APMA E4300: Introduction to Numerical Methods

Michael Cai

April 14, 2025

Abstract

This course introduces and analyzes fundamental numerical methods in scientific computation. The main topics include elementary numerical linear algebra, least-square methods, nonlinear equations and optimization, polynomial interpolation, numerical differentiation and integration, initial-value and boundary problems for ordinary differential equations and systems, eigenvalue problems. This course prepares students for more advanced studies in the field of numerical analysis and scientific computation (for instance, APMA courses E4301, E4302, and E6302).

Contents

Chapter 1

Linear Algebra Overview

Lecture 1

Jan 23 10:10

Remark.

$$Av = \lambda v$$

When $\{v_i\}_{i=1}^n$ is linearly independent, then we wil have that

$$Av_i = \lambda_i v_i$$

We can then write A as a product of two matrices such that we have

$$AQ = Q\alpha A = Q\alpha Q^{-1}$$

Remark. Suppose that some of the values do not contribute much in

$$A = Q\alpha Q^{-1}$$

Then we can throw away some nonzero elements given that they are small. Thus, we can create a approximation of the matrix which was a (256 x 256) matrix. We have converted it to a $n \times 2, 2 \times 2, 2 \times n$.

Example. We can have an image that is 256 x 256, we can compress it without losing too much information. However, there is never any free lunch. Any two dimensional object, 3d, etc. can be done. Thus, we can generalize it further in the course. . .

With λ being very small, it is negligible and can be omitted.

We would like to know how much error can be omitted.

Definition 1.0.1. l^2 norm is going to be denoted as $||\mathbf{x}||$ which is a negative number given by

$$\sqrt{\sum_{i=1}^{n} x_i^2}$$

Theorem 1.0.1. The l_2 norm satisfies the following properties:

- $||x||_{l_2} = 0 \Leftrightarrow x = 0$
- $||\alpha x|| = |\alpha|||x||_{l_2}$

Lecture 2

Jan 28 10:10

Remark. Gaussian Elimination:
Suppose we had an $n \times n$ system with n unknowns and n equations denoted as

$$Ax = y, A \in \mathbb{R}^{\Bbbk \times \Bbbk}$$

Example. Given an example of the equation

$$y = a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}$$

How do we figure out the coefficients. Now suppose that we are given a set of points, it is quite easy to solve a matrix for the given system of equations

$$a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} = y_1$$

$$\ldots$$

$$a_0 + a_1 x_n + \cdots = y_m$$

This results in a matrix where we can solve for the values of the coefficients now.

$$\begin{pmatrix} 1 & x_1 & x_1^{m-1} \\ \ldots & & \\ 1 & x_m & x_m^{m-1} \end{pmatrix} \begin{pmatrix} a_0 \\ \ldots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ \ldots \\ y_m \end{pmatrix}$$

In practice, we could measure data points which leads to a situation where there are more equations or x values than coefficients which is defined as a "overdetermined matrix". On the other hand, we can also have an undertermined problem.

Example. Suppose we have another example with a system of equations with $y = a_0 + a_1 \sin x + a_2 \sin 2x \ldots a_n \sin x$. If we continue this this is also a linear system of equations where the matrix of the equations will be

$$\begin{pmatrix} 1 & \sin x_1 & \sin(m-1)x_1 \\ \ldots & & \\ 1 & \sin x_m & \sin(m-1)x_m \end{pmatrix} \begin{pmatrix} a_0 \\ \ldots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ \ldots \\ y_m \end{pmatrix}$$

One useful example is a gaussian mixture model where you can suerimpose a number of gaussian models to get the value of y. Thus, solving $Ax = y$ gives many possible applications.

Elementary Row Operations:

Definition 1.0.2. Elementary row operations can be done with multiplication. Scaling is done with the following matrix

$$S_i(p) = \begin{pmatrix} 1 & & \\ & p & \\ & & 1 \end{pmatrix}$$

which gives us a scaling factor on a row of the matrix. We simply see that the inverse of the matrix would be simply be $\frac{1}{p}$ for $S_i(p)^{-1}$

Definition 1.0.3. Interchange: interchanging row i and row j of $\vec{A}$ gives us

$$E_{ij}A = \begin{pmatrix} 1 & & & & \\ & 0 & 1 & & \\ & 1 & 0 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} A$$

This showcases a exchange of rows i and j. Note that the inverse of such a matrix is just itself.

Definition 1.0.4. Replacement is also a method too. (finish later)

Remark. Gaussian Elimination:

We can start with an easy example

$$A = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

We can then apply a matrix L_1 which is our linear operator we use to transform our matrix using gaussian elimination.

