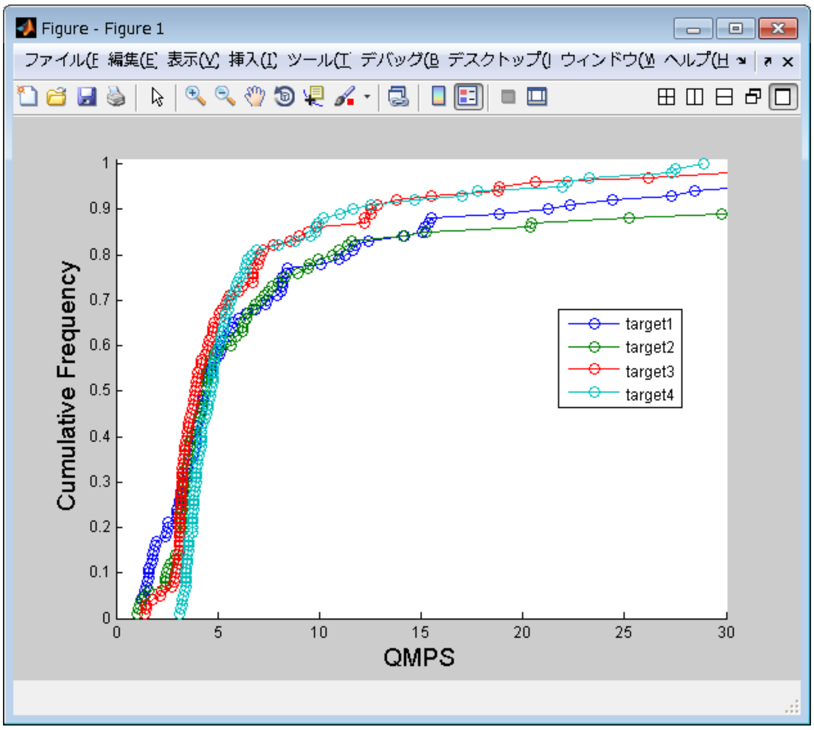



Fig.41 Cumulative frequency distribution of QMPS

TPStoQMPS.mat

| Variable | Meanings |
|---|---|
| QMPS | Value of QMPS |
| tag | Target number when “Dynamic Analysis”<br>Name of variable when “Steady-state Analysis” |

# 4 Execution on command line

Users can use the simulator, GA, TPS and QMPS without GUI.

## 4.1 CADLIVE_simulator

This is the command for executing simulation.

---

Preparations:

CADLIVE_initial.m, CADLIVE_param.m, CADLIVE_fvec.m and CADLIVE_fjac.m in the current folder.

Command:

```
SimResult = CADLIVE_simulator();
```

Return:

SimResult : Simulation result (structure variable)

---

[Structure variable “SimResult”]

| Field | Meanings |
|---|---|
| Result | Success or fail (true or false) |
| Result_data | Structure variable of simulation result according to analysis type<br>The details are described in follow Table. |
| Y | Final values after simulation |
| Y_pre | Initial values |
| tag | Name of each dependent variable |
| Eigen | Eigen values |
| S_Param | Parameters for S-system |

[Structure variable “Result_data” in dynamic analysis]

| Field | Meanings |
|---|---|
| time | Time |
| Y | Dynamics of each dependent variable |

[Structure variable “Result_data” in steady-state analysis]

| Field | Meanings |
|---|---|
| sensitivity | Sensitivity to each parameter and QMPS for the change of “NR_SENS_CW” of “CADLIVE_CTL” in section 1.6.1 |

[Structure variable “Result_data” in S-system analysis]

| Field | Meanings |
|---|---|
| STDValueFlux | STD Values and Fluxes |
| LogGainMetabo | Logarithmic Gains of Metabolites |
| SensMetaboRateConst | Sensitivities of Metabolites with respect to rate constants |
| SensMetaboKinOrder | Sensitivities of Metabolites with respect to kinetic orders |
| LogGainsFlux | Logarithmic Gains of Fluxes |
| SensFluxRateConst | Sensitivities of Fluxes with respect to rate constants |
| SensFluxKinOrder | Sensitivities of Fluxes with respect to kinetic orders |

When users simulate with change of initial values and/or parameters iteratively, users can use the following command.

