Maple Kinetics Package

Mark Holmes
Matthew Pelliccione

May 1, 2007

Contents

1 Introduction

The *Maple* kinetics package was designed to be used as a tool in analyzing systems of chemical reactions that takes advantage of the symbolic computing used in *Maple*. However, many other systems that can be modeled by simple "reactions" can also be analyzed, including predator-prey and epidemic models.

2 Installation

Two files are included with this software package; `kinetics.mpl`, which contains the source code and program data, and `kineticshelp.mdb`, which contains the help files associated with the functions in the program. The program has been designed to run in *Maple* versions 9.0 and later. To install the help files, copy the `kineticshelp.mdb` file into the `lib` directory in the *Maple* root folder. The next time *Maple* is run, it will recognize the new help files and add them to the help database. Then, in any *Maple* worksheet, is using Windows OS run the command

```
read "c:/kinetics.mpl":
```

or if using Mac OS run the command

```
read "/Users/holmes/Desktop/kinetics.mpl":
```

with the specific directory varying according to the folder `kinetics.mpl` is stored in. This loads all the kinetics functions into the *Maple* environment. The worksheet is now ready for input of the chemical reactions.

3 Data Input

Three different sets of input data are required for most of the functions implemented in the kinetics package. These data sets are the reactions, rate constants, and initial conditions.

3.1 Reactions

The reactions to be analyzed are entered into *Maple* using a list named `R`. All functions implemented in the kinetics package require the definition of `R`. The reactants and products of a reaction are separated by the greater than sign (`>`), with the reactants on the left of `>` and the products on the right of `>`. For example, Michaelis-Menten kinetics can be summarized with the reactions

$$S + E \rightleftharpoons C \rightarrow P + E$$

To enter these reactions, `R` should be defined as

```
R := [S + E > C, C > S + E, C > P + E]:
```

Also, note that the first reaction above can be written as `C < S + E`, which is equivalent to `S + E > C` as read by the kinetics program.

3.1.1 Restricted Variable Names

In general, it is best to enter the names of the species using capital letters, since many lower case letters such as `i`,`j`,`k`,`c`,`p` etc. are reserved by the program. However, the specific names `O` and `I` are reserved by *Maple* and the letter `R` is reserved by the program as the list of reactions.

3.2 Rate Constants

For each reaction, a corresponding rate constant can be defined that describes the speed at which the reaction occurs. The rate constants are entered in a list named `rate_constants`, similar to the definition of the reactions. If the rate constants are entered, each reaction must have a rate constant, and the order of the list of rate constants must correspond to the list of reactions. Therefore, the rate constant of the first reaction in `R` is the first entry in the `rate_constants` list, and so forth. No variables can be elements of the `rate_constants` list, it must only contain numbers. Using the Michaelis-Menten reactions as an example, if the system is described by the equations

$$\begin{aligned} S + E &\rightarrow C \qquad k_1 = 1 \text{ L} \cdot \text{ mol}^{-1} \cdot \text{s}^{-1} \\ C &\rightarrow S + C \qquad k_2 = 0.1 \text{ s}^{-1} \\ C &\rightarrow P + E \qquad k_3 = 10 \text{ s}^{-1} \end{aligned}$$

the reactions and rate constants would be entered

```
R := [S + E > C, C > S + E, C > P + E]: rate_constants := [1,0.1,10]:
```

Units are not included in the `rate_constants` list, and the kinetics program does not convert between different units. It is left up to the user to convert all the rate constants to comparable units. Also, the program does not accommodate for reactions whose rate constants may change with time or reaction conditions, all rate constants are assumed to be constant.

3.3 Initial Conditions

For the functions that involve solving the system of equations generated by the reactions, the initial conditions must be specified to obtain numerical solutions. The initial conditions are entered in a list named `initial`. The order in which the species are entered in the list does not matter. Thus, if the initial concentrations of the species in the Michaelis-Menten reactions are $S_0 = 10$ mol, $E_0 = 0.1$ mol, $P_0 = 0$ mol, and $C_0 = 0$ mol, the list `initial` would be defined as

```
initial := [S(0) = 10, E(0) = 0.1, P(0) = 0, C(0) = 0]:
```

As with the rate constants, units are not included when defining initial conditions. However, the implicit units used with the rate constants and initial conditions should match to obtain the correct result.

