NUMERICAL METHODS FOR SOLVING ORDINARY DIFFERENTIAL EQUATIONS

MARIA CAMERON

Contents

1. Introduction

Ref.: John Strain, Lecture 1.

1.1. Why do we need to study ODE solvers?

(1) Most ODEs except for very special cases, do not have analytical solutions and need to be solved numerically.
(2) Several commonly used solvers are implemented in high-level languages such as Matlab and Python,
   - Matlab: https://www.mathworks.com/help/matlab/math/choose-an-ode-solver.html,
   - Python, SciPy library: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html.

   Nonetheless, there are problems where these methods are not suitable. For example, such problems are those where certain first integrals (e.g. energy, angular momentum, etc.) need to be conserved.
(3) The built-in ODE solvers contain several useful features such as error control, dense output, etc. We will learn what it is and how to use it.
(4) In some cases, we would like to accomplish a more elaborate task than built-in tools allow us. Then we need to be able to implement an ODE solver ourselves.
(5) The theory for ODE solvers is very enlightening. Its study will allow us to understand why methods might fail to compute a solution accurately and find a remedy for it.
(6) ODE solvers are building blocks for PDE solvers and SDE solvers.

1.2. What will we study? Keywords:

- Basic theory for ODE problems: *well-posedness: existence, uniqueness, stability with respect to small perturbations*;
- Basic concepts for ODE solvers: *consistency, stability, convergence, order of the method*;
- Stiff problems;
- Linear stability theory;
- Runge-Kutta methods;
- Linear Multistep methods;
- Extra features: *error control, dense output*;
- Symplectic ODE solvers.

1.3. **Types of ODE problems.** The following types of problems involving ODEs are typically considered:

- **Initial Value Problem (IVP)**, $y' = f(t,y), \quad y(t_0) = y_0$;
- **Boundary Value Problem (BVP)**, e.g. $y' = f(t,y), \quad y^1(t_0) = y_{1,0}, \quad y^1(t_1) = y_{1,1}$, where y^1 is the first component of y;
- **Optimal control problem**: $y = f(t,y,u)$, $y(0) = y_0$, and there is either the target destination $y(t_1) = y_1$ or a cost functional that needs to be minimized;
- **Inverse problem**: given a collection of solutions $y(t)$, one needs to identify the right-hand side function $f(t,y)$.

In this course, we will focus on IVP. Find the solution to a given ODE with a given initial condition:

$$y' = f(t,y), \quad y(t_0) = y_0, \tag{1}$$

on the time interval $[t_0, T]$, where $y : [t_0,T] \to \mathbb{R}^d$, $f : [t_0,T] \times \mathbb{R}^d \to \mathbb{R}^d$. The examples below show that the solution may fail to exist throughout $[t_0,T]$, or can be nonunique.

Example 1.

$$y' = y^2, \quad y(t_0) = y_0. \tag{2}$$

The solution is obtained by the following sequence of steps:

$$\frac{dy}{y^2} = dt, \quad -\frac{1}{y} = t - C, \quad -\frac{1}{y_0} = t_0 - C, \quad y = \frac{y_0}{1 + y_0(t_0 - t)}.$$

The solution blows up at time $t^* = t_0 + 1/y_0$. Thus, in this example, the solution to IVP (2) exists, it is unique, but it blows up at a finite time. Hence, we must choose $T < t^*$.

Example 2.

$$y' = 2\sqrt{y}, \quad y(0) = 0. \tag{3}$$

One obvious solution is $y(t) \equiv 0$. The other solution, $y(t) = t^2$, is obtained by the following sequence of steps:

$$\frac{dy}{2\sqrt{y}} = dt, \quad \sqrt{y} = t + C, \quad 0 = C, \quad y = t^2.$$

Furthermore, there is a one-parameter family of solutions

$$y(t) = \begin{cases} 0, & 0 \le t \le t_0, \\ (t - t_0)^2, & t \ge t_0, \end{cases},$$

for any $0 \le t_0 < \infty$. Thus, in this example, the solution to IVP (3) exists for $0 \le T < \infty$ but it is not unique.

Example 3. *Let us consider an IVP for a linear ODE:*

$$y' = Ay + g(t), \quad y(0) = y_0, \tag{4}$$

where $A \in \mathbb{R}^{d\times d}$ is a constant matrix and $g : \mathbb{R} \to \mathbb{R}^d$. One can check directly that the solution to IVP (4) is given by

$$y(t) = e^{tA}y_0 + \int_0^t e^{(t-s)A}g(s)ds, \tag{5}$$

where the matrix exponential e^{tA} is defined as the fundamental solution matrix to

$$\Psi' = A\Psi, \quad \Psi(0) = I_{d\times d}, \tag{6}$$

or the sum of the infinite series

$$e^{tA} = \sum_{n=0}^{\infty} \frac{(tA)^n}{n!}. \tag{7}$$

If A is diagonalizable, i.e., $A = SDS^{-1}$, then $e^{tA} = Se^{tD}S^{-1}$.

In this example, the solution is unique, exists at all times, and is given by an explicit formula. However, this formula is not convenient for numerical evaluation unless $g \equiv 0$, d is small, and A is diagonalizable. Therefore, it is still more convenient to compute the solution numerically for all practical purposes.

1.4. **Basic theory for IVP.**

Theorem 1. *Consider IVP (1). Suppose f is a continuous function of t and y defined on the cylinder*

$$Q := \{t_0 \le t \le T, \|y - y_0\| \le r\}$$

and $\|f\| \le M$ on Q.

1. *Then IVP (1) has a solution which exists for $0 \le t - t_0 \le \min\left(\frac{r}{M}, T - t_0\right)$.*
2. *If, in addition, f is Lipschitz in y, i.e., for some constant L,*
$$\|f(t, y_1) - f(t, y_2)\| \le L\|y_1 - y_2\|,$$
then the solution is unique.
3. *If in addition the Jacobian matrix*
$$Df(t, y) = \left\{\frac{\partial f^i}{\partial y^j}\right\}_{i,j=1}^{d}$$
is continuous in Q then the solution y is differentiable with respect to the initial condition y_0.
4. *Suppose f also depends on parameters $u \in \mathbb{R}^m$ and the Jacobian matrix with respect to u,*
$$D_u f = \left\{\frac{\partial f^i}{\partial u^j}\right\}, \quad i = 1, \dots, d, \ j = 1, \dots, m,$$
exists and is continuous on Q for all u. Then the solution y is differentiable with respect to u.

Exercise 1. *Apply Theorem 1, part 1, to Example 1: fix $r \in \mathbb{R}$, define the cylinder Q, express M via r, and find the interval of time where the existence of the solution is guaranteed.*

Exercise 2. *Apply Theorem 1, parts 1 and 2, to Example 2. As in the previous exercise, find the interval where the existence of the solution is guaranteed. Then check that the function* $f(y) = 2\sqrt{y}$ *is not Lipschitz at* $y = 0$.

1.5. **Integral equations.** IVP (1) is equivalent to the integral equation

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds. \tag{8}$$

Integral equation (8) is used to construct various numerical methods including Runge-Kutta, Adams and BDF. Moreover, integral equation (8) gives a useful tool for proving Theorem 1 called the *Picard iteration*

$$y_{n+1}(t) = y_0 + \int_{t_0}^{t} f(s, y_n(s))ds. \tag{9}$$

When f is Lipschitz, it is easy to show that the Picard iterates converge uniformly to a unique continuous solution of the IVP on the interval of t specified in Theorem 1.

Exercise 3. *Compute the first few Picard iterates starting from* $y(t) = 1$ *for the IVP*

$$y' = y^2, \quad y(0) = 1. \tag{10}$$

Infer a general pattern and determine the interval on which the Picard iterates converge. Does it match the interval guaranteed by Theorem 1?

1.6. **Textbooks on ODE theory.**

- Witold Hurewitz, Lectures on Ordinary Differential Equations, first published in 1958 and then replicated multiple times. This is a very nice, concise and fun to read set of lectures on ODE covering a very good part of the ODE theory.
- Carmen Chicone, Ordinary Differential Equations with Applications, Springer, 1999. This is a nicely written book covering the ODE theory and its numerous applications. It is freely available online.
- E. A. Coddington and N. Levinson, Theory of Ordinary Differential Equations, 1955. This is a large textbook on the ODE theory. You can look into it if you cannot find something in the two textbooks above.

2. Construction of ODE solvers

Ref: John Strain, Lecture 2, section 2. Starting from now, we will denote the exact solution by y and the numerical solution by u. The time step will be denoted by h. In addition, we will use short-hand notation replacing the time argument t_n with the subscript n: $u_n \equiv u(t_n)$, $f_n \equiv f(t_n, u_n)$, etc.

2.1. **Approach 1: Taylor expansion.** A Taylor expansion of the solution $y(t)$ at t_n yields:

$$y(t_n + h) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \dots. \tag{11}$$

Truncating this expansion at the first-order term in h we obtain the **forward Euler** method:

$$\boxed{u_{n+1} = u_n + hf(t_n, u_n).} \tag{12}$$

Do not use the forward Euler method unless there is no other choice. It is very inaccurate!

Truncating the series in (11) at the second-order term we obtain the following *second order Taylor method* which is hardly ever used:

$$\boxed{u_{n+1} = u_n + hf(t_n, u_n) + \frac{h^2}{2}\left[\frac{\partial f}{\partial t}(t_n, u_n) + [Df(t_n, u_n)]f(t_n, u_n)\right].} \tag{13}$$

While this method is reasonably accurate (commonly used PDE solvers are approximately as accurate as this method), it is not popular because it requires additional input: the time derivative and the Jacobian matrix for the right-hand side function f. Other second-order accurate options do not require such input.

2.2. **Integral equation approach.** Let us consider the ansatz

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))ds. \tag{14}$$

We will approximate the integral using various quadrature rules. Applying the *left-hand rule* we obtain the **forward Euler** method (12). The *right-hand rule* results in the **backward Euler** method:

$$\boxed{u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}).} \tag{15}$$

While backward Euler is as inaccurate as forward Euler, backward Euler can be a reasonable choice in some cases due to its very good stability properties which we will discuss later.

The *trapezoidal rule* with u_{n+1} unknown in the right-hand side yields the method also called the **implicit trapezoidal rule**:

$$\boxed{u_{n+1} = u_n + \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right].} \tag{16}$$

The implementation of this method requires the use of a nonlinear solver for u_{n+1}. Another option is to predict u_{n+1} in the right-hand side of (16) using forward Euler. The resulting method belongs to the Runge-Kutta family and is called the **trapezoidal rule with Euler predictor**:

$$\boxed{u_{n+1} = u_n + \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_n + hf(t_n, u_n))\right].} \tag{17}$$

This method is not commonly used because, contrary to the implicit trapezoidal rule, its stability properties are not good enough, and there are other more accurate methods with stability properties similar to (17).

The *midpoint rule* gives the ansatz $u_{n+1} = u_n + hf(t_n + h/2, u_{n+1/2})$ where $u_{n+1/2}$ needs to be defined. One option is to approximate it as the half-sum of u_n and u_{n+1} resulting in the **implicit midpoint rule**:

$$\boxed{u_{n+1} = u_n + hf\left(t_n + \frac{h}{2}, \frac{u_n + u_{n+1}}{2}\right).} \tag{18}$$

Another option resulting in a different version of the implicit midpoint rule is to approximate $k \equiv f(t_n + h/2, u_{n+1/2})$ from the relationship $k = f(t_n + h/2, u_n + (h/2)k)$:

$$\boxed{k = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}k\right), \quad u_{n+1} = u_n + hk.} \tag{19}$$

We will see later that method (19) is a good choice for integrating Hamiltonian systems because it is *symplectic*. In particular, it conserves energy.

Another option is to predict $u_{n+1/2}$ using forward Euler. The resulting method, the **midpoint rule with Euler predictor**

$$\boxed{u_{n+1} = u_n + hf\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}f(t_n, u_n)\right),} \tag{20}$$

has properties similar to the trapezoidal rule with Euler predictor and hence is not used much.

