Three Body Problem using High-Order Runge-Kutta Interpolation

Lawrence Mulholland

Contents

1 NAG Library Mark 26 Reverse Communication Runge-Kutta Routines

1.1 Reverse Communication

Runge-Kutta processes are by far the most widely used methods to solve non-stiff ordinary differential equations (ODEs). Generally, software for these methods are presented in a form where the system to be solved is provided as routine argument to the interface. However, this is not always the most convenient way to provide such a system. For example, the system may incorporate parameters that are themselves the solution of an associated problem (sparse least-squares, say). A flexible alternative is to allow the system to be evaluated at a point in time outside of the solver routine:

```
Call setup routine
Loop1 over time steps
     Loop2 over reverse communcation Solver
          Call solver
          If step complete is flagged
              Exit Loop2
          Else
              evaluate system (y'=) f(y,t) at some given (y,t)
          End
     End

     solution at current time available, y(t)

     If final time reached
        Exit Loop1
     End
End
```

The routine d02pgf, introduced at Mark 26 of the NAG Libraries, is a one-step, reverse-communication Runge-Kutta routine which performs the same functionality as he forward-communication routine d02pff.

1.2 High order interpolation

The solution is available at each time step, but the solution at intermediate times might be required. A common situation is where we want to find any cases where a nonlinear system $G(y, t) = 0$ has a solution on the trajectory $y(t)$. That is, does $G(y, t)$ have any roots in the last time-step from $t = t_prev$ to t_now. o do this requires that the solution $y(t)$ be interpolated between the solutions available at the last and current time steps. To do this accurately is not straightforward; it involves solving the system from $t = t_prev$ to $t = t_now$ using a continuous Runge-Kutta process. This continuous process, once constructed, can then efficiently return accurate solutions for any number of points in time over the last time-step; the same order of accuracy as the original discrete process is achieved.

The contruction phase requires solving the original system, so if we require reverse communication to solve the ODE, then we require reverse communication to construct the continuous interpolator. The routine d02phf, introduced to the NAG Libraries at Mark 26, is a reverse communication continuous interpolation constructor. A particular advantage of this routine is that it can contruct an interpolator for the high-order 7(8) Runge-Kutta pair. The routine d02pjf uses the construction provided by d02phf to cheaply evaluate the ODE solution at any point in time over the last time-step.

1.3 Root Finding

Finding possible roots $G(y, t) = 0$, $t \in [t_p rev, t_n ow]$ requires the ODE solution $y(t)$ to be evaluated at a number of points in $[t_p rev, t_n ow]$ and is usually triggered by a change in sign of one of the components of G over the time interval. Thus, when triggered, the interpolator must be constructed and then evaluated as requested by the root finder.

2 The Three Body Problem

Three bodies, regarded as point masses, lie on a two-dimensional plane. The gravitational forces between the bodies governs their movement in time. At start time the mass, starting position and starting velocity of each body is given; each of these starting values plays a crucial role in the eventual trajectories of the bodies over time.

The system to solve the three body problem is a relatively simple one of order 12. The system could be solved using forward communication, but for the purposes of illustration, and to allow for high-order interpolation, reverse communication was used.

Interpolation is used to accurately determine the time-zones in which a pair of objects are considered to experience a near-miss. A near-miss constant is supplied and near-miss zones for each of the three pairs of objects is evaluated as the ODE slution proceeds.

A high-order 7(8) Runge-Kuta method is used with global error estimation, and a suitable initial time-step is determined internally.

