Goethe-Center for Scientific Computing (G-CSC)
Goethe-Universität Frankfurt am Main

Modeling and Simulation I

(Practical SIM1, WS 2018/19)
and
NeuroBioInformatik
(Übung NBI, WS 2018/19)

M. Huymayer, J. Wang, Dr. A. Nägel, Dr. M. Hoffer

Exercise sheet 4 (Due: Mo., 03.12.2018, 16h)

Until now, we only used constant step sizes h for our explicit and implicit ODE solvers. Although it is relatively easy to find acceptable step sizes for many problems that have been discussed so far, this is not always the case. Especially, if solutions vary strongly in parts of the time interval while they are almost constant in others it is impossible to find a suitable, uniformly distributed step size. Therefore, we will develop an adaptive step size control which ensures that the local discretization error of the solver is bounded by a user-defined, problem-dependent tolerance, usually referred to as TOL. The error estimation strategy we will implement is based on the following idea: Use a numerical method of order $p+1$ to estimate the error of a method of order p which ensures that the estimator is asymptotically correct for $h \to 0$. For the algorithmic treatment in the control loop we employ Algorithm 1, i.e. we have the time step size if required. This algorithm has to be embedded into the outer loop that we used in previous exercises to iterate over the time interval $[t_0, t_n]$.

Aufgabe 1 (10P)
We are going to implement the adaptive step size control using the so called Dormand-Prince 4/5 method (DOPRI). This method is of embedded Runge-Kutta type, i.e., it consists of two Runge-Kutta methods that share the same function evaluations per step, such that the error estimation can be implemented highly efficiently. Embedded Runge-Kutta methods can be conveniently denoted in a Butcher tableau

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$		$\ddots$		
c_L	a_{L1}	a_{L2}	$\dots$	$a_{L,L-1}$	0
	$b_1^{(p+1)}$	$b_2^{(p+1)}$	$\dots$	$b_{L-1}^{(p+1)}$	$b_L^{(p+1)}$
	$b_1^{(p)}$	$b_2^{(p)}$	$\dots$	$b_{L-1}^{(p)}$	$b_L^{(p)}$

and the computation for the next timepoint is carried out for both choices $\mathbf{b}^{(p)}$ and $\mathbf{b}^{(p+1)}$ using

$$\mathbf{k}_l := \mathbf{f}(t_{k-1,l}, \mathbf{y}_{k-1,l}), \qquad l = 1, \dots, L \tag{1}$$

$$t_{k-1,l} := t_{k-1} + c_l h_k, \tag{2}$$

$$\mathbf{y}_{k-1,l} = \mathbf{y}_{k-1} + h_k \sum_{s=1}^{l-1} a_{ls} \mathbf{k}_s, \tag{3}$$

$$\mathbf{y}_k = \mathbf{y}_{k-1} + h_k \sum_{l=1}^{L} b_l \, \mathbf{k}_l, \tag{4}$$

where the single stages $l = 1, ..., L$ can be explicitly computed via

$$\begin{aligned}
\mathbf{k}_1 &= f(t_{k-1}, \mathbf{y}_{k-1}), \\
\mathbf{k}_2 &= f(t_{k-1} + c_2 h_k, \mathbf{y}_{k-1} + h_k(a_{21}\mathbf{k}_1)), \\
&\vdots \\
\mathbf{k}_L &= f(t_{k-1} + c_s h_k, \mathbf{y}_{k-1} + h_k(a_{L1}\mathbf{k}_1 + a_{L2}\mathbf{k}_2 + \ldots + a_{L,L-1}\mathbf{k}_{L-1}))
\end{aligned}$$

The Butcher tableau for the DOPRI method is given by

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$\frac{-56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$\frac{-25360}{2187}$	$\frac{64448}{6561}$	$\frac{-212}{729}$			
1	$\frac{9017}{3168}$	$\frac{-355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$\frac{-5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$\frac{-92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

