

AM225: Assignment 2 Solutions*

Part I: ODE solution methods

1. Adaptive integration with a First Same As Last (FSAL) scheme

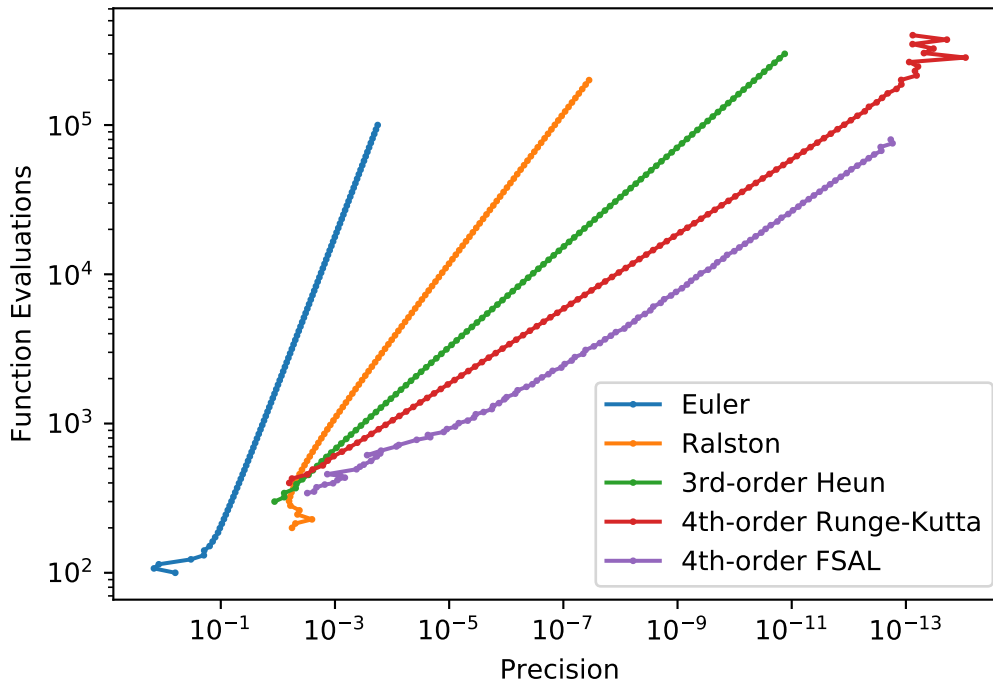


Figure 1: The fourth-order FSAL method is shown on a precision–work plot, comparing its performance to that of several other methods discussed in lecture.

(a) The Brusselator system is given by

$$y_1' = 1 + y_1^2 y_2 - 4y_1, \quad y_2' = 3y_1 - y_1^2 y_2 \quad (1)$$

with initial conditions $y_1(0) = 1.5, y_2(0) = 3$. Its position on a precision–work plot, when compared with the four methods from lecture, is shown in Figure 1. As expected, the method is fourth-order, but uses fewer function evaluations than fourth-order Runge-Kutta.

(b) The two-component system

$$y_1' = -xy_2, \quad y_2' = xy_1 \quad (2)$$

with initial conditions $y_1(0) = 1, y_2(0) = 0$ is an oscillating system. The exact solution is given by $y_1^{\text{exact}}(x) = \cos \frac{x^2}{2}, y_2^{\text{exact}} = \sin \frac{x^2}{2}$.

*Solution to Problem 2 written by Dan Fortunato. Solutions to all other problems written by Nick Derr.

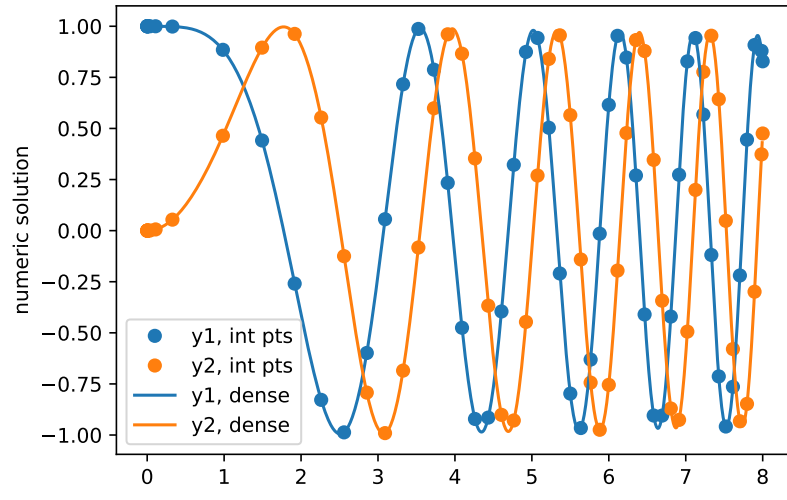


Figure 2: The two-component oscillator is plotted above. The integration points are marked by dots, and the dense output is plotted as a line.

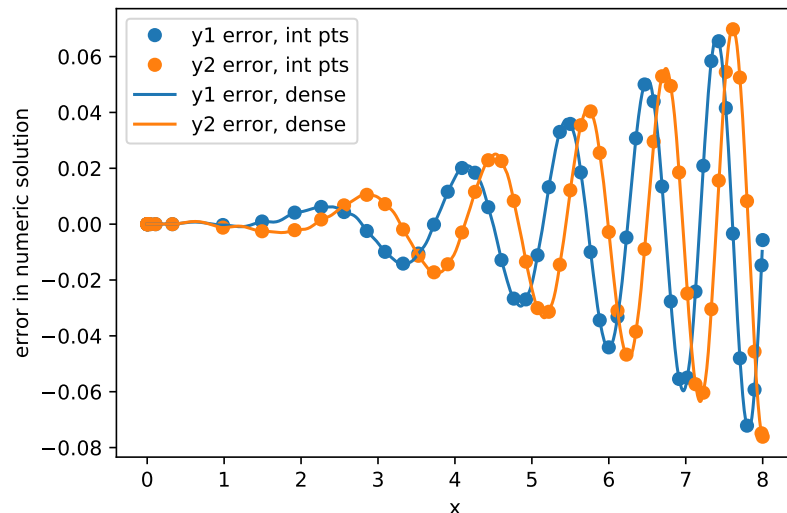


Figure 3: The error in the two-component oscillator system is plotted above. The integration points are marked by dots, and the dense output is plotted as a line.

Fig 2 shows the numerically integrated system, produced using $\lambda = 3 \times 10^{-3}$. The integration points are marked as points, and dense output is marked as lines. Fig 3 shows the integration error in each component of the solution. Integration points and dense output are marked as in Fig 2.

2. A high-order adaptive integrator using Richardson extrapolation

- (a) Fig. 4 shows a precision–work plot for the sixth-order Richardson-extrapolated Cash–Karp method applied to the Brusselator problem (1). For comparison, the plot includes other lower-order methods from lecture. The slope of the Cash–Karp line is $-1/6$.