A famous and commonly used method based on *Simpson's quadrature* rule

$$\int_{t_n}^{t_{n+1}} g(t)dt \approx \frac{h}{6}\left(g(t_n) + 4g(t_{n+1/2}) + g(t_{n+1})\right) \tag{21}$$

is the **four-stage fourth-order Runge Kutta method** (RK4).

$$k_1 = f(t_n, u_n), \tag{22}$$

$$k_2 = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}k_1\right), \tag{23}$$

$$k_3 = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}k_1\right), \tag{24}$$

$$k_4 = f(t_n + h, u_n + hk_3), \tag{25}$$

$$u_{n+1} = \frac{h}{6}\left(k_1 + 2(k_2 + k_3) + k_4\right). \tag{26}$$

2.3. **Polynomial interpolation approach.** Adams and BDF (backward differentiation formula) families of methods are constructed with the aid of polynomial interpolation using only values at grid point. These methods are often designed to have variable time step and variable order.

The **Adams-Bashforth** family is constructed by writing an interpolating polynomial through f_{n-k+1}, f_{n-k+2}, ..., f_{n-1}, and f_n, integrating this polynomial and evaluating its value at t_{n+1}. As you see, it involves extrapolation. On the other hand, it is a family of explicit methods, hence time-stepping is relatively cheap. The **Adams-Moulton** family

is constructed in a similar manner except for the interpolating polynomial also uses the unknown value f_{n+1} making the methods implicit. Hence time marching requires the use of a nonlinear solver. At the same time, the Adams-Moulton methods are more accurate and have better stability properties than Adams-Bashforth.

The **BDF** family is obtained by writing an interpolating polynomial through u_{n-k+1}, ..., u_n, and u_{n+1}, differentiating it, and matching its derivative at t_{n+1} with f_{n+1}.

2.4. **Undetermined coefficients.** The method of undetermined coefficients is a very general approach for constructing ODE solvers. It works as follows. First, we choose the general form of the method that contains a number of undetermined coefficients. Second, we apply consistency and stability constraints and solve for the undetermined coefficients. This method will be used to construct the Runge-Kutta methods and will be discussed in more detail later.

3. Consistency, stability, and convergence

Consistency and stability imply convergence. This easy-to-remember claim is true for ODE (and PDE) solvers. In this section, we will define these three concepts and prove the convergence theorem for first-order methods.

3.1. **Standard assumptions about solvers for IVP.** We will start with specifying the standard assumptions about a general ODE solver. Any ODE solver, i.e., a method for solving IVP $y' = f(t, y)$, $y(t_0) = y_0$, $t \in [t_0, T]$, can be written of the form

$$u_{n+1} + a_0 u_n + \ldots + a_{k-1} u_{n-k+1} = hF(u_{n+1}, u_n, \ldots, u_{n-k+1}; t_{n+1}, f, h). \tag{27}$$

We will make the following assumption about F.

Assumption 1. *F vanishes identically whenever f does.*

Assumption 2. *F is a Lipschitz function of all u-arguments whenever f is Lipschitz. Specifically, if f is Lipschitz with respect to y, there is a constant L such that*

$$\begin{aligned} &\|F(u_{n+1}, u_n, \ldots, u_{n-k+1}; t_{n+1}, f, h) - F(v_{n+1}, v_n, \ldots, v_{n-k+1}; t_{n+1}, f, h)\| \ \leq \\ &L\left(\|u_{n+1} - v_{n+1}\| + \|u_n - v_n\| + \ldots + \|u_{n-k+1} - v_{n-k+1}\|\right). \end{aligned} \tag{28}$$

3.2. **Classification of methods for IVP.** Look at the general IVP solver given by (27).

- If $k = 1$, the method is *one-step*, otherwise it is *multi-step*.
- If F does not depend on u_{n+1}, the method in *explicit*, otherwise it is *implicit*.
- If F is a linear function of the right-hand side f, the method is *linear*, otherwise, it is *nonlinear*.

3.3. **Definition of convergence.** Starting from this section, we will shift time so that the starting time is zero. In this case, the total number of steps N and the time step h satisfy a nice relationship $Nh = T$.

Definition 1. *A method for IVP is convergent if for all sufficiently smooth right-hand sides f, the numerical solution u_n converges to the true solution, i.e.,*

$$\max_{0\le t_n\le T}\|u_n - y(t_n)\| \to 0 \quad \text{as} \quad h\to 0 \quad \text{and} \quad u_j \to y_j,\ 0\le j\le k-1. \tag{29}$$

The method is accurate of order p if

$$\max_{0\le t_n\le T}\|u_n - y(t_n)\| = O(h^p) + O(\|u_0 - y_0\|) + \ldots + O(\|u_{k-1} - y_{k-1}\|) \tag{30}$$

as $h\to 0$ and the initial values converge.

3.4. **Consistency.**

Definition 2. *The local truncation error τ of method (27) applied to a smooth solution y of an IVP $y' = f(t,y)$, $y(0) = y_0$, is the error committed in one step starting from exact values:*

$$\tau_{n+1} = y_{n+1} + a_0 y_n + \ldots + a_{k-1}y_{n-k+1} - hF(y_{n+1}, y_n, \ldots, y_{n-k+1}; t_{n+1}, f, h). \tag{31}$$

Definition 3. *Method (27) is consistent of order p if*

$$\tau_{n+1} = O(h^{p+1}) \quad \forall 0\le n\le T/h \tag{32}$$

Note that consistency does not imply convergence. Consistency only tells us about the error committed over one step starting from the exact values.

Exercise 4. *Show that methods (16), (17), (18), (19), and (20) are consistent of order 2.*

Let us show that the trapezoidal rule with Euler predictor (17) is consistent of order 2. We need to plug in the exact solution to the ODE into the method and Taylor-expand all terms at t_n. We note that $f(t_n, y_n) = y_n'$ and

$$\begin{aligned} f(t_{n+1}, y_n + hf(t_n,y_n)) &= f(t_n,y_n) + h\frac{\partial f}{\partial t} + [Df(t_n,y_n)]f(t_n,y_n)h + O(h^2) \\ &= y_n' + hy_n'' + O(h^2). \end{aligned} \tag{33}$$

Here $Df(t_n,y_n)$ is the Jacobian matrix consisting of the partial derivatives of the components of f with respect to components of y evaluated at (t_n,y_n). For brevity, we will omit subscript n. Plugging in y and (33) into (17) and Taylor-expanding y_{n+1} we obtain:

$$y + hy' + \frac{h^2}{2}y'' - y - \frac{h}{2}y' - \frac{h}{2}y' - \frac{h^2}{2}y'' = O(h^3).$$

Note that we did not check that the coefficient at h^3 does not vanish. In order to do it, we would need to write out more expansion terms in (33) which is quite tedious. We will discuss how to simplify this process later in this chapter.

Exercise 5. *Show that the three-step Adams-Bashforth method*

$$u_{n+1} = u_n + h\left(\frac{23}{12}f(t_n,u_n) - \frac{4}{3}f(t_{n-1}u_{n-1}) + \frac{5}{12}f(t_{n-2},u_{n-2})\right) \tag{34}$$

is consistent of order 3.

Exercise 6. *Show that the three-step Adams-Moulton method*

$$u_{n+1} = u_n + h\left(\frac{9}{24}f(t_{n+1},u_{n+1}) + \frac{19}{24}f(t_n,u_n) - \frac{5}{24}f(t_{n-1}u_{n-1}) + \frac{1}{24}f(t_{n-2},u_{n-2})\right) \tag{35}$$

is consistent of order 4.

Exercise 7. *What is the order of consistency of the following explicit two-step method?*

$$u_{n+1} + 9u_n - 10u_{n-1} = \frac{h}{2}\left(13f(t_n,u_n) + 9f(t_{n-1}u_{n-1})\right) \tag{36}$$

3.5. **Stability.** A consistent method might not be convergent because of the way numerical errors accumulate over time steps. For example, consider the method (36). It is consistent with order 2. However, it is unstable. An easy way to see it is to apply it to solving the IVP $y' = 0$, $y(0) = a$. To initiate the method, we need two initial values $u(0) = u_0$ and $u(h) = u_1$. Since the right-hand side is zero, the method becomes the following linear recurrent relationship:

$$u_{n+1} + 9u_n - 10u_{n-1} = 0. \tag{37}$$

The general solution to (37) is $u_n = Ar_1^n + Br_2^n$ where r_1 and r_2 are the roots to the characteristic equation $r^2 + 9r - 10 = 0$, i.e., $r_1 = 1$, $r_2 = -10$. In order to obtain the constant solution $u_n = a$, we need $u_1 = u_2 = a$. Then $A = a$ and $B = 0$. If either of these values u_1 or u_2, will be slightly perturbed, the coefficient B will be nonzero and hence the solution will blow up. Note that the smaller the time step h will be, the more the solution will blow up over a fixed interval of time.

This example shows that besides requiring that the errors committed at each time step be small, they also need to accumulate stably.

Definition 4. *An IVP method is stable if and only if the numerical solution is Lipschitz with respect to perturbations of the initial values and the right-hand side. I.e., there is a Lipschitz constant S such that for any sequence of vectors δ_n, the solution v_n of the perturbed method*

$$v_{n+1} + a_0v_n + \ldots + a_{k-1}v_{n-k+1} = hF(v_{n+1},v_n,\ldots,v_{n-k+1};t_{n+1},f,h) + h\delta_{n+1} \tag{38}$$

with initial values $v_0 = u_0 + \delta_0$, ..., $v_{k-1} = u_{k-1} + \delta_{k-1}$ satisfies

$$\max_{0\le t_n\le T}\|u_n - v_n\| \le S\max_{0\le t_n\le T}\|\delta_n\|. \tag{39}$$

3.6. **Convergence of one-step methods.**

Theorem 2. *A one-step method*

$$u_{n+1} = u_n + hF(u_{n+1},u_n;t_{n+1},h,f) \tag{40}$$

which is consistent of order p converges with order p of accuracy for any IVP where f is sufficiently smooth (i.e., $f \in C^{p+1}$ in an appropriate cylinder).

This theorem implies that *a consistent one-step method is always stable.*

Proof. The local truncation error at step $n+1$ is given by

$$\tau_{n+1} = y_{n+1} - y_n - hF(y_{n+1}, y_n; t_{n+1}, h, f). \tag{41}$$

For the numerical solution u we have:

$$0 = u_{n+1} - u_n - hF(u_{n+1}, u_n; t_{n+1}, h, f). \tag{42}$$

Subtracting (42) from (41) we obtain

$$\tau_{n+1} = [y_{n+1} - u_{n+1}] - [y_n - u_n] - h\,[F(y_{n+1}, y_n; t_{n+1}, h, f) - F(u_{n+1}, u_n; t_{n+1}, h, f)]\,. \tag{43}$$

Introducing the notation $e_n = \|y_n - u_n\|$ and $\tau = \max_n \|\tau_n\|$, and using the fact that F is Lipschitz with respect to all u-arguments by Assumption 2, we obtain

$$e_{n+1} \le e_n + hL(e_n + e_{n+1}) + \tau. \tag{44}$$

Equation (44) can be rewritten as

$$e_{n+1} \le e_n \frac{1+hL}{1-hL} + \frac{\tau}{1-hL}. \tag{45}$$

We choose h small enough so that the coefficients in (45) are positive. By assumption, $\tau = O(h^{p+1})$. To obtain a bound for e_{n+1}, we consider the corresponding linear inhomogeneous recurrence relationship

$$x_{n+1} = ax_n + b, \quad \text{where} \quad a = \frac{1+hL}{1-hL}, \quad b = \frac{\tau}{1-hL}. \tag{46}$$

The solution to (46) is

$$x_n = a^n x_0 + \frac{a^n - 1}{a-1} b. \tag{47}$$

Since the coefficients a and b are positive, we obtain the following bound on e_n:

$$e_n \le \left(\frac{1+hL}{1-hL}\right)^n e_0 + \frac{\left(\frac{1+hL}{1-hL}\right)^n - 1}{\left(\frac{1+hL}{1-hL}\right) - 1} \frac{\tau}{1-hL}. \tag{48}$$

You can easily check that if $3hL < 1$ then

$$a = \frac{1+hL}{1-hL} \le 1 + 3hL. \tag{49}$$

It is convenient to simplify (48) using the following bound:

$$a = \frac{1+hL}{1-hL} \le 1 + 3hL \le e^{3hL} \quad \text{for} \quad hL < \frac{1}{3}. \tag{50}$$

We observe that $nh \le hN = T$, where N is the total number of time steps. Hence $a^n \le e^{3LT}$. We also need to calculate $a - 1$:

$$a - 1 = \frac{1+hL}{1-hL} - 1 = \frac{2hL}{1-hL}. \tag{51}$$

Then the bound (48) can be simplified to

$$e_n \le e^{3LT} e_0 + \frac{(e^{3LT} - 1)(1-hL)}{2hL} \frac{\tau}{1-hL} = e^{3LT} e_0 + \frac{e^{3LT} - 1}{2L} \frac{\tau}{h}. \tag{52}$$

