Getting Started with Maple Toolboxes

Getting Started with Maple Toolboxes

Copyright

This document was produced using a special version of Maple and DocBook.

Printed in Canada

ISBN 978-1-897310-76-2

Contents

Preface

Introduction to the Toolboxes

Maplesoft® offers a rich selection of add-on products. These toolboxes enhance and extend the power of Maple™ by adding solutions in specialized application areas, enhanced computation and deployment options, and connectivity with other technical tools. When taking advantage of these advanced products, you continue to have access to Maple's computational power, easy-to-use smart document interface, and rich technical documentation features.

The **Maple Toolbox for MATLAB**® provides tight integration between Maple and MATLAB. It gives you direct access to all the commands, variables, and functions of each product, while allowing you to work in either environment.

BlockImporter™ **for Simulink** allows you to import a Simulink model into Maple, and convert it to a set of mathematical equations. Maple provides the power to simplify and manipulate the model, before simulating it in Maple or exporting it back to Simulink.

The **Global Optimization Toolbox** provides world-class global optimization technology to return the best answer to your optimization model, robustly and efficiently.

The **Grid Computing Toolbox** provides tools for performing Maple computations in parallel, allowing you to distribute computations across a network of workstations, a supercomputer, or the CPUs of a multiprocessor machine. It includes a personal grid server, allowing you to simulate and test your parallel applications before running them on a real grid network.

The **Financial Modeling Toolbox** provides a collection of tools for mathematical finance covering a wide range of common tasks, such as equity price modeling, term structure and cash flow analysis, option pricing, and stochastic processes.

MapleNet® provides a system for deploying live Maple documents over the internet. Powered by Maple, these interactive applications are accessible through a standard Web browser.

In This Manual

This manual provides an introduction to each of these toolboxes. Each section contains an overview of the toolbox, set-up details, getting started information, and examples.

1 Maple Toolbox for MATLAB

1.1 Introduction to the Maple Toolbox for MATLAB

Overview

The Maple Toolbox for MATLAB provides tight integration between Maple and MATLAB. It gives you direct access to all the commands, variables, and functions of each product, while allowing you to work in either environment.

Features of this toolbox include:

- The combined functionality of Maple and MATLAB commands.
- Access to hundreds of built-in Maple packages, including statistics, optimization, and enhanced DE solvers, as well as the ability to include units and tolerances in your expressions, from the environment of your choice.
- The freedom to choose the user interface that best suits your task: Maple's technical document interface, where embedded plots, text, annotations, units, and many other features are all live within an executable document, or the MATLAB environment, using the command-based language to access Maple.

Requirements

Before installing the Maple Toolbox for MATLAB, you must install and activate Maple 13. You also need to install MATLAB before using this toolbox. For details on supported MATLAB versions and installation instructions, see the Install.html file on the product CD.

You need a purchase code to activate the Maple Toolbox for MATLAB. This code has been emailed to you. If you have not received your purchase code, contact Maplesoft Customer Service at custservice@maplesoft.com (US and Canada), or the Maplesoft reseller in your region. For a list of resellers, visit **http://www.maplesoft.com/contact/international/index.aspx**.

To open the main Maple Toolbox for MATLAB help page, start MATLAB and enter the following command:

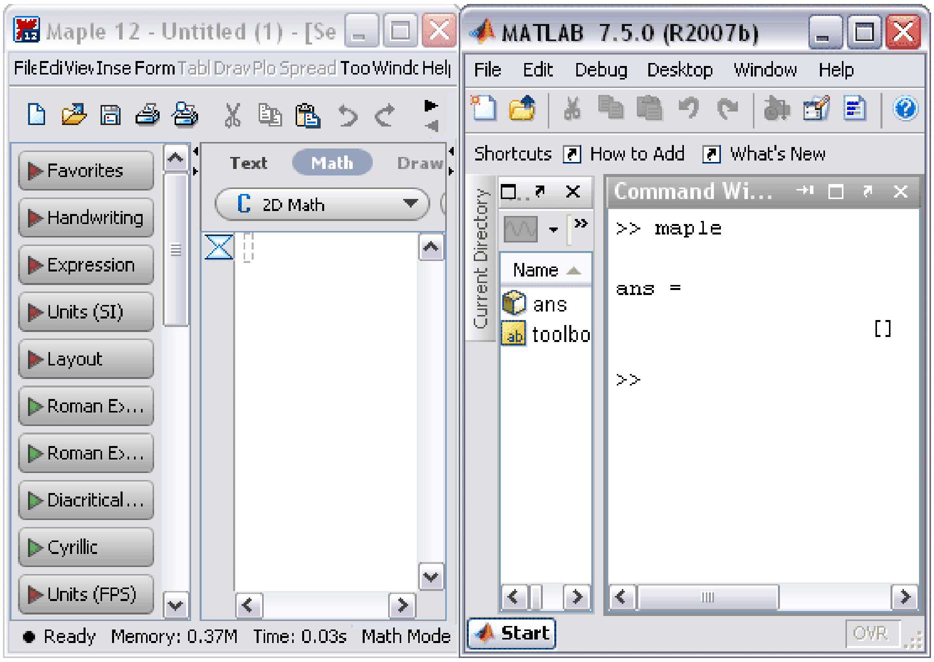
```
>> doc maple_overview
```

The overview page contains links to getting started documentation, examples and features, help system documentation, and more. The getting started documentation contains an overview of the tools in this product, and provides introductory examples.

1.2 Getting Started with the Maple Toolbox for MATLAB

Establishing a Connection

To start using the Maple Toolbox for MATLAB, first open a MATLAB window. Enter the command **maple** in the MATLAB command window. This opens a new Maple window and establishes a connection between Maple and MATLAB. You are now ready to start working in Maple and MATLAB together.


Maple 12 - Untitled (1) - [Se
File Edit View Insert Format Table Draw Plot Spreadsheet Tools Window Help
Favorites
Handwriting
Expression
Units (SI)
Layout
Roman E>...
Roman E>...
Diacritical...
Cyrillic
Units (FPS)
Text
Math
Draw
2D Math
Ready Memory: 0.37M Time: 0.03s Math Mode
MATLAB 7.5.0 (R2007b)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Current Directory
Name
ans
toolbo
Command Wi...
Start
OVR

```
>> maple

ans =

                    []

>>
```

Help with the Maple Toolbox for MATLAB

For a list of commands in Maple to support the Maple Toolbox for MATLAB, see the **?MTM** help page.

1.3 Working with the Maple Toolbox for MATLAB

Once the connection has been established between Maple and MATLAB, you can define variables and functions in one program, and then perform calculations on them in another. A few new commands have been introduced for this purpose, in particular, **setmaple**.

This simple example demonstrates the basic connection between the two programs.

1. In the MATLAB command window, define an expression. For example:

```
>> syms x y
>> cos(sqrt(x^2-y^2))/x^2

ans =

                  2     2 1/2
              cos((x  - y )    )
              -----------------
                      2
                     x
```

2. To make this expression available in Maple, you must give the expression a name. To give the expression the name *h*, use the **setmaple** command.

```
>> setmaple('h', ans)
```

3. Now you can access the expression from Maple. Switch to the Maple window and make sure the input format is **Math**. Type *h*, then press <**Ctrl**> + **=** (<**Command**> + **=**, Mac) to display the expression inline. Alternatively, press <**Enter**> to display the expression on the next line.