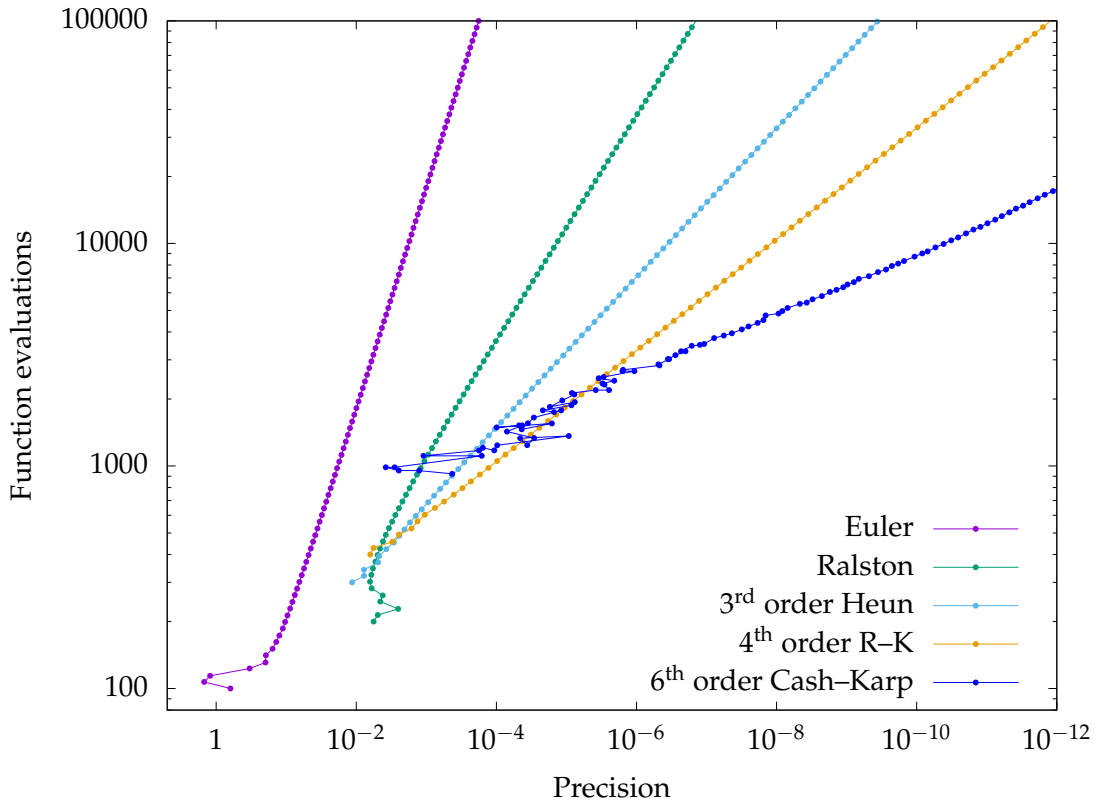
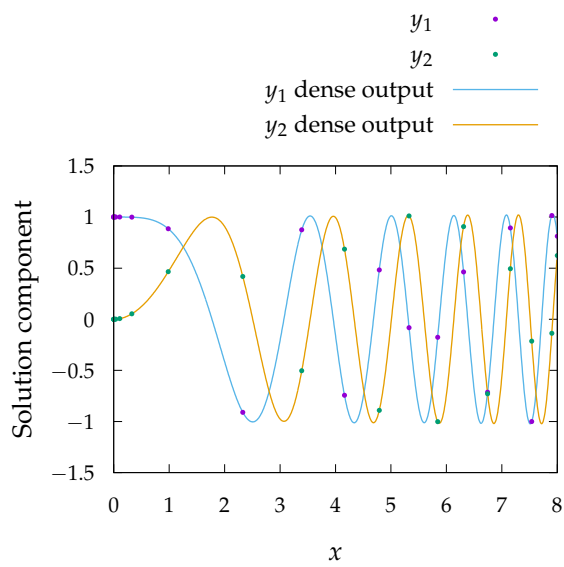
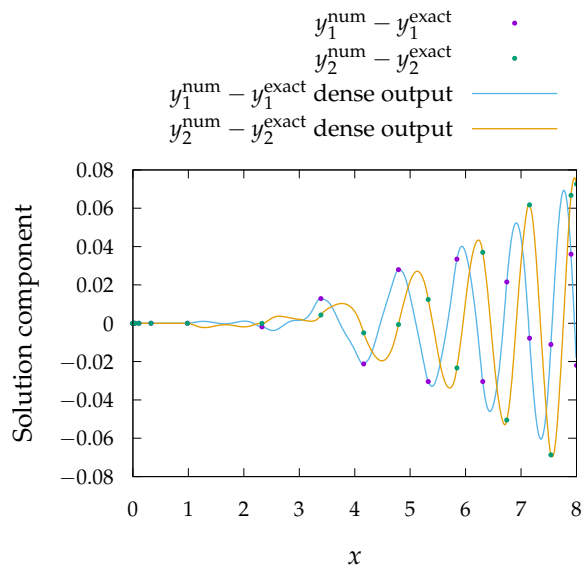


Figure 4: Precision–work plot for the Cash–Karp method applied to the Brusselator problem.

- (b) Fig. 5a shows the solution to (2) simulated to $x = 8$ using the sixth-order Richardson-extrapolated Cash–Karp scheme, with dense output based on quintic polynomial interpolation. The dense output is saved at intervals of $\frac{8}{1200}$ and step size adaptivity is performed with $Atol = Rtol = \lambda = 3 \times 10^{-3}$. The error is shown in Fig. 5b.



(a) Solution



(b) Error

Figure 5: Computed solution and error to the oscillator problem (2) using the sixth-order Richardson-extrapolated Cash–Karp method with dense output.

Part II: ODE applications and analysis

3. **Order condition trees.** How many unique trees of order k exist? We can consider the question from a combinatorics point of view. Consider an order- k tree whose root node has n children. Each child $i \in [1, n]$ is the root of an order- q_i subtree. Note the q_i 's must satisfy $\sum_{i=1}^n q_i = k - 1$, since the total tree including the root node must be order k .

Partitions We can take this combination of subtree orders and define the set

$$\mathbf{q} := \left\{ q_i : i \in [1, n] \text{ for some } n, q_i \in \mathbb{Z}, \sum_{i=0}^n q_i = k - 1 \right\}$$

corresponding to a integer partition of the integer $k - 1$. Some notes:

- The quantity \mathbf{q} is a set, not a vector, as **there is no ordering of elements**. Example: (2,3) and (3,2) represent the same partition of 5.
- In general, many such partitions of an integer exist and may have different lengths. Example: the partitions of 5 are (5), (1,4), (2,3), (1,1,3), (1,2,2), (1,1,1,2), and (1,1,1,1,1).

Let's define $\mathbf{q}^{(k,j)}$ as the j th partition of $k - 1$, and $n^{(k,j)}$ as its length.

Partition classes It may be clear that each partition $\mathbf{q}^{(k,j)}$ corresponds to a class of order- k trees with the following properties:

- root nodes of trees in partition class $\mathbf{q}^{(k,j)}$ have n_j children
- child $i \in [1, n^{(k,j)}]$ is the root of a subtree of order $q_i^{(k,j)}$

Note that, by definition, **a given tree is a member of exactly one partition class**. Thus, we can count the number of trees of a given order by summing the size of each of these classes. Before computing the size of a single class, we must consider the number of possible combinations of m subtrees of order o , since there is no ordering of node children.

Combinations of subtrees of the same order Let the number of trees of order o be N_o . We wish to choose m trees, with replacement, from N_o choices. While the ordering of the choices does not matter, the number of times a given tree is chosen does; in other words, the problem is analogous to the question of placing m distinguishable balls into N_o indistinguishable baskets. The correct number of combinations is therefore

$$M^{(m,o)} = \binom{N_o + m - 1}{m}^\dagger.$$

[†]This quantity can be computed, for example, using **stars and bars**.

k	N_k
1	1
2	1
3	2
4	4
5	9
6	20
7	48
8	115
9	286
10	719
11	1,842
12	4,766
13	12,486
14	32,973
15	87,811

Table 1: The number of unique trees N_k of order k , for $k \leq 15$

Size of a single partition class Let's define the sets $\mathbf{v}^{(k,j)}$, $\mathbf{m}^{(k,j)}$ and the integer $l^{(k,j)}$ as an alternate representation of the partition $\mathbf{q}^{(k,j)}$ and length $n^{(k,m)}$, such that the partition contains $l^{(k,j)}$ unique values $v_i^{(k,j)}$, $i \in [1, l^{(k,j)}]$, each of which is repeated $m_i^{(k,m)}$ times.