Now we recall that $\tau = O(h^{p+1})$. Therefore, equation (52) shows that the error decays as $O(h^p) + O(e_0)$ as $h \to 0$ uniformly on the interval $[0, T]$. Hence, the method converges. □

4. Stiff problems

The operational definition of stiff problems is the following: *we call an IVP stiff if we need to use an implicit method in order to obtain its solution at a reasonable computational cost.* Usually, stiff problems are associated with eigenvalues with large and negative real parts of the matrix A in the linear part in the right-hand side of ODEs of the form $y' = Ay + N(y,t)$ where $N(t,y)$ is a nonlinear function. Often such ODEs come from PDEs discretized in space. For example, consider the heat equation in 1D:

$$y_t = y_{xx} + f(t,x), \quad 0 < x < 1, \quad y(0,t) = y(1,t) = 0, \quad y(0,x) = y_0(x). \tag{53}$$

We partition the interval $[0,1]$ into $N+2$ points $0 = x_0 < x_1 < \ldots < x_N < x_{N+1} = 1$ where $x_j = jh$, $h = 1/(N+1)$. Since the solution at the boundary points is prescribed by the boundary conditions, we need to compute the solution only along the interior grid lines $x = hj$. Let $u_j(t)$ be the solution along the grid line $x_j = hj$. We also set $u(t) = [u_1(t), \ldots, u_N](t)^\top$. We use the second-order central difference scheme to approximate y_{xx}:

$$y_{xx}(t, x_j) = \frac{y(t, x_{j+1}) - 2y(t, x_j) + y(t, x_{j-1})}{h^2} + O(h^2). \tag{54}$$

Written in the matrix form, the resulting system of ODEs takes the form:

$$u' = -\frac{1}{h^2} Au + F \tag{55}$$

where

$$A := \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 \end{bmatrix}, \quad F = \begin{bmatrix} f(t,x_1) \\ f(t,x_2) \\ \vdots \\ f(t,x_N) \end{bmatrix}, \tag{56}$$

Exercise 8. *Show that the eigenvalues of A are $\lambda_k = 4\left(\sin \frac{\pi k}{2(N+1)}\right)^2$ and the corresponding eigenvectors are*

$$v_k = \frac{2}{N+1}\left[\sin\left(\frac{\pi k}{N+1}\right), \sin\left(\frac{2\pi k}{N+1}\right), \ldots, \sin\left(\frac{N\pi k}{N+1}\right)\right]^\top, \quad k = 1, 2, \ldots, N. \tag{57}$$

Therefore, the eigenvalues of the matrix $-h^{-2}A$ in the right-hand side of (55) are $-4h^{-2}\left(\sin \frac{\pi k}{2(N+1)}\right)^2$. $k = 1, 2, \ldots, N$. For $k = N$, the sine is close to 1, and hence the largest negative eigenvalue is about $-4h^{-2}$. This number tends to $-\infty$ as $h \to 0$.

A model problem designed to illuminate the effect of the large negative eigenvalues in the linear part of the right-hand side of an ODE is the Prothero-Robinson problem

$$y' = -L(y - \phi(t)) + \phi'(t), \quad y(0) = y_0. \tag{58}$$

Its numerical investigation is detailed in J. Strain's Lecture 4 `04.ps.pdf`.

Stiffness can be a purely nonlinear effect too. A classical example is the Van der Pol oscillator

$$\begin{cases} \dot{x} = y, \\ \dot{y} = \mu(1 - x^2)y - x, \end{cases} \tag{59}$$

with μ being a large parameter.

5. Linear stability theory

Linear stability region analyzes the action of methods for IVP on a very simple class of problems:

$$y' = \lambda y, \quad y(0) = y_0 \neq 0, \quad \lambda \in \mathbb{C}, \ \mathsf{Re}(\lambda) < 0, \text{ is a constant.} \tag{60}$$

The requirement that $\mathsf{Re}(\lambda) < 0$ means that the exact solution to this problem tends zero as $t \to \infty$.

A method for IVP with a fixed time step h applied to (60) yields a numerical solution u_n. With h fixed, we let $n \to \infty$ which means that $t \to \infty$.

Definition 5. *The region of absolute stability (RAS) (a. k. a. the stability region) is the region of the complex plane of values $h\lambda$ such that $u_n \to 0$ as $n \to \infty$, i.e.,*

$$RAS = \{(h\lambda) \in \mathbb{C} \mid u_n \to 0 \text{ as } n \to \infty\}. \tag{61}$$

Let us find the stability regions for some methods we have encountered. The forward Euler method applied to (60) is

$$u_{n+1} = u_n + h\lambda u_n = (1 + h\lambda)u_n. \tag{62}$$

The function $R(z)$, $z := h\lambda$, such that the numerical solver advances the solution according to $u_{n+1} = R(z)u_n$ is called the *stability function*. Therefore, the numerical solution is the geometric series

$$u_n = (1 + h\lambda)^n u_0. \tag{63}$$

The numerical solution u_n tends to zero if and only if $|1 + h\lambda| < 1$. Let $z = h\lambda \equiv a + ib \in \mathbb{C}$. Then the RAS is defined by the condition

$$|1 + h\lambda|^2 = |1 + z|^2 = (a + 1)^2 + b^2 < 1. \tag{64}$$

Hence, the RAS is the interior of the circle of radius 1 centered at $-1 + 0i$.

Exercise 9. *Show that the RAS for the backward Euler is the exterior of the the circle of radius 1 centered at $1 + 0i$.*

Exercise 10. *Show that the RAS for the implicit trapezoidal rule*

$$u_{n+1} = u_n + \frac{h}{2}\left(f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right)$$

is the the set $\mathsf{Re}(z) < 0$.

Let us find the RAS for the midpoint rule with Euler predictor:

$$u_{n+1} = u_n + hf\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}f(t_n, u_n)\right). \tag{65}$$

Applying this method to (60) we obtain

$$u_{n+1} = u_n + h\lambda\left(u_n + \frac{h}{2}\lambda u_n\right) = \left(1 + h\lambda + \frac{(h\lambda)^2}{2}\right)u_n. \tag{66}$$

Note that this method is second-order accurate and its *stability function* is the truncated Taylor expansion for $e^{\lambda h}$ at order 2. It is easy to check that this is not a coincidence. We will see a bit later that the stability function of explicit s-stage Runge-Kutta methods are polynomials of degree s. In the case if the number of stages s coincides with the order p of the method, the stability function must be the truncated Taylor series of $e^{h\lambda}$ at order p in order to satisfy the consistency requirement. Indeed, write the expansion

$$y(t_n + h) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \ldots = \left(1 + h\lambda + \frac{(h\lambda)^2}{2} + \ldots\right)y_n$$

and finish the argument. Next, we use write a code to compute and visualize the RAS. Here is my Python code:

```
import numpy as np
import matplotlib.pyplot as plt

nx = 100
ny = 160
x = np.linspace(-4,1,nx)
y = np.linspace(-4,4,ny)
xg,yg = np.meshgrid(x,y)

z = xg + 1j*yg
f = 1 + z + 0.5*z*z
absf = (f.real)**2 + (f.imag)**2

plt.rcParams.update({'font.size': 22})
fig, ax = plt.subplots(figsize=(8,8))
plt.contourf(xg,yg,absf,np.arange(2))
plt.title("RAS")
plt.xlabel("Re(z)")
plt.ylabel("Im(z)")
ax.set_aspect(1)
plt.grid(color='k', linestyle='--', linewidth=0.5)
plt.savefig('RK2_RAS.pdf')
```

The resulting figure is displayed in Figure 1

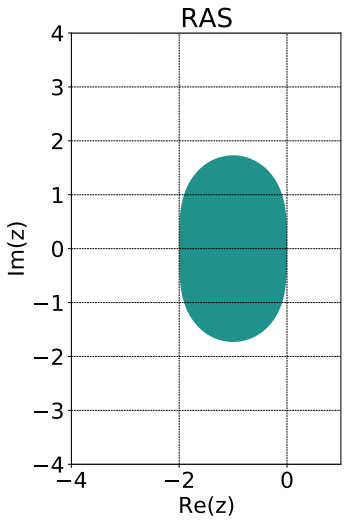


FIGURE 1. The region of the absolute stability ofr the midpoint rule with Euler predictor.

6. Runge-Kutta methods

Read Wiki article "List of Runge-Kutta methods" to learn what the Butcher array is and to see numerous examples of Runge-Kutta methods and their classification: ERK (explicit) vs IRK (implicit). ERK methods require s (the number of stages) evaluations of the right-hand side $f(t, y)$ per time step. IRK methods are much more expensive as one needs to solve a nonlinear system of equations with sd (d is the space dimension) unknowns at each time step. Usually, this is done by NEwton's method. The cost of inversion of the Jacobian matrix at every Newton's iteration is $O(s^3d^3)$. Among IRK special cases of interest are DIRK (diagonally implicit) where one can solve for each stage at-a-time reducing the computational cost to $O(sd^3)$. The cost can be further reduced for SDIRK methods (diagonally implicit with the same number along the diagonal).

6.1. **Facts about RK methods.**

- All RK methods of order $p \le 4$ have the same order for y being vector or scalar function. If $p > 4$, the order for y vector-function can be lower than for the single 1D ODE.
- Any s-stage ERK has order $p \le s$.

  Exercise 11. *Check this using the ODE* $y' = y$.

- Any ERK method of order p must have
  - $s \ge p + 1$ stages if $p > 4$,
  - $s \ge p + 2$stages if $p > 6$, and

  - $s \geq p+3$ stages if $p > 7$.
- The number of order conditions grows rapidly with the order p – see Table 1.

TABLE 1. The number of order conditions

p	1	2	3	4	5	6	7	8	9	10
N	1	2	4	8	17	37	85	200	480	1205

6.2. **Consistency.** First, we will show that

$$\boxed{c_j = \sum_{k=1}^{s} a_{jk}, \quad 1 \leq j \leq s.} \tag{67}$$

Indeed, let us consider the ODE $y' = t$ and apply an RK method to it. At each stage j at time step n, we have $k_j = t_n + c_j h$. Now let us treat t as a function are rewrite the ODE in the autonomous form:

$$\frac{d}{d\tilde{t}} \begin{bmatrix} y \\ t \end{bmatrix} = \begin{bmatrix} t \\ 1 \end{bmatrix}. \tag{68}$$

Here $\tilde{t}$ is a new independent variable. At stage j at time step n, we have

$$\begin{bmatrix} (k_y)_j \\ (k_t)_j \end{bmatrix} = \begin{bmatrix} t_n + h\sum_{l=1}^{s} a_{jl}(k_t)_l \\ 1 \end{bmatrix} = \begin{bmatrix} t_n + h\sum_{l=1}^{s} a_{jl} \\ 1 \end{bmatrix}. \tag{69}$$

Since $(k_y)_j$ must coincide with k_j above, we obtain (67).

