Manual: How to use COSMOS

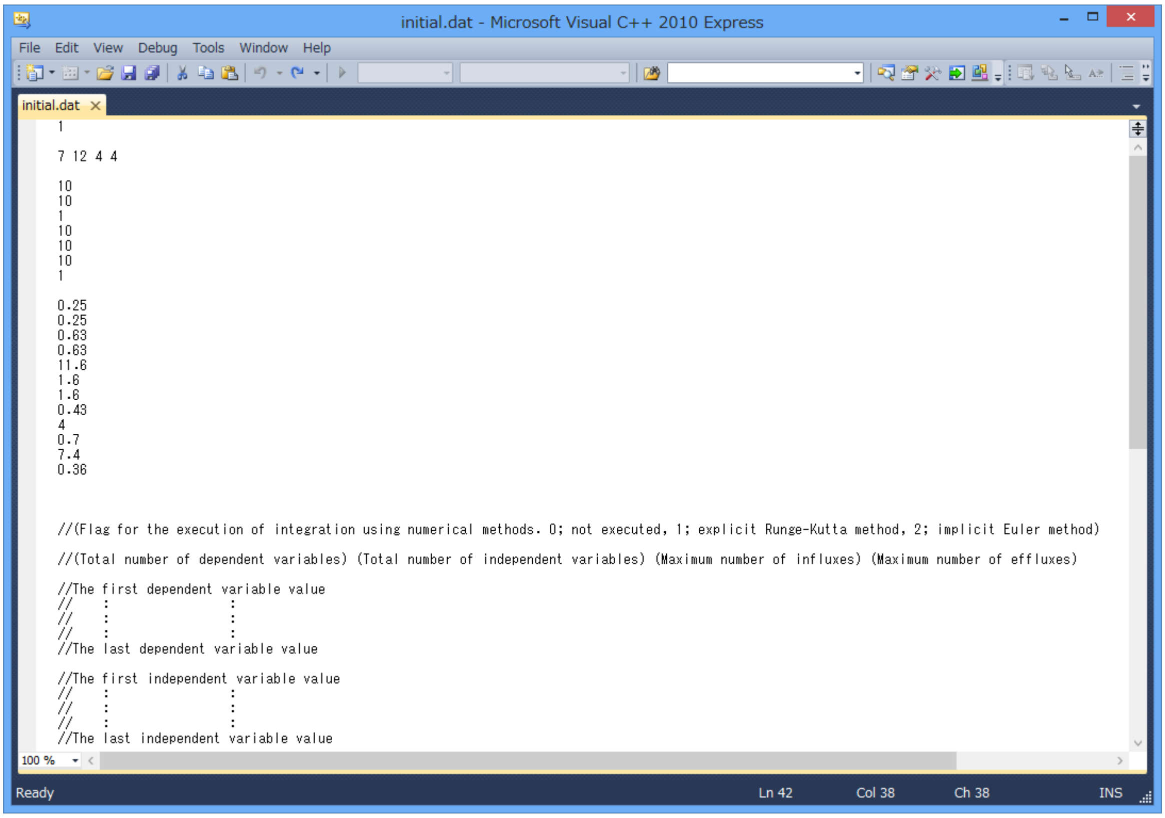
COSMOS was programmed using Visual C++. However, it should work with any other C++ language software. The calculation procedure using COSMOS is as follows.

1) Open the COSMOS program folder. You will find one file (**complex_COSMOS.cpp**) and two folders (**InputFile** and **OutputFile** folders).
2) To set the necessary data for calculation, open the **InputFile** folder. You will find three files (**initial.dat**, **equation.h**, **ingegration.dat**, and **print_format.dat**).

InputFile ………………Folder including the data files necessary to execute the calculation.
|- **initial.dat** ……………….. Data file defining the numbers of dependent and
| independent variables, maximum numbers of
| influxes and effluxes, initial guesses for
| dependent variables, and independent variable
| values.
|- **equation.h** ………………… System description file.
|- **integration.dat** Data file for integration by a numerical method.
|- **print_format.dat** Data file for controlling output of data.

