Supplementary Notes 11

Tiago Salvador (tiago.saldanhasalvador@mail.mcgill.ca)

Abstract

We discuss numerical methods for boundary value problems.

Contents

1 Numerical methods for BVP

We are interested in solving second order boundary value problems (BVP) of the form

$$u'' = f(x, u, u'), \quad x \in [a, b]$$

together with appropriate boundary conditions. We can have

- Dirichlet boundary conditions: $u(a) = \alpha$, $u(b) = \beta$.
- Neumann boundary conditions: $u'(a) = \alpha$, $u'(b) = \beta$.
- Mixed boundary conditions: $c_a u(a) + d_a u'(a) = \alpha$, $c_b u(b) + d_b u'(b) = \beta$

We will study two types of methods: shooting methods and finite difference methods.

2 Shooting method for second order BVP

The main idea is to solve the BVP like an initial value problem (IVP) which we already know how to solve. Consider then the BVP on interval $[a, b]$ given by

$$\begin{cases} u'' = f(x, u, u') \\ u(a) = \alpha \\ u(b) = \beta \end{cases} \tag{1}$$

We assume (1) has a unique solution which we denote by $u(x)$. Consider as well

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y'(a) = s \end{cases} \tag{2}$$

where $s \in \mathbb{R}$ is an (unknown) parameter. For a fixed $s \in \mathbb{R}$, denote by $y(x; s)$ the solution of (2).

Suppose s^* is such that $y(b; s^*) = \beta$, then $y(x; s^*)$ solves (1) and, since we assume the solution of (1) is unique, we have $u(x) = y(x; s^*)$.

The goal is then to find the correct slope s^*, i.e., s^* such that $y(b; s^*) = \beta$. Once we find s^* we can solve (2) for $s = s^*$ and obtain $y(x; s^*)$ which we will also be the solution of (1).

Given s we already know how to (numerically) solve (2): we can write it as first order system and solve it with one of the numerical methods for IVPs discussed before (e.g. forward Euler). The only thing we have left to do is to find s^* which can be accomplished with a root finding method: s^* is the root of $\phi(s)$ where $\phi(s) = y(b; s) - \beta$.

Remark 2.1. *We always $s^* = u'(a)$. However, this formula is not helpful since we it requires knowing u, which is precisely what we are trying to find.*

Example 2.1. *Consider the following problem*

$$\begin{cases} u'' = 0 \\ u(0) = 0 \\ u(1) = 1 \end{cases}$$

The exact solution is given by $u(x) = x$. The initial value problem we are interested is

$$\begin{cases} y'' = 0 \\ y(0) = 0 \\ y'(0) = s \end{cases}$$

We have $y(x; s) = sx$. In this case $s^* = 1$.

2.1 Shooting method with secant method

One way to find s^* is to use the secant method. For initial guesses of slopes s_0, s_1, the secant method gives the sequence

$$s_{k+1} = s_k - \frac{\phi(s_k)(s_k - s_{k-1})}{\phi(s_k) - \phi(s_{k-1})}.$$

For each k, we need to compute $\phi(s_k)$, i.e., we need to compute $y(b; s_k)$. Hence, for each s_k, we will approximate $y(x; s_k)$ by solving (2) with $s = s_k$ using any numerical method for IVPs. Denote by $\left\{y_0^{(k)}, \ldots, y_N^{(k)}\right\}$ the numerical solution of (2) with $s = s_k$. Hence we have

$$\phi(s_k) \approx y_N^{(k)} - \beta.$$

Remark 2.2. *To solve* (2) *numerically, we will need to write* (2) *as a first oder system.*

Example 2.2. *Consider the following problem*

$$\begin{cases} u'' = 2u^3, \\ u(-1) = \frac{1}{2}, \\ u(0) = \frac{1}{3}. \end{cases}$$

The initial value problem we are interested is

$$\begin{cases} y'' = 2y^3, \\ y(-1) = \frac{1}{2}, \\ y'(-1) = s. \end{cases}$$

Algorithm 1 Shooting method with secant method

1: Pick two guesses s_0, s_1 and N, *tol*.
2: Compute $\left\{y_0^{(0)}, \ldots, y_N^{(0)}\right\}$ and $\left\{y_0^{(1)}, \ldots, y_N^{(1)}\right\}$.
3: Set $k = 1$.
4: **while** $\left|y_N^{(k)} - \beta\right| < tol$ **do**
5: Set $s_{k+1} = s_k - \frac{y_N^{(k)} - \beta}{y_N^{(k)} - y_N^{(k-1)}}(s_k - s_{k-1})$.
6: Compute $\left\{y_0^{(k)}, \ldots, y_N^{(k)}\right\}$.
7: Set $k = k + 1$.
8: **end while**
9: Output $\left\{y_0^{(k)}, \ldots, y_N^{(k)}\right\}$.

