11. Boundary-value problems

- boundary value problems
- shooting method
- finite-difference methods

Boundary-value problems

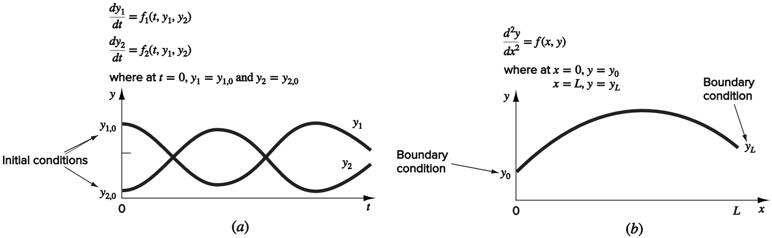
- an ODE solution requires *auxiliary conditions* to evaluate constants of integration
- for an nth-order ODE, n conditions are required

Initial-value problem (IVP)

conditions specified at same value of independent variable (*e.g.*, $x = 0$ or $t = 0$)

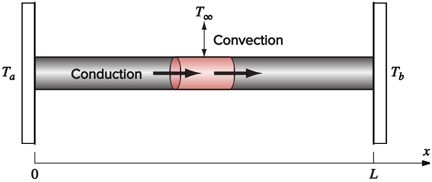
Boundary-value problem (BVP)

conditions specified at *different* points of independent variable (*e.g.*, at extreme or boundary points)



Example: heat balance in a rod

noninsulated uniform rod between two bodies with different constant temperatures



rod temperature (Kelvin) can be approximated by 2nd order linear ODE:

$$\frac{d^2T}{dx^2} + h'(T_\infty - T) = 0$$

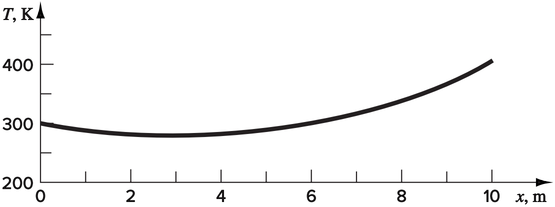
- h' = heat transfer coefficient (m^{-2})
- T_∞ = ambient gas temperature (K)
- boundary conditions: $T(0) = T_a, T(L) = T_b$

Example: heat balance in a rod

$$\frac{d^2T}{dx^2} + h'(T_\infty - T) = 0$$

- $h' = 0.05, T_\infty = 200$ K, $L = 10$
- boundary conditions: $T_a = T(0) = 300$ K, $T_b = T(10) = 400$
- the analytical solution is

$$T(x) = 200 + 20.4671e^{\sqrt{0.05}x} + 79.5329e^{-\sqrt{0.05}x}$$



Outline

- boundary value problems
- **shooting method**
- finite-difference methods

Shooting method

convert second-order ODE into two first-order ODEs:

$$\frac{dT}{dx} = z, \qquad \frac{dz}{dx} = -h'(T_\infty - T)$$

where z is the rate of change of temperature (called *gradient*)

Shooting method

- given $T(0) = T_a$, we guess $z(0) = z_{a_1}$, then solve (*e.g.*, RK4) $\Longrightarrow$ obtain T_{b_1}
- compare with boundary condition $T(L) = T_b = 200$
  - if, say $T_{b_1} > T_b$, we lower initial guess by adjusting $z(0) = z_{a_2}$ to generate T_{b_2}
- we keep adjusting until we get close enough solution, called *shooting method*
  - we adjust trajectory of our solution by guessing $z(0)$ until we hit our target $T(L) = T_b$
- for linear ODEs, linear interpolation between two erroneous shots (z_{a_1}, T_{b_1}) and (z_{a_2}, T_{b_2}) can be employed to arrive at required trajectory:

$$z_a = z_{a_1} + \frac{z_{a_2} - z_{a_1}}{T_{b_2} - T_{b_1}}(T_b - T_{b_1})$$

Example: shooting method for linear ODEs

consider same conditions as page 11.4

$$\frac{dT}{dx} = z, \quad \frac{dz}{dx} = -0.05(200 - T), \quad T(0) = 300, \quad T(10) = 400$$

