Prof. R. Hiptmair
G. Alberti,
F. Leonardi

AS 2015

Numerical Methods for CSE

ETH Zürich
D-MATH

Problem Sheet 12

Problem 1 Three-stage Runge-Kutta method (core problem)

The most widely used class of numerical integratos for IVPs is that of *explicit* Runge-Kutta (RK) methods as defined in [1, Def. 11.4.9]. They are usually described by giving their coefficients in the form of a Butcher scheme [1, Eq. (11.4.11)].

(1a) ⚃ Implement a header-only C++ class `RKIntegrator`

```
template <class State>
class RKIntegrator {
public:
  RKIntegrator(const Eigen::MatrixXd & A,
               const Eigen::VectorXd & b) {
    // TODO: given a Butcher scheme in A,b, initialize RK method for solution of an IVP
  }

  template <class Function>
  std::vector<State> solve(const Function &f, double T,
                           const State & y0,
                           unsigned int N) const {
    // TODO: computes N uniform time steps for the ODE y'(t) = f(y) up to time T of RK method with initial value y0 and store all steps (y_k) into return vector
  }
private:
  template <class Function>
```

```
  void step(const Function &f, double h,
            const State & y0, State & y1) const {
    // TODO: performs a single step from y0 to y1 with
       step size h of the RK method for the IVP with rhs f
  }

  // TODO: hold data for RK methods
};
```

which implements a generic RK method given by a Butcher scheme to solve the autonomous initial value problem $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, $\mathbf{y}(t_0) = \mathbf{y}_0$.

HINT: See `rkintegrate_template.hpp` for more details about the implementation.

(1b) ⚁ Test your implementation of the RK methods with the following data. As autonomous initial value problem, consider the predator/prey model (cf. [1, Ex. 11.1.9]):

$$\dot{y}_1(t) = (\alpha_1 - \beta_1 y_2(t))y_1(t) \tag{39}$$
$$\dot{y}_2(t) = (\beta_2 y_1(t) - \alpha_2)y_2(t) \tag{40}$$
$$\mathbf{y}(0) = [100, 5] \tag{41}$$

with coefficients $\alpha_1 = 3, \alpha_2 = 2, \beta_1 = \beta_2 = 0.1$.

Use a Runge-Kutta single step method described by the following *Butcher scheme* (cf. [1, Def. 11.4.9]):

$$\begin{array}{c|ccc} 0 & 0 & & \\ \frac{1}{3} & \frac{1}{3} & 0 & \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array} \tag{42}$$

Compute an approximated solution up to time $T = 10$ for the number of steps $N = 2^j, j = 7, \ldots, 14$.

Use, as reference solution, $\mathbf{y}(10) = [0.319465882659820, 9.730809352326228]$.

Tabulate the error and compute the experimental order of algebraic convergence of the method.

HINT: See `rk3prey_template.cpp` for more details about the implementation.

Problem 2 Order is not everything (core problem)

In [1, Section 11.3.2] we have seen that Runge-Kutta single step methods when applied to initial value problems with sufficiently smooth solutions will converge algebraically (with respect to the maximum error in the mesh points) with a rate given by their intrinsic order, see [1, Def. 11.3.21].

In this problem we perform empiric investigations of orders of convergence of several explicit Runge-Kutta single step methods. We rely on two IVPs, one of which has a perfectly smooth solution, whereas the second has a solution that is merely piecewise smooth. Thus in the second case the smoothness assumptions of the convergence theory for RK-SSMs might be violated and it is interesting to study the consequences.

(2a) Consider the scalar autonomous ODE

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{43}$$

where $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{y}_0 \in \mathbb{R}^n$. Using the class `RKIntegrate` of Problem 1 write a C++ function

```
template <class Function>
void errors(const Function &f, const double &T, const
    VectorXd &y0, const MatrixXd &A,
const VectorXd &b)
```

that computes an approximated solution $\mathbf{y}_N$ of (43) up to time T by means of an explicit Runge-Kutta method with $N = 2^k$, $k = 1, \ldots, 15$, uniform timesteps. The method is defined by the Butcher scheme described by the inputs `A` and `b`. The input `f` is an object with an evaluation operator (e.g. a lambda function) for arguments of type `const VectorXd &` representing $\mathbf{f}$. The input `y0` passes the initial value $\mathbf{y}_0$.