3) Open the **initial.dat** file and input necessary values in the following form. Save the data and close the file. The first value in the **initial.dat** file is set to judge whether one executes the integration of a differential equation model by the Runge-Kutta method in order to estimate better initial guesses for steady-state metabolite concentrations. If this value is set at 0, 1, and 2, the integration is not executed, the integration is executed by the explicit Runge-Kutta method, and the integration is executed by the implicit Euler method, respectively.

Example of **initial.dat** for the model for biosynthesis of aspartate-derived amino acid



```
1

7 12 4 4

10
10
1
10
10
10
1

0.25
0.25
0.63
0.63
11.6
1.6
1.6
0.43
4
0.7
7.4
0.36



//(Flag for the execution of integration using numerical methods. 0; not executed, 1; explicit Runge-Kutta method, 2; implicit Euler method)

//(Total number of dependent variables) (Total number of independent variables) (Maximum number of influxes) (Maximum number of effluxes)

//The first dependent variable value
//    :                :
//    :                :
//    :                :
//The last dependent variable value

//The first independent variable value
//    :                :
//    :                :
//    :                :
//The last independent variable value
```

4) Open the **equation.h** file and describe your system in the lines bound by the symbols “//*************************************”. Other sentences may not be changed.

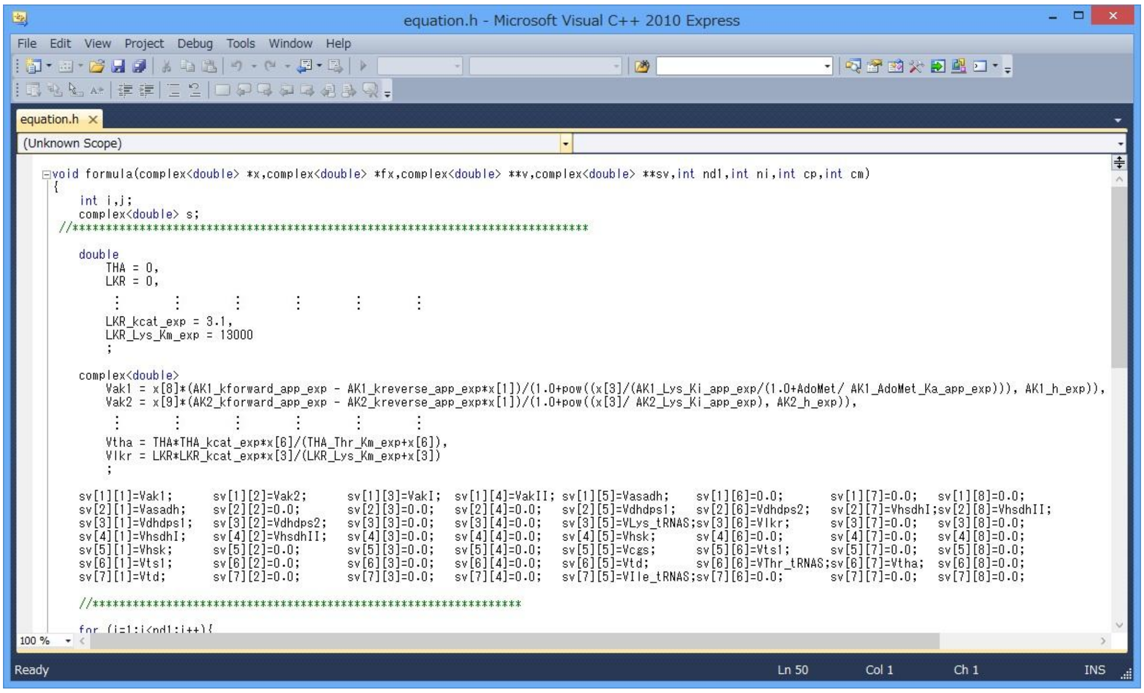
Define the variable names and give their values in **double** type in C++. Any variable name can be used if it is defined here.

