1

Neumann and insulated boundary conditions

Introduction

- In this topic, we will
  - Review Neumann boundary conditions
  - Approximate derivatives using the formula from calculus
  - Author a solution in MATLAB using this approximation
  - Observe there are weaknesses in such an approach
  - Consider a superior approach

2

2

Dirichlet boundary conditions

- Up to this point, we've used Dirichlet boundary conditions:

$$u(a) = u_a$$
$$u(b) = u_b$$

- Recall that this affected the first and last equations:

$$p_1 u_0 + q_1 u_1 + r_1 u_2 = 2g(x_1)h^2$$

$$p_{n-1} u_{n-2} + q_{n-1} u_{n-1} + r_{n-1} u_n = 2g(x_{n-1})h^2$$

3

3

Neumann and insulated boundary conditions

- What happens if a boundary has an insulated or more generally a Neumann boundary condition?

$$u^{(1)}(a) = 0 \qquad u^{(1)}(a) = u_a^{(1)}$$
$$u^{(1)}(b) = 0 \qquad u^{(1)}(b) = u_b^{(1)}$$

4

4

Neumann and insulated boundary conditions

- Suppose we have a Neumann boundary condition at $x = a$:

$$u^{(1)}(a) = u_a^{(1)}$$

  – How do we eliminate the unknown u_0?

$$p_1 u_0 + q_1 u_1 + r_1 u_2 = 2g(x_1)h^2$$

  – If we don't eliminate it, we will have fewer equations than unknowns…

Approximating the derivative

- Recall we used a divided-difference approximation of the derivative:

$$\frac{u(a+h) - u(a)}{h} \approx u^{(1)}(a) = u_a^{(1)}$$

  – But $u(a) \approx u_0$ and $u(a + h) \approx u_1$:

$$\frac{u_1 - u_0}{h} \approx u_a^{(1)}$$

  – Thus, we have an equation

$$u_0 = u_1 - u_a^{(1)} h$$

Approximating the derivative

- Thus, if we have a left-hand Neumann condition, we have

$$p_1 u_0 + q_1 u_1 + r_1 u_2 = 2g(x_1)h^2$$

$$u_0 = u_1 - u_a^{(1)} h$$

  – Substituting the second into the first yields

$$p_1\left(u_1 - u_a^{(1)} h\right) + q_1 u_1 + r_1 u_2 = 2g(x_1)h^2$$

$$(p_1 + q_1)u_1 + r_1 u_2 = 2g(x_1)h^2 + p_1 u_a^{(1)} h$$

  – Thus, entry (1,1) of the matrix needs to be updated and entry 1 of the target vector needs to be updated

Approximating the derivative

- Similarly, we can find on the right-hand boundary value:

$$p_{n-1} u_{n-2} + q_{n-1} u_{n-1} + r_{n-1} u_n = 2g(x_{n-1})h^2$$

$$u_n = u_{n-1} + u_b^{(1)} h$$

  – This yields

$$p_{n-1} u_{n-2} + (q_{n-1} + r_{n-1})u_{n-1} = 2g(x_{n-1})h^2 - r_{n-1} u_b^{(1)} h$$

  – Thus, entry ($n-1$, $n-1$) of the matrix needs to be updated and entry $n-1$ of the target vector needs to be updated

Implementation

```
function [xs, us] = bvp( a2, a1, a0, g, x_rng, u_bndry, dirichlet, n )
    h = (x_rng(2) - x_rng(1))/n;

    p = @(x)( 2.0*a2(x) -      a1(x)*h   );
    q = @(x)(-4.0*a2(x) + 2.0*a0(x)*h^2);
    r = @(x)( 2.0*a2(x) +      a1(x)*h   );

    xs = linspace( x_rng(1) + h, x_rng(2) - h, n - 1 )';

    A = zeros( n - 1, n - 1 );

    for k = 1:(n - 1)
        A(k, k) = q(xs(k));
    end

    for k = 1:(n - 2)
        A(k + 1, k     ) = p(xs(k + 1));
        A(k,     k + 1) = r(xs(k));
    end
```


