Assignment 3: Scalar initial value problems

The following is a brief report concerning the implementation of a C++ program to solve certain Initial Value Problems.

1 Runge-Kutta methods

In this assignment we will be considering *Runge-Kutta* (RK) methods to solve the scalar ordinary differential equation

$$u'(t) = f(u(t), t) \quad u(0) = u_0. \tag{1}$$

Let us define $I \coloneqq [t_0, t_0 + T]$ and discretise it with N uniform intervals $[t_n, t_{n+1}]$, such that $t_n \coloneqq t_0 + nh$. At each point t_i we let u_i be the approximation of our solution $y(t_i) = y_i$. Moreover we define $f(t_i, u_i) = f_i$. We say that a numerical method is a *one-step* method if they approximate the initial value problem (1) such that u_{n+1} only depends on u_n for all $n \geq 0$. In particular, the RK methods are one-step methods, and we are now in a position to define them.

Definition: *Runge-Kutta* methods are one-step methods of the form

$$u_{n+1} = u_n + hF(t_n, u_n, f, h),$$

where F is defined as

$$\begin{aligned} F(t_n, u_n, f, h) &= \sum_{i=1}^{s} b_i K_i, \\ K_i &= f(t_n + c_i h, u_n + h \sum_{j=1}^{s} a_{ij} K_j), \quad i \in \{1, \ldots, s\}. \end{aligned}$$

The integer s denotes the number of stages of the method.

Note that the coefficients $A = (a_{ij}) \in \mathbb{R}^{s \times s}$, $\mathbf{b}$, $\mathbf{c} \in \mathbb{R}^s$ completely determine the RK method. It follows that we can describe the method completely by what is known as its *Butcher Tableau*, which are of the form

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

We will be utilising the following five RK schemes:

- Forward Euler (FE):

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

- Backwards Euler (BE):

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

- Implicit Modpoint (IM):

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

- 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):

$$\begin{array}{c|cc} \delta & \delta & \\ 1-\delta & 1-2\delta & \delta \\ \hline & 1/2 & 1/2 \end{array}$$

1.1 Scheme Properties

Definition: A method is called explicit, if u_{n+1} can be computed directly in terms of previous values u_k, $k < n$. It is called implicit, if u_{n+1} implicitly depends on itself through f.

Note that the schemes will either be explicit or diagonally implicit, with the schemes categorised as follows:

Explcit	Implicit
FE	BE
Heun3	IM
	DIRK2

For the implicit cases, we will have to solve the implicit equation,

$$K_i = f(t + hc_i, u_n + h\sum_{j=1}^{i-1} a_{ij}K_j + ha_{ii}K_i)$$

in order to run the schemes. Since this equation only depends on previous K_i and itself, we can solve it stage by stage by utilising the *Newton Raphson* method for finding roots.

In addition to this, we can use **Theorem 4.3.9** from the lecture notes to deduce the minimum order of the schemes we are using. They are included in the table below.

Scheme	Order
FE	1
BE	1
IM	2
Heun3	3
DIRK2	2

1.2 Implementation

Equation (1) will be considered for two different right hand functions $f(y,t)$ defined as follows:

1. $f(y,t) = 1 - cos(t)(cos(t) - 1) - y^2; \quad y_0 = 0,\ T = 2\pi,$
   with exact solution $y = sin(t)$.

2. $f(y,t) = 1 - cos(t)(cos(t) - exp(\lambda t)) - exp(-2\lambda t)y^2 + \lambda y; \quad y_0 = 0,\ T = 10,$
   with exact solution $y = exp(-\lambda t)sin(t)$ and $\lambda = -0.1$.

We aim to investigate the convergence rate of the five RK methods for these two test cases, which we will denote as *Test 1* and *Test 2* respectively. To do this we compare the *experimental order of convergence* (eoc) with the theoretical order of convergence. For a series of time steps $(\tau_j)_{j=0}^{J}$ we define the eoc of the scheme by

$$eoc_j = \frac{\log(\frac{e_{\tau_j}}{e\tau_{j-1}})}{\log(\frac{\tau_j}{\tau_{j-1}})},$$

where e_{τ_j} is the discretisation error with the time-step τ_j.

