Numerical Solutions of Boundary-Value Problems in ODEs

Larry Caretto
Mechanical Engineering 501A

Seminar in Engineering Analysis

November 27, 2017

California State University Northridge

Outline

- Review stiff equation systems
- Definition of boundary-value problems (BVPs) in ODEs
- Numerical solution of BVPs by shoot-and-try method
- Use of finite-difference equations to solve BVPs
  - Thomas algorithms for solving finite-difference equations from second-order BVPs

California State University Northridge 2

Stiff Systems of Equations

- Some problems have multiple exponential terms with differing coefficients, a, in exp(-at)
- Coefficients with large values of a will require a small time step for stability, but will not be essentially zero after a short time after the start of the solution
- Need special algorithms for such systems

California State University Northridge 3

Solving Stiff ODEs

- If you try to solve a stiff problem with a conventional solver you will find that the solution is taking excessive time
- Stiff solvers do more work per step, but allow larger steps
  - Gear's Method and MATLAB stiff solvers ode15s, ode23s, ode23t, ode23tb
- Users may have to provide code to complete Jacobian matrix, $\partial f_i/\partial y_j$

California State University Northridge 4

Boundary-Value Problems

- All ODEs solved so far have initial conditions only
  - Conditions for all variables and derivatives set at t = 0 only
- In a boundary-value problem, we have conditions set at two different locations
- A second-order ODE $d^2y/dx^2 = g(x, y, y')$, needs two boundary conditions (BC)
  - Simplest are y(0) = a and y(L) = b
  - Mixed BC: ady/dx+by = c at x = 0, L

California State University Northridge 5

Boundary-value Problems II

- Solving boundary-value problems
  - Finite differences (considered later)
  - Shoot-and-try
    - Take an initial guess of derivative boundary conditions at x = 0 and use an initial-value routine to get $y_{(comp)}(L)$ at the other boundary
    - Compare the value of $y_{(comp)}(L)$ found from the previous step to the boundary condition on y(L)
    - Use the difference between $y_{(comp)}(L)$ and y(L) to iterate the initial value of $z = dy/dx|_{x=0}$ and continue until $y_{(comp)}(L) \approx y(L)$

California State University Northridge 6

Shoot-and-Try Example I

- Look at single, second order equation: y" = g(x, y, y'), y(0) = a and y(L) = b
- Define z = y' (y" = z') to get two first order equations: z' = g(x, y, z) and y' = z
- Steps in the shoot and try method
  - Guess initial condition for z(0) = y'(0); typically guess $z^{(0)}(0)$ = [y(L) – y(0)]/L
  - Solve equations for $y_{computed}$(L) and compare to specified boundary condition, y(L)

California State University Northridge 7

Shoot-and-Try Notation

- Notation for shoot and try
  - ODE: d^2y/dx^2 = g(x, y, dy/dx) = g(x, y, z)
  - System of two first order ODES: z = dy/dx and dz/dx = g(x, y, z)
  - Variables for iteration m
    - $z^{(m)}(0)$ guess for initial condition of dy/dx at x = 0
    - $y^{(m)}(L)$ result at x = L from solving system of two ODEs using $z^{(m)}(0)$
    - y(L) required boundary condition at x = L
    - Error at iteration m: $E^{(m)} = y^{(m)}(L) – y(L)$

California State University Northridge 8

Shoot-and-Try Iteration

- Adjust $z^{(m)}(0)$ until $|E^{(m)}| = |y^{(m)}(L) – y(L)|$ is less than the allowed error
- After first try with $z^{(0)}(0)$ = [y(L) – y(0)]/L try $z^{(1)}(0) = [2y(L) – y^{(0)}(L) – y(0)]/L$
- For subsequent tries use linear interpolation to give zero error

Set this to zero

$$z^{(m+1)}(0) = z^{(m)}(0) + \frac{z^{(m)}(0) - z^{(m-1)}(0)}{E^{(m)} - E^{(m-1)}}\left(E^{(m+1)} - E^{(m)}\right)$$

$$z^{(m+1)}(0) = z^{(m)}(0) - E^{(m)}\frac{z^{(m)}(0) - z^{(m-1)}(0)}{E^{(m)} - E^{(m-1)}}$$