Using the previous section, the size $N_k^{(j)}$ of partition class (k, j) must be equal to

$$N_k^{(j)} = \prod_{i=1}^{l^{(k,j)}} M(m_i^{(k,j)}, v_i^{(k,j)}).$$

Number of trees of a given order If the number of partitions P_k of $k - 1$ is known, then

$$N_k = \sum_{j=1}^{P_k} N_k^{(j)} = \sum_{j=1}^{P_k} \prod_{i=1}^{l^{(k,j)}} \binom{N_{v_i^{(k,j)}} + m_i^{(k,j)} - 1}{m_i^{(k,j)}}.$$

The number of unique trees at order $k \leq 15$ is shown in Table 1. All of the 48 order-7 trees are shown in Figure 6.

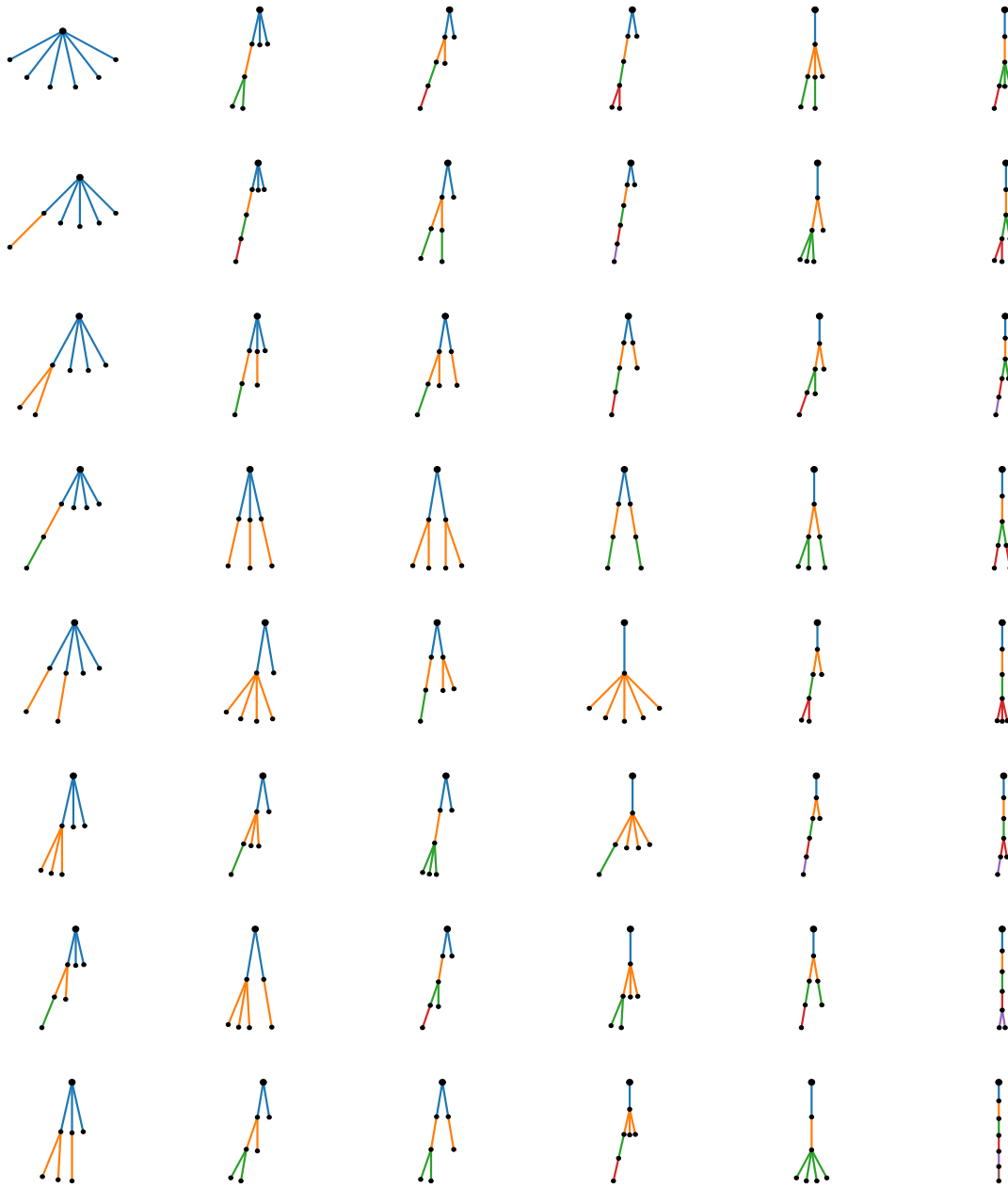


Figure 6: The 48 unique trees of order 7

4. Error analysis of a Richardson extrapolation scheme.

(a) The second-order Ralston method corresponds to the following Butcher tableau.

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

If we define the four stages

$$\begin{aligned} k_{11} &= f(t_k, y_k), \\ k_{12} &= f\left(t_k + \frac{2h}{3}, y_k + \frac{2hk_1}{3}\right), \\ k_{21} &= f\left(t_k + h, y_k + \frac{h}{4}(k_{11} + 3k_{12})\right), \\ k_{22} &= f\left(t_k + \frac{5h}{3}, y_k + \frac{h}{12}(3k_{11} + 9k_{12} + 8k_{21})\right), \end{aligned}$$

then the method yields the following second-order approximations to the function values $y(t_k + h)$ and $y(t_k + 2h)$:

$$\begin{aligned} y_{k+1} &= y_k + \frac{h}{4}(k_{11} + 3k_{12}), \\ y_{k+2} &= y_k + \frac{h}{4}(k_{11} + 3k_{12} + k_{21} + 3k_{22}). \end{aligned}$$

We can also use the same method to calculate a second-order approximation to $y(t_k + 2h)$ using a single step of size $H = 2h$, such that

$$\begin{aligned} k_{w1} &= f(t_k, y_k) = k_{11}, \\ k_{w2} &= f\left(t_k + \frac{2H}{3}, y_k + \frac{2Hk_{11}}{3}\right), \\ w &= y_k + \frac{H}{4}(k_{11} + 3k_{w2}) = y_k + \frac{h}{2}(k_{11} + 3k_{w2}). \end{aligned}$$

Using Richardson extrapolation, a third-order approximation for $y(t_k + 2h)$ is given by

$$\hat{y} = y_{k+2} + \frac{y_{k+2} - w}{2^p - 1} = \frac{1}{3}(4y_{k+2} - w),$$

where we've used $p = 2$, since Ralston is second-order. Substituting, this implies

$$\begin{aligned} \hat{y} &= y_k + \frac{h}{3}(k_{11} + 3k_{12} + k_{21} + 3k_{22}) - \frac{1}{6}(k_{11} + 3k_{w2}), \\ &= y_k + \frac{h}{6}(k_{11} + 6k_{12} + 2k_{21} - 3k_{w2} + 6k_{22}). \end{aligned}$$

Writing the Runge-Kutta stages and \hat{y} in terms of H , we find

$$\begin{aligned}
 k_{11} &= f(t_k, y_k), \\
 k_{12} &= f\left(t_k + \frac{H}{3}, y_k + \frac{Hk_{11}}{3}\right), \\
 k_{21} &= f\left(t_k + \frac{H}{2}, y_k + \frac{H}{8}(k_{11} + 3k_{12})\right), \\
 k_{w2} &= f\left(t_k + \frac{2H}{3}, y_k + \frac{2Hk_{11}}{3}\right), \\
 k_{22} &= f\left(t_k + \frac{5H}{6}, y_k + \frac{H}{24}(3k_{11} + 9k_{12} + 8k_{21})\right), \\
 \hat{y} &= y_k + \frac{H}{12}(k_{11} + 6k_{12} + 2k_{21} - 3k_{w2} + 6k_{22}).
 \end{aligned}$$

indicating the third-order approximation to $y(t_k + H)$ is a five-stage Runge-Kutta method with the following Butcher tableau.