Trial 1: solve with $T(0) = 300$ and $z_{a_1} = z(0) = -5$

```
%define right hand side as an Mfile
function dy = dydx(x,y)
dy = [y(2);-0.05*(200 - y(1))];
%we can then generate the solution as
[t,y] = ode45(@dydx,[0 10],[300,-5]);
Tb1 = y(length(y))
Tb1 =
569.7539
```

$T_{b_1} = 569.7539$ (too high; left plot)

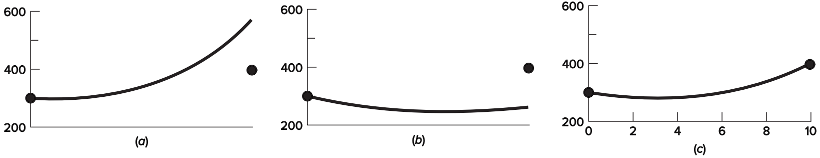
Trial 2: $z_{a_2} = -20 \Rightarrow T_{b_2} = 259.5131$ (too low; middle plot)

Example: shooting method for linear ODEs

because ODE is linear, $z(0)$ and $T(10)$ are linearly related; interpolate:

$$z_a = z(0) = -5 + \frac{-20-(-5)}{259.5131-569.7539}(400 - 569.7539) = -13.2075$$

using this initial value leads to correct solution shown on right



Nonlinear shooting method

for nonlinear boundary-value problems:

- linear interpolation between two shots is not sufficient
- quadratic interpolation (3 shots) may improve estimate but rarely exact
- alternative: recast as a *roots problem*

Root formulation

- think of solution as function of z_a

$$T_b = f(z_a)$$

- goal: $T_b = 400$
- adjust z_a such that

$$g(z_a) = f(z_a) - 200 = 0$$

thus, finding correct initial slope z_a reduces to a root-finding problem

Example: nonlinear shooting method

consider nonlinear ODE:

$$\frac{d^2T}{dx^2} + h'(T_\infty - T) + \sigma'(T_\infty^4 - T^4) = 0$$

- $h' = 0.05$, $\sigma' = 2.7 \times 10^{-9}$K^{-3}m^{-2} (heat transfer parameter)
- $T_\infty = 200$ K, $L = 10$, boundary conditions: $T(0) = 300$, $T(10) = 400$
- transformation:

$$\frac{dT}{dx} = z, \qquad \frac{dz}{dx} = -0.05(200 - T) - 2.7 \times 10^{-9}(1.6 \times 10^9 - T^4)$$

Solution strategy

1. guess initial slope $z(0)$
2. solve system (*e.g.*, using RK4)
3. obtain $T(10) = f(z_0)$
4. adjust $z(0)$ (*e.g.*, solver, bisection) until $g(z_0) = f(z_0) - 200 = 0$

Example: nonlinear shooting method

an M-file can be developed to compute the right-hand sides of these equations:

```
function dy = dydxn(x,y)
dy = [y(2);-0.05*(200 - y(1)) - 2.7e-9*(1.6e9 - y(1)^4)];
```

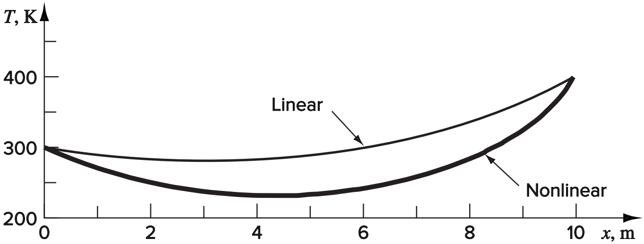
next, we can build a function to hold the residual that we will try to drive to zero as

```
function r = res(za)
[x,y] = ode45(@dydxn,[0 10],[300 za]);
r = y(length(x),1) - 400;
```

we can then find the root with the `fzero` function:

```
fzero(@res,-50)
ans =
-41.7434
```

Example: nonlinear shooting method



Remarks

- requires iterative root-finding for initial slopes
- more complex for higher-order problems (need multiple guesses)
- nonlinearities (*e.g.*, radiation, T^4) increase curvature
- for large systems, finite-difference or finite-element approaches are more practical

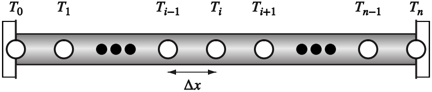
Outline

- boundary value problems
- shooting method
- **finite-difference methods**

Finite-difference methods

discretize ODE $\frac{d^2T}{dx^2} + h'(T_\infty - T) = 0$ by replacing derivatives with finite differences

