Chapter 4
Multi-Step Methods

In this chapter we will introduce a general form for multi-step methods. The forward Euler method and midpoint method are examples of multi-step methods. These methods are distinguished from multi-stage methods (see Chapter 10).

15 Self-Assessment

Before reading this chapter, you may wish to review...

- Taylor series expansions [18.01 Lecture 38: Video]
- solving linear systems [18.02SC Lecture 1.B Videos]

After reading this chapter you should be able to...

- determine the coefficients in the general form for any multi-step numerical method
- devise a numerical method meeting specifications on order of accuracy
- use MATLAB to solve linear systems using the backslash command

16 General form

Define the shorthand notation $f^n = f(v^n, t^n)$.

Definition 1 (Multi-step methods). The general form of an s-step multi-step method is

$$v^{n+1} + \sum_{i=1}^{s} \alpha_i v^{n+1-i} = \Delta t \sum_{j=0}^{s} \beta_j f^{n+1-j} \tag{29}$$

A multi-step method with $\beta_0 = 0$ is an *explicit* method since in this case we can rewrite the method in the form

$$v^{n+1} = g(v^n, v^{n-1}, \dots, t^n, t^{n-1}, \dots, \Delta t); \tag{30}$$

i.e., our approximation v^{n+1} depends only on information from the past. On the other hand, if $\beta_0 \neq 0$, the method is known as an *implicit* method (see Chapter 9). In that case, the method is written

$$v^{n+1} = g(v^{n+1}, v^n, v^{n-1}, \dots, t^{n+1}, t^n, t^{n-1}, \dots, \Delta t); \tag{31}$$

that is, the approximation v^{n+1} depends on the past data as well as the solution itself via the forcing function $f^{n+1} = f(v^{n+1}, t^{n+1})$. Although for scalar ODEs the distinction is unimportant in terms of computational cost, you will see in Chapter 9 that each time step of an implicit method will be more expensive than each time step of an explicit method.

That said, implicit methods do bring some important properties to the table; in particular, they have more attractive eigenvalue stability regions (see Chapter 7).

Example 1. The forward Euler method is a one-step method. We can determine the α_i and β_j for the method by comparing $v^{n+1} = v^n + \Delta t f^n$ to the general form in Definition 1. We find that

$$\alpha_i = \begin{cases} -1, & i = 1, \\ 0, & \text{otherwise}, \end{cases} \qquad \beta_j = \begin{cases} 1, & j = 1, \\ 0, & \text{otherwise}. \end{cases} \tag{32}$$

Exercise 1. What are the nonzero α_i and β_j for the midpoint method?

(a) $\alpha_1 = -1,\ \beta_1 = 2$
(b) $\alpha_2 = 1,\ \beta_1 = -2$
(c) $\alpha_1 = 1,\ \beta_1 = 1$
(d) $\alpha_2 = -1,\ \beta_1 = 2$

17 Create your own

Since we have a general form for an s-step multi-step method, we can create our own methods having certain desirable characteristics or meeting certain specifications. For example, we might wish to create the most accurate, explicit, two-step numerical method.

Example 2. In this example, we will derive the most accurate multi-step method of the following form:

$$v^{n+1} + \alpha_1 v^n + \alpha_2 v^{n-1} = \Delta t \left[\beta_1 f^n + \beta_2 f^{n-1}\right]$$

The local truncation error for this method is,

$$\tau = -\alpha_1 u^n - \alpha_2 u^{n-1} + \Delta t \left[\beta_1 f^n + \beta_2 f^{n-1}\right] - u^{n+1}$$

Substitution of $f^n = u_t^n$ and $f^{n-1} = u_t^{n-1}$ gives,

$$\tau = -\alpha_1 u^n - \alpha_2 u^{n-1} + \Delta t \left[\beta_1 u_t^n + \beta_2 u_t^{n-1}\right] - u^{n+1}$$

Then, Taylor series about $t = t^n$ are substituted for u^{n-1}, u_t^{n-1}, and u^{n+1} to give,

$$\begin{aligned} \tau = -\alpha_1 u^n - \alpha_2 & \left[u^n - \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n - \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{tttt}^n + O(\Delta t^5)\right] \\ & + \Delta t \beta_1 u_t^n + \Delta t \beta_2 \left[u_t^n - \Delta t u_{tt}^n + \frac{1}{2}\Delta t^2 u_{ttt}^n - \frac{1}{6}\Delta t^3 u_{tttt}^n + O(\Delta t^4)\right] \\ & - \left[u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 u_{tt}^n + \frac{1}{6}\Delta t^3 u_{ttt}^n + \frac{1}{24}\Delta t^4 u_{tttt}^n + O(\Delta t^5)\right] \end{aligned}$$

Next, collect the terms in powers of Δt, which gives the following coefficients:

$$\begin{array}{rcccccccc} u^n: & -\alpha_1 & - & \alpha_2 & & & & - & 1 \\ \Delta t u_t^n: & & & \alpha_2 & + \beta_1 & + & \beta_2 & - & 1 \\ \Delta t^2 u_{tt}^n: & & - & \frac{1}{2}\alpha_2 & & - & \beta_2 & - & \frac{1}{2} \\ \Delta t^3 u_{ttt}^n: & & & \frac{1}{6}\alpha_2 & & + & \frac{1}{2}\beta_2 & - & \frac{1}{6} \\ \Delta t^4 u_{tttt}^n: & & - & \frac{1}{24}\alpha_2 & & - & \frac{1}{6}\beta_2 & - & \frac{1}{24} \end{array}$$

With four unknowns, we can only force that the first four terms vanish. Later in the course we will be making frequent use of matrices and solving many linear systems. We remind you here that this becomes a linear system of four equations and four unknowns that can be written

$$\begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & -\frac{1}{2} & 0 & -1 \\ 0 & \frac{1}{6} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \frac{1}{2} \\ \frac{1}{6} \end{bmatrix} \tag{33}$$

To find the most accurate multi-step method of the given form, we solve for the values of α_1, α_2, β_1, and β_2.

Exercise 2. Fire up Matlab and solve for the unknowns in (33) using the backslash command in the form $\texttt{x} = \texttt{A}\backslash\texttt{b}$. Which of the following most closely matches your solution for $[\alpha_1, \alpha_2, \beta_1, \beta_2]^T$?

(a) $\begin{bmatrix} 4 \\ -5 \\ 2 \\ 4 \end{bmatrix}$

(b) $\begin{bmatrix} 1 \\ 0 \\ -1 \\ 5 \end{bmatrix}$

(c) $\begin{bmatrix} 4 \\ -5 \\ 4 \\ 2 \end{bmatrix}$

(d) $\begin{bmatrix} 4 \\ 2 \\ -4 \\ 1 \end{bmatrix}$

Exercise 3. What is the order of accuracy of the scheme you found in Exercise 2?

(a) $p = 1$
(b) $p = 2$
(c) $p = 3$
(d) $p = 4$

In Example 2 we defined the form of the numerical method and then found the coefficients leading to the highest order of accuracy. We can also specify order of accuracy, and then determine an appropriate scheme.

Thought Experiment Devise one scheme (other than midpoint method) that is second-order accurate.