Therefore, without the loss of generality, we can derive order conditions for autonomous systems ODEs. These conditions will involve only A and b. Then the components of the vector c are obtained using (67). Since the stages are nested functions, the direct calculation of the truncation error becomes tedious for $s > 2$. Instead, we will use the observation that the local truncation error can be written as a series in h:

$$\tau_{n+1} = y(t_n + h) - y(t_n) - h\sum_{l=1}^{s} b_l k_l = C_0 + C_1 h + C_2 \frac{h^2}{2} + \ldots + C_p \frac{h^p}{p!} + O(h^{p+1}), \tag{70}$$

where

$$C_r = \frac{d^r}{dh^r} \tau_{n+1}(h = 0). \tag{71}$$

A method of order p must have $C_0 = C_1 = \ldots = C_p = 0$. We see immediately that $C_0 = \tau_{n+1}(h = 0) = 0$. Let us compute C_1. We need the first derivative of τ. We will use the Taylor expansion of $y(t_n + h) = y + y'h + y''\frac{h^2}{2} + \ldots$:

$$\frac{d}{dh}\tau_{n+1} = y' + hy'' + \frac{h^2}{2}y''' + \ldots - h\sum_{l=1}^{s} b_l \frac{dk_l}{dh} - \sum_{l=1}^{s} b_l k_l. \tag{72}$$

Then, taking into account that each stage k is equal to y' at $h = 0$ we obtain

$$C_1 = \frac{d}{dh}\tau_{n+1}(h=0) = y' - \sum_{l=1}^{s} b_l y' = y'\left(1 - \sum_{l=1}^{s} b_l\right). \tag{73}$$

This gives us the single 1st order condition

$$\boxed{\sum_{l=1}^{s} b_l = 1} \tag{74}$$

Next, we calculate the second derivative of τ_{n+1}:

$$\frac{d^2}{dh^2}\tau_{n+1} = y'' + hy''' + \ldots - h\sum_{l=1}^{s} b_l \frac{d^2 k_l}{dh^2} - 2\sum_{l=1}^{s} b_l \frac{dk_l}{dh}. \tag{75}$$

We need to compute the derivatives of the stages and evaluate them at $h = 0$. The superscript i denotes the ith component of the vector-function f.

$$\begin{aligned} \frac{d}{dh}k_l^i &= \frac{d}{dh}f^i\left(y_n + h\sum_q a_{lq}\right) \\ &= \sum_m \frac{\partial f^i}{\partial y^m}\left(\sum_q a_{lq}k_q^m + h\sum_q a_{lq}\frac{d}{dh}k_q^m\right) \end{aligned} \tag{76}$$

Setting $h = 0$ in the derivative above we find:

$$\frac{d}{dh}k_l^i(h=0) = \sum_m \frac{\partial f^i}{\partial y^m}\left(\sum_q a_{lq}f_q^m\right) = \frac{d^2}{dt^2}y^i \sum_q a_{lq} = (y^i)'' \sum_q a_{lq} \tag{77}$$

Then

$$C_2 = \frac{d^2}{dh^2}\tau_{n+1}(h=0) = y''\left(1 - 2\sum_{l=1}^{s} b_l \sum_q a_{lq}\right). \tag{78}$$

This gives us the second-order condition

$$\boxed{2\sum_{l=1}^{s} b_l \sum_q a_{lq} = 1.} \tag{79}$$

Proceeding in a similar manner, one can obtain two third-order conditions:

$$\boxed{3\sum_l b_l \sum_q a_{lq} \sum_r a_{lr} = 1, \quad 6\sum_l b_l \sum_q a_{lq} \sum_r a_{qr} = 1.} \tag{80}$$

6.3. **Stability.** Since RK methods are one-step, they are stable as soon as the update function F in

$$u_{n+1} = u_n + hF(u_{n+1}, u_n; t_{n+1}, h, f) = u_n + h\sum_{l=1}^{s} b_l k_l \tag{81}$$

is Lipschits with respect to its u-arguments. Obviously, F is Lipschitz if and only if the vector of stages k is Lipschitz Let us show that the vector of stages is Lipschitz. If $y : \mathbb{R} \to \mathbb{R}^d$ then $k \in \mathbb{R}^{sd}$. We have the following equation for the stage vector k:

$$k(u) = f(t + \hat{c}h, u + h\hat{A}k(u)), \quad \text{where} \quad \hat{c} = c \otimes 1_{d\times 1}, \quad \hat{A} = A \otimes I_{d\times d}, \tag{82}$$

where the symbol $\otimes$ denotes the Kronecker product. To check that k is Lipschitz, we consider the difference $k(u) - k(v)$ and show that its norm is bounded by some constant times $\|u - v\|$:

$$\begin{aligned} k(u) - k(v) &= f(t + \hat{c}h, u + h\hat{A}k(u)) - f(t + \hat{c}h, v + h\hat{A}k(v)); \\ \|k(u) - k(v)\| &= \|f(t + \hat{c}h, u + h\hat{A}k(u)) - f(t + \hat{c}h, v + h\hat{A}k(v))\| \\ &\le L_f \left\|u - v + h\hat{A}(k(u) - k(v))\right\| \\ &\le L_f\|u - v\| + hL_f\|\hat{A}\|\|k(u) - k(v)\|. \end{aligned}$$

Therefore,

$$\|k(u) - k(v)\| \le \frac{L_f}{1 - hL_f\|\hat{A}\|}\|u - v\|, \tag{83}$$

which shows that the stage vector is Lipschitz if h is small enough.

6.4. **Linear stability analysis.** Linear stability analysis means that the application to the ODE $y' = \lambda y$ where $\lambda \in \mathbb{C}$ is examined. Applying RK method to $y' = \lambda y$ we get:

$$k = \lambda\left(u_n + hAk\right), \quad k = \lambda u_n (I_{s\times s} - h\lambda A)^{-1} 1_{s\times 1}$$

$$u_{n+1} = u_n + hb^\top k = \left[1 + h\lambda b^\top (I_{s\times s} - h\lambda A)^{-1} 1_{s\times 1}\right] u_n. \tag{84}$$

Therefore, the stability function is

$$R(z) = 1 + zb^\top (I_{s\times s} - zA)^{-1} 1_{s\times 1}. \tag{85}$$

Note that the following series expansion is valid if $|z|$ is small enough:

$$(I_{s\times s} - zA)^{-1} = 1 + zA + z^2A^2 + z^3A^3 + \dots. \tag{86}$$

You can check this identity directly by multiplying both sides by $I_{s\times s} - zA$. If the RK method is explicit, i.e., it is an ERK method, then the Butcher matrix A is nilpotent. Specifically $A^q = 0$ for $q \ge s$.

Exercise 12. *Show that if an $s \times s$ A is of the form*

$$A = \begin{bmatrix} 0 & & & \\ * & 0 & & \\ * & * & 0 & \\ \vdots & & \ddots & \end{bmatrix}, \tag{87}$$

i.e., it is strictly lower triangular and has nonzeros on the first subdiagonal, then A^2 has all zeros on the first subdiagonal and above it, A^3 has zeros on the second subdiagonal and above it, and so on, and $A^s = 0$.

This means that the stability function is a polynomial of degree s:

$$R(z) = 1 + zb^\top \left[I_{s\times s} + zA + z^2A^2 + \ldots + z^{s-1}A^{s-1} \right] 1_{s\times 1}. \tag{88}$$

This means that the stability region defined by

$$RAS = \{z \in \mathbb{C} \mid |R(z)| < 1\} \tag{89}$$

is bounded for any ERK. Hence no ERK is A-stable, i.e., no ERK is suitable for stiff problems.

6.5. **Stiff accuracy or L-stability.** Linear stability analysis shows that ERK methods are not suitable for stiff problems. The condition of A-stability that the negative half-plane belongs to the RAS is not strong enough for stiff problems. Let us consider the implicit trapezoidal rule (not an RK method):

$$u_{n+1} = u_n + \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1}) \right]. \tag{90}$$

Its stability function is

$$R(z) = \frac{1 + z/2}{1 - z/2}, \tag{91}$$

and its RAS is the negative half-plane $\{z \mid \mathsf{Im}(z) < 0\}$. However, in the stiff limit $|z| \to \infty$, the stability function for the trapezoidal rule tends to -1, which means that if λ is large, real, and negative, the numerical solution to $y' = \lambda y$ will not decay rapidly. Moreover, if the time step h is fixed, the larger is $|\lambda|$ (along the negative real semiaxis), the slower the numerical solution to $y' = \lambda y$ decays.

To address this problem, the requirement of L-stability was introduced.

Definition 6. *An RK method is L-stable (or stiffly accurate) if it is A-stable and the stability function satisfies*

$$\lim_{|z|\to\infty} R(z) = 0. \tag{92}$$

Note that the stability function of the RK methods is a rational function. Hence *an RK method is L-stable if it is A-stable and the degree of the numerator of the stability function is less than the degree of its denominator.*

Exercise 13. *Show that the DIRK method with the Butcher array of the form*

γ	γ	0
1	$1-\gamma$	γ
	$1-\gamma$	γ

and $\gamma = 1 - 1/\sqrt{2}$ is L-stable (or stiffly accurate) and 2nd order accurate. This method is often referred to as DIRK2.

Exercise 14. (1) *Show that the family of DIRK methods with the Butcher array of the form*

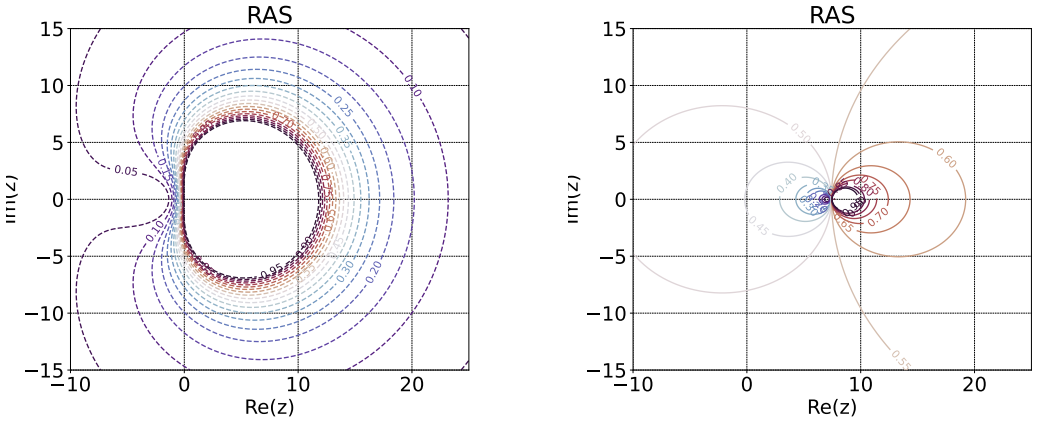
$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array}$$

is 2nd order accurate for any $\gamma \in (0, 1]$.

(2) *Check that* $\gamma = 1/2 + \sqrt{3}/6$ *renders this method 3rd order accurate.*

(3) *Find* γ *such that this method is L-stable. A-stability can be checked, e.g., by plotting the region of absolute stability.*

The level sets from 0 to 1 with step 0.05 for the methods in the exercises above with $\gamma = 1 - 1/\sqrt{2}$ and $\gamma = 1/2 + \sqrt{3}/6$ respectively are displayed in Fig. 2 (a) and (b), respectively. While both of these methods are A-stable, the stability function only of the first of them decays to zero as $|z| \to \infty$.



(a) (b)

FIGURE 2. The level sets of the stability functions of the DIRK methods in exercises 13 and 14.

6.6. **Stepsize control and dense output.** Read J. Strain's Lecture 9 `09.ps.pdf`.

6.6.1. *Stepsize control.* Step size control in good modern RK solvers is accomplished due to the use of two methods sharing the Butcher array. For example, consider the Butcher array for DOPRI5(4):

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
b	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\hat{b}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

The vector b makes the method of order 4, while $\hat{b}$ makes it of order 5. At each step, the solution of the method of order 5 is used as the output. In addition, the solution by the method of order 4 is also evaluated starting from the output of the previous time step and used for error estimate:

$$e = h(bk - \hat{b}k), \tag{93}$$

where k is the vector of stages. The step is accepted if

$$\texttt{abs(e)} < \texttt{atol} + \texttt{rtol} * \texttt{abs(û)}, \tag{94}$$

where `aatol` and `rtol` are the absolute and relative error tolerances, and `hatu` is the solution of the higher order method at the given step. Otherwise, the step is rejected and the stepsize is reduced.

Looking at the DOPRI5(4) Butcher array you can notice that the last row of the top part of the array is evaluated at t_{n+1} and the coefficients in it match the vector b for the 4th-order method. This property is called *first same as last* (FSAL) allows one to eliminate the extra computation of the lower-order solution because it is the last stage computed anyway.

Also notice that the last two stages are both evaluated at t_{n+1}. This gives rise to an automatic stiffness detector described in Strain's Lecture 9.

6.6.2. *Dense output.* The high orders of RK methods lead to their high efficiency: a highly smooth accurate solution can be computed with a few time steps. However, suppose the user needs to know the solution at time values in between the time steps. The dense output feature is designed to address this need. The solution between time steps can be found

using interpolation. The interpolation can be always chosen to be the Hermite interpolation as both u and u' are known at the end of the time interval.