4 Kinetics Functions

4.1 odes

Description

Finds the complete set of first order ordinary differential equations of the reaction system using the law of mass action.

Calling Sequence

`odes()`
Only the variable `R`, the reaction list, must be defined to run the function.

Execution

Running the `odes()` procedure using the Michaelis-Menten reactions previously defined returns the following to the *Maple* window:

```
The differential equations are:

        d
        -- E(t) = -k1 E(t) S(t) + k2 C(t) + k3 C(t)
        dt

        d
        -- S(t) = -k1 E(t) S(t) + k2 C(t)
        dt

        d
        -- P(t) = k3 C(t)
        dt

        d
        -- C(t) = k1 E(t) S(t) - k2 C(t) - k3 C(t)
        dt
```

Implementation

To determine the forms of the differential equations, the rate associated with each reaction is calculated according to the law of mass action. These reaction rates are then added together appropriately to determine the rate at which each species changes, which is essentially the differential equation returned by `odes()`. First, matrices representing the reactants ($\mathbf{M}_R$) and products ($\mathbf{M}_P$) are created from the list of input reactions `R`. The entries of these matrices are the stoichiometric coefficients of each species in each reaction, with each row of the matrices representing a different reaction. Thus, for the Michaelis-Menten reactions, these matrices are found to be

$$\mathbf{M}_R = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

and

$$\mathbf{M}_P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

with the columns of these matrices representing the species S, E, C and P respectively.

Then, using the reactants matrix $\mathbf{M}_R$, the rate associated with each reaction can be found. Let there be n species S_n involved in the reaction scheme. According to the law of mass action, the rate associated the i^{th} reaction is

$$r_i = k_i \prod_{j=1}^{n} S_j(t)^{\mathbf{M}_{R(i,j)}}.$$

Using these rates, the rates of the individual species can be deduced. Let there be l reactions in the reaction scheme. The rate of the j^{th} species S_j is

$$\frac{dS_j(t)}{dt} = \sum_{i=1}^{l} (\mathbf{M}_P(i,j) - \mathbf{M}_R(i,j)) r_i.$$

The result of this final summation is the output of the `odes()` function.

4.2 laws

Description

Finds all linearly independent conservation laws of the reaction system.

Calling Sequence

`laws()`
Only the variable `R`, the reaction list, must be defined to run the function.

Execution

Running the `laws()` procedure using the Michaelis-Menten reactions previously defined returns the following to the *Maple* window:

```
There are 2 conservation laws.
The conservation laws are:

                    -E(t) + S(t) + P(t) = C1
                         E(t) + C(t) = C2
```

Implementation

Let $\mathbf{S}$ be a vector of the species involved in a reaction scheme, and $\mathbf{c}$ be a vector of constants with the same dimension as $\mathbf{S}$. A vector $\mathbf{c}$ yields a conservation law iff

$$\mathbf{S} \cdot \mathbf{c} = \text{constant}$$

for all t. Thus, the function `laws()` must find all linearly independent vectors $\mathbf{c}$ that satisfy this equation. To do this, note that if the time derivative of the above equation is taken,

$$\begin{aligned} \frac{d}{dt}(\mathbf{S} \cdot \mathbf{c}) &= 0 \\ \frac{d\mathbf{S}}{dt} \cdot \mathbf{c} + \mathbf{S} \cdot \frac{d\mathbf{c}}{dt} &= 0 \\ \frac{d\mathbf{S}}{dt} \cdot \mathbf{c} &= 0 \end{aligned}$$

since $\frac{d\mathbf{c}}{dt} = \mathbf{0}$ because $\mathbf{c}$ is a constant vector. However, the vector $\frac{d\mathbf{S}}{dt}$ is known from the result of the `odes()` procedure. From the derivation of $\frac{d\mathbf{S}}{dt}$, it can be written as the matrix product

$$\frac{d\mathbf{S}}{dt} = (\mathbf{M}_P - \mathbf{M}_R)^T \mathbf{r},$$

where $\mathbf{r}$ is a vector of reaction rates. Thus, finding the linearly independent vectors $\mathbf{c}$ is equivalent to finding the kernel of the matrix $(\mathbf{M}_P - \mathbf{M}_R)^T \mathbf{r}$, which can be easily done using the `linalg[kernel]` command in *Maple*.

4.3 steady

Description

Finds some of the species that will reach a zero steady state.

