Ordinary Differential Equations with SCILAB

By

Gilberto E. Urroz, Ph.D., P.E.

Distributed by

infoClearinghouse.com

A "zip" file containing all of the programs in this document (and other SCILAB documents at InfoClearinghouse.com) can be downloaded at the following site:

http://www.engineering.usu.edu/cee/faculty/gurro/Software_Calculators/Scilab_Docs/ScilabBookFunctions.zip

The author's SCILAB web page can be accessed at:

http://www.engineering.usu.edu/cee/faculty/gurro/Scilab.html

Please report any errors in this document to: gurro@cc.usu.edu

Ordinary Differential Equations

The chapter starts with a review of concepts of differential equations and symbolic solution techniques that can be applied using SCILAB. Since SCILAB is not a symbolic environment, its applications to symbolic solutions of ordinary differential equations (ODEs) is limited. However, SCILAB can be used to calculate intermediate numerical steps in the solutions. The strength of SCILAB in solving ODEs is in its numerical applications. Thus, the chapter also includes a number of numerical solutions to ODEs through user-programmed and pre-programmed SCILAB functions.

Introduction to differential equations

Differential equations are equations involving derivatives of a function. Because many physical quantities are given in terms of rates of change of a certain quantity with respect to one or more independent quantities, derivatives appear frequently in the statement of physical laws. For example, the flux of heat, q [J/m^2], in a one-dimensional direction is given by

$$q = -k\cdot(dT/dx),$$

where T[K or ^{o}C] is the temperature, x [m] is positions, and k [J/(m K) or J/(m oC)]. This equation can be considered as a differential equation if q and k are known, and we are trying to solve for the temperature as a function of x, i.e., $T = T(x)$. The equation of conservation of energy for heat transfer in one-dimension, where there are no sources or sink of heat, requires that the rate of change of the heat flux across an area perpendicular to the x-axis be zero, i.e., $dq/dx = 0$,or,

$$\frac{d}{dx}\left[-k\cdot\frac{dT}{dx}\right] = 0.$$

If k is a constant, i.e., not a function of x, then, the equation of conservation of energy reduces to

$$d^2T/dx^2 = 0.$$

The last two expressions are also differential equations. The solution for these equations will be a function $T = T(x)$ representing the temperature.

Definitions

The following definitions allow us to classify equations, thus providing general guidelines for obtaining solutions.

Ordinary and partial differential equations

When the dependent variable is a function of a single independent variable, as in the cases presented above, the differential equation is said to be an *ordinary differential equation (ODE)*. If the dependent variable is a function of more than one variable, a differential equation involving derivatives of this dependent variable is said to be a *partial differential equation (PDE)*. An example of a partial differential equation would be the time-dependent would be the Laplace's equation for the stream function, $\psi(x,y,z)$, of a three-dimensional, inviscid flow:

$$\partial^2\psi/\partial x^2 + \partial^2\psi/\partial y^2 + \partial^2\psi/\partial x^2 = 0.$$

Order and degree of an equation

The *order* of a differential equation is the order of the highest-order derivative involved in the equation. Thus, the ODE

$$dy/dx + 3xy = 0$$

is a first-order equation, while Laplace's equation (shown above) is a second-order equation.

The *degree* of a differential equation is the highest power to which the highest-order derivative is raised. Therefore, the equation

$$(d^3y/dt^3)^2+(d^2y/dx^2)^5-xy = e^x,$$

is a third order, second-degree ODE, while the equation

$$\partial y/\partial t = c\cdot(\partial y/\partial x),$$

is a first-order, first-degree PDE.

Linear and non-linear equations

An equation in which the dependent variable and all its pertinent derivatives are of the first degree is referred to as a *linear differential equation*. Otherwise, the equation is said to be *non-linear*. Examples of linear differential equations are:

$$d^2x/dt^2 + \beta\cdot(dx/dt) + \omega_o\cdot x = A \sin \omega_f\, t,$$

and

$$\partial C/\partial t + u\cdot(\partial C/\partial x) = D\cdot(\partial^2 C/\partial x^2).$$

Constant or variable coefficients

The following equation:

$$d^3y/dt^3+\pi\cdot (d^2y/dx^2)^2-5\cdot y = e^x,$$

where all the coefficients accompanying the dependent variable and its derivative are constant, would be classified as a third order, linear ODE *with constant coefficients*. Instead, the equation

$$\partial^2C/\partial t^2 - u(x,t)\cdot (\partial C/\partial x) = 0,$$

would be classified as a second-order, linear PDE *with variable coefficients*.

Homogeneous and non-homogeneous equations

Typically, differential equations are arranged so that all the terms involving the dependent variable are placed on the left-hand side of the equation leaving only constant terms or terms involving the independent variable(s) only in the right-hand side. When arranged in this fashion, a differential equation that has a zero right-hand side is referred to as a *homogeneous equation*. Examples of homogeneous equations are:

$$d^2x/dt^2 + \beta\cdot(dx/dt) + \omega_o\cdot x = 0,$$

and

$$(x-1)\cdot(dy/dx) + 2\cdot x\cdot y = 0.$$

On the other hand, if the right-hand side of the equation, after placing the terms involving the dependent variable and its derivatives on the left-hand side, is non-zero, the equation is said to be *non-homogeneous*. Non-homogeneous versions of the last two equations are:

$$d^2x/dt^2 + \beta\cdot(dx/dt) + \omega_o\cdot x = A_o\cdot e^{-t/\tau},$$

and

$$(x-1)\cdot(dy/dx) + 2\cdot x\cdot y = x^2-2x.$$

Solutions

A *solution* to a differential equation is a function of the independent variable(s) that, when replaced in the equation, produces an expression that can be reduced, through algebraic manipulation, to the form 0 = 0. For example, the function

$$y = \sin x,$$

is a solution to the equation

$$d^2y/dx^2 + y = 0,$$

because when we replace y into the equation we have

$$-\sin x + \sin x = 0,$$

or, 0 = 0, for all values of x. This follows from the fact that $dy/dx = cos(x)$, and $d^2y/dx^2 =$ $-sin(x)$.

General and particular solutions

A *general solution* is one involving integration constants so that any choice of those constants represents a solution to the differential equation. For example, the function

$$x = C\cdot e^{-t},$$

is a general solution to the equation

$$dx/dt + x = 0,$$

because, substituting $C\cdot e^{-t}$ for x in the equation produces

$$- C\cdot e^{-t} + C\cdot e^{-t} = 0.$$

A *particular solution* is a solution corresponding to a specific value of the integration constants. For example, the function

$$y = x^2/2$$

is a particular solution to the equation,

$$dy/dx - x = 0.$$

A general solution for this equation would be

$$y = x^2/2 + C,$$

where C is an arbitrary integration constant.

Given the solution of the homogeneous equation, $y_h(x)$, the solution of the corresponding non-homogeneous equation, y(x), can be written as

$$y(x) = y_h(x) + y_p(x),$$

where $y_p(x)$ is a particular solution to the ODE.

Verifying solutions using SCILAB

Since SCILAB is not a symbolic environment it is not suitable for the verification of solutions other than polynomial solutions. As indicated in Chapter 8, SCILAB provides function *derivat* to calculate derivatives of polynomials. If we have a function that can be expressed as a polynomial, we can use function *derivat* to check if that function satisfies a particular differential equation as illustrated in the example below:

Check that the function $y(x) = x^3-2x+4$ is a solution to the differential equation, $d^2y/dx^2 - 6x = 0$ using SCILAB:

```
-->x = poly(0,'x'); y = x^3-2*x+4;

 -->derivat(derivat(y))-6*x
 ans  =

    0
```

Initial conditions and boundary conditions

To determine the specific value of the constant(s) of integration, we need to provide values of the solution, or of one or more of its derivatives, at specific points. These values are referred to as the *conditions* of the solution. For example, we could specify that the solution to the equation

$$d^2y/dt^2+y = 0,$$

must satisfy the conditions

$$y(0) = -5,$$

and

$$dy/dt = -1 \text{ at } t = 5.$$

Initial conditions are provided at a single value of the independent variable so that after evaluating those conditions at that point all the integration constants are uniquely specified. In general, first order differential equations include one integration constant, requiring only one condition to be evaluated to uniquely determine the solution. Thus, this type of equations needs only one initial condition. The term "initial condition" is used because many first order equations involve a derivative with respect to time, and the condition given to specify the solution is typically the value of the function at time equal to zero, i.e., an initial value of the function. *Boundary conditions*, on the other hand, are provided at more then one value of the independent variable(s). The term "boundary conditions" is used because the function is evaluated at the "boundaries" of the solution domain in order to specify the solution.

An example of *initial conditions* used in a solution will be to solve the equation

$$d^2u/dt^2 + 2\cdot(du/dt) = 0,$$

given

$$u(0) = 1,\ du/dt|_{t=0} = -1.$$

An example of boundary conditions used in a solution will be to solve the equation

$$d^2y/dx^2+y = A \sin x,$$

using

$$y(0) = A/2, \text{ and } y(1) = -A/2.$$

In general, the solution of a n-th order ODE requires n conditions.

Symbolic solutions to ordinary differential equations

By symbolic solutions we understand those solutions that can be expressed as a closed-form function of the independent variable. Because solution of first-order differential equations imply integrating the derivative involved in the equation, many of the techniques used for solving first-order ODEs follow from integration techniques. Details of some techniques used for solving ordinary differential equations follow.

Solution techniques for first-order, linear ODEs with constant coefficients

A first order equation is an equation of the form

$$a\cdot(dy/dx)^n + b\cdot y^m = f(x),$$

where a, b, n and m are, in general, real numbers. Some specific techniques for <u>linear equations</u>, i.e., when n = m = 1, follow. This catalog of solutions for linear ODEs is intended as a review of the techniques. Numerical solutions using SCILAB will be presented in a later chapter.

<u>Equations of the form: dy/dx = f(x) -- Direct integration</u>

An equation of the form dy/dx = f(x) can be re-written as

$$dy = f(x)dx,$$

and a general solution found by direct integration,

$$\int dy = \int f(x)dx,$$

or

$$y = \int f(x)dx + C.$$

If an initial condition $y(x_o) = y_o$, is given, then the integration can be calculated as

$$\int_{y_o}^{y} dy = \int_{x_o}^{x} f(x)dx,$$

or,

$$y - y_0 = \int_{x_o}^{x} f(x)dx.$$

If the function f(x) is a polynomial, we can use the SCILAB user-defined function *intpoly* to produce the indefinite integral. Consider the example in which $dy/dx = f(x) = x^3 + x + 2$. The solution, with the help of SCILAB, is calculated as:

```
-->f = poly([2,1,0,3],'x','coeff')
 f  =

             3
    2 + x + 3x

-->getf(‘intpoly’)

-->fInt = intpoly(f)

 Indefinite integral - Add integration constant

 fInt  =

           2       4
    2x +  .5x +  .75x
```

Thus, the general solution is :

$$y(x) = 2x+0.5*x^2+0.75*x^4 + C$$

Equations of the form: dy/dx = g(y) -- Inversion and direct integration

Equations of the form $dy/dx = g(y)$, can be re-written as

$$dy/g(y) = dx.$$

Thus, an indefinite integral will be given by

$$\int dy/g(y) = \int dx,$$

or

$$\int dy/g(y) = x + C.$$

From the latter expression, the dependent variable y may be solved for. A similar approach is followed when using a definite integral, i.e., one with initial condition $y(x_o) = y_o$. The integration in this case reads:

$$\int_{y_o}^{y} \frac{dy}{g(y)} = \int_{x_o}^{x} dx = x - x_o.$$

Equations of the form: dy/dx = f(x)g(y) -- Separation of variables

Equations of the form $dy/dx = f(x)g(y)$, can be separated into

$$dy/g(y) = dx/g(x),$$

and then integrated using indefinite integrals for general solutions, or definite integrals with initial conditions for particular solutions.

Equations of the form: dy/dx = g(y/x)

Using the change of variable

$$u = y/x,$$

we have

$$y = u\cdot x,$$

$$dy = u\cdot dx + x\cdot du,$$

then

$$(u\cdot dx + x\cdot du)/dx = g(u),$$

$$u\cdot dx + x\cdot du = g(u)\cdot dx,$$

$$[g(u)-u]\cdot dx = x\cdot du,$$

from which the variables x and u can be separated as

$$du/[g(u)-u] = dx/x.$$

After integration, we replace

$$u = y/x$$

back in the result, and isolate, if possible, y(x).

■ *Equations of the form: $a\cdot(dy/dx)+ b\cdot y = f(x)$ -- Integrating factors*

The expression

$$a\cdot(dy/dx)+ b\cdot y = f(x)$$

constitutes the most general form of a first-order, linear, ordinary differential equation. The equation can be re-written as

$$dy/dx + (b/a) \cdot y = (1/a)\cdot f(x),$$

You can prove that, by multiplying both sides of this form of the equation by a function,

$$IF(x) = exp(b\cdot x/a),$$

known as an *integrating factor*, the equation becomes:

$$\frac{d}{dx}\left(a\cdot \exp\left(\frac{b\cdot x}{a}\right)\cdot y(x)\right)=\frac{1}{a}\cdot \exp\left(\frac{b\cdot x}{a}\right)\cdot f(x).$$

This equation can be easily integrated to read:

$$y(x)=\exp\left(-\frac{b\cdot x}{a}\right)\cdot\left(\frac{1}{a}\cdot\int \exp\left(\frac{b\cdot x}{a}\right)\cdot f(x)+C\right)$$

In terms of the integrating factor, this solution will be:

$$y(x) = (1/IF(x))\cdot[(1/a) \cdot\int IF(x) \cdot f(x) \cdot dx + C].$$

Integrating factors for first-order, linear ODEs with variable coefficients

An equation with variable coefficients such as

$$K_1(x)(dy/dx) + K_2(x)y(x) = K_3(x),$$

can be reduced to the form,

$$dy/dx + g(x)y(x) = f(x),$$

by dividing the entire equation by $K_1(x)$. The latter equation can be solved by multiplying both sides of the equation by the integrating factor

$$IF(x) = \exp(\int g(x)dx).$$

After identifying the integrating factor, IF(x), the solution procedure is very similar to the case of a first-order, constant-coefficient ODEs, i.e.,

$$y(x) = (1/FI(x))\cdot[\int FI(x)\cdot f(x)\cdot dx + C].$$

Exact differential equations

An expression of the form,

$$F(x,y)\cdot dx + G(x,y)\cdot dy = 0,$$

Is said to be an exact differential equation in two dimensions, if the components *F(x,y)* and *G(x,y)* satisfy the conditions

$$\partial F/\partial y = \partial G/\partial x.$$

In such case, it is possible to find a function *u(x,y),* such that

$$F(x,y) = \partial u/\partial x,\ G(x,y) = \partial u/\partial y.$$

The equation*, u(x,y)* = *C*, where *C* is a constant, will represent a solution to the exact differential equation:

$$F(x,y)\cdot dx + G(x,y)\cdot dy = 0.$$

Solutions of homogeneous linear equations of any order with constant coefficients

Consider the linear, constant-coefficient, homogeneous ODE of order n:

$$d^{(n)}y/dx^{(n)} + b_{n-1}\cdot(dy^{(n-1)}/dx^{(n-1)}) + \ldots + b_2\cdot(d^2y/dx^2) + b_1\cdot(dy/dx) + b_0\cdot y = 0.$$

where the coefficients b_0, b_1, …, b_{n-1}, are constant. We can use the operator $D^{(k)} = d^{(k)}/dx^{(k)}$, to re-write the equation as

$$D^{(n)}y + b_{n-1}\cdot D^{(n-1)}y + \ldots + b_2\cdot D^2y + b_1\cdot Dy + b0\cdot y = f(x).$$

Treating the operators $D^{(k)}$, $(k = n, n-1, \ldots, 1)$, as algebraic terms, the equation is re-written as

$$[D^{(n)} + b_{n-1}\cdot D^{(n-1)} + \ldots + b_2\cdot D^2 + b_1\cdot D + b_0]\cdot y = 0.$$

The idea is that the linear combination of the operators, shown above in square brackets, is applied to the function y(x), in a similar manner as algebraic terms would be multiplied to it.

Associated with the latter expression is a polynomial known as the *characteristic equation* of the ODE, and written as

$$\lambda^n + b_{n-1}\cdot\lambda^{n-1} + \ldots + b_2\cdot\lambda^2 + b_1\cdot\lambda + b_0 = 0.$$

Suppose that the characteristic equation has n independent roots, then the general solution of the linear, constant-coefficient, homogeneous ODE of order n given earlier is

$$y = C_1 \cdot e^{\lambda_1 x} + C_2 \cdot e^{\lambda_2 x} + ... + C_{n-1} \cdot e^{\lambda_{n-1} x} + C_n \cdot e^{\lambda_n x}.$$

If out of the *n* roots there is one that has multiplicity m, then the m terms corresponding to this root λ in the solution, will be

$$C_{(1)} \cdot e^{\lambda x} + C_{(2)} \cdot x \cdot e^{\lambda x} + C_{(1)} \cdot x^2 \cdot e^{\lambda x} + ...+ C_{(1)} \cdot x^{m-1} \cdot e^{\lambda x}.$$

Example 1 – Determine the general solution to the homogeneous equation

$$d^3y/dx^3-4\cdot(d^2y/dx^2)-11\cdot(dy/dx)+30\cdot y = 0.$$

In terms of the D operator, this ODE can be written as

$$[\, D^3-4\cdot D^2-11\cdot D +30]y = 0.$$

The characteristic equation corresponding to this ODE is

$$\lambda^3-4\cdot\lambda^2-11\cdot\lambda +30 = 0.$$

To obtain solutions to this equation in SCILAB use:

```
-->lam = poly(0,'lam')
 lam  = lam

-->p = lam^3-4*lam^2-11*lam+30
 p  =
                    2     3
    30 - 11lam - 4lam + lam

-->roots(p)
 ans  =
!   2. !
! - 3. !
!   5. !
```

Thus, a general solution to the ODE under consideration is

$$y = C_1 \cdot e^{2x} + C_2 \cdot e^{-3x} + C_3 \cdot e^{5x}.$$

Example 2 – Determine the general solution to the homogeneous ODE:

$$d^4y/dx^4-7\cdot(d^3y/dx^3)+18\cdot(d^2y/dx^2)-20\cdot(dy/dx)+8\cdot y = 0.$$

In terms of the D operator, this ODE can be written as:

$$[D^4-7\cdot D^3+18\cdot D^2-20\cdot D+8]y = 0.$$

Thus, the characteristic equation is

$$\lambda^4-7\cdot \lambda^3+18\cdot \lambda^2-20\cdot \lambda +8 = 0.$$

To obtain the solution of this equation using SCILAB try the following commands:

```
-->p = lam^4-7*lam^3+18*lam^2-20*lam+8
 p  =
                     2      3     4
    8 - 20lam + 18lam - 7lam + lam

-->roots(p)
 ans  =

!   1. !
!   2. !
!   2. !
!   2. !
```

Since root $\lambda = 2$ has multiplicity of 3, the solution becomes:

$$y(x) = C_1e^x + e^{2x}(C_2 + C_3x + C_4x^2).$$

Obtaining the particular solution for a second-order, linear ODE with constant coefficients

Thus, how do we come up with a particular solution, y_p, to complete the solution to a non-homogeneous equation, $y = y_h+y_p$, given the solution to the homogeneous equation, y_h? In this section we present a general method to obtain y_p for second-order, linear ODEs with constant coefficients. The reason why we choose second-order equations is not only because they are the simpler equations to solve (not including first-order equations, which were discussed in great detail in an earlier section), but also because they are useful to model a number of real-life situations. Typical systems modeled by second-order ODEs are the damped and undamped oscillatory behavior in spring-mass and electric circuit systems.

The general expression for a second-order, linear, non-homogeneous ODE with constant coefficients is

$$d^2y/dx^2 + b_1\cdot(dy/dx) + b_0\cdot y = h(x).$$

The first step is to obtain the solution to the homogeneous equation

$$d^2y/dx^2 + b_1\cdot(dy/dx) + b_0\cdot y = 0,$$

by using the solutions to the characteristic equation

$$\lambda^2 + b_1\cdot \lambda + b_0 = 0.$$

Consider the case in which the solutions to the characteristic equation are real numbers. The solutions to this quadratic equation can be two different values of λ, say λ_1 and λ_2, in which case the homogeneous solution is written as

$$y_h(x) = C_1\cdot exp(\lambda_1\cdot x) + C_2\cdot exp(\lambda_2\cdot x),$$

or a single solution of multiplicity 2, say λ_0, in which case we write

$$y_h(x) = (C_1+C_2\cdot x)\ exp(\lambda_0\cdot x).$$

If the two solutions to the quadratic (characteristic) equation are complex numbers, they must be complex conjugates of each other as required by the fundamental theorem of algebra. In this case we can write

$$\lambda_1 = \alpha+\beta i, \text{ and } \lambda_2 = \alpha-\beta i,$$

where α and β are real numbers. Thus, the solution $C_1\cdot exp(\lambda_1\cdot x) + C_2\cdot exp(\lambda_2\cdot x)$, becomes

$$C_1\cdot e^{(\alpha+\beta i)x} + C_2\cdot e^{(\alpha-\beta i)x} = C_1\cdot e^{\alpha x}\cdot e^{i\beta x} + C_2\cdot e^{\alpha x}\cdot e^{-i\beta x} = e^{\alpha x}\cdot(C_1\cdot \cos \beta x + i\cdot C_1\cdot \sin \beta x + C_2\cdot \cos \beta x - i\cdot C_2\cdot \sin \beta x) = e^{\alpha x}\cdot[(C_1+C_2)\cdot \cos \beta x + i\cdot(C_1-C_2)\cdot \sin \beta x] = e^{\alpha x}\cdot(K_1\cdot \cos \beta x + K_2\cdot \sin \beta x),$$

where

$$K_1 = (C_1+C_2), \text{ and } K_2 = i\cdot(C_1-C_2).$$

Thus, for the case of two complex solutions to the characteristic equation, the homogeneous solution is a sinusoidal function whose amplitude grows (α>0) or decreases (α<0) with x:

$$y_h(x) = e^{\alpha x}\cdot(K_1\cdot \cos \beta x + K_2\cdot \sin \beta x).$$

If the solutions are imaginary numbers, i.e., if $\alpha = 0$ in the previous result, the homogeneous solution is a pure sinusoidal function:

$$y_h(x) = K_1\cdot \cos \beta x + K_2\cdot \sin \beta x.$$

To obtain the particular solution, $y_p(x)$, that will produce the overall solution of the non-homogeneous ODE, $y(x) = y_h(x) + y_p(x)$, follow this rule that refers to the sub-sequent table of functions:

- If $h(x)$, in the general non-homogeneous ODE, is given by one of the functions in the first column of the table shown below, choose for $y_p(x)$ a linear combination of $h(x)$ and its linearly independent derivatives, as shown in the second column of the table.

- If $h(x)$ is the sum of some of the functions shown in column 1 of the table below, choose for $y_p(x)$ the sum of the functions in the corresponding lines.

- If a term in $h(x)$ is a solution of the homogeneous equation corresponding to the ODE under consideration, modify your choice of $y_p(x)$ by multiplying the appropriate line of column 2 by x or x^2, depending on whether the root of the characteristic equation (column 3) is simple or double.

Term in h(x)	***Choice for $y_p(x)$***	***Root of char. eqn.***
$c\cdot e^{\alpha x}$	$C_0\cdot e^{\alpha x}$	α, real
$c\cdot x^n$ (n = 0, 1, …)	$C_n\cdot x^n + C_{n-1}\cdot x^{n-1} + \ldots + C_1\cdot x + C_0$	0
$c\cdot \sin \beta x$	$C_1\cdot \sin \beta x + C_2\cdot \sin \beta x$	$i\beta$, imaginary
$c\cdot \cos \beta x$	$C_1\cdot \sin \beta x + C_2\cdot \sin \beta x$	$i\beta$, imaginary

Once the particular solution is set up by following the rule above, the undetermined coefficients in $y_p(x)$ can be determined by substituting yp(x) into the ODE.

<u>*Example 1*</u> – Obtain the general solution to the non-homogeneous, second-order, linear ODE:

$$d^2y/dx^2 - 5\cdot(dy/dx) +6\cdot y = x^2.$$

The characteristic equation of the homogeneous equation is

$$\lambda^2-5\cdot\lambda +6 = 0,$$

or

$$(\lambda-3)\cdot(\lambda-2) = 0,$$

with solutions

$$\lambda = 2, \text{ and } \lambda = 3.$$

Thus, the homogeneous solution is

$$y_h(x) = K_1\cdot e^{2x} + K_2\cdot e^{3x}.$$

Since the right-hand side of the non-homogeneous equation is

$$h(x) = x^2,$$

from the table above we select

$$y_p(x) = C_2x^2+C_1x+C_0.$$

To obtain the values of C_0, C_1, and C_2, replace the solution $y_p(x)$ into the ODE. The derivatives are, $dy_p/dx = 2C_2x+C_1$, and $d^2y_p/dx^2 = 2C_2$, which replaced into the equation produce

$$2C_2 -5(2C_2x+C_1) + 6(C_2x^2+C_1x+C_0) = x^2,$$

or

$$6C_2x^2 + (6C_1-10C_2)x + (6C_0-5C_1+2C_2) = x^2.$$

Comparing the coefficients of the terms x^2, x^1, and x^0, in both sides of the resulting equation allows us to write the following system of linear equations:

$$\begin{aligned} 6C_2 &= 1 \\ 6C_1 - 10C_2 &= 0 \\ 6C_0-5C_1+\ \ 2C_2 &= 0 \end{aligned}$$

A solution, using SCILAB, produces:

```
-->A = [0,0,6;0,6,-10;6,-5,2], b = [1;0;0]
 A  =

!   0.    0.    6.  !
!   0.    6.  - 10. !
!   6.  - 5.    2.  !



 b  =

!   1. !
!   0. !
!   0. !

-->C = A\b
 C  =

!    .1759259 !
!    .2777778 !
!    .1666667 !
```

The solution is:

$$C_0 = 0.1759259,\ C_1 = 0.2777778,\ \text{and}\ C_2 = 0.1666667$$

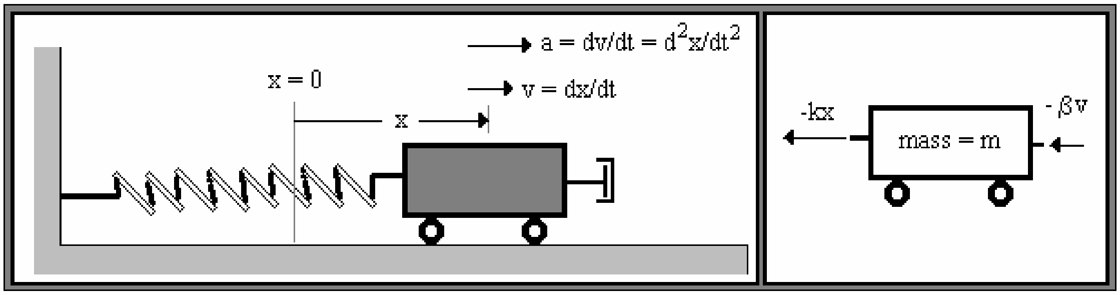
Thus, $y_p(x) = 0.1666667x^2+0.2777778x+0.1759259,$

and the general equation to the non-homogeneous equation becomes:

$$y(x) = y_h(x)+y_p(x) = K_1\cdot e^{2x} + K_2\cdot e^{3x} + 0.1666667x^2+0.2777778x+0.1759259.$$

Applications of ODEs I : analysis of damped and undamped free oscillations

Consider the mass-spring system shown in the figure below. The mass is removed from its equilibrium position (x = 0) and released at a position $x = x_0$ at $t=0$. At the moment of its release the body was moving with a speed $v = v_o$. The diagram shows the body of mass m being acted upon by the restoring force of the spring, $Fs = - k\cdot x$, and by a viscous damping force, $Fv = - \beta\cdot v = - \beta \cdot (dx/dt)$.



Newton's second law, when applied in the x-direction to the mass m is written as:

$$-kx - \beta\,(dx/dt) = m\,(d^2x/dt^2),$$

which results in the second-order, linear, ordinary differential equation:

$$d^2x/dt^2 + (\beta/m)\cdot(dx/dt)+(k/m)\cdot x = 0.$$

Undamped motion

Let us first consider the case in which the motion is undamped, i.e., b = 0. The equation in this case reduces to

$$d^2x/dt^2 +(k/m)\cdot x = 0.$$

The corresponding characteristic equation is

$$\lambda^2 + (k/m) = 0,$$

with solutions,

$$\lambda = \pm\, i\cdot\sqrt{(k/m)} = \pm i\cdot\omega_o.$$

This result suggest a solution of the form

$$x(t) = C_1 \cos \omega_o t + C_2 \sin\ \omega_o t.$$

Alternatively, by taking

$$C_1 = A \cos \phi, \text{ and } C_2 = - A \sin \phi,$$

the solution can be written as

$$x(t) = A\cdot\cos(\omega_o t + \phi).$$

The quantity

$$\omega_o = \sqrt{}(k/m)$$

is known as the *natural angular frequency* of the harmonic motion that results when no viscous damping is present. The frequency of the oscillation can be calculated from

$$f = 2\pi/\omega_o = 1/T,$$

where T is the period of the oscillation (i.e., the time that the mass takes to return to a pre-defined position in the motion). The quantity ϕ is known as the angular phase of the oscillation, and A is known as the amplitude.

The velocity of the motion is given by

$$v = dx/dt = -\omega_o\cdot A \sin(\omega_o t + \phi),$$

and its acceleration, is

$$a = dv/dt = -\omega_o^2\cdot A \cos(\omega_o t + \phi),$$

The initial conditions, $x(0) = x_o,\ v(0) = v_o$, can be used to evaluate the constants A and f, as follows:

$$x_o = x(0) = A \cos \phi\ ,$$

and

$$v_o = v(0) = -\omega_o\cdot A \sin \phi.$$

Thus,

$$\tan \phi = - v_o/(\omega_o\, x_o), \text{ or } \phi = \tan^{-1}(-v_o/(\omega_o\, x_o)),$$

and

$$A = [x_o^2 + (v_o/\omega_o)^2]^{1/2}.$$

Damped motion

If damping occurs ($\beta\neq0$), the characteristic equation becomes

$$\lambda^2 + \ (\beta/m)\cdot\lambda + \omega_o^2 = 0,$$

whose solutions are

$$\lambda = -(\beta/(2\cdot m))\pm \sqrt{}\,([\beta/(2\cdot m)]^2-\omega_o^2) = -\ \alpha \pm \sqrt{}\,(\alpha^2-\omega_o^2),$$

where

$$\alpha = \beta/(2\cdot m).$$

The nature of the solution will depend on the relative size of the coefficients α and ω_o, as follows:

- If $\alpha < \omega_o$, then $\sqrt{}\,(\alpha^2-\omega_o^2) = i\cdot\omega_I$, where

$$\omega_I = \sqrt{}(\omega_o^2-\alpha^2)$$

is real, and the solutions of the characteristic equation are

$$\lambda_1 = -\alpha + i\cdot\omega_I, \text{ and } \lambda_2 = -\alpha - i\cdot\omega_I.$$

The solution to the ODE, therefore, is written as

$$x(t) = e^{-\alpha t} (C_1 \cos \omega_I t + C_2\cdot \sin \omega_I t) = A_o\cdot e^{-\alpha t}\cdot \cos(\omega_I t + \phi_1).$$

The parameter

$$\omega_I = \sqrt{}(\omega_o^2-\alpha^2) = \sqrt{}[(k/m)^2-(\beta/(2m))^2] = \sqrt{}(4k^2-\beta^2)/(2m),$$

represents the damped angular frequency of the oscillation, and ϕ_1 represents the corresponding angular phase. A_o is the amplitude of the oscillation at t = 0. If we define a variable amplitude,

$$A(t) = A_o\cdot e^{-\alpha t},$$

then the solution to the ODE, also known as the signal, can be written as

$$x(t) = A(t)\cdot \cos(\omega_I t + \phi_1).$$

Please notice that this solution is very similar to the case of an undamped oscillation, except for the fact that in a damped oscillation the amplitude decreases with time. The amplitude decreases, or decays, with time because the parameter $\alpha = \beta/(2m)$ is positive. Therefore, the function *exp(-αt)* decreases with time.

- If $\alpha = \omega_o$, then the characteristic equation produces the solution $\lambda = -\alpha$, with multiplicity 2, in which case the solution becomes

$$x(t) = e^{-\alpha t} (C_1 + C_2\cdot t).$$

This solution represents a linear function of t subjected to a decay factor, *exp(-αt).*

- If $\alpha > \omega_o$, then $\sqrt{}\ (\alpha^2-\omega_o^2) = K$ is real, and $K < \alpha,$ the solutions of the characteristic equation become

$$\lambda_1 = -\alpha + K = -c_1, \text{ and } \lambda_2 = -\alpha - K = -\ c_2,$$

both negative. Therefore, the resulting signal can be written as:

$$x(t) = C_1\cdot exp(-c_1 t) + C_2\cdot exp(-c_2 t).$$

Notice that the last two cases, namely, $\alpha = \omega_o$ and $\alpha > \omega_o$, produce signals that decay with time. These cases correspond to harmonic motions that are said to be *over-damped*, i.e., the viscous damping is large enough to quickly damp out any oscillation after the body of mass m is released.

Initial conditions for damped oscillatory motion

The expression for the position of a damped oscillatory motion is given by

$$x(t) = A_o\cdot e^{-\alpha t}\cdot \cos(\omega_I t + \phi_1),$$

while the velocity, *v(t) = dx/dt*, is given by

$$v(t) = -A_0\ e^{-\alpha t}(\ \alpha \cos(\omega_I t+\phi_1) + \omega_I \sin(\omega_I t+\phi_1)).$$

Given the initial conditions $x(t_0) = x_0$ and $v(t_0) = v_0$, we can form a system of two non-linear equations in the unknowns A_0 and ϕ_1, namely,

$$f_1(A_0,\phi_1) \;=\; A_o\cdot e^{-\alpha t}{}_0\cdot\cos(\omega_I t_0 + \phi_1) - x_0,$$

$$f_2(A_0,\phi_1) \;=\; -A_0\, e^{-\alpha t}{}_0(\,\alpha\cos(\omega_I t_0+\phi_1) + \omega_I \sin(\omega_I t_0+\phi_1)) - v_0.$$

With appropriate values of the parameters α, ω_I, t_0, x_0, and v_0, we can use SCILAB function *fsolve* to obtain the values of A_0 and ϕ_1, as illustrated in an upcoming example.

