CS 450: Numerical Anlaysis

Lecture 26

Chapter 10 Boundary Value Problems for Ordinary Differential Equations

Numerical Methods for Boundary Value Problems

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

April 20, 2018

Finite Difference Methods

- Lets derive the finite difference method for the ODE BVP defined by

$$u'' + 1000(1+t^2)u = 0$$

with boundary conditions $u(-1) = 3$ and $u(1) = -3$.

$u_i \approx u(t_i)$, for $t_1 \dots t_n$ | $t_0 = -1$, $t_n = -3$

centered F-D approximation for u''

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + 1000(1+t_i^2)\,u_i = 0$$

$$\begin{array}{c} 1 \\ \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & & \ddots & \end{bmatrix} \\ \quad\quad\quad 1 \end{array} \begin{array}{c} u_1 \\ \begin{bmatrix} u_2 \\ \vdots \\ u_{n-1} \end{bmatrix} \\ u_n \end{array} + \begin{bmatrix} 1000(1+t_2) & & \\ & \ddots & \\ & & 1000(1+t_{n-1}) \end{bmatrix} \begin{bmatrix} u_2 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{array}{c} -3 \\ \begin{bmatrix} \\ 0 \\ \\ \end{bmatrix} \\ 3 \end{array}$$

Collocation Methods

- *Collocation methods* approximate $\boldsymbol{y}$ by representing it in a basis

$$\boldsymbol{y}(t) = \boldsymbol{v}(t, \boldsymbol{x}) = \sum_{i=1}^{n} x_i \boldsymbol{\phi}_i(t).$$

$t_1, \dots, t_n$

$v'(t_i, x_i) = f(t_i, v(t_i, x_i))$

ensure that each ϕ_i satisfies BCs $\xRightarrow{\text{often}}$ $y(t)$ satisfies BCs

- *Spectral methods* use polynomials or trigonometric functions for ϕ_i, which are nonzero over most of $[a, b]$, while *finite element* methods leverage basis functions with local support (e.g. B-splines).

spectral $\Rightarrow$ polynomials for ϕ_i, (spectral) - eigenfunctions of differential operators

finite element (FEM) $\Rightarrow$ B-splines (local support)

Solving BVPs by Optimization

- We reformulate the collocation approximation as an optimization problem:

$$r(t_i, x) = v'(t_i, x) - f(t_i, v(t_i, x))$$

$$= \sum_{j=1}^{n} x_j \varphi_j'(t) - f(t_i, v(t_i, x))$$

$$F(x) = \frac{1}{2} \int_a^b \|r(\cdot, x)\|_2^2 \, dt$$

- The first-order optimality conditions of the optimization problem are a system of linear equations $Ax = b$:

$$f(t, y) = f(t), \quad r(t_i, x) = \sum_{j=1} x_j \varphi_j'(t) - f(t)$$

$$0 = \frac{\partial F(x)}{\partial x_i} = \int_a^b r(t, x)^T \frac{\partial r}{\partial x}(t, x) \, dt$$

$$= \int_a^b r(t, x)^T \varphi_i'(t) \, dt = \sum_j x_j \underbrace{\int \varphi_j'(t) \varphi_i'(t) \, dt}_{A_{ij}} - \underbrace{\int_a^b f(t) \varphi_i'(t) \, dt}_{b_i}$$

$$Ax = b$$

Weighted Residual

- *Weighted residual methods* work by ensuring the residual is orthogonal with respect to a given set of weight functions: $\varphi_1(t) \dots \varphi_n(t)$

test

$$f(t,y) = f(t)$$

$$r(t,y) = \sum_j x_j \varphi_j(t) - f(t)$$

$$0 = \int_a^b r(t,x)^T \varphi_i(t)\,dt$$

$$0 = \sum_j x_j \int_a^b \varphi_j(t)\,\varphi_i(t)\,dt - \int_a^b f(t)\,\varphi_i(t)\,dt$$

- The Galerkin method is a weighted residual method where $w_i = \phi_i$.

$$0 = \sum_j x_j \underbrace{\int_a^b \varphi_j(t)\,\varphi_i(t)}_{a_{ij}} - \underbrace{\int_a^b f(t)\,\varphi_i(t)}_{b_i}$$

$$Ax = b$$

Linear BVPs by Optimization

- Lets apply the Galerkin method to the more general linear ODE $\boldsymbol{f}(t,y) = \boldsymbol{A}(t)\boldsymbol{y}(t) + \boldsymbol{b}(t)$ with residual equation,

$$r = v' - f = v' - Av - b$$

$$r(t,x) = \sum_j x_j \varphi_j'(t) - A \sum_j x_j \varphi_j(t) - b(t)$$

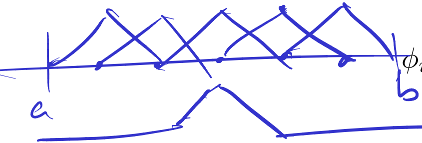
$$= \sum_j x_j \left(\varphi_j' - A \varphi_j(t) \right) - b(t)$$

$$0 = \sum_j x_j \underbrace{\int_a^b \left(\varphi_j' - A \varphi_j(t) \right) \varphi_i(t)\, dt}_{A_{ij}} - \underbrace{\int_a^b b(t) \varphi_i(t)\, dt}_{b_i}$$

Nonlinear BVPs: Poisson Equation

In practice, BVPs are at least second order and its advantageous to work in the natural set of variables.

