Efficiency Evaluation of Numerical Methods for 2nd-order ODEs.

JQX051[1]

Higher Level Mathematics Analysis and Approaches

I. Introduction

In writing an extended essay on airfoil efficiency optimization, I used a computational fluid dynamics solver (CFD) to approximate forces exerted on rigid bodies by fluids. Fundamentally, the program solved ordinary and partial differential equations of varying orders among other intermediary processes. Each iteration required considerable computational capacity and time, and thus, finding the most efficient numerical method for solving ordinary differential equations (ODEs) would allow processes like CFD solving to be optimized. The efficiency of a method, as defined for this exploration, is maximized when complexity—measured through runtime—is minimized, while the accuracy of the results is maximized.

II. Numerical Methods

Three approaches to approximating second order ODEs will be evaluated. Namely, the Runge-Kutta midpoint (RK2) method, the multistep predictor-corrector (P-C) method, and the backward Euler method. These methods provide varying approaches for computing solutions to differential equations, as will be detailed, which enables effective efficency comparisons.

In describing each method, the second-order ODE describing the simple harmonic motion of a mass-pendulum system,

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -\omega^2 x, \tag{1}$$

will be used for sample calculations. For all methods, the equation must be decomposed into a system of first-order ODEs:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= v \\ \frac{\mathrm{d}^2x}{\mathrm{d}t^2} = \frac{\mathrm{d}v}{\mathrm{d}t} &= -\omega^2 x \end{aligned} \tag{2}$$

[1]

The samples calculations will consider the initial conditions $\omega = \sqrt{10}$, $x_0 = 1$, $v_0 = 0$, and $h = 0.01$ where ω is the angular frequency of the system, x_0 is the initial displacement of the mass from equilibrium, v_0 is the initial velocity of the mass, and h is the iterative step size.

A. Runge-Kutta Midpoint (RK2) Method

The Runge-Kutta Midpoint (RK2) method is a numerical method for computing 2nd order ODEs based on the higher order RK4 method. RK2 works similar to the Euler method, but introduces a half step (midpoint) between iterations to improve the accuracy of the approximations, shown in Fig. 1.

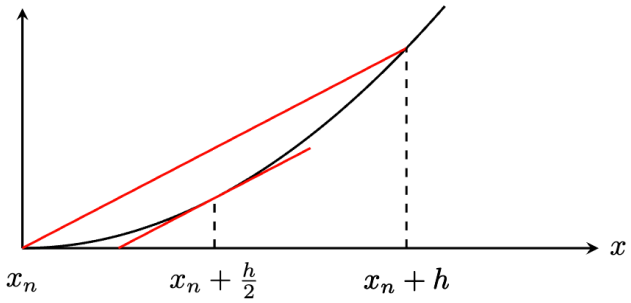

Fig. 1 RK2 Method

To find x_{n+1} and v_{n+1} at time $t_{n+1} = t_n + h$, the derivative k of the initial point is used to calculate values at midpoint m:

$$\begin{aligned} k1_v &= -\omega^2 x_n \\ k1_x &= v_n \\ m_v &= v_n + \frac{1}{2} h \times k1_v \\ m_x &= x_n + \frac{1}{2} h \times k1_x. \end{aligned} \tag{3}$$

The slopes at the midpoint are then used to calculate the final values:

$$\begin{aligned} k2_v &= -\omega^2 m_x \\ k2_x &= m_v \\ x_{n+1} &= x_n + h \times k2_x \\ v_{n+1} &= v_n + h \times k2_v. \end{aligned} \tag{4}$$

Given the prescribed initial conditions, the ODE can be approximated with RK2 as follows, starting with the half step,

$$
\begin{aligned}
k1_v &= -10 \times 1 = -10 \\
k1_x &= 0 \\
m_v &= 0 + \frac{1}{2} \times 0.01 \times (-10) = -0.05 \\
m_x &= 1 + \frac{1}{2} \times 0.01 \times 0 = 1,
\end{aligned}
\tag{5}
$$

followed by the full step,

$$
\begin{aligned}
k2_v &= -10 \times 1 = -10 \\
k2_x &= -0.05 \\
x_{n+1} &= 1 + 0.01 \times (-0.05) = 0.9995 \\
v_{n+1} &= 0 + 0.01 \times (-10) = -0.1.
\end{aligned}
\tag{6}
$$

To predict the motion of the pendulum system over time, these final values are used to approximate the next iteration. The RK2 method involves 8 main calculations per iteration for 2nd order ODEs, and thus, the number of operations is proportional to the number of iterations.