We can apply these operations which results in

$$L_3 L_2 L_1 A x = L_3 L_2 L_1 y$$

which will give us a upper triangular matrix. We can then solve and use backwards substitution toget our answer from the upper triangular matrix.

Let us denote this as

$$U = L_3 L_2 L_1 A$$

Thus, we can write everything as

$$Ux = L^{-1} y \Rightarrow LUx = y$$

where U is always an upper triangular matrix and the L is always a lower triangular matrix. This only works with elimination methods and not row swaps or replacements. This can be easily solvable by having

$$LUx = y \Rightarrow Lz = y$$

solve for z, do a backward substitution for U and we can easily get the coefficients for x

Lecture 3

Lecture 4: 10:10

Jan 30 10:10

Could not attend. Check lecture slides instead.

4 Feb 2025

Lecture 5: 10:10

6 Feb 2025

1.1 Linear Iterative Schemes for Linear Systems

Definition 1.1.1. Define the error at step k to be

$$e^k = x^k - x \Rightarrow e^{k+1} = Be^k$$

$$\Rightarrow e^k = B^k e^0$$

We see if B is greater than 1 , we are amplifying the noise of the answer. Thus, we want to use some method to get the size of the matrix $I - A$ to be less than 1

1.1.1 Jacobi Iteration

For the Jacobi Iteration we acn write that

$$A = P - N$$

$$Ax = y \Rightarrow Px = Nx + y$$

Remark. We have rewritten the equation in this form but is not in the form $x = Bx + y$. We can write instead

$$X = P^{-1}Nx + P^{-1}y$$

We can now perform the iteration here which makes the calculations much easier.

Recall that the Jacobi is going to consist of

$$P = D_A, N = L_A + U_A$$

Remark. Selecting $P = D_A$ is useful because the inverse of P is simply

$$\begin{pmatrix} \frac{1}{p_1} & & \\ & \frac{1}{p_2} & \\ & & \frac{1}{p_n} \end{pmatrix}$$

Thus given x_0,

$$x^{k+1} = D_A^{-1}(L_A + U_A)x^k + D_A^{-1}y, k \geq 0$$

Note that if we have the diagonal to be 0 for one part, the method is not possible. However, it is nice to see when the diagonal is very large.

In component form, we can write this as

$$x_1^{k+1} = \frac{y_1 - \sum_{j\neq 1} a_j x_j^K}{a_{11}}$$

$$\vdots$$

$$x_i^{k+1} = \frac{y_i - \sum_{i\neq j} a_{ij} x_j^k}{a_{ii}}$$

$$\vdots$$

$$x_n^{k+1} = \frac{y_n - \sum_{j\neq n} a_{nj} x_j^k}{a_{nn}}$$

Stopping Criteria We have to designate a stopping criteria for linear iterations. ONe of the methods is using a maximum amount of iterations and the other method is using the relative size of the residual. This is defined as the size of $y - Ax^k$ over the initial as

$$\frac{\|y - Ax^k\|_{l2}}{\|y - Ax^0\|_{l2}}$$

Lecture 6: 10:10

Could not attend. Check lecture slides instead. 11 Feb 2025

Lecture 7: 10:10

Could not attend. Check lecture slides instead. 13 Feb 2025

Lecture 8: 10:10

Could not attend. Check lecture slides instead. 18 Feb 2025

Lecture 9: 10:10

Could not attend. Check lecture slides instead. 20 Feb 2025

Lecture 10: 10:10

Could not attend. Check lecture slides instead. 25 Feb 2025

Lecture 11: 10:10

Could not attend. Check lecture slides instead. 4 Mar 2025

Lecture 12

6 Mar 10:10

Chapter 2

Interpolation

Consider the concept of interpolation where we are using extrapolation and generalization. Suppose we had a neural network where we have fitted it with data point $D(t)$. We are asking how general the neural network can be such that we can fit our data.

Definition 2.0.1. Interpolant: Given a discrete set of values y_i at locations x_i an interpolant is a piecewise continuous function $f(x)$ that passes through the data such that

$$f(x_i) = y_i \quad 1 \leq i \leq n$$

After we fit the function we want to evaluate the new data points. Notice that there is a factor that we can't control for such that we don't know the form of $D(t)$ where $D(t) = P_n t^n + P_{n-1} t^{n-1} \cdots + P_1$

Example. Suppose we had data points that looked like a linear line. Then we can guess the form

$$D(t) = a + bt$$

Given our plots we have a linear set of equations

$$a + bt_1 = D_1$$

$$a + bt_2 = D_2$$

$$\dots$$

Example. If we want a quadratic interpolation we need three data points
Let $P_2(x) = p_0 + p_1 x + p_2 x^2$, then we have the requirements

$$\sum_{i=1}^{3} p_0 + p_1 x_i + p_2 x_i^2 = y_i$$

which can be put into matrix form and solved which will give a form in terms of fractions.