---

Preparations:

CADLIVE_initial.m, CADLIVE_param.m, CADLIVE_fvec.m and CADLIVE_fjac.m in the current folder.

Command:

```
SimResult = CADLIVE_simulatorForIteration(CADLIVE_CTL, Y_START, param, event);
```

Return:

SimResult : Simulation result (structure variable)

Example:

```
[CADLIVE_CTL Y_START]=CADLIVE_initial();
[param, event]=CADLIVE_param();
for i=1:5
    Y_START(1).value = 0.1*i;
SimResult(i) = CADLIVE_simulatorForIteration(CADLIVE_CTL, Y_START, param, event);
end
```

---

## 4.2 CADLIVE_myGAcommand

This is the command for executing GA. If the search by GA is successful, OptimizedParameter.mat and CADLIVE_OptParam.m are output.

---

Preparations:

CADLIVE_initial.m, CADLIVE_param.m, CADLIVE_fvec.m, CADLIVE_fjac.m and CADLIVE_getFitness.m in the current folder.

Command:

```
[SimResult fitness] = CADLIVE_myGAcommand(...
    n_generation, n_population, n_children, allowable_error, searchType, alpha, beta);
```

Arguments:

| | | | |
|---|---|---|---|
| n_generation | : | Number of maximum generation | ; Integer $\geq$ 1 |
| n_population | : | Number of population | ; Integer $\geq$ 1 |
| n_children | : | Number of children generated | ; Integer $\geq$ 1 |
| allowable_error | : | Termination condition | ; Real value $\geq$ 0 |
| searchType | : | Search type | ; log, real |
| alpha | : | Alpha in UNDX | ; Real value > 0 |
| beta | : | Beta in UNDX | ; Real value > 0 |

Return:

| | | |
|---|---|---|
| SimResult | : | Simulation result with optimized parameters |
| fitness | : | Fitness values |

Example:

```
[SimResult fitness]= CADLIVE_myGAcommand(100, 5, 5, 1e-20, 'log', 0.5, 0.35);
```

---

## 4.3 CADLIVE_myTPScommand

This is the command for executing TPS. If the search by TPS is successful, two files (TPSresultinfo.dat (**Fig.39**), TPSresult.mat) are output.

---

Preparations:

CADLIVE_initial.m, CADLIVE_param.m, CADLIVE_fvec.m, CADLIVE_fjac.m and CADLIVE_getFitness.m in the current folder.

Command:

```
[solution searchParam info flag] = CADLIVE_myTPScommand(term_n_solution, …
AE, AEC, RRIG,...
                                                    n_generation, n_population, n_children, …
searchType, alpha, beta);
```

Arguments:

| | | | |
|---|---|---|---|
| term_n_solution | : | Number of solution | ; Integer ≥ 1 |
| AE | : | Allowable error for random search | ; Real value ≥ 0 |
| AEC | : | Allowable error for the coarse solution | ; Real value ≥ AE |
| RRIG | : | Relative value of the region of the initial population for the search by GAs (RIG) to the search region of each parameter | ; Real value ≥ 0 |
| n_generation | : | Number of maximum generation | ; Integer ≥ 1 |
| n_population | : | Number of population | ; Integer ≥ 1 |
| n_children | : | Number of children generated | ; Integer ≥ 1 |
| searchType | : | Search type | ; log, real |
| alpha | : | Alpha in UNDX | ; Real value > 0 |
| beta | : | Beta in UNDX | ; Real value > 0 |

Returns:

| | | |
|---|---|---|
| solution | : | Result (structure variable) |
| searchParam | : | Condition of search parameter (structure variable) |
| info | : | Information of the result (structure variable) |
| flag | : | Success or failed (flag=0 or flag <0) |

---

Example:

