Programming Report for Project 3

Kangjie Ding

2022.5.31

Programming design

We'll implement all the methods required in Section 11.7.2 and test them according to the requirements (a)-(c).In this project,we'll also apply a factory pattern and a singleton pattern to optimize our program,and all the necessary declarations are kept in generator.h.

Results

1. **Solution errors,convergence rates and CPU time for different methods**

   (a) tests for (10.198)

   - Euler's method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
10000	41.804		0.009149
20000	2.76105	0.066051	0.016594
40000	0.460488	0.166783	0.024498
80000	0.710865	1.543726	0.049588
160000	0.509269	0.716411	0.1067

   - the classical RK method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
10000	0.0180875		0.033445
20000	0.00246481	0.136271	0.045851
40000	0.000139935	0.056773	0.099819
80000	9.15436e-06	0.065419	0.178323
160000	1.54285e-06	0.168353	0.342385

   - Gauss-Legendre RK method with s=1

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
10000	1.72445		0.030225
20000	0.442179	0.256417	0.054751
40000	0.0983457	0.222411	0.096718
80000	0.00693685	0.070535	0.176873
160000	0.00705091	1.00000*	0.338933

   - Fehlberg 4(5) embedded RK method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
10000	0.00382153		0.040264
20000	1.39515e-05	0.003650	0.068657
40000	1.42501e-06	0.102140	0.138406
80000	9.52533e-07	0.668440	0.264566
160000	1.05172e-06	0.011099	0.518126

   - Dormand-Prince 5(4) embedded RK method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
10000	0.00127913		0.044351
20000	0.000174498	0.136419	0.083627
40000	8.6347e-06	0.049483	0.173724
80000	1.0259e-06	0.118811	0.298788
160000	5.35727e-07	0.522202	0.618553

   (b) tests for (10.199)

- Euler's method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
10000	11.8942		0.005284
20000	0.649077	0.054571	0.020699
40000	0.569678	0.877674	0.02827
80000	0.310928	0.545796	0.053841
160000	0.200387	0.644480	0.102206

- the classical RK method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
500	0.0117078		0.001881
1000	0.000238359	0.020359	0.00243
2000	0.000172768	0.724823	0.005358
4000	0.000181848	1.00000*	0.010152
8000	0.000182056	1.00000*	0.022913

- Gauss-Legendre RK method with s=1

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
500	0.0278584		0.003664
1000	0.0146671	0.526488	0.004968
2000	0.00749116	0.510746	0.010345
4000	0.00202822	0.270748	0.023154
8000	0.000644781	0.317905	0.029239

- Fehlberg 4(5) embedded RK method

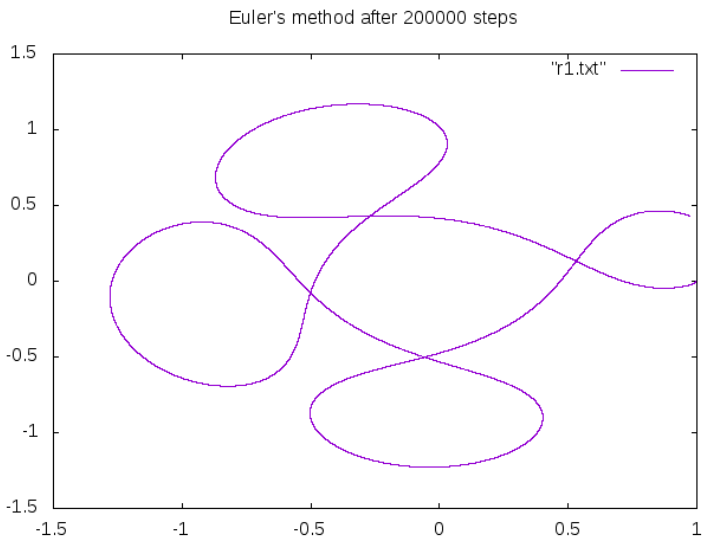
time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
500	0.00156061		0.001725
1000	0.000252416	0.161742	0.003665
2000	0.000184495	0.730916	0.008608
4000	0.000182136	1.00000*	0.021026
8000	0.00018206	1.00000*	0.025361

- Dormand-Prince 5(4) embedded RK method

time steps\items	$\|\|E^N\|\|_\infty$	rate	CPU time(s)
500	0.00154896		0.002304
1000	0.000241623	0.155997	0.007185
2000	0.000183923	0.761198	0.010467
4000	0.000182107	1.00000*	0.014415
8000	0.000182059	1.00000*	0.03725