- solution divided into series of node values



- for second derivative, the centered difference approximation is

$$\frac{d^2T}{dx^2} \approx \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2}$$

- substitute into differential equation and collect terms:

$$-T_{i-1} + (2 + h'\Delta x^2)T_i - T_{i+1} = h'\Delta x^2 T_\infty$$

  - applies at $n - 1$ interior nodes
  - first and last nodes, T_0 and T_n, are fixed by boundary conditions

- results in tridiagonal diagonally dominant linear system (can be solved efficiently)

Example: finite-difference solution

solve example on page 11.4 for rod with

$$L = 10, \quad h' = 0.05, \quad T_\infty = 200, \quad T_a = 300, \quad T_b = 400$$

using 4 interior nodes with segment length of $\Delta x = 2$ m:

writing equations for each node:

$$\begin{bmatrix} 2.2 & -1 & 0 & 0 \\ -1 & 2.2 & -1 & 0 \\ 0 & -1 & 2.2 & -1 \\ 0 & 0 & -1 & 2.2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 340 \\ 40 \\ 40 \\ 440 \end{bmatrix} \quad \Longrightarrow \quad T = \begin{bmatrix} 283.2660 \\ 283.1853 \\ 299.7416 \\ 336.2462 \end{bmatrix}$$

Comparison of methods

x	analytical solution	shooting method	finite difference
0	300.0000	300.0000	300.0000
2	282.8634	282.8889	283.2660
4	282.5775	282.6158	283.1853
6	299.0843	299.1254	299.7416
8	335.7404	335.7718	336.2462
10	400.0000	400.0000	400.0000

- both numerical methods accurate
- smaller Δx improves results
- finite-difference preferred for extension to complex systems

Derivative boundary conditions

$$0 = \frac{d^2T}{dx^2} + h'(T_\infty - T)$$

- fixed or *Dirichlet boundary condition* studied so far are one of several possibilities
- a comnmon alternative is *Neumann boundary condition* where derivative is given:

$$\frac{dT}{dx}(0) = T_a, \quad T(L) = T_b$$

  - one end has *derivative boundary condition*
  - other end has a *fixed boundary condition*
- we divide rod into a series of nodes and apply approximation to each interior node

$$-T_{i-1} + (2 + h'\Delta x^2)T_i - T_{i+1} = h'\Delta x^2 T_\infty$$

Finite-difference formulation with extra node

- because T_0 is not specified, the equation introduces an *imaginary node* T_{-1}

$$-T_{-1} + (2 + h'\Delta x^2)T_0 - T_1 = h'\Delta x^2 T_\infty$$

- we approximate the derivative at the left end ($x = 0$) using a centered difference:

$$\frac{dT}{dx}(0) = \frac{T_1 - T_{-1}}{2\Delta x}$$

- solving for T_{-1}:

$$T_{-1} = T_1 - 2\Delta x \cdot \frac{dT}{dx}(0)$$

- allows us to replace the imaginary node T_{-1}
- the derivative condition is embedded into the finite-difference scheme
- substituting the expression for T_{-1}:

$$(2 + h'\Delta x^2)T_0 - 2T_1 = h'\Delta x^2 T_a - 2\Delta x \cdot \frac{dT}{dx}(0)$$

Example

generate the finite-difference solution for a 10-m rod with

$$\Delta x = 2 \text{ m}, \qquad h' = 0.05 \text{ m}^{-2}, \qquad T_\infty = 200 \text{ K}$$

subject to the boundary conditions

$$T_a' = 0, \qquad T_b = 400 \text{ K}$$

- node 0 is represented by: $2.2T_0 - 2T_1 = 40$
- for the interior nodes, for example, at node 1: $-T_0 + 2.2T_1 - T_2 = 40$
- similar form is written for the remaining interior nodes

$$\begin{bmatrix} 2.2 & -2 & 0 & 0 & 0 \\ -1 & 2.2 & -1 & 0 & 0 \\ 0 & -1 & 2.2 & -1 & 0 \\ 0 & 0 & -1 & 2.2 & -1 \\ 0 & 0 & 0 & -1 & 2.2 \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 40 \\ 40 \\ 40 \\ 40 \\ 440 \end{bmatrix} \implies T = \begin{bmatrix} 243.0278 \\ 247.3306 \\ 261.0994 \\ 287.0882 \\ 330.4946 \end{bmatrix}$$