California State University Northridge 9

Shoot-and-Try Example II

- Solve $d^2y/dx^2 + 16\sin(y^2) = 0$ with y = 1 at x = 0 and y = 0 at x = L = 1
- Must find pair of first order equations
  - Set dy/dx = z as one ODE
  - Original ODE becomes dz/dx = –16sin(y)
  - We know y(0) = 1, but we need z(0) guess $z^{(0)}(0)$ = [y(L) – y(0)]/L = (0 – 1)/1 = –1
    - This gives $y^{(0)}(1)$ = –3.8870 (RK4, h = .05)
    - Try $z^{(1)}(0) = [2y(1) – y^{(0)}(1) – y(0)]/L$ = [2(0) – (–3.8870) – 1] = 2.8870

California State University Northridge $E^{(0)} = y^{(0)}(L) – y(L)$ = –3.8870 – 0 = -3.8870 10

Shoot-and-Try Example III

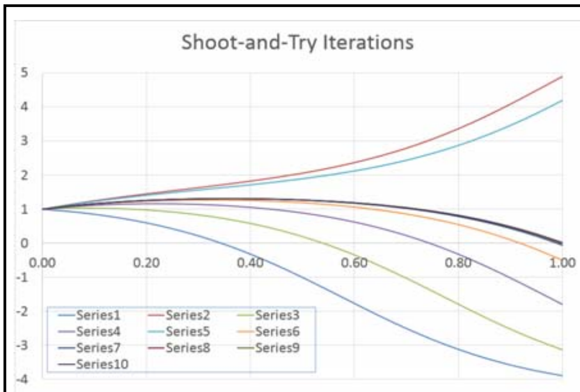
- RK4 with $z^{(1)}(0)$ = 2.8870 gives $y^{(1)}(L)$ = 8.1680 so $E^{(1)} = y^{(1)}(L) – y(L)$ = 8.1680 – 0 = 8.1680
- Apply general error formula to get $z^{(2)}(0)$

$$z^{(m+1)}(0) = z^{(m)}(0) - E^{(m)}\frac{z^{(m)}(0) - z^{(m-1)}(0)}{E^{(m)} - E^{(m-1)}}$$

$$z^{(2)}(0) = -1 - 8.1680\frac{2.8870 - -1}{8.1680 - -3.8870} = 0.71993$$

- Continue to apply Runge-Kutta to get $y^{(m)}(L=1)$, $E^{(m)} = y^{(m)}(L) – y(L)$, and $z^{(m+1)}(0)$ for $E^{(m+1)} \approx 0$
- Repeat calculations with new value of $z^{(m+1)}(0)$ from general error formula until $y^{(m)}(L) \approx y(L)$

California State University Northridge 11



California State University Northridge 12

Finite-Difference Introduction

- Finite-difference approach is alternative to shoot-and-try
  - Construct grid of step size h (variable h possible) between boundaries
    - Similar to grid used for numerical integration
    - $x_0 = x(0)$, $x_N = x(L)$, $h = L / N$, $x_k = x_0 + kh$
  - Replace differential equation at each interior node by finite difference equation
  - Solve resulting set of algebraic equations for interior points using Thomas algorithm

California State University Northridge 13

Finite Difference Grid

- Grid may be uniform or non-uniform, but uniform is easier and has higher order truncation error; note: $h = (x_N - x_0)/N$

●-----●--------●-------------●~ ~●-------●---●

x_0 x_1 x_2 x_3 x_{N-2} x_{N-1} x_N

- At each node write finite-difference equivalent to differential equation
- Handle boundary conditions at x_0 and x_N (simplest if $y_0 = y(0)$ and $y_N = y(L)$, but can have gradient or mixed boundaries)

California State University Northridge 14

Example Problem

- Solve $d^2T/dx^2 + a^2T = 0$
- Finite difference equation at node i
- $[d^2T/dx^2 + a^2T]_i = (T_{i+1} + T_{i-1} - 2T_i)/h^2 + a^2T_i$ $= 0$ **Ignore truncation error**
- Ignore truncation error and get finite-difference equation system
- $T_{i+1} + T_{i-1} - 2T_i + h^2a^2T_i = 0$
- Have N+1 nodes numbered from 0 to N with boundary conditions at 0 and N

California State University Northridge 15

General Boundary Conditions

- Must be able to handle three kinds
  - Dirichlet – specify variables at boundary
  - Neumann – specify boundary gradients
  - Mixed or third kind – specify relationship between value and gradient at boundary
- General format a dT/dx + bT = c (mixed)
  - Fixed T: a = 0, b = 1, c = boundary T
  - Gradient: b = 0, a = 1, c = value for dT/dx
- Use directional finite-difference equation for boundary gradients, dT/dx