For each k, the function should show the error at the final point $E_N = \|\mathbf{y}_N(T) - \mathbf{y}_{2^{15}}(T)\|$, $N = 2^k$, $k = 1, \ldots, 13$, accepting $\mathbf{y}_{2^{15}}(T)$ as exact value. Assuming algebraic convergence for $E_N \approx CN^{-r}$, at each step show an approximation of the order of convergence r_k (recall that $N = 2^k$). This will be an expression involving E_N and $E_{N/2}$.

Finally, compute and show an approximate order of convergence by averaging the relevant r_Ns (namely, you should take into account the cases before machine precision is reached in the components of $\mathbf{y}_N(T) - \mathbf{y}_{2^{15}}(T)$).

(2b) ⚂ Calculate the analytical solutions of the logistic ODE (see [1, Ex. 11.1.5])

$$\dot{y} = (1 - y)y, \quad y(0) = 1/2, \tag{44}$$

and of the initial value problem

$$\dot{y} = |1.1 - y| + 1, \quad y(0) = 1. \tag{45}$$

(2c) ⚁ Use the function `errors` from (2a) with the ODEs (44) and (45) and the methods:

- the explicit Euler method, a RK single step method of order 1,
- the explicit trapezoidal rule, a RK single step method of order 2,
- an RK method of order 3 given by the Butcher tableau

$$\begin{array}{c|ccc} 0 & & & \\ 1/2 & 1/2 & & \\ 1 & -1 & 2 & \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

- the classical RK method of order 4.

(See [1, Ex. 11.4.13] for details.) Set $T = 0.1$.

Comment the calculated order of convergence for the different methods and the two different ODEs.

Problem 3 Integrating ODEs using the Taylor expansion method

In [1, Chapter 11] of the course we studied single step methods for the integration of initial value problems for ordinary differential equations $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, [1, Def. 11.3.5]. Explicit single step methods have the advantage that they only rely on point evaluations of the right hand side $\mathbf{f}$.

This problem examines another class of methods that is obtained by the following reasoning: if the right hand side $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ of an autonomous initial value problem

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) , \qquad \mathbf{y}(0) = \mathbf{y}_0 , \tag{46}$$

with solution $\mathbf{y} : \mathbb{R} \to \mathbb{R}^n$ is smooth, also the solution $\mathbf{y}(t)$ will be regular and it is possible to expand it into a Taylor sum at $t = 0$, see [1, Thm. 2.2.15],

$$\mathbf{y}(t) = \sum_{n=0}^{m} \frac{\mathbf{y}^{(n)}(0)}{n!} t^n + R_m(t) \;, \tag{47}$$

with remainder term $R_m(t) = O(t^{m+1})$ for $t \to 0$.

A single step method for the numerical integration of (46) can be obtained by choosing $m = 3$ in (47), neglecting the remainder term, and taking the remaining sum as an approximation of $\mathbf{y}(h)$, that is,

$$\mathbf{y}(h) \approx \mathbf{y}_1 := \mathbf{y}(0) + \frac{d\mathbf{y}}{dt}(0)h + \frac{1}{2}\frac{d^2\mathbf{y}}{dt^2}(0)h^2 + \frac{1}{6}\frac{d^3\mathbf{y}}{dt^3}(0)h^3 \;.$$

Subsequently, one uses the ODE and the initial condition to replace the temporal derivatives $\frac{d^l\mathbf{y}}{dt^l}$ with expressions in terms of (derivatives of) $\mathbf{f}$. This yields a single step integration method called *Taylor (expansion) method*.

(3a) ⚂ Express $\frac{d\mathbf{y}}{dt}(t)$ and $\frac{d^2\mathbf{y}}{dt^2}(t)$ in terms of $\mathbf{f}$ and its Jacobian $\mathbf{Df}$.

HINT: Apply the chain rule, see [1, § 2.4.5], then use the ODE (46).

(3b) ⚄ Verify the formula

$$\frac{d^3\mathbf{y}}{dt^3}(0) = \mathbf{D}^2\mathbf{f}(\mathbf{y}_0)\big(\mathbf{f}(\mathbf{y}_0), \mathbf{f}(\mathbf{y}_0)\big) + \mathbf{Df}(\mathbf{y}_0)^2\mathbf{f}\big(\mathbf{y}_0\big) \;. \tag{48}$$

HINT: this time we have to apply both the product rule [1, (2.4.9)] and chain rule [1, (2.4.8)] to the expression derived in the previous sub-problem.

