Two-point Boundary Value Problems: Numerical Approaches

Math 615, Spring 2014
abbreviations

- ODE = ordinary differential equation
- PDE = partial differential equation
- IVP = initial value problem
- BVP = boundary value problem
Outline

1. classical IVPs and BVPs with by-hand solutions

2. a serious problem: a BVP for equilibrium heat

3. finite difference solution of two-point BVPs

4. shooting to solve two-point BVPs

5. a more serious example: solutions
Example 1: ODE IVP. find $y(x)$ if

$$y'' + 2y' - 8y = 0, \quad y(0) = 1, \quad y'(0) = 0$$

Example 2: ODE BVP. find $y(x)$ if

$$y'' + 2y' - 8y = 0, \quad y(0) = 1, \quad y(1) = 0$$
classical ODE problems: IVP vs BVP

Example 1: ODE IVP. find $y(x)$ if

$$y'' + 2y' - 8y = 0, \quad y(0) = 1, \quad y'(0) = 0$$

Example 2: ODE BVP. find $y(x)$ if

$$y'' + 2y' - 8y = 0, \quad y(0) = 1, \quad y(1) = 0$$

- both problems can be solved by hand
- in fact, the ODE has constant coefficients so we can find characteristic polynomial and general solution ... like this:
  if $y(x) = e^{rx}$ then $r^2 + 2r - 8 = (r + 4)(r - 2) = 0$ so

\[
y(x) = c_1 e^{-4x} + c_2 e^{2x}
\]

- Example 1 gives system
  \[
c_1 + c_2 = 1, \quad -4c_1 + 2c_2 = 0
\]
  for coefficients; get solution
  \[
y(x) = (1/3)e^{-4x} + (2/3)e^{2x}
\]

- Example 2 gives system
  \[
c_1 + c_2 = 1, \quad e^{-4}c_1 + e^{2}c_2 = 0
\]
  for coefficients; get solution
  \[
y(x) = (1 - e^{-6})^{-1}e^{-4x} + (1 - e^{6})^{-1}e^{2x}
\]
viewing solns with MATLAB

```matlab
x = 0:.001:1;
y1 = exp(-4*x); y2 = exp(2*x);
yIVP = (1/3)*y1 + (2/3)*y2;
yBVP = (1/(1-exp(-6)))*y1 + (1/(1-exp(6)))*y2;
plot(x,yIVP,x,yBVP), grid on
legend('IVP soln','BVP soln')
```
obvious name: “two-point BVP”

• *Example 2* above is called a “two-point BVP”
• a two-point BVP includes an ODE and the value(s) of the solution at two different locations
• the ODE can be of any order, as long as it is at least *two*, because first-order ODEs cannot satisfy two conditions (generally)
• *but* there is no guarantee that a two-point BVP can be solved (see below)
• we will also consider boundary value problems for PDEs in this course (i.e. problems including no initial values)
a standard manipulation of a 2nd order ODE

Consider the general linear 2nd-order ODE:

\[ y'' + p(x)y' + q(x)y = r(x) \tag{1} \]

Also consider the general 2nd-order ODE:

\[ y'' = f(x, y, y') \tag{2} \]

- these can be written as systems of coupled 1st-order ODEs
- equation (1) is equivalent to

\[
\begin{pmatrix} y' \\ v' \end{pmatrix} = \begin{pmatrix} v \\ -p(x)v - q(x)y + r(x) \end{pmatrix}
\]

- equation (2) is equivalent to

\[
\begin{pmatrix} y' \\ v' \end{pmatrix} = \begin{pmatrix} v \\ f(x, y, v) \end{pmatrix}
\]

- first order systems are the form in which to apply a numerical ODE solver
why IVP are *better* problems than BVPs

- IVPs have unique solutions
- we say they are “well-posed”; specifically:

**Theorem**

Consider the system of ODEs

\[
\frac{dy}{dt} = f(t, y),
\]

where \( y(t) = (y_1(t), \ldots, y_d(t)) \) and \( f = (f_1, \ldots, f_d) \) are vector-valued functions. If \( f \) is continuous for \( t \) in an interval around \( t_0 \) and for \( y \) in some region around \( y_0 \), and if \( \partial f_i / \partial y_j \) is continuous for the same inputs and for all \( i \) and \( j \), then the IVP consisting of (3) and \( y(t_0) = y_0 \) has a unique solution \( y(t) \) for at least some small interval \( t_0 - \epsilon < t < t_0 + \epsilon \) for some \( \epsilon > 0 \).