In general, given $n+1$ data points we can generate an n-th order polynomial of the form

$$\begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ & & & \\ 1 & & \dots & x_n^n \end{pmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$Ap = y$$

Theorem 2.0.1. Weierstrass Approximation Theorem: Let $f \in \mathcal{C}([a,b])$ be given. For $\forall \epsilon > 0, \exists$ a polynomial $P(x)$ such that

$$|P(x) - f(x)| \leq \epsilon, \quad \forall x \in [a,b]$$

2.1 Lagrange Interpolation

Theorem 2.1.1. Suppose we had $n+1$ distinct points. THen there exists a unique polynomial of degreen n such that $P(x)$ pass through these points. We must have

$$P(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}(x), \quad L_{n,k}(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

where L are the Lagrange interpolating polynomials at node x_k

Remark. Recall the coefficients of a nice structure in terms of fractions. This structure can go to arbitrary orders which lead to the lagrange interpolation. Thus we get the following form nice form where

$$P(x) = \sum_{k=0, x_n \neq x_k}^{n} f(x_k) \frac{x - x_0}{x_k - x_0} \frac{x - x_1}{x_k - x_1} \cdots \frac{x - x_n}{x_k - x_n}$$

This term is given as $L_{n,k}(x)$. Thus we get the form

$$P(x) = y_0 L_{n,0}(x) + y_1 L_{n,1}(x) + \cdots + y_n L_{n,n}(x)$$

Observe that

$$L_{n,k}(x_j) = \delta_{kj} = \begin{cases} 1, k = j; \\ 0, k \neq j; \end{cases}$$

Theorem 2.1.2. Error of Lagrange Interpolation: If $f \in \mathcal{C}^{n+1}(pa, b)$, then we have that

$$f(x) = \sum_{k=0}^{n} f(x_k) L_{n,x}(x) + \frac{f^{(n+1)}(\eta)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

where the second term is defined to be the error. We can see a taylor expansion where this is the residual terms.

Lecture 13: 10:10

11 Mar 2025

2.2 Lagrange Interpolation Proof

Proof. The Lagnrange polynomial $L_{nk}(x)$ is given as

$$L_{nk}(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

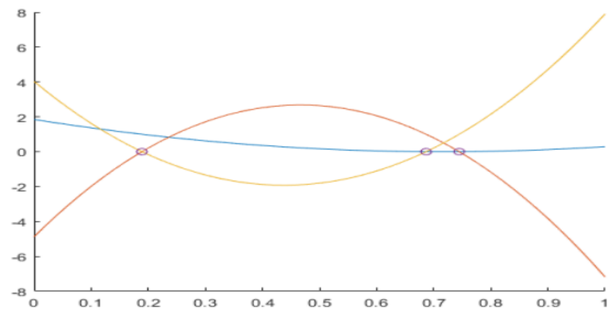
If $L_{nk}(x_j) = \delta_{kj}$ then if $k = j$ the whole expression cancels to be one. And 0 if otherwise. Let us now show that

$$P(x_j) = \sum_{k=0}^{n} f(x_k) L_{nk}(x_j) = f(x_j), \quad 0 \leq j \leq n$$

which we can do easily. Since $P(x)$ passes through all nodes, it must be a unique answer since for a function s.t $Q(x_j) = y_j$ must be that $P(x_j) - Q(x_j)$ which implies uniqueness. Since both are n

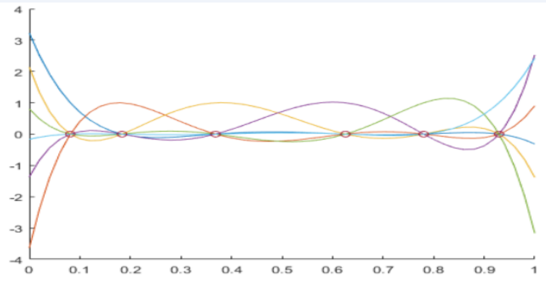
degree, by the fundamental theorem of algebra it must be that with n real roots they are the same. ■

Example. Below are examples of graphs generated by Lagrange Interpolation



The Lagrange polynomial $L_{2,0}(x)$, $L_{2,1}(x)$ and $L_{2,2}(x)$ in $[0,1]$ with points $x_0 = 0.188955015032545$, $x_1 = 0.686775433365315$ and $x_2 = 0.744692807074156$.