Verify that the variable *h* was properly set before moving on.

$$h = \frac{\cos\left(\sqrt{x^2 - y^2}\right)}{x^2}$$

4. You can use the context-sensitive menus in Maple to manipulate the expression. Right-click on the expression to see a list of possible actions. For example, to differentiate with respect to *x*, go to **Differentiate** > **x**.

$$h = \frac{\cos\left(\sqrt{x^2 - y^2}\right)}{x^2} \xrightarrow{\text{diff w.r.t. x}} -\frac{\sin\left(\sqrt{x^2 - y^2}\right)}{\sqrt{x^2 - y^2}\; x} - \frac{2\cos\left(\sqrt{x^2 - y^2}\right)}{x^3}$$

5. You can make this new expression available in MATLAB by using various methods. One method is to assign it a name by using the context-sensitive menu. Right click on the derivative and select **Assign to a Name**. Enter *k* in the dialog.

$$h = \frac{\cos\left(\sqrt{x^2 - y^2}\right)}{x^2} \xrightarrow{\text{diff w.r.t. x}} -\frac{\sin\left(\sqrt{x^2 - y^2}\right)}{\sqrt{x^2 - y^2}\; x} - \frac{2\cos\left(\sqrt{x^2 - y^2}\right)}{x^3}$$

$$\xrightarrow{\text{assign to a name}} -\frac{\sin\left(\sqrt{x^2 - y^2}\right)}{\sqrt{x^2 - y^2}\; x} - \frac{2\cos\left(\sqrt{x^2 - y^2}\right)}{x^3}$$

6. Return to the MATLAB window, where you can access the expression *k* and perform further operations on it. For example, you can factor the expression.

```
>> syms k
>> k

k =
                   2    2 1/2            2    2 1/2
             sin((x  - y )   )     cos((x  - y )   )
           - ----------------- - 2 -----------------
                2    2 1/2                 3
              (x  - y )     x             x
>> factor(k)

ans =
                          1/2   2
  - (sin(((x - y) (x + y))   ) x

                             1/2                   1/2   /
    + 2 cos(((x - y) (x + y))    )((x - y) (x + y))   ) /
                                                       /
                          1/2  3
       ( ((x - y) (x + y))    x )
```

Alternatively, you could have factored the expression in Maple by using the context-sensitive menu, or defined the original expression in Maple.

The following example illustrates how you can use Maple and MATLAB together to increase your productivity and the reliability of your results.

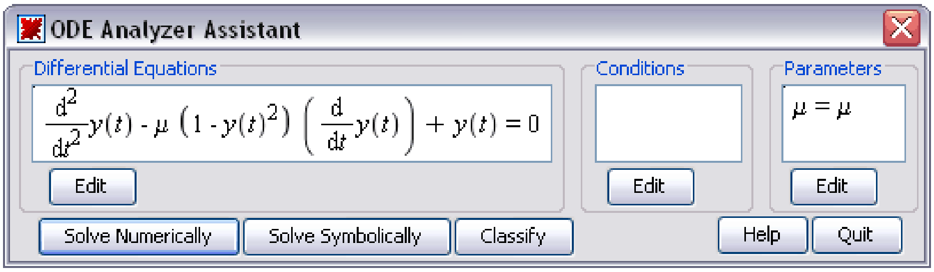
Example

Consider the Van der Pol equation, a non-linear second order differential equation. Traditionally in MATLAB, this equation needs to be linearized before it can be solved using MATLAB's ODE solvers; then some programming is required to obtain the final solution. Maple can perform computations with higher order differential equations, entered in a natural notation. Solutions from Maple can be pulled into MATLAB for further viewing and manipulation, by using the Maple Toolbox for MATLAB.

1. To begin, open a new Maple Document in the Maple window and define the Van der Pol equation by using dot notation. To enter dot notation, type <**Ctrl**> + " (<**Command**> + ", Mac), and use a period to represent a dot, or a differential equation in terms of time. Make sure the input format is **Math**.

$$\ddot{y} - \mu \cdot \left(1 - y^2\right) \cdot \dot{y} + y = 0$$

2. To solve this differential equation, right-click on the equation to see the context-sensitive menu, and select **Solve DE Interactively**. The Interactive ODE Assistant opens.



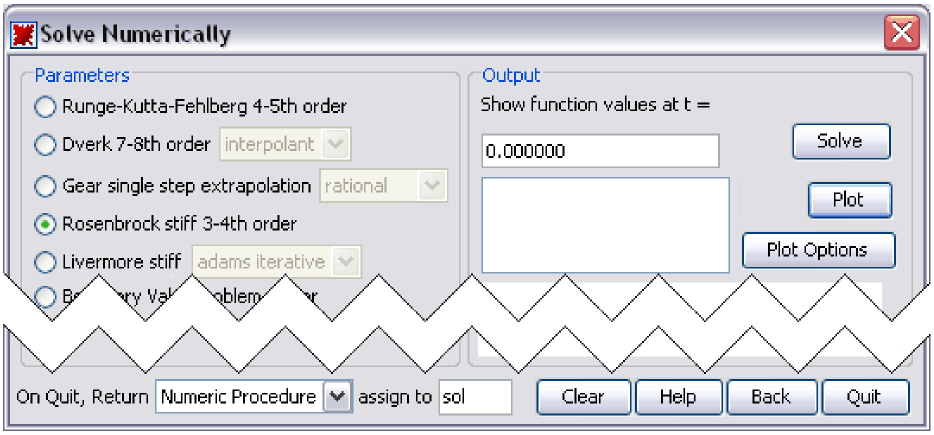
$$\frac{d^2}{dt^2}y(t) - \mu\left(1 - y(t)^2\right)\left(\frac{d}{dt}y(t)\right) + y(t) = 0$$

$\mu = \mu$

3. Within this assistant, you can define some initial conditions, enter a value for the parameter, and choose the solving method. For this example, enter $y(0) = 1$ and $y'(0) = 0$ as initial conditions, and set the parameter

$\mu = 7.5$.

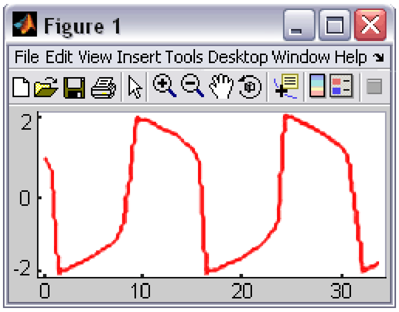
If you choose to solve numerically, a dialog appears in which you can specify a solving method to use. Select **Rosenbrock stiff 3-4th order** as the solver.


Solve Numerically
Parameters
Runge-Kutta-Fehlberg 4-5th order
Dverk 7-8th order
interpolant
Gear single step extrapolation
rational
Rosenbrock stiff 3-4th order
Livermore stiff
adams iterative
Output
Show function values at t =
0.000000
Solve
Plot
Plot Options
On Quit, Return
Numeric Procedure
assign to
sol
Clear
Help
Back
Quit

4. You can choose the form of the output from the solver, in the lower left corner of the window. For this example, return the **numeric procedure**, so that it can be used in MATLAB. Assign it the name *sol*, to make the procedure available in MATLAB.

5. When you quit the assistant, the numeric procedure to solve the differential equation is entered in your document. Then you can access the solution in MATLAB, and visualize it, with the following commands. The resulting plot appears in a new MATLAB window, as shown below.

```
>> syms sol
>> figure(1);maple('plots[odeplot]',sol,'0..35')
```



You can also manipulate the solution, and perform further calculations on it, in Maple or in MATLAB.

With the basic tools shown here, you are now ready to use the Maple Toolbox for MATLAB to solve many mathematical problems. See the help pages in Maple and MATLAB for more information about the commands used in this guide, and for access to other examples and functionality to help you learn more about the Maple Toolbox for MATLAB.

2 BlockImporter for Simulink

2.1 Introduction to BlockImporter for Simulink

Overview

BlockImporter for Simulink allows you to import a Simulink model into Maple, and convert it to a set of mathematical equations. Maple provides the power to simplify, manipulate, and simulate the model.