[solution searchParam info flag] = CADLIVE_myTPScommand(100, 1e-3, 1.01, 0.2,...
100, 5, 5, 'log', 0.5, 0.35);

[Structure variable “solution”]

| Field | Meanings |
|---|---|
| fitness | Value of fitness |
| paramValue | Values of search parameters |
| type | Search method obtained the result;<br>‘RS’, ‘GA’, ‘GAinit’ |

[Structure variable “searchParam”]

| Field | Meanings |
|---|---|
| paramName | Parameter name |
| paramIndex | Index of paramName |
| upperBound | Upper bound of parameter search |
| lowerBound | Lower bound of parameter search |
| newValue | Ignore |

[Structure variable “info”]

| Field | Meanings |
|---|---|
| n_trail | Number of trials for TPS |
| n_rs | Number of parameter set obtained in random search |
| n_solution | Total number of parameter set obtained by TPS |
| n_coarse_solution | Number of trials by GA in TPS |
| n_solution_obtained_in_GA | Number of parameter set obtained in GA |
| n_solution_obtained_in_first_generation | Number of parameter set obtained in first generation of GA |
| n_evalution | Number of evaluations for fitness function (EVA) |

*The indices of “paramValue” of solution correspond to those of “searchParam”.

## 4.4 CADLIVE_DispFigure

This is the command for displaying a figure of the result in dynamic analysis. CADLIVE_DispFigure.m is output in the current folder when “Dynamic analysis” is successful (section 1.7.1). When users execute the simulation on the MATLAB command window, the resultant simulation can be displayed by using this command (**Fig.42**).

---

Preparations:

SimResult = CADLIVE_simulator(); or load a file output by “Export MATLAB”

Command:

CADLIVE_DispFigure( timeStep, y , y_tag);

Arguments:

| | | |
|---|---|---|
| timeStep | : | Time |
| y | : | Dynamics of each dependent variable |
| y_tag | : | Names of each dependent variable |

Example:

CADLIVE_DispFigure(SimResult.Result_data.time, SimResult.Result_data.y, …
SimResult.tag);

---

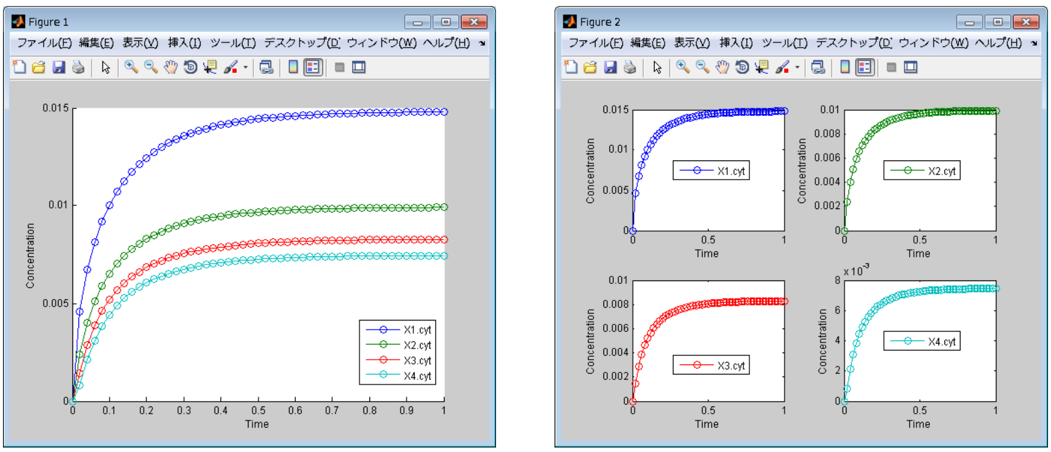



Fig.42 CADLIVE_DispFigure

## 4.5CADLIVE_DispCumulFreqQMPS, CADLIVE_DispHistFreqQMPS, CADLIVE_DispHistCumulFreqQMPS

These are the commands for displaying the QMPSs on TPS. In CADLIVE_SetTPS (GUI) in section 3, it displays the cumulative frequency distribution of QMPS on a certain state for the mathematical model by the command CADLIVE_DispCumulFreqQMPS. Using these commands on command line, it can simultaneously display the distributions for QMPSs among some models (e.g. wild-type model and knockout models or non-branching model and branching models), where robustness of them can be characterized by their QMPSs.