Before presenting the example, however, we will write out the expression for the acceleration so that we can use it in producing the graphics of the example:

$$a(t) = A_0\, e^{-\alpha t}{}_0(\,\alpha^2\cos(\omega_I t+\phi_1) + 2\alpha\,\omega_I \sin(\omega_I t+\phi_1) - \omega_I^2\cos(\omega_I t+\phi_1)).$$

Example 1 – Damped oscillatory motion: Plot position, velocity, and acceleration corresponding to the following parameters: $m = 1\ kg$, $\beta = 0.1 N\cdot s/m$, $k = 0.5\ N/m$. To determine the constants A_o and ϕ_1, use initial conditions, $x_0 = 1.5\ m$, and $v_0 = -5.0\ m/s$. With these values,

$$\omega_o = (k/m)^{1/2} = (0.5N/1kg\cdot m)^{1/2} = (0.5\ s^{-2})^{\,1/2} = 0.7071\ s^{-1} = 0.7071\ rad/s,$$

and

$$\alpha = \beta/(2m) = 0.1\ N\cdot s/\ (2\times 1\ kg\cdot m) = 0.05\ s^{-1} = 0.05\ rad/s.$$

Since, $\alpha < \omega_o$, the resulting signal is that of a damped oscillation with

$$\omega_I = \surd(\omega_o^2-\alpha^2) = \surd(0.7071^2 - 0.05^2) = 0.7053\ rad\ /s.$$

To solve for the constants A_o and ϕ_1 with SCILAB, we first define the set of non-linear equations to be solved. In the function thus defined A_0 is represented by *s(1)* and ϕ_1 is represented by *s(2)*:

```
-->deff('[FF]=f(s)',['f1=s(1)*exp(-a*t0)*cos(wI*t0+s(2))-x0';...
-->'f2 = -s(1)*exp(-a*t0)*(a*cos(wI*t0+s(2))+wI*sin(wI*t0+s(2)))-v0';...
-->'FF = [f1;f2]'])
```

Next, we enter the known values, and select a first guess for the solution *s0*:

```
-->w0 = 0.7071; a = 0.05; wI = 0.7053; x0 = 1.5; v0 = -5.0; t0 = 0;

-->s0 = [5;%pi/3]
 s0  =

!   5.        !
!   1.0471976 !
```

The solution for $s(1) = A_0$ and $s(2) = \phi_1$ is obtained by using:

```
-->fsolve(s0,f)
 ans  =
!   7.1421364 !
!   1.3591997 !
```

Thus, $A_0 = 7.1221364$ and $\phi_1 = 1.3591997$, and the position x(t) is given by

$$x(t) = A_o \, exp(-0.05t) \, cos(0.7053t - \phi_1).$$

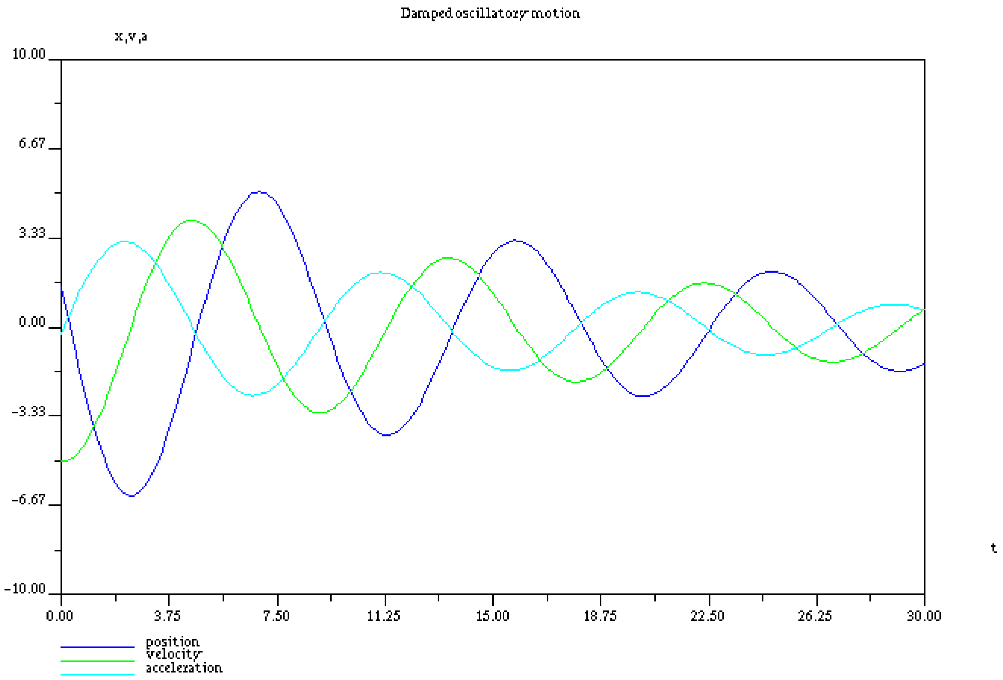
Expressions for the position x(t), velocity v(t), and acceleration a(t) for this motion can be entered into SCILAB by defining the following functions:

```
-->A0 = 7.1421364; phi1 = 1.3591997;

-->deff('[xs]=x(t)','xs = A0.*exp(-a.*t).*cos(wI.*t+phi1)')

-->deff('[vs]=v(t)',...
-->'vs =-A0.*exp(-a.*t).*(a.*cos(wI.*t+phi1)+wI.*sin(wI.*t+phi1))')

-->deff('[acc]=aa(t)',...
-->'acc=A0.*exp(-a.*t).*(a^2.*cos(wI.*t+phi1)+...
-->2.*a.*wI.*sin(wI.*t+phi1)-wI^2.*cos(wI.*t+phi1))')
```

To plot the signals x(t), v(t), and a(t) in the t-interval (0,30) use the following SCILAB commands:

```
-->tt = [0:0.1:30]; xx = x(tt); vv = v(tt); -->aaa = aa(tt);

-->plot2d([tt',tt',tt'],[xx',vv',aaa'],[2,3,4],'111',...
-->'position@velocity@acceleration',[0 -10 30 10])

-->xtitle('Damped oscillatory motion', 't', 'x,v,a')
```

The results are shown in the following graph:



Notice the oscillatory nature of the three functions, as well as their amplitudes' decay with time as expected.

Creating phase portraits of oscillatory motion

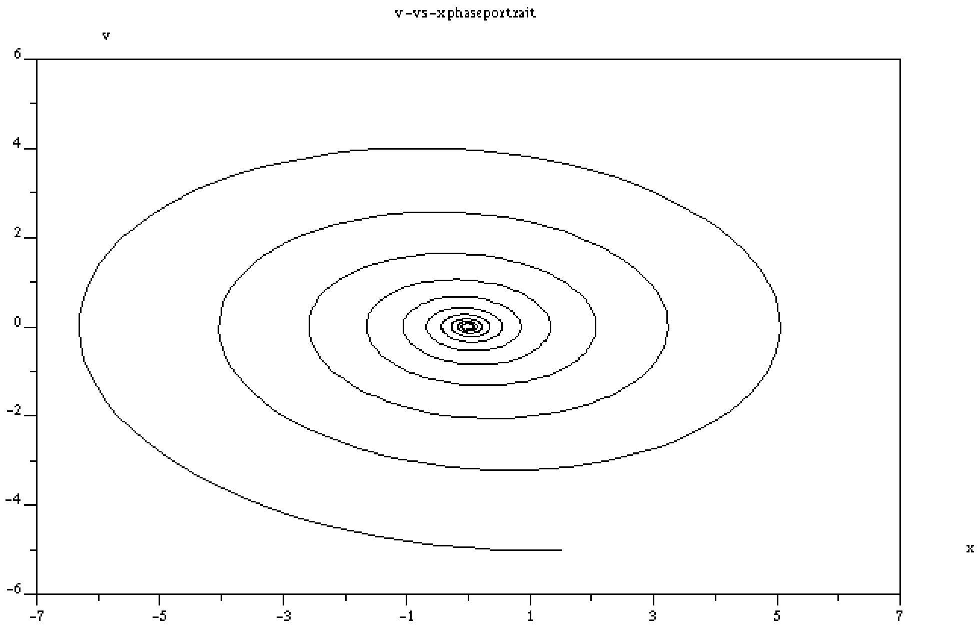
A phase portrait for oscillatory, or any kind of, motion is a plot involving the dependent variable and one of its derivatives, or two derivatives of the dependent variable. For example, a plot of velocity, v(t), versus position, x(t), represents a phase portrait. Other phase portraits would be *a(t)* vs. *x(t),* and *a(t)* vs. *v(t)*.

Example 1. Plot the time-dependent plots and phase portraits for the signal obtained in Example 1 in the previous section. To plot these phase portraits we generate data on position, velocity, and acceleration as function of time t in the interval [0,90], as follows:

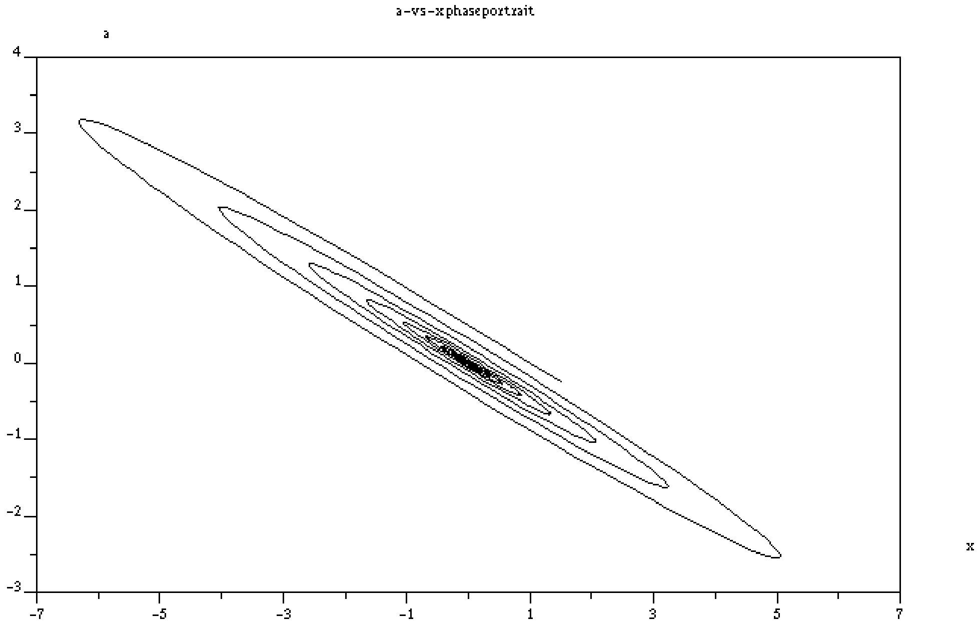
```
-->tt = [0:0.1:90]; xx = x(tt); vv = v(tt); aaa = aa(tt);
```

The phase portraits are generated as follows:

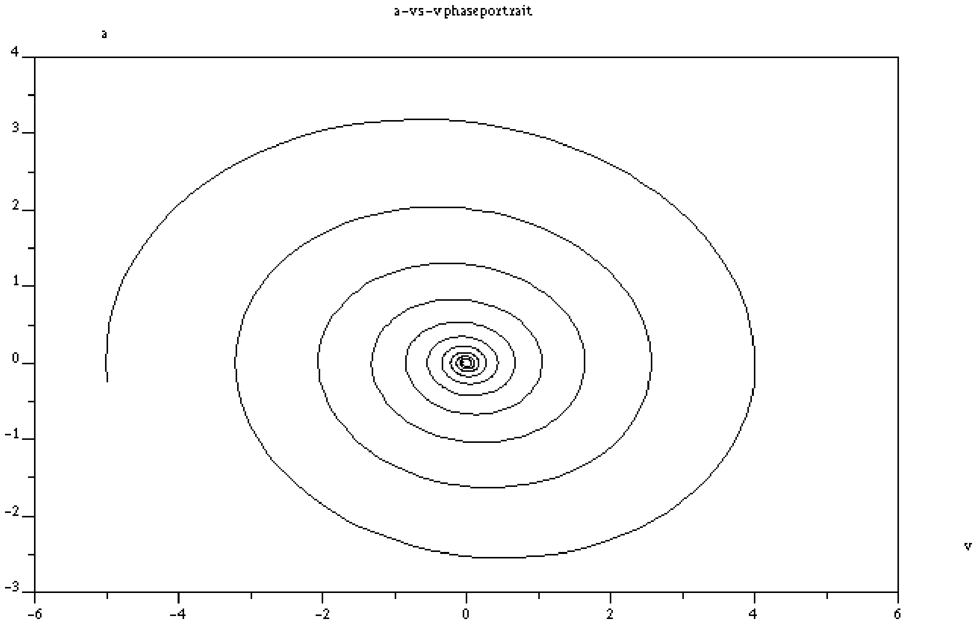
```
-->xset('window',1);plot(xx',vv')
-->xtitle('v-vs-x phase portrait','x','v')
```



```
-->xset('window',2);plot(xx',aaa')
-->xtitle('a-vs-x phase portrait','x','a')
```



```
-->xset('window',3);plot(vv',aaa')

-->xtitle('a-vs-v phase portrait','v','a')
```



The three phase portraits show orbits spiraling inwards towards the center of the picture, i.e., towards (0,0). This is because the amplitude of the variables included in the phase portrait decreases at about the same rate with time.

Applications of ODEs II : analysis of damped and undamped forced oscillations

Earlier we presented the analysis of damped and undamped free oscillations, meaning that, once the particle subjected to oscillatory motion is released, all forces acting on it (the restoring force of the spring, and the damping force from the dashpot) are internal to the system. If the particle is continuously subjected to an external force (an excitation), then the type of oscillations thus generated are termed *forced oscillations*. Of interest are excitations that are themselves oscillatory. The simplest case will be an external force,

$$F_e(t) = F_o \cos \omega t.$$

The differential equation for the mass-spring-dashpot system, including the excitation, $F_e(t)$, is now written as:

$$d^2x/dt^2 + (\beta/m)\cdot(dx/dt)+(k/m)\cdot x = (F_o/m)\cdot\cos \omega\cdot t.$$

Let's assume that the values of the parameters m, b, and k are such that the solution of the homogeneous equation is

$$x_h(t) = A_o\cdot e^{-at}\cdot\cos(\omega_o\cdot t+\phi).$$

Also, because the term $\cos \omega t$ shows up in the right-hand side term, the table for selecting the particular solution (shown earlier in this chapter), suggest that we try

$$x_p(t) = C_1 \cos \omega t + C_2 \sin \omega t.$$

Because this particular solution must satisfy the governing ODE, we can write

$$d^2x_p/dt^2 + (\beta/m)\cdot(dx_p/dt)+(k/m)\cdot x_p = (F_o/m)\cdot\cos\,\omega\cdot t.$$

The values of C_1 and C_2, using $\omega_0^2 = k/m$, are:

$$C_1 = \frac{F_0 m(\omega_0^2-\omega^2)}{\omega^2\beta^2+m^2(\omega_0^2-\omega^2)^2}.$$

$$C_2 = \frac{F_0\omega\beta}{\omega^2\beta^2+m^2(\omega_0^2-\omega^2)^2}.$$

The particular solution can be written now as

$$x_p(t) = F_0\cdot\frac{m(\omega_0^2-\omega^2)\cdot\cos(\omega\cdot t)+\omega\beta\cdot\sin(\omega\cdot t)}{\omega^2\beta^2+m^2(\omega_0^2-\omega^2)^2}.$$

Suppose that we want to write this solution as

$$x_p(t) = A_p\cos(\omega t+\phi_p) = A_p\cos\omega t\cos\phi_p - A_p\sin\omega t\sin\phi_p,$$

by comparing the last two expressions we find that

$$A_p\cos\phi_p = F_0m(\omega_o^2-\omega^2)/[\,\omega^2\beta^2+m^2\,(\omega_o^2-\omega^2)^2],$$

and

$$A_p\sin\phi_p = -\,F_0\omega\beta/[\,\omega^2\beta^2+m^2\,(\omega_o^2-\omega^2)^2],$$

from which,

$$A_p^2 = F_0^2/[\,\omega^2\beta^2+m^2\,(\omega_o^2-\omega^2)^2],$$

and

$$\tan\phi_p = -\,\omega\beta/(m(\omega_o^2-\omega^2)).$$

Thus, the particular solution can be written as:

$$x_p(t) = \frac{F_0}{\sqrt{\omega^2\beta^2+m^2(\omega_0^2-\omega^2)^2}}\cdot\cos(\omega\cdot t+\phi_p).$$

To analyze the behavior of this particular solution, first we study the case in which no damping is present, i.e., $\beta = 0$. In such case, $\phi_p = 0$, and the particular solution becomes

$$x_p(t) = \frac{F_0}{m(\omega_0^2-\omega^2)}\cdot\cos\omega\cdot t = \frac{F_0/(m\cdot\omega_0)}{1-(\omega/\omega_0)^2}\cdot\cos\omega\cdot t = A_p(\omega)\cdot\cos\omega\cdot t.$$

For this case, the amplitude of the oscillation, $A_p(\omega)$, becomes infinity as $\omega \rightarrow \omega_o$. This condition is known as *resonance*. Thus resonant conditions will occur if the exciting force has the same frequency as the natural frequency of the system. In practice, the amplitude of the undamped oscillations grows without bound until the system is severely damaged or destroyed. This is important for analyzing building response to earthquakes. Every building has a natural frequency of vibration. If a building is subjected for a long period of time to an earthquake with a frequency similar or equal to its natural frequency, the building may suffer severe damages as consequence of the earthquake.

If damping is present, then the amplitude of the oscillation is given by

$$A_p(\omega) = \frac{F_0}{\sqrt{\omega^2\beta^2 + m^2(\omega_0^2 - \omega^2)^2}},.$$

which has a maximum

$$A_p(\omega) = \frac{2mF_0}{\beta\sqrt{4m^2\omega^2 - \beta^2}},$$

when

$$\beta^2 = 2m^2\ (\omega_o^2 - \omega^2).$$

Since the general solution of the damped equation,

$$x_h(t) = A_o \cdot e^{-at} \cdot \cos(\omega_o \cdot t + \phi),$$

decreases with time, it will eventually become negligible when compared to the particular solution. Thus, it is said that the general solution represents the *transient (temporary) response* of the system to the exciting force, $F_e(t)$. The particular solution, which turns out to be a sinusoidal wave, represents *the steady-state response* of the system.

The following is a graph showing the amplitude, $A_p(\omega)$, as function of the angular frequency, w, for a particular set of values of the parameters, namely, F_0 = 25 N, k = 100 N/m, m = 1.0 kg, which gives w_0 = 3.162277 rad/s. The graph is obtained using SCILAB as shown below. First, we define the function for $A_p(\omega)$, and the constant parameters:

```
-->deff('[AA]=Ap(om)','AA=F0./sqrt(om.^2.*b.^2+m.^2.*(w0.^2-om.^2).^2)')

-->F0 = 25; k = 10; m = 1.0; w0 = sqrt(k/m)
 w0  = 3.1622777
```

Next, we define a vector *bb* containing 5 different values of the damping parameter β, and a vector *om* containing values of ω in the range (0,6). The lengths of vectors *bb* and *om* are the values *n* and *m*, respectively:

```
-->bb = [1.0, 5.0, 10.0, 50.0, 100.0]; om = [0:0.1:6];

-->n = length(bb); m = length(om);
```

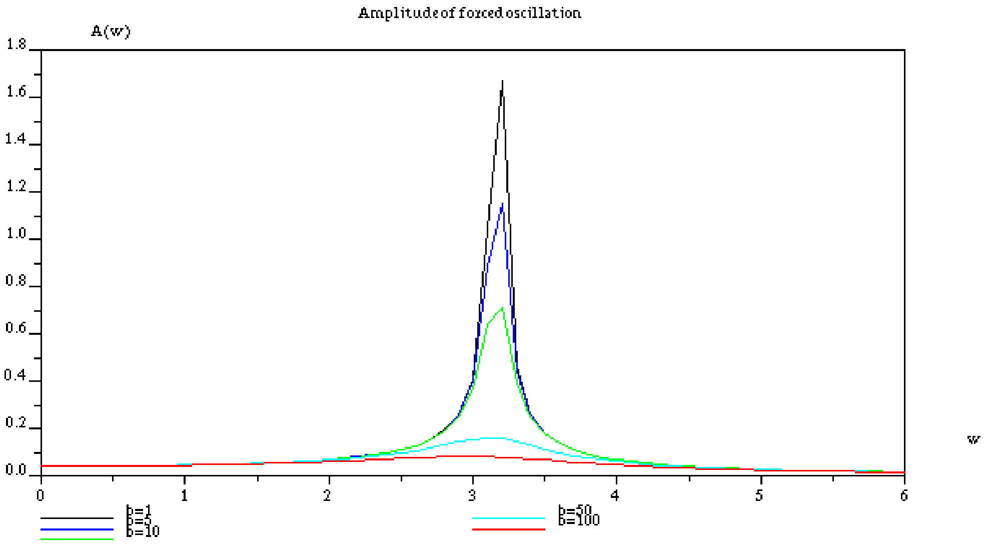
The next step is to create a matrix *Amp* (*Amp*litudes) with *m* rows and *n* columns whose elements will contain the values $Amp_{ij} = Ap(om_i,bb_j)$. The following commands show how to load matrix *Amp*:

```
->Amp = zeros(m,n);

-->for j = 1:n
```

```
-->      b = bb(j); Amp(:,j) = Ap(om');
-->end;
```

To produce the plot showing curves of $A_p(\omega)$ for different values of β is produced by using:

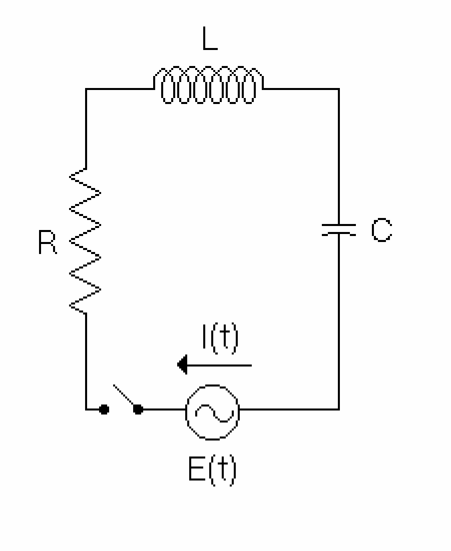
```
-->plot2d([om',om',om',om',om'],...
-->[Amp(:,1),Amp(:,2),Amp(:,3),Amp(:,4),Amp(:,5)],...
-->[1:1:5],'111','b=1@b=5@b=10@b=50@b=100',[0 0 6 1.8])

-->xtitle('Amplitude of forced oscillation','w','A(w)')
```



The plot shows that as the value of β decreases the amplitude reaches a maximum near the value of $\omega = \omega_0$. If there were no damping, i.e., $\beta \to 0$, then $A_p(\omega) \to \infty$ at that point, indicating the condition of resonance.

Applications of ODEs III: Oscillations in electric circuits

Electric circuits involving resistors, capacitors, and inductors are often times characterized by an oscillatory behavior represented by electric current I through the circuit. Consider a simple series RLC circuit as shown in the figure below.



In the figure, *E(t)* stands for the time-dependent voltage (volts), *I(t)* is the electric current through the circuit once the switch is set to ON (amperes), *R* is the equivalent resistance of the circuit (ohms), *L* is the equivalent inductance of the circuit (henrys), and *C* is the equivalent capacitance (farads). The electric current, *I(t),* is the rate of change of electric charge with respect to time, i.e., $I(t) = dq/dt$, where q = electric charge (coulombs), and t is time (sec). The properties of resistors, capacitors, and inductors are such that if V represents the voltage across one of those components the following relationships hold:

▨ Resistors,	$V_R = R\cdot I = R\cdot(dq/dt)$
▨ Capacitors,	$V_C = q/C$
▨ Inductors,	$V_L = L\cdot(dI/dt) = L\cdot(d^2q/dt^2)$

To put together a differential equation for this circuit we use Kirchoff's law of voltages around the series circuit: $V_R + V_L + V_C = E(t)$, i.e.,

$$R\cdot(dq/dt) + L\cdot(d^2q/dt^2) + q/C = E(t).$$

Alternatively, we can take the derivative of this equation with respect to t and write the equation in terms of the current, $I = dq/dt$, as follows:

$$R\cdot(dI/dt) \ + L\cdot(d^2I/dt^2) + (1/C)\cdot I = dE/dt.$$

Thus, we can either solve the equation in terms of the electric charge, *q(t)*, re-written as:

$$(d^2q/dt^2) + (R/L)\cdot(dq/dt) + q/(LC) = E(t)/L,$$

or, in terms of the electric current, I(t), re-written as:

$$(d^2I/dt^2) + (R/L)\cdot(dI/dt) + I/(LC) = (1/L)(dE/dt).$$

To simplify the solution we introduce the constants $\omega_o^2 = 1/(LC)$ and $\beta = R/(2L),$ and solve the equation in terms of the electric charge, *q(t),* i.e.,

$$(d^2q/dt^2) + 2\cdot\beta\cdot(dq/dt) + \omega_o^2\cdot q = E(t)/L.$$

Solution to the homogeneous equation

Consider first, the homogeneous case, i.e., *E(t) = 0*, which is the situation that would occur is the capacitor is charged and the voltage source is by-passed. The resulting governing equation is

$$(d^2q/dt^2) + 2\cdot\beta\cdot(dq/dt) + \omega_o{}^2\cdot q = 0,$$

which is the same as the case of free oscillations with damping for a spring-mass-dashpot system.

If the constant value

$$\varpi_1 = \sqrt{\varpi_0^2 - \beta^2}$$

is real, the solution is a damped oscillation, i.e.,

$$q(t) = Q_0 e^{-\beta t}\cos(\varpi_1 t + \phi).$$

If ω_1 is not real, then the solution is a combination of exponential functions, i.e.,

$$q(t) = C_1 e^{-\alpha_1 t} + C_2 e^{-\alpha_2 t}$$

Since the governing equation for *q(t)* in an RLC circuit is the same as the governing equation for the position of a particle in a mass-spring-dashpot system, we can borrow many of the results obtained in the previous two sections to analyze RLC circuits. Some applications are presented in the exercise section.

Finite differences and numerical solutions

To solve differential equations numerically we can replace the derivatives in the equation with finite difference approximations on a discretized domain. This results in a number of algebraic equations that can be solved one at a time (explicit methods) or simultaneously (implicit methods) to obtain values of the dependent function y_i corresponding to values of the independent function x_i in the discretized domain.

Finite differences

A finite difference is a technique by which derivatives of functions are approximated by differences in the values of the function between a given value of the independent variable, say x_0, and a small increment (x_0+h). For example, from the definition of derivative,

$$df/dx = \lim_{h\to 0} (f(x+h)-f(x))/h,$$

we can approximate the value of *df/dx* by using the finite difference approximation

$$(f(x+h)-f(x))/h$$

with a small value of *h*.

The following table shows approximations to the derivative of the function

$$f(x) = \exp(-x)\sin(x^2/2),$$

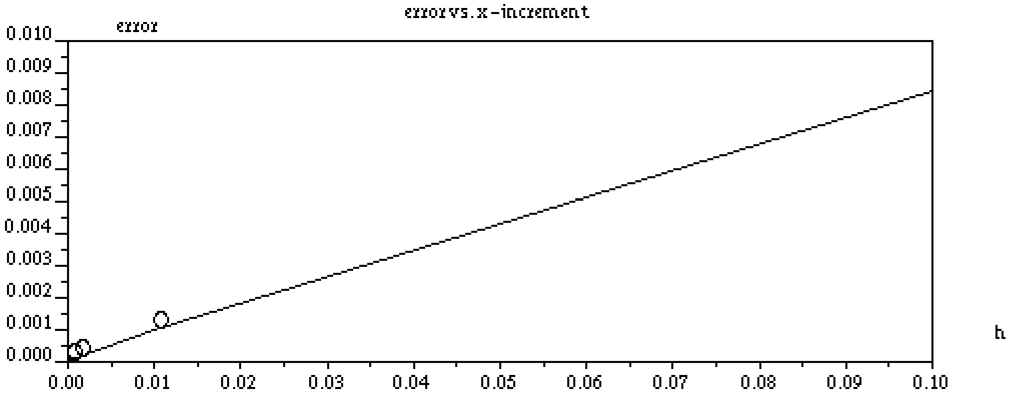
at *x* = *2*, using finite differences. The actual value of the derivative is -0.23569874791. The third column in the table shows the error in evaluating the derivative, i.e., the difference between the numerical derivative Δf/Δx and the actual value.

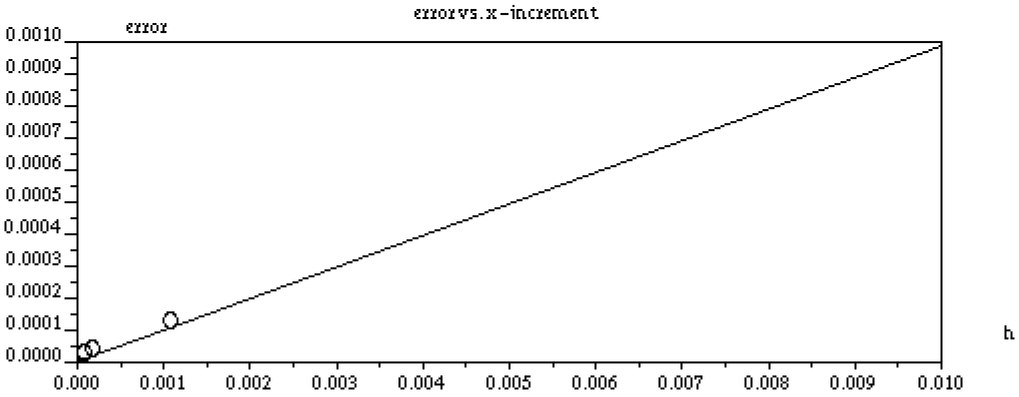
h	Δf/Δx	error
0.1	-0.244160077	0.00846132909
0.01	-0.236684829	0.00098608109
0.001	-0.235798686	0.00009993809
0.0001	-0.235708734	0.00000998609
0.00001	-0.235699726	0.00000097809
0.000001	-0.235698825	0.00000007709
0.0000001	-0.235698734	0.00000001391
0.00000001	-0.235698724	0.00000002391
0.000000001	-0.235698752	0.00000000409

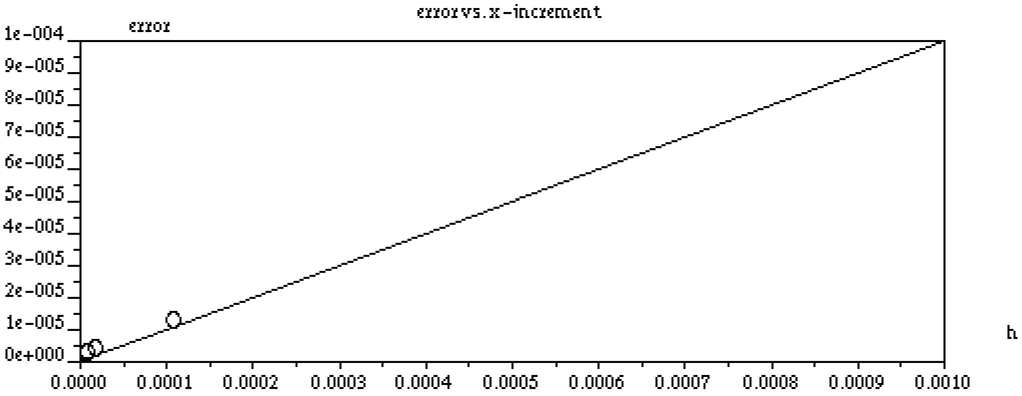
This exercise illustrates the fact that, as *h→0*, the value of the finite difference approximation, *(f(x+h)-f(x))/h*, approaches that of the derivative, *df/dx*, at the point of interest.

A plot of the error as a function of h also reveals the fact that the error is proportional to the value of the x-increment h. The following plots, using different ranges of h, are produced with SCILAB out of the data in the table.

```
-->h = [1e-1,1e-2,1e-3,1e-4,1e-5,1e-6,1e-7,1e-8,1e-9];

-->er = [0.00846132909,0.00098608109,0.00009993809,0.00000998609,...
-->      0.00000097809,0.00000007709,0.00000001391,0.00000002391,...
-->      0.00000000409];

-->xset('mark',-9,2)
-->plot2d(h,er,1,'011',' ',[0 0 0.1 0.01])
-->plot2d(h,er,-9,'011',' ',[0 0 0.1 0.01])
-->xtitle('error vs. x-increment','h','error')
```







The graphs seem indicate that the error varies linearly with the increment h in the independent variable. It is very common to indicate this dependency by saying that "the error is of order h", or *error = O(h)*. The magnitude of the error can be estimated by using Taylor series expansions of the function *f(x+h)*.

Finite difference formulas based on Taylor series expansions

The Taylor series expansion of the function *f(x)* about the point $x = x_0$ is given by the formula

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} \cdot (x - x_0)^n .$$

Where $f^{(n)}(x_0) = (d^n f/dx^n)|_{x=x0}$, and $f^{(0)}(x_0) = f(x_0)$.

If we let $x = x_0+h$, then $x-x_0 = h$, and the series can be written as

$$f(x_0 + h) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} \cdot h^n = f(x_0) + \frac{f'(x_0)}{1!} \cdot h + \frac{f''(x_0)}{2!} \cdot h^2 + O(h^3),$$

Where the expression $O(h^3)$ represents the remaining terms of the series and indicates that the leading term is of order h^3. Because h is a small quantity, we can write $1 > h$, and $h>h^2>h^3>h^4>\ldots$ Therefore, the remaining of the series represented by $O(h^3)$ provides the order of the error incurred in neglecting this part of the series expansion when calculating $f(x_0+h)$.

From the Taylor series expansion shown above we can obtain an expression for the derivative $f'(x_0)$ as

$$f'(x_0) = \frac{f(x_0+h)-f(x_0)}{h} + \frac{f''(x_0)}{2!}\cdot h + O(h^2) = \frac{f(x_0+h)-f(x_0)}{h} + O(h).$$

In practical applications of finite differences, we will replace the first-order derivative df/dx at $x = x_0$, with the expression $(f(x_0+h)-f(x_0))/h$, selecting an appropriate value for h, and indicating that the error introduced in the calculation is of order h, i.e., $error = O(h)$.

Forward, backward and centered finite difference approximations to the first derivative

The approximation

$$df/dx = (f(x_0+h)-f(x_0))/h$$

is called a <u>*forward difference formula*</u> because the derivative is based on the value $x = x_0$ and it involves the function $f(x)$ evaluated at $x = x_0+h$, i.e., at a point located forward from x_0 by an increment h.

If we include the values of $f(x)$ at $x = x_0 - h$, and $x = x_0$, the approximation is written as

$$df/dx = (f(x_0)-f(x_0-h))/h$$

and is called a <u>*backward difference formula*</u>. The order of the error is still $O(h)$.

A centered difference formula for df/dx will include the points $(x_0-h,f(x_0-h))$ and $(x_0+h,f(x_0+h))$. To find the expression for the formula as well as the order of the error we use the Taylor series expansion of $f(x)$ once more. First we write the equation corresponding to a forward expansion:

$$f(x_0+h) = f(x_0)+f'(x_0)\cdot h+1/2\cdot f''(x_0)\cdot h^2+1/6\cdot f^{(3)}(x_0)\cdot h^3 + O(h^4).$$

Next, we write the equation for a backward expansion:

$$f(x_0-h) = f(x_0)-f'(x_0)\cdot h+1/2\cdot f''(x_0)\cdot h^2-1/6\cdot f^{(3)}(x_0)\cdot h^3 + O(h^4).$$

Subtracting these two equations results in

$$f(x_0+h)- f(x_0-h) = 2\cdot f'(x_0)\cdot h+1/3\cdot f^{(3)}(x_0)\cdot h^3+O(h^5).$$

Notice that the even terms in h, i.e., $h^2, h^4, \ldots$, vanish. Therefore, the order of the remaining terms in this last expression is $O(h^5)$. Solving for $f'(x_0)$ from the last result produces the following <u>*centered difference formula for the first derivative*</u>:

$$\frac{df}{dx}\Big|_{x=x_0} = \frac{f(x_0+h)-f(x_0-h)}{2\cdot h} + \frac{1}{3}\cdot f^{(3)}(x)\cdot h^2 + O(h^4),$$

or,

$$\frac{df}{dx} = \frac{f(x_0+h) - f(x_0-h)}{2\cdot h} + O(h^2).$$

This result indicates that the centered difference formula has an error of the order $O(h^2)$, while the forward and backward difference formulas had an error of the order $O(h)$. Since $h^2<h$, the error introduced in using the centered difference formula to approximate a first derivative will be smaller than if the forward or backward difference formulas are used.

Forward, backward and centered finite difference approximations to the second derivative

To obtain a *centered finite difference formula for the second derivative*, we'll start by using the equations for the forward and backward Taylor series expansions from the previous section but including terms up to $O(h^5)$, i.e.,

$$f(x_0+h) = f(x_0)+f'(x_0)\cdot h+1/2\cdot f''(x_0)\cdot h^2+1/6\cdot f^{(3)}(x_0)\ \cdot h^3 + 1/24\cdot f^{(4)}(x_0)\ \cdot h^4 + O(h^5).$$

and

$$f(x_0-h) = f(x_0)-f'(x_0)\cdot h+1/2\cdot f''(x_0)\cdot h^2-1/6\cdot f^{(3)}(x_0)\ \cdot h^3 + 1/24\cdot f^{(4)}(x_0)\ \cdot h^4 - O(h^4).$$

Next, add the two equations and solve for $f''(x_0)$:

$$d^2f/dx^2 = [f(x_0+h)-2\cdot f(x_0)+f(x_0-h)]/h^2 + O(h^2).$$

Forward and backward finite difference formulas for the second derivatives are given, respectively, by

$$d^2f/dx^2 = [f(x_0+2\cdot h)-2\cdot f(x_0+h)+f(x0)]/h^2 + O(h),$$

and

$$d^2f/dx^2 = [f(x_0)\ -2\cdot f(x_0-h)+f(x_0-2\cdot h)]/h^2 + O(h).$$

Solution of a first-order ODE using finite differences - Euler forward method

Consider the ordinary differential equation,

$$dy/dx = g(x,y),$$

subject to the boundary condition,

$$y(x_1) = y_1.$$

To solve this differential equation numerically, we need to use one of the formulas for finite differences presented earlier. Suppose that we use the forward difference approximation for *dy/dx*;, i.e.,

$$dy/dx = (y(x+h)-y(x))/h.$$

Then, the differential equation is transformed into the following difference equation:

$$(y(x+h)-y(x))/h = g(x,y),$$

from which,

$$y(x+h) = y(x)+h\cdot g(x,y).$$

This result is known as *Euler's forward method* for numerical solution of first-order ODEs.

Since we know the boundary condition (x_1,y_1) we can start by solving for y at $x_2 = x_1+h$, then we solve for y at $x_3 = x_2+h$, and so on. In this way, we generate a series of points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, which will represent the numerical solution to the original ODE. The upper limit of the independent variable x_n is either given or selected arbitrarily during the solution.

