Numerical Methods II

Lab Session 1

February 9, 2004

1. Implement an adaptive Runge–Kutta method using the embedded Prince–Dormand pair of order 4,5 with coefficient table

i	a_i	b_{ij} for $j = 1, \dots, i-1$						
1	0							
2	$\frac{1}{5}$	$\frac{1}{5}$						
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
4	$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
5	$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
6	1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
7	1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
c_j		$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$
$\tilde{c}_j$		$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0

where

$$\boldsymbol{k}_i = \boldsymbol{f}\big(t_n + a_i h, \boldsymbol{y}_n + h(b_{i1}\,\boldsymbol{k}_1 + \cdots + b_{i,i-1}\,\boldsymbol{k}_{i-1})\big)$$

for $i = 1, \dots, 7$, and

$$\begin{aligned} \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\left(c_1\,\boldsymbol{k}_1 + \cdots + c_7\,\boldsymbol{k}_7\right), \\ \tilde{\boldsymbol{y}}_{n+1} &= \boldsymbol{y}_n + h\left(\tilde{c}_1\,\boldsymbol{k}_1 + \cdots + \tilde{c}_7\,\boldsymbol{k}_7\right). \end{aligned}$$

The solver should take arguments of the form

```
[y,t] = ode_rk54 ('f', [t0,t1], y0, tol, Nmax)
```

where `f` is the function $f(t,y)$ on the right hand side of the differential equation, `t0` is the initial time, `t1` is the final time, `y0` the initial condition, `tol` the local error tolerance and `Nmax` the maximum number of steps. The return valus `[y,t]` are the matrix of solution values $(\boldsymbol{y}_0, \dots, \boldsymbol{y}_N)$ and the row vector of evaluation times $(t_0, \dots, t_N)$, respectively.

Test your method with the van der Pol equation

$$\begin{aligned} \dot{x} &= y\,, \\ \dot{y} &= \mu\,(1-x^2)\,y - x\,, \end{aligned}$$

with initial data $x(0) = 2$ and $y(0) = 0$, and $\mu = 10$.

2. Modify your code for the implicit BDF4 solver to use Broyden's method rather than simple iteration to solve the system of nonlinear equations that arises at each time step.

   Recall that Broyden's method for iteratively solving $\boldsymbol{f}(\boldsymbol{x}) = 0$ constructs—in addition to the sequence $\boldsymbol{x}_k$ that (hopefully) converges to the solution $\boldsymbol{x}$—a sequence of approximate Jacobians of $\boldsymbol{f}$ via

$$\begin{aligned} B_k\,\Delta\boldsymbol{x}_k &= -\boldsymbol{f}(\boldsymbol{x}_k)\,, \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \Delta\boldsymbol{x}_k\,, \\ \Delta\boldsymbol{f}_k &= \boldsymbol{f}(\boldsymbol{x}_{k+1}) - \boldsymbol{f}(\boldsymbol{x}_k)\,, \\ B_{k+1} &= B_k + \frac{(\Delta\boldsymbol{f}_k - B_k\,\Delta\boldsymbol{x}_k)\,\Delta\boldsymbol{x}_k^T}{\Delta\boldsymbol{x}_k^T\,\Delta\boldsymbol{x}_k}\,. \end{aligned}$$

   Test your code with the van der Pol equation. How large time steps can you take in the initial part of the computation? How should your initialize your multi-step method?