two-point Boundary Value problems for ODEs

Def. A two-pt. BVP is a 2nd order differential equation in a scalar or a system form with not enough ICs

→ Note that the IVP of a 2nd order diffl eqn is well-defined with 2 ICs:

$$\begin{cases} u'' + p(t)u' + q(t)u = g(t), \\ u(t^0) = u^0, \quad u'(t^0) = u'^0 \end{cases} \quad \cdots (1)$$

($u(t^0) = u^0$: value; $u'(t^0) = u'^0$: slope)

with

$p(t), q(t) \in C(I), \quad t^0 \in I = [t^0, t^N]$

$\Rightarrow \exists!$ soln to (1).

→ But in BVP, we instead are given

$$u(t^0) = \alpha, \quad u(t^N) = \beta,$$

→ A couple of methods for BVP:

✱
1. shooting method
2. finite difference method
3. finite element method (Galerkin)
4. collocation method
5. relaxation method (Newton-Raphson - Kantorovich)
6. eigenvalue problems

III Shooting method

Ex. Newton's second law: $F = ma$

Written in ODE; we get

$$F(t, x(t), x'(t)) = m\frac{d^2x}{dt^2} \quad \cdots ② \qquad t \in [t_a, t_b]$$

F: time, position, velocity; m: acceleration, mass

→ ② can be written as a system of 1st order ODEs:

$$\begin{cases} y_1(t) = x(t) \\ y_2(t) = y_1'(t) = \dfrac{dx}{dt} \quad (\Rightarrow y_2' = \dfrac{d^2x}{dt^2}) \end{cases}$$

$$\Rightarrow \begin{bmatrix} y_1'(t) \\ y_2'(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ F/m \end{bmatrix} \quad \cdots ③$$

⇒ We would happily solve ② as an IVP **if** we are given two ICs for y_1 & y_2:

$$\underline{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} x(t_a) \\ x'(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad \cdots ④$$

⇒ We can solve ③ along ④ using those numerical methods we've learned so far.

→ Instead, if we don't have (4) but only know the two boundary conditions at $t = t_a$ & $t = t_b$:

$$\begin{bmatrix} x(t_a) \\ x(t_b) \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

→ $$\underset{\sim}{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} x(t_a) \\ x'(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \text{unknown}_1 \end{bmatrix}$$

$$\underset{\sim}{y}(t_b) = \begin{bmatrix} y_1(t_b) \\ y_2(t_b) \end{bmatrix} = \begin{bmatrix} x(t_b) \\ x'(t_b) \end{bmatrix} = \begin{bmatrix} \beta \\ \text{unknown}_2 \end{bmatrix}$$

→ We lack information on how to proceed time-marching of IVPs since there is no information on $x'(t_a)$, assuming we try to solve time-marching from t_a to t_b, not the other way around, from t_b to t_a.

→ This is the topic of BVPs. ▨

Rmk The BVPs can occur in many applications such as

(i) an elastic beam bending under a distributed transverse load,

(ii) temp. distribution over a rod with fixed temperatures at the end points

(iii) parabolic PDE $\xrightarrow[t\to\infty]{}$ elliptic PDE

$$\frac{\partial u}{\partial t} = k\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \qquad \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y} = 0$$

In general, given a 2nd order ODE

$$u'' = f(t, u, u'), \quad t_a < t < t_b \quad \text{--- (5)}$$

with BCs:

$$\begin{cases} u(t_a) = \alpha \\ u(t_b) = \beta, \end{cases} \quad \text{--- (6)}$$

We convert (5) into, using a new set of variables

$$\begin{cases} y_1(t) = u(t) \\ y_2(t) = y_1'(t) = u'(t), \quad (y_2' = u'') \end{cases}$$

$$\begin{bmatrix} y_1'(t) \\ y_2'(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ f(t, y_1, y_2) \end{bmatrix} \quad \text{--- (7)}$$

$t_a < t < t_b$

Where the BCs (6) can be written as

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_1(t_b) \\ y_2(t_b) \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad \text{--- (8)}$$

→ If we let $\underset{\sim}{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}$, then