For this assignment I have implemented two different header files, *scheme.hh* and *models.hh*. The first file includes a class *DIRK* which defines a general RK method, and speific inhertience classes for the individual shcemes. The class also includes a function *evolve* which updates the approximated solution at each time-step for a given scheme. Included in this function is an implementation of the Newton-Raphson method which is needed for calculating the K_i for implicit schemes. The latter header files includes a structure which defines the models relating to the two right-hand function defined at the beginning of this sub-section.

With these files in place, the included *main.cc* file allows the user to input the test, scheme, step-size and number of different time-steps (levels) via the terminal. The program is designed to compute the experimental order of convergence, with added functionality to automatically create data files needed to plot graphs of the error over time. The results from this implementation will be considered in the next section. In addition to this, the maximum error for each time-step and the corresponding eocs are outputted to the terminal.

2 Testing

In this section we look at the results of the tests which have been implemented by the C++ program. We first look at the errors.

2.1 Maximum Error and EOC

Recall the maximum error is defined to be $e_{\tau_j} = \max_{n=1,\ldots,N}|y_n - u_n|$, such that N is the total number of time-steps and u_n (resp. y_n) is the approximated (resp. exact) solution at time-step n. The following tables contains the maximum error and the experimental order of convergence for *test 1* and each scheme, for a sequence of time-steps $\{\tau_j\}_{j=0}^{J}$. In particular, we have $\tau_j = \tau_0 2^{-j}$, $\tau_0 = 0.1$ and $J = 12$. Note that in this subsection *the error* will be referring to the maximum error.

Test 1: FE

data.txt

```
Tau                 Error               EOC
0.1                 0.126438
0.05                0.035277            1.84163
0.025               0.00949667          1.89323
0.0125              0.00304776          1.63967
0.00625             0.00152847          0.995661
0.003125            0.000765525         0.997569
0.0015625           0.000383107         0.9987
0.00078125          0.000191643         0.999326
0.000390625         9.58445e-05         0.999656
0.000195313         4.7928e-05          0.999826
9.76563e-05         2.39654e-05         0.999913
4.88281e-05         1.19831e-05         0.999956
2.44141e-05         5.99164e-06         0.999978
```

Test 1: BE

data.txt

```
Tau                 Error               EOC
0.1                 0.396107
```

```
0.05                0.055257            2.84166
0.025               0.0117679           2.2313
0.0125              0.00310423          1.92255
0.00625             0.00154073          1.01062
0.003125            0.000768528         1.00345
0.0015625           0.000383855         1.00154
0.00078125          0.00019183          1.00073
0.000390625         9.58911e-05         1.00036
0.000195313         4.79397e-05         1.00018
9.76563e-05         2.39684e-05         1.00009
4.88281e-05         1.19838e-05         1.00004
2.44141e-05         5.99181e-06         1.00002
```

Test 1: IM

data.txt

```
Tau                 Error               EOC
0.1                 0.058537
0.05                0.0151887           1.94635
0.025               0.00383394          1.9861
0.0125              0.000960821         1.99649
0.00625             0.000240347         1.99915
0.003125            6.00952e-05         1.9998
0.0015625           1.50244e-05         1.99995
0.00078125          3.75613e-06         1.99999
0.000390625         9.39042e-07         1.99999
0.000195313         2.3469e-07          2.00044
9.76563e-05         5.86338e-08         2.00095
4.88281e-05         1.49431e-08         1.97226
2.44141e-05         3.88959e-09         1.94179
```