Figure 2.1



The Lagrange polynomial $L_{5,0}(x)$, $L_{5,1}(x)$, $L_{5,2}(x)$, $L_{5,3}(x)$, $L_{5,4}(x)$, and $L_{5,5}(x)$ in $[0,1]$ with points $x_0 = 0.081125768865785$, $x_1 = 0.183511155737270$, $x_2 = 0.368484596490336$, $x_3 = 0.625618560729690$, $x_4 = 0.780227435151377$, and $x_5 = 0.929385970968730$.

Figure 2.2

Remark. Lagrange interpolation is an explicit formula without any additional computation. For example, a interpolation from two points has the form

$$P_1(x) = p_0 + p_1 x$$

$$P_1(x) = y_0 \frac{y_1 y_0}{x_1 - x_0}(x - x_0)$$

2.3 Hermite Interpolation

Definition 2.3.1. Osculatory Polynomial Approximation: Suppose we had $n+1$ distinct points on a closed interval with a set of $n+1$ arbitrary integers taking $m \equiv \max(m_0, m_1, \ldots, m_n)$. The osculating polynomial approximation is a polynomial $P(x)$ s.t.

$$\frac{\mathrm{d}^{k_i} P}{\mathrm{d}x^{k_i}}(x_i) = \frac{\mathrm{d}^{k_i} f}{\mathrm{d}x^{k_i}}(x_i), \quad \forall 0 \leq k_i \leq m$$

We can understand two approximations: taylor and lagrange polynomial approximation where Taylor is given by

$$P(x) = \sum_{k=0}^{m_0} \frac{f^(k)(x_0)}{k!}(x-x_0)^k$$

For the taylor apprixmation, we are matching the derivatives of the function f to an order up to m_0 at x_0. For the Lagrange approximation we are matching all $n+1$ function values rather the values of the derivatives.

Definition 2.3.2. Hermite Approximation: When $m_i = 1$ for all $0 \leq i \leq n$ we wish to match the function values there as well as the first derivatives such that

$$P(x_i) = f(x_i), \quad P'(x_i) = f'(x_i)$$

Theorem 2.3.1. Let $f \in \mathcal{C}^1$ with points. There exists a unique polynomial that it matches the Hermite Approximation conditions with a degree of at most $2n+1$ where

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j)H_j(x) + \sum_{j=0}^{n} f'(x_j)\hat{H}_j(x)$$

where

$$H_j(x) = \left[1 - 2(x-x_j)L'_{nj}(x_j)\right] L^2_{nj}(x), \quad \hat{H}_j(x) = (x-x_j)L^2_{nj}(x)$$

Theorem 2.3.2. If $f \in C^{2n+2}$ then

$$f(x) = H_{2n+1}(x) + \frac{f^{(2n+2)}(\zeta)}{(2n+2)!}\prod_{i=0}^{n}(x-x_i)^2$$

Lecture 14: 10:10

13 Mar 2025

2.4 Spline Interpolation

Definition 2.4.1. Piecewise Polynomial Approximation: The lagrange and hermite are global interpolation schemes where they work for all interpolation nodes. We consider local alternatives

Definition 2.4.2. Piecewise Linear Interpolation: Constructs a linear interpolation between each point $[x_i, x_{i+1}]$ where each subinterval is a Lagrange interpolation

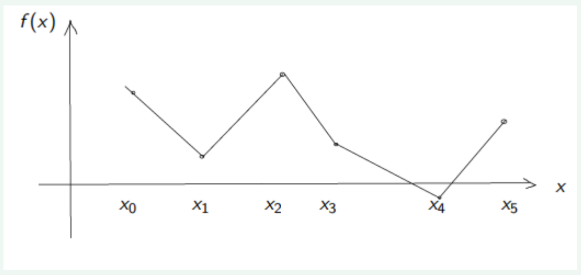


Figure 2.3

Definition 2.4.3. Piecewise Hermite Interpolation: Same as PLI but using a Hermite interpolation s.t. the polynomial is of degree 3.

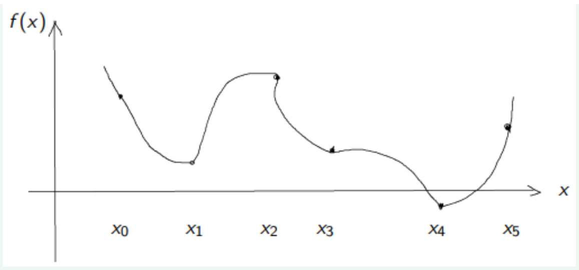


Figure 2.4

What we wish for for a local interpolation has the following properties

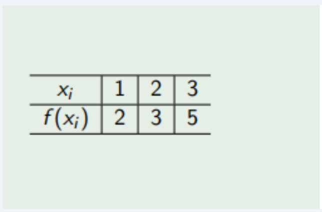
1. Smooth and differentiable at nodes
2. One does not need the derivative to approximate $f'(x_i)$

What we wish to find is the cubic spline interpolation.