The term "*discretizing the domain of the independent variable*" refers to obtaining a series of values of the independent variable, namely, x_i, $i = 1,2;,\ldots, n$, that will be used in the solution. Suppose that the range of the independent variable (a,b) is known, and that we use a constant value $h = \Delta x$ to divide the range into n equal intervals. By making $x_1 = a$, and $x_n = b$, then we find that the values of x_i, $i = 2,3, \ldots n$, are given by

$$x_i = x_1 +(i-1)\cdot\Delta x = a+(i-1)\cdot\Delta x,$$

and that for $i = n$, $x_n = x_1 +(n-1)\cdot\Delta x$. This latter result can be used to find n given Δx,

$$n = (x_n -x_1)/ \Delta x + 1 = (b-a)/ \Delta x + 1,$$

or, to find Δx given n,

$$\Delta x = (x_n-x_1)/(n-1) = (b-a)/(n-1).$$

The recurrent equation for solving for y is given by

$$y_{i+1} = y_i + \Delta x\cdot g(x_i,y_i),$$

for $i = 1,2, \ldots, n$-1. Because the method solves $y_{i+1} = f(x_i,y_i, \Delta x)$, i.e., one value of the dependent variable at a time, the method is said to be an *explicit method*.

The following example illustrates the application of the Euler first-order method to the solution of the differential equation dy/dx = g(x,y) = x + y using SCILAB. First, we define function g(x,y):

```
-->deff('[Df]=g(x,y)','Df=x+y')
```

We solve the equation in the range of values of x from x_0 = 0 to x_n = 2.0 with an increment Dx = 0.1. The initial condition is y_0 = 1.0 for x_0 = 0:

```
-->x0 = 0; y0 = 1; Dx = 0.1; xn = 2.0;
```

The following commands generate a vector of values of x, a vector y of the same length of x, initialized with zeros, and determines the value of n as the length of vector y (or x):

```
-->x=[x0:Dx:xn]; y = zeros(x); n = length(y);
```

The following *for…end* loop takes care of calculating the values of y_i for $i = 2,3, \ldots, n$:

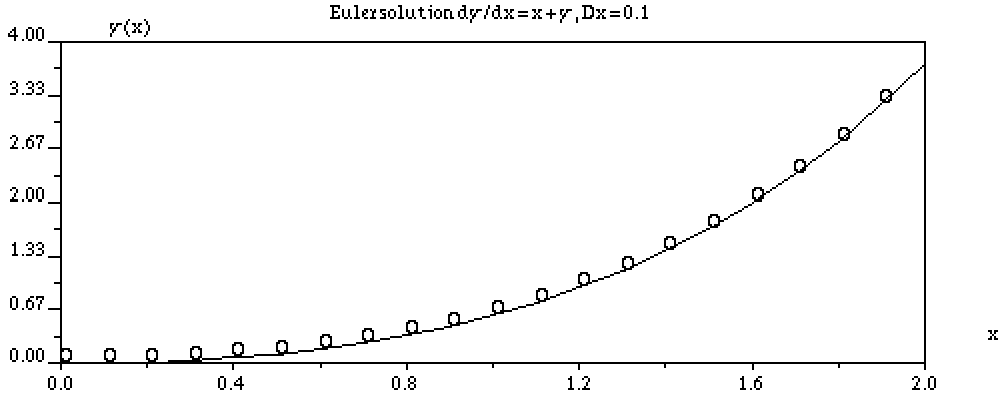
```
-->for j = 1:n-1
-->    y(j+1) = y(j) + Dx*g(x(j),y(j));
-->end;
```

To produce a plot of the results we determine the minimum and maximum values of y:

```
-->ymin = min(y), ymax = max(y)
 ymin  =

    0.
 ymax  =

    3.7274999
```

The plot is generated by using:

```
-->plot2d(x,y,1,'011',' ',[0 0 2 4])

-->plot2d(x,y,-9,'011',' ',[0 0 2 4])

-->xtitle('Euler solution dy/dx = x + y, Dx = 0.1','x','y(x)')
```



A function to implement Euler’s first-order method

The following function, *Euler1*, implements the calculation steps outlined in the previous example. The function detects if there is overflowing introduced in the solution and stops the calculation at that point providing the current results.

```
function [x,y] = Euler1(x0,y0,xn,Dx,g)

//Euler 1st order method solving ODE
//  dy/dx = g(x,y), with initial
//conditions y=y0 at x = x0.  The
//solution is obtained for x = [x0:Dx:xn]
//and returned in y

ymaxAllowed = 1e+100;

x = [x0:Dx:xn]; y = zeros(x); n = length(y); y(1) = y0;

for j = 1:n-1
       y(j+1) = y(j) + Dx*g(x(j),y(j));
       if y(j+1) > ymaxAllowed then
            disp('Euler 1 - WARNING: underflow or overflow');
```

```
          disp('Solution sought in the following range:');
           disp([x0 Dx xn]);
          disp('Solution evaluated in the following range:');
          disp([x0 Dx x(j)]);
           n = j; x = x(1,1:n); y = y(1,1:n);
          break;
       end;
end;

//End function Euler1
```

Next, we use function *Euler1* to solve the differential equation from the previous example, namely, *dy/dx = g(x,y) = x+y*, for different values of the x increment, Δx = 0.5, 0.2, 0.1, and 0.05, with the same initial conditions and range of values of x as before:

```
-->getf('Euler1')

-->deff('[Df]=g(x,y)','Df = x+y')

-->[x1,y1]=Euler1(0,1,2,0.5,g);

-->[x2,y2]=Euler1(0,1,2,0.2,g);

-->[x3,y3]=Euler1(0,1,2,0.1,g);

-->[x4,y4]=Euler1(0,1,2,0.05,g);
```

The exact solution for this equation is $y(x) = -x - 1 + 2e^{x}$.   Set of values of the exact solution are calculated as follows:

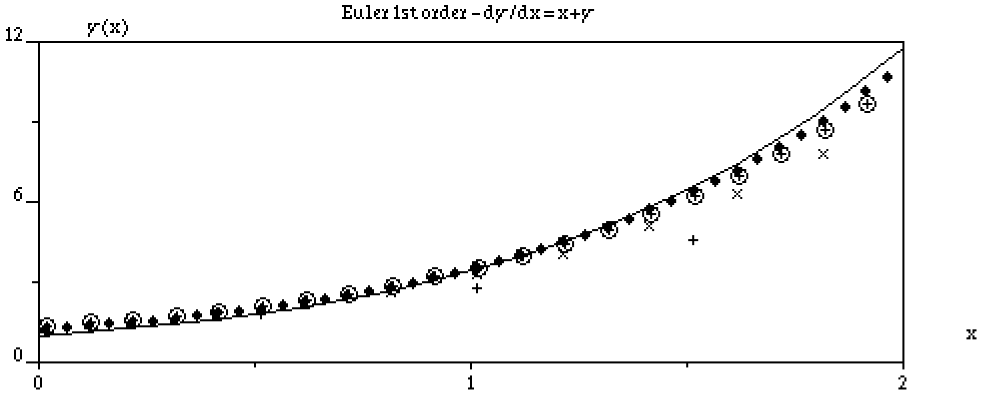
```
-->xx = [0:0.1:2]; yy = -xx-1+2.*exp(xx);
```

To plot the exact and numerical solutions we first determine the minimum and maximum values of y:

```
-->ymax = max([y1 y2 y3 y4 y5 yy])
 ymax  = 11.778112

-->ymin = min([y1 y2 y3 y4 y5 yy])
 ymin  = 1.
```

The plot of the solutions is produced through the use of the following calls to function *plot2d*:

```
-->plot2d(xx,yy,1,'011',' ',[0 0 2 12])
-->plot2d(x1,y1,-1,'011',' ',[0 0 2 12])
-->plot2d(x2,y2,-2,'011',' ',[0 0 2 12])
-->plot2d(x3,y3,-3,'011',' ',[0 0 2 12])
-->plot2d(x4,y4,-4,'011',' ',[0 0 2 12])
-->xtitle('Euler 1st order - dy/dx = x+y','x','y(x)')
```



A second example of application of function *Euler1* is shown next for the differential equation dy/dx = xy + 1, with initial condition x_0 = 0, y_0 = 1, in the range 0 < x < 2, with Δx = 0.5, 0.2, 0.1, 0.05, and 0.01. The SCILAB commands used are exactly the same as before except for the definition of function g(x,y) and the title of the plot. The function *g(x,y) = xy+1* is defined as:

```
-->deff('[Df]=g(x,y)','Df=x*y+1')
Warning :redefining function: g
```

Numerical solutions to the differential equation for the different values of Δx are obtained from:

```
-->[x1,y1]=Euler1(0,1,2,0.5,g);

-->[x2,y2]=Euler1(0,1,2,0.2,g);

-->[x3,y3]=Euler1(0,1,2,0.1,g);

-->[x4,y4]=Euler1(0,1,2,0.05,g);

-->[x5,y5]=Euler1(0,1,2,0.01,g);
```

Next, we determine the minimum and maximum values of y:

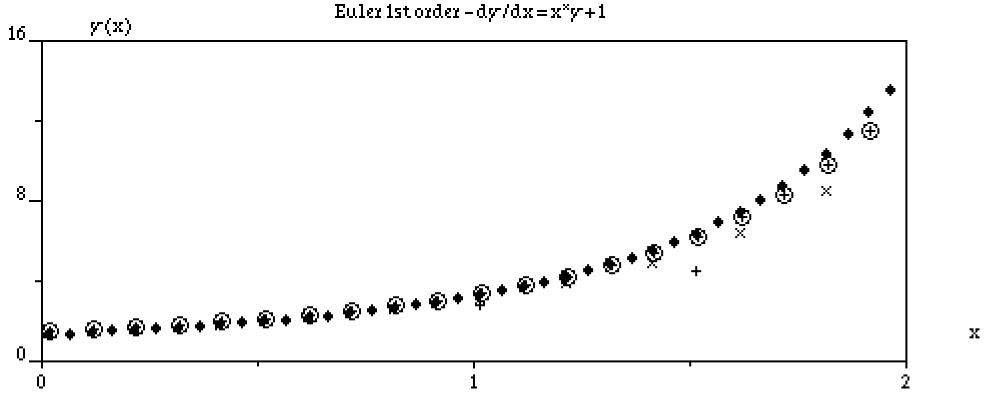
```
-->ymin = min([y1 y2 y3 y4 y5 yy])
 ymin  = 1.

-->ymax = max([y1 y2 y3 y4 y5 yy])
 ymax  = 15.872217
```

We define a plot rectangle as:

```
-->rect = [0 0 2 16]
 rect  =

!   0.    0.    2.    16. !
```

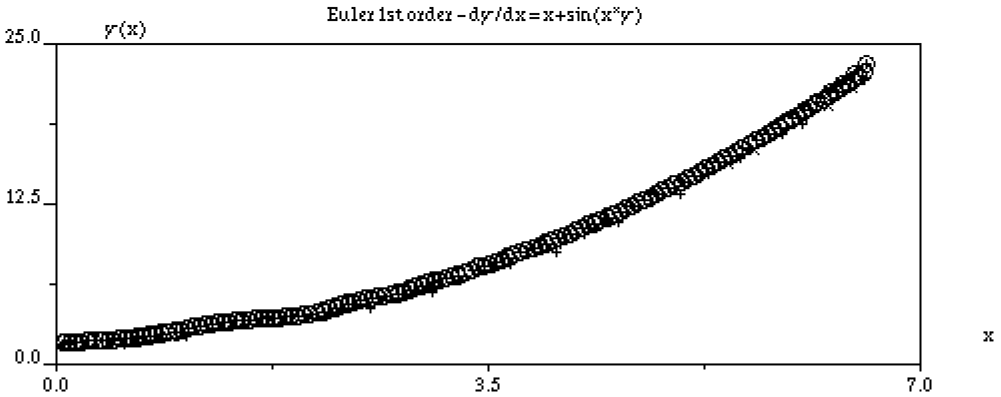
The plot of the numerical solution is accomplished through:

```
-->plot2d(x1,y1,-1,'011',' ',rect)
-->plot2d(x2,y2,-2,'011',' ',rect)
-->plot2d(x3,y3,-3,'011',' ',rect)
-->plot2d(x4,y4,-4,'011',' ',rect)
-->xtitle('Euler 1st order - dy/dx = x*y+1','x','y(x)')
```



The following example solves the differential equation *dy/dx = g(x,y) = x + sin(xy)* in the interval *0< x < 6.5*, with initial conditions $x_0 = 0$, $y_0 = 1$, for $\Delta x = 0.5, 0.2, 0.1, 0.05$, and *0.01*. The steps are the same as in the two previous example:

```
-->deff('[Df]=g(x,y)','Df=x+sin(x*y)')
Warning :redefining function: g


-->[x1,y1]=Euler1(0,1,6.5,0.5,g);

-->[x2,y2]=Euler1(0,1,6.5,0.2,g);

-->[x3,y3]=Euler1(0,1,6.5,0.1,g);

-->[x4,y4]=Euler1(0,1,6.5,0.05,g);

-->ymin = min([y1 y2 y3 y4 y5 yy])
 ymin  =  1.

-->ymax = max([y1 y2 y3 y4 y5 yy])
 ymax  = 22.628614

-->rect = [0 0 7 25]

 rect  =
!   0.    0.    7.    25. !


-->plot2d(x1,y1,-1,'011',' ',rect)
-->plot2d(x2,y2,-2,'011',' ',rect)
-->plot2d(x3,y3,-3,'011',' ',rect)
-->plot2d(x4,y4,-9,'011',' ',rect)
-->xtitle('Euler 1st order - dy/dx = x+sin(x*y)','x','y(x)')
```



Finite difference formulas using indexed variables

In the presentation of the Euler forward method, above, we demonstrated how you can get, from the general formula for the first derivative,

$$dy/dx = [y(x+h)-y(x)]/h,$$

the recurrence formula for the explicit solution, namely,

$$y_{i+1} = y_i + \Delta x \cdot g(x_i,y_i),$$

for $i = 1,2,\dots, n-1$. This suggest re-writing the formula for the derivative as,

$$dy/dx = (y_{i+1}-y_i)/\Delta x + O(\Delta x).$$

Using this sub-index notation, we can summarize the forward, centered, and backward approximations for the first and second derivatives as shown below:

First Derivative

FORWARD: $dy/dx = (y_{i+1}-y_i)/\Delta x+O(\Delta x).$

CENTERED: $dy/dx = (y_{i+1}-y_{i-1})/(2\cdot\Delta x)+O(\Delta x^2).$

BACKWARD: $dy/dx = (y_i-y_{i-1})/\Delta x+O(\Delta x).$

Second Derivative

FORWARD: $d^2y/dx^2 = (y_{i+2}-2\cdot y_{i+1}+y_i)/(\Delta x^2)+O(\Delta x).$

CENTERED: $d^2y/dx^2 = (y_{i+1}-2\cdot y_i+y_{i-1})/(\Delta x^2)+O(\Delta x^2).$

BACKWARD: $d^2y/dx^2 = (y_i–2\cdot y_{i-1}+y_{i-2})/(\Delta x^2)+O(\Delta x).$

Solution of a first-order ODE using finite differences - an implicit method

Consider again the ordinary differential equation, $dy/dx = g(x,y)$, subject to the boundary condition, $y(x_1) = y_1$. This time, however, we use the centered difference approximation for dy/dx, i.e.

$$dy/dx = (y(x+h)-y(x-h))/(2*h).$$

With this approximation the ODE becomes,

$$(y(x+h)-y(x-h))/(2*h) = g(x,y).$$

In terms of sub-indexed variables, this latter equation can be written as:

$$y_{i-1}+2\cdot\Delta x\cdot g(x_i,y_i)-y_{i+1} = 0,\ (\ i = 2,3,\ \dots,\ n-1\ )$$

where the substitutions $y(x) = y_i,y(x+h) = y_{i+1},y(x-h) = y_{i-1}$, and $h = \Delta x$, have been used.

If the function $g(x,y)$ is linear in y, then the equations described above consist of a set of ($n-2$) equations. For example, if $n = 5$, we have 3 equations:

$$y_1+2\cdot\Delta x\ \cdot g(x_2,y_2)-y_3 = 0$$
$$y_2+2\cdot\Delta x\ \cdot g(x_3,y_3)-y_4 = 0$$
$$y_3+2\cdot\Delta x\ \cdot g(x_4,y_4)-y_5 = 0$$

Since y_1 is known (it is the initial condition), there are still 4 unknowns, y_2, y_3, y_4, and y_5. We need to find a fourth equation to obtain a solution. We could use, for example, the forward difference equation applied to i = 1, i.e.,

$$(y_2-y_1)/\Delta x = g(x_1,y_1),$$

or

$$y_2-\Delta x\ \cdot g(x_1,y_1)-y_1= 0.$$

The values of x_i, and n (or Δx), can be obtained as in the Euler forward (explicit) solution.

<u>*Example 1*</u> -- Solve the ODE

$$dy/dx\ = y\ sin(x),$$

with initial conditions $y(0) = 1$, in the interval $0 < x < 5$. Use $\Delta x = 0.5$, or $n = (5-0)/0.5 + 1 = 11.$

Exact solution: the exact is $y(x) = exp(-cos(x))/(cosh(1)-sinh(1)).$

Numerical solution: Using a centered difference formula for dy/dx, i.e.,

$$dy/dx = (y_{i+1}–y_{i-1})/(2\cdot\Delta x),$$

into the ODE, we get $(y_{i+1}–y_{i-1})/(2\cdot\Delta x) = y_i\ sin(x_i)$, which results in the ($n-2$) implicit equations:

$$y_{i-1} + 2\cdot\Delta x\cdot sin(x_i)\cdot y_i - y_{i+1} = 0,\ (i = 2,\ 3,\ \dots,\ n-1).$$

We already know that

$$y_1 = 1$$

(initial condition), thus we have (*n-1*) unknowns left. We still need to come up with an additional equation, which could be obtained by using a forward difference formula for $i = 1$, i.e.,

$$dy/dx|_{x=1} = (y_2-y_1)/\Delta x = -y_1 \sin(x_1),$$

or

$$(1+\Delta x \sin(x_1))\cdot y_1 - \cdot y_2 = 0.$$

These equations can be written in the form of a matrix equation, for example, for n = 5:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1+\Delta x\cdot\sin(x_1) & -1 & 0 & 0 & 0 \\ 1 & 2\cdot\Delta x\cdot\sin(x_2) & -1 & 0 & 0 \\ 0 & 1 & 2\cdot\Delta x\cdot\sin(x_3) & -1 & 0 \\ 0 & 0 & 1 & 2\cdot\Delta x\cdot\sin(x_4) & -1 \end{bmatrix}\cdot\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}=\begin{bmatrix} y0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where *y0* represents the initial condition for y. [**Note**: The data requires $n = 11$. The example for $n = 5$ is presented above to provide a sense of the algorithm to fill out the matrix of data]. The matricial equation can be written as **A·y** = **b**. Matrix A and column vector b can be defined using SCILAB, as indicated below, and the solution found by using left-division. First, we enter the basic data for the problem:

```
-->x0=0;xn=5;Dx=0.5;y0=1;x=[x0:Dx:xn];n=(xn-x0)/Dx+1
 n  = 11.
```

Next, we fill the main diagonal, and the two diagonals below the main diagonal in matrix A using:

```
-->A=zeros(n,n); A(1,1) = 1; for j =2:n, A(j,j)=-1; end;

-->A(2,1)  =  1+Dx*sin(x(1));  for  j  =  3:n,  A(j,j-1)=2*Dx*sin(x(j-1));  end;
//second diagonal

-->for j = 3:n, A(j,j-2) = 1; end;        //Third diagonal
```

The right-hand side vector is defined as:

```
-->b = zeros(n,1); b(1) = 1;              //Right-hand side vector
```

The implicit solution is obtained from:

```
-->y = A\b;                              //Solving for y
```

To compare the implicit solution we calculate also the explicit solution obtained through the Euler first-order solution:

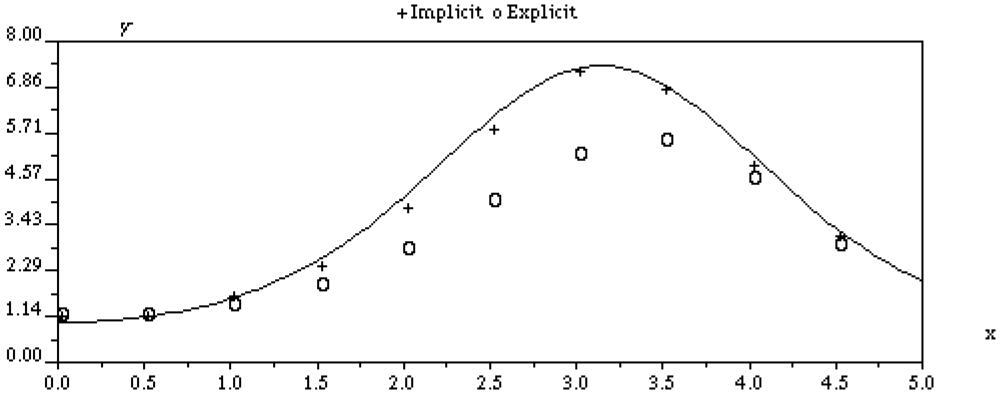
```
-->deff('[z]=ff(x,y)','z=y*sin(x)')
-->getf(‘Euler1’)
-->[xx,yy]=Euler1(x0,y0,xn,Dx,ff);
```

To produce data reproducing the exact solution we use:

```
-->deff('[y]=fE(x)','y=exp(-cos(x))/(cosh(1)-sinh(1))')
-->xE = [0:0.05:5]; yE = fE(xE);
```

The following commands will generate the plot showing the exact, implicit, and explicit solution in the same set of axes:

```
-->plot2d(xE',yE',1,'011',' ', [0 0 5 8])
-->plot2d(x',y,-1,'011',' ',[0 0 5 8])
-->plot2d(xx',yy',-9,'011',' ',[0 0 5 8])
-->xtitle('+ Implicit    o  Explicit','x','y')
```



Explicit versus implicit methods

The idea behind the ***explicit method*** is to be able to obtain values such as

$$y_{i+1} = f(xi, y_i),\ y_{i+2} = f(x_i,x_{i+1},y_i,y_{i+1}),\ \text{etc.}$$

In other words, your solution proceeds by solving explicitly for a new unknown value in the solution array, given all previous values in the array. On the other hand, ***implicit methods*** imply the simultaneous solution of n linear algebraic equations that provide, at once, the elements of the solution array. With this distinction in mind between explicit and implicit methods, we outline explicit and implicit solutions for second-order, linear ODEs.

Outline of explicit solution for a second-order ODE

For example, to solve the ODE

$$d^2y/dx^2+y = 0,$$

in the x-interval (0,20) subject to $y(0) = 1$, $dy/dx = 1$ at $y = 0$. Use $\Delta x = 0.1$.

First, we discretize the differential equation using the finite difference approximation

$$d^2y/dx^2 = (y_{i+2}-2\cdot y_{i+1}+y_i)/(\Delta x^2)\ ,$$

which results in

$$(y_{i+2}-2\cdot y_{i+1}+y_i)/(\Delta x^2)+y_i = 0.$$

An explicit solution can be obtained from the recurrence equation:

$$y_{i+2} = 2\cdot y_{i+1}-(1+\Delta x^2)\cdot y_i,\ \ i = 1, 2, \ldots, n-2;.$$

This equation is based on the two previous values of y_i, therefore, to get started we need the values $y = y_1$, and $y = y_2$. The value y_1 is provided in the initial condition, $y(0) = 1$, i.e.,

$$y_1 = 1.$$

The value of y_2 can be obtained from the second initial condition, $dy/dx = 1$, by replacing the derivative with the finite difference approximation:

$$dy/dx = (y_2 - y_1)/\Delta x,$$

which results in

$$(y_2 - y_1)/\ \Delta x = 1,$$

or

$$y_2 = y_1+\ \Delta x.$$

The x-domain is discretized in a similar fashion as in the previous examples for first derivatives, i.e., by making $x_1 = a$, and $x_n = b$, and computing the values of $x_i,\ i = 2,3, \ldots\ n$, with

$$x_i = x_1 +(i-1)\cdot\Delta x = a+(i-1)\cdot\Delta x,$$

where,

$$n = (x_n-x_1)/\Delta x+1 = (b-a)/\Delta x+1.$$

The implementation of the solution for this example is left as an exercise for the reader.

Outline of the implicit solution for a second-order ODE

We use the same problem from the previous section: solve the ODE

$$d^2y/dx^2+y = 0,$$

in the x-interval (0,20) subject to $y(0) = 1$, $dy/dx = 1$ at $x = 0$. Use $\Delta x = 0.1$.

We discretize the differential equation using the finite difference approximation

$$d^2y/dx^2 = (y_{i+2}-2\cdot y_{i+1}+y_i)/(\Delta x^2)\ ,$$

which results in

$$(y_{i+1}-2*y_i+y_{i-1})/(\Delta x^2)+y_i = 0.$$

From this result we get the following implicit equations:

$$y_{i-1}-(2-\Delta x^2)\cdot y_i+y_{i+1} = 0,$$

for $i = 2,3, \ldots, n-1$. There are a total of ($n-2$) equations. Since we have n unknowns, i.e., $y_1, y_2, \ldots,y_n$, we need two more equations to solve a system of linear equations. The remaining equations are provided by the two initial conditions:

From the initial condition, $y(0) = 1$, we can write $y_1 = 1$. For the second initial condition, $dy/dx = 1$, at $x = 0$, we will use a forward difference, i.e.,

$$dy/dx = (y_2 - y_1)/ \Delta x,$$

or

$$y_2 - y_1 = \Delta x.$$

The x-domain is discretized in a similar fashion as in the previous examples. The n equations resulting from discretizing the domain can be written as a matrix equation similar to that of Example 1. Solution to the matrix equation can be accomplished, for example, through the use of left-division for matrices. The implementation of the solution for this example is left as an exercise for the reader.

SCILAB provides a number of functions for the numerical solution of differential equations. These functions are designed to operate on single differential equations (i.e., similar to the examples presented so far), as well as on systems of differential equations. Therefore, before presenting the SCILAB functions for solving ordinary differential equations, we present some concepts related to systems of such equations.

Systems of ordinary differential equations

To introduce the idea of systems of differential equations we will limit the coverage of the subject to first-order, linear equations with constant coefficients. A system of ordinary differential equations consists of a set of two or more equations with an equal number of unknown functions, $y_1(x)$, $y_2(x)$, etc. As an example consider the following *homogeneous* system:

$$\frac{dy_1}{dx} + 3y_1 - 2y_2 = 0 \ , \quad \frac{dy_2}{dx} - y_1 + y_2 = 0 \ .$$

In a homogeneous system the right-hand sides of the equations are zero. The following example represents a *non-homogeneous* system of ordinary differential equations:

$$\frac{dy_1}{dx} + 2y_1 - 5y_2 = \sin(x) \ , \quad \frac{dy_2}{dx} - 4y_1 + 3y_2 = \mathbf{e}^x \ .$$

Systems of ordinary differential equations using matrices

A homogeneous system of ODEs can be written as a single matrix differential equation by using vector functions and a matrix of coefficients as illustrated in the following example. First, we re-write the homogeneous system presented above to read:

$$\frac{dy_1}{dx} = -3y_1 + 2y_2 \ ,$$

$$\frac{dy_2}{dx} = y_1 - y_2 \ .$$

Then, we define the vector function $\mathbf{f}(x) = [y_1(x)\ y_2(x)]^T$, and the matrix **A** = [-3 2; 1 -1], and write the differential equation:

$$\frac{d}{dx}\ \mathbf{f}(x) = \mathbf{A}\ \mathbf{f}(x).$$

This result is equivalent to writting:

$$\frac{d}{dx}\begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = \begin{bmatrix} -3 & 2 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix}.$$

The non-homogeneous system presented earlier can be re-written as

$$\frac{dy_1}{dx} = -2y_1 + 5y_2 - \sin(x) \ ,$$

$$\frac{dy_2}{dx} = 4y_1 - 3y_2 + \mathbf{e}^x \ .$$

For this system we will use the same vector function $\mathbf{f}$(x) defined earlier, but change the matrix **A** to **A** = [-2 5; 4 -3]. We also need to define a new vector function, $\mathbf{g}$(x) = [-sin(x) exp(x)]^T. With these definitions, we can re-write the non-homogeneous system as:

$$\frac{d}{dx}\ \mathbf{f}(x) = \mathbf{A}\ \mathbf{f}(x) + \mathbf{g}(x),$$

or

$$\frac{d}{dx}\begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = \begin{bmatrix} -2 & 5 \\ 4 & -3 \end{bmatrix} \cdot \begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} + \begin{bmatrix} -\sin(x) \\ \exp(x) \end{bmatrix}.$$

Systems of linear homogeneous ODEs - solution using matrices

Consider the system of linear nonhomogeneous ODEs with constant coefficients given by:

$$dy_1/dx = y_1+y_3,\ dy_2/dx = y_1+y_2-y_3,\ dy_3/dx = 5y_1+y_2+y_3.$$

In matricial form, this can be written as:

$$\frac{d}{dx}\begin{bmatrix} y_1(x) \\ y_2(x) \\ y_3(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & -1 \\ 5 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} y_1(x) \\ y_2(x) \\ y_3(x) \end{bmatrix}.$$

or,

$$\frac{d}{dx}\ \mathbf{f}(x) = \mathbf{A}\ \mathbf{f}(x).$$

We can use the eigenvalues and eigenvectors of matrix $\mathbf{A}$ to obtain the solution to the system of homogeneous equations by following this procedure:

1. Determine eigenvalues of the nxn matrix $\mathbf{A}$. Call these eigenvalues λ_1, λ_2, …, λ_n .

2. Determine the eigenvectors of the nxn matrix $\mathbf{A}$. Call these eigenvectors

$$x_1 = [\,x_{11}\,,\; x_{12}\,,\ldots,\, x_{1,n}\,],\;\; x_2 = [\,x_{21}\,, x_{22}\,,\ldots,\, x_{2,n}\,],\; \ldots.\;\; x_n = [\;x_{n,1}\,,\; x_{n,2}\,,\ldots,\, x_{n,n}\,].$$

3. The general solutions to the system are put together as follows:

$$y_1(x) \;=\; x_{11}\;B_1\,\exp(\,\lambda_1\,x)+x_{21}\;B_2\,\exp(\,\lambda_2\,x)+..\,x_{1,n}\;\;B_n\,\exp(\,\lambda_n\,x),$$

$$y_2(x) \;=\; x_{21}\;B_1\,\exp(\,\lambda_1\,x)+x_{22}\;B_2\,\exp(\,\lambda_2\,x)+..\,x_{2,n}\;\;B_n\,\exp(\,\lambda_n\,x),$$

$$\vdots$$

$$y_n(x) \;=\; x_{n,1}\;B_1\,\exp(\,\lambda_1\,x)+x_{n,2}\;B_2\,\exp(\,\lambda_2\,x)+..\,x_{n,n}\;\;B_n\,\exp(\,\lambda_n\,x).$$

i.e.,

$$y_k(x) \;=\; \sum_{j=1}^{n} x_{k,j}\,B_j\,\mathbf{e}^{(\lambda_j\,x)}\;,\ \mathrm{k} = 1,2,\ldots,\ \mathrm{n}$$

These general solutions include n unknown constants, B_1, B_2, …, B_n. We will need n initial conditions to solve for the n constants to uniquely determine the solution to the system.

For the system under consideration, the solution steps can be translated into SCILAB instructions as shown below. To obtain eigenvalues and eigenvectors we use the user-defined function *eigenvectors* defined in Chapter 5.

```
-->A = [1,0,1;1,1,-1;5,1,1]
 A  =

!   1.    0.    1. !
!   1.    1.  - 1. !
!   5.    1.    1. !

-->getf(‘eigenvectors’)


-->[x,lambda] = eigenvectors(A)
 lambda  =

!   3.1149075  -  .8608059     .7458983 !
 x  =

!    .4170021  -  .3827458  -  .1983289 !
! -  .2198294     .5884340     .9788391 !
!    .8819208     .7122156     .0503957 !
```

The solutions are, therefore,

$$\begin{aligned}
y_1(x) &= \;\; 0.4170021B_1e^{3.1149x} -0.3827458\,B_2e^{\,-0.8608059x} -0.198328\,B_3e^{0.7458983x},\\
y_2(x) &= -\,0.219829\,B_1e^{3.1149x} + 0.5884340B_2e^{\,-0.8608059x} +\;\; 0.9788391\,B_3e^{0.7458983x},\\
y_3(x) &= \;\;\; 0.8819208B_1e^{3.1149x} + 0.7122156\,B_2e^{\,-0.8608059x} + 0.0503957B_3e^{0.7458983x}.
\end{aligned}$$

Substituting the following initial conditions: $y_1(0) = 1$, $y_2(0) = 2$, and $y_3(0) = 3$, produce a system of linear equations:

$$0.4170021B_1 -0.3827458\,B_2 -0.198328\,B_3 = 1$$
$$-\,0.219829\,B_1 + 0.5884340B_2 + 0.9788391\,B_3 = 2$$
$$0.8819208B_1 + 0.7122156\,B_2 + 0.0503957B_3 = 3$$

which can be solved using left-division as follows:

```
-->AA=x;bb=[1;2;3]
 bb  =

!   1. !
!   2. !
!   3. !

-->B=AA\bb
 B  =

!   3.5181246 !
! -  .3600066 !
!   3.049763  !
```

The results are B_1 = `3.5181246`, B_2 = `-0.3600066`, and B_3 = `3.049763`.

To determine the coefficients in the solutions we can use:

```
-->C=AA.*[B B B]

 C  =

!   1.4670652  - 1.3465474  -  .6977459 !
!    .0791400  -  .2118401  -  .3523886 !
!   2.6896495    2.1720889     .153695  !
```

Thus, the solutions are:

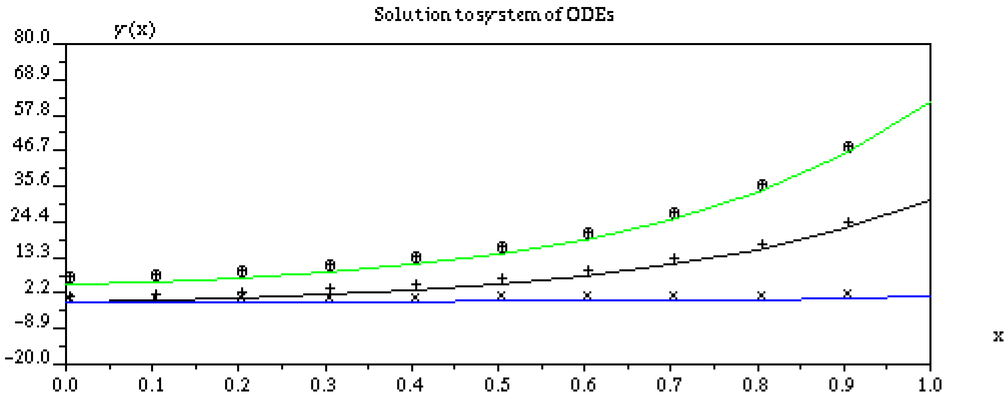
$$y_1(x) = 1.4670652e^{3.1149x} - 1.3465474e^{-0.8608059x} -0.6977459e^{0.7458983x},$$
$$y_1(x) = 0.0791400e^{3.1149x} -0.2118401e^{-0.8608059x} -0.3523886e^{0.7458983x},$$
$$y_1(x) = 2.6896495e^{3.1149x} + 2.1720889e^{-0.8608059x} + 0.153695e^{0.7458983x}.$$

These solutions can be shown graphically by defining the following three functions:

```
-->deff('[y]=y1(x)',['y=0';'for j=1:3';'y=y+C(1,j)*exp(lambda(j)*x)';'end'])

-->deff('[y]=y2(x)',['y=0';'for j=1:3';'y=y+C(2,j)*exp(lambda(j)*x)';'end'])

-->deff('[y]=y3(x)',['y=0';'for j=1:3';'y=y+C(3,j)*exp(lambda(j)*x)';'end'])
```

A plot of the solution is shown next:

```
-->xx=[0:0.1:1];yy1=real(y1(xx));yy2=real(y2(xx));yy3=real(y3(xx));

-->xset('window',1);xset('mark',[-1 -2 -3],1);

-->plot2d([xx' xx' xx'],[yy1' yy2' yy3'],[-1,-2,-3],'011',' ',[0 -20 1 80])

-->plot2d([xx' xx' xx'],[yy1' yy2' yy3'],[1,2,3],'011',' ',[0 -20 1 80])
```



The solution to the non-homogeneous system, $\frac{d}{dx}$ $\mathbf{f}$(x) = $\mathbf{A}$ $\mathbf{f}$(x), can be accomplished in a straightforward manner by using the equation,

$$\mathbf{f}(x) = \exp(\mathbf{A}x)\mathbf{b},$$

where $\mathbf{b}$ is a vector containing the initial conditions, i.e.,

$$\mathbf{b} = [y_1(0)\ y_2(0)\ \ldots\ y_n(0)]^T.$$

and the expression *exp(**A**t)* is a matrix defined as:

$$\exp(\mathbf{A}t) = \mathbf{I} + \mathbf{A}t + \frac{1}{2!}A^2t^2 + \ldots = I + \sum_{k=1}^{\infty}\frac{1}{k!}A^kt^k,$$

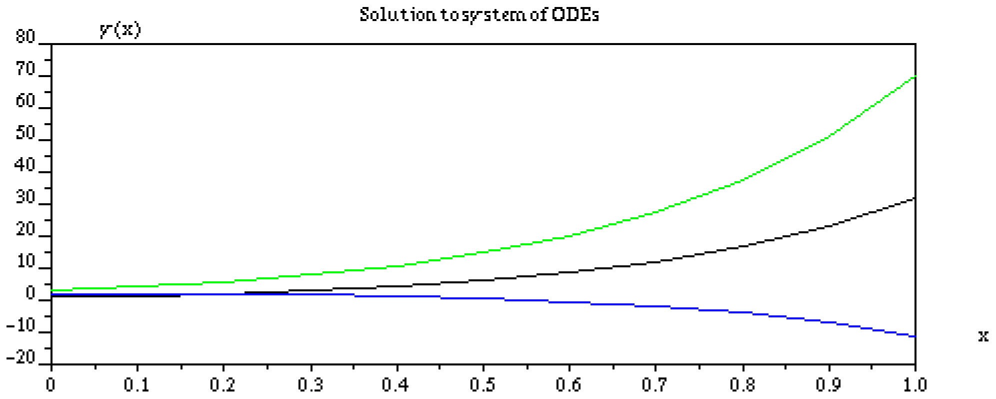
where $\mathbf{I}$ is the identity matrix, A^2 = $\mathbf{AA}$, $A^3 = A^2\mathbf{A}$, … To evaluate the expression *exp(**A**x)*, SCILAB provides function *expm*.