(7) $\Leftrightarrow$ $\underset{\sim}{y}'(t) = f(t, \underset{\sim}{y}(t))$, with

$$\underset{\sim}{g}(\underset{\sim}{y}(t_a), \underset{\sim}{y}(t_b)) \equiv \begin{bmatrix} y_1(t_a) - \alpha \\ y_2(t_b) - \beta \end{bmatrix} = \underset{\sim}{0},$$

$$\underset{\sim}{g} : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^n.$$

→ Note that the full set of ICs & BCs:

$$\underset{\sim}{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ \text{unknown}_1 \end{bmatrix},$$

$$\underset{\sim}{y}(t_b) = \begin{bmatrix} y_1(t_b) \\ y_2(t_b) \end{bmatrix} = \begin{bmatrix} \beta \\ \text{unknown}_2 \end{bmatrix}.$$

→ The strategy we take in the shooting method is the following:

Step 1 take an initial guess for unknown$_1$;

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

Step 2 With $y_2^{(0)}(t_a)$, we solve the IVP using

$$\underset{\sim}{y}(t_a) = \begin{bmatrix} y_1(t_a) \\ y_2(t_a) \end{bmatrix} = \begin{bmatrix} \alpha \\ y_2^{(0)}(t_a) \end{bmatrix}$$

$y_2(t_a) = u'(t_a)$

$y_2^{(0)}(t_a)$ → initial guess for $u'(t_a)$.

Step 3 Check how well the true-matching soln at $t = t_b$, denoted as

$$y_1^{(0)}(t_b; y_2^{(0)}(t_a)),$$ → $y_1^{(0)}$ at t_b as a fn of $y_2^{(0)}(t_a)$

compares with β, the true bdry value at $t = t_b$, $y_1(t_b) = u(t_b) = \beta$.

Step 4 If $y_1^{(0)}(t_b; y_2^{(0)}(t_a)) \simeq \beta$, then the search is successful and exit.

Otherwise, the seach continues with a new initial guess of the slope $u'(t_a)$, $y_2^{(k)}(t_a)$, $k = 1, 2, \ldots$ ... (9)

$$\Rightarrow \boxed{y_1^{(k)}(t_b; y^{(k)}(t_a)) \longrightarrow \beta, \quad k = 1, 2, \ldots}$$

Rmk The procedure @ is to be solved using a root finding problem of a fcn h defined by

$$h \equiv h\left(y_2^{(k)}(t_a)\right) \equiv y_1^{(k)}(t_b; y_2^{(k)}(t_a)) - \beta = 0$$

- $y_2^{(k)}(t_a)$: treat this as a sequence variable, $k = 1, 2, \ldots$
- $y_1^{(k)}(t_b; y_2^{(k)}(t_a))$: a shooting value of y_1 at $t = t_b$ using the initial guess $y_2^{(k)}(t_a)$, the slope of u at t_a (i.e., $u'(t_a)$).
- β: true value y_1 at $t = t_b$.

Rmk Newton's root finding Algorithm

→ Consider $f(x+h) \approx f(x) + f'(x)h$, and we are interested in finding the root of $f(x+h)$, i.e, we want to find $x^* = x+h$ s.t $f(x^*) = 0$.

→ If we approximate $f(x+h)$ using $f(x) + f'(x)h$, then

$$f(x^*) = 0 \iff h = -\frac{f(x)}{f'(x)}, \quad \text{assuming } f'(x) \neq 0,$$

⇒ **Algorithm**

x_0 = initial guess

estimator = large number (10^{10})

while estimator $> \varepsilon$, $k = 0, 1, 2, \ldots$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

estimator $= -f(x_k)/f'(x_k)$.

endwhile

Rmk If we use Newton's root finder for the function

$$h \equiv h\left(y_2^{(k)}(t_a)\right) = 0, \text{ then}$$

Algorithm

Initial guess $y_2^{(0)}(t_a)$

estimator = large

while estimator $> \varepsilon$, $k = 0, 1, \ldots$

$$y_2^{(k+1)}(t_a) = y_2^{(k)}(t_a) - \frac{h\left(y_2^{(k)}(t_a)\right)}{h'\left(y_2^{(k)}(t_a)\right)}$$

$$\text{estimator} = -\frac{h\left(y_2^{(k)}(t_a)\right)}{h'\left(y_2^{(k)}(t_a)\right)}$$

endwhile

→ Note that if estimator $\to 0$, then $y_2^{(k)}(t_a)$ converges to the true slope $u'(t_a) = y_2(t_a)$, which is unknown,

Rmk Note that, analytically, we set

$$h'\left(y_2^{(k)}(t_a)\right) = \left[y_1^{(k)}(t_b; y_2^{(k)}(t_a)) - \beta\right]'$$

$$= \left[y_1^{(k)}(t_b; y_2^{(k)}(t_a))\right]' - \underbrace{(\beta)'}_{=0}$$

$$= y_2^{(k)}(t_b).$$

($y_1' = y_2 \approx y_2^{(k)}$)