2.1 Data

```
tstart             = 0.0
tfinal             = 5.5
near_miss          = 0.3
object masses      =  6.0,    5.0,   5.0
starting positions = (1,-1), (1,3), (-2,-1)
velocities         =  0.0,    0.0,   0.0
thresholds         = 0.0e-8
```

2.2 Results

Pairs of objects are tested for near-misses. Pairing 1 is between objects 1 and 2; Pairing 2 is between objects 1 and 3; and Pairing 3 is between objects 2 and 3;

```
Pairing  2 had near-miss at t =  1.6607 dist =  0.3000 Start of near-miss zone
Pairing  2 had near-miss at t =  1.6621 dist =  0.2888
Pairing  2 had near-miss at t =  1.7000 dist =  0.1850
Pairing  2 had near-miss at t =  1.7131 dist =  0.3000 End   of near-miss zone
Pairing  2 had near-miss at t =  3.1540 dist =  0.3000 Start of near-miss zone
Pairing  2 had near-miss at t =  3.1559 dist =  0.2924
Pairing  2 had near-miss at t =  3.2000 dist =  0.2617
Pairing  2 had near-miss at t =  3.2103 dist =  0.3000 End   of near-miss zone
Pairing  1 had near-miss at t =  3.7276 dist =  0.3000 Start of near-miss zone
Pairing  1 had near-miss at t =  3.7290 dist =  0.2841
Pairing  1 had near-miss at t =  3.7446 dist =  0.3000 End   of near-miss zone
Pairing  3 had near-miss at t =  3.8110 dist =  0.3000 Start of near-miss zone
Pairing  3 had near-miss at t =  3.8115 dist =  0.2860
Pairing  3 had near-miss at t =  3.8115 dist =  2.4066 End   of near-miss zone
Pairing  1 had near-miss at t =  3.9157 dist =  0.3000 Start of near-miss zone
Pairing  1 had near-miss at t =  3.9162 dist =  0.2919
Pairing  1 had near-miss at t =  3.9324 dist =  0.3000 End   of near-miss zone
Pairing  1 had near-miss at t =  3.9860 dist =  0.3000 Start of near-miss zone
Pairing  1 had near-miss at t =  3.9868 dist =  0.2978
Pairing  1 had near-miss at t =  3.9932 dist =  0.3000 End   of near-miss zone
Pairing  3 had near-miss at t =  4.1102 dist =  0.3000 Start of near-miss zone
Pairing  3 had near-miss at t =  4.1102 dist =  0.2995
Pairing  3 had near-miss at t =  4.1102 dist =  1.1448 End   of near-miss zone
Pairing  1 had near-miss at t =  4.2563 dist =  0.3000 Start of near-miss zone
Pairing  1 had near-miss at t =  4.2576 dist =  0.2968
Pairing  1 had near-miss at t =  4.2623 dist =  0.3000 End   of near-miss zone
Pairing  1 had near-miss at t =  4.3267 dist =  0.3000 Start of near-miss zone
Pairing  1 had near-miss at t =  4.3284 dist =  0.2895
Pairing  1 had near-miss at t =  4.3422 dist =  0.3000 End   of near-miss zone
Pairing  3 had near-miss at t =  4.4226 dist =  0.3000 Start of near-miss zone
Pairing  3 had near-miss at t =  4.4228 dist =  0.2940
Pairing  3 had near-miss at t =  4.4466 dist =  0.3000 End   of near-miss zone
Pairing  1 had near-miss at t =  4.5049 dist =  0.3000 Start of near-miss zone
Pairing  1 had near-miss at t =  4.5060 dist =  0.2750
Pairing  1 had near-miss at t =  4.5223 dist =  0.3000 End   of near-miss zone
```