Implementation

```
    v = 2.0*g( xs )*h^2;

    if dirichlet( 1 )
        v(1)    = v(1)      - p(xs(1))*u_bndry(1);
    else
        A(1, 1) = A(1, 1) + p(xs(1));
        v(1)     = v(1)   + p(xs(1))*u_bndry(1)*h;
    end
```

$$\boxed{(p_1+q_1)u_1}+r_1u_2=2g(x_1)h^2\boxed{+p_1u_a^{(1)}h}$$

```
    if dirichlet( 2 )
        v(end)       = v(end)       - r(xs(end))*u_bndry(2);
    else
        A(end, end) = A(end, end) + r(xs(end));
        v(end)       = v(end)      - r(xs(end))*u_bndry(2)*h;
    end
```

$$p_{n-1}u_{n-2}+\boxed{(q_{n-1}+r_{n-1})u_{n-1}}=2g(x_{n-1})h^2\boxed{-r_{n-1}u_b^{(1)}h}$$

Implementation

```
us = A \ v;

xs = [x_rng(1); xs; x_rng(2)];

if dirichlet( 1 )
    us = [u_bndry(1); us];
else
    us = [us(1) - u_bndry(1)*h; us];
end

if dirichlet( 2 )
    us = [us; u_bndry(2)];
else
    us = [us; us(end) + u_bndry(2)*h];
end
end
```

$u_0 = u_1 - u_a^{(1)} h$

$u_n = u_{n-1} + u_b^{(1)} h$

Example

- Let us examine this BVP:

$$13x^2 u^{(2)}(x) - 5u^{(1)}(x) + 8xu(x) = \sin(x)$$

$$u^{(1)}(-1) = -0.4738221764482897$$

$$u(1) = 2$$

- If n = 10, then h = 0.2, so

$$\begin{aligned} p_k &= 2a_2(x_k) - a_1(x_k)h &&= 2\cdot 13x_k^2 - (-5)\cdot 0.2 \\ q_k &= -4a_2(x_k) + 2a_0(x_k)h^2 &&= -4\cdot 13x_k^2 + 2\cdot 8x\cdot 0.04 \\ r_k &= 2a_2(x_k) + a_1(x_k)h &&= 2\cdot 13x_k^2 + (-5)\cdot 0.2 \end{aligned}$$

- As before,
  the x-values are –1, –0.8, –0.6, –0.4, –0.2, 0, 0.2, 0.4, 0.6, 0.8, 1

Example

- We can call our function with the appropriate arguments:

```
>> [xs, us] = bvp( @(x)(x^2*13), @(x)(-5.0), @(x)(8*x),...
                   @sin, [-1 1], [-0.4738221764482897 2], ...
                   [false, true], 10 )
```


13

Example

- Solving this system of linear equations yields:

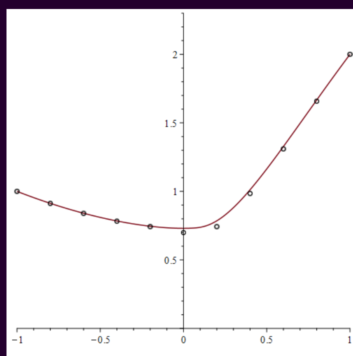
$$\mathbf{u} = \begin{array}{l} \begin{pmatrix} 0.975469169822180 \\ 0.896850697262553 \\ 0.835215756372538 \\ 0.792375715630198 \\ 0.745797266197350 \\ 0.792375715630198 \\ 1.029613008278919 \\ 1.343447491612049 \\ 1.676056679222552 \end{pmatrix} \end{array} \begin{array}{l} \\ \\ \\ \approx u(-0.2) \\ \\ \\ \\ \\ \approx u(0.8) \end{array}$$

$$u(1) = 2 \qquad u(-1) \approx 1.070233605111838$$

14

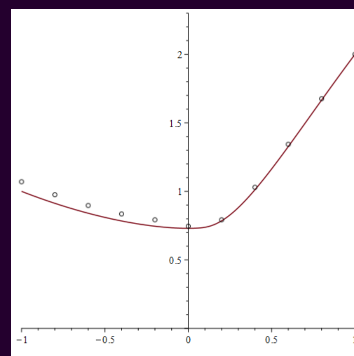
Example

- Here is a plot of the solution and the approximations:



$u(-1) = 1$

$u(1) = 2$



$u^{(1)}(-1) = -0.4738221764482897$

$u(1) = 2$

Error analysis

- Problem:
  - When we substituted the derivative and second derivative, we used O(h^2) approximations
  - When we approximated the derivatives, we used O(h) approximations

$$\frac{u(a+h) - u(a)}{h} \approx u^{(1)}(a) = u_a^{(1)}$$

  - Consequently, the overall error will now be O(h)

An O(h^2) approximation

- Recall we used a divided-difference approximation of the derivative:

$$\frac{-u_2+4u_1-3u_0}{2h}=u_a^{(1)} \qquad \frac{u_{n-2}-4u_{n-1}+3u_n}{2h}=u_b^{(1)}$$

$$u_0=-\frac{2}{3}u_a^{(1)}h+\frac{4}{3}u_1-\frac{1}{3}u_2 \qquad u_n=\frac{2}{3}u_b^{(1)}h+\frac{4}{3}u_{n-1}-\frac{1}{3}u_{n-2}$$

$$p_1u_0+q_1u_1+r_1u_2=2g(x_1)h^2 \qquad p_{n-1}u_{n-2}+q_{n-1}u_{n-1}+r_{n-1}u_n=2g(x_{n-1})h^2$$

$$\left(q_1+\frac{4}{3}p_1\right)u_1+\left(r_1-\frac{1}{3}p_1\right)u_2=2g(x_1)h^2+\frac{2}{3}p_1u_a^{(1)}h$$

$$\left(p_{n-1}-\frac{1}{3}r_{n-1}\right)u_{n-2}+\left(q_{n-1}+\frac{4}{3}r_{n-1}\right)u_{n-1}=2g(x_{n-1})h^2-\frac{2}{3}r_{n-1}u_b^{(1)}h$$

Implementation

```
function [xs, us] = bvp( a2, a1, a0, g, x_rng, u_bndry, dirichlet, n )
    h = (x_rng(2) - x_rng(1))/n;

    p = @(x)( 2.0*a2(x) -     a1(x)*h  );
    q = @(x)(-4.0*a2(x) + 2.0*a0(x)*h^2);
    r = @(x)( 2.0*a2(x) +     a1(x)*h  );

    xs = linspace( x_rng(1) + h, x_rng(2) - h, n - 1 )';

    A = zeros( n - 1, n - 1 );

    for k = 1:(n - 1)
        A(k, k) = q(xs(k));
    end

    for k = 1:(n - 2)
        A(k + 1, k    ) = p(xs(k + 1));
        A(k,     k + 1) = r(xs(k));
    end
```

Neumann and insulated boundary conditions

Implementation

```
v = 2.0*g( xs )*h^2;

if dirichlet( 1 )
    v(1)    = v(1)    - p(xs(1)  )*u_bndry(1);
else
    A(1, 1) = A(1, 1) + (4.0/3.0)*p(xs(1));
    A(1, 2) = A(1, 2) - (1.0/3.0)*p(xs(1));
    v(1)     = v(1)     + (2.0/3.0)*p(xs(1))*u_bndry(1)*h;
end

if dirichlet( 2 )
    v(end) = v(end) - r(xs(end))*u_bndry(2);
else
    A(end, end-1) = A(end, end-1) - (1.0/3.0)*r(xs(end));
    A(end, end)    = A(end, end)    + (4.0/3.0)*r(xs(end));
    v(end)          = v(end)          - (2.0/3.0)*r(xs(end))*u_bndry(2)*h;
end
```