California State University Northridge 16

Example Continued

- Equation is $T_{i-1} + (-2 + h^2a^2)T_i + T_{i+1} = 0$
- Specify boundary values T_A and T_B
  - $T_0 = T(x=0) = T_A$ and $T_N = T(x=L) = T_B$
- With specified boundary values equations at i = 1 and i = N-1 become
  - $(-2 + h^2a^2)T_1 + T_2 = -T_A$
  - $T_{N-2} + (-2 + h^2a^2)T_{N-1} = -T_B$
- Resulting system of equations forms tridiagonal matrix

California State University Northridge 17

Example Matrix Equations

- Finite-difference equations in matrix form with $\alpha = a^2h^2$ have tridiagonal form solved by Thomas Algorithm used with cubic spline (see end slides)

$$\begin{bmatrix} -2+\alpha & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2+\alpha & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2+\alpha & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & 0 & \cdots & -2+\alpha & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2+\alpha \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ \vdots \\ T_{N-2} \\ T_{N-1} \end{bmatrix} = \begin{bmatrix} -T_A \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ -T_B \end{bmatrix}$$

California State University Northridge 18

Analytical Solution Comparison

- Look at results for h = 0.1 (N = 10) with T_A = 0, T_B = 1, a = 2 and L = 1
- Compare to exact solution below
  - Exact gradients also used in comparison

$$T = \frac{T_B - T_A\cos(aL)}{\sin(aL)}\sin(ax) + T_A\cos(ax)$$

$$q_{x=0} = -k\left.\frac{dT}{dx}\right|_{x=0} = -ka\frac{T_B - T_A\cos(aL)}{\sin(aL)}$$

$$q_{x=L} = -k\left.\frac{dT}{dx}\right|_{x=L} = \frac{ka\left[T_A - T_B\cos(aL)\right]}{\sin(aL)}$$

California State University Northridge 19

Results of Finite-Difference Calculations

i	x_i	T_i	Exact T_i	Error
0	0.0	0	0	0
1	0.1	0.21918	0.21849	0.00070
2	0.2	0.42960	0.42826	0.00134
3	0.3	0.62284	0.62097	0.00187
4	0.4	0.79115	0.78891	0.00224
5	0.5	0.92783	0.92541	0.00242
6	0.6	1.02739	1.02501	0.00238
7	0.7	1.08585	1.08375	0.00211
8	0.8	1.10088	1.09928	0.00160
9	0.9	1.07188	1.07099	0.00089
10	1.0	1	1	0

Error and Error Order

- Get overall measure of error (like norm of a vector)
- Typically use maximum error (in absolute value) or root-mean-squared (RMS) error
- N = 10 has ε_{max} = 2.42x10^{-3} and ε_{RMS} = 1.83x10^{-3}. For N = 100, ε_{max} = 2.41x10^{-5} and ε_{RMS} = 1.73x10^{-5}
- Second-order error in solution

$$|\varepsilon|_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_{exact} - T_{numerical})_i^2}$$

California State University Northridge 21

Boundary Gradients

- Use second-order derivative expressions

$$q_0 = -k\left.\frac{dT}{dx}\right|_{x=x_0} = -k\frac{-3T_0 + 4T_1 - T_2}{2h}$$

$$q_N = -k\left.\frac{dT}{dx}\right|_{x=x_N} = -k\frac{3T_N - 4T_{N-1} + T_{N-2}}{2h}$$

x	$-q_{exact}/k$	h	-q/k	Error
0	2.1995	.1	2.2357	.03618
0	2.1995	.01	2.1999	.00036
1	-.9153	.1	-.9332	.01786
1	-.9153	.01	-.9155	.00021

California State University Northridge 22

MATLAB Boundary-value ODEs

- MATLAB has two solvers bvp4c and bvp5c for solving boundary-value ODEs
  - bvp5c: finite difference code implements four-stage Lobatto IIIa formula, a collocation formula that provides a C^1-continuous solution that is fifth-order accurate uniformly in [a,b]
  - bvp5c solves algebraic equations directly; bvp4c uses analytical condensation
  - bvp4c handles unknown parameters directly

California State University Northridge 23

Solve Tridiagonal Equations

- Finite-difference equations in matrix for example problem with $\alpha = a^2h^2$
- Use Thomas Algorithm for Solution

$$\begin{bmatrix} -2+\alpha & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2+\alpha & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2+\alpha & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & 0 & \cdots & -2+\alpha & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2+\alpha \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ \vdots \\ T_{N-2} \\ T_{N-1} \end{bmatrix} = \begin{bmatrix} -T_A \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ -T_B \end{bmatrix}$$