- given comments on last slide, this theorem also covers IVPs for 2nd-order scalar ODEs
warning about apparently-easy BVPs

Example 3: ODE BVP. find $y(x)$ if

$$y'' + \pi^2 y = 0, \quad y(0) = 1, \quad y(1) = 0$$

- this turns out to be impossible . . . there is no such $y(x)$
- in fact, the general solution to the ODE is

$$y(x) = c_1 \cos(\pi x) + c_2 \sin(\pi x)$$

so the first boundary condition implies $c_1 = 1$
- . . . but then the second condition says

$$0 = y(1) = -1 + c_2 \sin(\pi)$$

and this has no solution because $\sin(\pi) = 0$
- this is a constant-coefficient problem for which all the “parts” are “well-behaved” . . . but it is a BVP
Outline

1. classical IVPs and BVPs with by-hand solutions
2. a serious problem: a BVP for equilibrium heat
3. finite difference solution of two-point BVPs
4. shooting to solve two-point BVPs
5. a more serious example: solutions
an equilibrium heat example

- as noted in lecture and by Morton & Mayers, a PDE like this is a general description of heat flow in a rod:

\[
\rho c \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left( k(x) \frac{\partial u}{\partial x} \right) + r(x)u + s(x) \quad (4)
\]

- recall that, roughly speaking, \( \rho \) is a density, \( c \) a specific heat, \( k(x) \) a conductivity, \( r(x) \) a reaction coefficient, and \( s(x) \) is an external source of heat
an equilibrium heat example, cont

• *equilibrium* means no change in time; the equilibrium version of (4) is this:

\[ 0 = \frac{\partial}{\partial x} \left( k(x) \frac{\partial u}{\partial x} \right) + r(x)u + s(x) \]

• we can use ordinary derivative notation; the equilibrium equation is an ODE:

\[ (k(x)u')' + r(x)u = -s(x) \quad (5) \]

• suppose the rod has length $L$

• example boundary values are (i) insulation at the left end and (ii) zero temperature at the right end:

\[ u'(0) = 0, \quad u(L) = 0 \quad (6) \]
an equilibrium heat example, cont

- some concrete choices in my example include $L = 3$ and:

$$k(x) = \frac{1}{2} \arctan(20(x - 1)) + 1,$$

$$r(x) = r_0 = \frac{1}{2}, \quad s(x) = e^{-(x-2)^2}$$
• code used to produce the previous picture

```matlab
L = 3;
k = @(x) 0.5 * atan((x-1.0) * 20.0) + 1.0;
r0 = 0.5;
s = @(x) exp(-(x-2.0).^2);
J = 300;
dx = L / J;
x = 0:dx:L;
plot(x,k(x),x,r0*ones(size(x)),x,s(x))
grid on, xlabel x
legend('k(x)','r(x)=r_0','s(x)')
```
an equilibrium heat example, cont

- we have set up a non-constant-coefficient boundary value problem to solve:

\[(k(x)u')' + r_0 u = -s(x), \quad u'(0) = 0, \quad u(3) = 0\]  \hspace{1cm} (7)

- \(u(x)\) represents the equilibrium distribution of temperature in a rod with these properties:
  - conductivity \(k(x)\): the first third \([0, 1]\) is a material with much lower conductivity than the last two-thirds \([2, 3]\)
  - reaction rate \(r_0 > 0\): constant rate of linear-in-temperature heating
  - source term \(s(x)\): an external heat source concentrated around \(x = 2\)

- \textbf{Question}: what is \(u(0)\), the temperature at the left end?
- I will call this my “serious problem”, and solve it numerically two different ways
plan from here

1. introduce finite difference approach on really-easy “toy” two-point BVP
2. introduce shooting method on same toy problem
3. demonstrate both approaches on “serious problem”
Outline