Test 1: Heun3

data.txt

```
Tau                 Error               EOC
0.1                 0.000312788
0.05                1.96209e-05         3.99473
0.025               1.22761e-06         3.99847
0.0125              8.95918e-08         3.77634
0.00625             1.10018e-08         3.02563
0.003125            1.36331e-09         3.01255
0.0015625           1.69549e-10         3.00733
0.00078125          2.2073e-11          2.94135
0.000390625         1.39465e-11         0.662379
0.000195313         6.74941e-11         -2.27486
9.76563e-05         5.50012e-11         0.295299
4.88281e-05         2.70395e-10         -2.29754
2.44141e-05         2.21423e-10         0.288267
```

Test 1: DIRK2

data.txt

```
Tau                 Error               EOC
0.1                 0.00167294
0.05                0.000100321         4.05968
0.025               6.20988e-06         4.01392
0.0125              3.88116e-07         4
0.00625             3.73366e-08         3.37782
0.003125            4.47351e-09         3.06111
0.0015625           4.5313e-10          3.30341
0.00078125          8.35638e-11         2.43898
0.000390625         1.35223e-11         2.62754
0.000195313         6.77e-11            -2.32382
9.76563e-05         5.54452e-11         0.288094
4.88281e-05         2.70635e-10         -2.28721
2.44141e-05         2.21824e-10         0.286929
```

This data seems suggest that the smallest errors occur when we use the Heun3 scheme. Moreover, these errors decrease with each time-step in the sequence $\{\tau_j\}_{j=0}^{J}$ at a fast rate. This aligns with the theory, since Heun3 has the highest theoretical minimum convergence rate. The FE and BE produce very similar results and give the largest errors. This also aligns with the theory, since they have lowest minimum convergence order of all the schemes. DIRK2 produces errors which are much smaller than the IM scheme, although they both have minimum theoretical order 2. Note that this does not go against the theory, since it is the *minimum* order.

The error of convergence largely matches the order of the schemes deduced by the theory. Indeed, for the FE and BE schemes we have an eoc which suggest and order of 1. For the IM scheme we have an eoc which suggest an order of 2. There is a slight discrepancy with the Heun3 and DIRK2 schemes. For Heun3, the suggested order from the eoc is higher than the theoretical order for larger time-steps and much lower (even negative) for the last five smallest time-steps. These odd negative results can be attributed to an increase in error as time-steps decrease later on, since in these cases $\log(\frac{e_{\tau_j}}{e\tau_{j-1}}) > 0$ and $\log(\frac{\tau_j}{\tau_{j-1}}) < 0$. The eoc mostly matches the theoretical result for the DIRK2 scheme (for larger time-steps), but again, we get stranger results for smaller time-steps.

Next, we analyse the same results for *Test 2* case.

Test 2: FE

data.txt

```
Tau                 Error               EOC
0.1                 inf
```

```
0.05                inf                 -nan
0.025               inf                 -nan
0.0125              inf                 -nan
0.00625             0.481524            inf
0.003125            0.132811            1.85823
0.0015625           0.057307            1.2126
0.00078125          0.0269536           1.08823
0.000390625         0.013101            1.0408
0.000195313         0.0064618           1.01967
9.76563e-05         0.00320933          1.00967
4.88281e-05         0.00159934          1.00479
2.44141e-05         0.000798349         1.00239
```

Test 2: BE

data.txt

```
Tau                 Error               EOC
0.1                 0.639798
0.05                0.495569            0.368531
0.025               0.358309            0.467881
0.0125              0.239848            0.579082
0.00625             0.148207            0.694512
0.003125            0.0851809           0.799009
0.0015625           0.0462974           0.8796
0.00078125          0.0242502           0.93293
0.000390625         0.0124282           0.964386
0.000195313         0.00629377          0.981617
9.76563e-05         0.00316733          0.990657
4.88281e-05         0.00158885          0.995288
2.44141e-05         0.000795727         0.997634
```