Features of this toolbox include:

- The ability to perform calculations and simplifications on your Simulink models by using Maple's suite of tools.
- Perform advanced analysis on your systems by using Maple's built-in DynamicSystems package, including system object representation, graphical analysis, system manipulation procedures, signal generation tools, and system simulation.
- Reliable, tested computational routines to test the validity of your models, before performing simulations.
- Commands for performing frequency- and complex-domain analysis, stability and sensitivity analysis, and parameter optimization on your Simulink models.
- Symbolic methods for eliminating algebraic loops, which can significantly improve the efficiency and execution speed of your Simulink models.

Requirements

Before installing BlockImporter for Simulink, you must install and activate Maple 13. You must also install MATLAB and Simulink before using this toolbox. For details on supported Simulink versions and installation instructions, see the Install.html file on the product CD.

You need a purchase code to activate BlockImporter for Simulink. This code has been emailed to you. If you have not received your purchase code, contact Maplesoft Customer Service at custservice@maplesoft.com (US and Canada), or the Maplesoft reseller in your region. For a list of resellers, visit **http://www.maplesoft.com/contact/international/index.aspx**.

After installation, the BlockImporter.mw file appears on your desktop (Windows and Macintosh) or in your home directory (Linux). Open this file. Note that you can also access this file by entering **?BlockImporter,Tour** in Maple.

The document BlockImporter.mw contains links to getting started documentation, help system documentation, and more. The getting started documentation contains an overview of the tools in this product, and provides multiple examples. The examples illustrate how to define a model in Simulink, import the mathematical model into Maple, and perform analyses such as stability, sensitivity, and parameter optimization.

2.2 Getting Started with BlockImporter for Simulink

Establishing a Connection

To begin, open a new Maple window. Enter the following command to establish a connection with MATLAB.

```
> Matlab[openlink]()
```

A MATLAB command window should open. If one does not, follow the instructions in the **?Matlab[setup]** help page to configure the connection.

You are now ready to use BlockImporter for Simulink.

Help with BlockImporter

For a list of commands in Maple to support BlockImporter for Simulink, see the **?BlockImporter** help page. Links to related Maple commands are also provided; many built-in Maple functions, especially the commands in the Dynamic Systems package, can be extremely helpful in generating and manipulating physical system models. For more information, refer to the **?DynamicSystems** help page.

2.3 Working with BlockImporter for Simulink

You can import any Simulink system or subsystem model by using the **Import** command. For more information, refer to the **?BlockImporter/Import** help page.

1. Set the directory in which the Simulink files are located. For example, use the directory containing the pre-installed BlockImporter example models, which can be located by using the following command.

> $datadir := BlockImporter[DataDirectory]()$

$datadir :=$
"C:\Program Files\Maple 12\toolbox\BlockImporter\data"

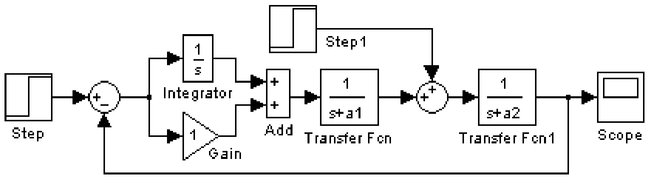
2. Using the **Import** command, specify the file to import. The *inplace* option indicates that the model will be converted in MATLAB to a Maple record. For more information, refer to the **?Record** help page.

> $with(BlockImporter)$

$[BuildDE, Import, PrintSummary, SimplifyModel]$ (2.2)

> $sys := Import(\text{"example"}, path = datadir, init = \text{"example\_init"}, inplace = true):$

This particular model has the following model diagram in Simulink.



3. A copy of the model is created, consisting of a list of mathematical equations. Use the **PrintSummary** command to display the equations, variables, and parameters. For more information, refer to the **?BlockImporter/PrintSummary** help page.

> *PrintSummary*(*sys*) :

```
Equations [24]:
```

$$\Big[u_{10, 1, 1} = y_{2, 1, 1}, u_{2, 2, 1} = y_{3, 1, 1}, u_{2, 1, 1} = y_{4, 1, 1}, u_{8, 1, 1} = y_{6, 1, 1}, u_{9, 1, 1} = y_{7, 1, 1}, u_{4, 1, 1} = y_{8, 1, 1}, u_{3, 1, 1} = y_{8, 1, 1}, u_{11, 1, 1} = y_{9, 1, 1}, u_{9, 2, 1} = y_{10, 1, 1}, u_{5, 1, 1} = y_{11, 1, 1}, u_{8, 2, 1} = y_{11, 1, 1}, y_{2, 1, 1} = u_{2, 1, 1} + u_{2, 2, 1}, y_{3, 1, 1} = u_{3, 1, 1}, \mathrm{D}\big(x_{4, 1}\big) = u_{4, 1, 1}, y_{4, 1, 1} = x_{4, 1}, Sink_{Scope, 5, 1, 1} = u_{5, 1, 1}, y_{6, 1, 1} = Source_{Step, 6, 1}, y_{7, 1, 1} = Source_{Step, 7, 1}, y_{8, 1, 1} = u_{8, 1, 1} - u_{8, 2, 1}, y_{9, 1, 1} = u_{9, 1, 1} + u_{9, 2, 1}, \mathrm{D}\big(x_{10, 1}\big) = u_{10, 1, 1} - K_{0, \text{"a1"}}\, x_{10, 1}, y_{10, 1, 1} = x_{10, 1}, \mathrm{D}\big(x_{11, 1}\big) = u_{11, 1, 1} - K_{0, \text{"a2"}}\, x_{11, 1}, y_{11, 1, 1} = x_{11, 1}\Big]$$

```
State Variables [3]:
```

$$\big[\big[x_{4, 1}, x_{10, 1}, x_{11, 1}\big]\big]$$

```
Initial Equations [3]:
```

$$\big[\big[x_{4, 1}([0]) = 0, x_{10, 1}([0]) = 0, x_{11, 1}([0]) = 0\big]\big]$$

```
Source Equations [2]:
```

$$\left[\mathit{Source}_{\mathit{Step},\,6,\,1} = \begin{cases} 0 & t < 1 \\ 1 & \mathit{otherwise} \end{cases}, \mathit{Source}_{\mathit{Step},\,7,\,1} = \begin{cases} 0 & t < 4 \\ 0.1 & \mathit{otherwise} \end{cases} \right]$$

```
Input Variables [2]:
```

$$\left[\mathit{Source}_{\mathit{Step},\,6,\,1}, \mathit{Source}_{\mathit{Step},\,7,\,1} \right]$$

```
Output Variables [1]:
```

$$\left[\mathit{Sink}_{\mathit{Scope},\,5,\,1,\,1} \right]$$

```
Parameters [2]:
```

$$\left[K_{0,\,\text{"a1"}} = 2.,\, K_{0,\,\text{"a2"}} = 1. \right] \tag{2.3}$$

4. Use Maple's built-in simplification and simulation routines, especially those in the DynamicSystems package, to manipulate the model, as shown in the next example.

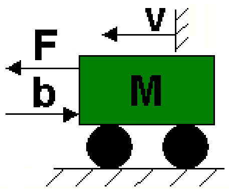
The following example provides details on how you can use BlockImporter to improve your Simulink models.

Example

Consider a simple example of a car traveling on a flat road as shown in the diagram below, where

- V is the horizontal velocity of the car (in m/s)
- F is the force created by the car's engine to propel it forward (in N)

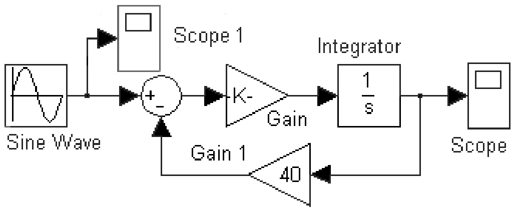
- b is the damping coefficient for the car, which is dependent on wind resistance, wheel friction, and other factors (in N*sec/m); assume the damping force to be proportional to the car's velocity
- M is the mass of the car (in kg)



Assume that $M = 1000\ kg$ and $b = 40\ N \cdot \frac{\sec}{m}$, then the system equation is

$$M \cdot \frac{d}{dt} V(t) = F(t) - b \cdot V(t)$$

The basic Simulink block diagram is shown below.