California State University Northridge 24

Thomas Algorithm

- General set of tridiagonal equations

$$\begin{bmatrix} B_0 & C_0 & 0 & 0 & \cdots & 0 & 0 \\ A_1 & B_1 & C_1 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & B_2 & C_2 & \cdots & 0 & 0 \\ 0 & 0 & A_3 & B_3 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & B_{N-1} & C_{N-1} \\ 0 & 0 & 0 & 0 & \cdots & A_N & B_N \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ \vdots \\ D_{N-1} \\ D_N \end{bmatrix}$$

California State University Northridge

25

Thomas Algorithm II

- Gauss elimination upper triangular form

$$\begin{bmatrix} 1 & -E_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -E_1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -E_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -E_{N-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ \vdots \\ F_{N-1} \\ F_N \end{bmatrix}$$

Have to find E_i and F_i

California State University Northridge

26

Thomas Algorithm III

- Forward computations
  - Initial: $E_0 = - C_0 / B_0$ $F_0 = D_0 / B_0$
  - Apply equations below for i = 1,… N-1:

$$E_i = \frac{-C_i}{B_i + A_i E_{i-1}} \qquad F_i = \frac{D_i - A_i F_{i-1}}{B_i + A_i E_{i-1}}$$

  - At final point

$$x_N = F_N = \frac{D_N - A_N F_{N-1}}{B_N + A_N E_{N-1}}$$

- Back substitute: $x_i = F_i + E_i x_{i+1}$

California State University Northridge

27

Result: h = .1, a = 2, T_0 = 0, T_N = 1

x_i	A_i	B_i	C_i	D_i	E_i	F_i	y_i
0.1	1	-1.96	1	0	0.51020	0	0.21918
0.2	1	-1.96	1	0	0.68975	0	0.42960
0.3	1	-1.96	1	0	0.78725	0	0.62284
0.4	1	-1.96	1	0	0.85270	0	0.79115
0.5	1	-1.96	1	0	0.90309	0	0.92783
0.6	1	-1.96	1	0	0.94616	0	1.02739
0.7	1	-1.96	1	0	0.98635	0	1.08585
0.8	1	-1.96	1	0	1.02706	0	1.10088
0.9	1	-1.96	1	-1		1.07188	1.07188

Input

Forward Calculations

Back substitute

California State University Northridge

28

Spreadsheet Formulas

	A	B	C	D	E	F	G
1	Ai	Bi	Ci	Di	Ei	Fi	x_i
2		-1.96	1	0	=-C2/B2	=D2/B2	=F2+E2*G3
3	1	-1.96	1	0	=-C3/(B3+A3*E2)	=(D3-A3*F2)/(B3+A3*E2)	=F3+E3*G4
4	1	-1.96	1	0	=-C4/(B4+A4*E3)	=(D4-A4*F3)/(B4+A4*E3)	=F4+E4*G5
5	1	-1.96	1	0	=-C5/(B5+A5*E4)	=(D5-A5*F4)/(B5+A5*E4)	=F5+E5*G6
6	1	-1.96	1	0	=-C6/(B6+A6*E5)	=(D6-A6*F5)/(B6+A6*E5)	=F6+E6*G7
7	1	-1.96	1	0	=-C7/(B7+A7*E6)	=(D7-A7*F6)/(B7+A7*E6)	=F7+E7*G8
8	1	-1.96	1	0	=-C8/(B8+A8*E7)	=(D8-A8*F7)/(B8+A8*E7)	=F8+E8*G9
9	1	-1.96	1	0	=-C9/(B9+A9*E8)	=(D9-A9*F8)/(B9+A9*E8)	=F9+E9*G10
10	1	-1.96		-1	=-C10/(B10+A10*E9)	=(D10-A10*F9)/(B10+A10*E9)	=F10

Formulas here solve only for interior points when fixed boundary conditions are specified

California State University Northridge

29

Other Boundary Conditions

- General condition a dT/dx + bT = c
  - a = 1, b = 0 for Neumann (gradient given)
  - a = 0, b = 1 for Dirichlet (value given)
  - Write gradient using second order forward ($x = x_0$) or backward difference ($x = x_N$)
  - Combine with equation for first node in from the boundary to eliminate term with second node from boundary
  - Result conforms to tridiagonal system

