

Solving a System of Second-order Ordinary Differential Equations with a Fourth-order Nystrom Method

This *Numerit* program (demodeq1) demonstrates how to solve a system of second-order vector differential equations using a *fixed step size* fourth-order Nystrom method. The function which solves this problem is called nym4.

This function solves a system of second-order vector differential equations of the form

$$\frac{d^2\vec{y}}{dt^2} = f\left(\vec{y}, \frac{d\vec{y}}{dt}, t\right) \quad (1)$$

Equation (1) indicates that the system of differential equations may be a function of the independent vector \vec{y} , the derivative of the independent vector $\frac{d\vec{y}}{dt}$, and time t .

The syntax of the nym4 function is as follows:

```
function nym4(n, tp, dt, ys, ydots, yf, ydotf)
` solve a system of second order differential equations
` fourth order Nystrom method (fixed step size)
` input
` n      = number of equations in user-defined
`        system of differential equations
` tp     = current simulation time
` dt     = integration step size
` ys     = y vector at initial time
` ydots  = ydot vector at initial time
` output
` yf     = y vector at time = tp + dt
` ydotf  = ydot vector at time = tp + dt
```

The function nym4 requires a user-coded *Numerit* function that evaluates the system of vector differential equations. The correct syntax for this function is as follows:

```
function nyeqm(t, y, ydot, yddot)
` first order equations of orbital motion
` input
` t      = simulation time
` y      = y vector at t
` ydot   = y dot vector at t
` output
` yddot  = y double dot vector at t
```

Numerical Analysis with Numerit

In the parameter list, t is the current integration time, y is the vector of independent variables, $ydot$ is the vector of first-order derivatives, and $yddot$ is the vector of second-order derivatives. Prior to calling the `nym4` function the user needs to *redirect* the actual name of the function to solve with a statement similar to `nyeqm -> eqm1`.

The differential equations function for this example is

```
function eqm1(t, y, ydot, yddot)
  ` first order equations of orbital motion
  ` input
  ` t    = simulation time
  ` y    = y vector at t
  ` ydot = y dot vector at t
  ` output
  ` yddot = y double dot vector at t
  ` Numerical Analysis with Numerit
  .....
mu = 1.407645794e+16
ymag = sqrt(y[1] * y[1] + y[2] * y[2] + y[3] * y[3])
ymag3 = ymag * ymag * ymag
for i = 1 to 3
  yddot[i] = -mu * y[i] / ymag3
```

The `nym4` function evaluates the differential equations for one step size each time it is called. The demonstration program illustrates a simple driver routine which uses a `while` construct and calls the `nym4` function for an integration period specified by the user. This driver simply checks to see if the time remaining is less than the initial step size input by the user. If this is true, it sets the final step size to this value and completes the numerical integration.

The *generic Numerit* source code for this routine is as follows:

```
while (abs(tf - ti) > 0.00000001)
  ` check current step size
  a = tf - ti
  if (abs(a) < dt) dt = a * sign(a)
  nym4(neq, ti, dt, yi, ydoti, yf, ydotf)
  ` update time, y and ydot vector
  ti = ti + dt
  yi = yf
  ydoti = ydotf
```

Numerical Analysis with Numerit

Within this code t_i is the initial time, t_f is the final time and dt is the user-defined step size. Notice how the initial time, y vector and y derivative vectors are reset after each integration step.

The demonstration program solves the system of three vector differential equations which describe the "unperturbed" or *Keplerian* motion of an Earth-orbiting spacecraft. The program requires an initial time, integration period, and step size. A step size between 10 and 60 seconds is recommended. The initial and final times are "hardwired".

Typical initial conditions for the position and velocity of the spacecraft are "hardwired" within the program. The position vector is in units of feet and the velocity vector input is feet per second. The demonstration program will print the results of the numerical integration in the same units. The system of three second-order vector differential equations solved by the software is as follows:

$$\begin{aligned}\frac{d^2 r_x}{dt^2} &= -\mathbf{m} \frac{r_x}{R^3} \\ \frac{d^2 r_y}{dt^2} &= -\mathbf{m} \frac{r_y}{R^3} \\ \frac{d^2 r_z}{dt^2} &= -\mathbf{m} \frac{r_z}{R^3}\end{aligned}\tag{2}$$

where

$$\begin{aligned}r_x, r_y, r_z &= \text{position vector of the spacecraft} \\ R &= \sqrt{r_x^2 + r_y^2 + r_z^2} = \text{position magnitude} \\ \mathbf{m} &= \text{gravitational constant of the Earth}\end{aligned}$$

These three second-order differential equations are also called the equations of motion of the spacecraft. Typical position (feet) and velocity vectors (feet per second) for the Space Shuttle are as follows:

$$\begin{aligned}-19472500.3, & 6587457.02, 7367882.5 \\ -4687.10293, & -23436.308, 8566.3774\end{aligned}$$

The orbital period is the time required for the Space Shuttle to complete one orbit around the Earth. For this example the orbital period is 5404.124635 seconds. If we numerically integrate for exactly one orbital period the Space Shuttle should return very close to the initial conditions.

This software can be used to assess the effect of different step sizes on how well the orbit returns to the initial conditions by integrating for exactly one or more orbital periods. This type of analysis can help you evaluate the "closure" errors and "calibrate" the integrator for any particular problem.

Numerical Analysis with Numerit

The following is a typical draft output created with this software.

```
program demodeq1  
< solution of differential equations with Nystrom method >  
step size 30 seconds  
position vector error (feet)  
-1.7270314283669 -8.60258507169783 3.14748471789062  
velocity vector error (fps)  
0.00966526026240899 -0.00326574406062718 -0.0036581579970516
```

For this example the step size was 30 seconds and the integration period was exactly one orbital period.