- Consider the Poission equation $u'' = f(t)$ with boundary conditions $u(a) = u(b) = 0$ and define a localized basis of hat functions:



$$\phi_i(t) = \begin{cases} (t - t_{i-1})/h & : t \in [t_{i-1}, t_i] \\ (t_{i+1} - t)/h & : t \in [t_i, t_{i+1}] \\ 0 & : \text{otherwise} \end{cases}$$

linear FEM

linear B-splines

where $t_0 = t_1 = a$ and $t_{n+1} = t_n = b$.

$r = v'' - f \Rightarrow$

undefined

$r(t, x) = \sum_j x_j \varphi_j''(x) - f(t)$

Weak Form and the Finite Element Method

perform an optimization over once-differentiable functions to find twice-differentiable solution

- The finite-element method permits a lesser degree of differentiability of basis functions by casting the ODE in *weak form*:

If φ_i satisfies boundary conditions

$$\underbrace{\int_c^b f(t)\varphi_i(t)\,dt}_{b_i} = \int_a^b u''(t)\varphi_i(t)\,dt$$

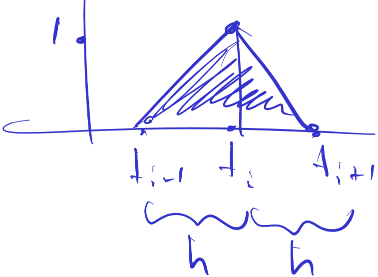
$$= u'(b)\overbrace{\varphi_i(b)}^{0} - u'(a)\overbrace{\varphi_i(a)}^{0} - \int_c^b u'(t)\varphi_i'(t)\,dt$$

$$\underline{u'(t)\varphi_i(t)\Big[_a^b}$$

$$u'(t) \approx v'(t) = \sum_j x_j \varphi_j'(t)$$

$$= -\int_c^b u'(t)\varphi_i'(t)\,dt \approx \sum_j x_j \overbrace{\int_c^b -\varphi_j'(t)\varphi_i'(t)\,dt}^{A_{ij}}$$

$$\underbrace{\int_a^b \varphi_i(t)\, f(t)\, dt}_{\text{if } f(t) = f_i \,\mid\, hf} = -\sum_j x_j \underbrace{\int_a^b \varphi_j'(t)\, \varphi_i'(t)\, dt}_{-a_{ij}}$$



$$\varphi_i(t) = \begin{cases} (t - t_{i-1})/h : & t \in [t_{i-1}, t_i] \\ (t_{i+1} - t)/h : & t \in [t_i, t_{i+1}] \\ 0 : & \text{otherwise} \end{cases}$$

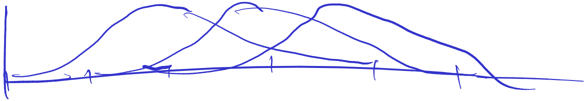
$$\varphi_i'(t) = \begin{cases} 1/h : & \\ -1/h : & \dots \\ 0 : & \end{cases}$$

$$A_{ij} = \int_a^b \varphi_j'(t)\, \varphi_i'(t)\, dt = \begin{cases} 2/h : & i = j \\ -1/h : & |i - j| = 1 \\ 0 : & \text{otherwise} \end{cases}$$

Finite Element Methods in Practice

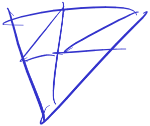
- Hat functions are linear instances of *B-splines*:

degree k, k-times differentiable



- Finite-element methods readily generalize to PDEs:

FEM with triangles tetrahedra



Eigenvalue Problems with ODEs

- A typical second-order scalar BVP eigenvalue problem has the form

$$u'' = \lambda f(t, u, u'), \quad \text{with boundary conditions } u(a) = 0, u(b) = 0$$

$$f(t, u, u') = g(t)\, u$$

u_i for $i = 1, \dots, n$ $\quad u_i \approx u(t_i)$

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = \lambda g_i u_i \Rightarrow \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2 g_i} = \lambda u_i$$

$$\begin{bmatrix} \ddots & \ddots & \\ & \ddots & \ddots \end{bmatrix} u = \lambda u$$

Eigenvalue Problems with ODEs

- Generalized eigenvalue problems arise from more sophisticated ODEs,

$$u'' = \lambda\underbrace{(g(t)u + h(t)u')}_{F(t,\,u,\,u')}, \quad \text{with boundary conditions } u(a) = 0, u(b) = 0$$

$g(t_i)$ $\quad$ $h(t_i)$

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = \lambda\left(g_i u_i + h_i \frac{u_{i+1} - u_{i-1}}{2h}\right)$$

$$Au = \lambda Bu$$