Let us recall how to construct a Hermite interpolant on $[0,1]$ with the data $u(0) = u_0$, $u(1) = u_1$, $u'(0) = u'_0$, $u'(1) = u'_1$. Now suppose that we want to construct a polynomial such that its values and its derivatives match with those of f at given points. Following Lagrange's idea, we design a polynomial $p_0(x)$ such that

$$p_0(0) = 1, \quad p'_0(0) = 0, \quad p_0(1) = 0, \quad p'_0(1) = 0.$$

Note that then the polynomial $p_1(x) := p_0(1-x)$ satisfies:

$$p_1(0) = 0, \quad p'_1(0) = 0, \quad p_1(1) = 1, \quad p'_1(1) = 0.$$

We also need to design a polynomial $g_0(x)$ such that

$$g_0(0) = 0, \quad g'_0(0) = 1, \quad g_0(1) = 0, \quad g'_0(1) = 0.$$

Then the polynomial $g_1(x) := -g_0(1-x)$ satisfies

$$g_1(0) = 0, \quad g'_1(0) = 0, \quad g_1(1) = 0, \quad g'_1(1) = 1.$$

For each polynomial $p_0(x)$ and $g_0(x)$ we have four conditions to meet. Hence they should have four coefficients which means that they should be cubic. We can find $p_0(x)$ by setting it to

$$p_0(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3, \quad \text{and, respectively,} \quad p'_0(x) = a_1 + 2a_2 x + 3a_3 x^2,$$

and solving the system of four linear equations:

$$\begin{aligned} &p_0(0) = a_0 = 1,\\ &p'_0(0) = a_1 = 0,\\ &p_0(1) = a_0 + a_1 + a_2 + a_3 = 0,\\ &p'_0(1) = a_1 + 2a_2 + 3a_3 = 0. \end{aligned}$$

We find:

$$p_0(x) = 1 - 3x^2 + 2x^3.$$

Similarly, we find

$$g_0(x) = x(1-x)^2.$$

Now we write out the desired interpolating polynomial:

$$\mathcal{H}_{[0,1]}(x) = u_0 p_0(x) + u_1 p_0(1-x) + u'_0 g_0(x) - u'_1 g_0(1-x). \tag{95}$$

Now we rescale this polynomial to the interval $[0,h]$:

$$\mathcal{H}_{[0,h]}(x) = u_0 p_0(x/h) + u_1 p_0(1-x/h) + h\left[u'_0 g_0(x/h) - u'_1 g_0(1-x/h)\right]. \tag{96}$$

The interpolation error for this polynomial is $O(h^4)$.

RK methods facilitate the construction of higher-order interpolation polynomials because the stages provide approximations to u and u' at the points $t_n + c_j h$.

7. Linear multistep methods

Linear multistep methods are of the form

$$u_{n+1} + \alpha_0 u_n + \alpha_1 u_{n-1} + \ldots + \alpha_{k-1} u_{n-k+1} = h_n \left(\beta_{-1} f_{n+1} + \beta_0 f_n + \ldots + \beta_{k-1} f_{n-k+1}\right), \tag{97}$$

where $f_j = f(t_j, u_j)$ and $h_n = t_{n+1} - t_n$.

Linear multistep methods are cheaper than RK methods of the same order as they require only one function evaluation per step if the method is explicit, and only one solution to a $d \times d$ generally nonlinear system per step, if the method is implicit.

Linear multistep methods are implemented in professionally written routines allowing for a variable timestep and a variable order.

On the downside, there are two issues associated with linear multistep methods.

- First, k-step methods with $k \geq 2$ require to initialize the solution at times t_0, ..., t_{k-1}, while only the solution at t_0 is defined by the initial condition to the ODE. Hence, another method, e.g. RK of the same order should be used to get started.
- Second, linear multistep methods have inferior stability issues in comparison with RK methods. As you have seen in the homework, the RASs of ERK methods tend to grow as their order grows, while the RASs of linear multistep methods of the Adams family shrink with the increase of the order and the RASs of the BDF family lose a region around the origin as their order increases. In fact, the only A-stable linear multistep method is the trapezoidal rule, which, as we discussed is not a good choice for stiff problems. Higher-order linear multistep methods at best can be *$A(\alpha)$-stable*, i.e., *their RASs contain a sector of angle 2α around the negative real semiaxis in the complex plane.*

7.1. Adams and BDF families: setup.

The Adams family of linear multistep methods is constructed from the ansatz

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t)) dt. \tag{98}$$

The function $f(t, y(t))$ in a k-step method is approximated by an interpolation polynomial through (t_{n-k+1}, f_{n-k+1}), ..., $f(t_n, y_n)$ if the method is explicit, and, additionally, through $f(t_{n+1}, y_{n+1})$, if the method is implicit. The explicit methods comprise the *Adams-Bashforth* subfamily, while the implicit methods constitute the *Adams-Moulton* subfamily. Then this polynomial is integrated exactly resulting in methods of the form

$$u_{n+1} = u_n + h_n \left(\beta_0 f_n + \ldots + \beta_{k-1} f_{n-k+1}\right), \quad \text{Adams-Bashforth} \tag{99}$$

$$u_{n+1} = u_n + h_n \left(\beta_{-1} f_{n+1} + \beta_0 f_n + \ldots + \beta_{k-1} f_{n-k+1}\right), \quad \text{Adams-Moulton} \tag{100}$$

For the case where the timestep is fixed, the coefficients for Adams-Bashforth and Adams-Moulton methods can be looked up in the Wiki article "Linear multistep methods".

The BDF (backward differentiation formula) family is derived from the ansatz

$$\frac{d}{dt} y(t_{n+1}) = f(t_{n+1} y(t_{n+1})). \tag{101}$$

The function $y(t)$ in the left-hand side of (101) in a k-step BDF method is approximated by an interpolation polynomial through t_{n+1}, t_n, ..., t_{n-k+1}.

Let us recall how to derive the interpolation polynomial. This can be done in two ways, Lagrange's and Newton's.

7.2. **Lagrangian interpolation.** Suppose a continuous function $f(x)$ defined on the interval $[a,b]$ is given at a finite number of points x_j, $j=0,1,2,\dots,n$, a total of $n+1$ points. We will denote $f(x_j)$ by f_j. The task of interpolation is to find a polynomial of degree at most n passing through all these points. Lagrange's approach to this task is to construct $n+1$ polynomials $L_j(x)$ of degree n such that each $L_j(x)$ is 1 at x_j and zero at all other x_k's. Then the linear combination

$$P(x) = f_0 L_0(x) + f_1 L_1(x) + \ldots + f_n L_n(x)$$

is a polynomial of degree at most n, and $P(x_j) = f_j$ for all $j=0,1,\dots,n$. The polynomial $P(x)$ written in the form above is called the Lagrange interpolation polynomial. The polynomials $L_j(x)$ are easily constructed. The error of interpolation can also be found. These results are summarized in the following theorem.

Theorem 3. *Given a function f that is defined at $n+1$ points $x_0 < x_1 < \ldots < x_n \in [a,b]$ there exists a unique polynomial of degree $\le n$ such that*

$$P_n(x_j) = f(x_j), \quad j=0,1,2,\dots,n.$$

This polynomial is given by

$$P_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x),$$

where $L_j(x)$ is defined by

$$L_j(x) = \frac{\pi_{n+1}(x)}{(x-x_j)\pi'_{n+1}(x_j)} = \frac{\prod_{k=0,k\neq j}^{n}(x-x_k)}{\prod_{k=0,k\neq j}^{n}(x_j-x_k)},$$

$\pi_{n+1}(x)$ being the nodal polynomial

$$\pi_{n+1}(x) = \prod_{k=0}^{n}(x-x_k).$$

Additionally, if f is $n+1$ times continuously differentiable in (a,b), then for any $x\in[a,b]$ there exists a value $\zeta_x \in (a,b)$ depending on x, such that

$$E_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\zeta_x)}{(n+1)!}\pi_{n+1}(x). \tag{102}$$

Proof. (1) *Existence* By construction, P_n is a polynomial of degree n passing through x_j, $j=0,1,2,\dots,n$.

(2) *Uniqueness* Suppose that there are two such polynomials, $P_n(x)$ and $Q_n(x)$. Then the polynomial $d(x) := P_n(x) - Q_n(x)$ is at most of degree n. On the other hand, it has at least $n+1$ roots x_j, $j=0,1,2,\dots,n$. Therefore, it must be identically zero.

(3) *Error formula* Consider the function

$$F(z) = f(z) - P_n(z) - [f(x) - P_n(x)]\frac{\pi_{n+1}(z)}{\pi_{n+1}(x)}.$$

This function has $n+2$ zeros at $z = x_j$, $j = 0, 1, 2, \dots, n$, and $z = x$. Recall Rolle's theorem that says that if a function $f(z)$ is continuously differentiable on $[a, b]$ and $f(a) = f(b)$ then there is $c \in (a, b)$ such that $f'(c) = 0$. Therefore, if we apply Rolle's theorem $n+1$ times we get that the function

$$F^{(n+1)}(z) = f^{(n+1)}(z) - P_n^{(n+1)}(z) - [f(x) - P_n(x)]\frac{(n+1)!}{\pi_{n+1}(x)}$$

has at least one zero in (x_0, x_n). We denote this zero by ζ_x. Therefore, taking into account that $P_n^{(n+1)}(z) \equiv 0$, we get

$$0 = f^{(n+1)}(\zeta_x) - [f(x) - P_n(x)]\frac{(n+1)!}{\pi_{n+1}(x)},$$

and Eq. (102) follows.

□

Example 4. *Let us find the Lagrange interpolant $p(x)$ passing through the points $(0, f_0)$, $(1, f_1)$, and $(2, f_2)$. The polynomial $p(x)$ is of the form*

$$\begin{aligned} p(x) &= f_0\frac{(x-1)(x-2)}{(0-1)(0-2)} + f_1\frac{(x-0)(x-2)}{(1-0)(1-2)} + f_2\frac{(x-0)(x-1)}{(2-0)(2-1)} \\ &= \tfrac{1}{2}f_0(x^2 - 3x + 2) - f_1(x^2 - 2x) + \tfrac{1}{2}f_2(x^2 - x) \\ &= f_0 + \tfrac{1}{2}(-3f_0 + 4f_1 - f_2)x + \tfrac{1}{2}(f_0 - 2f_1 + f_2)x^2. \end{aligned} \tag{103}$$

7.3. **The Newton interpolation polynomial.** Lagrangian interpolation is convenient because it gives an explicit formula for the interpolant. However, it does not provide a convenient way to modify the polynomial to accommodate additional interpolation points. An alternative form of the interpolation polynomial, the Newton form, gives such a way. The Newton interpolation formula is used, for example, for deriving linear multistep methods with varying time step for solving ODE's. The Newton interpolation formula is defined via the divided differences. We set

$$\begin{aligned} f[x_0] &= f(x_0), \\ f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0}, \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}, \\ &\dots \\ f[x_0, x_1, \dots, x_k] &= \frac{f[x_1, \dots, x_k] - f[x_0, x_1, \dots, x_{k-1}]}{x_k - x_0}, \\ &\dots . \end{aligned}$$

Theorem 4. *The polynomial interpolating $f(x)$ at x_j, $j = 0, 1, 2, \dots, n$ is given by*

$$\begin{aligned} P_n(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots \\ &\quad + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}). \end{aligned} \tag{104}$$

Example 5. *Let us find the Newton interpolant $p(x)$ passing through the points $(0, f_0)$, $(1, f_1)$, and $(2, f_2)$. The polynomial $p(x)$ is of the form*

$$p(x) = f[0] + f[0, 1](x - 0) + f[0, 1, 2](x - 0)(x - 1), \tag{105}$$

where

$$f[0] = f_0, \quad f[0, 1] = \frac{f_1 - f_0}{1 - 0} = f_1 - f_0, \quad f[1, 2] = \frac{f_2 - f_1}{2 - 1} = f_2 - f_1,$$

$$f[0, 1, 2] = \frac{f[1, 2] - f[0, 1]}{2 - 0} = \frac{1}{2}(f_2 - f_1 - (f_1 + f_0)) = \frac{1}{2}(f_0 - 2f_1 + f_2).$$

Plugging these coefficients into Eq. (105) we get

$$\begin{aligned} p(x) &= f_0 + (f_1 - f_0)x + \tfrac{1}{2}(f_0 - 2f_1 + f_2)x(x - 1) \\ &= f_0 + \tfrac{1}{2}\left(-3f_0 + 4f_1 - f_2\right)x + \tfrac{1}{2}(f_0 - 2f_1 + f_2)x^2. \end{aligned} \tag{106}$$

The polynomial $p(x)$ in Eq. (106) coincides with the one in Eq. (103) as it should.