Calling Sequence

`steady()`
Only the variable `R`, the reaction list, must be defined to run the function.

Execution

Running the `steady()` procedure using the Michaelis-Menten reactions previously defined returns the following to the *Maple* window:

```
The following species were found to have a zero steady state:

                                C(t)
```

Implementation

The function `steady()` takes the differential equation set created by `odes()` and finds all equations of the form

$$\frac{dS_j(t)}{dt} = kS_l(t)$$

where j need not equal l. The steady state condition for the system is $\frac{dS_j(t)}{dt} = 0$ for all j, and applying this condition to equations of the above form yields the condition $S_l(t) = 0$ when a steady state is achieved. Thus, $S_l(t)$ will always reach zero at a steady state, regardless of the initial conditions.

To find these equations, the stoichiometric matrix $\mathbf{M}_P - \mathbf{M}_R$ is computed, and the columns that contain exactly one nonzero entry are examined. From the derivation of $\frac{dS_j(t)}{dt}$, one nonzero entry in the j^{th} column of $\mathbf{M}_P - \mathbf{M}_R$ means $\frac{dS_j(t)}{dt}$ depends only on the rate of one reaction. Specifically, if the lone nonzero entry in the j^{th} column occurs in the i^{th} row, the rate for S_j depends only on the i^{th} reaction rate r_i. Once a column meeting this criteria has been found, the reaction rate r_i is examined to see if it depends on only one species. If r_i only depends on one species, then the rate equation for S_j fits the criteria for a zero steady state. The i^{th} row of $\mathbf{M}_R$ is searched to see if it contains only one nonzero entry. If this is true, and the nonzero entry occurs in the l^{th} column, the species $S_l(t)$ will reach zero when a steady state is achieved.

4.4 plot_ode

Description

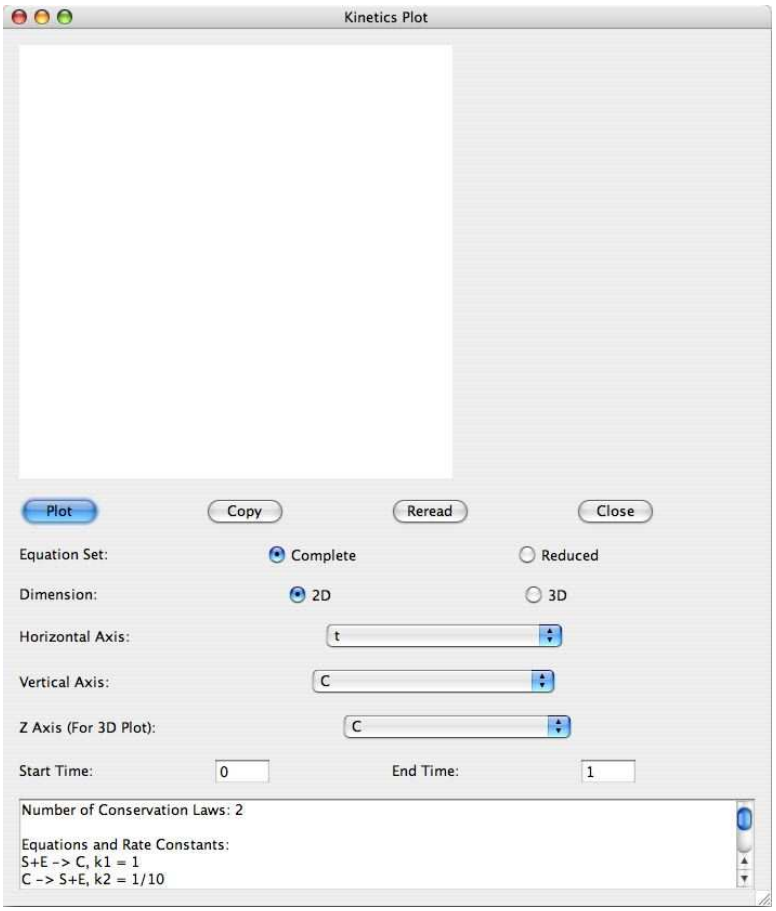
Plots the species solutions of the reaction scheme subject to user defined initial conditions.

Calling Sequence

`plot_ode()`
All three variables, `R`, `rate_constants`, and `initial`, must be defined to run the function.