Definition 2.4.4. Cubic Spline Interpolation: Given a function f defined with given nodes. The interpolant $S(x)$ for $f(x)$ satisfies the following conditions:

1. $S(x)$ is a cubic polynomial
2. $S_j(x_j) = f(x_j)$ and such for all points
3. $S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = f(x_{j+1})$
4. $S'_j = S'_{j+1}$
5. $S''_j = S''_{j+1}$
6. The boundary conditions must be satisfied where either $S''(x_0) = S''(x_n) = 0$ or we must have that $S'(x_0) = f'(x_0)$ and such for all other S'

Example. Suppose we have an example where we have the subintervals given as $[1, 2]$ and $[2, 3]$



x_i	1	2	3
$f(x_i)$	2	3	5

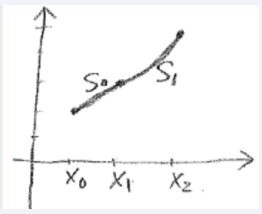


Figure 2.5

which are given by the equation of the form

$$S_i(x) = a_i + b_i x + c_i x^2 + d_i x^3$$

where the index i denotes the equation. Given 8 unknowns from the two subintervals we have 8 equations to solve.

1. Match $S_0(x_0) = f(x_0)$
2. Match $S_0(x_1) = f(x_1)$
3. Next match the second line segment S_1 to the B.C
4. Match the first derivatives
5. Match the second derivatives

6. Use the natural boundary condition at x_0 and x_2 where the second derivatives must be 0

Satisfying these 8 conditions we can solve which gives us the cubic spline interpolant

Theorem 2.4.1. Uniquness of Natural Cubic Spline Interpolant: If f is defined for an interval $[a, b]$ with points inside, f has a unique spline interpolant s.t. it satisfies the natural B.C. s.t. $S''(a) = S''(b) = 0$

From this we can define a general procedure to solve for the cubic spline. For notation, define $I_j = [x_j, x_{j+1}]$ for the jth interval. Then we have the segment as

$$S_j(x0) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Additionally define $h_j = x_{j+1} - x_j$ The cubic spline is then written as

$$S(x) = \begin{cases} S_0(x), x \in I_0; \\ \dots \\ S_{n-1}(x), x \in I_{n-1}; \end{cases}$$

1. Fit the function on the nodes s.t. $S_j(x_j) = f(x_j)$ where we are matching terms $a_j = f(x_j)$
2. Match interpolation on interior nodes for $S_j(x_{j+1}) = S_{j+1}(x_{j+1})$ and so on s.t. we get the form $a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 = a_{j+1}$
3. Match the derivatives of the interior nodes where we must have $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$ which gives the form
$$b_j + 2c_j h_j 3d_j h_j^2 = b_{j+1}$$
4. Match the second order derivatives at the interpolation nodes s.t. $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$ which gives
$$c_j + 3d_j h_j = c_{j+1}$$
5. Apply the natural boundary conditions s.t. $S''(x_0) = 0 \Rightarrow c_0 = 0$
6. Apply the natural B.C. on the left s.t. $S_{n-1}''(x_n) = 0$
$$x_n = 0 \quad c_{n-1} + 3d_{n-1}h_{n-1} = 0$$
Solve the coefficents to get that
$$d_j = \frac{1}{3h_j}(c_{j+1} - c_j)$$
$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1})$$
$$b_{j+1} = b_j + h_j(c_j + c_{j+1})$$
Continuing the process, we can solve for all the necessary coefficients needed given the conditions solved.

Remark.
1. We only find coefficients for $S_j(0 \leq j \leq n-1)$ or the set $[a, b, c, d]$. the coefficient $c_0 = 0$ is given by B.C.
2. Introducing extra variables isn't needed for the output of the algorithm, e.g. $a_n \equiv f(x_n), b_n = S'(x_n), c_n = \frac{1}{2}S''(x_n)$
3. Note that in the Natural B.C. case where $c_n = 0$, b_n is undetermined but can be determined if using the clamped B.C.