B. Multistep Predictor-Corrector Method (P-C)

The predictor-corrector (P-C) method is a multistep explicit method involving a predictor step, in which the values are predicted using the Euler method, then anjusted in the corrector step using the Adams-Bashforth Method. To find x_{n+1} and v_{n+1} at time $t_{n+1} = t_n + h$, the Euler method is applied to predict v and y at time t_{n+1}:

$$
\begin{aligned}
v_{\text{prediction}} &= v_n + h \times -\omega^2 x_n \\
x_{\text{prediction}} &= x_n + h \times v_n.
\end{aligned}
\tag{7}
$$

This is followed by the corrector step, which is applied conditionally. For the first iteration where a previous v value is unavailable,

$$
v_{\text{corrected}} = v_{\text{prediction}}. \tag{8}
$$

Otherwise, for later iterations where a previous v value is available,

$$
v_{\text{corrected}} = v_n + h\left(\frac{3}{2}(-\omega^2 x_n) - \frac{1}{2}(-\omega^2 x_{n-1})\right). \tag{9}
$$

In both cases, $x_{\text{corrected}}$ is calculated as

$$
x_{\text{corrected}} = x_n + \frac{h}{2}(v_n + v_{\text{corrected}}). \tag{10}
$$

For the computation of following iterations, the following assignments can be made:

$$
\begin{aligned}
x_{n+1} &= x_{\text{corrected}} \\
v_{n+1} &= v_{\text{corrected}}.
\end{aligned}
\tag{11}
$$

Given the prescribed initial conditions, the ODE can be approximated with P-C as follows, starting with the prediction,

$$\begin{aligned} v_{\text{prediction}} &= 0 + 0.01 \times (-10 \times 1) = -0.1 \\ x_{\text{prediction}} &= 1 + 0.01 \times 0 = 1, \end{aligned} \tag{12}$$

followed by the corrector,

$$\begin{aligned} v_{n+1} &= -0.1 \quad \text{(since it is the first step)} \\ x_{n+1} &= 1 + \frac{0.01}{2}(0 - 0.1) = 0.9995. \end{aligned} \tag{13}$$

As with RK2, to predict the motion of the pendulum system over time, these final values are used to approximate the next iteration. The P-C method involves 4 main calculations per iteration for 2nd order ODEs.

C. Backward Euler Method

The backward Euler method is an implicit method that uses the unknown slope at the next point to approximate the function's value at that point. To find x_{n+1} and v_{n+1} at time $t_{n+1} = t_n + h$, implicit equations are forumlated:

$$\begin{aligned} x_{n+1} &= x_n + h \times v_{n+1} \\ v_{n+1} &= v_n + h \times (-\omega^2 x_{n+1}). \end{aligned} \tag{14}$$

The system of equations is then solved. Although the system of equations is simple in this example, the potential nonlinearity of equations in more complex systems requires the use of numeric solvers, as will be implemented in the test scripts.

Given the prescribed initial conditions, the ODE can be approximated with the backward Euler method as follows:

$$\begin{aligned} x_{n+1} &= 1 + 0.01 \times v_{n+1} \\ v_{n+1} &= 0 - 0.01 \times 10 \times x_{n+1}. \end{aligned} \tag{15}$$

Using the numeric solver `fsolve` from the `scipy.optimize` Python module yields the following results:

$$\begin{aligned} x_{n+1} &= 0.9990 \\ v_{n+1} &= -0.0999. \end{aligned} \tag{16}$$

Although this method only involves 2 fundamental calculations per iteration, there is no definite number of operations involved due to the numerical solving step which involves external libraries and its dependency on the complexity of the equations themselves.

III. Complexity Evaluation

The worst-case time complexity of each numerical method can be represented using the Big O notation by dissecting each operation[†].

The complexity of each numerical method can be quantified by measuring their runtimes. Each method was applied to approximate three practical equations over 100 trials: The equation of motion of a simple harmonic oscillator (TC1),

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -\omega^2 x, \tag{17}$$

the equation of motion of a damped harmonic oscillator (TC2),

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + b\frac{\mathrm{d}x}{\mathrm{d}t} + kx = 0, \tag{18}$$

and the Van der Pol oscillator equation (TC3),

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \mu(1-x^2)\frac{\mathrm{d}x}{\mathrm{d}t} + x = 0. \tag{19}$$

Each test case was configured with the same step size, initial conditions, and number of iterations[‡]. As seen in Fig. 2, the RK2 method is the overall fastest, followed by the P-C method, then the Backward Euler method which has a consistently greater average runtime. The discrepancy can be attributed to the numeric solver `fsolve` for evaluating systems of equations within each step.

Table 1 Average Runtime / s[§]

Test Case	Backwards Euler	P-C	RK2
TC1	0.047272074	0.009758432	0.003964148
TC2	0.04873435	0.002453182	0.001983919
TC3	0.054947517	0.003014982	0.002310078

[†]The Backward Euler method cannot be evaluated due to its dependency on external modules for computing systems of equations.
[‡]Test scripts and initial conditions are in the appendix.
[§]Raw data is in the appendix.