Finite-difference approaches for nonlinear ODEs

for nonlinear ordinary differential equations, replacing derivatives with finite differences produces a system of *nonlinear simultaneous equations*

- most general solution approach is to use root-finding methods
- a common choice is the Newton-Raphson method
- although effective, this approach can be computationally involved
- an adaptation of **successive substitution** may provides a simpler alternative

Example

$$0 = \frac{d^2T}{dx^2} + h'(T_\infty - T) + \sigma''(T_\infty^4 - T^4)$$

using the finite-difference approximation for the second derivative,

$$\frac{d^2T}{dx^2} \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

and collecting terms yields

$$-T_{i-1} + \left(2 + h'\Delta x^2\right) T_i - T_{i+1} = h'\Delta x^2 T_\infty + \sigma''\Delta x^2 \left(T_\infty^4 - T_i^4\right)$$

- right-hand side contains the nonlinear term T_i^4
- left-hand side has the form of a *linear diagonally dominant system*
- if T_i^4 is approximated by previous iteration value, then T_i can be updated explicitly

Example

using the previous iteration for the nonlinear term, solve for T_i:

$$T_i = \frac{h'\Delta x^2 T_\infty + \sigma''\Delta x^2 \left(T_\infty^4 - T_i^4\right) + T_{i-1} + T_{i+1}}{2 + h'\Delta x^2}$$

- each T_i is treated as variable (i index for variable not iteration)
- this is similar in spirit to the Gauss-Seidel method
- each node temperature is updated successively
- the process is repeated until convergence is achieved

Remarks on convergence

- this approach does not work for all nonlinear problems
- however, it converges for many ODEs arising from physical systems
- therefore, it can be a practical tool for engineering and scientific applications

Example

consider

$$\sigma'' = 2.7 \times 10^{-9}\ \mathrm{K}^{-3}\mathrm{m}^{-2}, \qquad L = 10\ \mathrm{m}, \qquad h' = 0.05\ \mathrm{m}^{-2}, \qquad T_\infty = 200\ \mathrm{K}$$

and boundary conditions

$$T(0) = 300\ \mathrm{K}, \qquad T(10) = 400\ \mathrm{K}$$

let us use four interior nodes with $\Delta x = 2\ \mathrm{m}$

- boundary nodes are fixed:

$$T_0 = 300, \qquad T_5 = 400$$

- initial guesses for the interior nodes are taken as zero:

$$T_1 = T_2 = T_3 = T_4 = 0$$

Example

First iteration: for the first interior node,

$$T_1 = \frac{0.05(2)^2(200) + 2.7 \times 10^{-9}(2)^2(200^4 - 0^4) + 300 + 0}{2 + 0.05(2)^2} = 159.2432$$

for the second interior node,

$$T_2 = \frac{0.05(2)^2(200) + 2.7 \times 10^{-9}(2)^2(200^4 - 0^4) + 159.2432 + 0}{2 + 0.05(2)^2} = 97.9674$$

for the third interior node,

$$T_3 = \frac{0.05(2)^2(200) + 2.7 \times 10^{-9}(2)^2(200^4 - 0^4) + 97.9674 + 0}{2 + 0.05(2)^2} = 70.4461$$

for the fourth interior node,

$$T_4 = \frac{0.05(2)^2(200) + 2.7 \times 10^{-9}(2)^2(200^4 - 0^4) + 70.4461 + 400}{2 + 0.05(2)^2} = 226.8704$$

Example

iterative process is repeated until the solution converges to an acceptable tolerance:

$$T_0 = 300$$

$$T_1 = 250.4827$$

$$T_2 = 236.2962$$

$$T_3 = 245.7596$$

$$T_4 = 286.4921$$

$$T_5 = 400$$

References and further readings

- S. C. Chapra and R. P. Canale. *Numerical Methods for Engineers* (8th edition). McGraw Hill, 2021. (Ch.27)

- S. C. Chapra. *Applied Numerical Methods with MATLAB for Engineers and Scientists* (5th edition). McGraw Hill, 2023. (Ch.24)