Chapter 1

Two-point Boundary Value Problems for Ordinary Differential Equations

A two-point boundary value problem (BVP) is a second-order differential equation in a scalar or in a system form in which we do not have *two* initial conditions given. (Note here that we need *two* conditions because we are considering a second-order ODE. In general, a kth order ODE requires k *side* conditions in order to determine its solution.) Instead, in BVP, we have such side conditions given as the boundary conditions at the two boundary points of an interval over which we would like to seek for the solution.

There are a couple of methods available for solving BVPs including

- shooting methods,
- finite difference methods,
- finite element methods (or Galerkin methods),
- collocation methods,
- relaxation methods (Newton-Raphson-Kantorovich),
- eigenvalue problems

We are going to study only the first two approaches, shooting methods and finite difference methods, briefly in this chapter.

1. Shooting Method

Example: Consider Newton's second law $F = ma$ which can be specifically written as ODE:

$$F(t, x(t), x'(t)) = m\frac{d^2x}{dt^2} \tag{1.1}$$

where t is time chosen in an interval $[t_a, t_b]$, $x(t)$ is a position, dx/dt is a velocity, and d^2x/dt^2 is an acceleration. If we introduce two new variables y_1 and y_2:

$$y_1(t) = x(t), \;\; y_2(t) = y_1'(t) = \frac{dx}{dt}, \tag{1.2}$$

we can then convert this second-order ODE in to a first-order ODE system of two equations

$$\begin{bmatrix} y_1'(t) \\ y_2'(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ F/m \end{bmatrix}. \tag{1.3}$$

Note that we would solve Eq. 1.3 as an IVP if we are given the two side conditions as initial conditions:

$$\mathbf{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} x(t_a) \\ x'(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{1.4}$$

by which we know the initial conditions of the value $x(t_a)$ itself and the slope $x'(t_a)$. This IVP can uniquely determine the solution $x(t)$ over the entire interval $[t_a, t_b]$ using the various time-marching methods we learn Chapter 5, by integrating the numerical schemes from $t = t_a$ to $t = t_b$.

Instead, if we happen to know the two side conditions as boundary conditions rather than initial conditions, e.g.,

$$x(t_a) = \alpha, \;\; x(t_b) = \beta, \tag{1.5}$$

so that

$$\mathbf{y}(t_a) = \begin{bmatrix} x(t_a) \\ x'(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \text{unknown}_1 \end{bmatrix} \tag{1.6}$$

and

$$\mathbf{y}(t_b) = \begin{bmatrix} x(t_b) \\ x'(t_b) \end{bmatrix} = \begin{bmatrix} \beta \\ \text{unknown}_2 \end{bmatrix}, \tag{1.7}$$

we suddenly encounter a situation which lacks information on how to proceed time-marching using the techniques from IVPs because we do not know the slope information at $t = t_a$, which is essential for all time-marching schemes. Therefore, we need to develop different approaches to solve such BVPs. This is a main topic in this chapter. □

Remark: There are many important physical problems that have this form, including a few examples such as

- the bending of an elastic beam under a distributed transverse load,
- the temperature distribution over a rod whose end points are maintained at fixed temperature,

- the steady-state solution of parabolic PDEs which is equivalent to the corresponding elliptic PDEs.

□

As just introduced we can convert the two-point BVP for the second-order scalar ODE

$$u''(t) = f(t, u, u'), \;\; t_a < t < t_b, \tag{1.8}$$

with boundary contitions

$$u(t_a) = \alpha, \;\; u(t_b) = \beta, \tag{1.9}$$

to the equivalent first-order system of ODEs

$$\begin{bmatrix} y_1'(t) \\ \\ y_2'(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ \\ f(t, y_1, y_2) \end{bmatrix}, \;\; t_a < t < t_b, \tag{1.10}$$

where

$$y_1(t) = u(t), \;\; y_2(t) = y_1'(t) = u'(t). \tag{1.11}$$

The boundary conditions can be separated into a linear form,

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_1(t_b) \\ y_2(t_b) \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{1.12}$$