Execution

Running the `plot_ode()` procedure using the Michaelis-Menten reactions previously defined opens the following window:



Plot: Clicking the `Plot` button will display a plot with the options selected in the window. Changing the dimension from `2D` to `3D` will create a three dimensional plot with the species represented on the third axis selected in the last drop down box labeled `Z Axis (For 3D Plot)`.

Copy: The `Copy` button performs the exact same action as the `Plot` button, but returns the plot to the *Maple* worksheet from which the `plot_ode()` function was executed instead of displaying the plot in the window. This allows the user to copy the picture of the graph to the clipboard or save it as a picture file for later use. ***Note:*** The `Copy` button creates a plot based on the settings selected in the window when the `Copy` button is pressed. If a different plot is pictured in the window that was created using different settings, it will not be copied to the *Maple* worksheet.

Reread: The `Reread` button allows the user to change the reaction scheme used by the procedure without closing the window. This can be useful if the user would like to see the effect of changing the reaction rates or the initial conditions without exiting the plot window. Upon clicking `Reread`, a file dialog box comes up where the user can select an external file where the reaction data is stored. The easiest way to use this command is to store the values for `R`, `rate_constants`, and `initial` in a separate text file that can be edited with a text editor. The program then reads in this data and performs all the necessary calculations to plot the new reaction scheme. This allows for real time modifications of the data, letting the user see the results of a change in the input data without exiting the plot procedure. If an invalid file is used, the procedure exits without changing the current data.

Equation Set: This option lets the user decide which set of equations the plot procedure solves when plotting the solutions of the reaction scheme. In general, reaction schemes with many conservation laws can be solved faster using the reduced equation set, otherwise the complete equation set is faster. To help the user choose, the number of conservation laws is shown in the text box at the bottom of the window. However, both equation sets result in the same solution for all species, only speed is affected by the choice of the complete or the reduced equation set.

Equation Set: Complete/Reduced

The complete equation set is exactly the output of the `odes()` procedure coupled with the reaction rates and initial conditions specified in `rate_constants` and `initial`. The reduced equation set is the complete equation set with all the conservation laws given by `laws()` substituted into the differential equations to eliminate some of the species from the ODEs. Each conservation law allows for one species to be eliminated from the set of differential equations, with the eliminated species related to the other species through the conservation law.

For a speed comparison between the complete and reduced equation sets, two different reaction sets are taken, one being the Michaelis-Menten reactions previously cited, and the Oregonator reactions defined with the following parameters:

```
R := [A+Y>X+P,X+Y>2*P,A+X>2*X+2*Z,2*X>A+P,Z>1/2*Y]:
rate_constants := [1,1e6,2,2e3,10]:
initial := [X(0)=3,Y(0)=1,Z(0)=1,A(0)=100,P(0)=0]:
```

The Oregonator system has one conservation law. The graphs of the species $C(t)$ in the Michaelis-Menten scheme and $Z(t)$ in the Oregonator scheme are used in the following analysis. When a short time interval is used, from t=0 to t=1, the complete and reduced sets are almost identical in their computation time. However, a distinct difference is seen when a large time interval is used, from t=0 to 10 for the Oregonator and from t=0 to 1000 for the Michaelis-Menten reactions. For the Oregonator, with one conservation law, when $Z(t)$ is graphed with the complete set, it takes about 50 seconds to graph, whereas it takes 500 seconds to graph $Z(t)$ with the reduced equation

set. The opposite result is obtained when the Michaelis-Menten reaction scheme is graphed. Using the complete set, the graph of $C(t)$ takes about 23 seconds compute, whereas the reduced set takes about 1 second. The reason for the large time difference is the number of conservation laws for each reaction scheme. ***In general, reaction sets with more conservation laws will benefit from the use of the reduced equation set.*** Since the Oregonator has only one conservation law, the extra computation involved with using the reduced equation set does not outweigh the computation saved by solving the reduced system.

Implementation

The *Maple* procedure `dsolve()` is used by the `plot_ode()` procedure to obtain the plotted solutions. The *Maplet* library included with *Maple* is used to create the window.