Test 2: IM

data.txt

```
Tau                 Error               EOC
0.1                 0.209799
0.05                0.0725915           1.53113
0.025               0.0203657           1.83366
0.0125              0.00526197          1.95247
0.00625             0.0013267           1.98777
0.003125            0.000332424         1.99674
0.0015625           8.31018e-05         2.00008
0.00078125          2.07828e-05         1.99949
0.000390625         5.19649e-06         1.99978
0.000195313         1.29881e-06         2.00035
9.76563e-05         3.24474e-07         2.00101
4.88281e-05         8.2711e-08          1.97195
2.44141e-05         2.16242e-08         1.93543
```

Test 2: Heun3

data.txt

```
Tau                 Error               EOC
0.1                 0.00208592
0.05                6.50329e-05         5.00337
0.025               4.32878e-06         3.90914
0.0125              1.31976e-06         1.71369
0.00625             2.13607e-07         2.62724
0.003125            2.97629e-08         2.84337
0.0015625           3.92832e-09         2.92153
0.00078125          4.06279e-10         3.27337
0.000390625         1.98373e-11         4.35618
0.000195313         3.85881e-10         -4.28187
9.76563e-05         3.31041e-10         0.221147
4.88281e-05         1.50893e-09         -2.18845
2.44141e-05         1.31989e-09         0.193108
```

Test 2: DIRK2

data.txt

```
Tau                 Error               EOC
0.1                 0.00280039
0.05                0.000763438         1.87505
0.025               0.000154921         2.30098
0.0125              2.29212e-05         2.75678
0.00625             3.11024e-06         2.88158
0.003125            4.39083e-07         2.82446
0.0015625           1.01298e-07         2.11588
0.00078125          1.11014e-08         3.1898
0.000390625         1.31348e-09         3.07927
0.000195313         5.51325e-10         1.25242
9.76563e-05         3.5161e-10          0.648929
4.88281e-05         1.50797e-09         -2.10056
2.44141e-05         1.32136e-09         0.190593
```

This data also seems to suggest that the smallest errors occur when we use the Heun3 scheme, closely followed by DIRK2. The IM follows behind these two. FE and BE give the poorest errors. This is all as expected. For the FE scheme, we note that the error is ambiguous for the first four time-steps. This is not too concerning, as we can only guarantee convergence as the time-steps τ tend towards zero. As the time-steps get smaller, the FE and BE schemes give similar errors.
The eoc varies more widely for Test 2. Due to the errors *inf* found in the table for FE, the first four eoc values are unstable also. Afterwards, they seems to match the expected order. Although the eoc for BE seems inconsistent with the theory for larger time-steps, it is roughly 1 as expected for smaller time-steps. The IM scheme gives an eoc which is roughly 2 for most time-steps as expected.

It is hard to see an average for the eoc values in the tables for Heun3 and DIRK2 due to large fluctuations.

2.2 Plots

For a given time-step τ, we will now be considering the error between the approximate solution u_n and the exact solution y_n at each of the discretisation points $t_n = t_0 + n\tau$ for $n = 1, \dots, N$, which we defined at the beginning of section 1. We may write this error succinctly as $e_n = |u_n - y_n|$ for $n = 1, \dots, n$. It follows that $e_\tau = \max_{n=1,\dots,N} e_n$. Note that in this subsection, for a given n, *the error* will be referring to e_n.

Below are graphs which plot the error for Test 1 and 2 over time, for time-step sizes $\tau = 0.1$ and $\tau = 0.01$. Note that for Test 1 (resp. Test 2) we will be running over the interval $[0, 2\pi]$ (resp. $[0, T]$). For the first test, the graphs have been plotted for all of the schemes. For Test 2, the graphs have been plotted for all schemes apart from FE, due to the instability we encountered in the previous subsection. For readability, the y-axis (which describes the error) is set in log-scale for readability. For each scheme, the maximum error which we considered in subsection 2.1 will correspond to the global maximum of it's respective graph.