0					
1/3	1/3				
1/2	1/8	3/8			
2/3	2/3	0	0		
5/6	1/8	3/8	1/3	0	
	1/12	1/2	1/6	-1/4	1/2

We'll refer to this as the Ralston+Richardson (R+R) method.

(b) Now, recall that the error coefficients are given for each tree t by

$$e(t) = 1 - \gamma(t) \sum_{j=1}^s b_j \Phi_j(t),$$

where $\gamma(t)$ is a function of tree structure and $\Phi_j(t)$ is a function of tree structure and the Runge-Kutta coefficients a_{jk} . The values for these functions and the error coefficients are displayed for the R+R and Heun methods in Table 2.

(c) For each tree t , we find $|e_{R+R}(t)| \leq 1/2 |e_{\text{Heun}}(t)|$, indicating **the R+R method will have about double the precision of Heun at a given step size**. Similarly, since Heun requires three function evaluations and R+R requires five, we can see that **R+R takes about 5/3 as much work as Heun**. In other words, we could see a 100% increase in precision by switching from Heun to R+R, corresponding to a 67% increase in work. Is this worth the trade?

The answer is no. Since the methods are third order, we could also see a 100% increase in precision by keeping Heun and decreasing our step size by a factor of $2^{1/3} \approx 1.26$, corresponding to about 26% more work. We conclude that **Heun takes fewer function evaluations to reach a given precision**.





t				
$\gamma(t)$	4	8	12	24
$\Phi_j(t)$	$a_{jk}a_{jl}a_{jm}$	$a_{jk}a_{kl}a_{jm}$	$a_{jk}a_{kl}a_{km}$	$a_{jk}a_{kl}a_{lm}$
Ralston + Richardson extrapolation				
$\Phi_1(t)$	0	0	0	0
$\Phi_2(t)$	1/27	0	0	0
$\Phi_3(t)$	1/8	1/16	1/24	0
$\Phi_4(t)$	8/27	0	0	0
$\Phi_5(t)$	125/216	35/144	1/8	1/24
$e(t)$	-1/54	-1/18	1/6	1/2
Heun				
$\Phi_1(t)$	0	0	0	0
$\Phi_2(t)$	1/27	0	0	0
$\Phi_3(t)$	8/27	4/27	2/27	0
$e(t)$	1/9	1/9	1/3	1

Table 2: The functions $\gamma(t)$ and $\Phi_j(t)$ are shown for each of the order-4 trees (top). The function $\Phi_j(t)$ for each stage and resulting error coefficient $e(t)$ are plotted with respect to the Ralston+Richardson extrapolation method (middle). The same functions are plotted with respect to the Heun method (bottom).

As an aside, note that consideration of the methods' order is absolutely necessary for this analysis. For instance, if the methods were first order, the 67% increase in work from switching to R+R would have been more efficient than the 100% increase in work needed to get the corresponding increase in precision by halving the Heun step size.

5. A generalized Kuramoto model.

(a) Snapshots at $t = 10, 20, 50,$ and 200 are shown for each model. The corresponding movies are available on the `am225_solutions` GitHub repository.