1. classical IVPs and BVPs with by-hand solutions
2. a serious problem: a BVP for equilibrium heat
3. finite difference solution of two-point BVPs
4. shooting to solve two-point BVPs
5. a more serious example: solutions
finite differences

- finite difference methods for two-point BVPs generalize to PDEs . . . as demonstrated in the rest of Math 615
- here we are just solving ODEs

- recall:

\[
\frac{f(x - h) - 2f(x) + f(x + h)}{h^2} = f''(x) + \frac{f^{(4)}(\nu)}{12} h^2
\]
toy example problem

• consider this easy BVP:

\[ y'' = 12x^2, \quad y(0) = 0, \quad y(1) = 0 \]

• it has exact solution \( y(x) = x^4 - x \)

• please check my last claim

• make sure you could solve this yourself!
To solve the problem of approximating the function $y'' = 12x^2$ by finite differences, we follow these steps:

1. **Cut up the interval** $[0, 1]$ into $J$ subintervals:
   \[
   \Delta x = \frac{1}{J}
   \]
   \[
   x_j = 0 + (j - 1)\Delta x \quad (j = 1, \ldots, J + 1)
   \]

2. **Note that my indices run from** $j = 1$ to $j = J + 1$.

3. **Let** $Y_j$ **be the approximation to** $y(x_j)$.

4. **For each of** $j = 2, \ldots, J$ **we approximate**
   \[
   y'' = 12x^2
   \]
   **by**
   \[
   \frac{Y_{j-1} - 2Y_j + Y_{j+1}}{\Delta x^2} = 12x_j^2
   \]

5. **The boundary conditions are:** $Y_1 = 0$, $Y_{J+1} = 0$.

This approach allows us to approximate the second derivative of the function using finite differences, providing a numerical solution to the problem.
so now we have a linear system of $J + 1$ equations in $J + 1$ unknowns:

\[
\begin{align*}
Y_1 &= 0 \\
Y_1 - 2Y_2 + Y_3 &= 12x_2^2 \Delta x^2 \\
Y_2 - 2Y_3 + Y_4 &= 12x_3^2 \Delta x^2 \\
&\vdots & \vdots \\
Y_{J-1} - 2Y_J + Y_{J+1} &= 12x_J^2 \Delta x^2 \\
Y_{J+1} &= 0
\end{align*}
\]
Two-point Boundary Value Problems: Numerical Approaches

Bueker

classical IVPs and BVPs
serious problem
finite difference
shooting
serious example: solved

toy example: as matrix problem

- this is a matrix problem:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & \cdots & 0 \\
1 & -2 & 1 & 0 & \cdots & 0 \\
0 & 1 & -2 & 1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 1 & -2 & 1 & \\
0 & \cdots & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
Y_1 \\
Y_2 \\
Y_3 \\
\vdots \\
Y_J \\
Y_{J+1} \\
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
12x_2^2 \Delta x^2 \\
12x_3^2 \Delta x^2 \\
\vdots \\
12x_J^2 \Delta x^2 \\
0 \\
\end{bmatrix}
\]

- i.e.

\[AY = b\]
toy example: as matrix problem in OCTAVE

- the matrix $A$ is tridiagonal
- which is usually true of finite difference methods for two-point boundary value problems for second order ODEs
- $A$ has lots of zero entries
- use MATLAB’s sparse to store it
- the locations of nonzero entries, and the nonzero values, are stored; this saves space
- the backslash command in MATLAB is an “expert system”
  - recognizes sparsity pattern
  - exploits it to speed up matrix/vector operations
- use spy and full to see sparse matrices
toy example: as matrix problem in OCTAVE, cont

- setting up the matrix problem looks like:

```matlab
J = 10; dx = 1/J; x = (0:dx:1)';
b = zeros(J+1,1);
b(2:J) = 12 * dx^2 * x(2:J).^2;
A = sparse(J+1,J+1);
A(1,1) = 1.0; A(J+1,J+1) = 1.0;
for j=2:J
    A(j,[j-1, j, j+1]) = [1, -2, 1];
end
```

- solving the matrix problem looks like:

```matlab
Y = A \ b; % solve A Y = b
```