To solve it we need to transform it into a first order system. Let $y_1(x) = y(x)$ *and* $y_2(x) = y'(x)$. *Then*

$$y_1' = y' = y_2 \quad \textit{and} \quad y_2' = y'' = 2y^3 = 2y_1^3$$

We then have the first order system $\vec{y}\,' = F(x, \vec{y})$ *with* $\vec{y}(0) = \begin{bmatrix}\frac{1}{2} & s\end{bmatrix}^T$ *where*

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad F(x, \vec{y}) = \begin{bmatrix} y_2 \\ 2y_1^3 \end{bmatrix}.$$

2.2 Shooting method with Newton's method

For an initial guess of slope s_0, Newton's method gives the sequence

$$s_{k+1} = s_k - \frac{\phi(s_k)}{\phi'(s_k)}$$

For each k, we need to compute $\phi'(s_k)$, in addition to $\phi(s_k)$. Define $z(x; s) := \frac{\partial y(x;s)}{\partial s}$. One show that z solves the equation

$$z'' = f_y(x, y, y')z + f_{y'}(x, y, y')z'$$

with $z(a) = 0$ and $z'(a) = 1$. Here f_y and $f_{y'}$ denote the partial derivatives of f with respect to its second and third argument, respectively.

Thus for each s we are interested in

$$\begin{cases} y'' = f(x, y, y') \\ z'' = f_y(x, y, y')z + f_{y'}(x, y, y')z' \\ y(a) = \alpha \\ y'(a) = s \\ z(a) = 0 \\ z'(a) = 1 \end{cases} \tag{3}$$

Denote by $\left\{y_0^{(k)}, \ldots, y_N^{(k)}\right\}$, $\left\{z_0^{(k)}, \ldots, z_N^{(k)}\right\}$ the numerical solution of (3) with $s = s_k$.

Example 2.3. *Consider the following problem*

$$\begin{cases} u'' = u' cos(x) - u \log(u), \\ u(-1) = \frac{1}{2}, \\ u(0) = \frac{1}{3}. \end{cases}$$

Algorithm 2 Shooting method with Newton's method

1: Pick two guesses s_0, s_1 and N, *tol*.
2: Compute $\left\{y_0^{(0)}, \ldots, y_N^{(0)}\right\}$ and $\left\{z_0^{(0)}, \ldots, z_N^{(0)}\right\}$.
3: Set $k = 1$.
4: **while** $\left|y_N^{(k)} - \beta\right| < tol$ **do**
5: Set $s_{k+1} = s_k - \frac{y_N^{(k)} - \beta}{z_N^{(k)}}$.
6: Compute $\left\{y_0^{(k)}, \ldots, y_N^{(k)}\right\}$, $\left\{z_0^{(k)}, \ldots, z_N^{(k)}\right\}$.
7: Set $k = k + 1$.
8: **end while**
9: Output $\left\{y_0^{(k)}, \ldots, y_N^{(k)}\right\}$.

The initial value problem we are interested is

$$\begin{cases} y'' = y' cos(x) - y\log(y), \\ z'' = -(\log(y) + 1)z + \cos(x)z' \\ y(-1) = \frac{1}{2}, \\ y'(-1) = s. \\ z(0) = 0 \\ z'(0) = 1 \end{cases}$$

since

$$f_y(x, y, y') = -(\log(y) + 1) \quad \textit{and} \quad f_{y'}(x, y, y') = \cos(x).$$

To solve it we need to transform it into a first order system. Let $y_1(x) = y(x)$, $y_2(x) = y'(x)$, $z_1(x) = z(x)$, $z_2(x) = z'(x)$. *Then*

$$y_1' = y' = y_2 \quad \textit{and} \quad y_2' = y'' = 2y^3 = 2y_1^3$$

and

$$z_1' = z' = z_2 \quad \textit{and} \quad z_2' = z'' = -(\log(y) + 1)z + \cos(x)z' = -(\log(y_1) + 1)z_1 + \cos(x)z_2$$