In general, we can express Eq. 1.10 as

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \;\; t_a < t < t_b, \tag{1.13}$$

where

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \\ y_2(t) \end{bmatrix}, \tag{1.14}$$

with boundary conditions

$$\mathbf{g}(\mathbf{y}(t_a), \mathbf{y}(t_b)) \equiv \begin{bmatrix} y_1(t_a) - \alpha \\ \\ y_1(t_b) - \beta \end{bmatrix} = \mathbf{0}, \;\; \mathbf{g} : \mathbb{R}^{2n} \to \mathbb{R}^n. \tag{1.15}$$

As noted in Eq. 1.6 and Eq. 1.7 the full initial and boundary conditions of $\mathbf{y}$ are

$$\mathbf{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} u(t_a) \\ \\ u'(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \\ \text{unknown}_1 \end{bmatrix}, \tag{1.16}$$

and

$$\mathbf{y}(t_b) = \begin{bmatrix} y_1(t_b) \\ \\ y_2(t_b) \end{bmatrix} = \begin{bmatrix} u(t_b) \\ \\ u'(t_b) \end{bmatrix} = \begin{bmatrix} \beta \\ \\ \text{unknown}_2 \end{bmatrix}. \tag{1.17}$$

Now let us consider the following approach to solve the BVP ODE system in Eq. 1.13 along with Eq. 1.16 and Eq. 1.17.

1. If we assume we somehow know unknown_1 in Eq. 1.16, for instance, we make an initial guess for it and set

$$\text{unknown}_1 = y_2^{(0)}(t_a). \tag{1.18}$$

2. With this guess, we can now successfully proceed to solve the IVP ODE system in Eq. 1.13 with our guessed initial condition

$$\mathbf{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} u(t_a) \\ \\ u'(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \\ y_2^{(0)}(t_a) \end{bmatrix}. \tag{1.19}$$

   This is nothing but making an initial guess about the slope $u'(t_a)$ at $t = t_a$, with which we now have full information to embark on a time-marching of $\mathbf{y}$ successively over $[t_a, t_b]$.

3. Check how well the time-marching solution $y_1^{(0)}\Big(t_b; y_2^{(0)}(t_a)\Big)$ at $t = t_b$, which has been just produced by using the initial guess $y_2^{(0)}(t_a)$, compares with the true boundary value $y_1(t_b) = u(t_b) = \beta$.

4. If the resulting value $y_1^{(0)}(t_b)$ is close to $y_1(t_b) = u(t_b) = \beta$, the search is successful and exit. Otherwise, the process is repeated until the search is successful with a new slope guess

$$y_2^{(k)}(t_a), \quad k = 1, 2, \cdots \tag{1.20}$$

   which will produce a new IVP solution

$$y_1^{(k)}\Big(t_b; y_2^{(k)}(t_a)\Big) \to \beta, \quad k = 1, 2, \cdots \tag{1.21}$$

Remark: The meaning of Eq. 1.21 is that the kth IVP solution $y_1^{(k)}$ at $t = t_b$ based on the initial guess of the slope $y_2^{(k)}(t_a)$ at $t = t_a$ converges to β.

Remark: If we ever can solve the IVP *back in time* over $[t_a, t_b]$, integrating from t_b to t_a, we would make an initial guess on unknown_2 and solve the IVP instead. But we don't want to do this reverse solve in normal situations and don't consider that way. □

The above procedure can be thought as a root finding problem of a function h given by

$$h \equiv h\Big(y_1^{(k)}(t_b); y_2^{(k)}(t_a)\Big) \equiv y_1^{(k)}\Big(t_b; y_2^{(k)}(t_a)\Big) - \beta = 0 \tag{1.22}$$

For this we can use Newton's root finding method which can be written as the following:

Algorithm: Newton's method for finding root:

$y_2^{(0)}(t_a) =$ `initial guess`
`estimator = largeNumber (e.g.,` 10^{10}`)`

`while estimator` $> \epsilon$

$y_2^{(k+1)}(t_a) = y_2^{(k)}(t_a) - h^{(k)}/(h^{(k)})'$

`estimator =` $-h^{(k)}/(h^{(k)})'$

`# [if estimator becomes close to zero`
`# it implies it has found the root and converged]`

`# Please see Remark below for` $h^{(k)}$

`endwhile`

Remark: In the above algorithm, we used a notation

$$h^{(k)} \equiv h\Big(y_1^{(k)}(t_b); y_2^{(k)}(t_a)\Big), \tag{1.23}$$

and

$$(h^{(k)})' = h'\Big(y_1^{(k)}(t_b); y_2^{(k)}(t_a)\Big) = \Big[y_1^{(k)}\Big(t_b; y_2^{(k)}(t_a)\Big) - \beta\Big]' = y_2^{(k)}(t_b). \tag{1.24}$$