4.5 plot_multiple_ode

Description

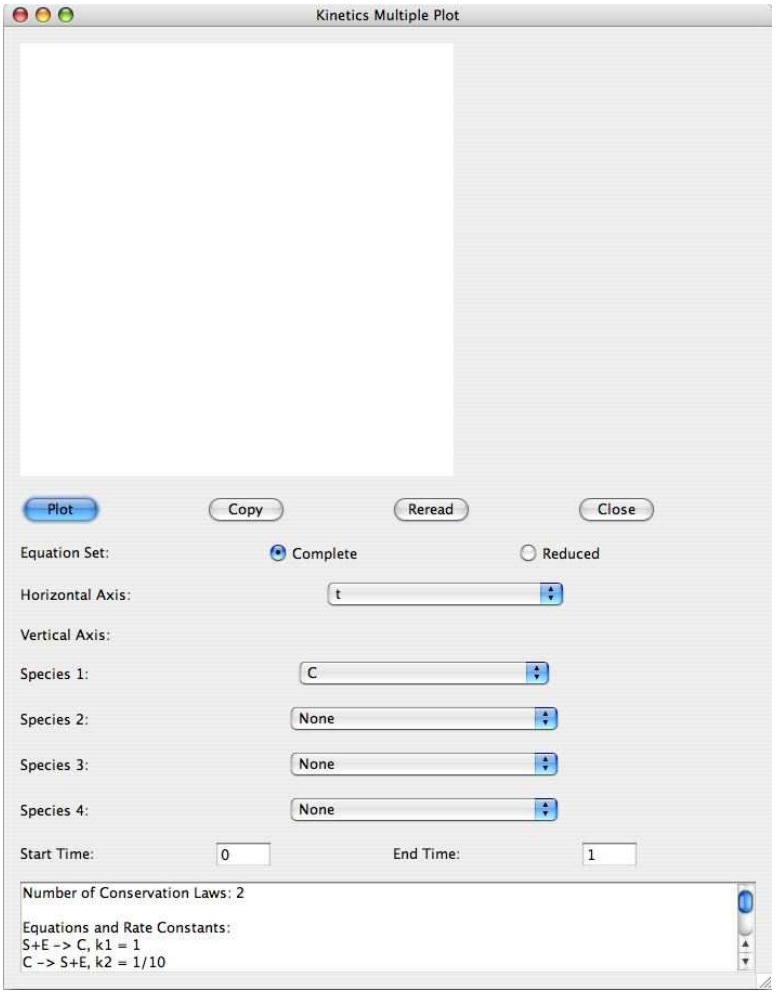
Plots the species solutions of the reaction scheme, allowing multiple species to be plot on the same axes.

Calling Sequence

`plot_multiple_ode()`
All three variables, `R`, `rate_constants`, and `initial`, must be defined to run the function.

Execution

Running the `plot_multiple_ode()` procedure using the Michaelis-Menten reactions previously defined opens the window pictured below. The available options are explained in the **Execution** section of the `plot_ode()` description.



Implementation

The *Maple* procedure `dsolve()` is used by the `plot_multiple_ode()` procedure to obtain the plotted solutions. The *Maplet* library included with *Maple* is used to create the window.

4.6 ode_data

Description

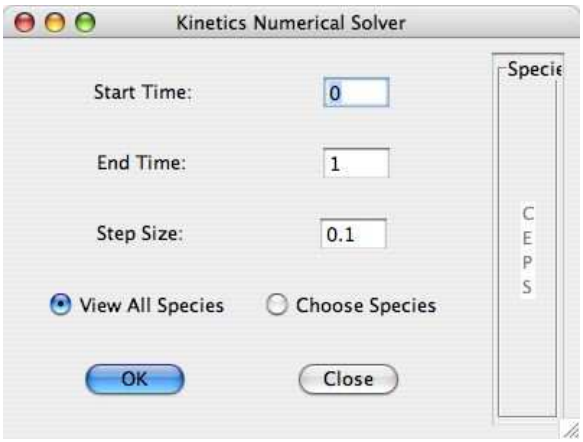
Returns the numerical solution of the reaction scheme subject to user defined initial conditions.

Calling Sequence

`ode_data()`
All three variables, `R`, `rate_constants`, and `initial`, must be defined to run the function.

Execution

Running the `ode_data()` procedure using the Michaelis-Menten reactions previously defined opens the following window:



The solution of every species in the specified time interval can be viewed by selecting the `View All Species` option. If only the solution for some of the species is needed, the `Choose Species` option can be selected, which activates the species list on the right hand side of the window. Any combination of species can be selected in the list, and the resultant table will only include solutions of the selected species.