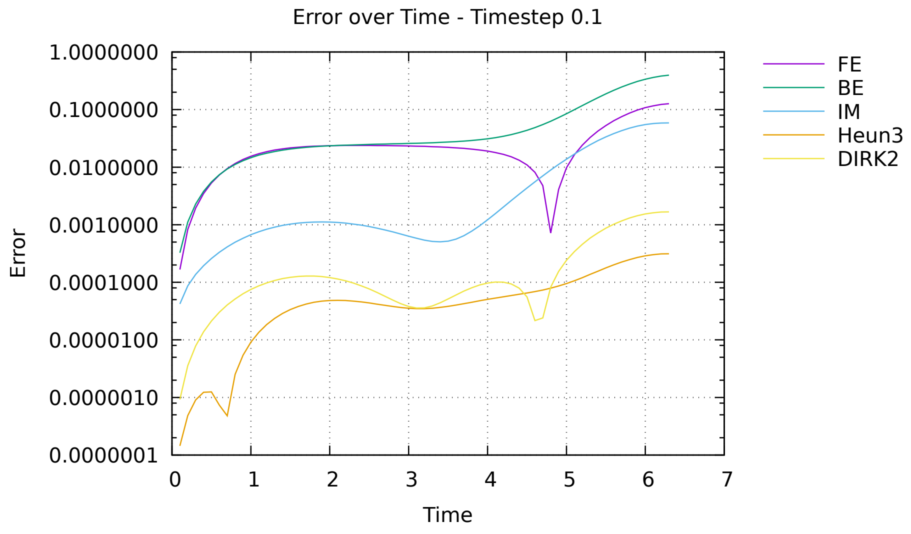

Figure 1: Test 1: time-step $\tau = 0.1$.

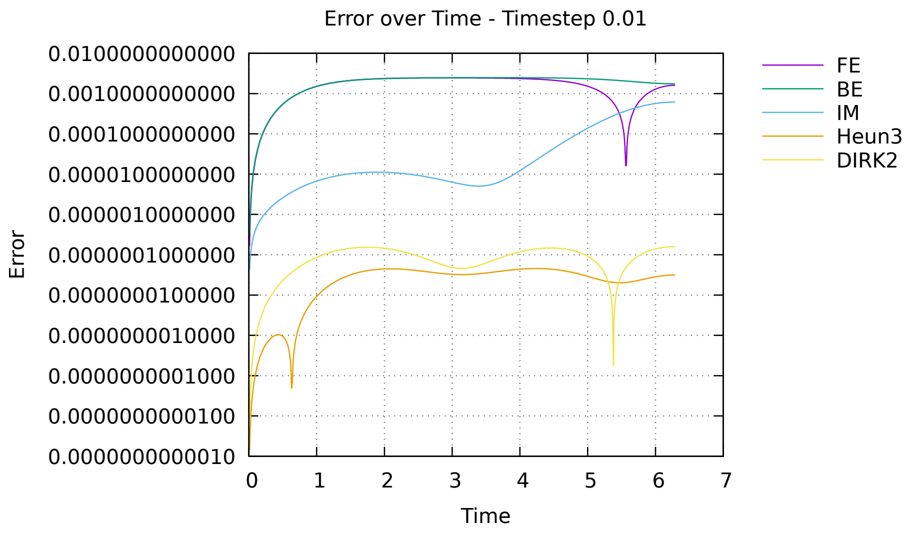

Figure 2: Test 1 : time-step $\tau = 0.01$.