The algorithm is given as shown below:

Algorithm 2.1: Natural Cubic Spline

```
Input: Integer $n$, arrays $x[0:n]$, $f[0:n]$
Initialize:
$a[0:n] \leftarrow 0$, $h[0:n-1] \leftarrow 0$, $y[0:n] \leftarrow 0$,
$b[0:n-1] \leftarrow 0$, $c[0:n] \leftarrow 0$, $d[0:n-1] \leftarrow 0$;
for $j = 0$ to $n$ do
    $a[j] \leftarrow f[j]$;
for $j = 0$ to $n-1$ do
    $h[j] \leftarrow x[j+1] - x[j]$;
for $j = 1$ to $n-1$ do
    $y[j] \leftarrow \frac{3}{h[j]}\Big(a[j+1] - a[j]\Big) - \frac{3}{h[j-1]}\Big(a[j] - a[j-1]\Big)$;
Initialize matrix $A[0:n, 0:n] \leftarrow 0$, with $A[0,0] \leftarrow 1$ and $A[n,n] \leftarrow 1$;
for $j = 1$ to $n-1$ do
    $A[j, j-1] \leftarrow h[j-1]$;
    $A[j, j] \leftarrow 2\Big(h[j-1] + h[j]\Big)$;
    $A[j, j+1] \leftarrow h[j]$;
$c \leftarrow \text{LinSolve}(A, y)$;
for $j = 0$ to $n-1$ do
    $d[j] \leftarrow \frac{1}{3h[j]}\Big(c[j+1] - c[j]\Big)$;
for $j = 0$ to $n-1$ do
    $b[j] \leftarrow \frac{1}{h[j]}\Big(a[j+1] - a[j]\Big) - \frac{h[j]}{3}\Big(2c[j] + c[j+1]\Big)$;
return $a$, $b$, $c$, $d$;
```

Chapter 3

Numerical Differentiation

Lecture 16: 10:10

25 Mar 2025

3.1 Definitions of Differentiation

Definition 3.1.1. We know for the definition of a derivative is given as Forward Difference:

$$f(x) = \lim_{h\to 0} \frac{f(x+j) - f(x)}{h}$$

Backwards Difference:

$$\lim_{h\to 0} \frac{f(x) - f(x-h)}{h}$$

Central Difference

$$\lim_{h\to 9} \frac{f(x+h) - f(x-h)}{2h}$$

We define the following wherefor numerical differentiation that h is small. Note the above definitions are going to be different types of numerical differentiations. To check which gives the best approximation, we can Taylor expand s.t

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f'(x)h^2 + \dots$$

$$f(x-h) = f(x) + f'(x)(-h) + \frac{1}{2}f''(x)h^2$$

$$\Rightarrow \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2}f''(x)h$$

$$\frac{f(x) - f(x-h)}{h} = f'(x) - \frac{1}{2}f''(x)h$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{1}{3!}f'''h^2$$

where we see the central difference gives a smaller error where $h^2 \ll 1$ thus the error term shows up only when f''' is very large.

3.2 General Approach $n+1$ Point Formula

Assume we want to approximate $n+1$points around x to approximate the derivative $f'(x)$. The Lagrange polynomial approximation is given as

$$f(x) = \sum_{k=0}^{n} f(x_k)L_{nk}(x) + \frac{f^{(n+1)}(\zeta)}{(n+1)!}\prod_{k=0}^{n}(x - x_k)$$

We can simply differentiate to have the new expression as

$$f'(x) = \sum_{k=0}^{n} f(x_k) L'_{nk}(x) + \frac{f^{(n+1)}(\zeta)}{(n+1)!} \left[\prod_{k=0}^{n} (x - x_k) \right]'$$

This is the generic method of differentiation from interpolation which the second term represents the error term.

Remark. If we naively use this method for points spread apart, the second error term ϵ will be very large. This is the case for the taylor expansion too where the points are very spread apart.

For the general formulation to find the derivatives at x_j we simply plug in to the lagrange interpolation and take its derivative. We then ask the question of whether or not we recover the finite difference approximation for the setup of x_0, x_1, x_2.

3.2.1 Three and Five Point Formula

Consider three points difference scheme to approximate the derivative. We can rewrite this in terms of x and h such that they are spatially selected to be uniform where we can plug in for the the points of $x-h$, x, and $x+h$. If we calculate the Lagrangian interpolation our result for a uniform mesh is given.

$$f'(x_0) = \frac{1}{2h}\left[-3f(x_0) + 4f(x_1) - f(x_2)\right] + \frac{f^{(3)}(\zeta_0)}{3!}h^2$$

$$f'(x_1) = \frac{1}{2h}\left[-f(x_0) + f(x_2)\right] + \frac{f'''(\zeta_1)}{3!}h^2$$

$$f'(x_2) = \frac{1}{2h}\left[f(x_0) - 4f(x_1) + 3f(x_2)\right] + \frac{f'''(\zeta_2)}{3!}h^2$$

Similarly there exists a five point formula that does the same thing with a higher order accuracy.