Rmk One can use ~~Secant method~~ to replace the analytical derivative

$$h'\left(y_2^{(k)}(t_a)\right)$$

with a numerical approximation

$$h'\left(y_2^{(k)}(t_a)\right) \approx \frac{h\left(y_2^{(k)}(t_a)\right) - h\left(y_2^{(k-1)}(t_a)\right)}{y_2^{(k)}(t_a) - y_2^{(k-1)}(t_a)}.$$

Ex. $\begin{cases} u'' = 6t, & 0 < t < 1 \\ u(0) = 0, \quad u(1) = 1 \end{cases}$

$\Rightarrow \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ 6t \end{bmatrix}$ where $\begin{cases} y_1 = u \\ y_2 = y_1' = u' \end{cases}$

$\Rightarrow$ we know two BCs only;

$$\underset{\sim}{y}(0) = \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ ? \end{bmatrix} \equiv \text{let} \begin{bmatrix} 0 \\ y_2^{(0)}(0) \end{bmatrix},$$

$y_2^{(0)}(0)$ ← initial guess of the slope

$$\underset{\sim}{y}(1) = \begin{bmatrix} y_1(1) \\ y_2(1) \end{bmatrix} = \begin{bmatrix} 1 \\ ? \end{bmatrix}$$

1 ← target

$\Rightarrow$ let $h(y_2^{(k)}(0)) = y_1(1; y_2^{(k)}(0)) - 1$

$\Rightarrow$ If using <u>secant method</u> to evaluate h', and if using <u>FE</u> for each IVP given by $y_2^{(k)}(0)$; we get

Algorithm

$y_2^{(0)}(0)$ = initial guess

for $k = 0, 1, 2, \cdots$

$y_1^n = y_1(t^n)$

$$\begin{cases} y_1^{n+1} = y_1^n + \Delta t\, y_1'(t^n) = y_1^n + \Delta t\, y_2^n \\ y_2^{n+1} = y_2^n + \Delta t\, y_2'(t^n) = y_2^n + \Delta t\,(6t^n) \end{cases}$$

More precisely, if we write the first step calculation $n = 1$;

use the initial guess

$$\begin{cases} y_1^1 = y_1^0 + \Delta t\, y_2^0 = y_1^0 + \Delta t\, y_2^{(0)}(0) \\ y_2^1 = y_2^0 + \Delta t\,(6t^0) = y_2^{(0)}(0) + \Delta t(6\cdot 0) \\ \qquad\qquad = y_2^{(0)}(0). \end{cases}$$

$t^0 = 0\cdot\Delta t = 0$

Note $t^n = n\Delta t$

The second step $n = 2$;

$$\begin{cases} y_1^2 = y_1^1 + \Delta t\, y_2^1 \\ y_2^2 = y_2^1 + \Delta t(6t^1) \end{cases}$$

$$\vdots$$

$$\begin{cases} y_1^N = y_1^{N-1} + \Delta t\, y_2^{N-1} \\ y_2^N = y_2^{N-1} + \Delta t\,(6t^{N-1}) \end{cases}$$, where $t^N = 1$.

If $t^N = 1$, stop.

Once reached to $t^N = 1$, then compute

$$y_2^{(k+1)}(0) = y_2^{(k)}(0) - \frac{h\left(y_2^{(k)}(0)\right)}{h'\left(y_2^{(k)}(0)\right)}$$

$$= y_2^{(k)}(0) - h\left(y_2^{(k)}(0)\right) \frac{y_2^{(k)}(0) - y_2^{(k-1)}(0)}{h\left(y_2^{(k)}(0)\right) - h\left(y_2^{(k-1)}(0)\right)}$$

new estimator using Secant method

If the new estimator

$$h\left(y_2^{(k)}(0)\right) \frac{y_2^{(k)}(0) - y_2^{(k-1)}(0)}{h\left(y_2^{(k)}(0)\right) - h\left(y_2^{(k-1)}(0)\right)} \approx 0$$

then stop.

else $k = k+1$, continue

Note: If we used the exact method for $h'(y_2^{(k)}(0))$, then as seen before,

$$h'(y_2^{(k)}(0)) = \left[y_1(1; y_2^{(k)}(0)) - 1\right]'$$

$$= y_1'(1; y_2^{(k)}(0))$$

$$y_1' = y_2 \approx y_2^{(k)} \quad = y_2^{(k)}(1),$$

then the estimator looks like

$$\frac{h(y_2^{(k)}(0))}{h'(y_2^{(k)}(0))} = \frac{y_1(1; y_2^{(k)}(0)) - 1}{y_2^{(k)}(1)} \quad \text{--- (10)}$$

and we stop when (10) ≈ 0.