Describe the flux formulas that include the dependent variables, X[1] to X[*n*], and the independent variables, X[*n*+1] to X[*n*+*m*], in **complex<double>** type in C++.

Assign each local flux to the array using the specified symbol “sv[][]= …….; “. If the local influx number is less than the maximum number of influxes, set sv[][] at zero. Likewise, if the local efflux number is less than the maximum number of effluxes, set sv[][] at zero.

Save the data and close the file.

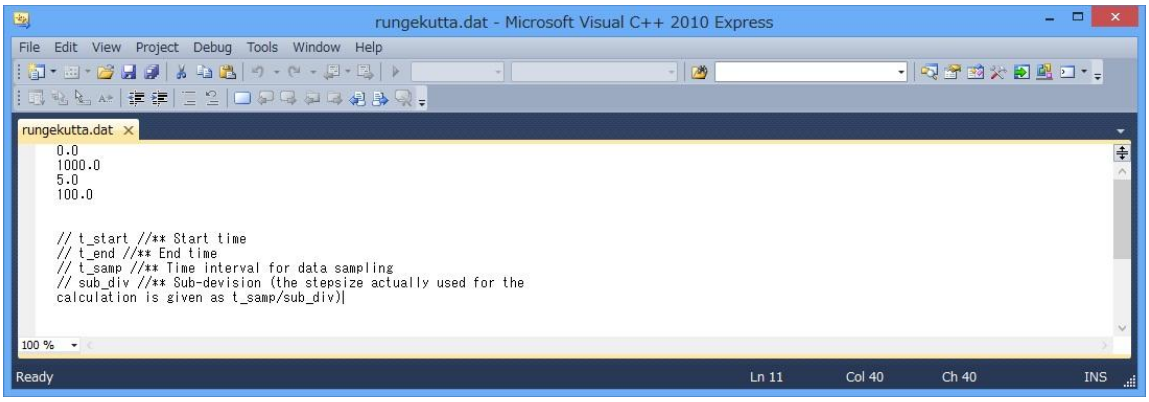
Example of the **equation.h** file for the model for biosynthesis of aspartate-derived amino acid



