Lecture 23: Starting to put it all together #2...
More 2-Point Boundary value problems

Outline

1) Our basic example again:
\[-u'' + u = f(x); \quad u(0)=\alpha, \ u(L)=\beta\]

2) Solution of 2-point Boundary value problems
   A) Shooting Methods
   B) Finite-Difference methods
   C) Higher-order: Spectral - Collocation methods
   D) Variation #2: Non-linear problems
   E) Galerkin Finite Elements

An Example problem: Simplest Steady-state 2-point boundary value problem
Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #2: Finite Difference Schemes

Finite Differences and Linear algebra: Boundary conditions

Matlab implementation:

```matlab
function [ D1, D2, x ] = centerDifference(N, xBounds)
% CenterDifference - generates 2nd order sparse Difference matrices and coordinates for
% N evenly spaced points in the domain xBounds = [xMin xMax]
% 
% [ D1, D2, x ] = centerDifference(N, xBounds)
% 
% D1: first derivative Matrix 1/2h [-1 0 1];
% D2: second derivative matrix 1/h^2 [1 -2 1]
% x: uniform grid
% 
% N: number of grid points
% xBounds: array holding coordinates of domain [xMin xMax]
% 
% % generate uniform grid
% x = linspace(xBounds(1), xBounds(2), N);
% x=x(:);
% h = x(2)-x(1);
% 
% % generate sparse Difference matrices
% e = ones(N,1);
% D1=spsdiags([-e e], [-1 1], N, N)/2/h; % First derivative matrix
% D2=spsdiags([-2*e e], [-1 1], N, N)/h^2; % second derivative matrix
% 
% % fix boundary points for consistent 1-sided derivatives
% D1(1,1:3) = [-3 4 -1]/2/h;
% D1(end,end-2:end) = [1 -4 3]/2/h;
% 
% D2(1,1:3) = [1 -2 1]/h/h;
% D2(end,end-2:end) = [1 -2 1]/h/h;
```
Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #2: Finite Difference Schemes

Matlab implementation:

```matlab
function [A,x] = finiteDifferenceOperator(N,xBounds,bcType)
% FINITEDIFFERENCEOPERATOR - Calculates the sparse difference matrix
% for -u''+u with N points and variable boundary conditions on the
% domain x in [xMin xMax]
%
% A = finiteDifferenceOperator(N,xBounds,bcType)
% N: number of grid points in domain
% xBounds: array with [xMin xMax] defining x domain
% bcType: array with flags for boundary condition types for left and right
% edges
% A: sparse matrix form of I-d_xx (uses SPDIAGS)
% x: vector of evenly spaced grid points
%
% get grid and centered difference matrices
[D1,D2,x] = centerDifference(N,xBounds);

% clear first and last rows of D2 for dirichlet BC's
D2([ 1 end],:) = zeros(2,N);

% set sparse finite difference operator for -u''+u = -D2+I
A = speye(N) - D2;

% correct A for Neumann BC's
if (bcType(1) == 1)
A(1,:) = D1(1,:);
end
if (bcType(2) == 1);
A(end,:) = D1(end,:);
end
```

```matlab
function [u,x] = solveBVPfd(N,xBounds,func,alpha,beta,bcType)
% get the differential operator matrix and grid
[A,x] = finiteDifferenceOperator(N,xBounds,bcType);

% evaluate the rhs function
f = func(x);
f = f(:); % convert to column vector

% set boundary conditions
f(1)=alpha;
f(end)=beta;

% solve the linear system
u=A;
```
Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #2: Finite Difference Schemes

Finite Differences: pros and cons

Pros:
- Easy to implement (and debug)...good for quick and dirty solutions
- Easy to modify
- Relatively easy to add non-linearities
- Fast given sparse linear solvers

Cons:
- low order FD requires significant number of points for accuracy
- 2nd order accuracy only guaranteed for uniform mesh

Solving 2-pt ODE Boundary Value Problems (BVPs)

Convergence of 2nd-order FD schemes

![Graph showing error behavior for 2nd order finite difference schemes](image)
Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #3: Spectral Collocation Schemes: Increasing accuracy
General Collocation method

Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #3: Spectral Collocation Schemes: Polynomial Basis on fixed interval \([a b]\)
Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #3: Spectral Collocation Schemes: Polynomial Basis on fixed interval \([ a b ]\)

```matlab
function [A,x] = chebDifferenceOperator(N,xBounds)
% chebDifferenceOperator - Calculates the difference matrix
% for \(-u''+u\) with N points and dirichlet BC's on the
% domain x in \([xMin xMax]\)
% using the chebyshev function basis collocation method
% and returns it as a sparse matrix A=I - D*D
%
% [A,x] = chebDifferenceOperator(N)
%
% N total number of chebyshev points
% xBounds: array with domain xBounds = \([ xMin xMax]\)
%
% A - sparse matrix form of I-d_xx
% x - chebyshev points on xBounds = \([ xMin,xMax]\)

% get the chebyshev difference matrix and coordinates in the domain
% y=[-1 1];
[Dy,y]=cheb(N-1);

% convert to domain x = \([xMin xMax]\)
H=xBounds(2)-xBounds(1);  % domain size
x=xBounds(1) + H/2*(y+1);  % chebyshev points in x coordinate frame
D2=(2/H)^2*Dy*Dy;  % 2nd Derivative matrix in x coordinate frame

% calculate full linear operator with Dirichlet Boundary conditions
D2([1 end],[1]) = zeros(2,N);  % zero out top and bottom rows
A=-D2 +speye(N);  % Linear operator for \(-u''+u\)
```
Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #3: Spectral Collocation Schemes: Matlab Implementation

function [u,x] = solveBVPcheb(N,xBounds,func,alpha,beta)
% function [u,x]= solveBVPcheb(N,xBounds,f,alpha,beta)
% SOLVEBVPcheb - Solves the ODE Boundary value problem
% -u'' + u = f(x), u(x_0)=alpha, u(x_N)=beta
%
% uses N Chebyshev points on the interval xBounds = [ xMin xMax ]
%
% N - Number of chebyshev points (uses chebyshev polynomials up to order N-1
% xBounds - array with coordinates of left and right boundaries [xMin xMax]
% func - function handle to rhs f(x)
% alpha - left Boundary condition
% beta  - right Boundary condition
%
% u - the solution u(x)
% x - the chebyshev grid

% get the linear operator and chebyshev points
[A,x] = chebDifferenceOperator(N,xBounds);

% evaluate the rhs function
f=func(x);
f=f(:);  % convert to column vector

% set boundary conditions, need to be careful as x is reversed in cheb
f(x==xBounds(1))=alpha;
f(x==xBounds(2))=beta;

% solve the linear system
u=A;

Solving 2-pt ODE Boundary Value Problems (BVPs)
Method #3: Spectral Collocation Schemes: Increasing accuracy
Solving 2-pt ODE Boundary Value Problems (BVPs)

Theme and variation #2: adding non-linearity

Let's change the radiation model to a generalized black-body radiation:

Method #1: Shooting --easy (but sensitive)
Solving 2-pt ODE Boundary Value Problems (BVPs)

Theme and variation #2: adding non-linearity

Method #2: Non-linear finite difference: still replace differential operators with their discrete approximations, however the resulting equations are no longer linear

Solving 2-pt ODE Boundary Value Problems (BVPs)

Method #2: Non-linear finite difference: Newton's method