We then have the first order system

$$\begin{cases} y_1' = y_2, \\ y_2' = 2y_1^3, \\ z_1' = z_2, \\ z_2' = -(\log(y_1) + 1)z_1 + \cos(x)z_2, \\ y_1(-1) = \frac{1}{2}, \\ y_2(-1) = s, \\ z_1(-1) = 0, \\ z_2(-1) = 1. \end{cases}$$

3 Finite difference method for linear second order BVP

Consider the boundary-value problem

$$\begin{cases} -u'' + p(x)u' + q(x)u = r(x), \quad x \in [a, b] \\ u(a) = \alpha \\ u(b) = \beta \end{cases} \tag{BVP}$$

As we did with initial value problems, we first discretize the interval $[a, b]$ in N equal subintervals with endpoints given by $x_k = a + kh$ for $k = 0, 1, \ldots, N$, where $h = (b - a)/N$.

At the interior grid points, we have

$$u''(x_k) + p(x_k)u'(x_k) + q(x_k)u(x_k) = r(x_k).$$

The idea is to approximate the first and second derivative of u using the finite difference formulas obtained before. Recall that

$$u''(x_k) = \frac{u(x_{k+1}) - 2y(x_k) + u(x_{k-1})}{h^2} + \mathcal{O}(h^2),$$

where $\xi_k \in (x_{k-1}, x_{k+1})$ and

$$u'(x_k) = \frac{u(x_{k+1}) - u(x_{k-1})}{2h} + \mathcal{O}(h^2),$$

where $\eta_k \in (x_{k-1}, x_{k+1})$. This leads to

$$-\frac{u(x_{k+1}) - 2y(x_i) + u(x_{k-1})}{h^2} + p(x_k)\frac{u(x_{k+1}) - u(x_{k-1})}{2h} + q(x_k)u(x_k) = r(x_k) + \mathcal{O}(h^2).$$

We have then obtained a finite-difference approximation with truncation error of order $\mathcal{O}(h^2)$. Using u_k to denote the approximation of $u(x_k)$ and ignoring the error term in the equation above we get

$$-\frac{u_{k+1} - 2u_k + u_{k-1}}{h^2} + p(x_k)\frac{u_{k+1} - u_{k-1}}{2h} + q(x_k)u_k = r(x_k)$$

which we can rewrite as

$$-\left(\frac{1}{h^2} + \frac{1}{2h}p(x_k)\right)u_{k-1} + \left(\frac{2}{h^2} + q(x_k)\right)u_k - \left(\frac{1}{h^2} - \frac{1}{2h}p(x_k)\right)u_{k+1} = r(x_k)$$

for $k = 1, \ldots, N - 1$. The Dirichlet boundary conditions in (BVP) lead naturally to

$$u_0 = \alpha, \quad u_N = \beta.$$

We end up then we a linear system $A_h\vec{u_h} = \vec{f_h}$ where

$$A_h = \begin{bmatrix} \frac{2}{h^2} + q(x_1) & -\left(\frac{1}{h^2} - \frac{1}{2h}p(x_1)\right) & 0 & \cdots & 0 \\ -\left(\frac{1}{h^2} + \frac{1}{2h}p(x_2)\right) & \frac{2}{h^2} + q(x_2) & -\left(\frac{1}{h^2} - \frac{1}{2h}p(x_2)\right) & & \\ 0 & \ddots & \ddots & \ddots & 0 \\ & & & & -\left(\frac{1}{h^2} - \frac{1}{2h}p(x_{N-2})\right) \\ 0 & \cdots & 0 & -\left(\frac{1}{h^2} + \frac{1}{2h}p(x_{N-1})\right) & \frac{2}{h^2} + q(x_{N-1}) \end{bmatrix},$$

$$\vec{u_h} = \begin{bmatrix} u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}, \quad \text{and} \quad \vec{f_h} = \begin{bmatrix} r(x_1) + \left(\frac{1}{h^2} + \frac{1}{2h}p(x_1)\right)\alpha \\ r(x_2) \\ \vdots \\ r(x_{N-2}) \\ r(x_{N-1}) + \left(\frac{1}{h^2} - \frac{1}{2h}p(x_{N-1})\right)\beta \end{bmatrix}.$$

In general, we have a linear system $A_h\vec{u_h} = \vec{f_h}$ to solve. Let $\vec{u} = (u(x_1), \ldots, u(x_{N-1}))^T$ be a vector of the exact values of u at $x = x_k$.

We are interested in measuring the error $\vec{u} - \vec{u_h}$ and to determine if $\vec{u_h}$ converges to $\vec{u}$. Before we do that, we need a way to measure "sizes" of vectors and matrices.