The solution to the linear system under consideration, using the equation, $\mathbf{f}$(x) = exp($\mathbf{A}$x)$\mathbf{b}$, can be obtained as follows using SCILAB (matrix *A* is the same matrix of coefficients of the ODE system defined earlier):

```
-->deff('[y] = fm(t)','y=real(expm(A*t)*bb)')
```

A plot of the solution using this matrix approach is produced with the following SCILAB statements:

```
-->xx=[0:0.1:1]; n = length(xx)
 n  = 11.

-->ym=[];for j=1:n, ym=[ym fm(xx(j))]; end;

-->plot2d([xx',xx',xx'],[ym(1,:)',ym(2,:)',ym(3,:)'],[1,2,3])

-->xtitle('Solution to system of ODEs','x','y(x)')
```



Systems of linear nonhomogeneous ODEs - solution using matrices

Consider the system of linear nonhomogeneous ordinary differential equations with constant coefficients given by

$$\frac{dy_1}{dx} = y_1 + y_3 + \mathbf{e}^x \text{ , } \frac{dy_2}{dx} = y_2 - y_3 + \sin(x) \text{ , and } \frac{dy_3}{dx} = 5\,y_1 + y_2 + y_3 + \cos(x)$$

This system can be written as a matricial differential equation as:

$$\begin{bmatrix} \frac{\partial}{\partial x}\,\mathrm{y1}(x) \\ \frac{\partial}{\partial x}\,\mathrm{y2}(x) \\ \frac{\partial}{\partial x}\,\mathrm{y3}(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 5 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{y1}(x) \\ \mathrm{y2}(x) \\ \mathrm{y3}(x) \end{bmatrix} + \begin{bmatrix} \mathbf{e}^x \\ \sin(x) \\ \cos(x) \end{bmatrix}$$

In general, a nonhomogeneous system can be written in matricial form as

$$\frac{d}{dx}\,\mathbf{f}(x) = \mathbf{A}\,\mathbf{f}(x) + \mathbf{g}(x).$$

For the case under consideration, we have:

$$A := \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 5 & 1 & 1 \end{bmatrix}$$

$$g := \begin{bmatrix} \mathbf{e}^x \\ \sin(x) \\ \cos(x) \end{bmatrix}$$

A solution to the system of ODEs is given by

$$\mathbf{f}(x) = \Phi\,(x)\mathbf{C} + \Phi\,(x) \int \Phi(x)^{(-1)}\, \mathrm{g}(x)\, dx \text{ ,}$$

where *$\Phi(x)$ = exp(**A**x)* is known as a *fundamental matrix* of the system, *$\Phi(x)^{(-1)}$* is the inverse matrix of *$\Phi(x)$,* and ***C*** is a vector of constants, i.e., ***C*** *= [C_1 C_2 … C_n].*

Unlike the homogeneous case, the presence of the integral in the solution ***f****(x)* complicates the calculation or programming of the solution using SCILAB. The reader is referred to symbolic packages such as *Maple* or *Mathematica* to obtain such solutions. Numerical solutions, however, are possible with SCILAB, as will be demonstrated in subsequent sections of this book.

Converting second-order linear equations to a system of equations

A second-order linear ODE of the form $\frac{d^2y}{dx^2}+\frac{b\,dy}{dx}+c\,y=\mathrm{r}(x)$, can be transformed into a linear system of equations by introducing the relationship, $\mathrm{u}(x)=\frac{dy}{dx}$, so that $\frac{d^2y}{dx^2}=\frac{du}{dx}$, thus, the equation reduces to $\frac{du}{dx}+b\,u+c\,y=\mathrm{r}(x)$, or $\frac{du}{dx}=-b\,u-c\,y+\mathrm{r}(x)$. The resulting system of equations is:

$$\frac{du}{dx}=-b\,u-c\,y+\mathrm{r}(x)\ ,$$

$$\frac{dy}{dx}=u\ .$$

Which can be written in matricial form as d**f**/dx = **A f**(x)+**g**(x), with

$$f(x)=\begin{bmatrix}u\\y\end{bmatrix}, A=\begin{bmatrix}-b & -c\\1 & 0\end{bmatrix}, g(x)=\begin{bmatrix}r(x)\\0\end{bmatrix}.$$

For example, the solution to the second order differential equation

$$\frac{d^2y}{dx^2}+\frac{5\,dy}{dx}-3\,y=x\ ,$$

can be obtained by solving the equivalent first-order linear system:

$$\frac{du}{dx}=-5\,u+3\,y+x\ ,$$

$$\frac{dy}{dx}=u\ .$$

The procedure outlined above to transform a second order linear equation can be used to convert a linear equation of order n into a system of first-order linear equations. For example, if the original ODE is written as:

$$\frac{d^n y}{dx^n} + a_{n-1}\frac{d^{(n-1)}y}{dx^{(n-1)}} + \ldots + a_2\frac{d^2 y}{dx^2} + \frac{a_1\,dy}{dx} + a_0 y = \mathrm{r}(x)\ ,$$

we can re-write it as

$$\frac{d^n y}{dx^n} = -a_{n-1}\frac{d^{(n-1)}y}{dx^{(n-1)}} - \ldots - a_2\frac{d^2 y}{dx^2} - \frac{a_1\,dy}{dx} - a_0 y + \mathrm{r}(x)\ ,$$

and transform it into a system of n first-order linear equations given by:

$$\frac{du_{n-1}}{dx} = -a_{n-1}\,u_{n-1} - a_{n-2}\,u_{n-2} - \ldots - a_2\,u_2 - a_1 u_1 - a_0 y + \mathrm{r}(x)\ ,$$

$$\frac{du_{n-2}}{dx} = u_{n-1}\ ,\quad \frac{du_{n-3}}{dx} = u_{n-2}\ ,\ \ldots,\ \frac{du_1}{dx} = u_2\ ,\ \frac{dy}{dx} = u_1\ ,$$

or, in matricial form,

$$\mathbf{f}(x) = \begin{bmatrix} u_{n-1} \\ u_{n-2} \\ \vdots \\ u_2 \\ u_1 \\ y \end{bmatrix},\quad A =, \begin{bmatrix} -a_{n-1} & -a_{n-2} & -a_{n-3} & \cdots & -a_1 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}\quad g(x) = \begin{bmatrix} r(x) \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

For example, to transform the following fourth-order ($n=4$) linear ODE

$$\frac{d^4 y}{dx^4} + \frac{3\,d^3 y}{dx^3} - \frac{2\,d^2 y}{dx^2} + \frac{5\,dy}{dx} + y = 0\ ,$$

subjected to $y = 1$, $\frac{dy}{dx} = -1$, $\frac{d^2 y}{dx^2} = 0$, $\frac{d^3 y}{dx^3} = -1$, at $x = 0$, into a first-order linear system, we would write:

$du_3/dx = -3u_3(x)+2u_2(x)-5u_1(x)-y(x)+x^2/2$, $du_2/dx = u_3(x)$, $du_1/dx = u_2(x)$, and *$dy/dx = u_1(x)$,*

or

$$\frac{d}{dx}\begin{bmatrix} u_3(x) \\ u_2(x) \\ u_1(x) \\ y(x) \end{bmatrix} = \begin{bmatrix} -3 & 2 & -5 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_3(x) \\ u_2(x) \\ u_1(x) \\ y(x) \end{bmatrix} + \begin{bmatrix} x^2/2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

with $v(x) = [u_3(x);u_2(x);u_1(x);\ y(x)]^T$, $A = [-3,2,-5,-1;1,0,0,0;0,1,0,0;0,0,1,0];$ and $g(x) = [x^2/;0;\ 0;\ 0]^T$, the system of differential equations is written as $dv/dx = Av+g(x)$. The initial conditions are $y(0) = 1$, $u_1(0) = dy/dx = -1$, $u_2(0) = du_1/dx = d^2y/dx^2 = 0$, $u_3(0) = du_2/dx = d^2u_1/dx^2 = d^3y/dx^3 = -1$, or $u_0 = [\ -1;0;-1;1]$.

SCILAB functions for the numerical solutions of initial value problems (IVP)

SCILAB provides function *ode* for the solution of initial value problems, i.e., those subject to initial, rather than boundary, conditions.

Function *ode*

The simplest call to function *ode* is

y=ode(y0,x0,x,f)

where the point *(x0,y0)* represents the initial conditions for the differential equation

$$dy/dx = f(x,y),$$

and *x* is a vector of values of the independent variable *x*.

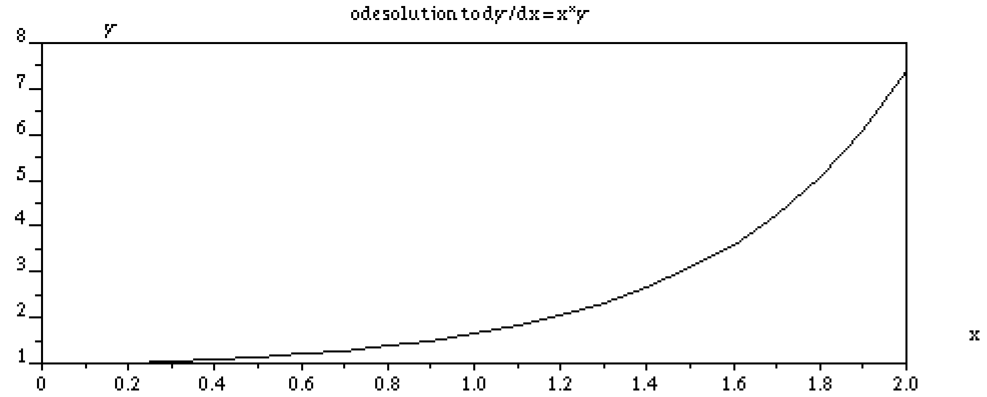
The function *f* can be a single function *f(x,y)* or a vector of functions,

$$f(x,y) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \\ \vdots \\ f_n(x,y) \end{bmatrix} = \begin{bmatrix} f_1(x,y_1,y_2,\cdots,y_n) \\ f_2(x,y_1,y_2,\cdots,y_n) \\ \vdots \\ f_n(x,y_1,y_2,\cdots,y_n) \end{bmatrix}.$$

In the latter case, *y* is also a vector representing the variables $y = [y_1\, y_2\, \dots\, y_n]^T$.

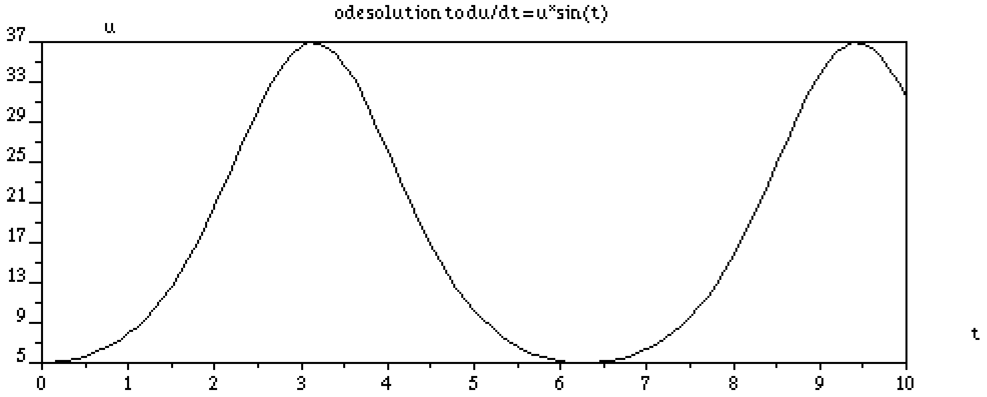
The first example shows the solution to the differential equation *dy/dx = xy*, in the range *0<x<2*, with initial conditions *y(0) = 1*.

```
-->deff('[z]=f(x,y)','z=x.*y')

-->x0 = 0; y0 = 1; Dx = 0.1; xn = 2; x = [x0:Dx:xn];

-->y = ode(y0,x0,x,f);

-->plot(x,y,'x','y','ode solution to dy/dx = x*y')
```



In the second example, we solve the differential equation *du/dt = u sin(t),* in the range *0<t<10*, with initial condition *u(0) = 5*.

```
->deff('[w]=g(t,u)','w=u.*sin(t)')

-->t0=0;Dt=0.1;tn=1;u0=5;t=[t0:Dt:tn];

-->u = ode(u0,t0,t,g);

-->plot(t,u,'t','u','ode solution to du/dt = u*sin(t)')
```



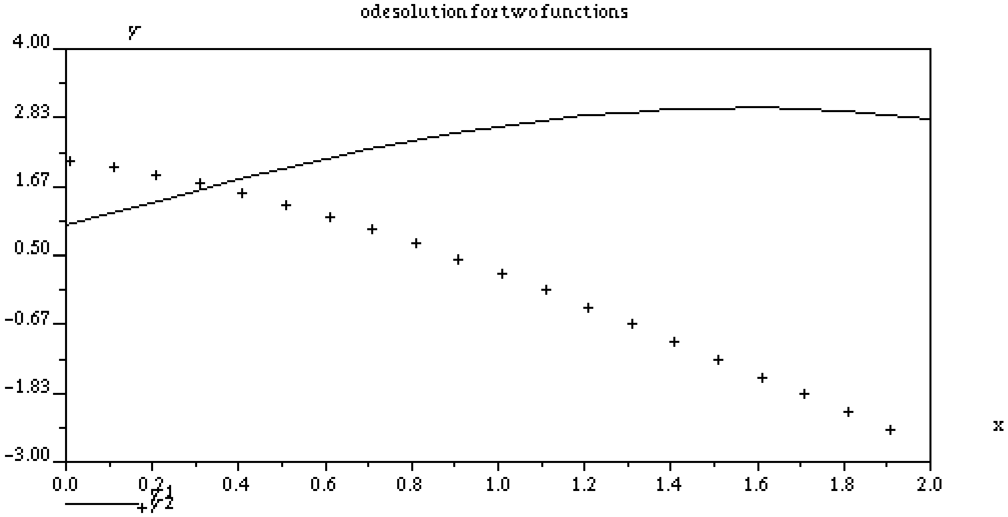
The next example uses a system of two differential equations, namely,

$$dy_1/dx = y_2 + x$$

$$dy_2/dx = -y_1$$

with initial conditions $y_1=1$, $y_2 = 2$, for $x_0 = 0$.

```
-->deff('[w]=f(x,y)',['y1=y(2)+x','y2=-y(1)','w = [y1;y2]'])

-->x0 = 0; Dx = 0.1; xn = 2; y0 = [1;2];

-->x = [x0:Dx:xn]; y = ode(y0,x0,x,f);

-->min(y),max(y)
 ans  = - 2.8322936
 ans  =   2.999147

-->plot2d([x',x'],[y(1,:)', y(2,:)'],[1,-1],'111','y1@y2',[0 -3 2 4])

-->xtitle('ode solution for two functions','x','y')
```



Numerical methods used in function *ode*

Function *ode* uses as default the numerical method known as the Adams method. In general, Adams methods for solving the differential equation *dy/dx = f(x,y)* basically consist in obtaining $y_{k+1} = y(x_{k+1})$ through the fitting of a polynomial that interpolates *f(x,y)* at selected points (x_j,y_j) of the solution. The *k*-th order Adams-Bashforth method is an explicit method that uses the current point (x_n,y_n) and *k-1* points for the solution. The *k*-th order Adams-Moulton method is an implicit method using points (x_{n+1},y_{n+1}), (x_n,y_n) and *k-2* previously calculated points. Adams methods are also known as *predictor-corrector* methods because their solution involves an initial *predictor* step, which is then modified in a *corrector* step that improves the solution.

Function *ode* can be called with an additional argument that specifies the numerical method to be used for a solution. This argument becomes the first argument in the call to function *ode*. The function call, if such optional argument is used, is

[y] = ode([type],y0,t0,t,f)

where *type* is one of the following character string: *"adams", "stiff", "rk", "rkf", "fix", "discrete",* or *"roots".* The meaning of each of these options is presented next:

"adams": Default value. Uses the Adams predictor-corrector method for the solution.

"stiff": This option invokes the use of a Backward Differentiation Formula (BDF) for the solution of stiff ordinary differential equations.

"rk": Invokes an adaptive, 4-th order, Runge-Kutta method.

"rkf": Invokes a Fehlberg's Runge-Kutta method of order 4 and 5 (RKF45).

"fix": Same solver as "rkf", a simpler user interface.

"root": ODE solver with root finding capabilities.

"discrete": Discrete time simulation.

The type options “root” and “discrete” do not represent different methods for solving ordinary differential equations. Instead, they represent enhanced abilities for function *ode* for root finding and combining discrete and continuous systems. The applications of these options will be presented later.

Adams-Bashforth methods

These methods involve explicit functions for y_{n+1} in terms of *k* points in the solution. The equations describing the Adams-Bashforth methods of orders 1 through 5 are shown below. In these equations the term f_n represents $f(x_n,y_n)$ and Δx is the increment in the independent variable x:

$y_{n+1} = y_n + \Delta x \cdot f_n$ (same as Euler 1st order method)
$y_{n+1} = y_n + (\Delta x /2)(3f_n - f_{n-1})$
$y_{n+1} = y_n + (\Delta x /12)(23f_n - 16f_{n-1} + 5f_{n-2})$
$y_{n+1} = y_n + (\Delta x /24)(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$
$y_{n+1} = y_n + (\Delta x /720)(1901f_n - 2774f_{n-1} + 2616f_{n-2} - 1274f_{n-3} + 251f_{n-4})$

Adams-Moulton methods

The following equations represent the Adams-Moulton methods of orders 1 through 5. These are implicit methods, in which the term $f_{n+1} = f(x_{n+1},y_{n+1})$:

$y_{n+1} = y_n + \Delta x \cdot f_{n+1}$
$y_{n+1} = y_n + (\Delta x /2)(f_{n+1} - f_n)$
$y_{n+1} = y_n + (\Delta x /12)(f_{n+1} + 8f_n - f_{n-1})$
$y_{n+1} = y_n + (\Delta x /24)(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$
$y_{n+1} = y_n + (\Delta x /720)(251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3} + 251)$

Adams-Bashforth-Moulton methods

The Adams-Bashforth-Moulton predictor-corrector method uses one of the Adams-Bashforth formulas to produce a first approximation for the solution (a predictor value), followed by an evaluation of the function $f(x_{n+1},y_{n+1}) = f_{n+1}$, which is used in an Adams-Moulton formula to produce a better approximation to the solution (a corrector value). Depending on the order of the formula used for the predictor part of the algorithm, we need one or more values of the solution to get the algorithm started. The Runge-Kutta method, described below, can be used to generate those few initial data values necessary to get the Adams-Bashforth-Moulton procedure started.

Runge-Kutta methods

The general approach for the Runge-Kutta methods consists of the following equations

$$k_1 = \Delta x\ f(x_n,y_n)$$
$$k_2 = \Delta x\ f(x_n+\alpha h,y_n+\beta k_1)$$
$$y_{n+1} = y_n + \alpha k_1 + \beta k_2$$

where the values of the parameters α and β are selected from Taylor series expansions representing different orders of the approximation. For example, for an error of order Δx^3, the Runge-Kutta method is given by

$$k_1 = \Delta x\ f(x_n,y_n)$$
$$k_2 = \Delta x\ f(x_n+ h,y_n+k_1)$$
$$y_{n+1} = y_n + (k_1 + k_2)/2.$$

A fourth-order Runge-Kutta method is given by the formulas

$$k_1 = \Delta x\ f(x_n,y_n)$$
$$k_2 = \Delta x\ f(x_n+ \Delta x\ /2,y_n+k_1/2)$$
$$k_3 = \Delta x\ f(x_n+ \Delta x\ /2,y_n+k_2/2)$$
$$k_4 = \Delta x\ f(x_n+ \Delta x,\ y_n + k_3)$$
$$y_{n+1} = y_n + (k_1 + 2k_2+ 2k_3 +k_4)/6.$$

A fourth-fifth order Runge-Kutta method produces a fourth order and a fifth order estimate (y_{n+1},z_{n+1}) of the function through the following algorithm:

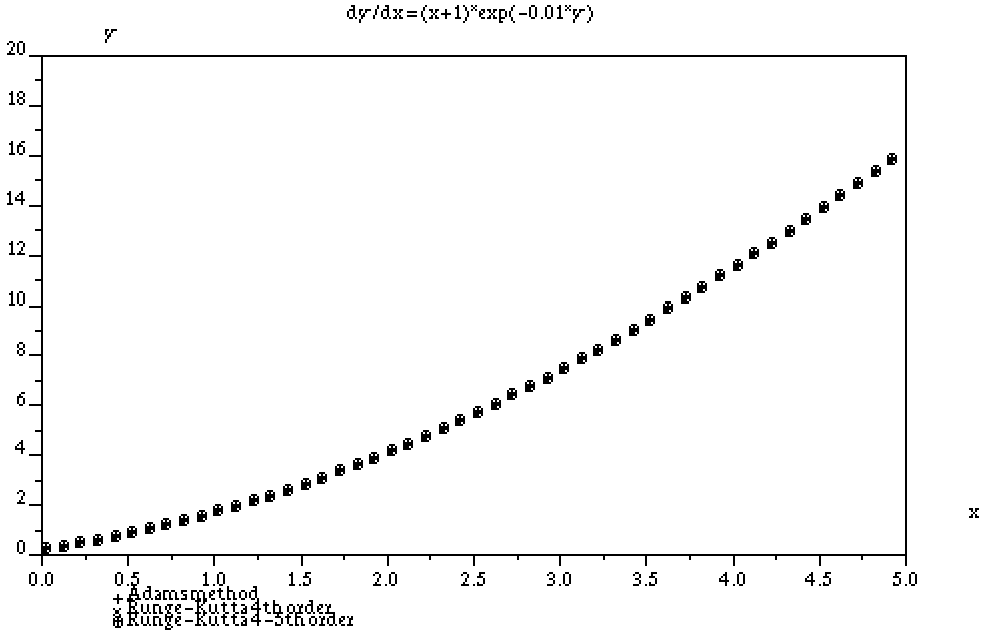
$$k_1 = \Delta x\cdot f(x_n,y_n)$$
$$k_2 = \Delta x\cdot f(x_n+\frac{\Delta x}{4},y_n+\frac{k_1}{4})$$
$$k_3 = \Delta x\cdot f(x_n+\frac{3\Delta x}{8},y_n+\frac{3k_1}{32}+\frac{9k_2}{32})$$
$$k_4 = \Delta x\cdot f(x_n+\frac{12\Delta x}{13},y_n+\frac{1932k_1}{2197}-\frac{7200k_2}{2197}+\frac{7296k_3}{2197})$$
$$k_5 = \Delta x\cdot f(x_n+\Delta x,y_n+\frac{439k_1}{216}-8k_2+\frac{3680k_3}{513}-\frac{845k_4}{4104})$$
$$k_6 = \Delta x\cdot f(x_n+\frac{\Delta x}{2},y_n-\frac{8k_1}{27}+2k_2-\frac{3544k_3}{2565}+\frac{1859k_4}{4104}-\frac{11k_5}{40})$$

$$y_{n+1} = y_n+\frac{25}{216}k_1+\frac{1408}{2565}k_3+\frac{2197}{4104}k_4-\frac{1}{5}k_5$$
$$z_{n+1} = y_n+\frac{16}{135}k_1+\frac{6656}{12825}k_3+\frac{28561}{56430}k_4-\frac{9}{50}k_5+\frac{2}{55}k_6$$

Examples of function *ode* with different numerical methods

In this first example we solve the differential equation $dy/dx = (x+1)e^{-0.01y}$ in the range $0<x<5$, with initial condition $y(0) = 0$ using different methods within function *ode*:

```
-->deff('[z]=f(x,y)','z = (x+1).*exp(-0.01.*y)')

-->x0=0;Dx=0.1;xn=5.0;y0=0;x=[x0:Dx:xn];

-->y1 = ode('adams',y0,x0,x,f);

-->y2 = ode('rk',y0,x0,x,f);

-->y3 = ode('rkf',y0,x0,x,f);

-->min([y1 y2 y3]),max([y1 y2 y3])
 ans  = 0.
 ans  = 16.126815

-->plot2d([x',x',x'],[y1',y2',y3'],[-1,-2,-3],'111',...
-->'Adams method@Runge-Kutta 4th order@Runge-Kutta 4-5th order',...
-->[0 0 5 20])

-->xtitle('dy/dx = (x+1)*exp(-0.01*y)','x','y')
```



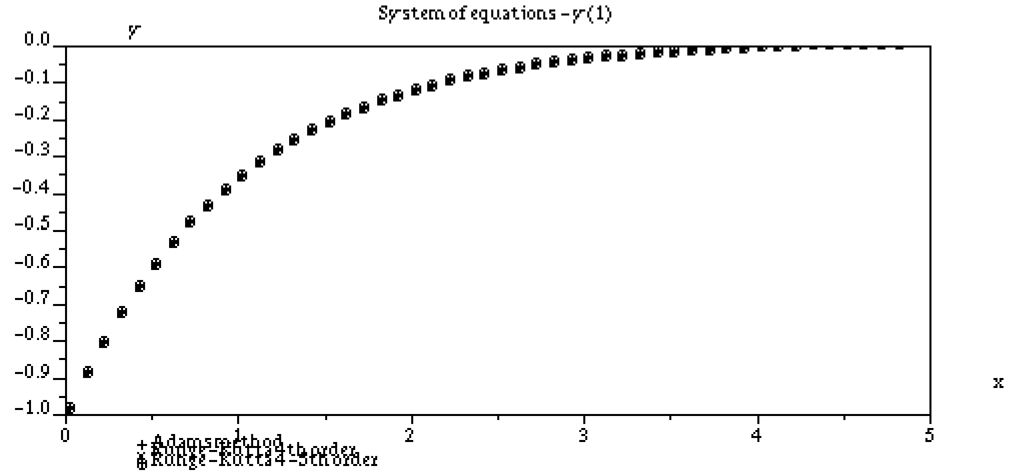
In the following example we solve the system of equations

$$dy_1/dx = - y_1,\ dy_2/dx = y_2$$

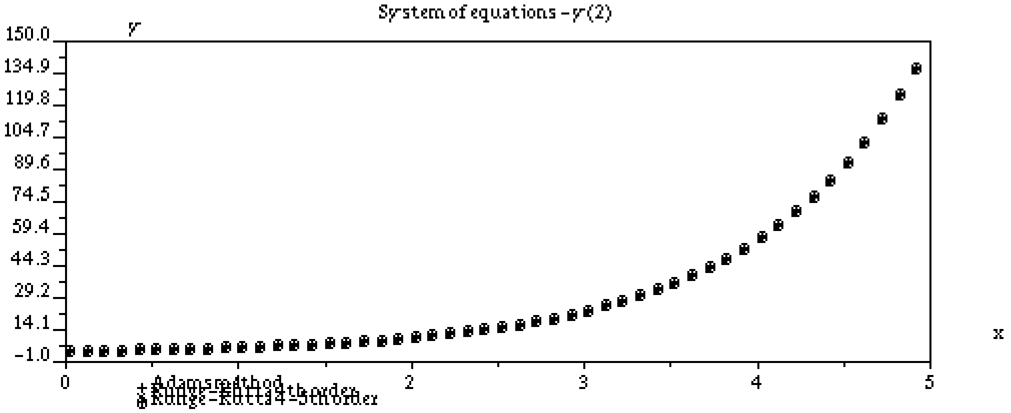
in the range 0<x<5, with initial conditions $y_1(0) = -1$, $y_2(0) = 1$:

```
-->deff('[z]=f(x,y)',['z1=-y(1)','z2=y(2)','z=[z1;z2]'])
-->x0=0;Dx=0.1;xn=5.0;y0=[-1;1];x=[x0:Dx:xn];
-->y1 = ode('adams',y0,x0,x,f);
```

```
-->y2 = ode('rk',y0,x0,x,f);
-->y3 = ode('rkf',y0,x0,x,f);

-->min(y1), max(y1)
 ans  =   - 1.
 ans  =   148.4132

-->plot2d([x',x',x'],[y1(1,:)',y2(1,:)',y3(1,:)'],[-1,-2,-3],'111',...
-->'Adams method@Runge-Kutta 4th order@Runge-Kutta 4-5th order',...
-->[0 -1 5 0])

-->xtitle('System of equations - y(1)','x','y')
```



```
-->plot2d([x',x',x'],[y1(2,:)',y2(2,:)',y3(2,:)'],[-1,-2,-3],'111',...
-->'Adams method@Runge-Kutta 4th order@Runge-Kutta 4-5th order',...
-->[0 -1 5 150])
```



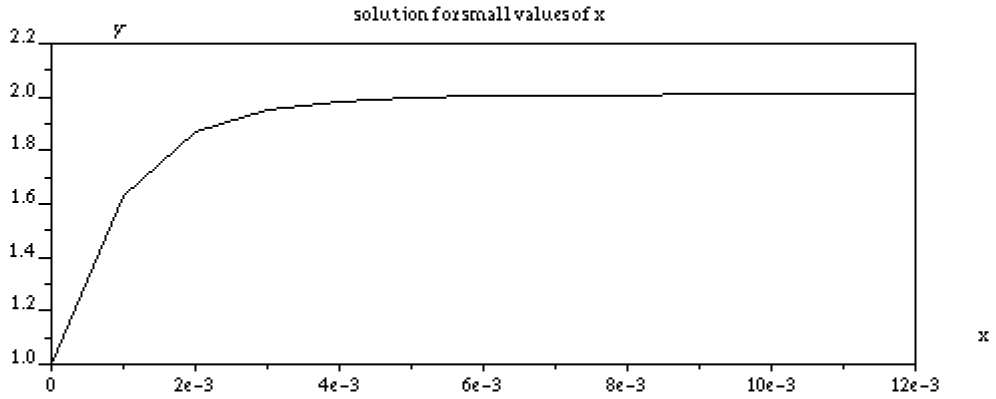
Stiff ordinary differential equations

There are a number of definitions of a stiff matrix (see, for example, Hoffman, J.D., 1992, "*Numerical Methods for Engineers and Scientists*," McGraw-Hill, Inc., New York) based on their requirements for stability or on the presence of a rapidly decaying transient. The concept of *stability* in the numerical solution of an ordinary differential equation involves the analysis of the error inherent in the numerical method. If a small error introduced in the first step of the numerical solution remains constant or decays as solution steps accumulate, the solution is said to be *stable*. If, on the other hand, a small error involved in the first step of the solution increases without bound as the solution steps accumulate the solution is said to be *unstable*.

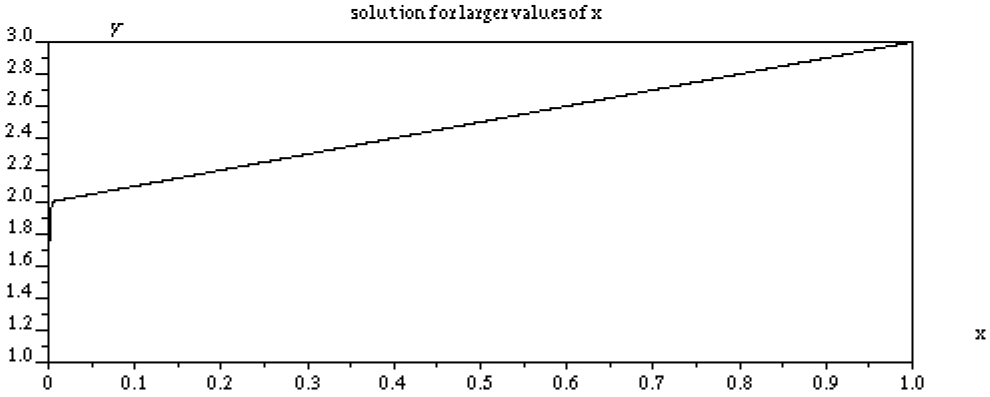
Thus, based on the numerical solution of an ordinary differential equation, a *stiff* equation is defined as one which requires a step size so small for stability that round-off errors (inherent in computer calculations) become significant. Alternatively, a *stiff* equation is one that contains some transient component that decays rapidly compared to the main transient component.

As an example, consider the ordinary differential equation $dy/dx = f(x,y) = -1000(y-x)+2001$, whose exact solution is $y(x) = -exp(-1000x)+x+2$. The solution is composed of two parts, $y_1(x) = -exp(-1000x)$, and $y_2(x) = x+2$, which vary at significantly different rates as x varies. The first component, $y_1(x)$, goes quickly to zero, while the second component, $y_2(x)$, produces a simple linear solution. Component $y_1(x)$ is only significant for very small values of x. The following two graphs illustrate the behavior of the solution $y(x)$ for small and relatively large values of x.

```
-->deff('[y]=f(x)','y=-exp(-1000.*x)+x+2')

-->plot(xs,ys,'x','y','solution for small values of x')
```

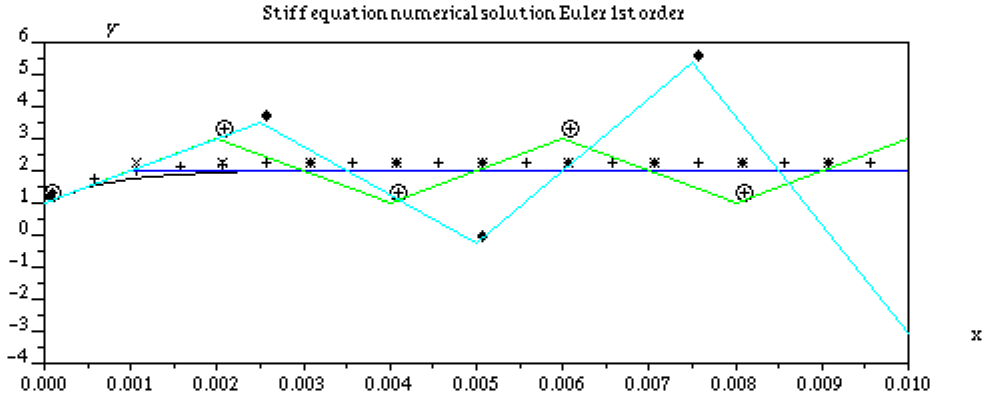


```
-->xl=[0:0.001:1]; yl=f(xl);

-->plot(xl,yl,'x','y','solution for larger values of x')
```



To determine the effect that the increment in the independent variable, Δx, has on the solution, we attempt solutions using function *Euler1* that implements Euler's first-order method using values of Δx = 0.0005, 0.0010, 0.0020, 0.0025. We then plot the solution to compare the results for different values of Δx.