To begin, load the BlockImporter package.

> *with*(*BlockImporter*)

[*BuildDE*, *Import*, *PrintSummary*, *Simp*, *SimplifyModel*] (2.4)

Next, import the Simulink model into Maple. Specify the name of the model, as well as the MATLAB script that initializes the variable names.

> *model1* := *Import*("FirstOrderExample", *path* = "C:\\matlabfiles", *init* = "FirstOrderExample", *inplace* = *true*) :

Before displaying the model, you can simplify it to reduce the number of equations.

> *SimplifiedModel* := *SimplifyModel*(*model1*) :

> *PrintSummary*(*SimplifiedModel*)

```
Equations [3]:
```

$$\left[\mathrm{D}\left(x_{4,\,1}\right) = 100\, \mathit{Sink}_{\mathit{Scope},\, 6,\, 1,\, 1} - 4000\, x_{4,\,1},\ \mathit{Sink}_{\mathit{Scope},\, 5,\, 1,\, 1} = x_{4,\,1},\ \mathit{Sink}_{\mathit{Scope},\, 6,\, 1,\, 1} = \mathit{Source}_{\mathit{Sin},\, 7,\, 1}\right]$$

```
State Variables [1]:
```

$$\left[x_{4,\,1}\right]$$

```
Initial Equations [1]:
```

$$\left[x_{4,\,1}(0) = 0\right]$$

```
Source Equations [1]:
```

$$\left[\mathit{Source}_{\mathit{Sin},\, 7,\, 1} = \sin(t)\right]$$

```
Input Variables [1]:
```

$$\left[\mathit{Source}_{\mathit{Sin},\, 7,\, 1}\right]$$

```
Output Variables [2]:
```

$$\left[\mathit{Sink}_{\mathit{Scope},\, 6,\, 1,\, 1},\ \mathit{Sink}_{\mathit{Scope},\, 5,\, 1,\, 1}\right] \tag{2.5}$$

Finally, to simulate this system in Maple, build a set of differential equations from the system, and assign the result to a variable *sys1*.

> *sys1* := *BuildDE*(*SimplifiedModel*) :

The following table illustrates the different components of the variable *sys1*.

Differential Equations	> $print(sys1[1])$ $\left[\frac{d}{dt} x_{4,1}(t) = 100\, Sink_{Scope,6,1,1}(t) - 4000\, x_{4,1}(t), Sink_{Scope,5,1,1}(t) = x_{4,1}(t), Sink_{Scope,6,1,1}(t) = Source_{Sin,7,1}(t)\right]$ (2.6)
Differential Equations with the parameter values substituted in	> $print(sys1[2])$ $\left[\frac{d}{dt} x_{4,1}(t) = 100\, Sink_{Scope,6,1,1}(t) - 4000\, x_{4,1}(t), Sink_{Scope,5,1,1}(t) = x_{4,1}(t), Sink_{Scope,6,1,1}(t) = Source_{Sin,7,1}(t)\right]$ (2.7)
Initial conditions	> $print(sys1[3])$ $\left[x_{4,1}(0) = 0\right]$ (2.8)
Equations for the sources (inputs)	> $print(sys1[4])$ $\left[Source_{Sin,7,1}(t) = \sin(t)\right]$ (2.9)

List of sinks (outputs)	> $print(sys1[5])$ $\left[Sink_{Scope,\,6,\,1,\,1}(t),\ Sink_{Scope,\,5,\,1,\,1}(t)\right]$ (2.10)

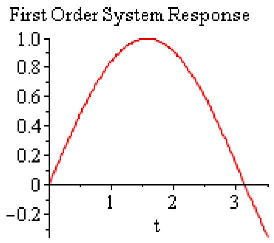
Using this information, you can construct a simulation procedure. You can use the command below, or use the ODE analyzer assistant for a step-by-step method of solving this DE. For more information, refer to the **?dsolve/interactive** help page.

```
> sol1 := dsolve([eval(sys1[2], sys1[4])[ ], sys1[3][ ]],
      numeric)
```

$$\mathbf{proc}(x\_rkf45\_dae)\ \ldots\ \mathbf{end\ proc} \tag{2.11}$$

Plot the simulation results.

```
> plots[odeplot](sol1, [t, sys1[5][1]], 0 ..3.5, numpoints
      = 200, title = "First Order System Response")
```



Now you can manipulate and simulate the model further in Maple.

With the basic tools outlined in these examples, you are now ready to use BlockImporter to create many engineering design solutions. See the Maple help system for more information about the commands used in this guide, or more ways in which BlockImporter for Simulink can help you.

3 Global Optimization Toolbox

3.1 Introduction to the Global Optimization Toolbox

Overview

The Global Optimization Toolbox provides world-class global optimization technology to return the best answer to your optimization model, robustly and efficiently.

Features of this toolbox include:

- Several solver modules for nonlinear optimization problems, including branch-and-bound global search, global adaptive random search, multi-start based global random search, and a generalized reduced gradient local search.
- The ability to solve models with thousands of variables and constraints, with Maple's arbitrary precision capabilities in their calculations, to greatly reduce numerical instability.
- Support for arbitrary objective and constraint functions, including those defined in terms of special functions (for example, Bessel or hypergeometric), derivatives and integrals, piecewise functions, and Maple procedures.
- The interactive GlobalOptimization Assistant, an easy-to-use interface for entering your problem and choosing the solving method.
- Maple's built-in model visualization capabilities for viewing one- or two-dimensional subspace projections of the objective function, with visualization of the constraints as planes or lines on the objective surface.

Requirements

Before installing the Global Optimization Toolbox, you must install and activate Maple13. For details and installation instructions, see the Install.html file on the product CD.

You need a purchase code to activate the Global Optimization Toolbox. This code has been emailed to you. If you have not received your purchase code, contact Maplesoft Customer Service at custservice@maplesoft.com (US and Canada), or the Maplesoft reseller in your region. For a list of resellers, visit **http://www.maplesoft.com/contact/international/index.aspx**.

After installation, enter **?GlobalOptimization** in Maple to see an overview of the toolbox.

3.2 Getting Started with the Global Optimization Toolbox

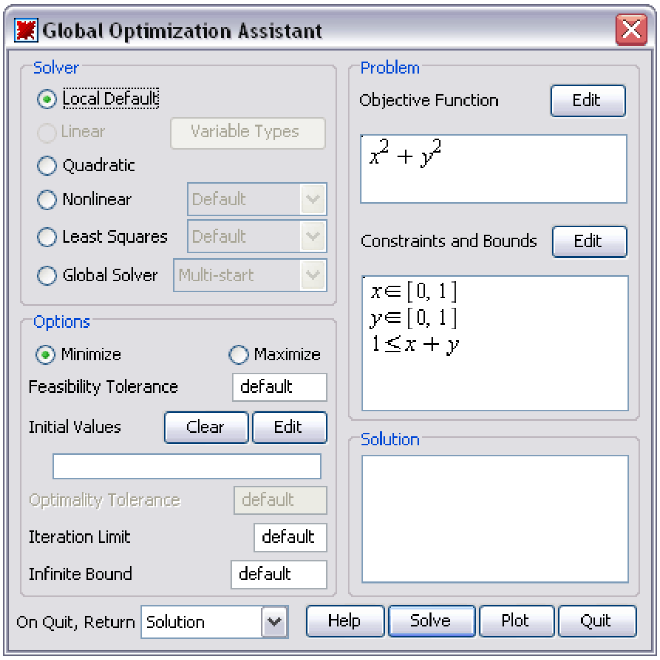
Initialization

To start using the Global Optimization Toolbox, open a new Maple window and load the **GlobalOptimization** package by using the following command.