Note that we just used a relationship $y_1'(t) = y_2(t)$ and $y_1^{(k)}(t) \approx y_1(t)$. $\square$

Example: Consider the BVP given by

$$u''(t) = 6t, \;\; 0 < t < 1, \tag{1.25}$$

with boundary conditions

$$u(0) = 0, \;\; u(1) = 1. \tag{1.26}$$

We now convert this into a system of first-order ODE equations

$$\begin{bmatrix} y_1'(t) \\ \\ y_2'(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ \\ 6t \end{bmatrix}, \tag{1.27}$$

where $y_1 = u$ and $y_2 = y_1' = u'$. Let's make an initial guess on the slope of u at $t = t_a$, so that we solve the IVP using the first guess $y_2^{(0)}(t_a)$:

$$\mathbf{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} 0 \\ \\ y_2^{(0)}(t_a) \end{bmatrix} \tag{1.28}$$

We also take the forward Euler method to integrate the IVP over the temporal domain $[0, 1]$. The function h for root finding becomes

$$h\Big(y_1^{(k)}(1); y_2^{(k)}(0)\Big) \equiv y_1^{(k)}\Big(1; y_2^{(k)}(0)\Big) - 1 = 0. \tag{1.29}$$

For each guess for $y_2^{(k)}(0)$, we will integrate the ODE using the forward Euler method, for instance, the first integration with the first initial guess $y_2^{(0)}(0)$ becomes:

- $k = 0$:
  For $n = 1$, or $t^1 = 1 \cdot \Delta t$ (recall $t^n = n\Delta t$):

$$\begin{aligned} y_1^1 &= y_1^0 + \Delta t(y_2^0) && (1.30)\\ y_2^1 &= y_2^0 + \Delta t(6t^0) = y_2^{(0)} + \Delta t(6 \cdot 0) = y_2^{(0)} && (1.31) \end{aligned}$$

  For $n = 2$, or $t^2 = 2\Delta t$:

$$\begin{aligned} y_1^2 &= y_1^1 + \Delta t(y_2^1) && (1.32)\\ y_2^2 &= y_2^1 + \Delta t(6t^1) = y_2^1 + 6\Delta t^2 && (1.33) \end{aligned}$$

  Continue the time-marching until $n = N$ such that $t^N = t_b = 1$ is reached:
  For $n = N$, or $t^N = N\Delta t = t_b = 1$:

$$\begin{aligned} y_1^N &= y_1^{N-1} + \Delta t(y_2^{N-1}) && (1.34)\\ y_2^N &= y_2^{N-1} + \Delta t(6t^{N-1}) = y_2^{N-1} + 6(N-1)\Delta t^2 && (1.35) \end{aligned}$$

- These final values at $t = t_b = 1$ we just obtain are

$$\begin{aligned} y_1^{(0)}(1) &= y_1^N && (1.36)\\ y_2^{(0)}(1) &= y_2^N && (1.37) \end{aligned}$$

- Perform the root finding for the next guess $y_2^{(1)}(0)$:

$$y_2^{(1)}(0) = y_2^{(0)}(0) - \frac{h^{(0)}}{(h^{(0)})'} = y_2^{(0)}(0) - \frac{y_1^{(0)}(1) - 1}{y_2^{(0)}(1)} \tag{1.38}$$

- Repeat this process with the new initial guess on the slope $y_2^{(k)}(0)$ until the estimator $h^{(k)}/(h^{(k)})'$ gets closer to zero:

$$\frac{h^{(k)}}{(h^{(k)})'} = \frac{y_1^{(k)}(1) - 1}{y_2^{(k)}(1)} \to 0. \tag{1.39}$$

□

Remark: The pros and cons in the shooting method approach include:

- the shooting method is conceptually simple and is easy to implement using existing software for IVPs and for root finding methods,
- the shooting method inherits the stability of the associated IVP and might become unstable even when the BVP is stable and thus results in extreme difficulties in convergence,
- for some initial guesses for the IVP, the solution of the IVP may not exist over the entire interval (or domain) of integration in that the solution may become unbounded even before reaching the right-hand endpoint of the BVP.