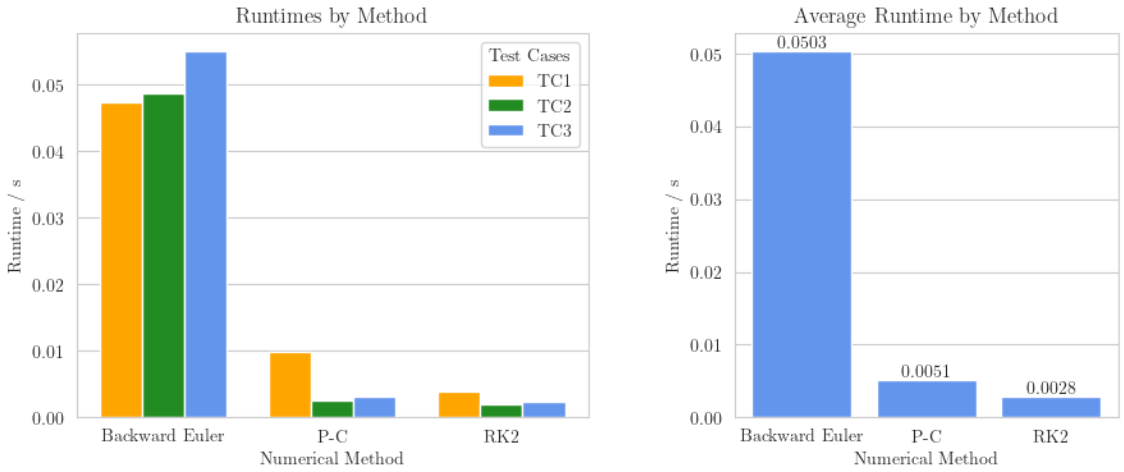

Fig. 2 Runtime Charts

Test-case specific results reveal that P-C and RK2 share similar runtimes, with an anomaly in TC1 where the P-C method struggled. The relative runtime distribution between test cases using the Backward Euler method do not align with those of P-C—the increase in runtime between test cases occurs due to the increasing complexity the test cases, which requires the intense computation of the systems of equations.

Dissecting the

IV. Accuracy Evaluation

- for SHM and damped equations: plot against analytical solution
- plot absolute error
- global error analysis: RMSE
- analyse accuracy via convergence rate:
- run method w different stepsizes (magnitudes of 2)
- calculate rmse of each step size
- analyze error reductiona nd calc convergence rate (convergence rate formula)

$$p \approx \frac{\log\left(\frac{E(h_1)}{E(h_2)}\right)}{\log\left(\frac{h_1}{h_2}\right)} \tag{20}$$

compare convergence rate across methods higher order methods = higher efficiency

results:

- for SHM, only RK2 converges, while the other methods increase in abs. in an increasing oscillating pattern for every iteration
- for damped, all solutions converge. RK2 still converges the fastest, but P-C and Reverse Euler vary between test cases.
- visual comparison of graphs (global)

V. Efficiency Evaluation

RK2 is the best overall

Reverse Euler is lowest performing

P-C is quite accurate across the board and can be used for stiffer equations

Defining Efficiency: accuracy/time -> comparison of this figure leads to RK2 being the most efficient.

Important consideration + extension: Reverse euler is much more stable, so it performs well with larger step sizes. This was not changed as it was a control, but it greatly affects its performance as its runtime is comparable to that of the other methods at large step sizes of up to h=3, which is 100x more than the tested size. This does not work for the other two methods.

- this may be considered and explored for the final submission, if advised.

Extension: Test step sizes for full optimization.

Note: "optimization" is better done within method classes themselves. This exploration only covers and outlines the differences between each method class. Focusing on optimizing one solution/method will allow for proper modeling of step size vs. time vs. convergence rate for full optimization. "optimization" in this case is more like "evaluation", which was an oversight when this exploration was planned: method types cannot be quantified for proper optimization.

Thus, the paper title may be changed to match that: Efficiency Evaluation…

note to Examiner: All the data has been collected, and all of the code and graphs have been generated. I will work to add all of these results for the final submission.

See the full source code at https://github.com/perakasem/hlaa-ia

- contains code for each numerical method: will be added to appendix for final submission.

VI. Conclusion

References

[1] Burkardt, J., "The Midpoint and Runge Kutta Methods", Jul 13 2011.

[2] P. Kavitha, and K. Prathiba, "Adams-Bashforth Corrector-Predictor Method Using MATLAB", International Journal of Mechanical Engineering, Kalahari Journals, Apr2022.. Retrieved 8 January 2024. https://kalaharijournals.com/resources/APRIL_93.pdf

[3] Brorson, S., "Backward Euler Method".