i. $J = 0.5, K = 0.5, 507$ timesteps

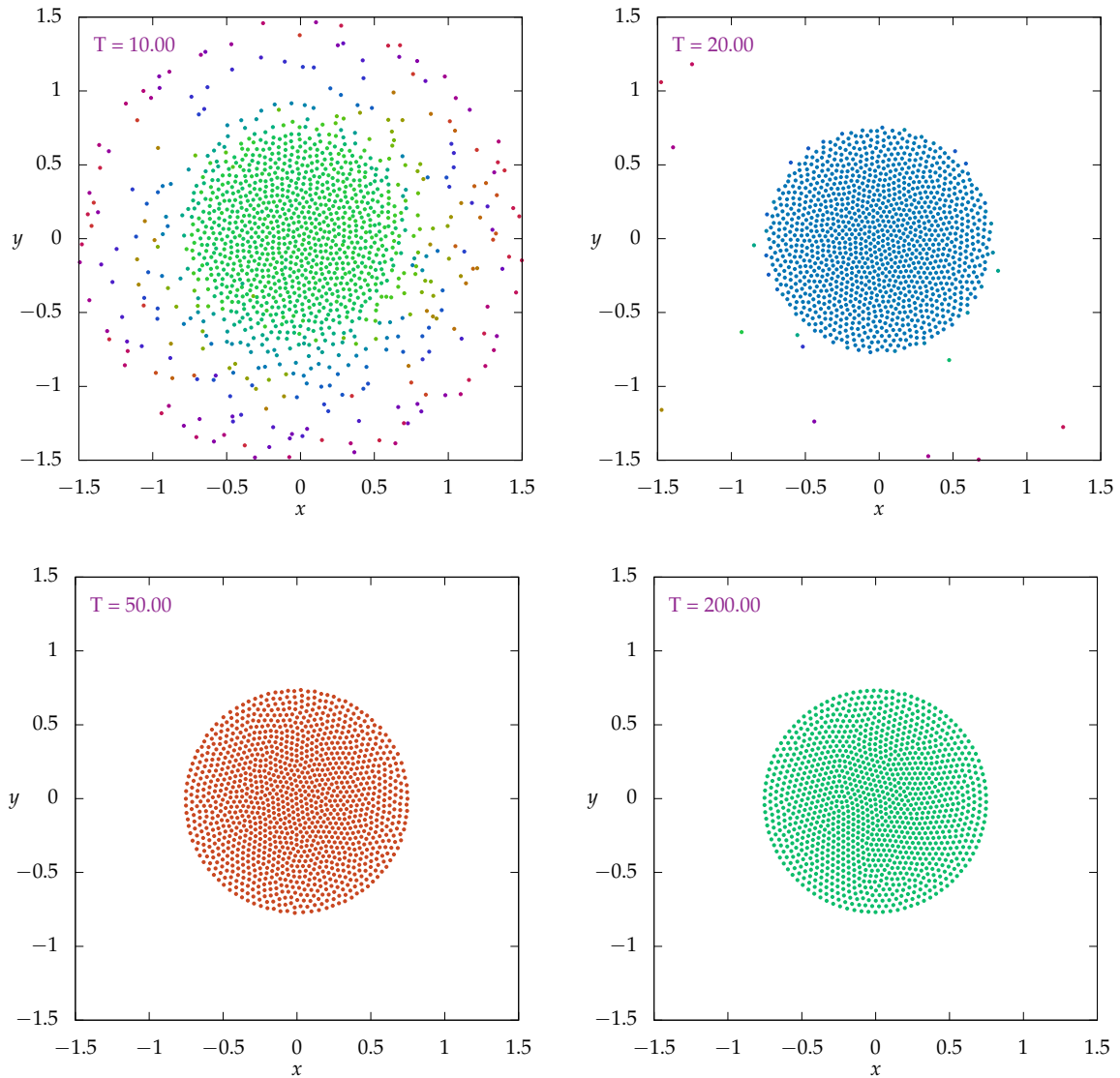


Figure 7: Four snapshots of the first Kuramoto swarming model

ii. $J = 0.3, K = -0.2, 1148$ timesteps

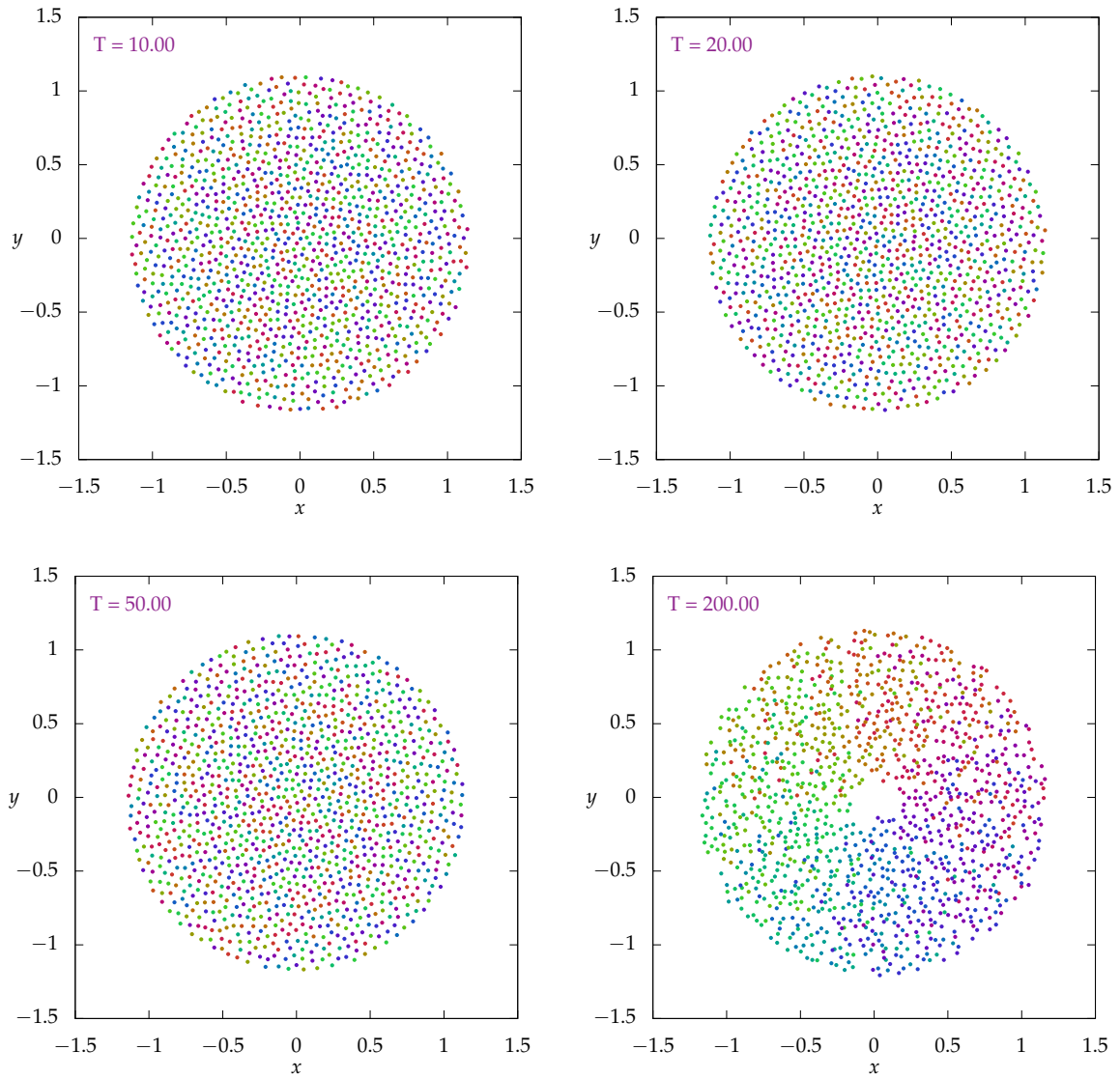


Figure 8: Four snapshots of the second Kuramoto swarming model

iii. $J = 1, K = -0.2, 1950$ timesteps

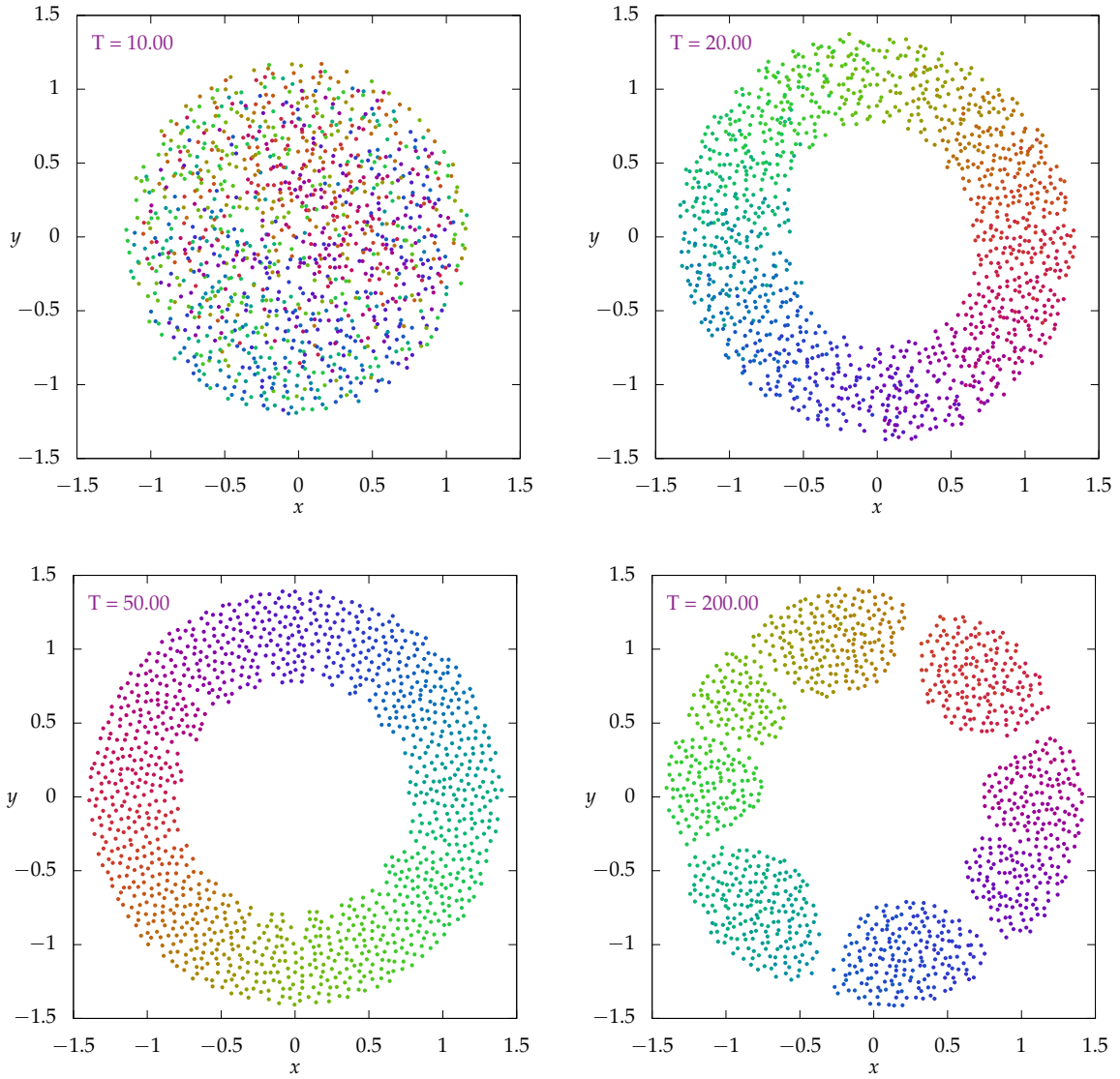


Figure 9: Four snapshots of the third Kuramoto swarming model

6. Symplectic integration for galactic dynamics.

The system can be written explicitly as

$$\dot{\mathbf{q}} = \begin{pmatrix} p_1 + \Omega q_2 \\ p_2 - \Omega q_1 \\ p_3 \end{pmatrix}, \quad \dot{\mathbf{p}} = \begin{pmatrix} \Omega p_2 \\ -\Omega p_1 \\ 0 \end{pmatrix} - \frac{2A^2}{V(\mathbf{q})} \begin{pmatrix} q_1/a^2 \\ q_2/b^2 \\ q_3/c^2 \end{pmatrix},$$

and with the provided values, the values of p_2 for which $H = 2$ are

$$p_2 = \frac{1}{40} \left(25 \pm \sqrt{6961 - 3200 \log 5} \right).$$

The larger root is $p_2 \approx 1.68884$.

- The convergence of Geng's method applied to the Brusselator is shown in Figure 10. Because the method is implicit, the number of function evaluations required for a given step size is orders of magnitude higher than the 4th-order explicit Runge-Kutta method. Since the method is fifth-order, it rapidly reaches machine precision as that step size is decreased.
- The simulated trajectory of the galactic system and the corresponding Hamiltonian value H are plotted from time $t = 0$ to 2000 in Figures 11 and 12, respectively. Note that the quantity $H(t)$ is preserved by the symplectic method, in the sense that any long-term drift in the value is small compared to variations in the value between time steps (on the order of 10^{-5}).