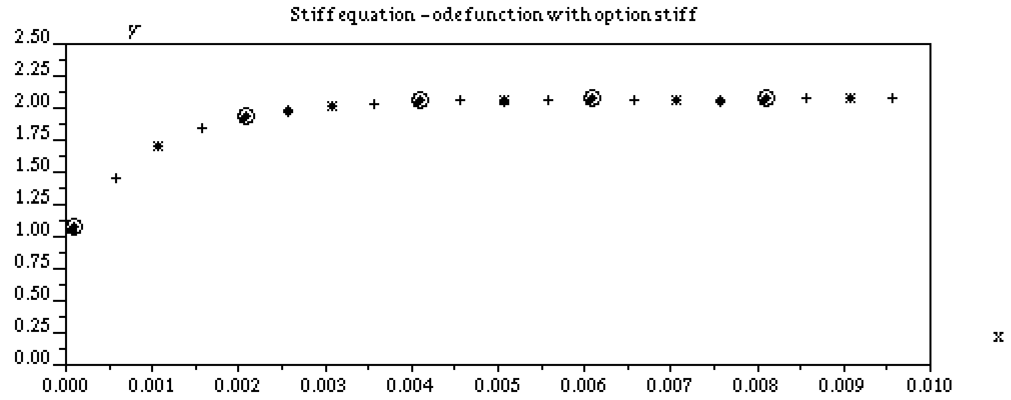
```
-->deff('[dydx]=g(x,y)','dydx=-1000.*(y-x)+2001')
-->getf('Euler1')
-->Dx=0.0005;x0=0;xn=0.01;y0=1;[x1,y1]=Euler1(x0,y0,xn,Dx,g);
-->Dx=0.0010;x0=0;xn=0.01;y0=1;[x2,y2]=Euler1(x0,y0,xn,Dx,g);
-->Dx=0.0020;x0=0;xn=0.01;y0=1;[x3,y3]=Euler1(x0,y0,xn,Dx,g);
-->Dx=0.0025;x0=0;xn=0.01;y0=1;[x4,y4]=Euler1(x0,y0,xn,Dx,g);

-->min([y1 y2 y3 y4]), max([y1 y2 y3 y4])
 ans  = - 3.0525
 ans  =   5.3825

-->rect = [0 -4 0.01 6];
-->plot2d(x1,y1,-1,'011',' ',rect)
-->plot2d(x2,y2,-2,'011',' ',rect)
-->plot2d(x3,y3,-3,'011',' ',rect)
-->plot2d(x4,y4,-4,'011',' ',rect)
-->plot2d(x1,y1,1,'011',' ',rect)
-->plot2d(x2,y2,2,'011',' ',rect)
-->plot2d(x3,y3,3,'011',' ',rect)
-->plot2d(x4,y4,4,'011',' ',rect)
-->plot(xl,yl,'x','y','solution for larger values of x')
-->xtitle('Stiff equation numerical solution Euler 1st order','x','y')
```



The points indicated by the crosses and asterisks correspond to the values of Δx = 0.0005 and 0.0010. The values indicated by the crosses enclosed in circles correspond to Δx = 0.0020, and the dark diamond marks correspond to Δx = 0.0025. The result indicates that Δx must be of size 0.0010 or smaller to ensure stability. This is a characteristic of a stiff ordinary differential equation since a very small step Δx is required for stability.

Function *ode* allows the use of the argument ‘stiff’ to change the numerical method for the solution to a Backward Differentiation Formula (BDF) for its solution. The following SCILAB commands demonstrate the use of function *ode* with the argument ‘stiff’ for the solution of the differential equation under consideration. The graph thus produced shows that the solutions are stable for the four values of Δx used.

```
-->deff('[dydx]=g(x,y)','dydx=-1000.*(y-x)+2001')

-->Dx=0.0005;x0=0;xn=0.01;y0=1;x1=[x0:Dx:xn];y1=ode('stiff',y0,x0,x1,g);

-->Dx=0.0010;x0=0;xn=0.01;y0=1;x2=[x0:Dx:xn];y2=ode('stiff',y0,x0,x2,g);

-->Dx=0.0020;x0=0;xn=0.01;y0=1;x3=[x0:Dx:xn];y3=ode('stiff',y0,x0,x3,g);

-->Dx=0.0025;x0=0;xn=0.01;y0=1;x4=[x0:Dx:xn];y4=ode('stiff',y0,x0,x4,g);

-->min([y1 y2 y3 y4]), max([y1 y2 y3 y4])
 ans  = 1.
 ans  = 2.0099544

-->rect = [0 0 0.01 2.5];

-->plot2d(x1,y1,-1,'011',' ',rect)
-->plot2d(x2,y2,-2,'011',' ',rect)
-->plot2d(x3,y3,-3,'011',' ',rect)
-->plot2d(x4,y4,-4,'011',' ',rect)

-->xtitle('Stiff equation - ode function with option stiff','x','y')
```



Function *ode* with root finding option

Function *ode* can be invoked with the first argument *‘root’* if the solution of the ordinary differential equation *dy/dx = f(x,y)* is subject to the constraint *g(x,y) = 0*. The solution is calculated for a range of values of x (a vector) and stopped when the constraint *g(x,y)=0* is satisfied. For example, suppose that we are solving the differential equation dy/dx = f(x,y) =

x sin(y), with initial conditions y(0) = 1, subjected to the constraint g(x,y) = x + y - 5 = 0, in the range 0 < x < 10. We will obtain the solution by using:

```
-->deff('[dydx]=f(x,y)','dydx = x*sin(y)')

-->deff('[w]=g(x,y)','w = x+y-5')

-->y0=1;x0=0;x=[0:0.1:10];

-->[y,rd]=ode('root',y0,x0,x,f,1,g);
```

We can check that the solution was stopped before reaching the maximum value of x by checking the lengths of the vectors x and y:

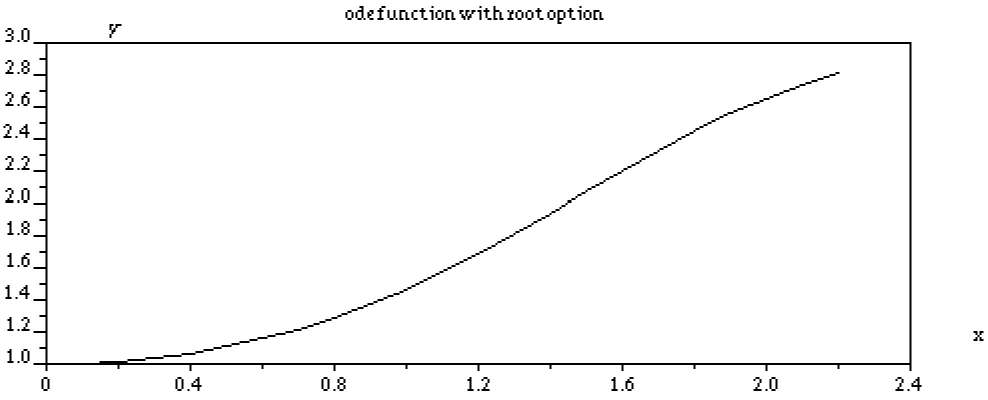
```
-->length(x), length(y)
 ans  =  101.

 ans  =   23.
```

Vector *rd* contains information on the point where the calculation was stopped.

```
-->rd
rd  =

!  2.1889184     1. !
```

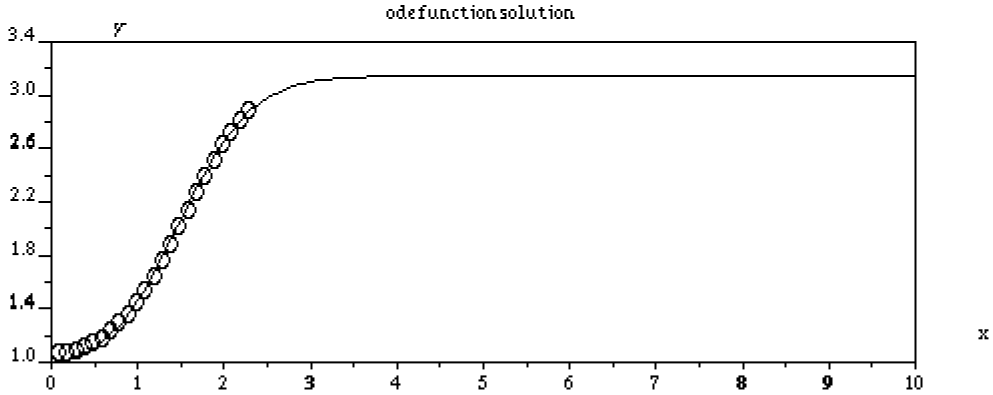
To plot the results of the recent call to function *ode* we create a new vector of x values, xx:

```
-->xx = x(1:23);

-->plot(xx,y,'x','y','ode function with root option')
```



The following plot shows the truncated solution altogether with the solution for the full x range (0,10):

```
-->yf = ode(y0,x0,x,f);

-->plot(x,yf,'x','y','ode function solution')

-->plot2d(xx,y,-9,'011',' ',[0 1 10 3.4])
```



Discrete solutions with function *ode*

Function *ode* provides the option *"discrete"* for the solution of a differential equation *dy/dx = f(x,y)* based on a set of discrete (integer) indices *kvect*, that starts with *k0*. The initial condition is *y(k0) = y0*. The simplest call to the function is

[y] = ode('discrete',y0,k0,kvect,f).

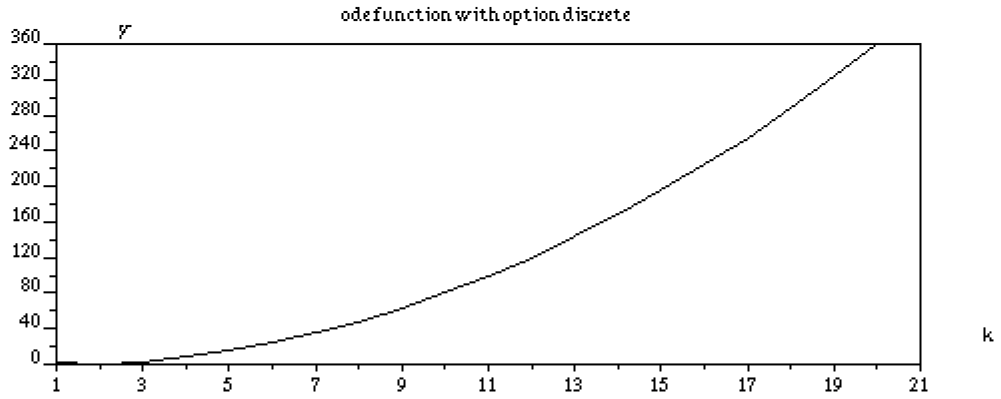
When using this option, the solution is calculated recursively by using $y_{k+1}=f(k,y_k)$, for *k=1,2,…*

As an example, we solve the differential equation dy/dx = x^2-1, in a discrete domain *k=1,2,…,20*, with y(1) = 2.5:

```
-->deff('[dydx]=f(x,y)','dydx=x.^2-1')

-->k0 = 1; kvect = [1:1:20];  y0 = 2.5;

-->y = ode('discrete',y0,k0,kvect,f);
```

The following plot shows the results of y as function of k:

```
-->plot(kvect,y,'k','y','ode function with option discrete')
```



Changing ODE numerical solution parameters with *odeoptions*

Function *ode* uses a number of parameters which can be redefined through the use of the function *odeoptions()*. When this command is used, SCILAB generates an input form where the parameters of the numerical methods for ordinary differential equations can be modified. Application of the function *odeoptions()* can be used to modify the global variable *%ODEOPTIONS*. This variable is a vector containing the following information:

[itask,tcrit,h0,hmax,hmin,jactyp,mxstep,maxordn,maxords,ixpr,ml,mu]

where the different elements are described as follows:

itask - identifies the task required from the numerical method. Possible values of *itask* are:

1 : normal computation at specified times
2 : computation at mesh points (given in first row of output of *ode*)
3 : calculate one step at one internal mesh point and return
4 : normal computation without overshooting *tcrit* (see below)
5 : one step, without passing *tcrit* (see below), and return

tcrit - this value applies when *itask = 4* or *5*, representing a critical value of the independent variable *t*.

h0 - represents the first step in the independent variable tried in the numerical solutions when adaptive methods are used.

hmax - maximum step size

hmin - minimum step size

jactype - Jacobian type. This option can be used when additional derivatives (a Jacobian) are provided for the solution. Possible values of this option are:

0 : functional iterations, no Jacobian used ("adams" or "stiff" only)
1 : user-supplied full Jacobian
2 : internally generated full Jacobian
3 : internally generated diagonal Jacobian ("adams" or "stiff" only)
4 : user-supplied banded Jacobian (see *ml* and *mu* below)

5 : internally generated banded Jacobian (see *ml* and *mu* below)

mxstep - Maximum number of iterations allowed.

maxordn - Maximum non-stiff order allowed for the numerical method (maximum allowed is 12).

maxords - Maximum stiff order allowed, at most 5-th order.

ixpr - print level, either 0 or 1.

ml,mu - see description below:

If *jactype* = 4 or 5, *ml* and *mu* are the lower and upper half-bandwidths of the banded Jacobian. The band includes those terms of the Jacobian, J_{ij}, with i-ml <= j <= ny-1.

If *jactype = 4* the Jacobian function must return a matrix *J* which has (*ml+mu+1*) rows and *ny* columns (where *ny* is the number of elements of *y* in *dy/dt =f(t,y)*), such that the first column of *J* is made of *mu* zeros followed by $\partial f_1/\partial y_1$, $\partial f_2/\partial y_1$, $\partial f_3/\partial y_1$, ... (*1+ml* possibly non-zero entries). Column 2 of *J* is made of *mu-1* zeros followed by $\partial f_1/\partial x_2$, $\partial f_2/\partial x_2$, etc

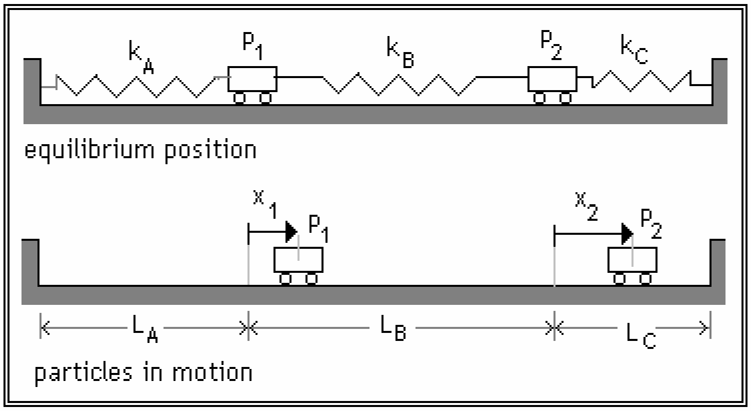
The default value of the global variable *%ODEOPTIONS* is *[1,0,0,%inf,0,2,500,12,5,0,-1,-1]*, i.e., *itask = 1, tcrit = 0, h0 = 0, hmax = , hmin = 0, jactype = 2, mxstep = 500, maxordn = 12, maxords = 5, ixpr = 0, ml = -1, mu = -1.*

Applications of numerical solutions to IVPs

This section presents the solution to initial value problems (IVPs) some of which are obtained from the physical sciences, e.g., mechanical systems, electric circuits, etc.

Systems of ODEs from mechanical systems

Systems of ODEs can obtained from the analysis of two linked particles in oscillatory motion or from the analysis of electric circuits. For example, the figure below shows two particles, P_1 and P_2, linked by three springs. The figure at the top represents the system in their state of equilibrium, while the one at the bottom shows the system at any generic point at time t>0.



The displacement of particle P_1 with respect to its equilibrium position is given by $x_1(t)$, while that of particle P_2 is given by $x_2(t)$. The magnitude of the force applied by a spring on an attached particle is given by Hooke's law,

$$F = -\mathrm{k}(L - L_0)\ ,$$

where k is the spring constant, L is the stretched length of the spring, and L_0 is the unstretched length of the spring. The force that spring A, with constant k_A, applies on particle P_1 is given by

$$F_A = k_A(L_A + x_1 - L_A) = k_A x_1\ ,$$

while that applied by spring B on particle P_1 is

$$F_B = k_B(L_A + L_B + L_C - L_A + x_1 - L_C - x_2 - L_B) = k_B(x_1 - x_2)$$

For the position illustrated in the Figure above, both forces will act in the negative direction of x_1, thus, writting Newton's second law for particle P_1 of mass m_1 results in:

$$-F_A - F_B = m_1 a_1\ , \qquad \text{or} \qquad -k_A x_1 - k_B(x_1 - x_2) = m_1 \frac{d^2 x_1}{dt^2}$$

Similarly, for particle P_2, of mass m_2, with the magnitude of the force applied by spring C given by

$$F_C = |k_C(L_C - x_2 - L_C)| = k_C x_2\ ,$$

we can write

$$F_B - F_C = m_2 a_2\ , \qquad \text{or} \qquad k_B(x_1 - x_2) - k_C x_2 = m_2 \frac{d^2 x_2}{dt^2}$$

The resulting system of equations can be written as:

$$\frac{d^2 x_1}{dt^2} = -\frac{k_A + k_B}{m_1} x_1 + \frac{k_B}{m_1} x_2\ , \qquad \text{and} \qquad \frac{d^2 x_2}{dt^2} = \frac{k_B}{m_2} x_1 - \frac{k_B + k_C}{m_2} x_2\ .$$

These two second-order equations can generate a system of four first order equations if we define $x_3 = dx_1/dt$, and $x_4 = dx_2/dt$. The resulting system is:

$$dx_1/dt = x_3$$

$$dx_2/dt = x_4$$

$$dx_3/dt = -(k_A+k_B)x_1/m_1 + k_Bx_2/m_1$$

$$dx_4/dt = k_Bx_1/m_2 - (k_B+k_C)x_2/m_2$$

or, using vectors, $dx/dt = f(t,x)$, where $x = [x_1\ x_2\ x_3\ x_4]^T$, and

$$f(t,x) = \begin{bmatrix} x_3 \\ x_4 \\ -\frac{k_A+k_B}{m_1}x_1 + \frac{k_B}{m_1}x_2 \\ \frac{k_B}{m_2}x_1 - \frac{k_B+k_C}{m_2}x_2 \end{bmatrix}.$$

Suppose we use the following values $k_A = 20$ *N/m*, $k_B = 40$ *N/m*, $k_C = 50$ *N/m*, $m_1 = 2$ *kg*, $m_2 = 5$ *kg*, and given the initial conditions $x_1 = 0.5$ *m*, $x_2 = 0.2$ *m*, $x_3 = dx_1/dt = -0.1$ *m/s*, $x_4 = dx_2/dt = 0.1$ *m/s*, *at* $t = 0$, we will solve the system of equations in the interval $0 < t < 100$ s, using a time increment of $\Delta t = 0.1$. This is the SCILAB solution:

First, we define the function *f* from $dy/dt = f(t,y)$ and enter the values of the parameters:

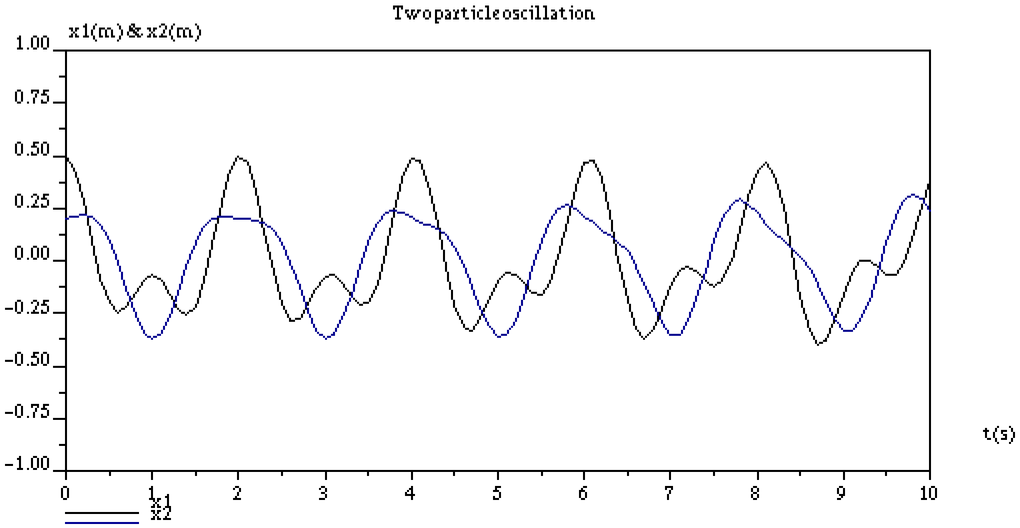
```
-->deff('[ff]=f(t,x)','ff=[x(3);x(4);-(kA+kB).*x(1)/m1+kB.*x(2)/m1;...
-->kB.*x(1)/m2-(kB+kC).*x(2)/m2]')

-->kA = 20; kB = 40; kC = 50; m1 = 2; m2 = 5;
```

Next, we calculate the parameters for the solution and the solution itself:

```
-->t0=0;Dt=0.1;tn=10;x0=[0.5;0.2;-0.1;0.1];t=[t0:Dt:tn];

-->x = ode(x0,t0,t,f);
```

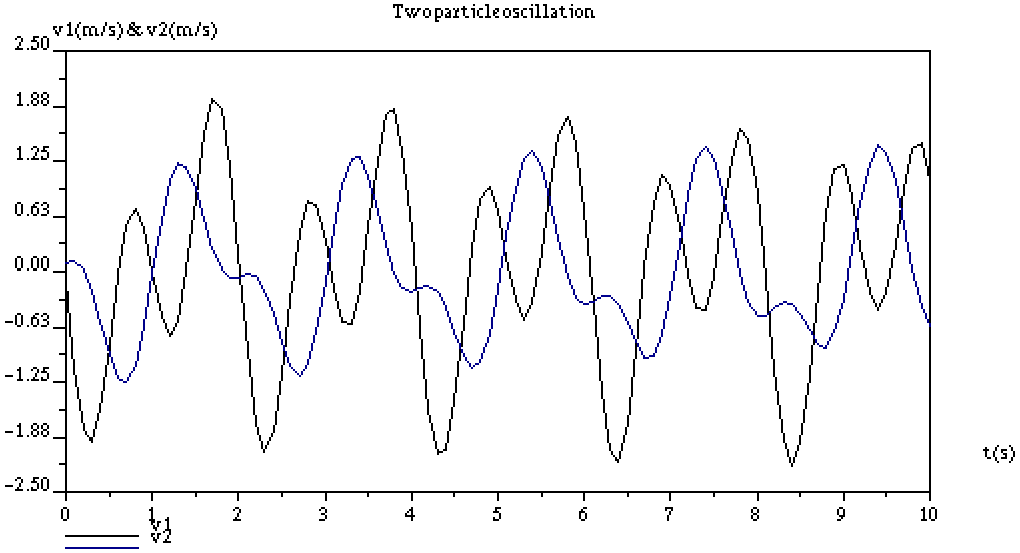
To produce a plot of the positions $x_1(t)$ and $x_2(t)$ we use:

```
-->plot2d([t',t'],[x(1,:)',x(2,:)'],[1,9],'111','x1@x2 ',[0 -1 10 1])

-->xtitle('Two particle oscillation','t(s)','x1(m) & x2(m)')
```



The corresponding velocities, $v_1(t)$ and $v_2(t)$, are plotted by using:

```
-->plot2d([t',t'],[x(3,:)',x(4,:)'],[1,9],'111','v1@v2 ',[0 -2.5 10 2.5])
-->xtitle('Two particle oscillation','t(s)','v1(m/s) & v2(m/s)')
```

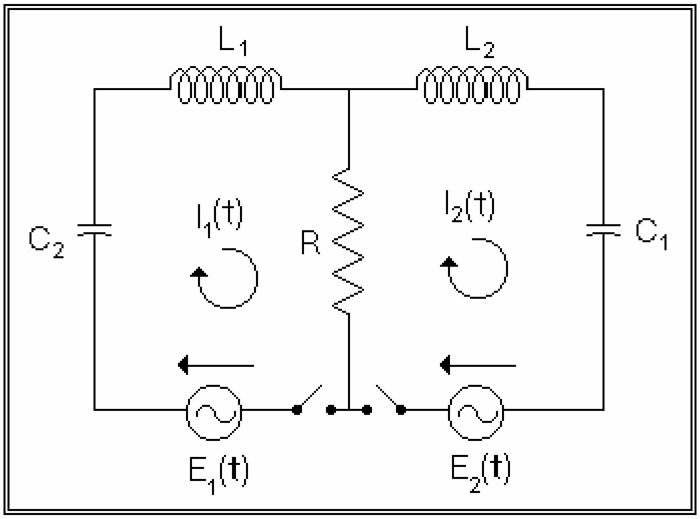


System of ODEs from Electric Circuits

A second application of systems of ordinary differential equations can be obtained from electric circuits. Consider the circuit shown in the figure below. The circuit consists of two loops each with a capacitor, an inductor, and a voltage source. The two loops share the resistor R. The electrical current circulating in the left-hand side loop is referred to as $I_1(t)$ and it is assumed to be positive in the counterclockwise direction. A similar electric current, $I_2(t)$, circulates in the right-hand side loop. The voltages $E_1(t)$ and $E_2(t)$ are positive in the sense indicated by the arrow. The equations for the voltage across individual components are:

- Resistor, $V_R = R\,I$
- Capacitor, $V_C = \frac{q}{C}$
- Inductor, $V_L = \frac{L\,dI}{dt}$

where V = voltage (volts), R = resistance (ohms), I = electric current (amperes), q = electric charge (coulombs), C = capacitance (farads), L = inductance (henrys), t = time (seconds). By definition, $I = \frac{dq}{dt}$.



For the purpose of writing the governing equations we use Kirchoff law of conservation of voltage in a closed loop in a circuit with voltages across resistors, capacitors, and inductors decreasing in the direction of circulation of the current. Thus, for loops number 1 and 2, we would write, respectively:

$$E_1 - \frac{q_1}{C_1} - \frac{L_1\,dI_1(t)}{dt} - R\,(I_1 - I_2) = 0\ ,$$

and

$$E_2 - R\,(I_2 - I_1) - \frac{L_2\,dI_2}{dt} - \frac{q_2}{C_2} = 0\ .$$

Introducing the definition of the electric currents in the equations, $I_1 = \frac{dq_1}{dt}$, and $I_2 = \frac{dq_2}{dt}$, and rearranging terms we get a system of four first-order differential equations:

$$\frac{dI_1}{dt} = -\frac{R\,I_1}{L_1} + \frac{R\,I_2}{L_1} - \frac{q_1}{L_1\,C_1} + \frac{E_1}{L_1}\quad,\quad \frac{dq_1}{dt} = I_1\ ,\quad \frac{dI_2}{dt} = \frac{R\,I_1}{L_2} - \frac{R\,I_2}{L_2} - \frac{q_2}{L_2\,C_2} + \frac{E_2}{L_2}\quad,\quad \frac{dq_2}{dt} = I_2\ .$$

These four equations can be transformed into a system of differential equations *dy/dt = f(t,y),* with y = $[y_1\ y_2\ y_3\ y_4]^T = [q_1\ q_2\ I_1\ I_2]^T$, and

$$f(t,y)=\begin{bmatrix} y_3 \\ y_4 \\ -\frac{Ry_3}{L_1}+\frac{Ry_4}{L_1}-\frac{y_1}{L_1C_1}+\frac{E_1}{L_1} \\ \frac{Ry_3}{L_2}-\frac{Ry_4}{L_2}-\frac{y_2}{L_2C_2}+\frac{E_2}{L_2} \end{bmatrix}.$$

Suppose that we use the values *R = 1800 ohms*, *L_1 = 500 henrys*, *L_2 = 800 henrys*, *C_1 = 0.005 farads*, *C_2 = 0.010 farads*, *E_1 = E_2 = 0*, with initial conditions *q_1 = 100 coulombs*, *q_2 = I_1 = I_2 = 0*, the following two example illustrate how to produce the solution for the charges *$q_1(t)$*, *$q_2(t)$*, and the currents *$I_1(t)$*, *$I_2(t)$*.

The first example uses constant voltages E_1 and E_2. First we define the system function *f(t,y)* and the parameters of the circuit:

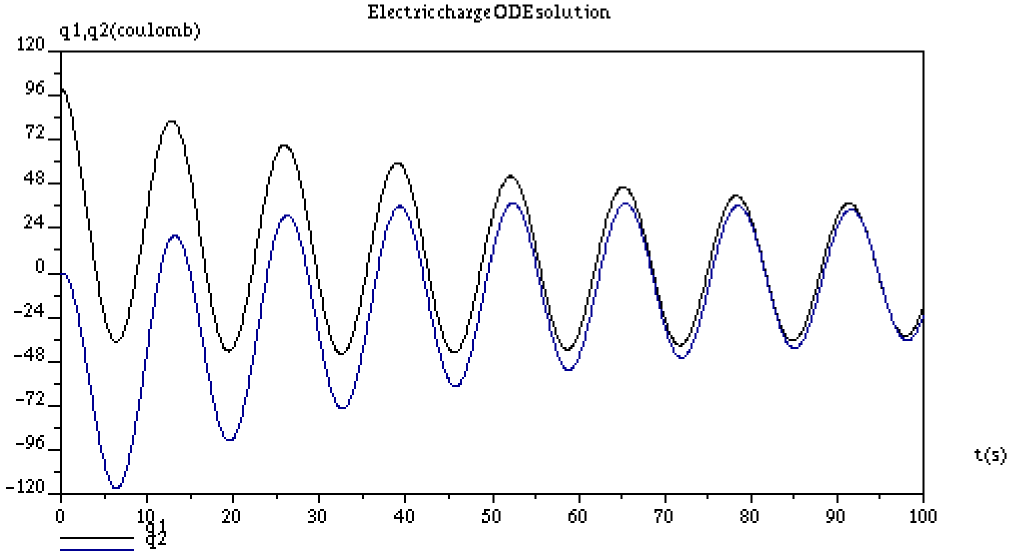
```
-->deff('[dydt]=f(t,y)','dydt = [y(3);y(4);...
-->-R*y(3)/L1+R*y(4)/L1-y(1)/(L1*C1)+E1/L1;...
--> R*y(3)/L2-R*y(4)/L2-y(2)/(L2*C2)+E2/L2]')

-->R=1800;L1=500;L2=800;C1=0.005;C2=0.010;E1=0;E2=0;
```

Next, we define the parameters of the solution and calculate the solution using function ode:

```
-->t0 = 0; Dt = 0.1; tn = 10; t = [t0:Dt:tn];y0 = [100;0;0;0];
-->y = ode(y0,t0,t,f);
```

To produce plots of electric charge we use:

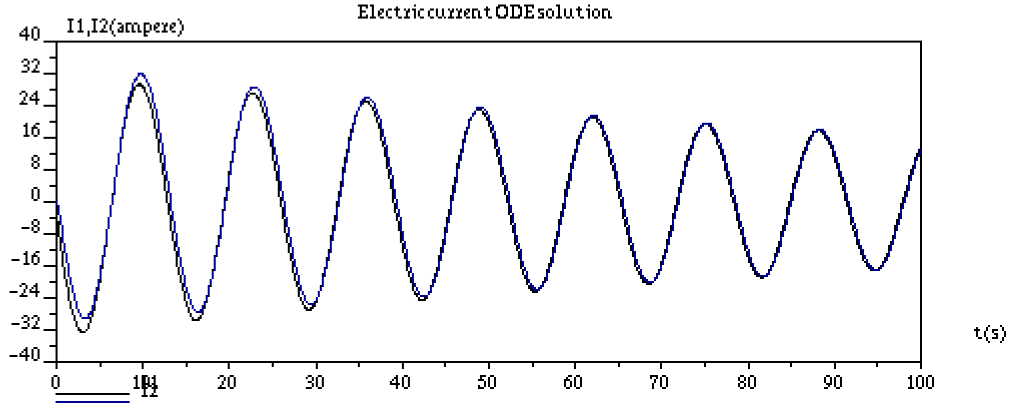
```
-->min([y(1,:) y(2,:)]), max([y(1,:) y(2,:)])
 ans  = - 117.01482, ans  =   100.
-->plot2d([t',t'],[y(1,:)',y(2,:)'],[1,9],'111','q1@q2',[0 -120 100 120])
-->xtitle('Electric charge ODE solution','t(s)','q1,q2(coulomb)')
```



A plot of electric current follows:

```
-->min([y(3,:) y(4,:)]), max([y(3,:) y(4,:)])
```

```
 ans  = - 32.655809
 ans  =   31.88677

-->plot2d([t',t'],[y(3,:)',y(4,:)'],[1,9],'111','I1@I2',[0 -40 100 40])
-->xtitle('Electric current ODE solution','t(s)','I1,I2(ampere)')
```



In the following example we introduce sinusoidal driving voltages, $E_1(t)$ and $E_2(t)$:

```
-->deff('[EE1]=E1(t)','EE1=12*cos(120*%pi*t)')

-->deff('[EE2]=E2(t)','EE2= 6*cos( 60*%pi*t)')
```

Next, we redefine the function f(t,y) to include $E_1(t)$ and $E_2(t)$:

```
-->deff('[dydt]=f(t,y)','dydt = [y(3);y(4);...
-->-R*y(3)/L1+R*y(4)/L1-y(1)/(L1*C1)+E1(t)/L1;...
--> R*y(3)/L2-R*y(4)/L2-y(2)/(L2*C2)+E2(t)/L2]')
```

The parameters for the solution will be now:

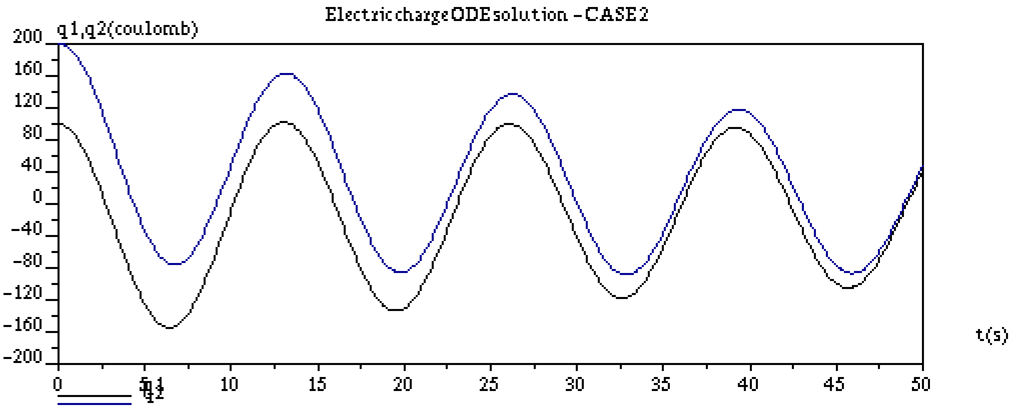
```
-->t0 = 0; Dt = 0.1; tn = 50; t = [t0:Dt:tn];
```

Because the system function f(t,y) is now complicated by the presence of time-dependent voltages E1(t) and E2(t), function ode will take a few minutes to complete the solution.

```
-->y = ode(y0,t0,t,f);
```

To produce a plot of electric charges use:

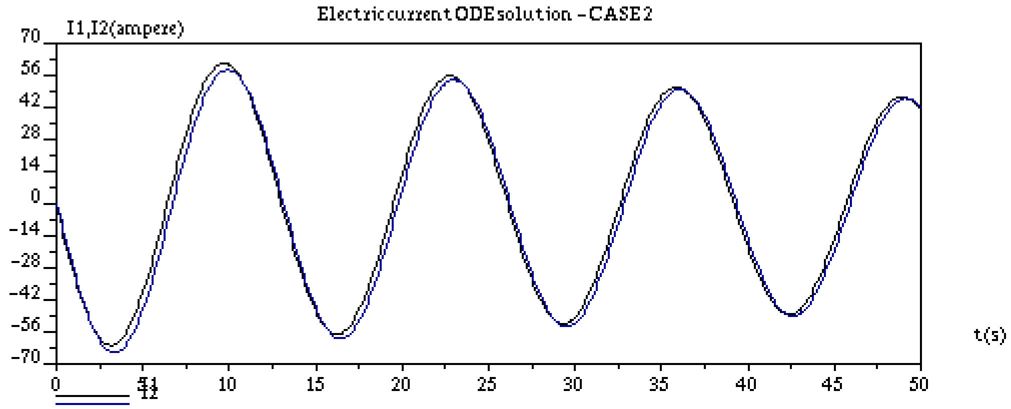
```
-->min([y(1,:) y(2,:)]), max([y(1,:) y(2,:)])
 ans  = - 154.45894
 ans  =   200.

-->plot2d([t',t'],[y(1,:)',y(2,:)'],[1,9],'111','q1@q2',[0 -200 50 200])

-->xtitle('Electric charge ODE solution - CASE 2','t(s)','q1,q2(coulomb)')
```



For a plot of the electric currents use:

```
-->min([y(3,:) y(4,:)]), max([y(3,:) y(4,:)])
 ans  = - 65.0047
 ans  =   61.126823
```

```
-->plot2d([t',t'],[y(3,:)',y(4,:)'],[1,9],'111','I1@I2',[0 -70 50 70])
-->xtitle('Electric current ODE solution - CASE 2','t(s)','I1,I2(ampere)')
```



Solving a fourth-order equation

Consider the fourth-order linear equation presented in an earlier section

$$\frac{d^4 y}{dx^4} + \frac{3\,d^3 y}{dx^3} - \frac{2\,d^2 y}{dx^2} + \frac{5\,dy}{dx} + y = 0\ ,$$

subjected to $y = 1$, $\frac{dy}{dx} = \text{-}1$, $\frac{d^2 y}{dx^2} = 0$, $\frac{d^3 y}{dx^3} = \text{-}1$, at $x = 0$. To solve this equation we transform the fourth-order equation into a first-order system of linear equations:

$$du_3/dx = -3u_3(x)+2u_2(x)-5u_1(x)-y(x)+x^2/2,$$

$du_2/dx = u_3(x),$

$du_1/dx = u_2(x),$

$dy/dx = u_1(x),$

or

$$\frac{d}{dx}\begin{bmatrix} u_3(x) \\ u_2(x) \\ u_1(x) \\ y(x) \end{bmatrix} = \begin{bmatrix} -3 & 2 & -5 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_3(x) \\ u_2(x) \\ u_1(x) \\ y(x) \end{bmatrix} + \begin{bmatrix} x^2/2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

With

$$v(x) = \begin{bmatrix} u_3(x) \\ u_2(x) \\ u_1(x) \\ y(x) \end{bmatrix}, \quad A = \begin{bmatrix} -3 & 2 & -5 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad g(x) = \begin{bmatrix} x^2/2 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

the system of differential equations is written as

$$dv/dx = Av+g(x).$$

The initial conditions are $y(0) = 1,\ u_1(0) = dy/dx = -1,\ u_2(0) = du_1/dx = d^2y/dx^2 = 0,\ u_3(0) = du_2/dx = d^2u_1/dx^2 = d^3y/dx^3 = -1,$

$$u(0) = \begin{bmatrix} -1 \\ 0 \\ -1 \\ 1 \end{bmatrix}.$$

To implement the solution using SCILAB we first define the function f(x,v) and the parameters for the solution

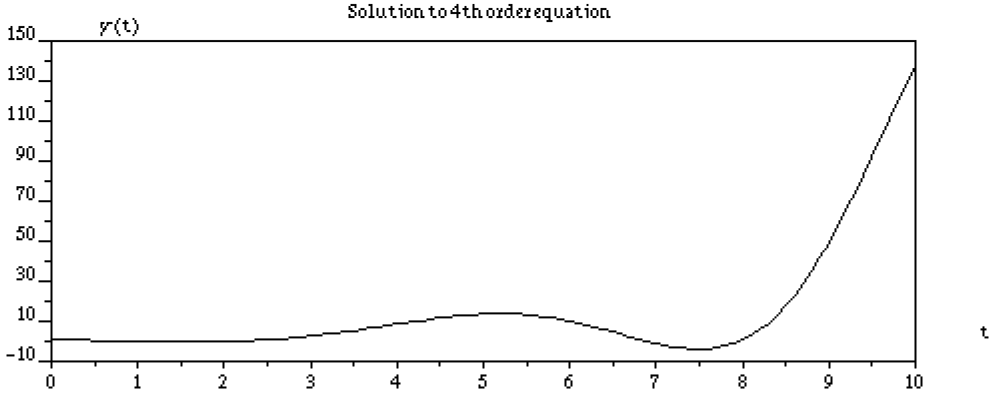
```
-->deff('[dvdx]=f(x,v)','dvdx=A*[v(1);v(2);v(3);v(4)]+[x^2;0;0;0]')
-->A = [-3,2,-5,-1;1,0,0,0;0,1,0,0;0,0,1,0];

-->v0 = [-1;0;-1;1];t0=0;Dt=0.1;tn=10;t=[t0:Dt:tn];
```

The numerical solution to the system of differential equations is obtained through:

```
-->v = ode(v0,t0,t,f);
```

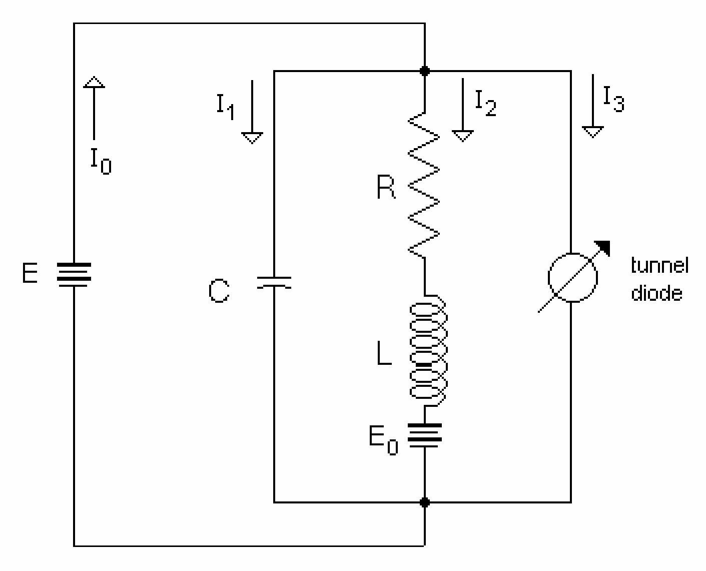
To produce a plot of the solution y(t) we use:

```
-->min(v(4,:)),max(v(4,:))
 ans  =  - 3.903516
 ans  = 137.7293

-->plot(t,v(4,:),'t','y(t)','Solution to 4th order equation')
```



The Van der Pol equation

The Van der Pol equation results from the analysis of the Van der Pol oscillator circuit shown in the figure below.