Proof. We will proceed by induction. For $n = 1$ Eq. (104) holds. Suppose the polynomials $P[x_0, \dots, x_{n-1}](x)$ and $P[x_1, \dots, x_n](x)$ of the form of Eq. (104) interpolate f at the points $x_0, \dots, x_{n-1}$ and $x_1, \dots, x_n$ respectively. Note that both of them are of degree $n - 1$. Hence they differ by a polynomial of degree at most $n - 1$, and this polynomial has zeros at x_1, ..., x_{n-1}. Therefore,

$$P[x_1, \dots, x_n](x) - P[x_0, \dots, x_{n-1}](x) = a(x - x_1)\dots(x - x_{n-1})$$

where a is a number. Obviously, a is the difference of the leading coefficients of the polynomials $P[x_0, \dots, x_{n-1}](x)$ and $P[x_1, \dots, x_n](x)$, i.e.,

$$a = f[x_1, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}].$$

At the same time,

$$a = \frac{P[x_1, \dots, x_n](x) - P[x_0, \dots, x_{n-1}](x)}{(x - x_1)\dots(x - x_{n-1})}.$$

Now consider the polynomial

$$P[x_0, x_1, \dots, x_n](x) = P[x_0, \dots, x_{n-1}](x) + \frac{a}{x_n - x_0}(x - x_0)(x - x_1)\dots(x - x_{n-1}). \tag{107}$$

The last term of this polynomial is chosen so that it is zero at $x_0, \dots, x_{n-1}$, and the coefficient $a/(x_n - x_0)$ is our guess that we will verify below. By construction, $P[x_0, x_1, \dots, x_n](x)$

interpolates f at $x_0, \dots, x_{n-1}$. Let us verify that it also does so at $x = x_n$. We evaluate $P[x_0, x_1, \dots, x_n](x)$ at x_n and obtain:

$$\begin{aligned}
&P[x_0, x_1, \dots, x_n](x_n)\\
=&P[x_0, \dots, x_{n-1}](x_n) + \frac{a}{x_n - x_0}(x_n - x_0)(x_n - x_1)\dots(x_n - x_{n-1})\\
=&P[x_0, \dots, x_{n-1}](x_n)\\
+&\frac{P[x_1, \dots, x_n](x_n) - P[x_0, \dots, x_{n-1}](x_n)}{(x_n - x_0)(x_n - x_1)\dots(x_n - x_{n-1})}(x_n - x_0)(x_n - x_1)\dots(x_n - x_{n-1})\\
=&P[x_1, \dots, x_n](x_n) = f(x_n).
\end{aligned}$$

Therefore, the polynomial given by Eq. (107) interpolates f at x_j, $j = 0, 1, 2, \dots, n$. □

Remark The function $f[x_0, \dots, x_n]$ is a symmetric function of its arguments i.e., it does not change if we permute x_0, ..., x_n. This is because the interpolation polynomial is independent of the order of nodes.

7.4. **Adams methods: derivation.** As an example, we will derive a three-step Adams-Bashforth method with a variable timestep. Newton's interpolant through t_n, t_{n-1}, and t_{n-2} is

$$p(t) = f_n + f[t_n, t_{n-1}](t - t_n) + f[t_n, t_{n-1}, t_{n-2}](t - t_n)(t - t_{n-1}), \tag{108}$$

where

$$f[t_n, t_{n-1}] = \frac{f_n - f_{n-1}}{h_{n-1}}, \quad f[t_n, t_{n-1}, t_{n-2}] = \frac{\frac{f_n - f_{n-1}}{h_{n-1}} - \frac{f_{n-1} - f_{n-2}}{h_{n-2}}}{h_{n-1} + h_{n-2}}. \tag{109}$$

We integrate the polynomial $p(t)$ on the interval $[t_n, t_{n+1}]$ and get the 3-step Adams-Bashforth method:

$$\begin{aligned}
u_{n+1} &= u_n + \int_{t_n}^{t_{n+1}} \left[f_n + f[t_n, t_{n-1}](t - t_n) + f[t_n, t_{n-1}, t_{n-2}](t - t_n)(t - t_{n-1})\right] dt\\
&= u_n + h_n\left[f_n + f[t_n, t_{n-1}]\frac{h_n}{2} + f[t_n, t_{n-1}, t_{n-2}]\left(\frac{h_n^2}{3} + \frac{h_{n-1}h_n}{2}\right)\right]. \qquad (110)
\end{aligned}$$

Now let us calculate the coefficients of this method in the case where the timestep h is constant. We have:

$$f[t_n, t_{n-1}] = \frac{f_n - f_{n-1}}{h}, \quad f[t_n, t_{n-1}, t_{n-2}] = \frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2}, \tag{111}$$

$$\begin{aligned}
u_{n+1} &= u_n + h\left[f_n + (f_n - f_{n-1})\frac{1}{2} + \frac{f_n - 2f_{n-1} + f_{n-2}}{2}\frac{5}{6}\right]\\
&= u_n + h\left[\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right]. \qquad (112)
\end{aligned}$$

Exercise 15. *Derive a three-step Adams-Moulton method with a variable timestep. Then compute its coefficients for the case where the timestep is constant.*

The interpolation error allows us to predict the order of the Adams methods. The interpolation error $f(t,y(t)) - p(t)$ is $O(h^{k+1})$ for a k-step Adams-Moulton method and $O(h^k)$ for a k-step Adams-Bashforth method. The integration increases the order of error by 1 predicting that the local truncation errors for k-step Adams-Moulton and Adams-Bashforth methods are, respectively, $O(h^{k+2})$ and $O(h^{k+1})$. Hence, k-step Adams-Moulton and Adams-Bashforth methods are of orders$p = k+1$ and $p = k$ respectively. Soon we will see that all Adams methods are stable.

7.5. **BDF methods: derivation.** As an example, we will derive a BDF method with variable timestep. Newton's interpolant through t_{n+1}, t_n, t_{n-1}, and t_{n-2} is

$$\begin{aligned} p(t) &= u_{n+1} + u[t_{n+1},t_n](t-t_{n+1}) + u[t_{n+1}t_n,t_{n-1}](t-t_{n+1})(t-t_n) \\ &\quad + u[t_{n+1},t_n,t_{n-1},t_{n-2}](t-t_{n+1})(t-t_n)(t-t_{n-1}), \end{aligned} \tag{113}$$

where

$$u[t_{n+1},t_n] = \frac{u_{n+1}-n_n}{h_n}, \quad u[t_{n+1},t_n,t_{n-1}] = \frac{\frac{u_{n+1}-u_n}{h_n} - \frac{u_n-u_{n-1}}{h_{n-1}}}{h_n+h_{n-1}}, \tag{114}$$

$$u[t_{n+1},t_n,t_{n-1},t_{n-2}] = \frac{\frac{\frac{u_{n+1}-u_n}{h_n} - \frac{u_n-u_{n-1}}{h_{n-1}}}{h_n+h_{n-1}} - \frac{\frac{u_n-u_{n-1}}{h_{n-1}} - \frac{u_{n-1}-u_{n-2}}{h_{n-2}}}{h_{n-1}+h_{n-2}}}{h_n+h_{n-1}+h_{n-2}}. \tag{115}$$

Then we differentiate the polynomial $p(t)$, evaluate its derivative at $t = t_{n+1}$, and equate it with f_{n+1}:

$$\begin{aligned} p'(t_{n+1}) &= u[t_{n+1},t_n] + u[t_{n+1}t_n,t_{n-1}]h_n \\ &\quad + u[t_{n+1},t_n,t_{n-1},t_{n-2}]h_n(h_n+h_{n-1}) = f_{n+1}, \end{aligned} \tag{116}$$

The interpolation error $y(t) - p(t)$ is $O(h^{k+1})$ for a k-step BDF. Hence a k-step BFD method is of order $p = k$. Soon we will see that BDF methods will lose stability as k increases.

7.6. **Theory for linear multistep methods with constant step and constant order.** Despite linear multistep methods being mostly implemented with variable timestep and variable order (VSVO), the theory for fixed timestep and fixed order is still useful because its conclusions accurately predict the behavior of VSVO.

7.6.1. *Notation.* A general linear multistep method with fixed timestep h is of the form

$$u_{n+1} + \alpha_0 u_n + \alpha_1 u_{n-1} + \ldots + \alpha_{k-1}u_{n-k+1} = h\left(\beta_{-1}f_{n+1} + \beta_0 f_n + \ldots + \beta_{k-1}f_{n-k+1}\right), \tag{117}$$

where $f_j = f(t_j,u_j)$ and $h_n = t_{n+1} - t_n$. We define the polynomials $\rho(z)$ and $\sigma(z)$ by

$$\rho(z) := \alpha_{-1}z^k + \alpha_0 z^{k-1} + \alpha_1 z^{k-2} + \ldots + \alpha_{k-1}, \tag{118}$$

$$\sigma(z) := \beta_{-1}z^k + \beta_0 z^{k-1} + \beta_1 z^{k-2} + \ldots + \beta_{k-1}. \tag{119}$$

Usually the normalization is chosen so that $\alpha_{-1} = 1$.

7.6.2. *Dahlquist barrier theorems.* These theorems show that stiff stability properties are mostly irrelevant for linear multistep methods.

Theorem 5. *A stable k-step method has order $p \le k+1$ if k is even, $p \le k+1$, if k is odd, and $p \le k$, if it is explicit.*

This theorem is consistent with our order calculation for the Adams and BDF methods.

Theorem 6. *An A-stable k-step method has order $p \le 2$, Among A-stable 2-step methods the most accurate is the trapezoidal rule.*

7.6.3. *The main theorem.*

Theorem 7. (1) *A k-step is consistent iff*

$$\rho(1) = 0, \quad \rho'(1) = \sigma(1).$$

(2) *A k-step method is consistent of order p iff*

$$\rho(e^h) - h\sigma(e^h) = O(h^{p+1}).$$

(3) *A k-step method is stable iff ρ satisfies the root condition: all roots z of $\rho(z) = 0$ have $|z| \le 1$ and those with $|z| = 1$ have multiplicity 1 (i.e., $\rho'(z) \neq 0$).*
(4) *The region of absolute stability of a k-step method is*

$$\mathsf{RAS} = \{h\lambda \in \mathbb{C} \mid \rho(z) - h\lambda\sigma(z) \text{ satisfies the root condition}\}.$$

Remark Note that this definition of the RAS is different from the one we used earlier: this one is the closure of the previously defined RAS.

The polynomial $\rho(z)$ has degree k, hence it has k roots. The total number of coefficients of a k-step linear multistep is $2k+1$ (taking into account the normalization). One of the roots is 1 required by the consistency criterion. The remaining $k-1$ roots must be kept inside the unit circle in the complex plane. Hence we need to use $k-1$ degrees of freedom to meet the root condition and only $k+2$ degrees of freedom can be used to satisfy consistency conditions of order p. This explains the first Dahlquist barrier theorem.

For Adams methods, we have $\rho(z) = z^{k-1}(z-1)$. Hence they are stable according to Theorem 7. On the other hand, BDF methods do not have to be stable.

For Forward Euler, we have $\rho(z) = z - 1$ and $\sigma(z) = 1$. Hence for its RAS we have: $z - 1 - h\lambda$ must satisfy the root condition. The only root of this polynomial is $z = 1 + h\lambda$. The root condition requires $|z| = |1 + h\lambda| \le 1$. Hence $h\lambda$ must belong to the unit circle centered at $-1 + 1i$.