19

19

Neumann and insulated boundary conditions

Implementation

```
    us = A \ v;

    xs = [x_rng(1); xs; x_rng(2)];

    if dirichlet( 1 )
        us = [u_bndry(1); us];
    else
        us = [(-1.0/3.0)*us(2) + (4.0/3.0)*us(1) ...
                                 - (2.0/3.0)*u_bndry(1)*h; us];
    end

    if dirichlet( 2 )
        us = [us; u_bndry(2)];
    else
        us = [us; (-1.0/3.0)*us(end-1) + (4.0/3.0)*us(end) ...
                                         + (2.0/3.0)*u_bndry(2)*h];
    end
end
```

20

20

Example

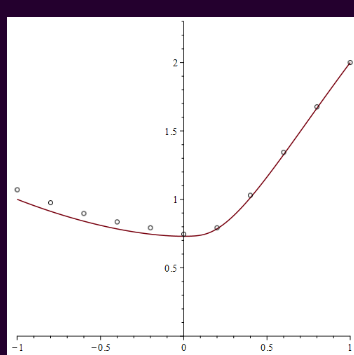
- Solving this system of linear equations yields:

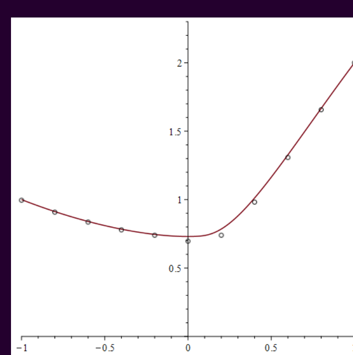
$$\mathbf{u} = \begin{pmatrix} 0.908106526012137 \\ 0.835681556241632 \\ 0.778912134340048 \\ 0.739455748895171 \\ 0.696099826080860 \\ 0.739455748895171 \\ 0.981688077550621 \\ 1.307561538118296 \\ 1.656738191074670 \end{pmatrix}$$

$$u(1) = 2 \qquad u(-1) \approx 0.995424472795410$$

Example

- The O(h) approximation is on the left,
  the O(h^2) approximation is on the right





Insulated boundary conditions

- Recall that insulated boundary conditions are when the derivatives are zero
  - Consequently, only the matrix is modified, as the change to the target vector is zero

$$\left(q_1+\frac{4}{3}p_1\right)u_1+\left(r_1-\frac{1}{3}p_1\right)u_2=2g\left(x_1\right)h^2+\cancelto{0}{\frac{2}{3}p_1u_a^{(1)}h}$$

$$\left(p_{n-1}-\frac{1}{3}r_{n-1}\right)u_{n-2}+\left(q_{n-1}+\frac{4}{3}r_{n-1}\right)u_{n-1}=2g\left(x_{n-1}\right)h^2-\cancelto{0}{\frac{2}{3}r_{n-1}u_b^{(1)}h}$$

Summary

- Following this topic, you now
  - Understand better what Neumann conditions are
  - Understand that better approximations cannot compensate for poorer approximations
  - Know how to approximation a BVP with Neumann conditions
  - Have gone through an implementations in MATLAB
  - Have seen an example

Neumann and insulated boundary conditions

References

[1] https://en.wikipedia.org/wiki/Finite_difference_method

25

25

Neumann and insulated boundary conditions

Acknowledgments

None so far.

26

26

Colophon

These slides were prepared using the Cambria typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

Mathematical equations are prepared in MathType by Design Science, Inc.

Examples may be formulated and checked using Maple by Maplesoft, Inc.

The photographs of flowers and a monarch butter appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens in October of 2017 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

27

27

Disclaimer

These slides are provided for the ECE 204 *Numerical methods* course taught at the University of Waterloo. The material in it reflects the author's best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.

28

28