Implementation

The *Maple* procedure `dsolve()` is used by the `ode_data()` procedure to obtain the numerical solutions. The *Maplet* library included with *Maple* is used to create the window.

4.7 eqpts

Description

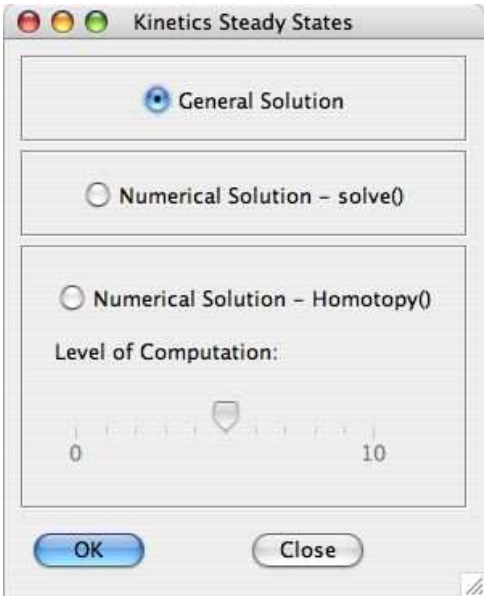
Procedure for finding the steady states of a reaction scheme.

Calling Sequence

`eqpts()`
All three variables, `R`, `rate_constants`, and `initial`, must be defined to run the function.

Execution

Running the `eqpts()` procedure opens a *Maplet* window seen below:



The execution of each of the options that appears in the window is described in an appropriate section that follows.

4.7.1 eqpts - General Solution

Description

Finds general steady state solutions to the reaction scheme using the `solve()` procedure.

Execution

Choosing this option in the `eqpts()` procedure using the Michaelis-Menten reactions previously defined returns the following to the *Maple* window:

```
{E(t) = E(t), S(t) = 0, C(t) = 0, P(t) = P(t)},

{E(t) = 0, C(t) = 0, P(t) = P(t), S(t) = S(t)}
```

The above output represents two conditions for a steady state in this reaction scheme. The first condition states that a steady state is reached if `S(t)` and `C(t)` are both zero, with `P(t)` and `E(t)` unrestricted. The second condition states that a steady state is reached if `E(t)` and `C(t)` are both zero, with `P(t)` and `S(t)` unrestricted. To determine the conditions that should be placed on the unrestricted variables, use the output of the `laws()` function. The initial conditions of the system give values for the constants in the conservation laws, and the specific steady state resulting from a specific set of initial conditions can be deduced.

If any formulas exist for nontrivial steady states of the reaction scheme, they will be expressed in the most general terms by this function. Thus, the procedure will give steady state conditions that depend on the rate constants, allowing for a more general analysis of the reaction scheme. For example, when the following reaction scheme is input:

```
R := [2*X>2*Y,Y+X>2*X]:
```

the procedure outputs:

```
{X(t) = 0, Y(t) = Y(t)}, {X(t) = X(t), Y(t) = 2*k1*X(t)/k2}
```

The `laws()` procedure outputs:

```
There is 1 conservation law.
The conservation law is:

                        X(t) + Y(t) = C1
```

In this example, the unrestricted value for `X(t)` is restricted by the conservation law, where the constant `C1` is determined by the initial conditions.

Implementation

This option takes the output of the `odes()` procedure and sets all the derivatives of the species equal to zero, the condition for a steady state solution. The `solve()` procedure is then run on the resultant system to find the conditions to be placed on the species to obtain a steady state solution.

4.7.2 eqpts - Numerical Solution - solve()

Description

Finds specific numerical steady state solutions to the reaction scheme using the `solve()` procedure. This procedure is the analog of the `General Solution` option of the `eqpts()` procedure, but this option produces steady states that depend on the specified rate constants and initial conditions.

Execution

Choosing this option in the `eqpts()` procedure using the Michaelis-Menten reactions previously defined returns the following to the *Maple* window:

```
The steady states are:
ss[1]:
                  [P = 10., C = 0., S = 0., E = 0.10000000000000000000]
```

However, if the input variables are instead defined using fractions instead of decimals, the output of this function is given in terms of fractions instead of decimals:

```
The steady states are:
ss[1]:
                                                1
                          [P = 10, C = 0, S = 0, E = --]
                                                10
```

Defining the input variables in terms of fractions instead of decimals can lead to higher accuracy when the steady state solution is expressed in a fractional form. This can be especially useful when the output of this function is used in the `find_stable()` procedure for finding the stability of steady states. If any steady states are found, they will be labeled with a variable name, `ss[i]`. Each steady state is saved to the list `ss` for use in other functions, specifically the `find_stable()` function.

