Commentary and Instruction for CHECKDAE

Mathematical Equation Conversion System

For User Function Creation

A table of contents

1 Objectives

The variables employed in the checkdae file are named by the index made of the combination of species' name and its localization, as exemplified by y[RNAP.cyt], so that users can understand the meaning of the variables. However, in order for computers to simulate mathematical equations (TT, CMA, GMA, MM, DAEs, S-system), it is required to replace the indexes of the variable name by sequential integers, to make the user functions for mathematical equations, and to produce their Jacobian function that is the partial derivatives of the user functions with respect to each variable in symbolic form. The checkdae module generates computer-readable parameters, mathematical functions, and their Jacobian. These functions and parameter files are described in the C programming language.

The checkdae module examines if a checkdae file (mathematical equations), which the parsedae module has generated or users have edited manually, is acceptable for the rules for converting it into simulation programs (user functions and parameters). The checkdae module generates not only the user functions for mathematical equations and their associated Jacobian, but also temporary mathematical expressions such as fluxes. In addition, the checkdae module is able to convert ordinary differential equations (TT, CMA, GMA, MM) into S-system at the steady state, generating the user functions for S-system with its associated coefficient functions and their parameters. Simultaneously, it creates the parameter files for setting initial values and kinetic parameters.

In summary, from a checkdae file, the checkdae module creates the five files: a parameter file, a user function file, a temporary mathematical expression file, an S-system coefficient file, and an S-system user function file.

2. Function

Finishing checking the grammar of a mathematical model (checkdae file), the checkdae module divides it into the nodes consisting of one operator in order to expand them on a binary tree structure. The checkdae first examines if the employed parameters and variables are defined in a checkdae file. Second, it converts the index of species' names to the numerical one, and solves the partial derivatives of the mathematical equations to obtain their Jacobian in symbolic form, resulting in a user function file with a parameter file that contains initial values and kinetic constants. The use of GMA or MM conversion creates another file that contains the temporary mathematical expressions for its fluxes. When S-system is employed, it produces an S-system coefficient file and an S-system user function file.

2.1 Grammar check for mathematical models

The checkdae examines the correctness of the checkdae file (mathematical equations) and temporary mathematical expressions from the standpoint of grammar, which are restricted to five operators consisting of addition (+), subtraction (-), multiplication (*), division (/), and power (^). The symbols except numbers are all regarded as variables. The detailed grammar is explained in the parsedae instruction.

2.2 Expansion to mathematical equation trees

Mathematical equations are expanded on binary trees whose nodes consist of an operator. The subsequent operations are carried out based on the binary trees.

2.3 Function for checking parameters and variables

This function checks if the variables, parameters, and temporary mathematical expressions employed in mathematical equations have been defined in the checkdae file.

2.4 Function for expanding temporary expressions

The temporary mathematical equations are expanded to generate a user function, and then the temporary equations are removed from the user function.

2.5 Function for indexing variables and parameters

The sequential integers are provided for all the variables and parameters that users have defined.

2.6 Function for replacing the indexes

The numerical indexes are substituted for the variables named with the combination of species' name and its compartment.

2.7 Symbolic partial differentiation

The partial derivatives are solved with respect to the variables of interest in symbolic form.

2.8 Function for S-system conversion

Ordinary differential equations are converted into S-system. The detailed explanation is described elsewhere.

2.9 Parameter survey

Users are able to explore the parameter spaces by using the parameter file, where the values of parameters are varied in the arithmetic or geometric series. Details are described in the parameter file (3.2.2).

3 Specification of input and output

3.1 Specification of input

The checkdae module reads the checkdae file (mathematical equations) to make the user functions and their associated parameter files necessary for simulation. The details of the checkdae file are explained in the instruction of the parsedae module.

3.2 Specification of output

3.2.1 User function file

The user function consists of mathematical equations and their Jacobian. The Jacobian is obtained in symbolic form by solving the partial derivatives of mathematical equations with respect to each variable. The user function contains two functions: usr_fvec and usr_fjac, as follows.

```
/*
 * TITLE:test
```

```
 * INFO:test
 * CONVERSION TYPE
 *        GENE-PROTEIN:TPP_RAPID
 *        METABOLIC:NONE
 */

#include "life.h"

void usr_fvec(double fvec[], double y[], double Gene, Parameter *p)
{
        double *constantPlayer= p->val[1];
        double *Kb= p->val[2];
        double *kp = p->val[3];
        double *kpd             = p->val[4];
        double *km              = p->val[5];
        double *kmd             = p->val[6];

        fvec[    1] = y[12] - (y[1] + y[6] + y[7] + pow(y[8], 1.5));
        fvec[    2] = constantPlayer[2] - (y[2] + pow(y[6], 2) + pow(y[7], 2) + pow(y[8], 3));
        fvec[    3] = constantPlayer[3] - (y[3] + pow(y[7], 1.5) + y[10]);
                --- omission ---
        fvec[ 12] = -kpd[1]*y[1] - kpd[2]*y[6] - kpd[3]*y[7] - 1.5*kpd[4]*y[8];
        fvec[ 13] = kp[1]*y[11] - kpd[5]*y[5] - kpd[6]*y[9] - kpd[7]*y[10];
}

void usr_fjac(double fjac[][MAX_VAR_NUM], double y[], double Gene, Parameter *p)
{
        double *constantPlayer= p->val[1];
        double *Kb= p->val[2];
        double *kp = p->val[3];
        double *kpd             = p->val[4];
        double *km              = p->val[5];
        double *kmd             = p->val[6];

        fjac[    1][    1] = -1;
        fjac[    1][    6] = -1;
                --- omission ---
        fjac[ 13][    9] = -kpd[6];
        fjac[ 13][ 10] = -kpd[7];
        fjac[ 13][ 11] = kp[1];
}
```