2. **The step size for plot of the solutions being indistinguishable from the exact plot**

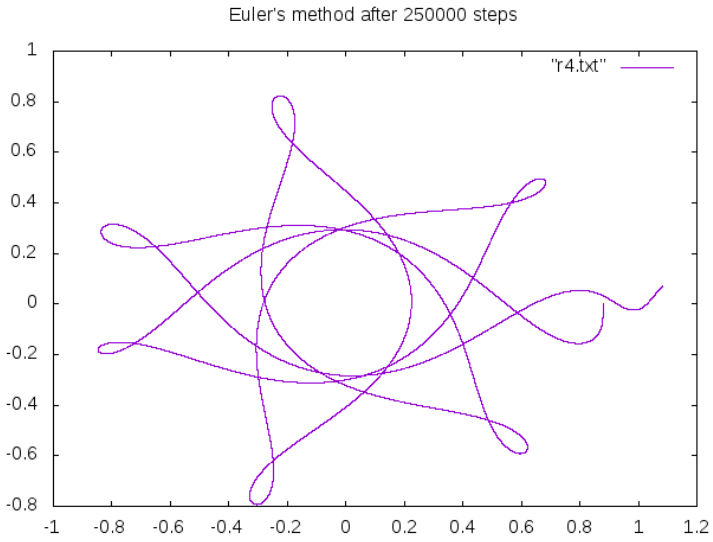
(a) tests for (10.198)
Due to its low accuracy order and not being L-stable,we can not decide a possible step size under allocated memory for Euler's method to make its plot visually indistinguishable,and the figure below shows the plot for 200000 steps($k \approx 8.53 \times 10^{-5}$):



For other methods we list a tableau below to show the corresponding step size.

method	step size
the classical RK method	$1.71 \times 10^{-3}(steps \approx 10000)$
Gauss-Legendre RK method with s=1	$2.43 \times 10^{-4}(steps \approx 70000)$
Fehlberg 4(5) embedded RK method	2.43×10^{-3}
Dormand-Prince 5(4) embedded method	2.84×10^{-3}

(b) tests for (10.199)
Just as talked earlier,there always exists a obvious gap at the end point for Euler's method,and the figure below shows the results after 250000 steps:



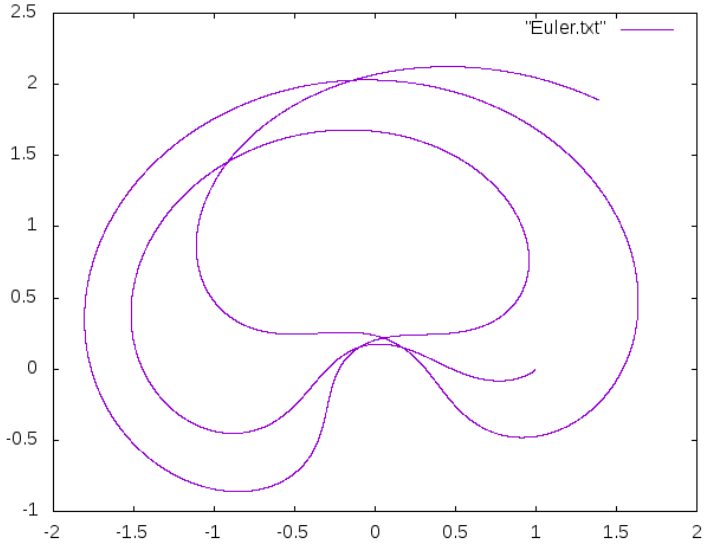
We also list a tableau for other methods.

method	step size
the classical RK method	$0.0273(steps \approx 700)$
Gauss-Legendre RK method with s=1	$9.57 \times 10^{-3}(steps \approx 2000)$
Fehlberg 4(5) embedded RK method	0.0638
Dormand-Prince 5(4) embedded method	0.0684