- plot on next page from

```matlab
% also get exact soln on fine grid:
xf = 0:1/1000:1; yexact = xf.^4 - xf;
plot(x,Y,'o','markersize',12,xf,yexact)
grid on, xlabel x, legend('finite diff','exact')
```
toy example: as matrix problem in OCTAVE, cont, cont

- gives result which is better than we have any reason to expect:
Two-point Boundary Value Problems: Numerical Approaches

Bueler

classical IVPs and BVPs
serious problem
finite difference
shooting
serious example: solved

toy example with finite differences: brief analysis

regarding the result on the previous slide:

• recall the exact solution is \( y(x) = x^4 - x \)

• and

\[
\frac{f(x - h) - 2f(x) + f(x + h)}{h^2} = f''(x) + \frac{f^{(4)}(\nu)}{12} h^2
\]

• applied to \( f(x) = y(x) \), for which \( y^{(4)}(x) = 24 \), we see that the finite difference approximation to the second derivative in the ODE \( y'' = 12x^2 \) has error at most

\[
\frac{y^{(4)}(\nu)}{12} \Delta x^2 = \frac{24}{12} (0.1)^2 = 0.02
\]

because \( \Delta x = 0.1 \)

• this is a rare case where the truncation error is known!
toy example with finite differences: brief analysis, cont

- let $e_j = Y_j - y(x_j)$, the *error* we care about
- by subtraction,

$$\frac{e_{j-1} - 2e_j + e_{j+1}}{\Delta x^2} = 0.02$$

and $e_0 = e_{J+1} = 0$

- so (after bit of not-too-hard thought)

$$e_j = 0.01x_j(x_j - 1)$$

- so

$$\max_j |Y_j - y(x_j)| = \max_j |e_j| = 0.0025$$

- which explains why the picture a few slides back was good
  … but showed slight errors close to screen resolution
Outline

1. classical IVPs and BVPs with by-hand solutions
2. a serious problem: a BVP for equilibrium heat
3. finite difference solution of two-point BVPs
4. shooting to solve two-point BVPs
5. a more serious example: solutions
toy example problem again: shooting

• recall this “toy” ODE BVP:

\[ y'' = 12x^2, \quad y(0) = 0, \quad y(1) = 0 \]

which has exact solution \( y(x) = x^4 - x \)

• this time we think: *if only it were an ODE IVP then we could apply a numerical ODE solver like MATLAB’s ode45*

• indeed, this ODE IVP

\[ w'' = 12x^2, \quad w(0) = 0, \quad w'(0) = A \]

*can be solved by a numerical ODE solver, for any A*

• solving this ODE IVP involves “aiming” by guessing an initial slope \( w'(0) = A \)

• “hitting the target” is getting the desired boundary value \( w(1) = 0 \)

• “aiming” and “hitting the target” is *shooting*
for illustrating the method on this easy problem, I’ll skip using a numerical ODE solver because the ODE IVP

\[ w'' = 12x^2, \quad w(0) = 0, \quad w'(0) = A \]

has a solution we can get by-hand:

\[ w(x) = x^4 + Ax \]

plotting for \( A = -2.5, -1.5, -0.5, 0.5, 1.5 \) gives this figure:
toy example shooting, cont, cont

- we have “aimed” (by choosing $A$) and “shot” five times
- a “shot” is a computation of the solution to an ODE IVP
  - generally this would be a numerical solution
- on previous slide we missed every time
- but we have bracketed the correct right-hand boundary condition $y(1) = 0$ with the two values $A = -1.5$ and $A = -0.5$
- a numerical *equation* solver can refine the search to converge to the correct $A$ value
shooting: solving the boundary condition equation

- recall our ODE BVP
  \[ y'' = 12x^2, \quad y(0) = 0, \quad y(1) = 0 \]

  is replaced by this ODE IVP when “shooting”:
  \[ w'' = 12x^2, \quad w(0) = 0, \quad w'(0) = A \quad (8) \]

- the \( x = 1 \) endpoint value of \( w(x) \) is a function of \( A \):
  \[ F(A) = \left( w(1), \text{ where } w \text{ solves } (8) \right) \]

- and so we solve this equation because we want \( y(1) = 0 \):
  \[ F(A) = 0 \]