CADLIVE_DispCumulFreqQMPS displays a cumulative frequency distribution with all data points for the values of QMPS (**Fig.43**). CADLIVE_DispHistFreqQMPS displays a histogram for a frequency distribution (**Fig.44**). CADLIVE_DispHistCumulFreqQMPS displays a cumulative frequency distribution whose data points are smoothed by the values of the bin for histogram instead of the actual values of QMPS (**Fig.45**).

---

Preparations:

```
load TPStoQMPS.mat
```

Commands:

```
CADLIVE_DispCumulFreqQMPS(QMPS, tag);
CADLIVE_DispHistFreqQMPS(QMPS, tag, nbin, lb, ub);
CADLIVE_DispHistCumulFreqQMPS(QMPS, tag, nbin, lb, ub);
```

Arguments:

| | | |
|---|---|---|
| QMPS | : | QMPS; each column is corresponding to QMPS for each target |
| tag | : | Name of target function |
| nbin | : | Number of classes for the histograms |
| lb | : | Lower bound of the histograms |
| ub | : | Upper bound of the histograms |

Example:

```
cd (the folder for the StraightChain model);
load TPStoQMPS.mat;
Q(:,1)=QMPS(:,1); t(1)=cellstr('Straight');
cd (the folder for the model having two branches);
```

```
load TPStoQMPS.mat;
Q(:,2)=QMPS(:,1); t(2)=cellstr(‘Branch2’);
cd (the folder for the model having three branches)
load TPStoQMPS.mat;
Q(:,3)=QMPS(:,1); t(3)=cellstr(‘Branch3’);
cd (the folder for the model having five branches)
load TPStoQMPS.mat;
Q(:,4)=QMPS(:,1); t(4)=cellstr(‘Branch5’);
CADLIVE_DispCumulFreqQMPS(Q, t);
CADLIVE_DispHistFreqQMPS(Q, t, 20, 6, 11);
CADLIVE_DispHistCumulFreqQMPS(Q, t, 20, 6, 11);
```