Equations are indexed in the sequential order of algebraic and differential equations. Variables are marked by y. The fjac is described only when its value is nonzero.

3.2.2 Parameter file

The checkdae module outputs the parameter file for setting initial values and kinetic parameters. The line up of PARAM_NAME corresponds to the numerical index order of the parameters in the user function. When users gave values to parameters and initial concentrations in the chekdae file, the values appeared at the corresponding order of PARAM_NAME.

```
##### Model ################
# TITLE:test
# INFO:test
# CONVERSION TYPE
#        GENE-PROTEIN:TPP_RAPID
#        METABOLIC:NONE
N_VAR     ;    13; # num of variables(all)
```

```
N_ALGEBR;    10; # num of variables(Algebraic Eq)

#Y_START ; index; initial_value; tag #comment
Y_START ;      1;    1.2000e+00;    A.cyt            #
Y_START ;      2;    2.1000e+00;    B.cyt            #
Y_START ;      3;    3.1000e+00;    C(pro).cyt       #
                --- omission ---
Y_START ;    11;    1.0000e-01;    mRNA(test).cyt              #
Y_START ;    12;    1.7650e+00;    TA.cyt           #
Y_START ;    13;    6.0000e-02;    Ttest.cyt        #

#PARAM;name;index;val;val_start;num_survey;D/R/S;change-val;GA_start;GA_end;tag #comment
PARAM      ;constantPlayer;      1;    1.2000e+00;;0;D;;;; G(test).cyt        #
PARAM      ;constantPlayer;      2;    3.2300e+00;;0;D;;;; TB.cyt             #
                --- omission ---
PARAM      ;kmd          ;      1;      ;;0;D;;;; decomposition_rate_constant_mRNA(test).cyt          #

#T_EVENT ; name; index; time; value; #comment


######## Don't edit the following lines. ######
PARAM_NAME ;constantPlayer    #
PARAM_NAME ;Kb                    #
PARAM_NAME ;kp                   #
PARAM_NAME ;kpd                  #
PARAM_NAME ;km                    #
PARAM_NAME ;kmd                   #
```

Details of the #PARAM line are explained as follows,

name: the name of the parameter

index: the index of the parameter

val: the value of the parameter

val_start: the starting value of the parameters, which must be set as the same value as "val" when the parameter survey is carried out.

num_survey: the number of surveyed parameters. When the parameter survey is not performed, set one (1).

D/R/S: The option for the parameter survey. "D" indicates arithmetic series, "R" indicates geometric series, and "S "means the parameter search by GAs or RS (Details are written elsewhere).

change_val: changing value.

GA_start; The starting value of the explored variable region, which is employed for search for GAs or RS.

GA_end; The ending value of the explored variable region, which is employed for search for GAs or RS.

For example, when users change the parameter kx[3] as 1, 2, 3, and 4 in the arithmetic series, the line of #PARAM is provided by:

```
PARAM      ;kx;      3;    1;    1 ;    4;    D;    1;    ;    ;    tag  #
```

When users change the parameter kx[3] as 1, 2, 4, and 8 in the geometric series, the line of #PARAM is provided

by:

```
PARAM    ;kx;    3;   1;   1 ;   4;   R;   2;   ;   ;   tag  #
```

3.2.3 Temporary mathematical expression file

When temporary mathematical equations such as a flux and total enzyme are defined in the checkdae file, the corresponding user function is generated automatically. The temporary expressions appear in the checkdae file, when GMA or MM conversion is selected. Mathematical equations for fluxes is shown as follows.

```
/*
 * TITLE:test
 * INFO:test
 * CONVERSION TYPE
 *        GENE-PROTEIN:TPP_RAPID
 *        METABOLIC:NONE
 */

#include "life.h"

void usr_flux(double total[], double flux[], double y[], double Gene, Parameter *p)
{
    double *constantPlayer= p->val[1];
    double *Kb= p->val[2];
    double *kp = p->val[3];
    double *kpd              = p->val[4];
    double *km               = p->val[5];
    double *kmd              = p->val[6];
    double *Kg= p->val[7];
    double *f   = p->val[8];

    flux[   1] = Kg[1]*pow(y[1], f[1]); /* flux_A.cyt_to_B.cyt */

    /* No Total expression. */
}
```