- in this easy problem, \( w(x) = x^4 + Ax \)
- so we solve \( F(A) = 1 + A = 0 \) and get \( A = -1 \)
- generally we solve \( F(A) = 0 \) numerically, e.g. by the \textit{bisection} or \textit{secant} methods
shooting: general strategy for two-point ODE BVPs

- identify one end of the interval \( x = b \) as the target
- at the other end \( x = a \), identify some additional initial conditions which would give a well-posed ODE IVP
- for various guesses of those additional initial conditions, “shoot” by solving the corresponding ODE IVP from \( x = a \) to \( x = b \)
- ask whether you “hit the target” by asking whether the boundary conditions at \( x = b \) are satisfied
- automate the adjustment process by using an equation solver (e.g. bisection or secant method) on the equation that says “the discrepancy between the solution of the ODE IVP at \( x = b \) and the desired boundary conditions at \( x = b \), as a function of the additional initial condition \( A \), should be zero: \( F(A) = 0 \)”
Outline

1. classical IVPs and BVPs with by-hand solutions
2. a serious problem: a BVP for equilibrium heat
3. finite difference solution of two-point BVPs
4. shooting to solve two-point BVPs
5. a more serious example: solutions
recall the serious example

- recall the "serious" non-constant-coefficient BVP:

\[
(k(x)u')' + r_0 u = -s(x), \quad u'(0) = 0, \quad u(3) = 0, \quad (9)
\]

- \(u(x)\) is the equilibrium temperature in a rod
- the conductivity \(k(x)\) has a big jump at \(x = 1\) and the heat source \(s(x)\) is concentrated near \(x = 2\):

![Graph of k(x), r(x)=r_0, s(x) over x from 0 to 3]
finite differences: need staggered grid

- finite difference approach first
- as before: \( J \) subintervals, \( \Delta x = 1/J \), and

\[
x_j = (j - 1) \Delta x \quad \text{for } j = 1, \ldots, J + 1
\]

- let \( U_j \) be our finite diff. approx. to \( u(x_j) \)
- let \( k_j = k(x_j) \) and \( s_j = s(x_j) \); we know these exactly
- note: if \( q(x) = -k(x)u'(x) \), i.e. Fourier’s law for heat flow, then we are solving

\[
-q' + r_0 u = -s(x)
\]

- the finite difference version looks like

\[
- \frac{q_{j+1/2} - q_{j-1/2}}{\Delta x} + r_0 U_j = -s(x_j)
\]

- or

\[
\frac{k(x_{j+1/2})}{\Delta x} \frac{U_{j+1} - U_j}{\Delta x} - \frac{k(x_{j-1/2})}{\Delta x} \frac{U_j - U_{j-1}}{\Delta x} + r_0 U_j = -s(x_j)
\]
finite differences: need staggered grid, cont

- or

\[
\frac{k_{j+\frac{1}{2}} (U_{j+1} - U_j) - k_{j-\frac{1}{2}} (U_j - U_{j-1})}{\Delta x^2} + r_0 U_j = -s_j
\]

- or (clear denominators)

\[
k_{j+\frac{1}{2}} (U_{j+1} - U_j) - k_{j-\frac{1}{2}} (U_j - U_{j-1}) + r_0 \Delta x^2 U_j = -s_j \Delta x^2
\]

- or

\[
k_{j-\frac{1}{2}} U_{j-1} - \left( k_{j-\frac{1}{2}} + k_{j+\frac{1}{2}} - r_0 \Delta x^2 \right) U_j + k_{j+\frac{1}{2}} U_{j+1} = -s_j \Delta x^2
\]

- like the “toy” example earlier, this last form is a tridiagonal matrix equation \( A \mathbf{U} = \mathbf{b} \)

- note we evaluate the conductivity \( k(x) \), and the flux \( q \), on the staggered grid (i.e. \( x_{j+\frac{1}{2}} \) and \( x_{j-\frac{1}{2}} \))

- the deeper reason why we use the staggered grid will be revealed later in class . . .
finite differences: remember the boundary conditions

- recall we have boundary condition $u'(0) = 0$
- approximate this by
  $$\frac{U_2 - U_1}{\Delta x} = 0$$
- or
  $$-U_1 + U_2 = 0$$
- we will see there is a more-accurate way later . . .
- also we have $u(L) = 0$ so
  $$U_{J+1} = 0$$
finite differences for the “serious problem”