7.6.4. *Proof of Theorem 7, parts (1) and (2).*

Proof. Let us calculate the local truncation error at step n. We will do Taylor expansion at t_{n-k+1} and set $\alpha_{-1} = 1$:

$$\begin{aligned}
\tau_{n+1} &= \alpha_{-1}y(t_{n+1}) + \alpha_0 y(t_n) + \ldots + \alpha_{k-1}y(t_{n-k+1}) \\
&\quad - h\left(\beta_{-1}y'(t_{n+1}) + \beta_0 y'(t_n) + \ldots + \beta_{k-1}y'(t_{n-k+1})\right) \\
&= \alpha_{-1}\left(y + hky + \frac{(hk)^2}{2}y'' + \ldots + \frac{(hk)^p}{p!}y^{(p)}\right) \\
&\quad + \alpha_0\left(y + h(k-1)y + \frac{(h(k-1))^2}{2}y'' + \ldots + \frac{(h(k-1))^p}{p!}y^{(p)}\right) + \ldots + \alpha_{k-1}y \\
&\quad - h\beta_{-1}\left(y' + hky'' + \frac{(hk)^2}{2}y''' + \ldots + \frac{(hk)^{p-1}}{(p-1)!}y^{(p)}\right) \\
&\quad - h\beta_0\left(y' + h(k-1)y'' + \frac{(h(k-1))^2}{2}y''' + \ldots + \frac{(h(k-1))^{p-1}}{(p-1)!}y^{(p)}\right) \\
&= \left(\sum_{j=-1}^{k-1}\alpha_j\right)y + \left(\sum_{j=-1}^{k-1}\left[(k-1-j)\alpha_j - \beta_j\right]\right)hy' + \ldots \\
&\quad + \left(\sum_{j=-1}^{k-1}\frac{(k-1-j)^p}{p!}\alpha_j - \frac{(k-1-j)^{p-1}}{(p-1)!}\beta_j\right) + O(h^{p+1}) \\
&=: C_0 y + C_1 y' + \ldots + C_p y^{(p)} + O(h^{p+1}).
\end{aligned}$$

The method is consistent iff the first two terms in the last expression to vanish, i.e.

$$\rho(1) \equiv C_0 \equiv \sum_{j=-1}^{k-1}\alpha_j = 0 \quad \text{and} \quad \rho'(1) - h\sigma(1) \equiv C_1 \equiv \sum_{j=-1}^{k-1}(k-j-1)\alpha_j - \beta_j = 0$$

The method is consistent of order p iff $C_0 = C_1 = \ldots = C_p = 0$. The coefficients C_l, $0 \le l \le p$, are independent of the function y, hence the method is consistent of order for any ODE iff it is consistent of order p for $y' = y$. Plugging the exact solution $y_n = e^{hn}y_0$ to $y' = y$, $y(0) = y_0$ into the method we obtain

$$\rho(e^h)y_{n-k+1} - h\sigma(e^h)y_{n-k+1} = \left[C_0 + C_1 + \ldots + C_p\right]y_{n-k+1} + O(h^{p+1})$$

The method is consistent of order p iff $C_0 = C_1 = \ldots = C_p = 0$, i.e., the right-hand side of the last equation is $O(h^{p+1})$. Hence its left-hans side must be also $O(h^{p+1})$, i.e.,

$$\rho(e^h) - h\sigma(e^h) = O(h^{p+1}).$$

□

To prove parts (3) and (4) we will need some background on the theory of linear difference equations.

7.6.5. *Linear difference equations.* We consider a difference equation of the form

$$u_{n+1} + \alpha_0 u_n + \ldots + \alpha_{k-1} u_{n-k+1} = 0. \tag{120}$$

The *characteristic equation* for (120) is

$$\rho(z) = z^k + \alpha_0 z^{k-1} + \ldots + \alpha_{k-1} = 0. \tag{121}$$

Theorem 8. *The difference equation (120) has k linearly independent solutions, and these solutions are of the form*

$$u_n = n^l r^n, \tag{122}$$

where r is a root of (121) of multiplicity m, and $l = 0, 1, \ldots, m-1$. The general solution to (120) is a linear combination of these solutions.

To prove this theorem, we need to

- check that (122) is a solution via direct substitution.
- Then use the fact that (121) has k roots counting multiplicities.
- Then form k column vectors where entries of vector j are the values of the jth linearly independent solution of the form (121) at $n = 0, 1, \ldots, k-1$ and show that these vectors are linearly independent.

The last step in this proof relies on the well-known fact that the generalized Vandermonde matrix is nonsingular – see e.g. https://www.jstor.org/stable/2690290?seq=7.

Exercise 16. *Check that (122) is a solution to (120).*

7.6.6. *Companion matrix.* It is convenient to rewrite a linear multistep method in a vector form. We introduce k-component vectors for $n \geq k-1$

$$U_n := \begin{bmatrix} u_{n-k+1} \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}, \quad U_{n+1} = \begin{bmatrix} u_{n-k+2} \\ \vdots \\ u_n \\ u_{n+1} \end{bmatrix}, \quad \text{and} \quad G_n = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_{-1} f_{n+1} + \ldots + \beta_{k-1} f_{n-k+1} \end{bmatrix} \tag{123}$$

and a $k \times k$ matrix A that is called the *companion matrix* for the polynomial $\rho(z)$

$$A := \begin{bmatrix} 0 & 1 & & \\ \vdots & \ddots & \ddots & \\ 0 & & & 1 \\ -\alpha_{k-1} & -\alpha_{k-2} & \ldots & -\alpha_0 \end{bmatrix} = \begin{bmatrix} 0_{(k-1)\times 1} & & I_{(k-1)\times(k-1)} \\ -\alpha_{k-1} & \ldots & -\alpha_0 \end{bmatrix}. \tag{124}$$

Then the linear multistep method

$$u_{n+1} + \alpha_0 u_n + \alpha_1 u_{n-1} + \ldots + \alpha_{k-1} u_{n-k+1} = h\left(\beta_{-1} f_{n+1} + \beta_0 f_n + \ldots + \beta_{k-1} f_{n-k+1}\right), \tag{125}$$

can be rewritten as

$$U_{n+1} = AU_n + hG_n. \tag{126}$$

Exercise 17. *Prove that the companion matrix for a polynomial ρ has exactly the same set of characteristic roots as ρ with the same multiplicities.*

Exercise 18. *Prove that all powers of the companion matrix A are uniformly bounded, i.e., $\|A^m\| \le K$ for all $m \in \mathbb{Z}$ is ρ satisfies the root condition.*

Hint: use the Jordan decomposition of A.

Exercise 19. *Check that the solution to (126) is*

$$U_n = A^{n-k+1}U_{k-1} + h\sum_{j=0}^{n-k} A^j G_{n-j-1}, \quad \text{where } U_{k-1} \text{ is the vector of initial data.} \tag{127}$$

7.6.7. *Proof of Theorem 7, part (3).*

Proof. In order to show that a linear multistep method is stable we need to prove that the numerical solution is a Lipschitz function with respect to perturbations of the initial conditions and the right-hand side. This means that we need to prove that if $\{u_n\}$ satisfies

$$u_{n+1} + \alpha_0 u_n + \ldots + \alpha_{k-1}u_{n-k+1} = h\left(\beta_{-1}f(t_{n+1}, u_{n+1}) + \ldots + \beta_{k-1}f(t_{n-k+1}, u_{n-k+1})\right), \tag{128}$$

and $\{v_n\}$ satisfies

$$v_{n+1}+\alpha_0 v_n+\ldots+\alpha_{k-1}u_{n-k+1} = h\left(\beta_{-1}f(t_{n+1}, v_{n+1}) + \ldots + \beta_{k-1}f(t_{n-k+1}, v_{n-k+1})\right)+\delta_{n+1}, \tag{129}$$

and

$$v_j = u_j + \delta_j, \quad 0 \le j \le k-1, \tag{130}$$

Then

$$\max_{0\le n\le T/h} \|u_n - v_n\| \le S \max_{0\le n\le T/h} \|\delta_n\|, \quad \text{where } S \text{ is a constant independent of } h. \tag{131}$$

We will conduct the proof for the case where y is a scalar function. The case where y is a vector function can be proven similarly. In the vector form, difference equations (128) and (129) are

$$U_{n+1} = AU_n + hG_n(u), \tag{132}$$

$$V_{n+1} = AV_n + hG_n(v) + D_n], \tag{133}$$

where $D_n = [0, \ldots, 0, \delta_{n+1}]^\top$. By Lipschitz continuity of f we have:

$$\|G_n(u) - G_n(v)\| \le BL(\|V_n - U_n\| + \|V_{n+1} - U_{n+1}\|). \tag{134}$$

Subtracting (132) from (133) and introducing the notation $V_n - U_n =: E_n$ and $\tilde{G}_n = G_n(v) + D_n G_n(u)$ we get

$$E_{n+1} = AE_n + h\left[\tilde{G}_n\right]. \tag{135}$$

By Lipschitz continuity of f we have:

$$\|G_n(v) + D_n - G_n(v)\| \le BL(\|E_n\| + \|E_{n+1}\|) + \delta, \quad \delta = \max_n |\delta_n| \tag{136}$$

The solution to (135) is

$$E_n = A^{n-k+1}E_{k-1} + h\sum_{j=0}^{n-k} A^j\left[\tilde{G}_{n-j-1}\right]. \tag{137}$$

Taking norms and using (136) we obtain:

$$\|E_n\| \le \|A^{n-k+1}\|\|E_{k-1}\| + h\sum_{j=0}^{n-k}\|A^j\|\left[BL(\|E_{n-j-1}\| + \|E_{n-j}\|) + D_{n-j-1}\right].$$

Since the polynomial ρ satisfies the root condition, the powers of A are uniformly bounded by some constant K: $\|A^m\| \le K$ for all $m \in \mathbb{N}$. Therefore,

$$\|E_n\| \le K\|E_{k-1}\| + h\sum_{j=0}^{n-k} K\left[BL(\|E_{n-j-1}\| + \|E_{n-j}\|) + D_{n-j-1}\right].$$

Now we move $\|E_n\|$ from the right-hand side to the left-hand side and get:

$$\|E_n\|(1+hKBL) \le K\|E_{k-1}\|+h\sum_{j=1}^{n-k} K\left[BL(\|E_{n-j-1}\| + \|E_{n-j}\|) + D_{n-j-1}\right]+BKL\|E_{n-1}\|+KD_{n-1}.$$

Assuming that h is small enough so that $hBKL \le 1/2$ and hence $[1 - hKBL]^{-1} \le 2$ we obtain

$$\begin{aligned}\|E_n\| &\le 2\left[K\|E_{k-1}\| + 2h\sum_{j=1}^{n-k} BKL\|E_{n-j-1}\| + Knh\delta\right]\\ &\le \left[2K\|E_{k-1}\| + 2KT\delta\right] + 4BKLh\sum_{j=k-1}^{n-1}\|E_j\|,\end{aligned} \tag{138}$$

where δ is the uniform bound on $\|D_n\|$. In the last equation, we use d the fact that $hn \le T$, the maximal time till which we integrate the solution. Lastly, we will use the *discrete Gronwall inequality*:

Lemma 1. *Let*

$$0 \le x_n \le a + b\sum_{j=0}^{n-1} x_j \quad \text{where} a > 0, \quad b > 0.$$

Then

$$x_n \le ae^{bn} \quad \forall n \ge 0.$$

Proof. Indeed, let

$$\hat{x}_n = a + b\sum_{j=0}^{n-1} x_j \ge x_n.$$

Then

$$\hat{x}_{n+1} - \hat{x}_n = bx_n \le b\hat{x}_n.$$

Hence,

$$\hat{x}_{n+1} = \hat{x}_n + bx_n \le (1+b)\hat{x}_n.$$

Therefore,

$$\hat{x}_n \le (1+b)\hat{x}_{n-1} \le \ldots \le (1+b)^n \hat{x}_0 \le a(1+b)^n \le ae^{bn}.$$

□

Applying Lemma 1 to (138) we obtain a bound for $\|E_n\|$:

$$\|E_n\| \le [2K\|E_{k-1}\| + 2KT\delta]\, e^{4BKLT}. \tag{139}$$

This shows that the numerical solution is a Lipschitz function with respect to perturbations of the right-hand side of the ODE (the term δ) and the initial data (the term $\|E_{k-1}\|$). □

7.6.8. *Proof of Theorem 7, part (4).*

Proof. Applying a linear multistep method to $y' = \lambda y$ we obtain:

$$u_{n+1} + \alpha_0 u_n + \ldots + \alpha_{k-1} u_{n-k+1} - h\lambda\left[\beta_{-1}u_{n+1} + \beta_0 u_n + \ldots + \beta_{k-1}u_{n-k+1}\right] = 0. \tag{140}$$

Equation (140) is a linear difference equation. To find its solutions, we must write the characteristic equation and find roots with their multiplicities. All solutions to (140) will be bounded provided that its characteristic polynomial

$$P(z) = \rho(z) - h\lambda\sigma(z) \tag{141}$$

satisfies the root condition. This completes the proof of part (4). □

7.7. **Plotting RAS.** The regions of absolute stability for linear multistep methods are plotted using the *boundary locus technique*. Since the roots of (141) have absolute value 1 on the boundary of RAS, the boundary of the RAS is a subset of the following set:

$$\partial\mathsf{RAS} \subset \left\{h\lambda \mid \rho(e^{i\theta}) - h\lambda\sigma(e^{i\theta}) = 0,\ 0 \le \theta < 2\pi\right\}. \tag{142}$$

Thus, to find the RAS, we first plot

$$\left\{h\lambda = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})} \mid 0 \le \theta < 2\pi\right\}.$$

This set is a closed curve that divides the complex plane into several components. Then it is sufficient to check one point in each component for being in the RAS or not by verifying the root condition. J. Strain's Matlab code `ras.m` plots the regions of absolute stability for Adams-Bashforth, Adams-Moulton, and BDF methods of orders from 2 up to 9. The results are displayed in Fig. 3, 4, and 5 respectively.