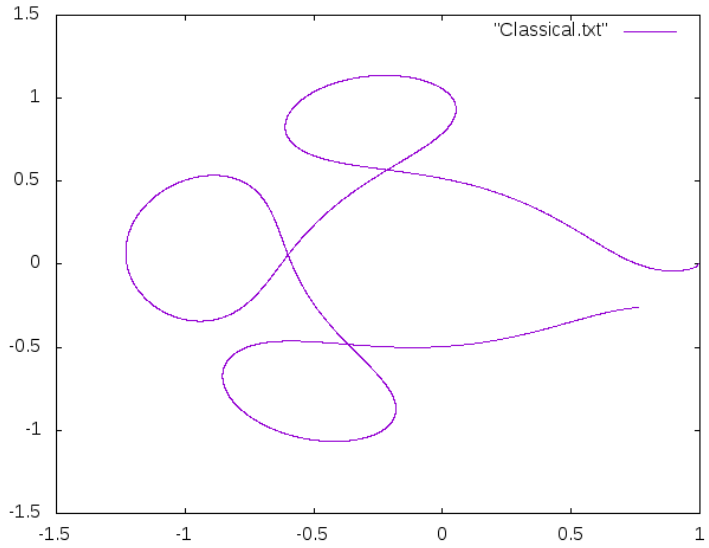
By our observation,it usually takes much smaller step size for (10.199) than (10.198) to achieve a visually indistinguishable plot compared to the exact plot.

3. **The comparison between Euler's method,the classical RK method and Dormand-Prince 5(4) embedded RK method**

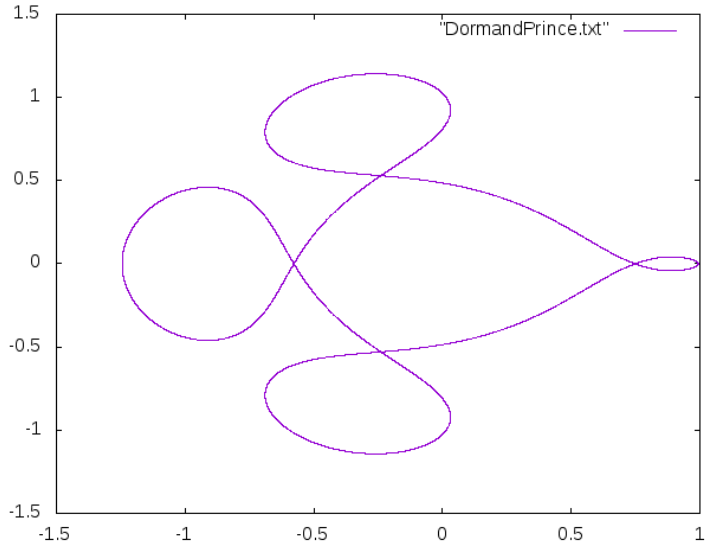
- The corresponding plots
  - Euler's method with 24000 steps



– the classical RK method with 6000 steps



– Dormand-Prince5(4) embedded RK method with 100 steps



- Performance comparison
  By our several tests,we find that Euler's method can not achieve an error of 10^{-3} even we have used up the size allocated to a double array,which finally results in core dumped,and this may be caused by its low order of accuracy and not being L-stable.It takes approximately 43400 steps for the classical RK method and 100 steps for Dormand-Prince5(4) to achieve the required error norm,and the corresponding CPU time cost for the former is approximately 0.103688s and 0.058086s for the latter.
  Therefore,Dormand-Prince5(4) embedded RK method is the winner in terms of total CPU time.

Analysis

- From our previous tests,we can see all methods converges for sufficiently small time-step size and their convergence rates mainly depend on accuracy order.And as grid number doubles,each method take a double calculations and this reflects on the approximately doubly increase of CPU time.

- Although Euler's method converges to the solution theoretically,it is not numerically useful here since it costs a lot memory and CPU time to reach a relatively good result.Besides,compared to other methods,it takes a much smaller time-step size for Euler's method to be convergent and this is why in our tests,its error norm may oscillate at large time-step size.

- The performance of the classical RK method is already much better and converges faster to the solution.However,it

is clear that most accuracy is lost in those parts of the orbit which are close to a singularity,that is the method with 6000 steps has a relatively large error at the end points.

- For the implicit methods(like Adams-Moulton methods,BDFs,the ESDIRK method and Gauss-Legendre RK methods),the performance for each methods is closely related to the numerical method applied to solve the corresponding implicit equations,and in our project,we have utilized the fixed point iteration to solve them.However,the iteration function in discussion is not Lipschitz continuous everywhere,and this may lead to the divergence of the solutions.Hence,some of the implicit methods perform not very well.Besides,the process of iteration usually makes these methods cost more CPU time which is a price for pursuing higher order of accuracy.