Chapter 4

Numerical Differentiation

Lecture 15: 10:10

25 Mar 2025

4.1 Definitions of Differentiation

Definition 4.1.1. We know for the definition of a derivative is given as Forward Difference:

$$f(x) = \lim_{h\to 0} \frac{f(x+j) - f(x)}{h}$$

Backwards Difference:

$$\lim_{h\to 0} \frac{f(x) - f(x-h)}{h}$$

Central Difference

$$\lim_{h\to 9} \frac{f(x+h) - f(x-h)}{2h}$$

We define the following wherefor numerical differentiation that h is small. Note the above definitions are going to be different types of numerical differentiations. To check which gives the best approximation, we can Taylor expand s.t

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f'(x)h^2 + \dots$$

$$f(x-h) = f(x) + f'(x)(-h) + \frac{1}{2}f''(x)h^2$$

$$\Rightarrow \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2}f''(x)h$$

$$\frac{f(x) - f(x-h)}{h} = f'(x) - \frac{1}{2}f''(x)h$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{1}{3!}f'''h^2$$

where we see the central difference gives a smaller error where $h^2 \ll 1$ thus the error term shows up only when f''' is very large.

4.2 General Approach $n+1$ Point Formula

Assume we want to approximate $n+1$points around x to approximate the derivative $f'(x)$. The Lagrange polynomial approximation is given as

$$f(x) = \sum_{k=0}^{n} f(x_k)L_{nk}(x) + \frac{f^{(n+1)}(\zeta)}{(n+1)!}\prod_{k=0}^{n}(x - x_k)$$

We can simply differentiate to have the new expression as

$$f'(x) = \sum_{k=0}^{n} f(x_k) L'_{nk}(x) + \frac{f^{(n+1)}(\zeta)}{(n+1)!} \left[\prod_{k=0}^{n} (x - x_k) \right]'$$

This is the generic method of differentiation from interpolation which the second term represents the error term.

Remark. If we naively use this method for points spread apart, the second error term ϵ will be very large. This is the case for the taylor expansion too where the points are very spread apart.

For the general formulation to find the derivatives at x_j we simply plug in to the lagrange interpolation and take its derivative. We then ask the question of whether or not we recover the finite difference approximation for the setup of x_0, x_1, x_2.

4.2.1 Three and Five Point Formula

Consider three points difference scheme to approximate the derivative. We can rewrite this in terms of x and h such that they are spatially selected to be uniform where we can plug in for the the points of $x - h$, x, and $x + h$. If we calculate the Lagrangian interpolation our result for a uniform mesh is given.

$$\begin{aligned}
f'(x_0) &= \frac{1}{2h} \left[-3f(x_0) + 4f(x_1) - f(x_2) \right] + \frac{f^{(3)}(\zeta_0)}{3!} h^2 \\
f'(x_1) &= \frac{1}{2h} \left[-f(x_0) + f(x_2) \right] + \frac{f'''(\zeta_1)}{3!} h^2 \\
f'(x_2) &= \frac{1}{2h} \left[f(x_0) - 4f(x_1) + 3f(x_2) \right] + \frac{f'''(\zeta_2)}{3!} h^2
\end{aligned}$$

Similarly there exists a five point formula that does the same thing with a higher order accuracy.

Lecture 17: 10:10

1 Apr 2025

4.3 Richardson's Extrapolation

Definition 4.3.1. Richardson extrapolation is used to construct higher order approximations for lower order approximations

For the setup, we approximate ff with $f_1(h)$ of the form

$$f = f_1(h) K_1 h + k_2 h^2 + \dots$$

where the Ks don't depend on x. With this scheme, we can construct higher order approximations. For example given that

$$f = f_1(\frac{h}{2}) + K_1(\frac{h}{2}) + K_2(\left(\frac{h}{2}\right)^2)$$

THen subtracting twice of this and the second we have

$$f = 2f_1(\frac{h}{2}) - f_1(h) - K^2 \frac{h^2}{2} + \mathcal{O}(h^3)$$

$$\begin{aligned}
2f_1(\frac{h}{2}) - f_1(h) &= 4\frac{f(x + \frac{h}{2}) - f(x)}{h} - \frac{f(x+h) - f(x)}{h} \\
&= \frac{4f(x + \frac{h}{2}) - 3f(x) - f(x+h)}{h} \\
&= \frac{1}{h} \left(-f(x+h) + 4f(x + \frac{h}{2}) - 3f(x) \right)
\end{aligned}$$

Lecture 19: 10:10

4.4 Gaussian Quadrature

7 Apr 2025

Lecture 19: 10:10

We now assume that the nodes are not given and we would like to pick nodes and weights $\{w_i, x_i\}$ s.t. it minimizes the error where

