C1 MA913 Scientific Computing 2015/2016

Assignment 2

Assignment marking and evaluation. This assignment will count 15% of the total mark. Please notice that the sum of all the scores associated with each task is greater than 15 (approximately 20). This means you can get the maximum even if some tasks/questions/results are missing. The choice of which question or task to skip will be evaluated based on the justification provided in the report and is part of evaluation of your overall understanding, critical thinking, and conciseness presentation of results. No mark above 15% can be given in any case.

Scalar initial value problems. Write a program to solve a scalar ordinary differential equation

$$u'(t) = f(u(t), t) \quad u(0) = u_0$$

using the following five DIRK (Diagonally Implicit Runge-Kutta) methods:

Forward Euler (FE)

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}.$$

Backward Euler (BE)

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}.$$

Crank Nicholson (CN)

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}.$$

3 stage Heun (Heun3)

$$\begin{array}{c|ccc} 0 & & & \\ 1/3 & 1/3 & & \\ 2/3 & 0 & 2/3 & \\ \hline & 1/4 & 0 & 3/4 \end{array}.$$

2 stage DIRK method (DIRK2)

$$\begin{array}{c|cc} \delta & \delta & \\ 1-\delta & 1-2\delta & \delta \\ \hline & \frac{0.5-\delta}{1-2\delta} & \frac{0.5-\delta}{1-2\delta} \end{array}. \quad \text{with } \delta = 0.5 + \frac{\sqrt{3}}{6}.$$

Consider the following right hand side functions:

(1) $f(u,t) = 1 - \cos(t)(\cos(t) - 1) - u^2$ and $u_0 = 0$, $T = 2\pi$ with exact solution $u = \sin(t)$.
(2) $f(u,t) = 1 - \cos(t)(\cos(t) - \exp(\lambda t)) - \exp(-2\lambda t)u^2 + \lambda u$ and $u_0 = 0$, $T = 10$ with exact solution $u = \exp(\lambda t)\sin(t)$; use $\lambda = -0.1$.

A method to test your implementation is to compare the *experimental order of convergence* (eoc) with the convergence rate given by the error analysis of the RK method. By computing the error for a sequence of time steps $(\tau_j)_{j=0}^J$ one can compute the eoc of the scheme given by

$$\text{eoc}_j = \frac{\log\left(\frac{e_{\tau_j}}{e_{\tau_{j-1}}}\right)}{\log\left(\frac{\tau_j}{\tau_{j-1}}\right)} \quad \text{for } j = 1, \ldots, J.$$

where e_{τ_j} is the discretization error with the time step τ_j. The eoc should (for small enough time steps) correspond with the theoretical convergence rate of the scheme. But keep in mind that the scheme is not necessarily stable for all time steps used.

For each combination of right hand side and scheme and given time step τ, compute the maximum error over all time steps: $e_\tau = \max_{n=1,\ldots,N} |u(t^n) - u^n|$. and compare the experimental order of convergence with the theoretical value (which you should first determine). Your code should output a table giving the errors and eocs for the time step sequence $\tau_j = 2^{-j}\tau_0$ with given starting time step $\tau_0 > 0$. Your code should compute the eoc, i.e., should include the loop over j and produce the error and eoc table. As command line arguments the user should be able to choose which method and which right hand side to use, as well as the first time step τ_0 and the number J of eocs to compute. For your test you can use $\tau_0 = 0.1, J = 12$.
In addition to the table, your report should include plots of the approximation error over time for two different time step sizes. You might want to use a log scale for the y-axis (error).
If you count evaluations of f and derivatives of f (lets say thy both cost one unit of time), then which method is the most efficient?
For this assignment we are providing a starting point for your implementation.

- Download link

You can extend the code in the following steps:

(1) add computation of eoc
(2) add virtual interface class ModelInterface and add the additional problems (see this link for a simple introduction to virtual methods
(3) Implement the method DIRK::evolve for arbitrary DIRK methods using the a ,b ,c vectors and implementing the Newton iteration.
(4) Define classes FE, BE, CN, Heun3, DIRK2 which derives from DIRK and have an empty constructor calling the constructor from DIRK with correct number of steps and in which the arrays a ,b ,c are initialized

External libraries. Try to solve the same problems with the ODE library `odeint` with a few different available schemes. See documentation at the link

- `http://headmyshoulder.github.io/odeint-v2/doc/index.html`

particularly the Getting Started section, the first tutorial (that shows how to store the variables at each time step in a container), and the section about Steppers (to see available schemes) and Integrate functions (to know how to integrate the equations).
Like in the assignment 1, we expect you to practice the usage of these external libraries, compare and comment on its functionalities, test their performances.

Submission. For the submission follow the same instructions provided with assignment 1

Deadline: Mon 30 November 2015 6:00 pm