In order to control the step size, the local error is computed for a test step size h_{test} and it is checked if the error $h_{\text{test}}\|\tilde{\tau}_k(h_{\text{test}})\| \leq TOL$ is below a prescribed tolerance. The test step size is halved $h_{\text{test}} := \frac{1}{2}h_{\text{test}}$ until we can accept the computed solution (or the computation is aborted if a minimum step size $h_{\min}$ is reached). The estimation for the local error can be computed using the difference of the solutions of the two Runge-Kutta methods of order p and $p+1$, namely

$$\begin{aligned} h_{\text{test}}\tilde{\tau}_k(h_{\text{test}}) &= h_{\text{test}}\left\{\Phi^{(p+1)}(\mathbf{y}_{k-1}) - \Phi^{(p)}(\mathbf{y}_{k-1})\right\} \\ &= h_{\text{test}}\sum_{l=1}^{L}(b_l^{(p+1)} - b_l^{(p)})\,\mathbf{k}_l. \end{aligned}$$

For the norm $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$ we use the euclidean norm.
Once a step size $h_k := h_{\text{test}}$ has been accepted with an error $\tilde{\tau}_k \in \mathbb{R}^d$ below the tolerance, $h_{\text{test}}\|\tilde{\tau}\|_k \leq TOL$, the test step size for the next timestep is predicted using the optimal test step size

$$h_{\text{test}} := h_{k,opt} = \min(\max(0.9 \cdot \sqrt[p+1]{\frac{TOL}{h_k\|\tilde{\tau}_k\|}} \cdot h_k, h_{\min}), h_{\max}).$$

The algorithm for finding an acceptable step size in pseudo-code:

Note that $localErrorNorm = h_{\text{test}}\|\tilde{\tau}\|_k$

Algorithm 1 Adaptive Time-Step

```
while localErrorNorm > TOL do
    if h < h_min then
        throw exception (Runtime-Error: h_k < h_min)
    end if
    h = h/2
    TODO: implementation:
     - evaluate dopri
     - update localErrorNorm
end while
```

Hints:

- The `EmbeddedRKTemplate` shows how to return multiple values from a single method (see `http://bit.ly/2gzZBB8`). Please implement your DOPRI-scheme in the template supplied by latter link.

- Use the filter input parameter in `EmbeddedRKTemplate` to control the amount of data points in your trajectory (for example if the filter value is 2 then store every second t_i and u_i)
- Introduce a `stop()` method that allows to stop the solver in case the simulation runs with wrong parameters and to prevent infinite loops introduced by programming errors.
- To save solution trajectories, use the `VectorTrajectoryToFile.groovy` component that can be downloaded from http://bit.ly/2gbflNV.

Tasks/Questions:

(a) Analogously to the previously implemented one-step methods, implement the Dormand-Prince 4/5 scheme with adaptive time step control as specified above. In addition to the solution, return the local error and the step size. See the `EmbeddedRKTemplate` on how to accomplish that.

Aufgabe 2 (3P + 2P)
To test the implementation of your Dormand-Prince 4/5 scheme, we will solve the following problems:

(1)

$$\begin{cases} \text{Find } u : [t_0, t_n] \mapsto \mathbb{R}, \text{ such that} \\ u'(t) = -2ctu^2, c = 100 \\ u(0) = 0.00039968 \\ t_0 = -5, t_n = 5 \end{cases}$$

(2) Let us revisit the Lotka-Volterra model from `Sheet02, Exercise 2c`. Instead of an inhibition factor of 0.001 use 0.003. The initial values of $N_0(t_0) = 1000$ and $P_0(t_0) = 100$ shall be used. For the time step control use $TOL =$ `1e-8`, $h_{\min} =$ `1e-4` and $h_{\max} =$ `1.0`.

Tasks/Questions:

(a) Plot the solution, the local error and the step size for both, 1) and 2). For 1), simulate and plot your results with $TOL =$ `1e-8`, $TOL =$ `1e-10` and $TOL =$ `1e-12`.

(b) Do you see correlations between the solution and the step sizes chosen by the time step control? Discuss your results.

Remark: Send your implemented source code as VRL-Studio project (.vrlp file) and the answers to the questions as plain text in an email. Append the pdfs produced with the TrajectoryPlotter to the email.
Send your solution to `practical.sim1@gcsc.uni-frankfurt.de` until Monday, 03.12.2018, 16h.