3.2.4 S-system coefficient file

The S-system coefficient file is used for calculating the coefficients of S-system at the steady state, which are sent to the program for the sensitivity and stability analysis of S-system.

```
/*
 * TITLE:Glycolysis.plc
 * INFO:Glycolytic-Glycogenolytic Pathway Model in Chapter 11 PLUS
 */

#include "life.h"

void get_GHdata(double **gd, double **gi, double **hd, double **hi,
    double *v_plus, double *v_minus, double *y, Parameter *p)
{
    double *constantPlayer= p->val[1];

    v_plus [    1] = 0.0778843*pow(constantPlayer[1], 0.66)*constantPlayer[3];
```

```
    gi[    1][    1] = 0.66*pow(constantPlayer[1], -0.34)*0.0778843*constantPlayer[3];
    gi[    1][    3] = 0.0778843*pow(constantPlayer[1], 0.66);
    v_minus[    1] = 1.06271*pow(y[1], 1.53)*pow(y[2], (-0.59))*constantPlayer[4];
    hd[    1][    1] = 1.53*pow(y[1], 0.53)*1.06271*pow(y[2], (-0.59))*constantPlayer[4];
    hd[    1][    2] = (-0.59)*pow(y[2], (-0.59) - 1)*1.06271*pow(y[1], 1.53)*constantPlayer[4];
    hi[    1][    4] = 1.06271*pow(y[1], 1.53)*pow(y[2], (-0.59));
                --- omission ---
    gi[    3][    5] = 0.000793456*pow(y[2], 3.97)*pow(y[3], (-3.06));
    v_minus[    3] = 1.05881*pow(y[3], 0.3)*constantPlayer[6];
    hd[    3][    3] = 0.3*pow(y[3], -0.7)*1.05881*constantPlayer[6];
    hi[    3][    6] = 1.05881*pow(y[3], 0.3);
}


void get_GHnum(int *n_var, int *n_const)
{
    *n_var = 3;
    *n_const = 7;
}
```

3.2.5 S-system user function file

The checkdae module converts ordinary differential equations into S-system, generating the user function of S-system, as follows.

```
/*
 * TITLE:Glycolysis.plc
 * INFO:Glycolytic-Glycogenolytic Pathway Model in Chapter 11 PLUS
 */

/*

#include "life.h"

void usr_fvec(double fvec[], double y[], double Gene, Parameter *p)
{
    double *constantPlayer= p->val[1];
    double *kplus              = p->val[2];
    double *kminus           = p->val[3];
    double *g   = p->val[4];
    double *h   = p->val[5];

    fvec[    1] = kplus[1]*pow(constantPlayer[1],g[1])*pow(constantPlayer[3],g[2])
                    - kminus[1]*pow(y[1],h[1])*pow(y[2],h[2])*pow(constantPlayer[4],h[3]);
                --- omission ---
    fvec[    3] = kplus[3]*pow(y[2],g[8])*pow(y[3],g[9])*pow(constantPlayer[5],g[10])
                - kminus[3]*pow(y[3],h[7])*pow(constantPlayer[6],h[8]);
}

void usr_fjac(double fjac[][MAX_VAR_NUM], double y[], double Gene, Parameter *p)
{
    double *constantPlayer= p->val[1];
    double *kplus              = p->val[2];
    double *kminus           = p->val[3];
    double *g   = p->val[4];
    double *h   = p->val[5];
```

```
    fjac[    1][    1] =    - kminus[1]*h[1]*pow(y[1],h[1]-1)*pow(y[2],h[2])*pow(constantPlayer[4],h[3]);
    fjac[    1][    2] =    - kminus[1]*pow(y[1],h[1])*h[2]*pow(y[2],h[2]-1)*pow(constantPlayer[4],h[3]);
                --- omission ---
    fjac[    2][    3] =    - kminus[2]*pow(y[2],h[4])*h[5]*pow(y[3],h[5]-1)*pow(constantPlayer[5],h[6]);
    fjac[    3][    2] = kplus[3]*g[8]*pow(y[2],g[8]-1)*pow(y[3],g[9])*pow(constantPlayer[5],g[10]);
    fjac[    3][    3] = kplus[3]*pow(y[2],g[8])*g[9]*pow(y[3],g[9]-1)*pow(constantPlayer[5],g[10])
                    - kminus[3]*h[7]*pow(y[3],h[7]-1)*pow(constantPlayer[6],h[8]);
}
```