equation.h - Microsoft Visual C++ 2010 Express
File Edit View Project Debug Tools Window Help
equation.h
(Unknown Scope)
100 %
Ready
Ln 50
Col 1
Ch 1
INS

```
void formula(complex<double> *x,complex<double> *fx,complex<double> **v,complex<double> **sv,int nd1,int ni,int cp,int cm)
{
    int i,j;
    complex<double> s;
//*******************************************************************************

    double
        THA = 0,
        LKR = 0,
         :       :       :       :       :       :
        LKR_kcat_exp = 3.1,
        LKR_Lys_Km_exp = 13000
        ;

    complex<double>
        Vak1 = x[8]*(AK1_kforward_app_exp - AK1_kreverse_app_exp*x[1])/(1.0+pow((x[3]/(AK1_Lys_Ki_app_exp/(1.0+AdoMet/ AK1_AdoMet_Ka_app_exp))), AK1_h_exp)),
        Vak2 = x[9]*(AK2_kforward_app_exp - AK2_kreverse_app_exp*x[1])/(1.0+pow((x[3]/ AK2_Lys_Ki_app_exp), AK2_h_exp)),
         :       :       :       :       :       :
        Vtha = THA*THA_kcat_exp*x[6]/(THA_Thr_Km_exp+x[6]),
        Vlkr = LKR*LKR_kcat_exp*x[3]/(LKR_Lys_Km_exp+x[3])
        ;

    sv[1][1]=Vak1;      sv[1][2]=Vak2;      sv[1][3]=VakI;  sv[1][4]=VakII; sv[1][5]=Vasadh;    sv[1][6]=0.0;       sv[1][7]=0.0;   sv[1][8]=0.0;
    sv[2][1]=Vasadh;    sv[2][2]=0.0;       sv[2][3]=0.0;   sv[2][4]=0.0;   sv[2][5]=Vdhdps1;   sv[2][6]=Vdhdps2;   sv[2][7]=VhsdhI;sv[2][8]=VhsdhII;
    sv[3][1]=Vdhdps1;   sv[3][2]=Vdhdps2;   sv[3][3]=0.0;   sv[3][4]=0.0;   sv[3][5]=VLys_tRNAS;sv[3][6]=Vlkr;      sv[3][7]=0.0;   sv[3][8]=0.0;
    sv[4][1]=VhsdhI;    sv[4][2]=VhsdhII;   sv[4][3]=0.0;   sv[4][4]=0.0;   sv[4][5]=Vhsk;      sv[4][6]=0.0;       sv[4][7]=0.0;   sv[4][8]=0.0;
    sv[5][1]=Vhsk;      sv[5][2]=0.0;       sv[5][3]=0.0;   sv[5][4]=0.0;   sv[5][5]=Vcgs;      sv[5][6]=Vts1;      sv[5][7]=0.0;   sv[5][8]=0.0;
    sv[6][1]=Vts1;      sv[6][2]=0.0;       sv[6][3]=0.0;   sv[6][4]=0.0;   sv[6][5]=Vtd;       sv[6][6]=VThr_tRNAS;sv[6][7]=Vtha;  sv[6][8]=0.0;
    sv[7][1]=Vtd;       sv[7][2]=0.0;       sv[7][3]=0.0;   sv[7][4]=0.0;   sv[7][5]=VIle_tRNAS;sv[7][6]=0.0;       sv[7][7]=0.0;   sv[7][8]=0.0;

    //******************************************************************

    for (i=1;i<nd1;i++){
```

5) If arbitrarily set initial guesses do not lead to convergence of the Newton-Raphson method, one needs to estimate appropriate initial guesses. One way to do this is to integrate a differential equation model until the metabolite concentrations become close to their steady-state values and use the roughly estimated values as initial guesses. The **integration.dat** file is a data file for this integration. Open the **integration.dat** file and set the values (the start time, the end time, the data sampling time, and the value for subdividing the stepsize for data sampling in order to increase the accuracies of calculated values) as follows.



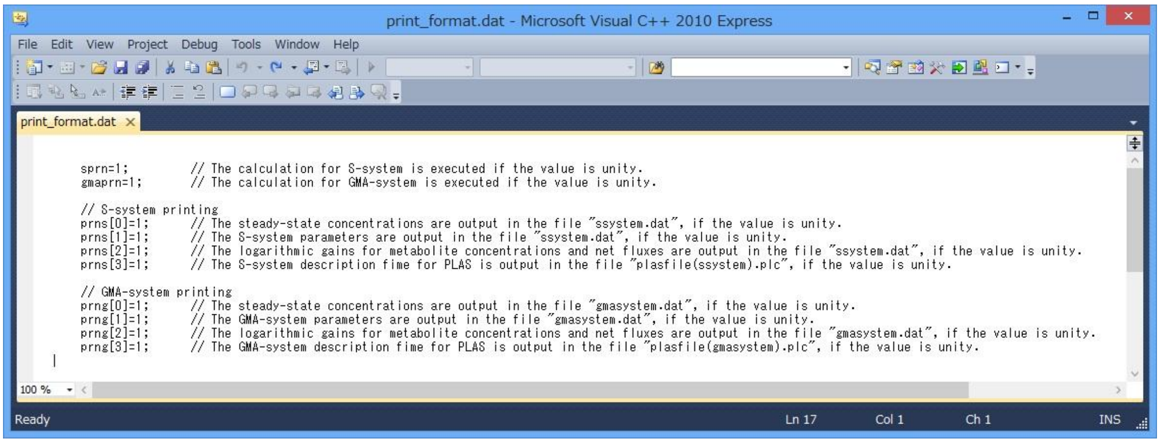
rungekutta.dat - Microsoft Visual C++ 2010 Express
File Edit View Project Debug Tools Window Help
rungekutta.dat
100 %
Ready
Ln 11
Col 40
Ch 40
INS

```
0.0
1000.0
5.0
100.0


// t_start //** Start time
// t_end //** End time
// t_samp //** Time interval for data sampling
// sub_div //** Sub-devision (the stepsize actually used for the
calculation is given as t_samp/sub_div)
```