Implementation

This option takes the reduced set of equations used in the plotting procedures and sets all the derivatives of the species equal to zero, the condition for a steady state solution. The `solve()` procedure is then run on the resultant system to find the conditions to be placed on the species to obtain a steady state solution. However, since the reduced equation set contains the undetermined constants of the conservation laws, the initial conditions and reaction rates are substituted into the system before it is solved, resulting in solutions that depend on the rate constants and initial conditions.

4.7.3 eqpts - Numerical Solution - Homotopy()

Description

Finds specific numerical steady state solutions to the reaction scheme using the `Homotopy()` procedure.

Execution

Choosing this option in the `eqpts()` procedure using the Michaelis-Menten reactions previously defined activates the following in the *Maplet* window shown earlier. Included in that window is the following:

This slider bar can be used to define the specific parameters used in the `Homotopy()` function that specify the tolerance and number of steps used to solve the system. Depending on the specific reaction scheme, choosing a level of computation too high or too low may result in the procedure not finding any steady state solutions. After choosing an appropriate level of computation, in this example 5, the following is returned to the *Maple* window:

```
The steady states are:
ss[1]:
                    [C = 0., P = 10.000, E = 0.1, S = 0.]
```

Since the `Homotopy()` procedure is a numerical procedure, the output will always be numerical, and no general formulas will be output by the procedure. If any steady states are found, they will be labeled with a variable name, `ss[i]`. Each steady state is saved to the list `ss` for use in other functions, specifically the `find_stable()` function. Due to the implementation of the procedure, the steady states it produces depend on the rate constants and initial conditions input with the reactions.

Implementation

The `Homotopy()` procedure in *Maple* is a numerical procedure that finds the zeroes of a set of polynomial equations. However, the `Homotopy()` procedure requires that the number of variables in the polynomial equations equal the number of equations in the system, which is not satisfied in general by the complete set of equations output by the `odes()` procedure. If a certain species appears only as a product in the reaction equations, it does not appear in the rates of any of the reactions, and is therefore not involved in any of the polynomial equations solved to find steady states. However, the reduced equation set used in the plotting procedures does satisfy the criteria for the `Homotopy()` procedure, so it is used in finding the steady states. However, introducing the reduced equation set also introduces the undetermined constants that appear in the conservation laws, which must be determined in order for the numerical `Homotopy()` procedure to find a solution. Thus, the final steady states found by the procedure depend on both the initial conditions and rate constants input by the user.

In order for steady state solutions to be found by this procedure, the solution space must consist of discrete points. Thus, from the output of the General Solution option of the `eqpts()` procedure,

the solution space of the Michaelis-Menten reaction scheme is two dimensional, since two different species can be varied independently. However, the Michaelis-Menten reaction scheme has two conservation laws, which reduces the solution space to zero dimensions and a discrete number of solutions.

The results of the `steady()` procedure are utilized in finding the steady state solutions with the `Homotopy()` function. Since the species returned by the `steady()` procedure are always zero at any steady state solution to the reaction scheme, the final solution output from the `Homotopy()` procedure takes this information into account. This helps avoid numerical rounding errors that may cause problems in other functions, such as the `find_stable()` procedure.

4.8 find_stable

Description

Determines the stability of a steady state solution of a reaction scheme.

Calling Sequence

`find_stable(steady_state)`
`steady_state` is a list that contains all species values at the steady state.
All three variables, `R`, `rate_constants`, and `initial`, must be defined to run the function.