To gain confidence, it is advisable to consider the scalar case $d = 1$ first, where $f : \mathbb{R} \to \mathbb{R}$ is a real valued function.

Relevant for the case $d > 1$ is the fact that the first derivative of $\mathbf{f}$ is a linear mapping $\mathbf{Df}(\mathbf{y}_0) : \mathbb{R}^n \to \mathbb{R}^n$. This linear mapping is applied by multiplying the argument with the Jacobian of $\mathbf{f}$. Similarly, the second derivative is a *bilinear* mapping $\mathbf{D}^2\mathbf{f}(\mathbf{y}_0) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$. The i-th component of $\mathbf{D}^2\mathbf{f}(\mathbf{y}_0)\big(\mathbf{v}, \mathbf{v}\big)$ is given by

$$\mathbf{D}^2\mathbf{f}(\mathbf{y}_0)\big(\mathbf{v}, \mathbf{v}\big)_i = \mathbf{v}^T\mathbf{Hf}_i(\mathbf{y}_0)\mathbf{v},$$

where $\mathbf{H}f_i(\mathbf{y}_0)$ is the Hessian of the i-th component of $\mathbf{f}$ evaluated at $\mathbf{y}_0$.

(3c) We now apply the Taylor expansion method introduced above to the *predator-prey* model (39) introduced in Problem 1 and [1, Ex. 11.1.9].

To that end write a header-only C++ class `TaylorIntegrator` for the integration of the autonomous ODE of (39) using the Taylor expansion method with uniform time steps on the temporal interval $[0, 10]$.

HINT: You can copy the implementation of Problem 1 and modify only the `step` method to perform a single step of the Taylor expansion method.

HINT: Find a suitable way to pass the data for the derivatives of the r.h.s. function $\mathbf{f}$ to the `solve` function. You may modify the signature of `solve`.

HINT: See `taylorintegrator_template.hpp`.

(3d) Experimentally determine the order of convergence of the considered Taylor expansion method when it is applied to solve (39). Study the behaviour of the error at final time $t = 10$ for the initial data $\mathbf{y}(0) = [100, 5]$.

As a reference solution use the same data as Problem 1.

HINT: See `taylorprey_template.cpp`.

(3e) What is the disadvantage of the Taylor method compared with a Runge-Kutta method?

Problem 4 System of ODEs

Consider the following initial value problem for a second-order system of ordinary differential equations:

$$
\begin{aligned}
2\ddot{u}_1 - \ddot{u}_2 &= u_1(u_2 + u_1) \, , \\
-\ddot{u}_{i-1} + 2\ddot{u}_i - \ddot{u}_{i+1} &= u_i(u_{i-1} + u_{i+1}) \, , \qquad i = 2, \dots, n-1 \, , \\
2\ddot{u}_n - \ddot{u}_{n-1} &= u_n(u_n + u_{n-1}) \, , \\
u_i(0) &= u_{0,i} \qquad i = 1, \dots, n \, , \\
\dot{u}_i(0) &= v_{0,i} \qquad i = 1, \dots, n \, ,
\end{aligned}
\tag{49}
$$

in the time interval $[0, T]$.

(4a) Write (49) as a first order IVP of the form $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, $\mathbf{y}(0) = \mathbf{y}_0$ (see [1, Rem. 11.1.23]).

(4b) ⚁ Apply the function `errors` constructed in Problem 2 to the IVP obtained in the previous subproblem. Use

$$n = 5 , \qquad u_{0,i} = i/n , \qquad v_{0,i} = -1 , \qquad T = 1,$$

and the classical RK method of order 4. Construct any sparse matrix encountered as a sparse matrix in EIGEN. Comment the order of convergence observed.

Issue date: 03.12.2015

Hand-in: 10.12.2015 (in the boxes in front of HG G 53/54).

Version compiled on: December 6, 2015 (v. 1.0).

References

[1] R. Hiptmair. *Lecture slides for course "Numerical Methods for CSE"*. http://www.sam.math.ethz.ch/~hiptmair/tmp/NumCSE/NumCSE15.pdf. 2015.