$> \ with(GlobalOptimization);$

$$[GetLastSolution, GlobalSolve, Interactive] \tag{3.1}$$

You can either use the commands to solve your pre-defined global optimization problem, or use the Global Optimization Assistant, shown below. The following examples demonstrate the use of the commands, but the assistant could also be used to solve any of the problems presented.


Global Optimization Assistant

Solver: [✓ Local Default] Linear Quadratic Nonlinear (Default) Least Squares (Default) Global Solver (Multi-start)

Options: [✓ Minimize] Maximize

Feasibility Tolerance: default

Initial Values:

Optimality Tolerance: default

Iteration Limit: default

Infinite Bound: default

Problem

Objective Function: $x^2 + y^2$

Constraints and Bounds:
$x \in [0, 1]$
$y \in [0, 1]$
$1 \leq x + y$

Solution:

On Quit, Return: Solution

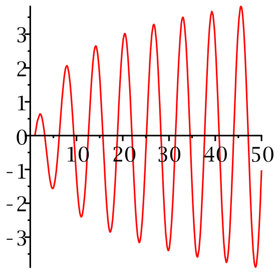
Help with the Global Optimization Toolbox

For help with the commands in the Global Optimization Toolbox, see the **?GlobalOptimization** help page. You may also find the related commands in the built-in Optimization package helpful for solving optimization problems. For more information, refer to the **?Optimization** help page.

3.3 Working with the Global Optimization Toolbox

In many optimization problems, simple methods are not sufficient to find the best solution. They will only find a local optimum, usually the one closest to the search starting point, which is often given by the user. For example, consider the expression $\ln(x)\cdot\sin(x)$.

```
> plot(ln(x)·sin(x), x = 1 ..50)
```



You could likely approximate the global minimum in the given domain, and find that minimum easily, by simply looking at the plot. But if you were unable to properly approximate it, or if you approximated incorrectly, you would not find the global minimum by using the usual optimization techniques.

```
> Optimization[Minimize](ln(x)·sin(x), x = 1 ..50)
```

```
[-3.39618690740209404, [x = 29.8549920107436756]]        (3.2)
```

According to the **Minimize** command, the minimum is at approximately *x* = 30. However, you can see in the plot above that this is not the global minimum.

By using the global optimization equivalent of this command, you can be assured that you have found the global minimum in the specified interval.

> *with*(*GlobalOptimization*) :

> $GlobalSolve(\ln(x) \cdot \sin(x), x = 1 ..50)$

$$[-3.88562417153593120, [x = 48.6999705692544112]] \tag{3.3}$$

To view the solving methods that were used to determine the global minimum, change the *infolevel* variable and re-execute the command. At *infolevel* = 3, the command displays the input model size and type, solver operational mode and parameter, and detailed runtime information.

> $infolevel[GlobalOptimization] := 3$:

> $GlobalSolve(\ln(x) \cdot \sin(x), x = 1 ..50)$

```
GlobalSolve: calling NLP solver
SolveGeneral: calling global optimization solver
SolveGeneral: number of problem variables 1
SolveGeneral: number of nonlinear inequality constraints 0
SolveGeneral: number of nonlinear equality constraints 0
SolveGeneral: method multistart
SolveGeneral: merit function evaluation limit 1000
SolveGeneral: non-improving merit function evaluation limit 200
SolveGeneral: constraint penalty multiplier 100.0
SolveGeneral: target merit function value -0.10e11
SolveGeneral: local search target objective function value -0.10e11
SolveGeneral: local search feasibility tolerance 0.10e-5
SolveGeneral: local search optimality tolerance 0.10e-5
SolveGeneral: time limit in seconds 100
SolveGeneral: trying evalhf mode
LGORUN: total number of function evaluations 1120
LGORUN: runtime in external solver 0.
LGORUN: maximum constraint infeasibility 0.
LGORUN: cycling or stall detected in solver
```

$$[-3.88562417153593120, [x = 48.6999705692544112]] \tag{3.4}$$

Typically, the *infolevel* is set to 0, the default.

> $infolevel[GlobalOptimization] := 0:$

The following is an example of a situation in which you cannot approximate the global minimum by using linear methods; however, the global solver finds the best solution.

Example

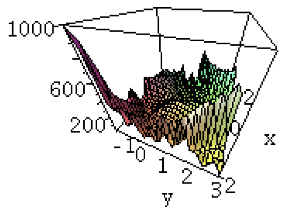
Consider a non-linear system of equations.

> $eq1 := \sqrt{x^2 + y^4} + e^{x - y^2} + 5 \sin(2\,x - 4\,x\,y)$
> $\quad - 12 \cos(x\,y):$

> $eq2 := 5 \ln\left(1 + x^2\right) + e^{-y + x} + 5 \sin(6\,x\,y):$

The induced least-squares error function is highly multi-extremal, making it difficult to minimize the error of the least squares approximation.

> $plot3d\left(eq1^2 + eq2^2, x = -2 ..2, y = -1 ..3, grid = [30, 30],\right.$
> $\quad \left. lightmodel = light4, axes = boxed\right)$



To determine the global minimum of the least-squares error, define the objective function and constraints to be optimized.

> $objf := eq1^2 + eq2^2$:

> $cons := \{eq1, eq2\}$:

First, try to find a local solution by using Maple's built-in Optimization package. This system is sufficiently complex that linear optimization solvers cannot find a feasible solution.

> $localsoln := Optimization[Minimize](objf, \{eq1 = 0, eq2 = 0\}, x = -1 ..2, y = -2 ..1)$

```
Error, (in Optimization:-NLPSolve) no improved point could be found
```

However, global optimization techniques can be used to find a global minimum.

> $sol := GlobalSolve(objf, \{eq1 = 0, eq2 = 0\}, x = -1 ..2, y = -2 ..1)$:
$sol[1]; sol[2]$

$$5.25083559996863880\ 10^{-23}$$

$$[x = -0.558972497937727298, y = -1.58234523718627806] \quad (3.5)$$

Substitution into the constraints shows that the least squares approximation can be a fairly precise solution. That is, the error of the least squares approximation is very small at the minimum point.

> $eval(cons, sol[2])$

$$\{-2.\ 10^{-9}, -5.\ 10^{-9}\} \quad (3.6)$$

With the examples demonstrated here, you are now ready to use the Global Optimization Toolbox to solve many complex mathematical problems. See the Maple help system for more information about the commands used in this guide, or more ways in which the Global Optimization Toolbox can help you.

4 Grid Computing Toolbox

4.1 Introduction to the Grid Computing Toolbox

Overview

The **Grid Computing Toolbox** provides tools for performing Maple computations in parallel, allowing you to distribute computations across a network of workstations, a supercomputer, or the CPUs of a multiprocessor machine. It includes a personal grid server, allowing you to simulate and test your parallel applications before running them on a real grid network.

Features of this toolbox include:

- A self-assembling grid in local networks, with an easy-to-use interactive interface for launching parallel jobs.
- Integration with job scheduling systems, such as PBS.
- High-level parallelization operations, such as *map* and *seq*, as well as a generic, parallel divide-and-conquer algorithm.
- Automatic deadlock detection and recovery.

Requirements

Before installing the Grid Computing Toolbox, you must install and activate Maple 13. For details and installation instructions, see the Install.html file on the product CD.

You need a purchase code to activate the Grid Computing Toolbox. This code has been emailed to you. If you have not received your purchase code, contact Maplesoft Customer Service at custservice@maplesoft.com (US and Canada), or the Maplesoft reseller in your region. For a list of resellers, visit **http://www.maplesoft.com/contact/international/index.aspx**.

Note that if you do not have a valid Grid license, the Grid Computing Toolbox will run in a limited capacity, allowing at most one server per machine and eight nodes in a job.