Execution

Running the `find_stable(ss[1])` procedure after running the `eqpts() - Numerical Solution - Homotopy()` procedure using the Michaelis-Menten reactions previously defined returns the following to the *Maple* window:

```
Eigenvalues:
```

$$\begin{bmatrix} -10.1010000000000009 + 0.\ I \\ -19 \\ -6.79389999999999992\ 10 \quad + 0.\ I \\ -0.0990000000000000046 + 0.\ I \\ 0.\ + 0.\ I \end{bmatrix}$$

```
The specified steady state is stable.
```

The argument of the `find_stable` procedure must be a list of the species values at the steady state. In this example, `ss[1] := [C = 0., P = 10.000, E = 0.1, S = 0.]`, a variable defined by the `eqpts()` procedure. However, the user can input any list of species values to determine stability. As an example,

```
> find_stable([C=1,S=1,P=1,E=1]);

WARNING: Input is not a steady state solution. The following
species do not have a rate of zero:

                          d
                          -- C(t) = -9.1
                          dt

                          d
                          -- E(t) = 9.1
                          dt

                          d
                          -- S(t) = -.9
                          dt

                          d
                          -- P(t) = 10
                          dt

Eigenvalues:
                            -11.2080000000000002 + 0. I
```

```
                                -16
 2.84060000000000025 10    + 0. I
   -0.892240000000000033 + 0. I
                                -17
 1.91310000000000009 10    + 0. I
```

```
The specified steady state is not stable.
```

If the argument of the procedure is not a steady state, a warning is issued to the user, displaying the species whose rate is not zero. In some situations, if the argument to `find_stable` is not precise enough, a warning will result, even though the point does represent a steady state. Thus, if the rates displayed by the procedure are close enough to zero, within numerical error, the user can disregard the warning and use the stability analysis that follows. However, in this example, the input is not a steady state, and the species rates are not close to zero, so the stability analysis that follows does not apply. It is up to the user to decide whether or not to use the stability analysis if a warning is issued based on the nonzero rates output by the procedure.

In some cases, due to numerical error, the real part of an eigenvalue may be negligibly small, on the order of 10^{-15}, when the eigenvalue should be zero if exact steady state conditions were used. This may lead to the incorrect statement that the steady state is not stable, since the eigenvalue is not exactly zero. Thus, when investigating a steady state, if the procedure reports that it is not stable, the user may want to check the real parts of the eigenvalues to make sure numerical error did not lead to the report of instability.

Implementation

Let $\mathbf{S}$ be a vector of the species involved in a reaction scheme of dimension j by 1, and $\mathbf{S}_0$ be a vector that represents the value of each species at a steady state. Let the rate of the i^{th} species S_i be defined as

$$\frac{dS_i}{dt} = f_i(\mathbf{S}).$$

Expanding $f_i(\mathbf{S})$ in a Taylor series about $\mathbf{S} = \mathbf{S}_0$ gives

$$\frac{dS_i}{dt} = f_i(\mathbf{S}_0) + \sum_j \left[\left. \frac{\partial f_i}{\partial S_j} \right|_{\mathbf{S}_0} (\mathbf{S}_j - (\mathbf{S}_0)_j) \right] + \dots$$

However, $f_i(\mathbf{S}_0) = 0$ by the definition of $\mathbf{S}_0$. Thus, near $\mathbf{S} = \mathbf{S}_0$,

$$\frac{dS_i}{dt} \approx \vec{\nabla} f_i(\mathbf{S}_0) \cdot (\mathbf{S} - \mathbf{S}_0)$$

Or, using matrix notation,

$$\frac{d\mathbf{S}}{dt} \approx \left. \frac{\partial(f_1, \dots, f_j)}{\partial(S_1, \dots, S_j)} \right|_{\mathbf{S}_0} (\mathbf{S} - \mathbf{S}_0)$$

Introducing the variable $\hat{\mathbf{S}} = \mathbf{S} - \mathbf{S}_0$,

$$\frac{d\hat{\mathbf{S}}}{dt} \approx \left. \frac{\partial(f_1, \dots, f_j)}{\partial(S_1, \dots, S_j)} \right|_{\mathbf{S}_0} \hat{\mathbf{S}}$$

Thus, the stability of the system near $\mathbf{S}_0$ will be determined by the eigenvalues of the Jacobian matrix $\left. \frac{\partial(f_1,\dots,f_j)}{\partial(S_1,\dots,S_j)} \right|_{\mathbf{S}_0}$. The `find_stable` procedure finds the eigenvalues of this matrix, and the eigenvalues are output by the function. If the none of the real parts of the eigenvalues of the matrix are positive, the system is stable about $\mathbf{S}_0$. However, if $\mathbf{S}_0$ is not a steady state, the assumption that $f_i(\mathbf{S}_0) = 0 \ \forall \ 1 \leq i \leq j$ is not true, which is the reason behind the implementation of the warning given by the procedure if $\mathbf{S}_0$ is not a steady state.