Definition 3.1. *Given a vector space V over $\mathbb{R}$, a norm of V is a function $||\cdot|| : V \to \mathbb{R}$ satisfying the following three properties: for any scalar $a \in \mathbb{R}$ and $\vec{x}, \vec{y} \in V$,*

1. $||a\vec{x}|| = |a|\,||\vec{x}||$,
2. $||\vec{x} + \vec{y}|| \leq ||\vec{x}|| + ||\vec{y}||$,
3. *If $||\vec{x}|| = 0$, then $\vec{x} = \vec{0}$.*

Example 3.1. *The Euclidean norm or l_2 norm,*

$$||\vec{x}|| = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}} = \sqrt{\vec{x}^T\vec{x}}$$

is a norm for $V = \mathbb{R}^n$, where $\vec{x} = (x_1, \ldots, x_n)^T$.

Definition 3.2. *Given a $n \times m$ matrix A and norms $||\cdot||_{\mathbb{R}^n}$, $||\cdot||_{\mathbb{R}^m}$ for $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively, the operator norm of A is*

$$||A||_{op} = \max_{\substack{\vec{x}\in\mathbb{R}^m \\ \vec{x}\neq\vec{0}}} \frac{||A\vec{x}||_{\mathbb{R}^n}}{||\vec{x}||_{\mathbb{R}^m}}$$

The operator norm of a matrix is indeed a norm and satisfies

$$||A\vec{x}||_{\mathbb{R}^n} \leq ||A||_{op}\,||\vec{x}||_{\mathbb{R}^m}$$

Example 3.2. *The operator norm given by*

$$||A||_2 = \max_{\substack{\vec{x}\in\mathbb{R}^m \\ \vec{x}\neq\vec{0}}} \frac{||A\vec{x}||_2}{||\vec{x}||_2}$$

is the spectral norm.

Theorem 3.3. *Let A be a real symmetric matrix (i.e. $A = A^T$ with real entries). Denoting $\lambda_i(A)$ as the eigenvalues of A for $i = 1, \ldots, n$, then*

$$||A||_2 = \max_{i=1,\ldots,n} |\lambda_i(A)|\,.$$

We can now discuss consistency, stability and convergence of a finite difference method.

Definition 3.4. *Consider the finite difference method $A_h\vec{u_h} = \vec{f_h}$ and denote by $\vec{u}$ the exact solution. The consistency error is $\left|\left|A_h\vec{u_h} - \vec{f_h}\right|\right|_2$ and we say the method is consistent if*

$$\lim_{h\to 0} \left|\left|A_h\vec{u_h} - \vec{f_h}\right|\right|_2 = 0.$$

The method is stable is there exists constants $C > 0$, $h_0 > 0$ such that $\left|\left|A_h^{-1}\right|\right| \leq C$ for all $h \leq h_0$. The method is convergent if

$$\lim_{h\to 0} ||\vec{u} - \vec{u_h}||_2 = 0.$$

Theorem 3.5 (Lax equivalence theorem)**.** *If the discretization $A_h\vec{u_h} = \vec{f_h}$ is consistent, then the method is convergent if and only if the method is stable.*

Proof. Suppose the discretization is consistent. We only show that if the method is stable than it is convergent. The converse is also true but it goes beyond what is studied in this course.

Suppose then that the method is stable. We have that

$$\begin{aligned} ||\vec{u} - \vec{u_h}||_2 &= \left|\left|(A_h^{-1} A_h(\vec{u} - \vec{u_h})\right|\right|_2 \\ &\leq \left|\left|A_h^{-1}\left(A_h\vec{u} - \vec{f_h}\right)\right|\right|_2 \\ &\leq \left|\left|A_h^{-1}\right|\right|_2 \left|\left|A_h\vec{u} - \vec{f_h}\right|\right|_2 \\ &\leq C \left|\left|A_h\vec{u} - \vec{f_h}\right|\right|_2 \end{aligned}$$

where we used the stability property and the fact that $A_h\vec{u_h} = \vec{f_h}$. The convergence then follows from the consistency property. □

References

[1] R. L. Burden and J. D. Faires. *Numerical Analysis*. 9^{th} edition. Brookes/Cole, 2004.

[2] A. Quarteroni, R. Sacco and F. Saleri. *Numerical Mathematics*. 2^{nd} edition. Springer, 2006.

[3] Tutorial notes written by Jan Feys

[4] Class notes written by Andy Wan for Math 317 Fall 2014.