After installation, the GridComputing.mw file appears on your desktop (Windows and Macintosh) or in your home directory (Linux). Open this file.

4.2 Getting Started with the Grid Computing Toolbox

Establishing a Server and Network

You can run a server on your desktop machine to develop and test your parallel applications. The Personal Grid Server worksheet is an interactive Maple document that can help you start a personal server with any number of nodes. For more information, refer to the **?Grid/wks/PersonalGridServer** help page.

The following Maple commands are a guideline for configuring your computer for a personal server. The given broadcast mask should ensure that you will not be added to any Grid networks. Note that these settings can be set by default. See the Install.html file for instructions. Configuration details can be found on the **?Grid/Properties** help page.

```
> host := "localhost" :
  port := 2000 :
  numNodes := 3 :
  broadcastAddress := "127.0.0.255" :
  broadcastPort := 4400 :
  logFile := "logs/gridlog.txt" :

> Grid[Setup]( host, port )
```

$$2000 \tag{4.1}$$

To run Grid Computing applications on a network of machines, you must start Grid servers on each machine that you want to be part of the network. The toolbox will automatically detect new machines as they are added to the network, or you can turn off auto-discovery for use in a controlled environment, such as a PBS setup.

You will need to re-configure the Grid so that you can create or be added to a Grid network. Change the *broadcastAddress* and *broadcastPort* to your network's settings, and then start a server with those settings.

A few simple interactive examples using multiple nodes over multiple machines are in the Grid Computing worksheet. For more information, refer to the **?Grid/wks/Grid** help page.

You are now ready to use the Grid Computing Toolbox, either on your own computer as a test server, or on a network of computers to run parallel applications.

Help with Grid Computing

For a list of commands in Maple to support the Grid Computing Toolbox, see the **?Grid** help page.

4.3 Working with the Grid Computing Toolbox

As mentioned in the previous section, a few simple examples using multiple nodes over multiple machines can be found in the Grid Computing worksheet. For more information, refer to the **?Grid/wks/Grid** help page.

The following provides a sample of the commands required to execute such examples. To begin, start a personal server using the local settings from the previous section, or use your network settings to connect with other computers running servers in the Grid Computing Toolbox. The code for this example can be found in the Grid Computing worksheet.

```
> with(Grid) :

> Server[StartServer](port, numNodes, broadcastAddress,
      broadcastPort, logFile)
                                3748                              (4.2)
```

Define the application or procedure that you want to perform, call it *sampleCode*. Then execute the following commands to deploy the application *sampleCode* over the network.

```
> fname := cat(kernelopts(toolboxdir = "Grid"),
     "/samples/Simple.mpl") :

> sampleCode := FileTools[Text][ReadFile](fname) :

> AvailableNodes := Status( )[2]
                         AvailableNodes := 4                      (4.3)
```

> $result := Launch(AvailableNodes, sampleCode, printf,$
> $\quad \mathbf{proc}(\,)\ false\ \mathbf{end}\ )$

```
Node 2: This is node 2 out of 4
Node 3: This is node 3 out of 4
Node 0: This is node 0 out of 4
Node 1: This is node 1 out of 4
```

$$result := [foo0bar, foo1bar, foo2bar, foo3bar] \tag{4.4}$$

When you are finished, make sure to close the connection before you close Maple, by executing the following command.

> $Server[StopServer](2000, 4)$

$$0 \tag{4.5}$$

The following is a more detailed example of how you can use the Grid Computing Toolbox to create and perform parallel applications on a network of computers.

Example

Consider a fractal, specifically a Julia set, that you want to create and display in Maple. For any more than a few iterations, this can take a long time on a single computer. But you can divide the construction between an arbitrary sized network of computers by using the Grid Computing Toolbox.

First define a Julia set as a procedure, which will determine the color of the plot at point (a, b). A Julia set is formally defined as

$$\left\{ z \in C \,\middle|\, z_{i+1} = z_i^2 + c \text{ converges for } z_0 = z \right\}$$

where c is a constant.

```
> JuliaSet := proc(a, b) global c1, c2;
   local z1, z2, z1s, z2s, m;
    (z1, z2) := (a, b) :
    z1s := z1^2 : z2s := z2^2;
    for m from 1 to 30 while z1s + z2s < 4 do
      (z1, z2) := (z1s - z2s + c1, 2*z1*z2 + c2) :
      z1s := z1^2 : z2s := z2^2;
    end do;
    m;
  end proc:
```

Here it is assumed that for *m* over 30, the sequence *z* converges. Increasing this number will make the computation more accurate, but take much more time. Define the constant $c = c1 + I \cdot c2$ and an appropriate view and grid size for this fractal. If *c* is changed, then *v* and *gridsize* may also need to be changed for the fractal to display properly.

```
> c1, c2 := -1.1107, .3089 :
```

```
> v := [-1.7 ..1.6, -1 ..1] :
```

```
> gridsize := [250, 250] :
```

To display the fractal as a plot, use the Julia Set as the color. The procedure *plotsegment* displays the fractal, and is called by the nodes of the grid separately for each section of the plot.

```
> plotsegment := subs(js = eval(JuliaSet), proc(viewgrid)
      plot3d(0, viewgrid[1], viewgrid[2], orientation = [
      -90, 0],
      grid = viewgrid[3], style = patchnogrid,
      scaling = constrained, color = js) :
                              end proc) :
```

Define the procedure to be run by the **Launch** command, which includes a procedure for splitting up the computation. For more information, refer to the **?Grid/Launch** help page.

```
> code :=proc( ) local msg, split;
       split := proc(view, gridsize, n)
                local i, gs, splitview, viewi, viewn;
       global vlength, vs;
         vs := op(view[1])[1] :
         vlength := abs(op(view[1])[2] - op(view[1])[1])
       / n:
         gs := floor(gridsize[1] / n) :
         splitview := [vs..vs + vlength, view[2], [gs,
       gridsize[2]]] :
          vs := vs + vlength:
         for i from 2 to n do
           if i = n then
             viewn := vs..op(view[1])[2] :
             splitview := splitview, [viewn, view[2],
       [gridsize[1] − (n − 1)·gs, gridsize[2]]] :
           else
             viewi := vs..vs + vlength:
             vs := vs + vlength:
             splitview := splitview, [viewi, view[2], [gs,
       gridsize[2]]] :
           end if:
         end do:
        splitview;
       end proc:

       msg := Util[Map][N](plotsegment, [split(v, gridsize,
       N)]) :
       msg;
   end proc:
```

Restart the Grid with your computer as the server, as in the above example. Configure the server for your network to run this example over a Grid.

```
> Server[StartServer](port, numNodes, broadcastAddress,
      broadcastPort, logFile)
```

$$3736 \tag{4.6}$$

```
> N := Status()[2]
```

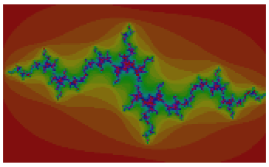
$$N := 4 \tag{4.7}$$

Now you are ready to perform the computation and display the results. Ensure that you pass all assigned variables to the **Launch** command, or they will not be available on the other nodes of the grid.

The printer is specified as *printf*, but could be any method of output, including writing to a text area component or a text file. The procedure provides a mechanism for you to stop the computation at any time, on all nodes. In this example, it is always false, so the computation cannot be stopped.

```
> picture := Launch(N, code, printf, proc() false end,
      ["plotsegment", "v", "gridsize", "JuliaSet", "c1", "c2",
     "N"]) :

> plots[display](picture);
```



With the basic tools shown here, you are now ready to use the Grid Computing Toolbox to solve many large mathematical problems. See the Maple help system for more information about the commands used in this guide, or more examples illustrating how the Grid Computing Toolbox can help you.