Application of Kirchoff's laws to this circuit results in the following equations:

$$C(dE/dt)=I_1,\ RI_2 + L(dI_2/dt) = E + E_0,\ I_3 = f(E),\ I_0 = I_1 + I_2 + I_3$$

This set of four equations can be reduced to two differential equations

$$C(dE/dt) = I_0 - I_2 - f(E)$$
$$L(dI_2/dt) = E + E_0 - RI_2$$

If we eliminate the resistor from the circuit ($R=0$), the two differential equations can be combined into a single second order differential equation

$$C(d^2E/dt^2) + f'(E)(dE/dt) + (E+E_0)/L = dI_o/dt.$$

The function $f(E)$ represents the response of the tunnel diode to the voltage E, and can be taken to be the third-order polynomial

$$f(E) = KE(E^2/3 - (E_1+E_2)E/2 + E_1E_2).$$

Tuning the voltage source E so that $E_0 = -(E_1+E_2)/2,$ we can write

$$f(E) = KE(E^2/3 + E_0E/2 + E_1E_2).$$

If we define $E = (E_2-E_1)y/2 + (E_1+E_2)/2 = (E_2-E_1)y/2 + E_0$, and scale the time so that $\tau = t(LC)^{1/2}$, we can transform the governing equation into the Van der Pol equation

$$d^2y/d\tau^2 + \kappa(y^2-1)(dy/d\tau) + y = 0,$$

where $\kappa = K((E_2-E_1)/2)^2\ (L/C)^{1/2}$ and $dI_0/dt = 0$.

The Van der Pol equation, being a second-order equation, can be transformed into a set of two first-order equations, by using $dy/d\tau = u_1(\tau)$, and $y = u_2(\tau)$. The system of equations is

$$du_1/d\tau = -\kappa(u_2^2-1)u_1 - u_2$$
$$du_2/d\tau = u_1$$

or,

$$du/dt = f(\tau,u),$$

with

$$u(\tau) = \begin{bmatrix} u_1(\tau) \\ u_2(\tau) \end{bmatrix}, \quad f(\tau,u) = \begin{bmatrix} -\kappa(u_2^2-1)u_1 - u_2 \\ u_1 \end{bmatrix}.$$

The Van der Pol equation is solved next using SCILAB's function *ode*. The initial conditions used are $u_1(\tau) = du_2/d\tau = dy/d\tau = 2,\ u_2(\tau) = y = 1$ at $\tau = 0$. The solution is obtained for the range $0 < \tau < 100$, and for values of $\kappa = 0.01$ and $\kappa = 4.0$. First, we define the function $f(\tau,u)$:

```
-->deff('[ff]=f(t,u)','ff=[-k*(u(2)^2-1)*u(1)-u(2);u(1)]')
-->u0=[1;2];t0=0;Dt=0.1;tn=100.0;t=[t0:Dt:tn];
```

The solution for k = 0.01 is found first:

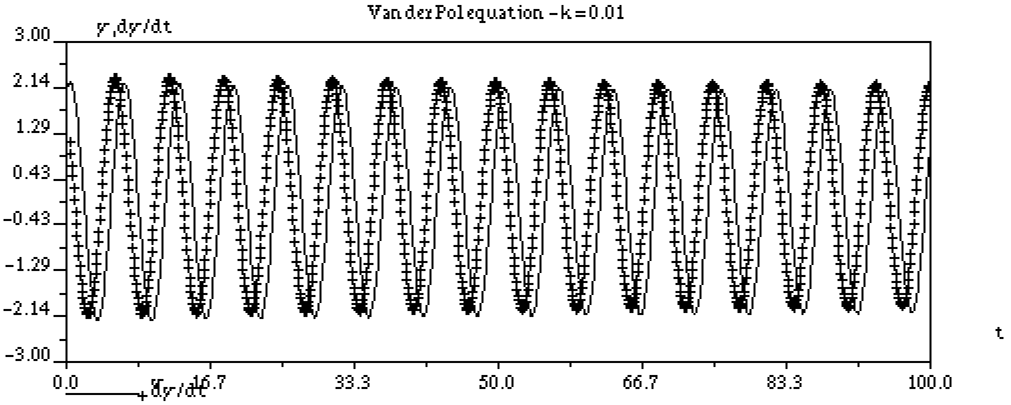
```
-->k=0.01;u1=ode(u0,t0,t,f);
```

To produce plots of the function we determine the maximum and minimum values of u:

```
-->min(u1),max(u1)
 ans  =  - 2.2268766
 ans  =    2.231914
```

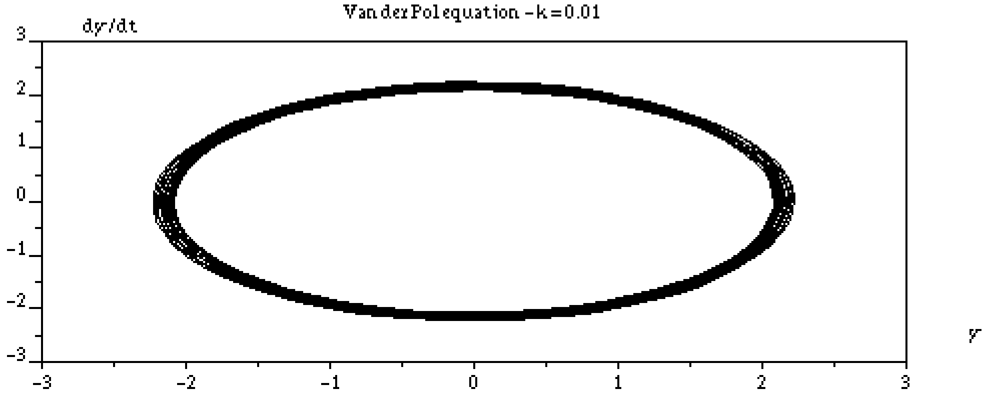
First, we plot the signals $u_2(\tau) = y$, and $u_1(\tau) = dy/d\tau$ against τ:

```
-->plot2d([t' t'],[u1(2,:)' u1(1,:)'],[1,-1],'111','y@dy/dt',[0 -3 100 3])
-->xtitle('Van der Pol equation - k = 0.01','t','y,dy/dt')
```



The phase portrait *dy/dτ* vs. *y* is shown next:

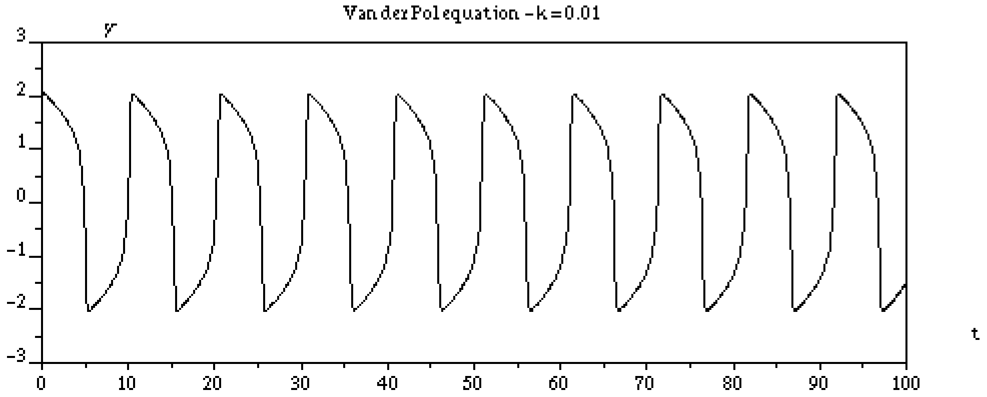
```
-->plot(u1(1,:),u1(2,:),'y','dy/dt','Van der Pol equation - k = 0.01')
```



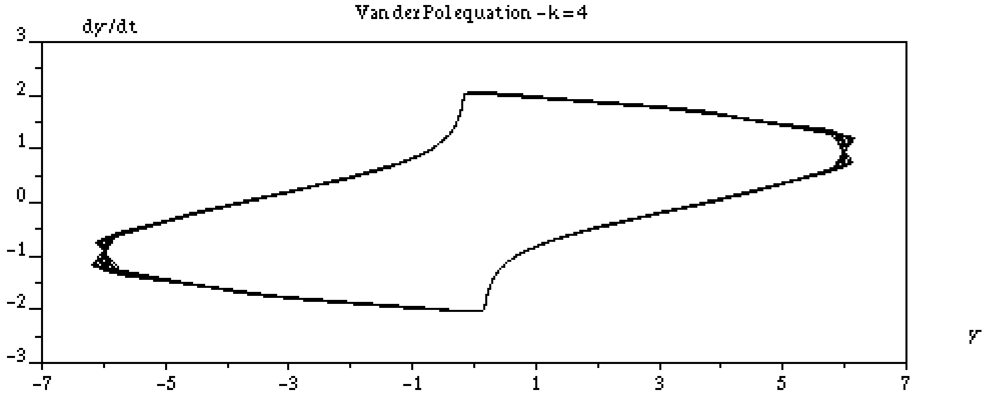
The solution to the Van der Pol equation for $\kappa = 4$, and the corresponding plots are obtained with the following SCILAB commands:

```
-->k=4.00;u2=ode(u0,t0,t,f);
```

```
-->plot(t,u2(2,:),'t','y','Van der Pol equation - k = 4')  //Signal y vs. t
```



```
-->plot(u2(1,:),u2(2,:),'y','dy/dt','Van der Pol equation - k = 4')   //Phase
portrait
```



The Rössler flow

In the analysis of chaotic dynamical systems, a three-dimensional flow is described by a set of three ordinary differential equations involving three variables *x(t), y(t),* and *z(t).* The Rössler flow is given by the equations (see Bergé et al., 1984, "*Order within Chaos - Towards a deterministic approach to turbulence*," John Wiley & Sons, New York):

$$dx/dt = - y - z$$
$$dy/dt = \quad x + ay$$
$$dz/dt = b + xz - cz$$

To solve this system using SCILAB function *ode* we re-write the system as

$$du/dt = f(t,u)$$

with

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \quad g(t,x) = \begin{bmatrix} -u_2 - u_3 \\ u_1 + au_2 \\ b + u_1 u_3 - cu_3 \end{bmatrix}.$$

In the following exercise we solve for the Rössler flow using *a = b = 0.2, c = 5.7*, in the interval *0 < t < 200*, with initial conditions *x(0) = y(0) = z(0) = 1*. We use a time increment *Δt = 0.1*. The solution starts by defining the function *f(t,u)* and the parameters of the flow and of the solution:

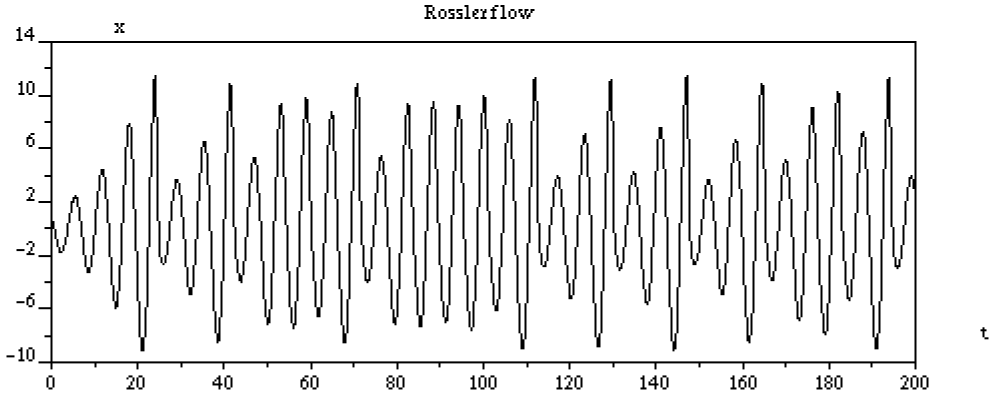
```
-->deff('[w]=f(t,u)','w=[-u(2)-u(3);u(1)+a*u(2);b+u(1)*u(3)-c*u(3)]')
-->a=0.2; b=0.2; c=5.7; t0=0; Dt=0.1; tn=200; t=[t0:Dt:tn]; u0=[1;1;1];
```

The solution is stored in variable u from:

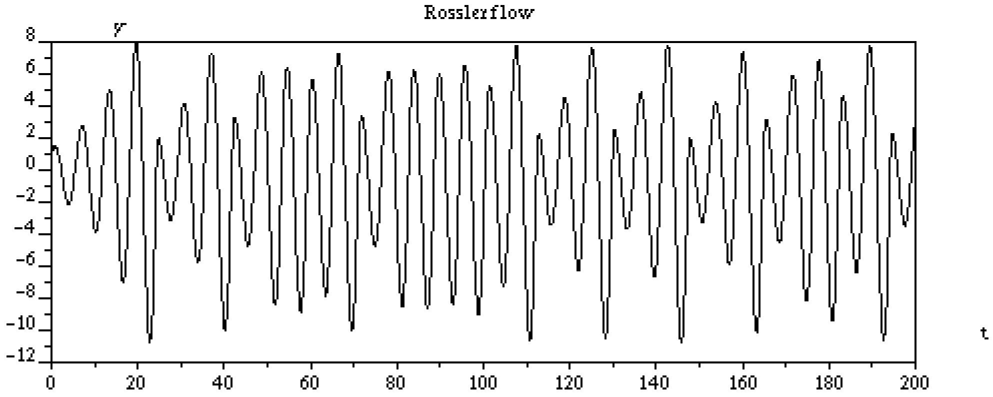
```
 -->u=ode(u0,t0,t,f);
```

The following are plots of the signals *x(t), y(t),* and *z(t)* resulting from the solution:

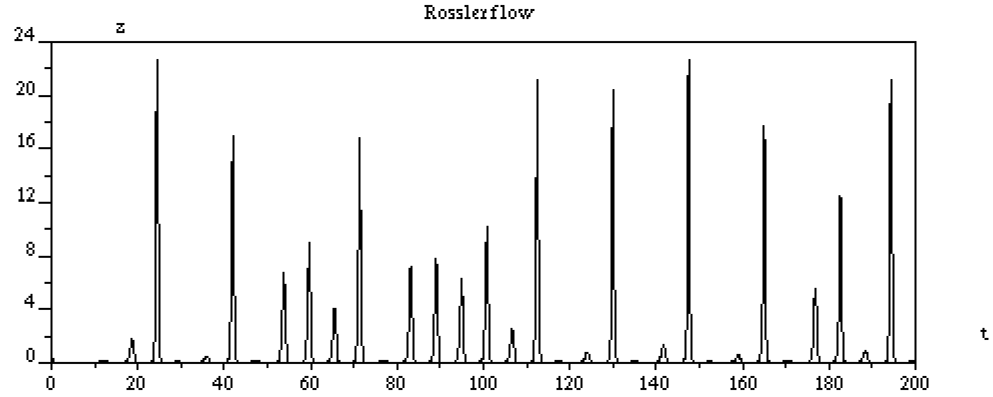
```
-->plot(t,u(1,:),'t','x','Rossler flow')
```



```
-->plot(t,u(2,:),'t','y','Rossler flow')
```

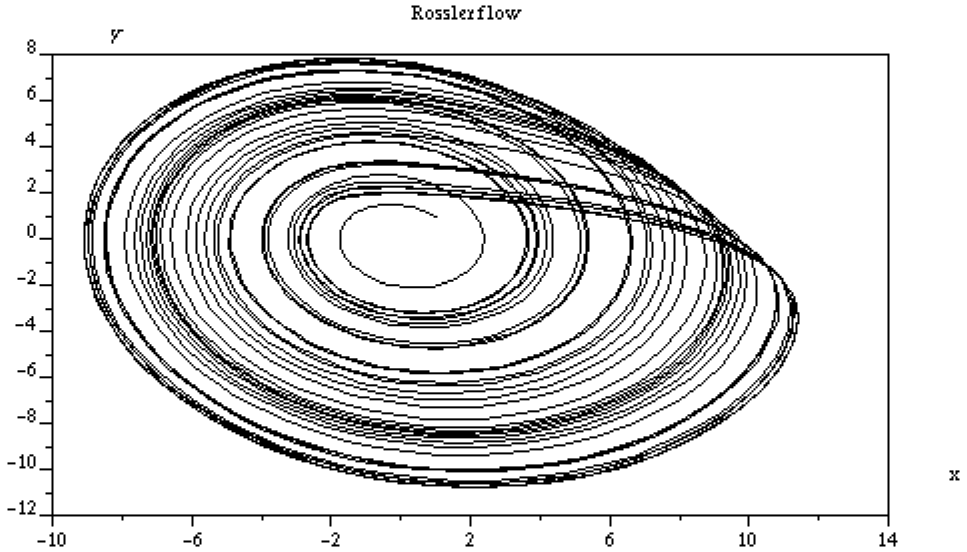


```
-->plot(t,u(3,:),'t','z','Rossler flow')
```



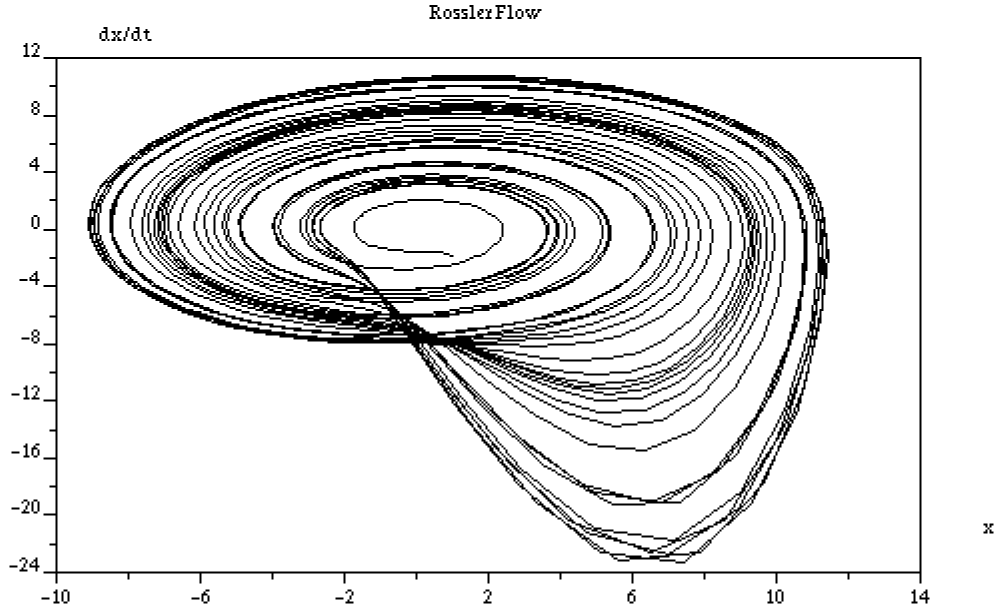
The behavior of the signals is, in general, aperiodic or chaotic. Phase portraits of signals may provide additional information regarding strange attractors resulting from the solution. A phase portrait of signal y vs. x follows:

```
-->plot(u(1,:),u(2,:),'x','y','Rossler flow')
```



A phase portrait of the rate of change in *x*, *dx/d*t, vs. *x*, is obtained next. First, we determine the length of vector *x = u(1,:)*, and use function *mtlb_diff* (see Chapter …) to estimate the derivative *dx/dt*. The next step is to produce a vector *xx* consisting of the data in *u(1,:)* but reduced by one to make it of the same length as *dx*. The phase portrait is shown below:

```
-->n=length(u(1,:))
 n  = 2001.
-->dx = mtlb_diff(u(1,:))/Dt;
-->xx=u(1,1:n-1);
-->plot(xx,dx,'x','dx/dt','Rossler Flow')
```

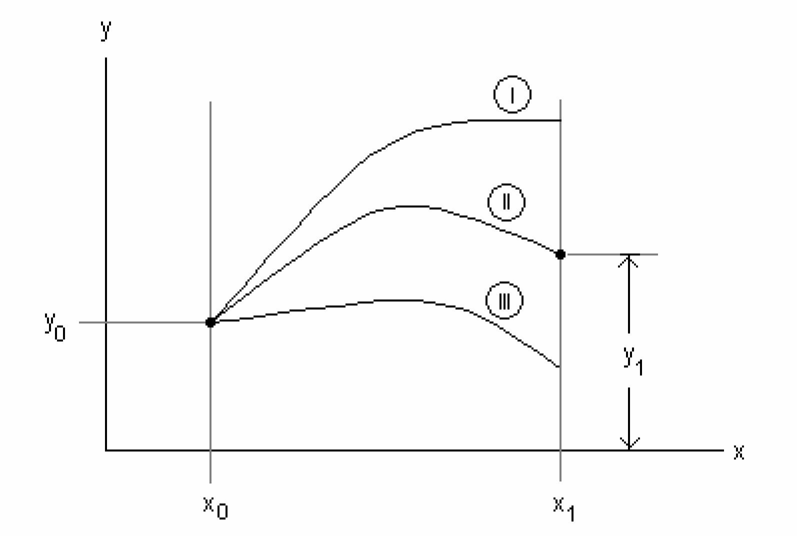


Solutions to boundary value problems (BVPs)

Boundary value problems consist of ordinary differential equations with conditions provided at different values of the independent variable. Unlike initial boundary problems, which can be solved using the same numerical solution after transforming the differential equation into a system of first-order differential equations with appropriate initial conditions, boundary value problems are not suitable for solution through a single numerical method. In this section, we explore some approaches to the solution of boundary value problems.

The shooting method

The simplest type of boundary value problems consists of a second order differential equation, $d^2y/dx^2 = g(x,y,dy/dx)$ subject to the boundary conditions $y(x_0) = y_0$ and $y(x_1) = y_1$. The figure below illustrate three possible solutions to the initial value problem represented by the same differential equation, namely, $d^2y/dx^2 = g(x,y,dy/dx)$, subject to the initial condition $y(x_0) = y_0$, and different initial derivative conditions. There is one value of dy/dx at $x = x_0$ that produces the solution that satisfies the second boundary condition $y(x_1) = y_1$. The solution to the boundary value problem will be curve (II) in the figure.



The so-called "shooting" method consists in solving the initial value problem $d^2y/dx^2 = g(x,y,dy/dx)$, subject to the initial condition $y(x_0) = y_0$, $dy/dx|_{x = x0} = y'(x_0) = y_0'$, for different values of y_0', and obtaining the corresponding boundary values $y_1 = y(x_1)$. As a result we get a set of data values { $[(y_0')_1,(y_1)_1]$, $[(y_0')_2,(y_1)_2]$, …, $[(y_0')_n,(y_1)_n]$ } from which we can interpolate the value of y_0' corresponding to the given value of y_1. The solution to the boundary value problem, thus, becomes the solution to a number of initial value problems followed by an interpolation. Once the proper value of y_0' is determined, the initial value problem is solved one last time.

A function to implement the shooting method

The following SCILAB user-defined function, *shooting*, produces the solution to a second-order boundary value problem given the boundary conditions $y(x_0) = y_0$ and $y(x_1) = y_1$. The second-order differential equation $d^2y/dx^2 = g(x,y,dy/dx)$ is transformed into a first-order system of two ordinary differential equations of the form $du/dx = f(x,u)$, with $u_1 = y(x)$, $u_2 = du_1/dx = dy/dx$, and

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad f(x,u) = \begin{bmatrix} u_2 \\ g(x,u_1,u_2) \end{bmatrix}.$$

The general call to the function is

$$[u,\ table,\ y0p] = shooting(yb,yp,x,f)$$

The boundary conditions are passed on to the function as a vector $yb = [y_0\ y_1]$. The solution is obtained on a range represented by vector *x*. We also need to provide a vector of values of the derivative y_0', referred to as *yp*. The function returns the values of *u* as a $2\times n$ matrix, where the first row is the function y(x) and the second row is the derivative dy/dx corresponding to the values of x. The function *shooting* also returns variable *table* which is a table of values of y_0' and the corresponding values of $y(x_1)$ that were used to interpolate the initial condition for the derivative at $x = x_0$ for the final solution. The first row in the table are the values of y_0', while the second row in the table are the values of $y(x_1)$. In addition to matrices *u* and *table*, function *shooting* returns also the derivative boundary condition, $y0p = dy/dx|_{x=x0}$, interpolated from the shooting method.

A listing of the function is shown next:

```
function [y,xyTable,yderiv] = shooting(yb,yp,x,f)

//Shooting method for a second order
//boundary value problem
//yb = [y0 y1] -> boundary conditions
//x = a vector showing the range of x
//f = function defining ODE, i.e.,
//    dy/dx = f(x,y), y = [y(1);y(2)].
//yp = vector with range of dy/dx at x=x0
//xyTable = table for interpolating derivatives
//yderiv  = derivative boundary condition

n  = length(yp);
m  = length(x);
y1 = zeros(yp);

for j = 1:n
       y0    = [yb(1);yp(j)];
       yy    = ode(y0,x(1),x,f);
       y1(j) = yy(1,m);
end;

xyTable = [y1;yp];
yderiv  = interpln(xyTable,yb(2));
y0      = [yb(1);yderiv];
y       = ode(y0,x(1),x,f);
```

Consider the case of the second order boundary value problem defined by the ordinary differential equation

$$d^2y/dx^2 + dy/dx + y = sin(3x),$$

subject to the boundary conditions $y(0) = 1$, $y(5) = -1$. We re-cast the ODE into the following first-order system,

$$du/dx = f(x,u),$$

with

$$u_1 = y(x),\ u_2 = du_1/dx = dy/dx,$$

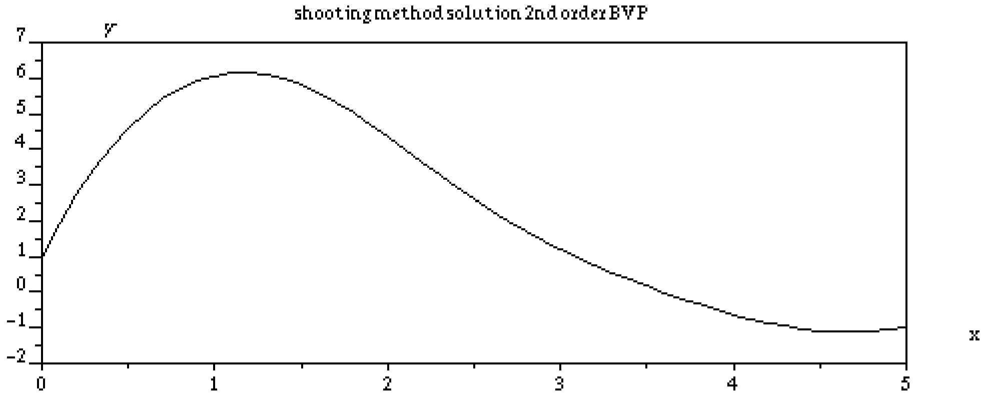
$$u(x) = \begin{bmatrix} u_1(x) \\ u_2(x) \end{bmatrix}, \quad f(x,u) = \begin{bmatrix} u_2 \\ \sin(3x) - u_1 - u_2 \end{bmatrix}.$$

The boundary conditions are now, $u_1(0) = 1$, $u_1(5) = -1$. The solution to this boundary value problem with SCILAB is accomplished as follows:

```
-->deff('[w]=f(x,u)','w=[u(2);sin(3*x)-u(1)-u(2)]')

-->yb=[1,-1];x0=0;Dx=0.1;xn=5;x=[x0:Dx:xn];yp=[-10:1:10];

-->[u,tabl,y0p]=shooting(yb,yp,x,f);
```

The plot of the solution is produced by using:

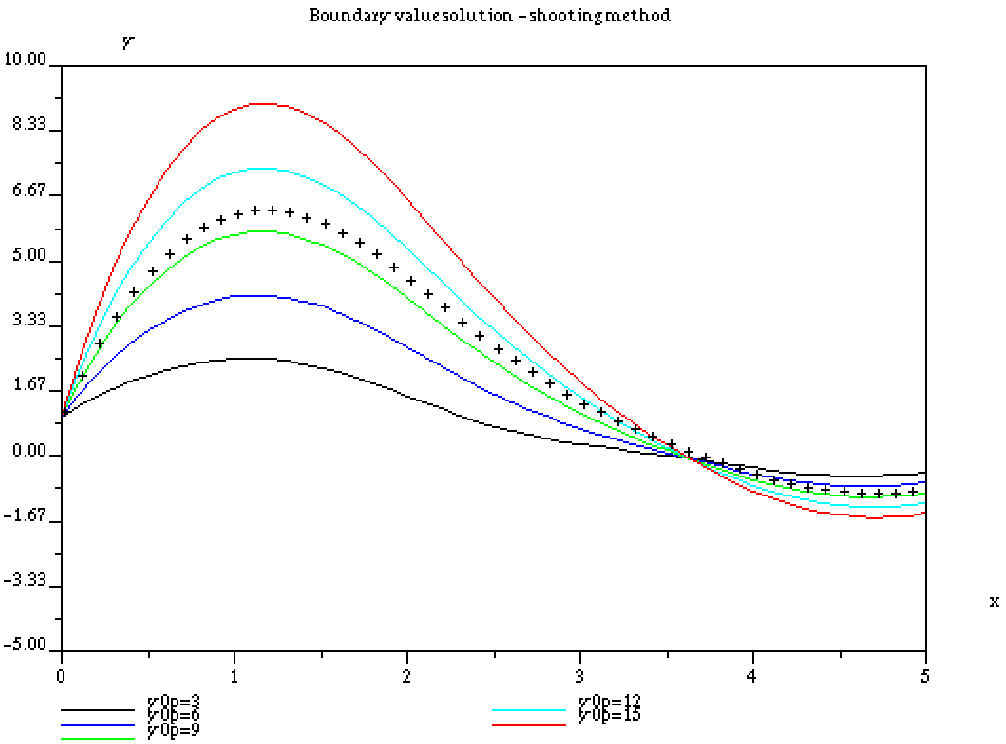
```
-->plot(x,u(1,:),'x','y','shooting method solution 2nd order BVP')
```



To illustrate the shooting method applied to the second-order boundary value problem presented above, we produce numerical solutions to the corresponding second-order ordinary differential equation using initial conditions $y(0) = 1$ and different derivative boundary conditions to produce the following graph. The solution to the boundary value problem with boundary conditions $y(0) = 1$ and $y(5) = -1$ is discontinuous curve in the graph.

```
-->yp0 = [3,6,9,12,15];    //Different values of dy/dx at x = 0

-->um = zeros(5,m);        //Calculate different solutions

-->for k =1:5
-->    y0 = [1;yp0(k)];
-->    uu = ode(y0,x0,x,f);
-->    um(k,:) = uu(1,:);
-->end;
```

```
-->plot2d([x',x',x',x',x'],[um(1,:)',um(2,:)',um(3,:)',um(4,:)',um(5,:)'],...
-->[1,2,3,4,5],'111','y0p=3@y0p=6@y0p=9@y0p=12@y0p=15',...
-->[0 -5 5 10])

-->plot2d(x',u(1,:)',-1,'011',' ',[0 -5 5 10])

-->xtitle('Boundary value solution - shooting method','x','y')
```



Outline of the implicit solution for a second-order BVP

In this section we present the outline for an implicit, finite-difference based solution to the second order boundary value problem

$$d^2y/dx^2+y = 0,$$

in the x-interval (*0*,*5*) subject to *y(0) = 1*, and *y(5) = 0*. Use *Δx = 0.1*.

We discretize the differential equation using the finite difference approximation

$$d^2y/dx^2 = (y_{i+2}-2\cdot y_{i+1}+y_i)/(\Delta x^2)\ ,$$

which results in

$$(y_{i+1}-2^*y_i+y_{i-1})/(\Delta x^2)+y_i = 0.$$

From this result we get the following implicit equations:

$$y_{i-1}-(2-\Delta x^2)\cdot y_i+y_{i+1} = 0,$$

for *i = 2,3, ..., n-1*. There are a total of (*n-2*) equations, corresponding to the *(n-2)* unknowns y_2, y_3, ..., y_{n-2}, y_{n-1} [$y_1 = y(0)$ and $y_n = y(5)$]. Therefore, the resulting set of linear algebraic equations has a unique solution. The implementation of the solution for this example is left as an exercise for the reader.

Function *bvode* for the solution of boundary value problems

SCILAB provides function *bvode* for the numerical solution of boundary value problems. The function works on a general boundary value problem, which, as indicated earlier, requires a lot of detailed information in its set up. The problem solved through function *bvode* consists of the boundary value problem:

$$\frac{d^{m*}y}{dx^{m*}} = f(x,u), \quad \text{with} \quad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{m*} \end{bmatrix} = \begin{bmatrix} y \\ dy/dx \\ d^2y/dx^2 \\ \vdots \\ d^{m*-1}y/dx^{m*-1} \end{bmatrix} = \begin{bmatrix} y \\ du_1/dx \\ du_2/dx \\ \vdots \\ du_{m*-1}/dx \end{bmatrix}.$$

For example, the second-order differential equation

$$d^2y/dx^2 + 5(dy/dx) + y = cos(x),$$

is re-written as

$$d^2y/dx^2 = cos(x) - y - 5(dy/dx),$$

or, with, $u_1 = y(x)$, $u_2 = dy/dx$, as

$$d^2y/dx^2 = f(x,u) = cos(x) - u_1 -5u_2.$$

External SCILAB functions used with *bvode*

Using SCILAB, the function f(x,u) can be defined, for example, as

```
deff('[ff] = f(x,u)','ff=cos(x)-u(1)-5*u(2)')
```

The general boundary value problem $d^{m*}y/dx^{m*} = f(x,u)$ is solved in the interval [*xL,xR*], with the boundary conditions provided through a function *g(i,u) = 0*, so that $g(i,u) = u_i - bc_j$, with $u_i = d^{i-1}y/dx^{i-1}$, and bc_j being the corresponding boundary condition at point $x = \zeta_j$. The values of ζ_j, *j = 1,2, ..., m**, are provided in vector ζ(*zeta*), and they are either *xL* or *xR*, i.e., the location of the boundaries for the problem.