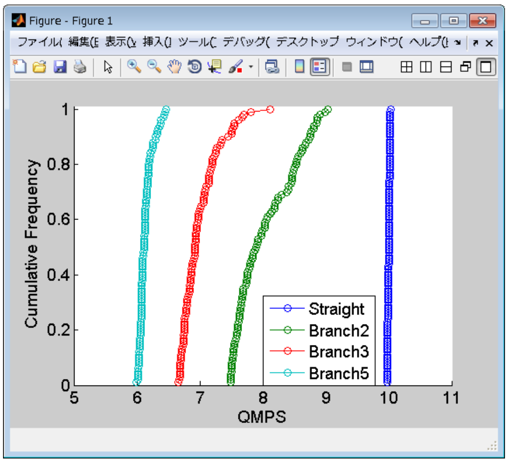



Fig.43 CADLIVE_DispCumulFreqQMPS

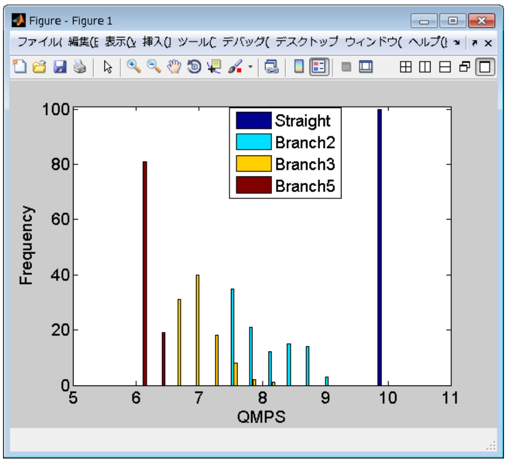



Fig.44 CADLIVE_DispHistFreqQMPS

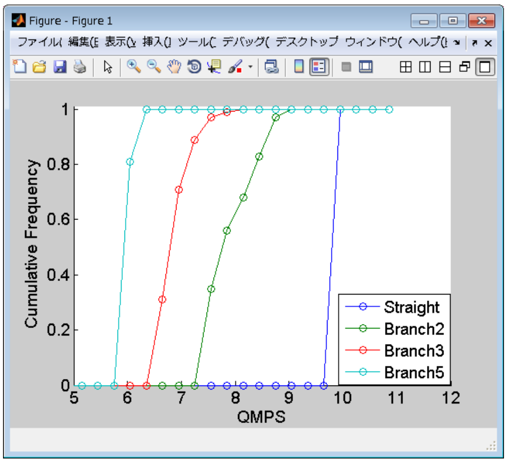



Fig.45 CADLIVE_DispHistCumulFreqQMPS

## 4.6 CADLIVE_optimtool

This is the command for executing GAs by using the optimization toolbox (www.mathworks.co.jp/jp/help/gads/genetic-algorithm.html). To use this command, users buy the optimization toolbox provided by Mathworks Inc. If the search by GAs is successful, OptimizedParameter.mat and CADLIVE_OptParam.m are output.

---

Preparations:

CADLIVE_initial.m, CADLIVE_param.m, CADLIVE_fvec.m, CADLIVE_fjac.m and CADLIVE_getFitness.m in the current folder.

Command:

[SimResult fitness] = CADLIVE_optimtool(options)

Arguments:

| | | |
|---|---|---|
| options | : | Option according to optimization toolbox (http://www.mathworks.co.jp/jp/help/gads/genetic-algorithm-options.html)<br>No argument is set options = gaoptimset(@ga). |

Return:

| | | |
|---|---|---|
| SimResult | : | Simulation result with optimized parameters |
| fitness | : | Best fitness value |

Example:

[SimResult fitness] = CADLIVE_optimtool();

---

# References

[1] Hiroyuki Kurata, Nana Matoba, Natsumi Shimizu, CADLIVE for constructing a large-scale biochemical network based on a simulation-directed notation and its application to yeast cell cycle, *Nucleic Acids Res*.31: 4071-4084, 2003

[2] Hiroyuki Kurata, Kouichi Masaki, Yoshiyuki Sumida, Rei Iwasaki, CADLIVE Dynamic Simulator: Direct Link of Biochemical Networks to Dynamic Models,*Genome Res*., 15: 590-600, 2005

[3] Hiroyuki Kurata, Kentaro Inoue, Kazuhiro Maeda, Koichi Masaki, Yuki Shimokawa, Quanyu Zhao, Extended CADLIVE: a novel graphical notation for designing a biochemical network map that enables computational pathway analysis, *Nucleic Acids Res*. 35(20):e134, 2007

[4] Kentaro Inoue, Sayaka Tomeda, Shinpei Tonami, Yuki Shimokawa, Masayo Ono, Hiroyuki Kurata, CADLIVE Converter for constructing a biochemical network map, *Biochemical Engineering Journal*, 54(3): 200-206, 2011

[5] Kentaro Inoue, Kazuhiro Maeda, Yuki Kato, Shinpei Tonami, Shogo Takagi, Hiroyuki Kurata, CADLIVE Optimizer: Web-based Parameter Estimation for Dynamic Models, *Source Code for Biology and Medicine*, 7:9, 2012

[6] Kazuhiro Maeda, Hiroyuki Kurata, Two-phase search (TPS) method: Nonbiased and high-speed parameter search for dynamic models of biochemical networks, *IPSJ Transaction on Bioinformatics* 2:2-14, 2009

[7] Kazuhiro Maeda, Hiroyuki Kurata, Quasi-multiparameter sensitivity measure for robustness analysis of complex biochemical networks *J Theor Biol* 272:174–186, 2011