California State University Northridge

30

General Boundary Example

- Look at $x = x_0$ boundary; results for $x = x_N$ follow similar derivation

$$a_0 \left.\frac{dy}{dx}\right|_{x=x_0} + b_0 y_0 = a_0 \frac{-3y_0 + 4y_1 - y_2}{2h} + b_0 y_0 = c_0$$

- Add these two equations eliminating y_2

$$\left(b_0 - \frac{3a_0}{2h}\right) y_0 + \frac{4a_0}{2h} y_1 - \frac{a_0}{2h} y_2 = c_0$$

$$\left(y_0 + (\alpha - 2) y_1 + y_2 = 0\right) \frac{a_0}{2h}$$

$$\left(b_0 - \frac{2a_0}{2h}\right) y_0 + \frac{(\alpha + 2) a_0}{2h} y_1 = c_0$$

Northridge 31

General Boundary Example II

- Equation just derived is seen to give correct Dirichlet result for $a_0 = 0$, $b_0 = 1$

$$\left(b_0 - \frac{2a_0}{2h}\right) y_0 + \frac{(\alpha + 2) a_0}{2h} y_1 = c_0$$

- Similar derivation at $x = x_N$ gives

$$\left(b_N + \frac{2a_N}{2h}\right) y_N + \frac{(2 - \alpha) a_N}{2h} y_{N-1} = c_N$$

- Equations shown here will work for a = 0 or b = 0, but at least one must be nonzero

Northridge 32

Nonlinear Problems

- Shoot-and-try requires no special procedures for nonlinear problems
- For finite difference or finite elements, solve a linearized equation
  - Example is pendulum equation $d^2\theta/dt^2 = (-g/l) \sin\theta$ (usually solved with $\sin\theta \approx \theta$)
  - Taylor series: $\sin\theta = \sin\theta_0 + [d(\sin\theta)/d\theta]_0 (\theta - \theta_0) = \sin\theta_0 + \cos\theta_0 (\theta - \theta_0)$
  - Replace $\sin\theta$ by linear result to iterate

Northridge 33

Nonlinear Example

- Start with $d^2\theta/dt^2 = (-g/l) \sin\theta$
- Replace $\sin\theta$ by linearized series
- Write in iterative form with $\theta^{(m+1)}$ as new iteration and use $\theta^{(m)}$ in nonlinear terms
- $d^2\theta^{(m+1)}/dt^2 = (-g/l) [\sin\theta^{(m)} + \cos\theta^{(m)} (\theta^{(m+1)} - \theta^{(m)})$
- Define $\omega^2 = g/l$ and rearrange
  $d^2\theta^{(m+1)}/dt^2 + \omega^2\theta^{(m+1)} \cos\theta^{(m)} = -\omega^2 [\sin\theta^{(m)} - \theta^{(m)}\cos\theta^{(m)}] = r$

Northridge 34

Nonlinear Example II

- Convert $d^2\theta^{(m+1)}/dt^2 + \omega^2\theta^{(m+1)} \cos\theta^{(m)} = r$ to (linear) finite-difference form in $\theta^{(m+1)}$

$$\frac{\theta_{i+1}^{(m+1)} + \theta_{i-1}^{(m+1)} - 2\theta_i^{(m+1)}}{h^2} + \omega^2 \theta_i^{(m+1)} = r_i$$

$$r_i = \omega^2 \left[\theta_i^{(m)} \cos\theta_i^{(m)} - \sin\theta_i^{(m)}\right]$$

- Have tridiagonal system

$$\theta_{i-1}^{(m+1)} + \left[\omega^2 h^2 - 2\right] \theta_i^{(m+1)} + \theta_{i+1}^{(m+1)} = h^2 r_i$$

Northridge 35

Nonlinear Example III

- Make initial guesses for $\theta^{(0)}$
  - Linear profile $\theta^{(0)}(t) = \theta(0) + [\theta(L) - \theta(0)]t/T$
- Find all nodal values for $\theta^{(1)}$ using $\theta^{(0)}$ to compute the nonlinear terms
- Repeat the process until the differences between iterations is good enough
  - Compute residuals to test convergence

$$R_i = \theta_{i-1}^{(m+1)} + \left[\omega^2 h^2 - 2\right] \theta_i^{(m+1)} + \theta_{i+1}^{(m+1)} - h^2 r_i^{(m+1)}$$

Northridge 36

Summary

- Boundary value problems require special treatment
  - Shoot-and-try
  - Finite differences or finite elements
- Shoot-and-try is usually better for nonlinear problems and finite differences are better for linear ones
- Finite elements are more applicable to complex geometry 2D and 3D problems

California State University Northridge

37