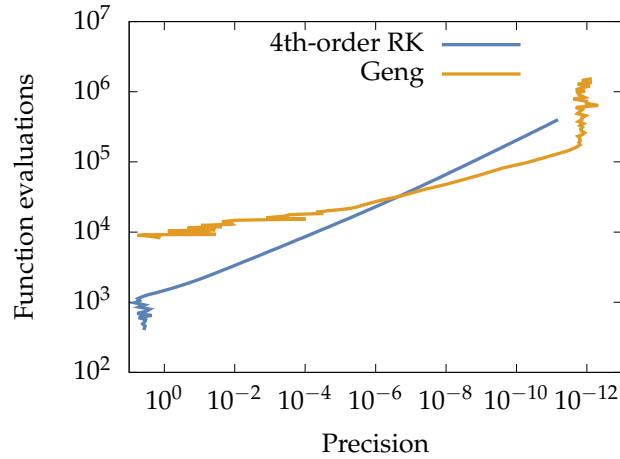


Figure 10: The convergence of Geng's method compared with that of the fourth-order Runge-Kutta method, applied to the Brusselator

(c) A few definitions:

- In general, the solution to a periodic ODE can be represented by a trajectory through n -dimensional space. In this problem, the galactic system is represented by a trajectory within a six-dimensional $(q_1, q_2, q_3, p_1, p_2, p_3)$ space.

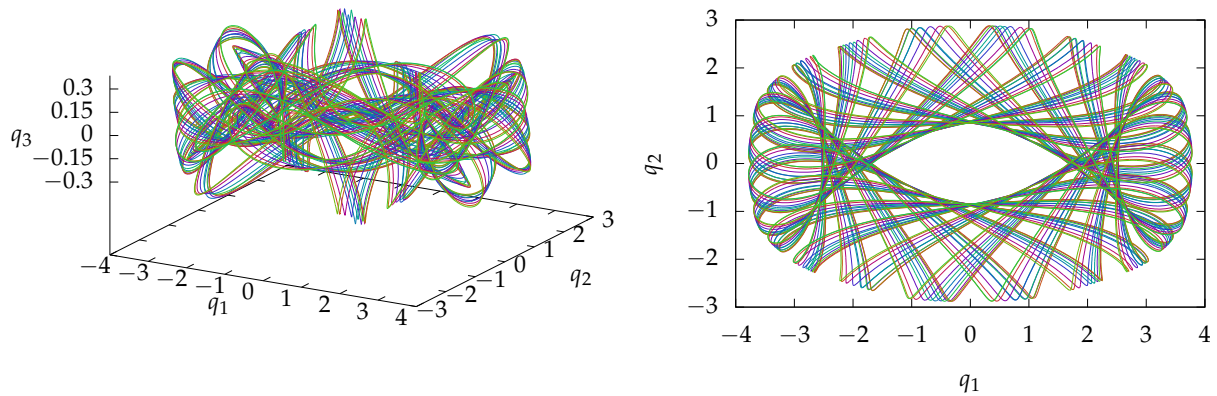


Figure 11: The star's trajectory plotted from $t = 0$ to 2000, from a canted view (left) and from directly above (right), where time is expressed by line color

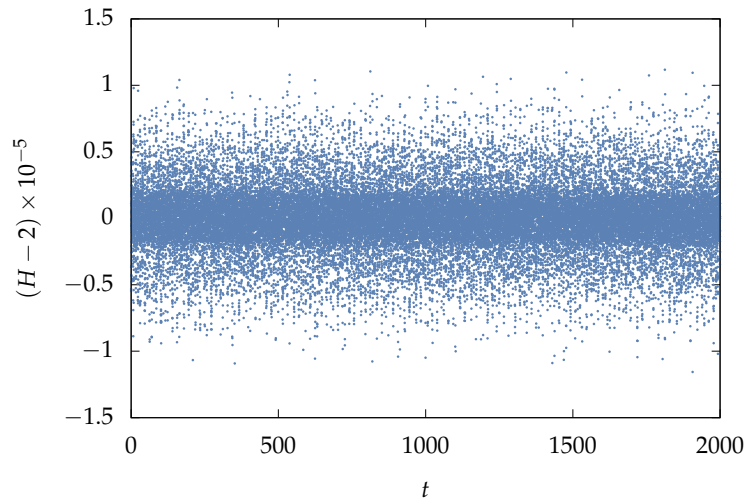


Figure 12: The deviation from the system's initial Hamiltonian ($H_0 = 2$) for $t = 0$ to 2000

- A *Poincaré section* is an $(n - 1)$ -dimensional subspace which is repeatedly intersected by the solution trajectory. In this case, the section is the five-dimensional half-half-space with $q_1 > 0, p_2 > 0, q_2 = 0$.
- The *Poincaré map* is a mapping from the Poincaré section to itself. Consider time t and location within the Poincaré section v . Let a trajectory intersect the section at series of times t_k , at the locations v_k . The Poincaré map P maps an intersection point to the next intersection point; that is, $P(v_k) = v_{k+1}$ for all k .

There are many ways we can try to visualize the distribution of crossings with the Poincaré section and qualities of the Poincaré map. Two that may reveal some information about the system are shown in the next few figures: we can look at the distribution of interaction locations projected onto 2D planes, and we can look at mappings of the value of individual coordinates at one intersection to their values at the next.