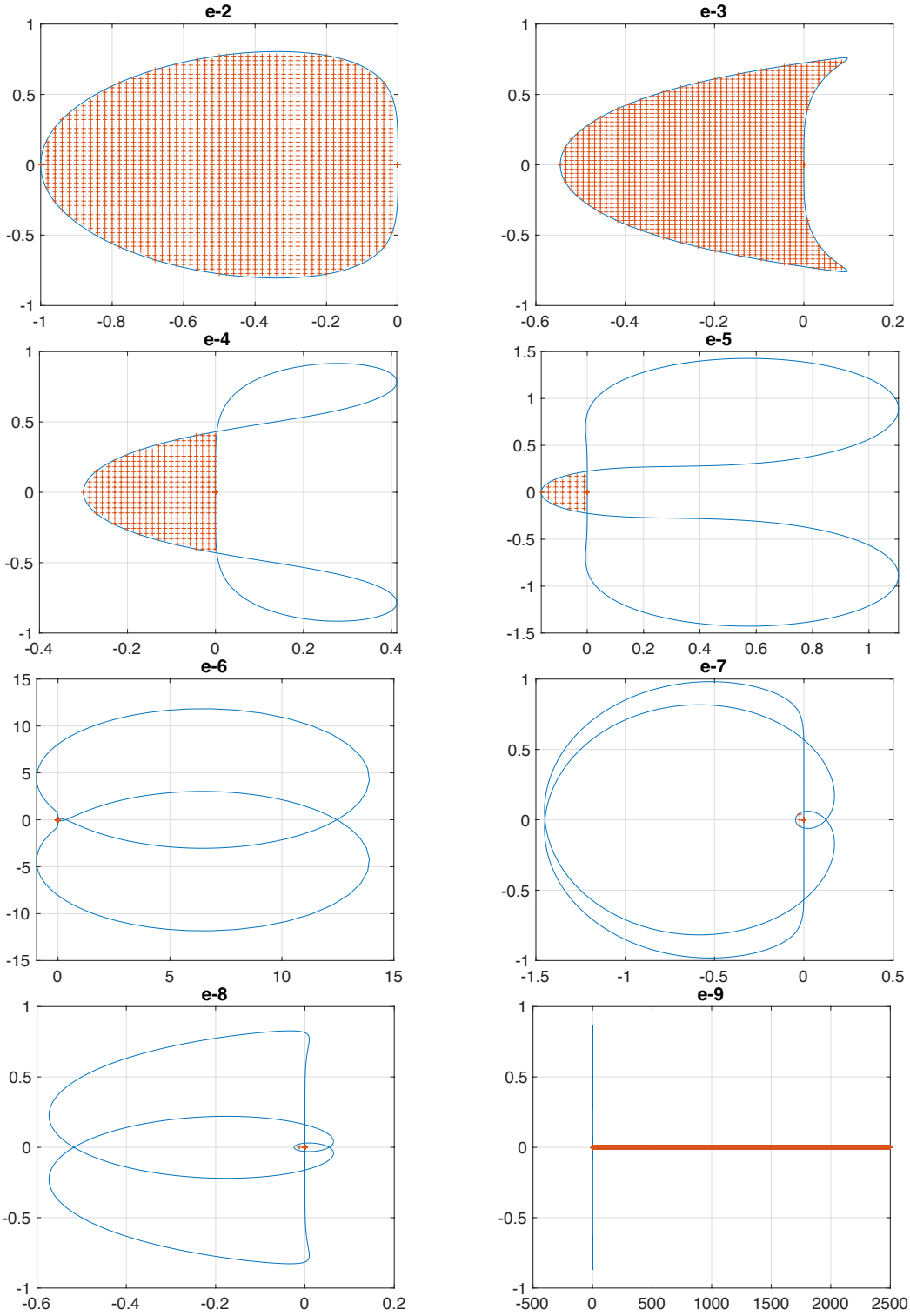


FIGURE 3. Regions of absolute stability for k-step Adams-Bashforth methods with $2 \leq k \leq 9$. The orders of these methods are equal to their number of steps, i.e. $p = k$. Note how the RASs shrink as the number of steps increases.

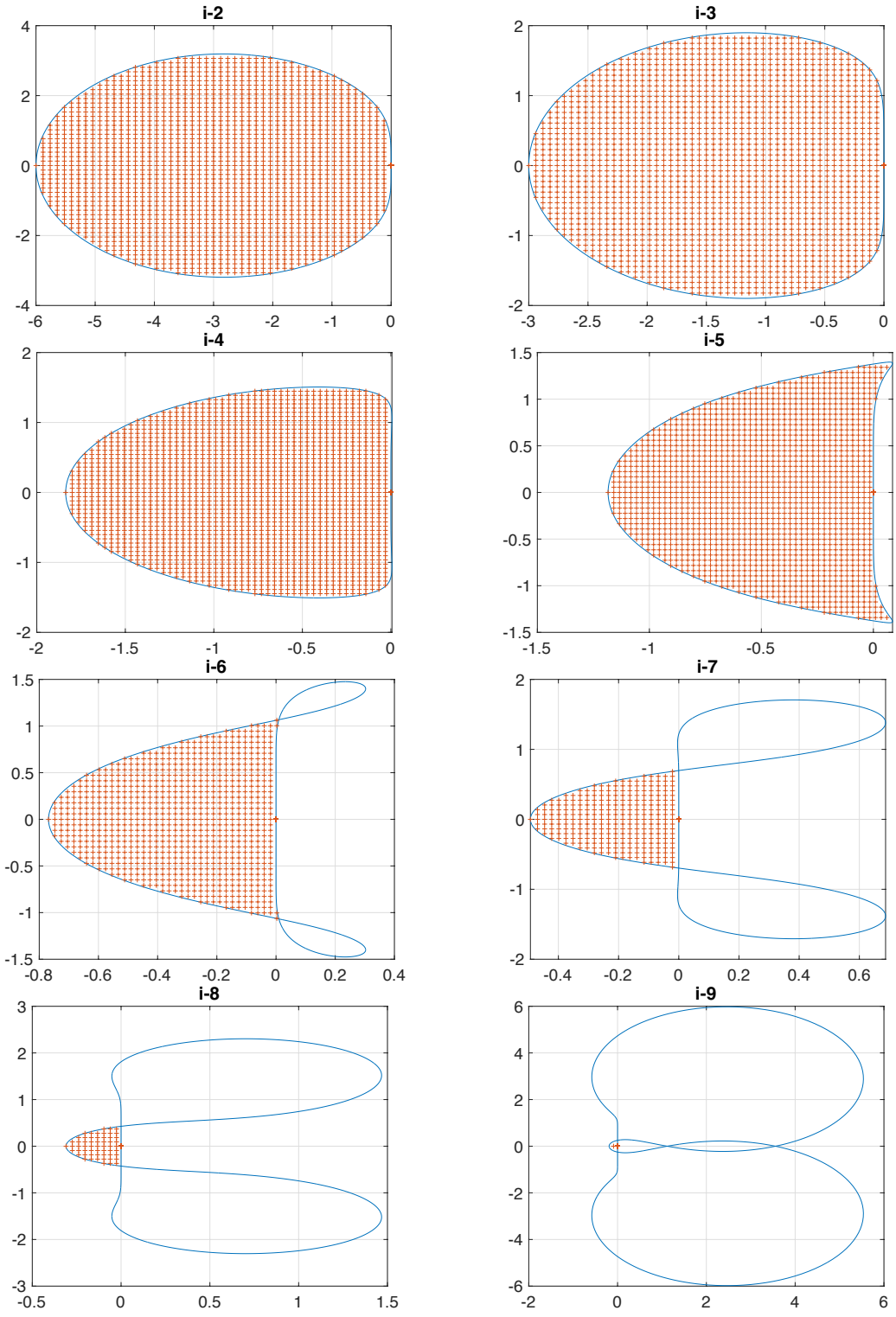


FIGURE 4. Regions of absolute stability for k-step Adams-Moulton methods with $2 \le k \le 9$. The orders of these methods are one plus their number of steps, i.e. $p = k + 1$. The RASs are bounded despite these methods being implicit but larger than the RASs of Adams-Bashforsth methods with the same number of steps. As RASs for Adams-Bashforth, the RASs shrink as the number of steps increases.

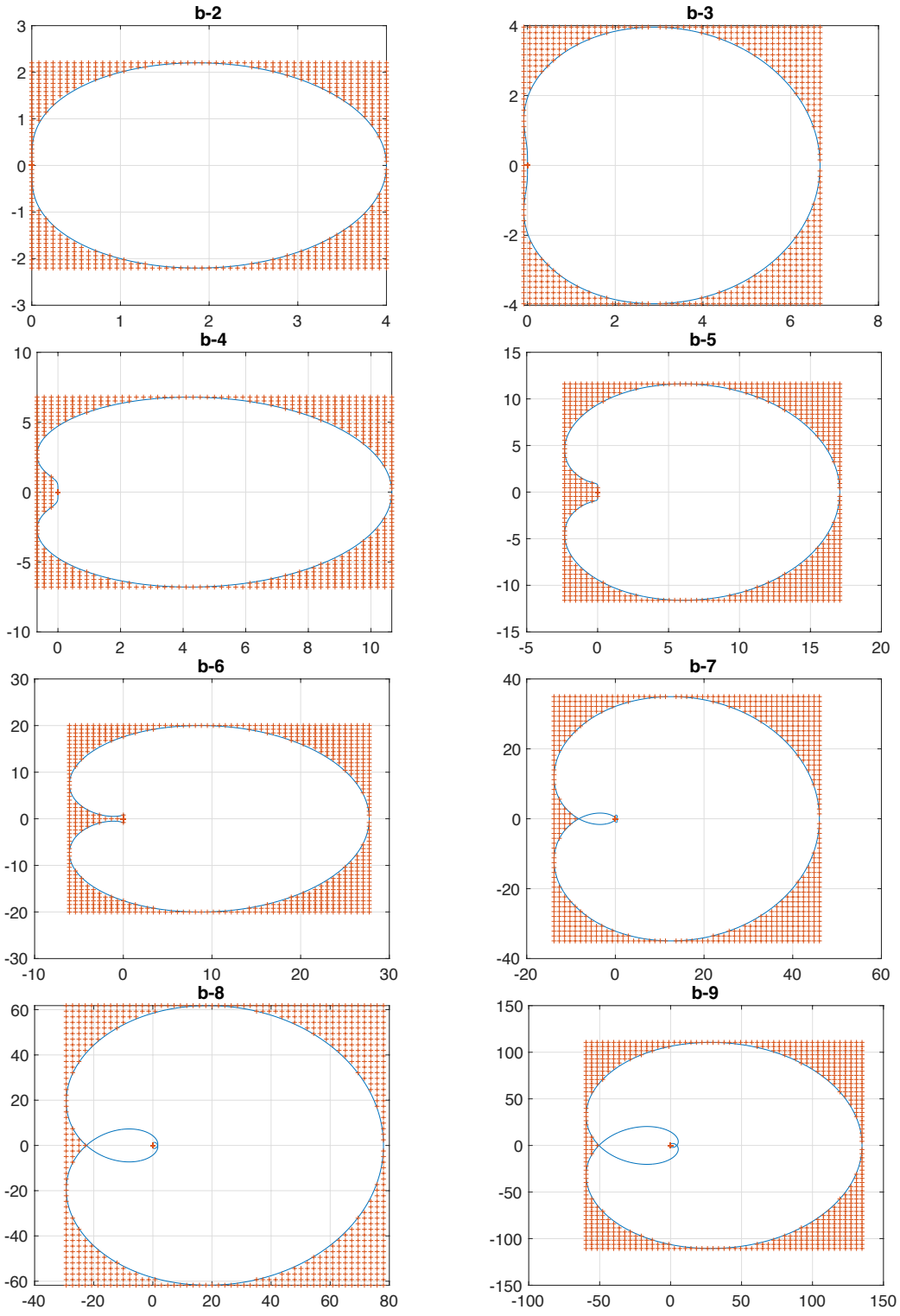


FIGURE 5. Regions of absolute stability for k-step BDF methods with $2 \leq k \leq 9$. The orders of these methods are their number of steps i.e. $p = k$. The RASs are unbounded. None of these methods is A-stable. BDF methods of orders 2,3,4,5, and 6. BDF methods become unstable for $k \geq 7$.