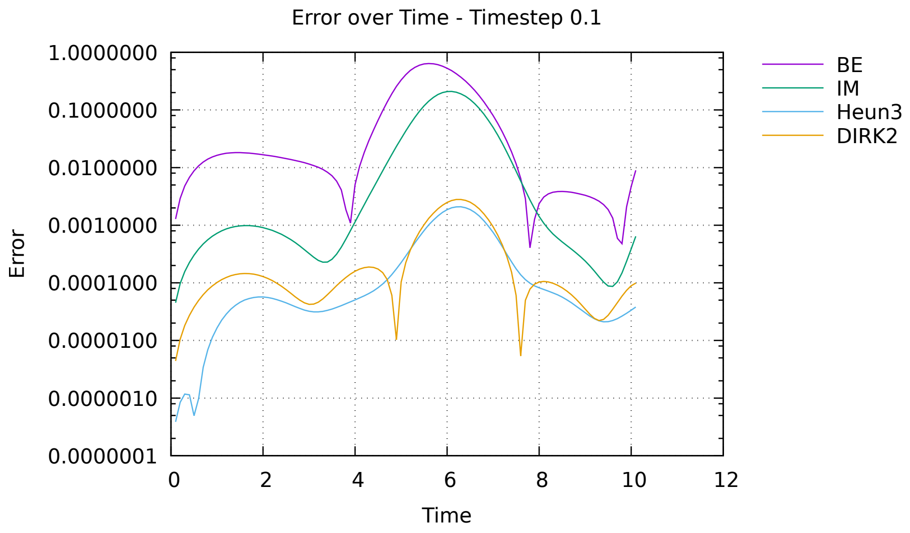

Figure 3: Test 2: time-step $\tau = 0.1$.

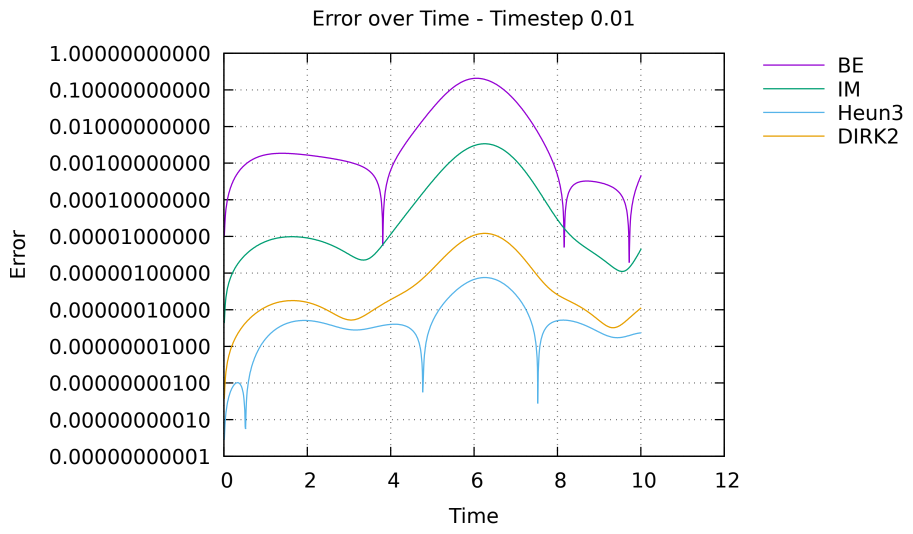

Figure 4: Test 2: time-step $\tau = 0.01$.

2.3 The Number of Evaluations

We now look at how many times we need to evaluate f and df during the course of running the C++ code for each scheme. We do this by implementing a counter which increases by 1 at each evaluation. This total is outputted to the terminal for each time-step in the sequence $\{\tau_j\}_{j=0}^{12}$, which we defined earlier. By comparing the counters, we can conjecture which scheme is the most efficient. Noting that the general pattern is the same from any chosen time-step, I have only included a table for the counter of each scheme (for Test 1 and Test 2) for $\tau = 0.1$. These results can be seen below.

$\tau = 0.1$	FE	BE	IM	Heun3	DIRK2
Counter for Test 1	63	382	378	189	1004
Counter for Test 2	101	674	606	303	1650

For both tests, the FE scheme seems to be the most efficient. Indeed, the schemes FE and Heun3 are more efficient than the rest, due to the fact that they are explicit (the implicit schemes need numerous evaluations of f and df due to the Newton-Raphson implementation). We note that DIRK2 is the seems to be the least efficient scheme, although it has a better theoretical order than FE. I would conjecture that Heun3 is the best scheme to choose for implementation since the data suggest it has the best order of all of the schemes and, except for FE, is the most efficient scheme.