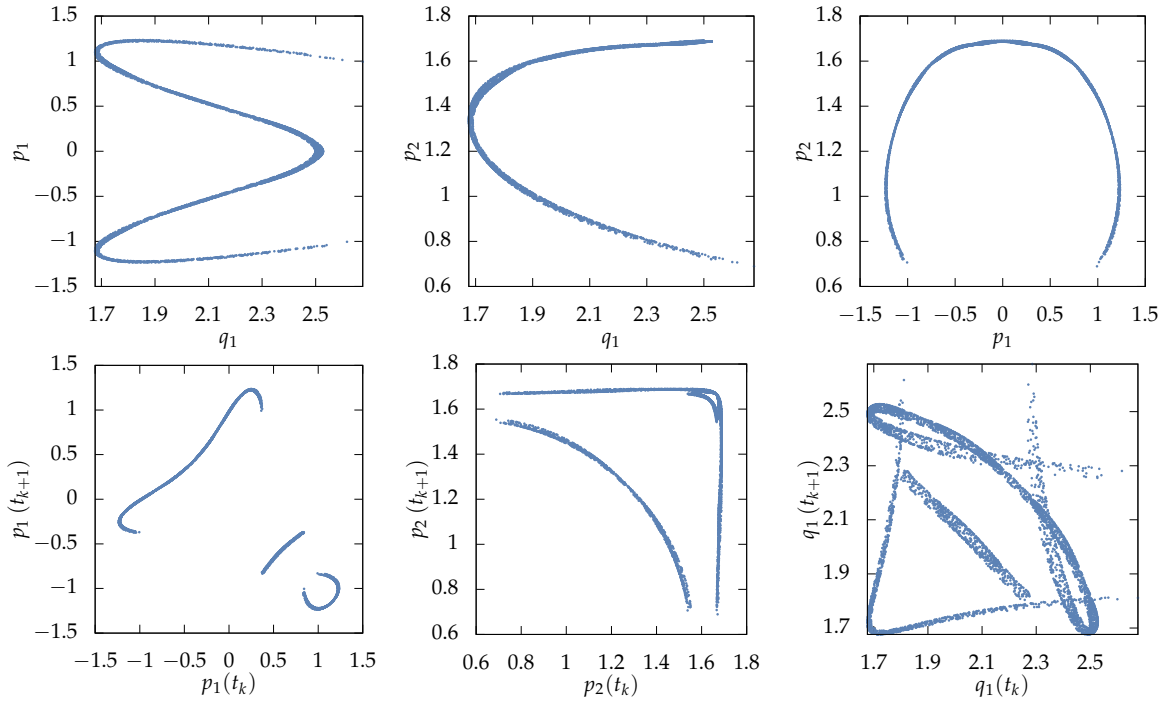


Figure 13: (Top) distributions of crossing points of the Poincaré section, projected onto 2D planes in the q_1, p_1, p_2 subspace. (Bottom) one-dimensional Poincaré maps for the three coordinates q_1, p_1, p_2 , such that the point (x, y) represents a mapping from a coordinate's value x as it intersects the Poincaré section to its value y on its next intersection with the section.

- At first order, the orbit is very similar to a planar trajectory in (q_1, q_2, p_1, p_2) space. Accordingly, Figure 13 shows Poincaré map-related visualizations of q_1, p_1, p_2 with tight grouping and clear structure.
- At next order, we expect the orbit would display oscillation about the $q_3 = 0$ plane. Figure 14 shows visualizations of q_3, p_3 which show clear structure, but are not quite as clean as those in Figure 13.
- These visualizations of crossings of the Poincaré section (the top row of Figure 13 and the left plot in Figure 14) are quite clean, as they represent projections of the crossings onto planes where we've reasoned the coordinate values are highly correlated.
- We expect that projections onto planes of less correlated variables, i.e. combinations of (q_1, p_1, p_2) and (q_3, p_3) , will produce maps of a different sort, as the relationships between the variables is not so obvious as a planar elliptical orbit or oscillation about a plane. Figure 15 shows these projections, which largely confirm our expectation: there is clear structure, but it is just as clear that trajectory crossings wander over a larger region in a more scattershot way.

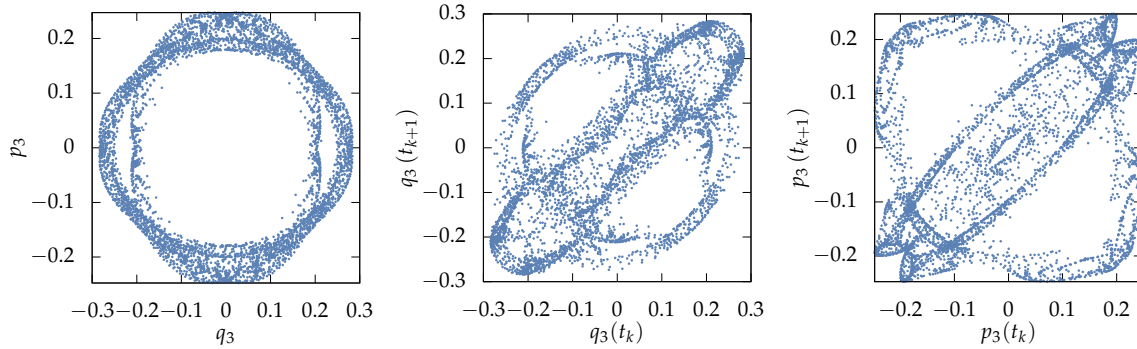


Figure 14: (Left) distributions of crossing points of the Poincaré section, projected onto the q_3, p_3 plane. (Middle, right) Poincaré maps of the two coordinates.

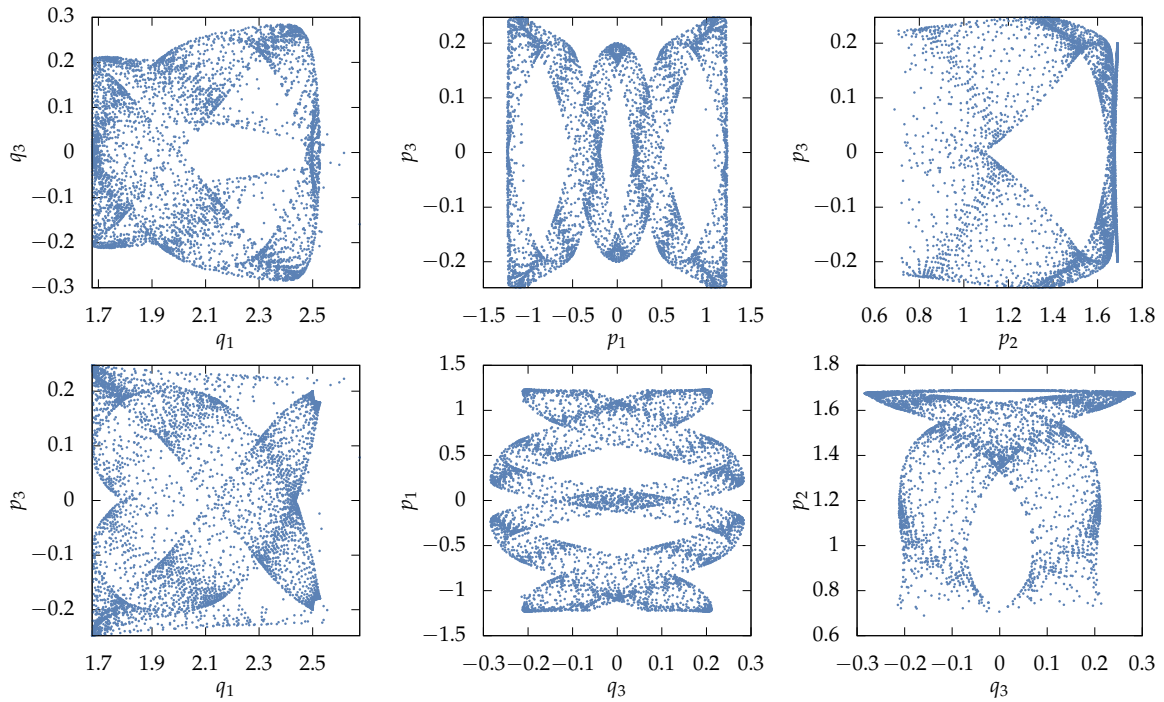


Figure 15: Distributions of crossing points of the Poincaré section projected onto $(q_1, p_1, p_2) \times (q_3, p_3)$ planes.