Rmk Pros & Cons of the shooting method:

(1) conceptually simple, easy to implement using existing software for IVPs, plus for root finding methods

(2) the stability of the shooting method is inherited by that of the associated IVP,

→ hence the shooting method may become unstable (if the IVP is unstable) even though the BVP is stable

→ thus it can be extremely difficult for convergence

(3) for some initial guesses for the IVPs, the soln of the IVP may not exist over the entire domain of integration

→ this can occur when the soln become unbounded even before reaching at $t = t^{max}$ (the end point of the BVP).

[2] Finite difference method (FDM)

- FDM can be an alternative to the shooting method which iterates until the BCs are met with the convergent condition

- FDM converts BVPs directly into a system of algebraic eqns rather than a sequence of IVPs as in the shooting method.

- FDM approximates its soln on a set of discrete mesh points over the domain.

- Given a scalar two-pt. BVP:

$$\begin{cases} u''(t) = f(t, u, u'), \quad t_a < t < t_b \\ \text{BC: } u(t_a) = \alpha, \quad u(t_b) = \beta. \end{cases}$$

FDM introduces mesh pts:

$$t^n = t_a + n\,\Delta t, \quad n = 0, 1, \cdots, N+1,$$

where $$\Delta t = \frac{(t_b - t_a)}{N+1}.$$

Note $$t^{N+1} = t_a + (N+1)\,\Delta t = t_a + (t_b - t_a) = t_b.$$

— We seek for approximate solns

$$v^n \approx u(t^n), \quad n = 1, 2, \cdots, N$$

(Note, we do not approximate at $n = 0$ & $n = N+1$ which are the BCs)

— We now replace $u'(t^n)$ & $u''(t^n)$ with finite difference appns:

$$\begin{cases} u'(t^n) \approx \dfrac{v^{n+1} - v^{n-1}}{2\Delta t} & \text{(central difference)} \\ u''(t^n) \approx \dfrac{v^{n+1} - 2v^n + v^{n-1}}{\Delta t^2} \end{cases}$$

— They are both $\sim O(\Delta t^2)$ for approximating $u'(t^n)$ & $u''(t^n)$, respectively.

— The resulting difference relation:

$$\frac{v^{n+1} - 2v^n + v^{n-1}}{\Delta t^2} = f\left(t^n, v^n, \frac{v^{n+1} - v^{n-1}}{2\Delta t}\right),$$

$n = 1, 2, \cdots, N$

Ex Consider the same 2-pt BVP

$$\begin{cases} u''(t) = 6t, & 0 < t < 1 \\ u(0) = 0, \quad u(1) = 1 \end{cases}$$

($0 = \alpha$, $1 = \beta$)

$$\rightarrow \frac{v^{n-1} - 2v^n + v^{n+1}}{\Delta t^2} = 6t^n, \quad n = 1, 2, \cdots N$$

with BCs: $\begin{cases} v(0) = v^0 = 0 \\ v(1) = v(t^{N+1}) = 1 \end{cases}$

→ The system of eqns become $\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b}$:

$$\underset{\sim}{A} = \frac{1}{\Delta t^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}$$

$$\underset{\sim}{b} = \begin{bmatrix} f(t^1) - \alpha/\Delta t^2 \\ f(t^2) \\ \vdots \\ f(t^{N-1}) \\ f(t^N) - \beta/\Delta t^2 \end{bmatrix} \xrightarrow{\alpha = 0,\ \beta = 1} \begin{bmatrix} 6t^1 \\ 6t^2 \\ \vdots \\ 6t^{N-1} \\ 6t^N - 1/\Delta t^2 \end{bmatrix}$$

and $\underset{\sim}{x} = \begin{bmatrix} v^1 \\ v^2 \\ \vdots \\ v^N \end{bmatrix}$

→ The tridiagonal linear system is nonsingular and can be solved for $\underset{\sim}{x}$ using numerical methods for inverting $\underset{\sim}{A}$ such as

(i) Gaussian elimination (LU factorization)

(ii) Gauss - Jordan elimination

(iii) Cholesky factorization for symmetric positive definite $\underset{\sim}{A}$

(iv) Crout's method for LU decomposition