The time courses of metabolite concentrations by the integration are output into the **timecourse.dat** file in the COSMOS program folder. When the Newton-Raphson method does not converge, check whether the metabolite concentrations roughly approached their steady-state values by opening the file. If not so, change the end time or the value for subdividing the stepsize in order to increase the accuracies of calculated values.

6) If necessary, open the **print_format.dat** file and change the output control data. It is usually unnecessary to change the values. Give the value ‘1’ if you want to output the data file or the value ‘0’ if you do not. Save the data and close the file.

<u>Example of the **print_format.dat** file</u>



print_format.dat - Microsoft Visual C++ 2010 Express
File Edit View Project Debug Tools Window Help
print_format.dat
100 %
Ready
Ln 17
Col 1
Ch 1
INS

```
sprn=1;         // The calculation for S-system is executed if the value is unity.
gmaprn=1;       // The calculation for GMA-system is executed if the value is unity.

// S-system printing
prns[0]=1;      // The steady-state concentrations are output in the file "ssystem.dat", if the value is unity.
prns[1]=1;      // The S-system parameters are output in the file "ssystem.dat", if the value is unity.
prns[2]=1;      // The logarithmic gains for metabolite concentrations and net fluxes are output in the file "ssystem.dat", if the value is unity.
prns[3]=1;      // The S-system description fime for PLAS is output in the file "plasfile(ssystem).plc", if the value is unity.

// GMA-system printing
prng[0]=1;      // The steady-state concentrations are output in the file "gmasystem.dat", if the value is unity.
prng[1]=1;      // The GMA-system parameters are output in the file "gmasystem.dat", if the value is unity.
prng[2]=1;      // The logarithmic gains for metabolite concentrations and net fluxes are output in the file "gmasystem.dat", if the value is unity.
prng[3]=1;      // The GMA-system description fime for PLAS is output in the file "plasfile(gmasystem).plc", if the value is unity.
```

7) Close the **InputFile** folder.

8) Set the file (**complex_COSMOS.cpp**) and two folders (**InputFile** and **OutputFile** folders) in the C++ language software with which you want to run the program.

9) Run the execution file **complex_COSMOS.cpp**. You will get the calculated values quickly if the data are set correctly.

10) Open the **OutputFile** folder to see the calculated results. You will find the following five files. All the calculated values should be given within round-off errors.

OutputFile ……………. Folder including the files generated as a result of the calculation.

|- **ssystem.dat** …………………….. Calculated data file including steady-state metabolite concentrations, rate constants and kinetic orders in an S-system, sensitivities for metabolite concentrations and net fluxes.

|- **gmasystem.dat** ……………… Calculated data file including steady-state metabolite concentrations, rate constants and kinetic orders in a GMA-system, sensitivities for metabolite concentrations and local fluxes.

|- **eigenvalue.dat** ……………… Calculated data file for eigenvalues.

|- **plasfile(ssystem).plc** ……….. S-system description file for PLAS.

|- **plasfile(gmasystem).plc** …. GMA-system description file for PLAS.

11) If you want to extend your calculation further using PLAS, run PLAS and select **plasfile(ssystem).plc** or **plasfile(gmasystem).plc** to open it. You will see the S-system description file or GMA-system description file on the PLAS editor.

11) Operate PLAS according to its function.