□

2. Finite Difference Method

In solving a BVP the shooting method approximately satisfies the ODE from the outset (by using an IVP solver) and iterates until the boundary conditions are met. There are potential issues as discussed in the shooting method approach. An alternative is to satisfy the boundary conditions from the outset and iterate until the ODE is approximately satisfied. This approach is taken in finite difference methods, which convert a BVP directly into a system of algebraic equations rather than a sequence of IVPs as in the shooting method.

In a finite difference method, a set of mesh of points is introduced within the interval of integration and then any derivatives appearing in the ODE or boundary conditions are replaced by finite difference approximations at the mesh points.

For a scalar two-point BVP

$$u''(t) = f(t, u, u'), \quad t_a < t < t_b, \tag{1.40}$$

with boundary conditions

$$u(t_a) = \alpha, \quad u(t_b) = \beta, \tag{1.41}$$

we introduce mesh points

$$t^n = t_a + n\Delta t, \quad n = 0, 1, \cdots, N+1, \tag{1.42}$$

where $\Delta t = (t_b - t_a)/(N+1)$ (note that we obtain $t^{N+1} = t_b$), and we seek for approximate solution values

$$U^n \approx u(t^n), \quad n = 1, \cdots, N. \tag{1.43}$$

Next we use finite difference approximations to replace the first and second derivatives

$$u'(t^n) \approx \frac{U^{n+1} - U^{n-1}}{2\Delta t} \tag{1.44}$$

$$u''(t^n) \approx \frac{U^{n+1} - 2U^n + U^{n-1}}{\Delta t^2}. \tag{1.45}$$

Note that these finite difference approximations are of second-order accurate having the local truncation errors of order $\mathcal{O}(\Delta t^2)$.

As a result the difference relation becomes a system of algebraic equations

$$\frac{U^{n+1} - 2U^n + U^{n-1}}{\Delta t^2} = f\Big(t^n, U^n, \frac{U^{n+1} - U^{n-1}}{2\Delta t}\Big), \quad n = 1, \cdots, N. \tag{1.46}$$

Example: We illustrate the finite difference method on the two-point BVP from the previous example

$$u''(t) = 6t, \quad 0 < t < 1, \tag{1.47}$$

with boundary conditions

$$u(0) = 0, \quad u(1) = 1. \tag{1.48}$$

We find that the difference equations over $[0, 1]$ are

$$\frac{U^{n+1} - 2U^n + U^{n-1}}{\Delta t^2} = 6t^n, \quad n = 1, \cdots, N, \tag{1.49}$$

with the boundary conditions

$$U^0 = U(0) = 0, \quad U^{N+1} = U(t^{N+1}) = U(1) = 1. \tag{1.50}$$

The system of equations can be then written as a linear system $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \frac{1}{\Delta t^2} \begin{bmatrix} -2 & 1 & & & & & & & \\ 1 & -2 & 1 & & & & & & \\ & 1 & -2 & 1 & & & & & \\ & & & \cdot & \cdot & \cdot & & & \\ & & & & \cdot & \cdot & \cdot & & \\ & & & & & \cdot & \cdot & \cdot & \\ & & & & & & 1 & -2 & 1 \\ & & & & & & & 1 & -2 \end{bmatrix}, \tag{1.51}$$

and

$$\mathbf{b} = \begin{bmatrix} f(t^1) - \alpha/\Delta t^2 \\ f(t^2) \\ \vdots \\ f(t^{N-1}) \\ f(t^N) - \beta/\Delta t^2 \end{bmatrix} = \begin{bmatrix} 6t^1 \\ 6t^2 \\ \vdots \\ 6t^{N-1} \\ 6t^N - 1/\Delta t^2 \end{bmatrix}, \tag{1.52}$$

with the solution vector

$$\mathbf{x} = \begin{bmatrix} U^1 \\ U^2 \\ \vdots \\ U^{N-1} \\ U^N \end{bmatrix}, \tag{1.53}$$

This tridiagonal linear system is nonsingular and can be easily solved for $\mathbf{x}$ using the methods we learned in AMS 213A such as

- Gaussian elimination (or LU factorization),
- Gauss-Jordan elimination,
- Cholesky factorization for symmetric positive definite for $\mathbf{A}$,
- Crout's method for LU decomposition.