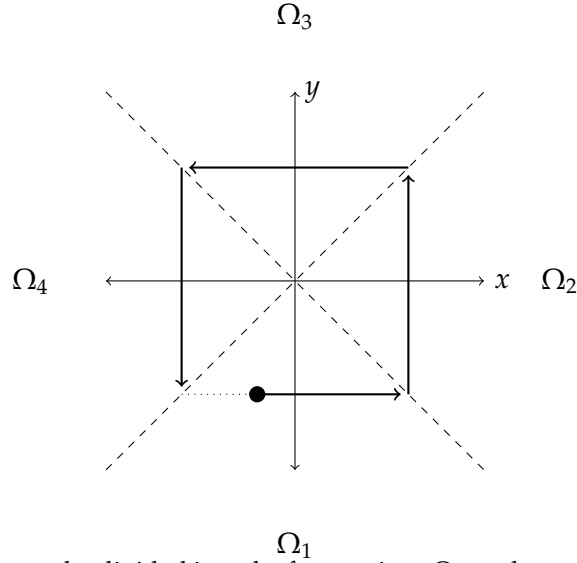


Figure 16: The cartesian plane can be divided into the four regions Ω_i , each consisting of one of the quadrants created by the intersection of the lines $y = x$ and $y = -x$. The black dot above will follow the square trajectory shown.

7. Integrating ODEs with discontinuities

(a) We can write our ODE in terms of a vector $\mathbf{r}(t) = [x(t), y(t)]^\top$, so that

$$\frac{d\mathbf{r}}{dt} = \begin{cases} -y\mathbf{x}, & \mathbf{r} \in \Omega_1 \text{ or } \Omega_3, \\ xy, & \mathbf{r} \in \Omega_2 \text{ or } \Omega_4, \end{cases}$$

where the Ω_i 's label quadrants of the Cartesian plane as shown in Figure 16. From the same figure, it's clear that trajectory will move along the perimeter of a square of size determined by the initial position. In turn, this shows the magnitude of the time rate of change must be constant (i.e. that y for $\mathbf{r} \in \Omega_1$ or Ω_3 will equal x for $\mathbf{r} \in \Omega_2$ or Ω_4 .)

The initial condition $\mathbf{r} = [1, 0]^\top$ corresponds to a square of side length 2 and a velocity magnitude 1; the perimeter of the square is therefore 8 and the period of the trajectory must be 8 as well. By inspection, we can write down the solution

$$\mathbf{r}(t) = \mathbf{r}(t \bmod 8),$$

for the piecewise functions defined on $\tau \in [0, 8)$:

$$x(\tau) = \begin{cases} 1, & \tau \in [0, 1) \text{ or } \tau \in [7, 8), \\ 2 - \tau, & \tau \in [1, 3), \\ -1, & \tau \in [3, 5), \\ \tau - 6, & \tau \in [5, 7), \end{cases} \quad y(\tau) = \begin{cases} \tau, & \tau \in [0, 1), \\ 1, & \tau \in [1, 3), \\ 4 - \tau, & \tau \in [3, 5), \\ -1, & \tau \in [5, 7), \\ \tau - 8, & \tau \in [7, 8). \end{cases}$$

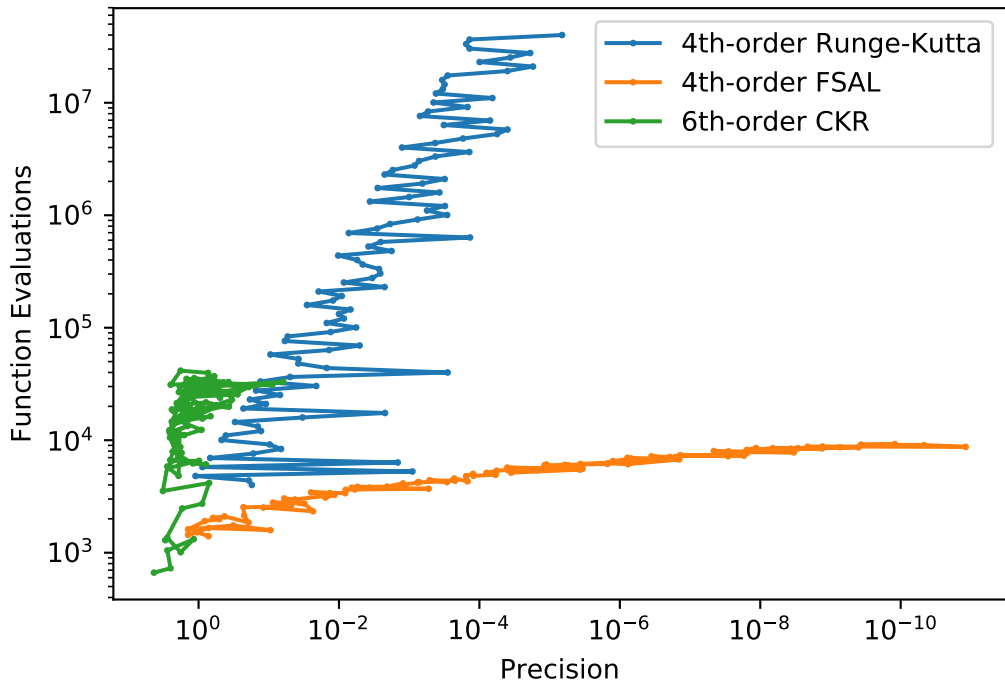


Figure 17: A precision–work plot showing the performance of 4th-order classic RK, 4th-order adaptive FSAL, and 6th-order adaptive Cash–Karp with Richardson extrapolation.

- (b) A precision–work plot showing the performance of classic, fixed-step Runge Kutta on the above system is shown in Figure 17. While it’s noisier than most work–precision plots we’ve seen, it shows roughly first-order convergence.

Note that any integrator of first-order or higher will exactly integrate the “sides” of the square—the only error introduced into the problem occurs at the corners. The integrator has no way of knowing it’s required to immediately turn left at $y = x$ or $y = -x$; as a result, it will slightly overshoot, and move to a square of slightly larger size. The amount of the overshoot will be less than or equal to the step size, indicating the error should scale with h . This is the source of the “envelope” slope in the work–precision plot.

A final note: why doesn’t the integration at the corner get better faster than $O(h)$ given we’re using a multi-step method? Runge-Kutta methods are based on zeroing out terms in Taylor expansions with the assumption that the remaining terms will be small (i.e. of some order or below, as determined, for instance, by the order tree analysis in Problem 3.) At the corner, though, the derivative is not continuous (i.e. the second derivative becomes infinite), and much of this Taylor series analysis is no longer valid.

- (c) The performances of the adaptive FSAL and Cash–Karp with Richardson extrapolation integrators are also plotted in Figure 17. In general, an adaptive method, upon hitting the corner, can reduce the time step until the error added is on the scale of the tolerance, while still taking large steps to exactly integrate the straight regions. As a result, we see extremely fast convergence.