2.3 Figures

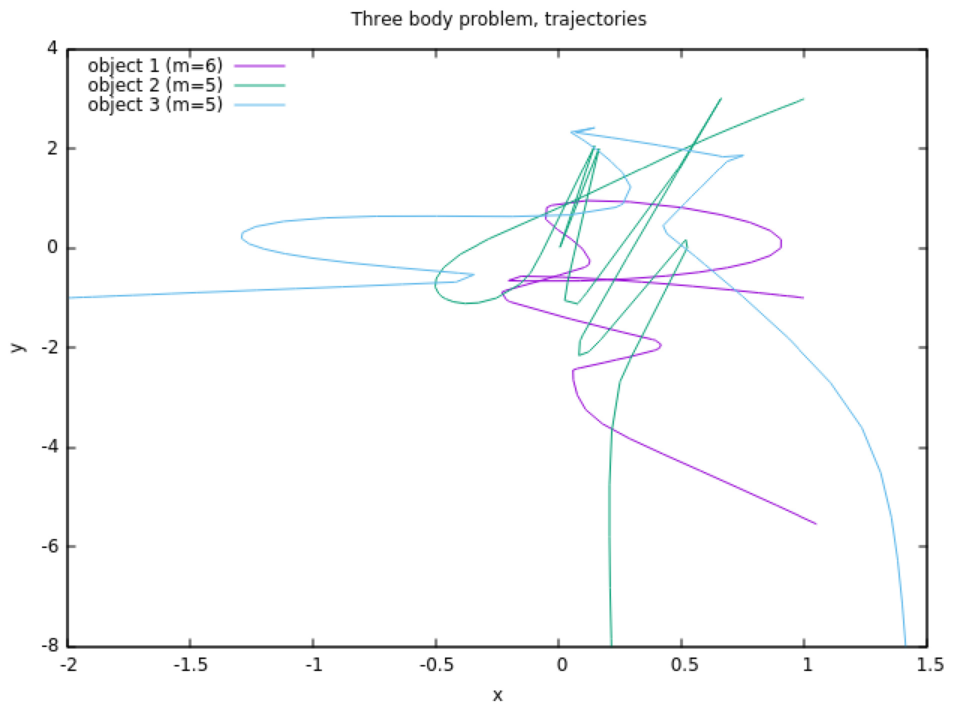

Figure 1: Three body trajectories for the given data

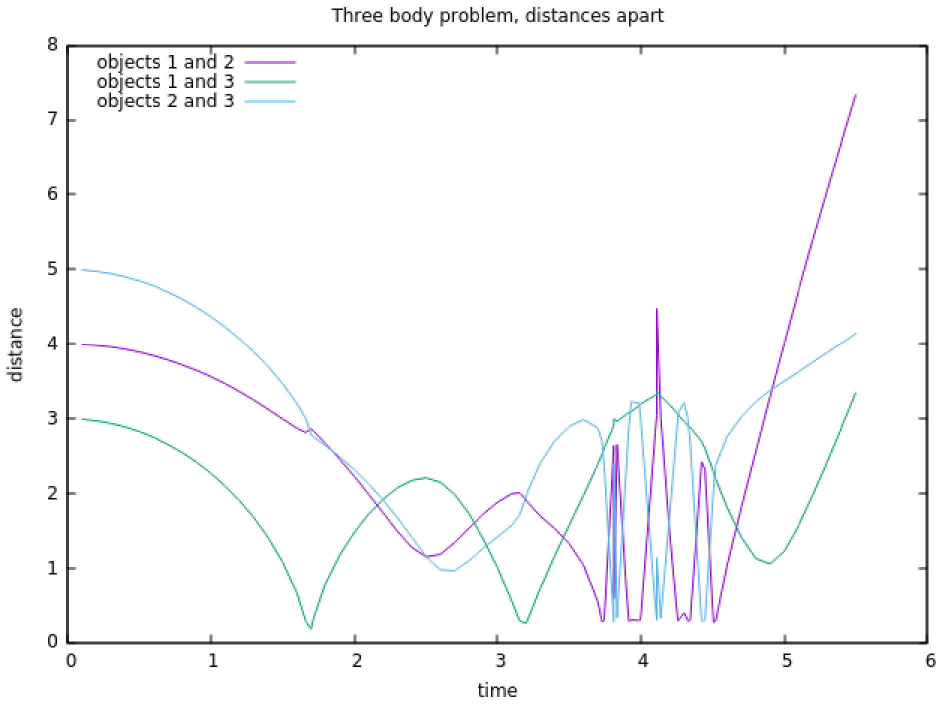

Figure 2: Three body pairing distances