- now for an actual code: see varheatFD.m online
- the ODE setup:

```matlab
L = 3;
k = @(x) 0.5 * atan((x-1.0) * 20.0) + 1.0;
s = @(x) exp(-(x-2.0).^2);
r0 = 0.5;

dx = L / J;
x = (0:dx:L)'; % regular grid
xstag = ((dx/2):dx:L-(dx/2))'; % staggered grid
kstag = k(xstag); % k(x) on staggered grid

% right side is J+1 length column vector
b = [0; - dx^2 * s(x(2:J)); 0];

% matrix is tridiagonal
A = sparse(J+1,J+1);
A(1,[1 2]) = [-1.0 1.0];
for j=1:J-1
    A(j+1,j) = kstag(j);
    A(j+1,j+1) = - kstag(j) - kstag(j+1) + r0 * dx^2;
    A(j+1,j+2) = kstag(j+1);
end
A(J+1,J+1) = 1.0;
```
finite differences for the “serious problem”, cont

- it is good to use \( \text{spy}(A) \) at this point to see the matrix structure; this is the \( J = 10 \) case

nz = 30
finite differences for the “serious problem”, cont, cont

- the matrix solve:
  \[ U = A \backslash b; \quad \% \text{soln is } J+1 \text{ column vector} \]

- the plot details:
  ```matlab
  figure(1)
  plot(x,k(x),'r',x,s(x),'b',... 
    x, U','g*','markersize',3)
  grid on, xlabel x
  legend('k(x)','s(x)','solution U_j')
  ```
finite difference solution to “serious problem”

- the picture when $J = 60$:

![Graph showing finite difference solution with $u(0) = -5.666658$]
finite difference solution to “serious problem”, cont

- recall our concrete goal was to estimate $u(0)$
- clearly we should try different $J$ values to estimate:

<table>
<thead>
<tr>
<th>$J$</th>
<th>estimate of $u(0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>-13.86507</td>
</tr>
<tr>
<td>20</td>
<td>-7.20263</td>
</tr>
<tr>
<td>60</td>
<td>-5.66666</td>
</tr>
<tr>
<td>200</td>
<td>-5.27443</td>
</tr>
<tr>
<td>1000</td>
<td>-5.15199</td>
</tr>
<tr>
<td>4000</td>
<td>-5.12965</td>
</tr>
</tbody>
</table>

- this suggests that $u(0) \approx -5.13$
- *How do we know how wrong we are?*
shooting for the “serious problem”

- shooting is implemented in this code online:
  - varheatSHOOT.m

- the setup:
  
  ```matlab
  L = 3;
k = @(x) 0.5 * atan((x-1.0) * 20.0) + 1.0;
s = @(x) exp(-(x-2.0).^2);
r0 = 0.5;

  % ODE \( Y' = G(x,Y) \) is described by this right-hand side
  G = @(x,Y) [- Y(2) / k(x); % Y(1) = u
              r0 * Y(1) + s(x)]; % Y(2) = q

  % bracket unknown \( u(0) \)
a = -10.0; % produces \( u(3) \) which is too high
b = 0.0; % ... \( u(3) \) which is too low
  ```
shooting for the “serious problem”, cont

- the \textit{bisection} implementation, which starts from initial bracket \([a, b] = [-10.0, 0.0]\):

\begin{verbatim}
N = 100;
for n = 1:N
    c = (a+b)/2;
    [xout,Y] = ode45(G,[0.0 3.0],[c; 0.0]);
    F = Y(end,1);
    if abs(F) < 1e-12
        break % we are done
    elseif F >= 0.0
        a = c;
    else
        b = c;
    end
end
\end{verbatim}
shooting solution to “serious problem”

- the picture:

```
result of SHOOTING:  u(0) = -5.122257
```

- default use of `ode45` gives estimate \( u(0) = -5.122257 \)

- *How do we know how wrong we are?*
minimal conclusion

- finite difference and shooting methods give comparable solutions to this “serious problem”
- closer inspection of the programs above will help understand the methods
- better understanding will also follow from doing the exercises on Assignment # 5