5 Financial Modeling Toolbox

5.1 Introduction to the Financial Modeling Toolbox

Overview

The Financial Modeling Toolbox provides a collection of tools for mathematical finance covering a wide range of common tasks, such as equity price modeling, term structure and cash flow analysis, option pricing, and stochastic processes.

Features of this toolbox include:

- The ability to create basic short rate models and binomial and trinomial trees.
- Tools for building complex processes based the standard procedures already implemented by Maple.
- Access to Maple's state-of-the-art visualization capabilities viewing statistical models of financial concepts.

Requirements

Before installing the Financial Modeling Toolbox, you must install and activate Maple 13. For details and installation instructions, see the Install.html file on the product CD.

You need a purchase code to activate the Financial Modeling Toolbox. This code has been emailed to you. If you have not received your purchase code, contact Maplesoft Customer Service at custservice@maplesoft.com (US and Canada), or the Maplesoft reseller in your region. For a list of resellers, visit **http://www.maplesoft.com/contact/international/index.aspx**.

After installation, the FinancialModeling.mw file appears on your desktop (Windows and Macintosh) or in your home directory (Linux). Open this file. Note that you can also access this file by entering **?Finance,Tour** in Maple.

The document FinancialModeling.mw contains links to the Financial Modeling Toolbox overview, sample applications, help system documentation, and more. The Financial Modeling Toolbox overview contains a summary of the tools in this product, and provides links to various examples. The examples illustrate how to use the commands in the toolbox, perform various calculations, and create graphical representations of the results.

5.2 Getting Started with the Financial Modeling Toolbox

Initialization

To start using the Financial Modeling Toolbox, open a new Maple window and load the Finance package by using the following command.

```
> with(Finance)
```

The basic categories of commands in this toolbox are:

- **Date Arithmetic**, including calendars and day counters
- **Cash Flow Analysis**
- **Financial Instruments** for American, European, and Asian markets
- **Interest Rates and Term Structures**
- **Lattice Methods**, including commands for visualization
- **Stochastic Processes**

In the following section, only some of these concepts are illustrated through examples. For more examples, see the example worksheets for this toolbox, listed below.

To access these worksheets in the Maple help system, from the **Resources** drop-down menu, select **Tutorials**. Then search for the topic Finance.

AsianOptions	Calendars	CalendarsAndDayCounters
CashFlowAnalysis	DayCounters	EuropeanOptions
LatticeMethods	LocalVolatility	ShortRateModels
StochasticProcesses	TermStructures	

Help with Financial Modeling

For help with the commands in the Financial Modeling Toolbox, see the **?Finance** help page. This toolbox is supported by commands from several Maple packages, including LinearAlgebra, Statistics, Optimization, and CurveFitting. See the help page for each package for more information.

5.3 Working with the Financial Modeling Toolbox

One of the most basic tools in the Financial Modeling Toolbox is the Calendar. You can use the existing calendars, which include the New York, London, Tokyo, and Toronto stock exchanges, or you can create your own calendar.

This is the standard calendar for the Toronto stock exchange.

```
> with(Finance) :
```

```
> IsHoliday("September 1, 2008", Toronto)
```

$$true \tag{5.1}$$

You can adjust this date in your calculations by using one of several supported business day conventions.

```
> AdjustDate("September 1, 2008", Toronto, convention
        = Following)
```

$$\text{"September 2, 2008"} \tag{5.2}$$

```
> AdjustDate("September 1, 2008", Toronto, convention
        = Preceding)
```

$$\text{"August 29, 2008"} \tag{5.3}$$

New calendars can be created from either an empty calendar or one of the existing calendars in the system. The *Simple* calendar has only Saturdays, Sundays, and January 1st as holidays.

```
> C1 := Calendar(Simple) :
```

Add the holidays for Beijing in the year 2008.

```
> NewYear := seq(AdvanceDate("February 1, 2008", i), i
      = 0 ..6) :

> SpringFest := seq(AdvanceDate("March 21, 2008", i), i
      = 0 ..6) :

> LaborDay := seq(AdvanceDate("May 1, 2008", i), i = 0
      ..6) :

> NatDay := seq(AdvanceDate("October  1, 2008", i), i = 0
      ..6) :

> AddHoliday(C1, [NewYear, SpringFest, LaborDay,
      NatDay]) :
```

Other routines from the toolbox perform date adjustments in accordance with the specified calendars.

```
> SetEvaluationDate("February 2, 2008") :

> schedule1 := Schedule("February 5, 2008",
      "February 5, 2009", Quarterly, calendar = Toronto) :

> schedule2 := Schedule("February 5, 2008",
      "February 5, 2009", Quarterly, calendar = C1) :

> libor := BenchmarkRate(1, Months, EURIBOR, 0.05) :
```

Create a simple interest rate swap, which exchanges floating-rate payments based on the first schedule with fixed-rate payments based on the second schedule.

```
> RateSwap := InterestRateSwap(100, libor, schedule1,
      0.05, schedule2, 0.) :
```

```
> pay := CashFlows(RateSwap, paying) :
  pay[1]; pay[2]; pay[3]; pay[4];
```

```
                    1.237097721 on 'May 5, 2008'

                  1.264761910 on 'August 5, 2008'

                 1.264761910 on 'November 5, 2008'

                 1.266126367 on 'February 5, 2009'                 (5.4)
```

```
> receive := CashFlows(RateSwap, receiving) :
  receive[1]; receive[2]; receive[3]; receive[4];
```

```
                    1.229508197 on 'May 8, 2008'

                  1.215846995 on 'August 5, 2008'

                 1.256830601 on 'November 5, 2008'

                 1.258140579 on 'February 5, 2009'                 (5.5)
```

Any other calculations available in this toolbox that depend on dates can use any calendar.

The following examples demonstrate some everyday uses of the Financial Modeling Toolbox, and the ways in which the commands work together.

Example

The stochastic processes available in this toolbox include mean-reverting processes, pure jump processes, jump diffusions, and multivariate Ito processes. More complicated processes can also be created, using these simple ones as a basis.

Consider a stochastic process involving multiple stocks. This is a Weiner process, which is a Gaussian process W with independent increments such that

$$W(0) = 0 \text{ with probability } 1,$$
$$E(W(t)) = 0, \text{ and}$$
$$Var(W(t)) - W(s) = t - s \text{ for all } 0 \leq s \leq t.$$

First, define the parameters of the process, as well as some other parameters that will be needed in further calculations. Σ is the covariance matrix used to generate W, μ is the drift of the process, and σ is its volatility.

> $\Sigma := \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}$:

> $\mu := \frac{55}{1000}$:
$\sigma := \frac{3}{10}$:

> $\lambda := \mu + \frac{1}{2} \cdot \sigma^2$:

Define the process and its exponential.

> $W := WienerProcess(\Sigma)$:

> $S_1 := t \rightarrow 100 \exp\left(\mu\, t + \sigma\left(W(t)_1\right)\right)$:

> $S_2 := t \rightarrow 100 \exp\left(\mu\, t + \sigma\left(W(t)_2\right)\right)$:

You can perform calculations and manipulations with this process, and display its properties, as shown here.

> $DiscountFactor(1, \lambda)$

$$0.9048374180 \tag{5.6}$$

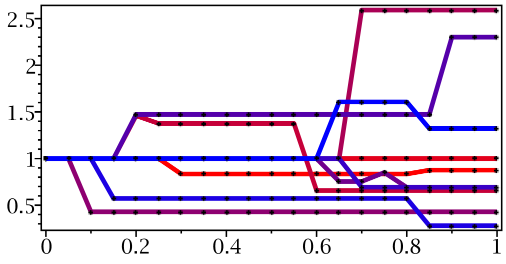
> $DiscountFactor(1, \lambda) \cdot ExpectedValue\Big(\max\big(S_2(1) - S_1(1), 0\big), replications = 10^4, timesteps = 10^2, output = value\Big)$

$$3.577999234 \tag{5.7}$$

P is a different type of process, incorporating jumps. Visualize this process by using the PathPlot command.