For example, if the equation is of order 2 (*m*=2*), and the boundary conditions are $y(xL) = y_0$ and $y(xR) = y_1$, we will write the vector zeta as $\zeta = [xL,xR]$, or $\zeta_1 = xL$, $\zeta_2 = xR$, and $g(1,u) = u_1 - y_0$, $g(2,u) = u_2 - y_1$. The latter results are equivalent to $u_1(\zeta_1) = y(xL) = y_0$ and $u_2(xR) = y(xR) = y_1$.

As a second example, assume that the equation to be solved is of order 3 (*m*=3*) subject to boundary conditions $y(xL) = y_0$, $dy/dx|_{x=xL} = y_0'$, and $y(xR) = y_1$. The vector *zeta* is written as $\zeta = [xL,xL,xR]$, and the function *g(i,u)* given by $g(1,u) = u_1 - y_0$, $g(2,u) = u_2 - y_0'$, $g(3,u) = u_1 - y_1$. If instead, the boundary conditions are $y(xL) = y_0$, and $y(xR) = y_1$, $dy/dx|_{x=xR} = y_1'$, the vector *zeta* is written as $\zeta = [xL,xR,xR]$, and function *g(i,u)* given by $g(1,u) = u_1 - y_0$, $g(2,u) = u_1 - y_1$,

$g(3,u) = u_2 - y_1'$. Another possibility is that two of the boundary conditions are derivatives, e.g., $y(xL) = y_0$, $dy/dx|_{x=xL} = y_0'$, and $dy/dx|_{x=xR} = y_1'$. In this case we would have $\zeta = [xL,xL,xR]$, and $g(1,u) = u_1 - y_0$, $g(2,u) = u_2 - y_0'$, $g(3,u) = u_2 - y_1'$. Thus, there is a one-to-one relationship between the elements of vector zeta and values of function *g(i,u)* representing boundary conditions of the problem.

Function *bvode* requires that we pass also derivatives of the functions *f(x,u)* and *g(x,u).* Let's refer to the functions that calculate those derivatives as *df(x,u)* and *dg(i,u)*. Function *df* represents a vector [df_1, df_2, …, df_{m^*}] with $df_i = df(x,u)/du_i$, while function *dg* represents a vector whose elements are $dg_{ij} = dg_i/du_j$ for a fixed value of *i*. Referring to the second-order boundary value problem described earlier, namely, $d^2y/dx^2 = f(x,u) = \cos(x) - u_1 -5u_2$, subject to the boundary conditions $y(1) = 0.5$ and $y'(2) = -1.5$, we would define the *zeta* vector as ζ = [1,2], with $g_1 = g(1,u) = u_1 - 0.5$, and $g_2 = g(2,u) = u_2 + 1.5$. The derivative functions would be $df_1 = \partial f/\partial u_1 = -1$, $df_2 = \partial f/\partial u_2 = -5$, $dg_{11} = \partial g_1/\partial u_1 = 1.0$, $dg_{12} = \partial g_1/\partial u_2 = 0.0$, $dg_{21} = \partial g_2/\partial u_1 = 0$, $dg_{22} = \partial g_2/\partial u_2 = 1.0$. Using SCILAB, we will define the functions *f(x,u), g(i,u), df(x,u), dg(i,u)* as follows:

```
   deff('[ff]  = f(x,u) ','ff=cos(x)-u(1)-5*u(2)');
   deff('[dff] = df(x,u)','dff = [-1,-5]');
   deff('[gg]  = g(i,u) ',['gg=[u(1)-1,u(2)+1]','gg=gg(i)']);
   deff('[dgg] = dg(i,u)',['dgg = [1,0;0,1]';'dgg=dgg(i,:)']);
```

Notice that functions *f(x,u)* and *g(x,u)* return a single value, while functions *df(x,u)* and dg(x,u) return vectors.

The user may also define a function that provides initial guesses of the solution. The general call for this function is

[u0,du0] = guess(x)

where *x* is a vector, e.g., *x = [xL:Dx:xR*], with *Dx* = increment in x. Function *guess* provides initial guesses for the solution $u0 = [(u_1)_0; (u_2)_0; …;(u_{m^*})_0]$ and for the derivative of the solution, $du0 = [(du_1/dx)_0, (du_2/dx)_0, …, (du_{m^*}/dx)_0]$. Typically, this function is not used in the solution, i.e., no initial guesses of the solution and its derivative are provided, and the function *guess* can simply be defined as

```
deff(‘[u0,du0] = guess(x)’ , [‘u0= 0’, ‘du0 =0’])
```

General call to function *bvode*

The general call to the function *bvode* is as follows:

[u] = =bvode(x,n,m,xL,xR,zeta,ipar,ltol,tol,fixpnt,f,df,g,dg,guess)

where *x* is a vector of values of the independent variable, x = [xL:DX:xR]; *n* is the number of differential equations to be solved (*n = 1* for the case under consideration); *m* is a vector whose elements indicate the order each of the equations being solved (*m =[m*]* in this case); *xL* and *xR* are the extremes of the interval where the solution is sought; *zeta* is the vector specifying the location of the boundary conditions (examples of how to put together vector *zeta* were presented earlier); *ipar* is a vector of 11 solution parameters whose components are described below; *ltol* and *tol* are vectors specifying the number of the component of *u* and the tolerance for the solution of those components; *fixpnt* indicates the number of fixed points in the interval [*xL*, *xR*] other than the extremes (typically, *fixpnt = 0*); *f, df, g, dg,* and *guess* are external SCILAB functions described earlier.

Description of elements of vector *ipar*

Next, we describe the elements of vector *ipar*, showing typical values for the solution of simple boundary value problems. For a more detailed description of these parameters, use

```
--> help bvode
```

- *ipar(1)* determines whether the boundary value problem is linear (*ipar(1) = 0*) or non-linear (*ipar(1) = 1*).

- *ipar(2)* determines the number of collocation points per subinterval. Typically, *ipar(2) = 0*.

- *ipar(3)* = number of subintervals in the initial mesh. Typically, *ipar(3) = 0*, in which case the number of subintervals is set to 5.

- *ipar(4)* = number of solution and derivative tolerances, with $0 < ipar(4) \leq m^*$.

- *ipar(5), ipar(6)* = dimensions of workspaces (vectors). Select a relatively large number for those parameters, say 10000 or 15000.

- *ipar(7)* determines the type of output produced by the function *bvode*. With *ipar(7)* = -1, *bvode* produces a full diagnostic printout, with *ipar(7) = 0* bvode produces selected printout, and with *ipar(7) = 1* bvode produces no printout.

- *ipar(8)* controls the type of mesh used in the solution. If *ipar(8)* = 0 *bvode* generates a uniform initial mesh.

- *ipar(9)* is used to indicate whether or not initial guesses for the solution are provided. For example, if *ipar(9) = 0*, *bvode* understands that no initial guess for the solution is provided.

- *ipar(10)* offers options for breaking out of function *bvode* in case convergence problems are detected. For regular problems use *ipar(10) = 0*.

- *ipar(11)* = number of fixed points in the mesh other than *xL* and *xR*. This number is the same as the dimension of *fixpnt*. Typically, *ipar(11) = 0*.

For the solution of the second-order boundary value problem described earlier in this section, the vector *ipar* can be put together as:

```
ipar=0*ones(1,11)
ipar(3)=1;ipar(4)=2;ipar(5)=10000;ipar(6)=2000;ipar(7)=1
```

Application of function *bvode* to a second order boundary value problem

The problem to be solved is the one that has been described above, namely,

$$d^2y/dx^2 = f(x,u) = \cos(x) - u_1 - 5u_2,$$

subject to the boundary conditions

$$y(1) = 0.5 \text{ and } y'(2) = -1.5.$$

The parameters *n* and *m* in the call to function *bvode* for this problem are *n = 1* and *m = [2].* Also, *fixpnt = 0*. Functions *f,df,g,dg,* and *guess* for this problem were defined earlier. We also defined the vectors *zeta* and *ipar*. The boundaries are located at *xL = 1* and *xR = 2*. Using an x-increment *Dx = 0.1*, we can define the solution points as *x = [xL:Dx:xR].* There is a couple of arguments, namely, *ltol* and *tol* that need to be defined. Vector *ltol* indicates the indices of *u* for which a tolerance for convergence will be defined. For this case we can take *ltol = [1,2]* to define convergence tolerance for both the first and second components of *u*. The tolerances are relatively small numbers, say, *tol = [1e-10, 1e-10].*

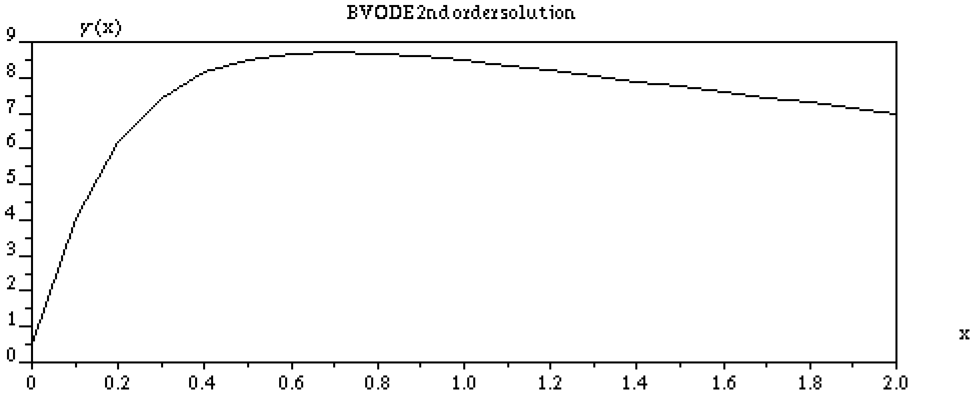
Having defined all the arguments for function *bvode* we put all the steps leading to the function call together in the following SCILAB script:

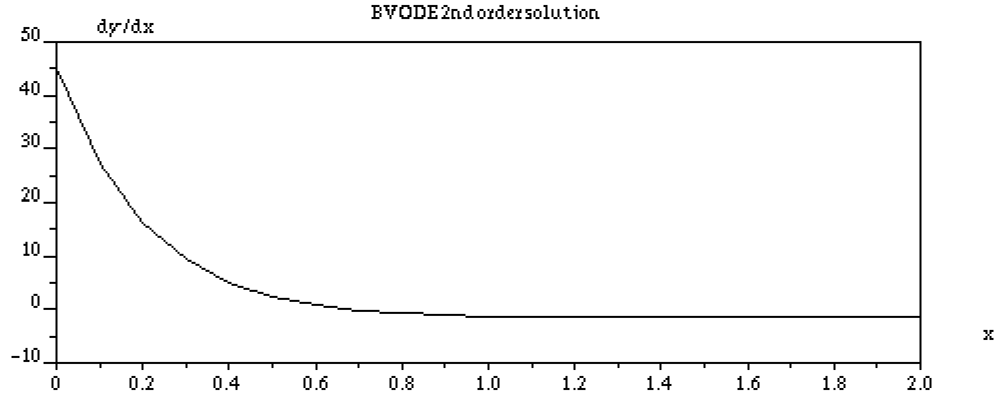
```
//Script for 2nd order boundary value problem solution with bvode

   deff('[ff]  = f(x,u) ','ff=cos(x)-u(1)-5*u(2)');
   deff('[dff] = df(x,u)','dff = [-1,-5]');
   deff('[gg]  = g(i,u) ',['gg=[u(1)-1,u(2)+1]','gg=gg(i)']);
   deff('[dgg] = dg(i,u)',['dgg = [1,0;0,1]';'dgg=dgg(i,:)']);
   deff('[u0,du0] = guess(x)' , ['u0= 0', 'du0 =0']);

   n=1;m=[2];fixpnt=0;xL=0;xR=2;Dx=0.1;
   x = [xL:Dx:xR];
   zeta=[xL,xR];

   ipar=zeros(1,11);
   ipar(3)=1;ipar(4)=2;ipar(5)=10000;ipar(6)=2000;ipar(7)=1;

   ltol=[1,2];
   tol=[1.e-5,1.e-5];

   u=bvode(x,n,m,xL,xR,zeta,ipar,ltol,tol,fixpnt,f,df,g,dg,guess);

   xset('window',1)
   plot(x,u(1,:),'x','y(x)','BVODE 2nd order solution')

   xset('window',2)
   plot(x,u(2,:),'x','dy/dx','BVODE 2nd order solution')
```

Placing this script in file *bvp2* within the SCILAB working directory, it can be executed through:

```
-->exec('bvp2')
```

The resulting plot shows the solution for *y(x)* and *dy/dx* versus *x*:





Function *bvode* applied to a third-order boundary value problem

Consider the third-order ordinary differential equation

$$x^2(d^3y/dx^3) + (1-x)(d^2y/dx^2) -3(dy/dx) + y = x+1$$

subject to the boundary conditions, $y(0) = 2.5$, $y(10) = 4.5$, $dy/dx|_{x=10} = -1$. To solve this problem using function *bvode* we first re-cast the differential equation as

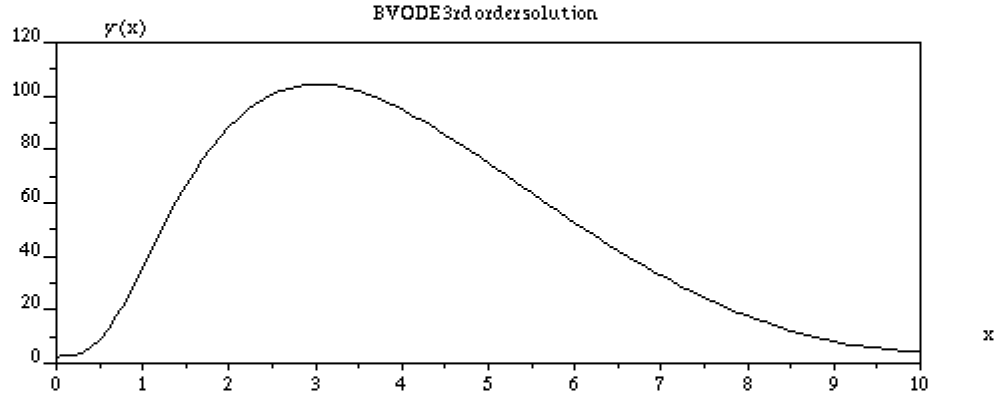
$$d^3y/dx^3 = f(x,u) = (x+1-(1-x)u_3 -3u_2 + u_1)/x^2$$

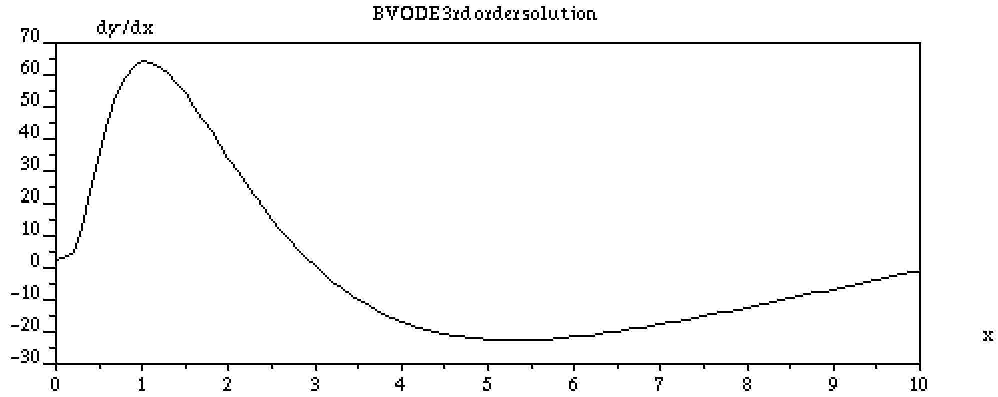
with $u_1 = y(x)$, $u_2 = dy/dx$, $u_3 = d^2y/dx^2$, and $u = [u_1;u_2;u_3]$. The boundary condition functions will be $g_1 = g(1,u) = u_1 - 2.5$, $g_2 = g(2,u) = u_1 - 4.5$, and $g_3 = g(3,u) = u_2 + 1$. The derivative functions will be $df = [1/x^2, -3/x^2, (1-x)]$, and $dg = [1,0,0;1,0,0;0,1,0]$. The zeta vector is $zeta = [0,10,10]$. Also, $n = 1$, $m = [3]$, $fixpnt = 0$.

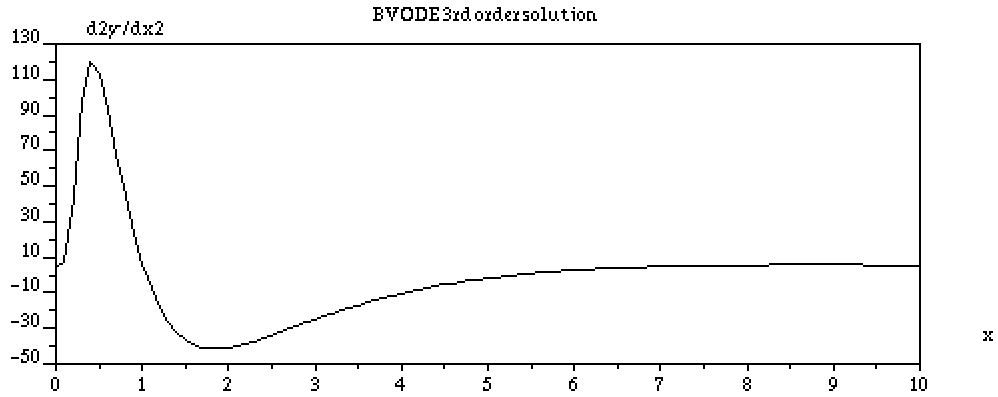
The following script solves the boundary value problem and produces graphs of the function y(x) and its first two derivatives:

```
//Script for 3rd order boundary value problem solution with bvode
   deff('[ff]  = f(x,u) ','ff=(x+1-(1-x)*u(3)-3*u(2)+u(1))/x^2');
   deff('[dff] = df(x,u)','dff = [1/x^2,-3/x^2,(1-x)/x^2]');
   deff('[gg]  = g(i,u) ',['gg=[u(1)-2.5,u(1)-4.5,u(2)+1]','gg=gg(i)']);
   deff('[dgg] = dg(i,u)',['dgg = [1,0,0;1,0,0;0,1,0]';'dgg=dgg(i,:)']);
   deff('[u0,du0] = guess(x)' , ['u0= 0', 'du0 =0']);
   n=1;m=[3];fixpnt=0;xL=0;xR=10;Dx=0.1;
   x = [xL:Dx:xR];
   zeta=[xL,xR,xR];
   ipar=zeros(1,11);
```

```
ipar(3)=1;ipar(4)=3;ipar(5)=50000;ipar(6)=50000;ipar(7)=1;
ltol=[1,2,3];
tol=[1.e-5,1.e-5,1e-5];
u=bvode(x,n,m,xL,xR,zeta,ipar,ltol,tol,fixpnt,f,df,g,dg,guess);
xset('window',1);plot(x,u(1,:),'x','y(x)','BVODE 3rd order solution')
xset('window',2);plot(x,u(2,:),'x','dy/dx','BVODE 3rd order solution')
xset('window',3);plot(x,u(3,:),'x','d2y/dx2','BVODE 3rd order solution')
```





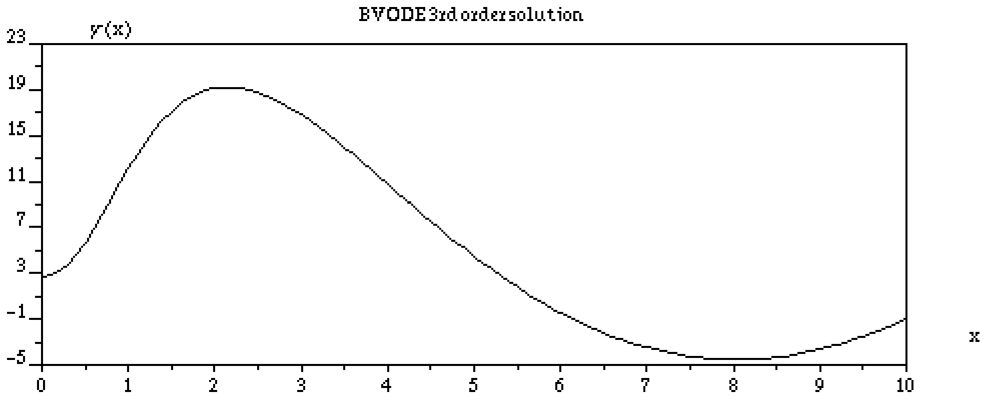


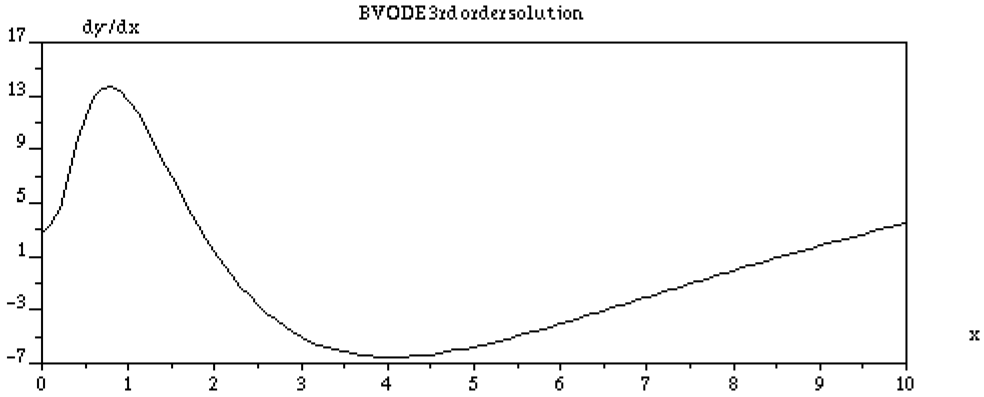
Application of *bvode* to a third-order problem with one interior fixed point

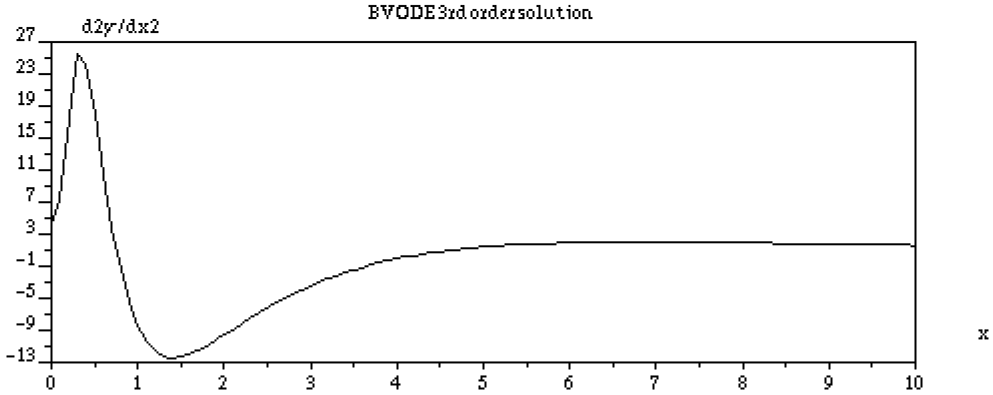
The following example solves the same third-order boundary value problem as before, except that now the boundary conditions are located at three different locations: *y(0) = 2.5, y(5) =*

4.5, y(10) = -1. To account for boundary conditions at three different locations, it is necessary to change a few of the parameters in the script, including introducing *xM = 5*, and using *fixpnt = [5]*, *zeta=[xL,xM,xR]*, *ltol=[1]*, and *tol=[1.e-5]*. Also, the option *ipar(11)=1* is introduced.

```
//Script for 3rd order boundary value problem solution with bvode
//Case of three different boundary points
   deff('[ff]  = f(x,u) ','ff=(x+1-(1-x)*u(3)-3*u(2)+u(1))/x^2');
   deff('[dff] = df(x,u)','dff = [1/x^2,-3/x^2,(1-x)/x^2]');
   deff('[gg]  = g(i,u) ',['gg=[u(1)-2.5,u(1)-4.5,u(1)+1]','gg=gg(i)']);
   deff('[dgg] = dg(i,u)',['dgg = [1,0,0;1,0,0;1,0,0]';'dgg=dgg(i,:)']);
   deff('[u0,du0] = guess(x)' , ['u0= 0', 'du0 =0']);
   n=1;m=[3];fixpnt=[5];xL=0;xR=10;Dx=0.1;xM = 5;
   x = [xL:Dx:xR];
   zeta=[xL,xM,xR];
   ipar=zeros(1,11);
   ipar(3)=1;ipar(4)=1;ipar(5)=50000;ipar(6)=50000;ipar(7)=1;ipar(11)=1;
   ltol=[1];
   tol=[1.e-5];
   u=bvode(x,n,m,xL,xR,zeta,ipar,ltol,tol,fixpnt,f,df,g,dg,guess);
   xset('window',1)
   plot(x,u(1,:),'x','y(x)','BVODE 3rd order solution')
   xset('window',2)
   plot(x,u(2,:),'x','dy/dx','BVODE 3rd order solution')
   xset('window',3)
   plot(x,u(3,:),'x','d2y/dx2','BVODE 3rd order solution')
```







Application of *bvode* to a fourth-order problem with two interior fixed points

Consider the fourth-order ordinary differential equation

$$x^3(d^4y/dx^4) +6x^2(d^3y/dx^3) +6x(d^2y/dx^2) = (x+1)^{1/2},$$

subject to the boundary conditions $y(1) = 1$, $dy/dx|_{x=1.2} = -2$, $d^2y/dx^2|_{x=1.7} = -0.5$, $d^3y/dx^3|_{x=7} = -0.1$. With $u_1 = y$, $u_2 = dy/dx$, $u_3 = d^2y/dx^2$, $u_4 = d^3y/dx^3$, and $u_5 = d^4y/dx^4$, the fourth-order system is written as

$$d^5y/dx^5 = f(x,u) = (1-6x^2u_4 - 6xu_3)/x^3 + (x+1)^{1/2}.$$

The boundary conditions will be described by

$$g_1=g(1,u) = u_1 - 1,\ g_2=g(2,u) = u_2 + 2,\ g_3=g(3,u) = u_3 + 0.5,\ g_4=g(4,u) = u_3 + 0.1.$$

The derivatives of functions *f* and *g* are given by

$$df_1 = 0,\ df_2 = 0,\ df_3 = -6/x^2,\ df_4 = -6/x,$$

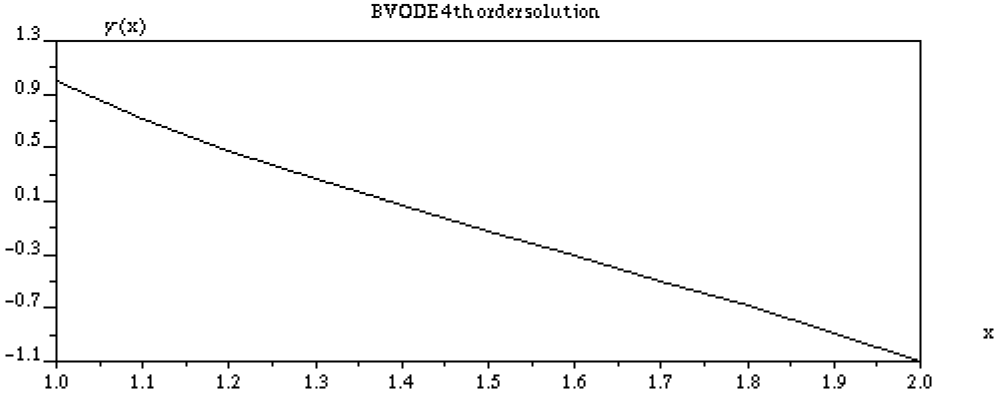
and

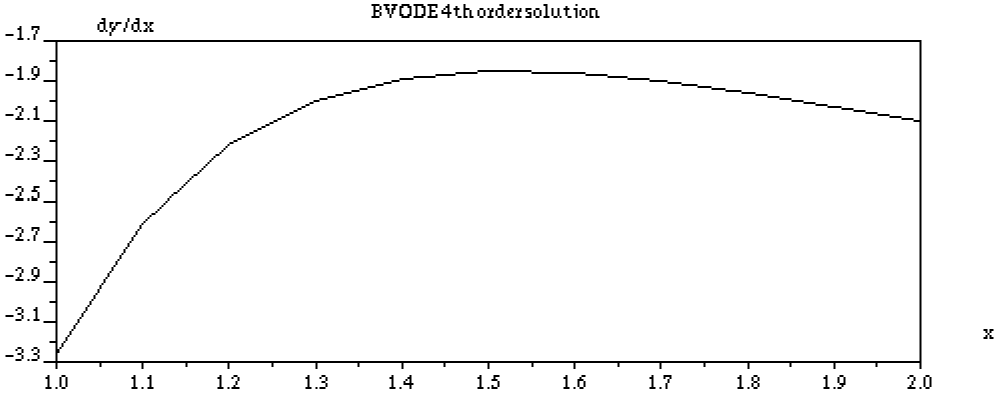
$$dg = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

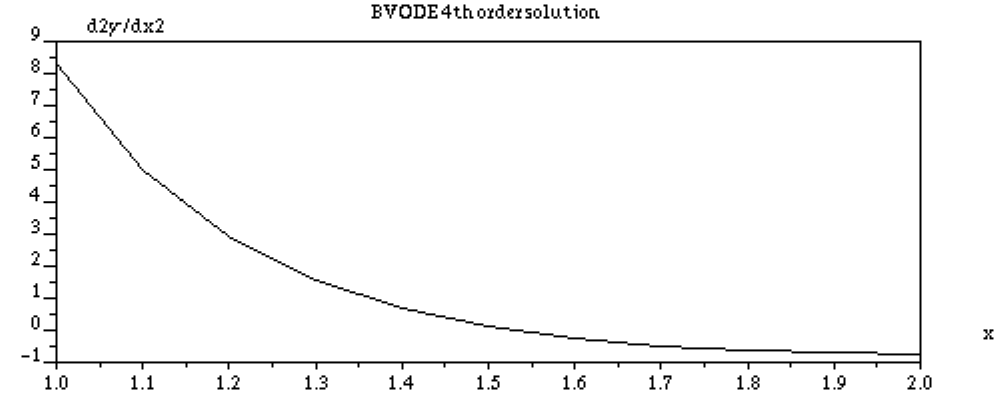
The parameters *fixptn* and *ipar(11)* are redefined as *fixptn = [1.2,1.7]* and *ipar(11) = 2*. Since we have four possible derivatives to deal with in the boundary conditions we want to redefine *Itol* and *tol* as Itol = *[1,2,3,4]* and *tol=[1e-5,1e=5,1e-5,1e-5]*, respectively. The value of *ipar(4)* is changed to *ipar(4) = 4.* The value of *m* is changed to *m = [4]*.

The following SCILAB script produces the solution for the problem just described.

```
//Script for 4th order boundary value problem solution with bvode
//Problem includes 2 interior points
   deff('[ff]  = f(x,u) ','ff=(1-6*x**2*u(4)-6*x*u(3))/x**3+sqrt(x+1)');
   deff('[dff] = df(x,u)','dff = [0,0,-6/x**2,-6/x]');
   deff('[gg]  = g(i,u) ',['gg=[u(1)-1,u(2)+2,u(3)+0.5,u(4)+0.1]','gg=gg(i)']);
   deff('[dgg] = dg(i,u)',['dgg = [1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1]';...
    'dgg=dgg(i,:)']);
   deff('[u0,du0] = guess(x)' , ['u0= 0', 'du0 =0']);
   n=1;m=[4];fixpnt=[1.3,1.7];xL=1;xR=2;Dx=0.1;x = [xL:Dx:xR];
   zeta=[1,1.3,1.7,2];
   ipar=zeros(1,11);
   ipar(3)=1;ipar(4)=4;ipar(5)=20000;ipar(6)=20000;ipar(7)=1
   ipar(11)=2;
   ltol=[1,2,3,4]
   tol=[1.e-11,1.e-11,1.e-11,1.e-11]
   u=bvode(x,n,m,xL,xR,zeta,ipar,ltol,tol,fixpnt,f,df,g,dg,guess);
   xset('window',1); plot(x,u(1,:),'x','y(x)','BVODE 4th order solution')
   xset('window',2); plot(x,u(2,:),'x','dy/dx','BVODE 4th order solution')
   xset('window',3); plot(x,u(3,:),'x','d2y/dx2','BVODE 4th order solution')
```







Boundary value problems with eigenvalues

Consider the boundary value problem

$$\frac{d^2y}{dx^2} + \lambda y = 0,$$

subject to $y(0) = 0$ and $y(L) = 0$, where λ is an unknown value. Assuming that $\lambda > 0$, the solution is a sinusoidal wave, i.e.,

$$y(x) = C_1 \sin(\sqrt{\lambda} x) + C_2 \cos(\sqrt{\lambda} x).$$

Replacing the boundary condition $y(0) = 0$ produces $0 = C_2$, thus, the solution reduces to

$$y(x) = C_1 \sin(\sqrt{\lambda} x).$$

The second boundary condition, namely, $y(L) = 0$, produces the *eigenvalue* equation

$$0 = C_1 \sin(\sqrt{\lambda} L).$$

Since, in general, we want the constant C_1 to be different from zero, the equation is satisfied if

$$\sin(\sqrt{\lambda} L) = 0,$$

i.e., if $\sqrt{\lambda} L = \pi, 2\pi, \ldots$ Thus, the problem has an infinite number of eigenvalues, $\lambda_n = n^2\pi^2/L^2$, $n=1,2,\ldots$ Associated with each eigenvalue is the eigenfunction $\sin(n\pi x/L)$. The most general solution to the original boundary value problem is a linear combination of the eigenfunctions, i.e.,

$$y(x) = \sum_{n=1}^{\infty} C_n \cdot \sin\left(\frac{n^2\pi^2 x}{L^2}\right).$$

Notice that the ordinary differential equation that defines the boundary value problem has a solution only for specific values of the constant λ.

Numerical solution to a boundary value problem with eigenvalues

Using finite difference approximations for the derivatives, it is sometimes possible to find a few eigenvalues of a boundary value problem such as the one described above. Using, for example, a centered finite difference formula for the derivative d^2y/dx^2, i.e.,

$$\frac{d^2y}{dx^2} \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{(\Delta x)^2},$$

into the differential equation

$$\frac{d^2y}{dx^2}+\lambda y=0,$$

produces the following difference equation

$$\frac{y_{i-1}-2y_i+y_{i+1}}{(\Delta x)^2}=-\lambda\cdot y_{i,}$$

for *i=2,3,…,n-1*. Implied in the latter result is the fact that the range of values of *x*, i.e., *0 < x < L*, is divided into *n-1* increments of size *Δx = L/(n-1).* Thus, $y_i = y(x_i)$, where $x_i = i\cdot\Delta x$. Also, $y_1 = y(0) = 0$, and $y_n = y(L) = 0$.

The problem involves *n-2* unknowns $y_2,y_3,\ldots,y_{n-1}$, in *n-2* equations. For example, for *L = 1*, *n = 5, Δx = 1/(5-1) = 0.25,* $1/(\Delta x)^2 = 16$. The general equation becomes

$$16\cdot y_{i-1}-32y_i+16y_{i+1}=-\lambda\cdot y_i,$$

for *i=2,3,4.* We have, therefore, the following three equations:

$$16y_1-32y_2+16y_3 = -\lambda y_2,$$
$$16y_2-32y_3+16y_4 = -\lambda y_3,$$
$$16y_3-32y_4+16y_5 = -\lambda y_4.$$

With $y_1 = y_5 = 0$, the three equations result in

$$-32y_2+16y_3 = -\lambda y_2,$$
$$16y_2-32y_3+16y_4 = -\lambda y_3,$$
$$16y_3-32y_4 = -\lambda y_4.$$

The resulting system of equations can be written in matricial form as

$$\begin{bmatrix}32&-16&0\\-16&32&-16\\0&-16&32\end{bmatrix}\cdot\begin{bmatrix}y_2\\y_3\\y_4\end{bmatrix}=\lambda\cdot\begin{bmatrix}y_2\\y_3\\y_4\end{bmatrix},$$

or, with

$$\mathbf{A}=\begin{bmatrix}32&-16&0\\-16&32&-16\\0&-16&32\end{bmatrix},\quad \mathbf{y}=\begin{bmatrix}y_2\\y_3\\y_4\end{bmatrix},$$

as,

$$\mathbf{Ay} = \lambda\mathbf{y}.$$

This is the classical eigenvalue problem which can be solved using SCILAB function *spec* or the user-defined function *eigenvectors* developed in Chapter 5.