> $P := PoissonProcess(2, Normal(0.0, 0.5)) :$

> $PathPlot(\exp(P(t)), t = 0..1, timesteps = 20, replications = 10, \; color = red..blue, axes = BOXED, thickness = 3)$



From these processes, you can generate data values and perform statistical analyses.

```
> A := SampleValues(S_2(1) - S_1(1) exp(P(1)), timesteps
      = 100, replications = 10^5) :

> A := Statistics[SelectInRange](A,-900 ..900) :
```

> $summary := Statistics[DataSummary](A):$
$summary[1]; summary[2]; summary[3]; summary[4];$
$summary[5]; summary[6]; summary[7];$

$$mean = -65.23233217$$

$$standarddeviation = 135.9301631$$

$$skewness = -2.145336076$$

$$kurtosis = 9.824695863$$

$$minimum = -899.6010270$$

$$maximum = 285.5238707$$

$$cumulativeweight = 99215. \tag{5.8}$$

You can construct binomial, trinomial, and implied trees. Here is an example of an implied binomial tree with initial value 100, risk free rate r, dividend rate d, implied volatility ς, time 3 years, and having 21 steps.

> $r := 0.11:$
$d := 0.04:$

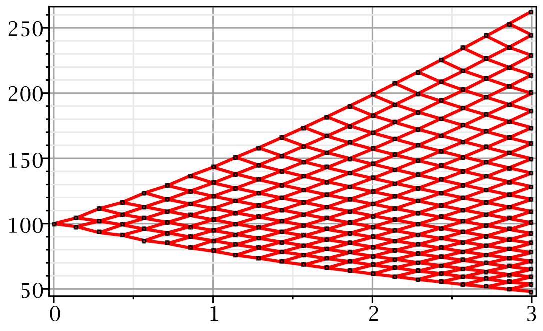
> $\varsigma := ImpliedVolatilitySurface\left(0.11 - \frac{(K-100)\cdot 0.001}{10}, t, K\right):$

> $S := t \to c\int_{1}^{U(t)} \frac{1}{x\cdot b(x)}\,\mathrm{d}x:$

> $T := ImpliedBinomialTree(100, r, d, \varsigma, 3, 21):$

Plot the binomial tree and examine its properties.

> $TreePlot(T, thickness = 2, axes = BOXED, gridlines = true, color = red)$



> $GetProbabilities(T, 1, 1)$

$$[0.5000000000, 0.5000000000] \tag{5.9}$$

> $GetProbabilities(T, 2, 1)$

$$[0.3745187925, 0.6254812075] \tag{5.10}$$

Construct another implied binomial tree and use it to price some American and European options.

> $T1 := ImpliedBinomialTree(100, r, d, \varsigma, 1, 500) :$

> $E := EuropeanOption(t \rightarrow \max(t - 100, 0), 1.0) :$

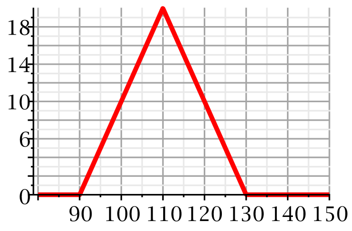
> $LatticePrice(E, T1, r)$

$$8.132615446 \tag{5.11}$$

You can price other kinds of options using this tree.

> $q := t \rightarrow \mathit{piecewise}(t < 90, 0, t < 110, t - 90, t < 130, 130 - t, 0)$:

> $\mathit{plot}(q, 80 \,..150, \mathit{gridlines}, \mathit{thickness} = 3)$



> $E := \mathit{EuropeanOption}(q, 1.0)$:

> $\mathit{LatticePrice}(E, T, r)$

$$10.02600735 \tag{5.12}$$

> $A := \mathit{AmericanOption}(q, 0, 1.0)$:

> $\mathit{LatticePrice}(A, T, r)$

$$13.68709417 \tag{5.13}$$

With the examples illustrated here, you are now ready to use the Financial Modeling Toolbox to solve many mathematical finance problems. See the Maple help system for more information about the commands used in this guide, or more ways in which the Financial Modeling Toolbox can help you.

6 MapleNet

6.1 Introduction to MapleNet

Overview

MapleNet provides a system for deploying live Maple documents over the internet. Powered by Maple, these interactive applications are accessible through a standard Web browser.

Features of this toolbox include:

- An easy-to-use system for publishing interactive Maple documents online.
- The ability to distribute live Maple applications to readers who do not have a full copy of Maple.
- Documents that are interactive, through the use of buttons, sliders, text and math input fields, and other interactive components.
- The ability to perform Maple calculations live on the Web with these applications.

Requirements

Before deploying documents with MapleNet, you must install and activate Maple 13 and MapleNet on both your Server and Publisher machine(s). The Server needs to be installed on a machine that can act as a Web server, and must also have installed Web server software, such as Tomcat, and Java SDK 1.5 or later. The Publisher can be on any machine on which you want to create and upload Maple documents to MapleNet, and must have installed file transfer software, such as FTP or SCP, for copying files to the server, and Java SDK and RE, version 1.4.2 or later.

It is recommended that you first install Maple, then the MapleNet Server, and finally, the MapleNet Publisher. Note that the Server and Publisher can be on the same machine. For installation, setup, and activation instructions, see the Install.html file on the product CD.

6.2 Creating and Publishing your Documents

You can publish any Maple document on MapleNet. It can be static, displaying math and documentation from Maple, or interactive, taking input from users to demonstrate your ideas further. Maplets™ provide pop-up dialogs for parameter input or messages, and embedded components such as buttons, math input fields, sliders, and plots can be used to input or display information. See the **?Maplets** and **?EmbeddedComponents** help pages for more information.

You can use both Maple-based Applets, called Maplets, or Java-based Applets in your documents. Note that the process for deploying each type of applets is different, as is the way in which they interact with Maple documents. Maplets within a Maple document are disabled, and must be specially implemented to be functional through MapleNet. See the Publisher's Guide, Chapters 2 and 3, for details.

MapleNet can display Maple calculations in two formats: on a JavaServer page, or as a Maple worksheet.

JavaServer pages allow you to use HTML pages to display Maple calculations:

- Enter text as you would in a regular HTML page, and enter Maple commands by using the specialized <maple> tags, described in Chapter 4 of the Publisher's Guide.
- Interact with the calculations through Maplets or Applets, as mentioned above.

Classic and Standard Maple worksheets can be displayed in a Web browser through MapleNet, but many interactive components are not available from the Classic interface:

- Any Maple worksheet can by published on MapleNet, even if the worksheet has not been specifically designed for it.
- Add embedded components such as buttons, math input containers, labels, sliders, and plot components to make a Standard worksheet interactive. You can also interact with the worksheet through Maplets or Applets.

Publishing a page to the MapleNet server requires copying the JSP or worksheet file to the appropriate directory on the server (as recommended by your site administrator). The file can be copied to the server by using any network file transfer tool, such as FTP, SSH, or WebDAV.

Once the file is copied to the server, your content will be available by accessing the appropriate URL. For example, assume the MapleNet server is *myserver.com*. If your page is *mypage.jsp* and your JSP file is in the *jsp* directory of the MapleNet Web application, then the page can be accessed by pointing your browser to http://myserver.com/maplenet/jsp/mypage.jsp. Or if your worksheet is *myworksheet.mw*, and it is in the *worksheet* directory of the MapleNet Web application, then the URL is http://myserver.com/maplenet/worksheet/myworksheet.mw.

With your Maple document creation knowledge, you are now ready to use MapleNet to publish your interactive Maple documents online. For further help with MapleNet, see the Publisher's Guide for information on deploying worksheets on an existing Server, and the Administrator's Guide for technical specifications for the Server, located in your MapleNet folder.