A function for calculating eigenvalues for a boundary value problem

The following function, *BVPeigen1*, programs the solution to the boundary value problem described earlier, namely,

$$\frac{d^2y}{dx^2} + \lambda y = 0,$$

subject to *y(0) = 0* and *y(L) = 0*, where λ is an unknown value. The function call is

```
[x,y,lam]  = BVPeigen1(L,n)
```

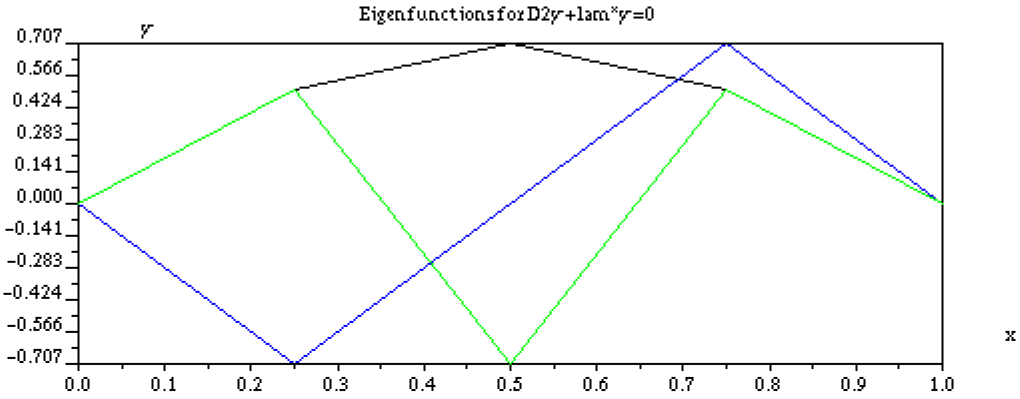
where x is a vector containing the values x_i, y is a matrix whose columns are the eigenvectors of the eigenvalue problem developed earlier (these eigenvectors are computed using function *eigenvectors*, developed in Chapter 5), and *lam* is a vector containing the *n* eigenvalues of the problem. The arguments of the function are the domain length *L* and the number of points in the solution *n*. The function also plots the eigenvectors for the first five eigenvalues found. These eigenvectors represent eigenfunctions $y_n(x)$. A listing of the function follows:

```
function [x,y,lam] = BVPeigen1(L,n)

Dx = L/(n-1);
x=[0:Dx:L];
a = 1/Dx^2;
k  = n-2;

A = zeros(k,k);
for j = 1:k
    A(j,j) = 2*a;
end;
for j = 1:k-1
    A(j,j+1) = -a;
    A(j+1,j) = -a;
end;

getf('eigenvectors');

[yy,lam]=eigenvectors(A);
//disp('yy');disp(yy);

y = [zeros(1,k);yy;zeros(1,k)];
//disp('y');disp(y);


xmin=min(x);xmax=max(x);ymin=min(y);ymax=max(y);
rect = [xmin ymin xmax ymax];
xset('window',1);xset('mark',[-1:-1:-10],1);
if k>=5 then
   m = 5;
else
   m = k;
end

for j = 1:m
    plot2d(x',y(:,j), j,'011',' ',rect);
    //plot2d(x',y(:,j),-j,'011',' ',rect);
end;
xtitle('Eigenfunctions for D2y+lam*y=0','x','y')
```

For example, for *L* = *1* and *n* = *5*, the following solution is obtained:

```
-->getf('BVPeigen1')

-->[x,y,lam]=BVPeigen1(1,5)

 lam  = !   9.372583    32.    54.627417 !


y  =

!   0.           0.           0.         !
!    .5        -  .7071068     .5        !
!    .7071068    3.140E-16  -  .7071068 !
!    .5           .7071068     .5        !
!   0.           0.           0.         !
 x  =

!   0.     .25     .5     .75    1. !
```

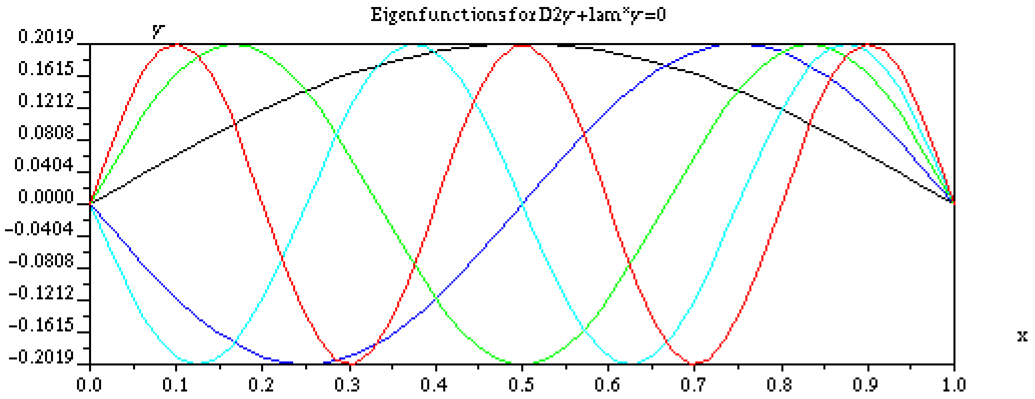
Notice that the first eigenvalue found is λ_1 = 9.372583, close to the theoretical value of π^2 = *9.8696*. A plot of the eigenfunctions follows.



To see the eigenfunctions in a more continuous fashion, we call function *BVPeigen1* using n = 50:

```
-->[x,y,lam]=BVPeigen1(1,50);
```

The first eigenvalue for *n=50* is 9.866224, closer to the theoretical value of π^2 = *9.8696* than the first eigenvalue found earlier for *n=5*. The plot of the eigenfunctions for *n=50* is shown below.



Notice that if the equation under consideration, $d^2y/dx^2 + \lambda y = 0$, represents the equation of a vibrating string, the eigenfunctions represent what are referred to as the different modes of vibration of the string. The eigenvalues represent the different natural angular frequencies of vibration of the string.

Exercises

[1]. Determine the general solution to the following linear ordinary differential equations using the corresponding characteristic equation:

(a) $d^3y/dx^3 + 4(dy/dx) + 5y = 0$. (b) $d^2y/dx^2 + 2(dy/dx)+y = 0.$
(c) $d^4y/dx^4+ d^2y/dx^2 + dy/dx + 3y = 0$ (d) $d^2y/dx^2 - 3y = 0$

[2]. Obtain the particular solution to the following second order equations:

(a) $d^2y/dx^2 + 3(dy/dx) + y = 2e^{-x}$ (b) $d^2y/dx^2 + 2y = 2x^2 + x$
(c) $d^2y/dx^2 + dy/dx = \sin(2x)$ (d) $d^2y/dx^2 + y = \cos x$

[3]. Plot the time variation of position, velocity, and acceleration of a damped mechanical oscillator for the following parameters:

(a) $m = 2$ kg, $\beta = 0.01$ Ns/m, $k = 2$ N/m, $x_0 = 0.2$ m, $v_0 = 1.2$ m/s
(b) $m = 4$ kg, $\beta = 0.10$ Ns/m, $k = 2$ N/m, $x_0 = 0.2$ m, $v_0 = 1.2$ m/s
(c) $m = 1$ kg, $\beta = 0.02$ Ns/m, $k = 2$ N/m, $x_0 = 0.2$ m, $v_0 = 1.2$ m/s
(d) $m = 0.5$ kg, $\beta = 0.25$ Ns/m, $k = 2$ N/m, $x_0 = 0.2$ m, $v_0 = 1.2$ m/s

[4]. Plot *v-vs-x*, *a-vs-x*, and *a-vs-v* phase portraits of the motions described in problem [3].

[5]. The mechanical oscillators described in problem [3] are subjected, respectively, to the driving forces shown below. In each case, plot the time variation of position, velocity, and acceleration of the resulting motions. Also, plot the *v-vs-x*, *a-vs-x*, and *a-vs-v* phase portraits of the motions.

(a) $F_0 = 2.5$ N, $\omega = 0.5$ rad/s (b) $F_0 = 10$ N, $\omega = 0.05$ rad/s
(c) $F_0 = 0.5$ N, $\omega = 1.5$ rad/s (d) $F_0 = 4$ N, $\omega = 0.25$ rad/s

[6]. For the following functions approximate the derivative df/dx at $x = a$ with $(f(a+h)-f(a))/h$ using values of $h = 0.1, 0.01, 0.001, 0.0001, 0.00001$. Plot the error involved in the numerical estimate of the derivative against the value of h.

(a) $f(x) = \sin(2x)$, $x = \pi$ (b) $f(x) = (x^2+3x)/(x+1)$, $x = 2$

(c) $f(x) = 1/(1+x^2)$, $x = -1$ (d) $f(x) = \tan(x)$, $x = \pi/4$

[7]. Repeat problem [4] but using a centered difference, i.e., $(f(a+h)-f(a-h))/(2h)$.

[8]. Repeat problem [4] for the second derivative d^2f/dx^2 using the forward difference approximation $(f(a+2h)-2f(a+h)+f(h))/h^2$.

[9]. Repeat problem [4] for the second derivative d^2f/dx^2 using the centered difference approximation $(f(a-h)-2f(a)+f(a+h))/h^2$.

[10]. Given the ODE,

$$dy/dx = y\cdot \sin(x),$$

and the boundary condition,

$$y(0) = 1,$$

write a SCILAB function that uses the Euler method to obtain a numerical solution to this ODE in the interval $0 < x < 2.5$. Use $\Delta x =$ 0.25, 0.1, and 0.05. The exact solution is given by

$$y = 2/(\cos x+1).$$

Plot the numerical solution against the exact solution for comparison.

[11]. Write a SCILAB function to produce an implicit solution for the first-order ODE from problem [10]. The exact solution is $y(x) = 1/x$.

[12]. Write a SCILAB function to complete the explicit solution for the second-order ODE $d^2y/dx^2 + y = 0$. An outline of the solution is presented elsewhere in this chapter. The exact solution is $y(x) = \sin x + \cos x$.

[13]. Write a SCILAB function to complete the implicit solution for the second-order ODE from problem [12]. The exact solution is $y(x) = \sin x+\cos x$.

[14]. Use SCILAB function *ode* to solve the following ordinary differential equations numerically. Plot the numerical results for the different values of Δx.

(a) $dy/dx = xy + \sin(x)$, $y(0) = 1$, $a = 0$, $b = 1$, $\Delta x = 0.2, 0.1, 0.05, 0.01$
(b) $dy/dx = \sin(x)\cos(y)$, $y(0) = 0$, $a = 0$, $b = \pi$, $\Delta x = \pi/10, \pi/20, p/50, \pi/100$
(c) $dy/dx = \exp(xy)$, $y(0) = 1$, $a = 0$, $b = 10$, $\Delta x = 1, 0.5, 0.2, 0.1$
(d) $dy/dx = x^2 - \sin(x)$, $y(0) = 2$, $a = 0$, $b = 5$, $\Delta x = 0.5, 0.25, 0.1, 0.05$

[15]. Solve the following systems of differential equations using matrices. Plot the solutions against x.

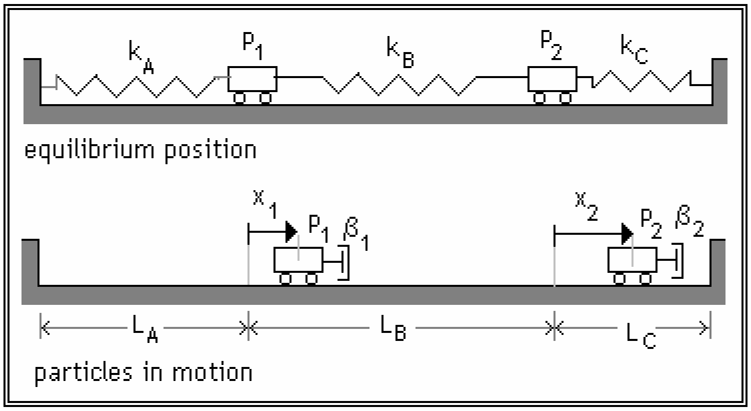
(a) $dy_1/dx = 2y_1 - y_2$, $dy_2/dx = 2(y_2-y_1)$, $y_1(0) = 1$, $y_2(0) = -1$
(b) $dy_1/dx = -5y_1 + y_2 + x^2$, $dy_2/dx = -y_1-y_2 - 5x$, $y_1(0) = 0$, $y_2(0) = 2$
(c) $dy_1/dx = 2y_1 - y_2$, $dy_2/dx = 2y_2-y_1$, $dy_3/dx = y_3-y_1$, $y_1(0) = 1$, $y_2(0) = -1$, $y_3(0) = 2$
(d) $dy_1/dx = 2y_1 - y_2 + y_3 + x$, $dy_2/dx = -y_1+ 2y_2 -2x$, $dy_3/dx = y_3-y_1$, $y_1(0) = 1$, $y_2(0) = -1$, $y_3(0) = 2$

[16]. Solve the systems of differential equations of problem [15] using SCILAB function *ode*. Plot the solutions against x.

[17]. Convert the following linear differential equations into systems of first-order ODEs and solve for y(x) using SCILAB function *ode*. Plot the solution y(x):

(a) $d^2y/dx^2 + dy/dx + 2y = x$, $y(0) = 1$, $dy/dx = -1$ *at* $x = 0$.
(b) $d^3y/dx^3 - 5(dy/dx) + y = 0$, $y(0) = 0$, $dy/dx = -1$ *and* $d^2y/dx^2 = 1$ *for* $x = 0$
(c) $d^3y/dx^3 - (d^2y/dx^2) + y = x$, $y(0) = 0$, $dy/dx = -1$ *and* $d^2y/dx^2 = 1$ *for* $x = 0$
(d) $d^2y/dx^2 + 2y = \sin(x)$, $y(0) = 1$, $dy/dx = -1$ *at* $x = 0$.

[18]. The figure below shows two particles P_1 and P_2, of mass m_1 and m_2, respectively, linked by three springs (k_A, k_B, k_C). The figure at the top represents the system in their state of equilibrium, while the one at the bottom shows the system at any generic point at time $t>0$. The displacement of particle P_1 with respect to its equilibrium position is given by $x_1(t)$ while that of particle P_2 is given by $x_2(t)$. The corresponding velocities are $v_1 = dx_1/dt$ and $v_2 = dx_2/dt$. The magnitude of the forces applied by the springs on the particles are given by Hooke's law, $F = k(L-L_0)$, where L is the stretched length of the spring, L_0 is the unstretched length of the spring, and k is the spring constant. The particles are also provided by dashpots that produce a viscous damping force whose magnitude is given by $F = \beta v$, where b is a damping constant and v is the speed of the particle, i.e., $F = \beta(dx/dt)$, where x = position of the particle.



(a) Write down the differential equations describing the motion of the two linked particles including spring and damping forces as shown in the figure above.

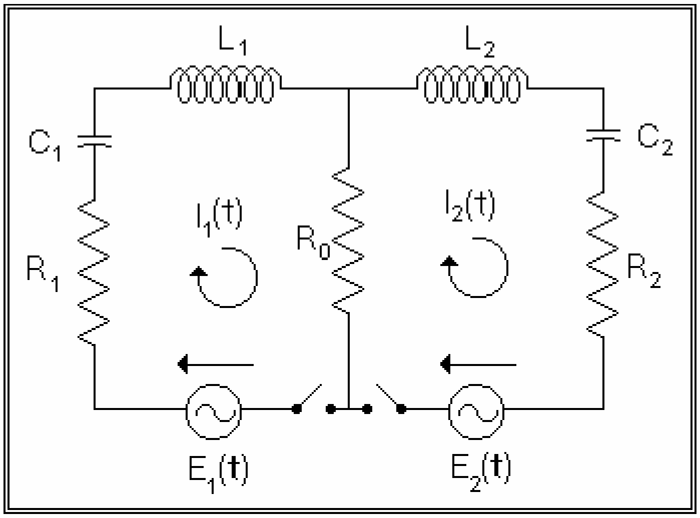
(b) Solve for $x_1(t)$ and $x_2(t)$ if $m_1 = 10$ *kg*, $m_2 = 20$ *kg*, $k_A = 80$ *N/m*, $k_B = 120$ *N/m*, $k_C = 100$ *N/m*, $\beta_1 = 1$ *N·s/m*, $\beta_2 = 5$ *N·s/m*. The initial conditions are given by $x_1(0) = 0.5$ *m*, $x_2(0) = 0.25$ *m*, $v_1(0) = 0$, $v_2(0) = 1.0$ *m/s*. Plot the signals and the velocity versus time for $0 < t < 5$ *s*.

(c) Solve for $x_1(t)$ and $x_2(t)$ if $m_1 = 10$ *kg*, $m_2 = 20$ *kg*, $k_A = 80$ *N/m*, $k_B = 120$ *N/m*, $k_C = 100$ *N/m*, $\beta_1 = 0$, for values of $\beta_2 =0$, *0.1, 0.5, 1*, and *5 N·s/m*. The initial conditions are given by $x_1(0) = 0.5$ *m*, $x_2(0) = 0.25$ *m*, $v_1(0) = 0$, $v_2(0) = 1.0$ *m/s*. Plot the signals and the velocity versus time for $0 < t < 5$ *s* for the different values of β_2.

(d) Write the differential equations for $x_1(t)$ and $x_2(t)$ if an external force $F_1 = F_0 \sin(\omega_0 t + \phi_0)$ is applied to particle P_1 in addition to the spring and damping forces.

(e) Using the conditions of part (a) of this problem solve for $x_1(t)$ and $x_2(t)$ for the case in which particle P_1 is subject to the external force $F_1 = F_0 \sin(\omega_0 t + \phi_0)$ with $F_0 = 20$ *N*, $\omega_0 = 2.5$ *rad/s*, and $\phi_0 = 1.5$ ^{rad}. Plot the signals, velocity, and acceleration versus time for $0 < t < 5$ *s* for the different values of β_2.

[19]. Consider the following two-loop electric circuit with R_0 = 2500 ohms, R_1 = 1500 ohms, R_2 = 1000 ohms, L_1 = 500 henrys, L_2 = 800 henrys, C_1 = 0.00006 farads, and C_2 = 0.001 farads.



(a) Write down the system of differential equations describing the electric charges $q_1(t)$ and $q_2(t)$ in the capacitors C_1 and C_2, respectively, and the electric currents $I_1(t)$ and $I_2(t)$ in the loops when the switches are turned on.

(b) For $q_1(0) = 0$, $q_2(0) = 0$, $I_1(0) = 0$, $I_2(0) = 0$, $E_1(t) = 0$, $E_2(t) = 120 \cos(30t)$ volts, determine the electric currents $I_1(t)$ and $I_2(t)$. Plot the results for $0 < t < 120$ s.

(c) For $q_1(0) = 0$, $q_2(0) = 0$, $I_1(0) = 0.2$ amperes, $I_2(0) = 0.1$ amperes, $E_1(t) = 6$ volts, $E_2(t) = 12$ volts, determine the electric currents $I_1(t)$ and $I_2(t)$. Plot the results for $0 < t < 120$ s.

(d) For $q_1(0) = 50$ coulombs, $q_2(0) = 100$ coulombs, $I_1(0) = 0$, $I_2(0) = 0.12$ amperes, $E_1(t) = 6 \sin(10\ t)$ volts, $E_2(t) = 12 \cos(30t)$, determine the electric currents $I_1(t)$ and $I_2(t)$. Plot the results for $0 < t < 120$ s.

(e) For $q_1(0) = 0$, $q_2(0) = 0$, $I_1(0) = 0$, $I_2(0) = 0$, $E_1(t) = E_2(t) = 120 \cos(120t)$, determine the electric currents $I_1(t)$ and $I_2(t)$. Plot the results for $0 < t < 120$ s.

[20]. The Zeeman's equations can be used to model the fluctuations on the length of the heart's fibers as the heart pumps blood through the blood vessels of a human body:

$$dx/dt = k(-y - x^3/3 + rx)$$
$$dy/dt = x/k,$$

where *x* is a measure of the fiber length fluctuation, *y* is a measure of the electrical stimulus that produces the fiber fluctuations, and *k* and *r* are constants. Solve the Zeeman's equation for the following parameters:

(a) $k = 0.5,\ p = 1,\ x(0) = 0,\ y(0) = -1$

(b) $k = 0.5,\ p =5,\ x(0) = 0,\ y(0) = -1$

(c) $k = 0.5,\ p = 10,\ x(0) = 0,\ y(0) = -1$

(d) $k = 0.5,\ p = 20,\ x(0) = 0,\ y(0) = -1$

Plot the signals *x vs. t*, *y vs. t*, and the phase portrait *x vs. y.*

[21]. The Lorenz equations are used to simulate the convection of a layer of fluid of infinite horizontal extent heated from below. The model is a simplified version of the heating of the atmosphere. The equations are obtained by expanding the terms for temperature and pressure involved in the problem with their Fourier series expansion and simplifying the expansion to the first three modes represented by the variables *x, y,* and *z*. The resulting system of equations is

$$dx/dt = \sigma(-x+y)$$
$$dy/dt = rx - y - xz$$
$$dz/dt = xy - bz$$

where σ, *r*, and *b* are constants that result from combining physical parameters of the problem. (For a detailed derivation refer to Berge, P., Y. Pomeau, and C. Vidal, 1984, "*Order within Chaos - Towards a deterministic approach to turbulence,*" John Wiley & Sons, New York).

Solve the Lorenz equations for the following combination of parameters:

(a) $\sigma = 10,\ r = 25,\ b = 2.666,\ x_0 = 1,\ y_0 = 1,\ z_0 = 1$
(b) $\sigma = 10,\ r = 75,\ b = 2.666,\ x_0 = 1,\ y_0 = 1,\ z_0 = 1$
(c) $\sigma = 10,\ r = 25,\ b = 2.666,\ x_0 = 1,\ y_0 = 1,\ z_0 = 1$
(d) $\sigma = 10,\ r = 25,\ b = 2.666,\ x_0 = 1,\ y_0 = 1,\ z_0 = 1$

Plot the signals *x-vs-t, y-vs-t, z-vs-t*, as well as the phase portraits *x-vs-y, x-vs-z*, and *y-vs-z*.

[22]. The governing equation for a pendulum of length L is the second-order ODE,

$$d^2\theta/dt^2 + (g/L)\ \theta = 0,$$

where *g* is the acceleration of gravity, and θ is the angle measured from the vertical position of the string. Solve the pendulum equation for the following conditions:

(a) $L = 1.2\ m,\ g = 9.806\ m/s^2,\ \theta_0 = \pi/3,\ (d\theta/dt)_0 = -1$
(b) $L = 3\ ft,\ g = 32.2\ ft/s^2,\ \theta_0 = \pi/6,\ (d\theta/dt)_0 = 1$
(c) $L = 2.0\ m,\ g = 9.806\ m/s^2,\ \theta_0 = \pi/2,\ (d\theta/dt)_0 = -0.5$
(d) $L = 6\ ft,\ g = 32.2\ ft/s^2,\ \theta_0 = 3\pi/4,\ (d\theta/dt)_0 = -0.5$

Plot the signals θ*-vs-t*, and *(dθ/dt)-vs-t*. Also, plot the phase portrait *dθ/dt-vs-* θ.

[23]. Repeat the solutions of problem [22] if the pendulum is subjected to a periodic excitation so that the governing equation becomes

$$d^2\theta/dt^2 + (g/L)\ \theta = (F_0/(mL))\ \cos(\omega t+\phi).$$

The values to use for each of the cases in problem [22] are as follows:

(a) $F_0 = 2.5\ N,\ m = 0.2\ kg,\ \omega = \pi/2\ rad/s,\ \phi = \pi/3$
(b) $F_0 = 0.5\ N,\ m = 0.8\ kg,\ \omega = 1.0\ rad/s,\ \phi = 0$
(c) $F_0 = 1.5\ N,\ m = 1.2\ kg,\ \omega = 0.1\ rad/s,\ \phi = 2\pi/3$
(d) $F_0 = 3.0\ N,\ m = 0.1\ kg,\ \omega = 0.05\ rad/s,\ \phi = -\pi/3$

[24]. Solve the following boundary value problem using the shooting method.

(a) $d^2y/dx^2 + 3(dy/dx) + 2y = \sin(2x),\ y(0) = 1,\ y(1) = 0$
(b) $d^2y/dx^2 - 3y = 1 + x,\ y(0) = -1,\ dy/dx|_{x = 1} = -0.5$
(c) $d^2y/dx^2 + dy/dx = \ln(x),\ dy/dx|_{x = 0} = 0,\ y(1) = 1$
(d) $d^2y/dx^2 - dy/dx - y = 2 \sin(x),\ y(1) = -2,\ y(2) = 3$

[25]. Solve the boundary value ODEs of problem [24] by using an implicit solution with finite differences.

[26]. Solve the boundary value ODEs of problem [24] by using function *bvode*.

[27]. Solve the following boundary value ODEs using function *bvode*:

(a) $d^3y/dx^3 + y = 1 + x^2,\ y(0) = 1,\ y(1) = 2,\ y(3) = -1$
(b) $d^3y/dx^3 + d^2y/dx^2 = x,\ y(0) = 1,\ dy/dx|_{x=1} = -1,\ y(2) = 0$
(c) $d^2y/dx^2 - dy/dx = e^{-x/2},\ dy/dx|_{x=0} = -1,\ y(1) = 2$
(d) $d^3y/dx^3 + dy/dx = -1 + x,\ y(0) = 1,\ dy/dx|_{x=2} = -1,\ d^2y/dx^2|_{x=2} = 1$

[28]. Determine the first n eigenvalues of the problem $d^2y/dx^2 + \lambda y = 0$, subject to $y(0) = 0$, $y(L) = 0$, for the following combinations of values of n and L:

(a) $n = 10,\ L =10$
(b) $n = 20,\ L = 5$
(c) $n = 15,\ L = 1$
(d) $n = 30,\ L = 100$

[29]. For the differential equation

$$x^2(d^2y/dx^2)+x(dy/dx)'+(1+\lambda)y=0,\ y(1) = y(2) = 0,$$

use centered-difference approximations for the derivatives to perform an implicit numerical solution. Obtain the first 10 eigenvalues of the problem. Plot the corresponding eigenfunctions.

[30]. Solve the following system of equations

$$dx/dt = -y(x^2+y^2),$$

$$dy/dt = x(x^2+y^2),$$

for the initial conditions $x(0) = 2,\ y(0) = 1$. Plot the signals *x-vs-t* and *y-vs-t*, as well as the phase portraits *x-vs-y, (dx/dt)-vs-x, (dy/dt)-vs-y,* and *(dy/dt)-vs-(dx/dt).*

[31]. The following system of equations is known as a set of coupled logistic equations and can be used to model the behavior of two linked populations $x_1(t)$ and $x_2(t)$:

$$dx_1/dt = kx_1(1 - (x_1+x_2)/N),$$

$$dx_2/dt = kx_2(1 - (x_1+x_2)/N).$$

Solve the system for the following combination of parameters and initial conditions:

(a) $k = 1,\ N = 1,\ (x_1)_0 = 1,\ (x_2)_0 = 1$
(b) $k = 0.5,\ N = 5,\ (x_1)_0 = 0,\ (x_2)_0 = -1$
(c) $k = 2.5,\ N = 10,\ (x_1)_0 = -2,\ (x_2)_0 = 0$
(d) $k = 12,\ N = 25,\ (x_1)_0 = 1,\ (x_2)_0 = 1$

[32]. The governing equation for gradually varied flow in an open channel is given by

$$\frac{dy}{dx} = \frac{S_0 - S_f}{1 - F^2},$$

where S_0 is the slope of the channel bed, i.e., the rate of change of bed elevation, z, with distance, x, along the channel bed ($S_0 = -dz/dx$), S_f is the slope of the energy head, i.e., the rate of change of the total energy head, H (energy per unit weight), with x ($S_f = -dH/dx$), and F is a dimensionless quantity known as the Froude number. The energy head, H, is the sum of the bed elevation z, the water depth y, and the velocity head (kinetic energy per unit weight) $V^2/(2g)$, i.e.,

$$H = z + y + V^2/2g,$$

where V is the flow velocity in the cross section (V = Q/A, Q = flow discharge, A = cross-sectional area). The energy slope is calculated by using Manning's equation with

$$S_f = \left(\frac{nQ}{C_u}\right)^2 \frac{P^{4/3}}{A^{10/3}},$$

where n is the Manning coefficient (a measure of the channel bed roughness), C_u is a constant that depends on the system of units used ($C_u = 1.0$ for the S.I., $C_u = 1.486$ for the English system), and P is the wetted perimeter of the cross-section (part of the channel cross-sectional perimeter in contact with the water).

The Froude number squared is calculated from

$$F^2 = \frac{Q^2 T}{gA^3},$$

where T is the top-width of the cross-section (i.e., the length of the free surface at the cross section).

For a trapezoidal section of bottom width b, side slope z, and depth y, the area, wetted perimeter, and top width are given by

$$A = (b+zy)y$$
$$P = b + 2y(1+z^2)^{1/2}$$
$$T = b + 2zy$$

For a trapezoidal cross-section open channel with $b = 2.5\ ft$, $z = 1$, $S_0 = 0.00001$, $g = 32.2\ ft/s^2$, $C_u = 1.486$, $n = 0.012$, $Q = 5.0\ ft^3/s$, and with initial conditions $y = 2.5\ ft$ at $x = 10000\ ft$, plot the solution y(x) between $x = 0$ and $x = 10000\ ft$.

Notes on problem [32]:
(1). A plot of the solution y-vs-x is called the water surface profile or a backwater curve. Backwater curves are created whenever there is a so-called *control point* in the channel. For example, the depth of *2.5 ft* at position $x = 10000\ ft$, used as initial conditions in this problem, could be created by a small sill placed across the channel at that position.

(2). At points where the energy slope, S_f, is the same as the bed slope, S_0, i.e., $S_f = S_0$, then dy/dx = 0. At those points the flow is said to have reached uniform conditions (uniform flow) and the constant depth thus achieved is referred to as the *normal depth*, y_n. You can check that for the conditions of the present problem $y_n = 2.311973\ ft$ by solving for y from the Manning's equation ($S_f(y) = S_0$).

(3). At a point where the Froude number is equal to 1, the flow is said to be critical. The corresponding depth is referred to as the critical depth, y_c. You can check that for the conditions of the present problem $y_c = 0.4673298\ ft$ by solving for y from the equation defining the Froude number squared, i.e., $F^2(y) = 1$.

REFERENCES (for all SCILAB documents at InfoClearinghouse.com)

Abramowitz, M. and I.A. Stegun (editors), 1965,"*Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*," Dover Publications, Inc., New York.

Arora, J.S., 1985, "*Introduction to Optimum Design*," Class notes, The University of Iowa, Iowa City, Iowa.

Asian Institute of Technology, 1969, "*Hydraulic Laboratory Manual*," AIT - Bangkok, Thailand.

Berge, P., Y. Pomeau, and C. Vidal, 1984,"*Order within chaos - Towards a deterministic approach to turbulence*," John Wiley & Sons, New York.

Bras, R.L. and I. Rodriguez-Iturbe, 1985,"*Random Functions and Hydrology*," Addison-Wesley Publishing Company, Reading, Massachussetts.

Brogan, W.L., 1974,"*Modern Control Theory*," QPI series, Quantum Publisher Incorporated, New York.

Browne, M., 1999, "*Schaum's Outline of Theory and Problems of Physics for Engineering and Science*," Schaum's outlines, McGraw-Hill, New York.

Farlow, Stanley J., 1982, "*Partial Differential Equations for Scientists and Engineers*," Dover Publications Inc., New York.

Friedman, B., 1956 (reissued 1990), "*Principles and Techniques of Applied Mathematics*," Dover Publications Inc., New York.

Gomez, C. (editor), 1999, "*Engineering and Scientific Computing with Scilab*," Birkhäuser, Boston.

Gullberg, J., 1997, "*Mathematics - From the Birth of Numbers*," W. W. Norton & Company, New York.

Harman, T.L., J. Dabney, and N. Richert, 2000, "*Advanced Engineering Mathematics with MATLAB® - Second edition*," Brooks/Cole - Thompson Learning, Australia.

Harris, J.W., and H. Stocker, 1998, "*Handbook of Mathematics and Computational Science*," Springer, New York.

Hsu, H.P., 1984, "*Applied Fourier Analysis*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, San Diego.

Journel, A.G., 1989, "*Fundamentals of Geostatistics in Five Lessons*," Short Course Presented at the 28th International Geological Congress, Washington, D.C., American Geophysical Union, Washington, D.C.

Julien, P.Y., 1998,"*Erosion and Sedimentation*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Keener, J.P., 1988, "*Principles of Applied Mathematics - Transformation and Approximation*," Addison-Wesley Publishing Company, Redwood City, California.

Kitanidis, P.K., 1997,"*Introduction to Geostatistics - Applications in Hydogeology*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Koch, G.S., Jr., and R. F. Link, 1971, "*Statistical Analysis of Geological Data - Volumes I and II*," Dover Publications, Inc., New York.

Korn, G.A. and T.M. Korn, 1968, "*Mathematical Handbook for Scientists and Engineers*," Dover Publications, Inc., New York.

Kottegoda, N. T., and R. Rosso, 1997, "*Probability, Statistics, and Reliability for Civil and Environmental Engineers*," The Mc-Graw Hill Companies, Inc., New York.

Kreysig, E., 1983, "*Advanced Engineering Mathematics - Fifth Edition*," John Wiley & Sons, New York.

Lindfield, G. and J. Penny, 2000, "*Numerical Methods Using Matlab®,*" Prentice Hall, Upper Saddle River, New Jersey.

Magrab, E.B., S. Azarm, B. Balachandran, J. Duncan, K. Herold, and G. Walsh, 2000, "*An Engineer's Guide to MATLAB®*", Prentice Hall, Upper Saddle River, N.J., U.S.A.

McCuen, R.H., 1989,"*Hydrologic Analysis and Design - second edition*," Prentice Hall, Upper Saddle River, New Jersey.

Middleton, G.V., 2000, "*Data Analysis in the Earth Sciences Using Matlab®,*" Prentice Hall, Upper Saddle River, New Jersey.

Montgomery, D.C., G.C. Runger, and N.F. Hubele, 1998, "*Engineering Statistics*," John Wiley & Sons, Inc.

Newland, D.E., 1993, "*An Introduction to Random Vibrations, Spectral & Wavelet Analysis - Third Edition*," Longman Scientific and Technical, New York.

Nicols, G., 1995, "*Introduction to Nonlinear Science*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Parker, T.S. and L.O. Chua, , "*Practical Numerical Algorithms for Chaotic Systems*," 1989, Springer-Verlag, New York.

Peitgen, H-O. and D. Saupe (editors), 1988, "*The Science of Fractal Images*," Springer-Verlag, New York.

Peitgen, H-O., H. Jürgens, and D. Saupe, 1992, "*Chaos and Fractals - New Frontiers of Science*," Springer-Verlag, New York.

Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, 1989, "*Numerical Recipes - The Art of Scientific Computing (FORTRAN version)*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Raghunath, H.M., 1985, "*Hydrology - Principles, Analysis and Design*," Wiley Eastern Limited, New Delhi, India.

Recktenwald, G., 2000, "*Numerical Methods with Matlab - Implementation and Application*," Prentice Hall, Upper Saddle River, N.J., U.S.A.

Rothenberg, R.I., 1991, "*Probability and Statistics*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, San Diego, CA.

Sagan, H., 1961,"*Boundary and Eigenvalue Problems in Mathematical Physics*," Dover Publications, Inc., New York.

Spanos, A., 1999,"*Probability Theory and Statistical Inference - Econometric Modeling with Observational Data*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Spiegel, M. R., 1971 (second printing, 1999), "*Schaum's Outline of Theory and Problems of Advanced Mathematics for Engineers and Scientists*," Schaum's Outline Series, McGraw-Hill, New York.

Tanis, E.A., 1987, "*Statistics II - Estimation and Tests of Hypotheses*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, Fort Worth, TX.

Tinker, M. and R. Lambourne, 2000, "*Further Mathematics for the Physical Sciences*," John Wiley & Sons, LTD., Chichester, U.K.

Tolstov, G.P., 1962, "*Fourier Series*," (Translated from the Russian by R. A. Silverman), Dover Publications, New York.

Tveito, A. and R. Winther, 1998, "*Introduction to Partial Differential Equations - A Computational Approach*," Texts in Applied Mathematics 29, Springer, New York.

Urroz, G., 2000, "*Science and Engineering Mathematics with the HP 49 G - Volumes I & II*", www.greatunpublished.com, Charleston, S.C.

Urroz, G., 2001, "*Applied Engineering Mathematics with Maple*", www.greatunpublished.com, Charleston, S.C.

Winnick, J., , "*Chemical Engineering Thermodynamics - An Introduction to Thermodynamics for Undergraduate Engineering Students*," John Wiley & Sons, Inc., New York.