8 Apr 2025

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=1}^n w_i f(x_i)$$

The criteria we must determine are the $2n$ parameters where the quadrature has a degree of accuracy $K = 2n - 1$ where the polynomials are no greater than $2n - 1$

Example. Consider the integral

$$\int_{-1}^1 f(x)\mathrm{d}x = w_1 f(x_1) + w_2 f(x_2)$$

where we have $2n - 1 = 3$. Looking for polynomial of degree $k = 0$ we have $f(x) = 1 \Rightarrow \int f = 2$ which requires

$$w_1 f(x_1) + w_2 f(x_2) = 2 \Rightarrow w_1 + w_2 = 2$$

Next we do this requirement for $k = 1$, $k = 2$, $k = 3$. Solving we get a system of equations s.t.

$$x_1 = \frac{-\sqrt{3}}{3}, x_2 = \frac{\sqrt{3}}{3}, w_1 = 1, w_2 = 2$$

Chapter 5

ODE and Boundary Problems

Lecture 20

5.1 Initial Value Problems

9 Apr 2025 10:1

Consider an initial value problem we thus want to solve for

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y), \quad a \leq t \leq b$$

$$y(a) = y_a$$

Definition 5.1.1. A function $f(t, y)$ is Lipschitz continuous with Lipschitz constant L in the variable y if

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|. \quad \forall (t, y_1), (t, y_2) \in \mathrm{Dom}(f)$$

Example. Consider

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

$$D = \{(t, y)|1 \leq t \leq 2, -3 \leq y \leq 4\}$$

We can check this that

$$|ty_1 - ty_2| \leq L|y_1 - y_2|$$

Therefore we must have that $L = 2$

Theorem 5.1.1. Let $\mathcal{D} = \{(t, y)|a \leq t \leq b, -\infty < y < \infty\}$ and f is Lipschitz continuous. Then the IVP must have a unique solution.

Example. Consider

$$y'(t) = 1 + t\sin(ty), \quad 0 \leq t \leq 2$$

$$y(0) = 0$$

It must be that my MVT

$$\frac{f(t, y_2) - f(t, y_1)}{y_2 - y_1} = \frac{\partial f}{\partial y}(t, \zeta)$$

$$|f(t, y_2) - f(t, y_1)| = t^2\cos(t\zeta)|y_2 - y_1| \leq 4|y_2 - y_1|$$

Thus $L = 4$

5.2 Forward Euler

Definition 5.2.1. Time grid is the time instances covered in $[a, b]$

$$a = t_0 < t_1 < \cdots < t_{N-1} < t_N = b$$

Therefore

$$t_i = a + ih, \quad h = \frac{b-a}{N}$$

With this we can construct the derivatives at each point to calculate the values of y

Definition 5.2.2. Forward Euler's uses a forward finite difference scheme

$$\frac{\mathrm{d}y}{\mathrm{d}t}(t_i) \approx \frac{y(t_{i+1}) - y(t_i)}{h}$$

Rewriting we must have taht

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)), \quad i = 0, 1, \ldots, N-1$$

This makes any problem trivial to find future values given an IVP:

Algorithm 5.1: Forward Euler Method

```
Input: $a, b, y_0, N, f$
Output: $y$
$h \leftarrow \frac{b-a}{N}$;
$y[0 : N] \leftarrow 0$; $y[0] \leftarrow y_0$;
for $j = 0$ to $N - 1$ do
    $t \leftarrow a + j \cdot h$;
    $y[j + 1] \leftarrow y[j] + h \cdot f(t, y[j])$;
return $y$;
```

5.3 Backward Euler Method

Definition 5.3.1. Consider the backward finite difference where

$$\frac{\mathrm{d}y}{\mathrm{d}t}(t_i) \approx \frac{y(t_i) - y(t_{i-1})}{h}$$

$$y(t_i) - hf(t_i, y(t_i)) = y(t_{i-1})$$

We can then backwards solve for the next values.

Algorithm 5.2: Backward Euler Method

```
Input: $a, b, y_0, N, f$
$h = \frac{b-a}{N}$;
for $j = 1$ to $N$ do
    $t = a + i \cdot h$;
    $y[j] = \text{solve}\left(g(x, t), y[i - 1]\right)$ ;
return $y$;
```

5.4 Elementary Error Analysis

Choose $\{Y\}_{i=0}^{N}$ to be the numerical solution of an IVP, then we have

$$\frac{Y_{i+1} - Y_i}{h} = f(t_i, Y_I)$$

Definition 5.4.1. Trucation Error is defined to be the residual we get from truncating our solution when we replace Y_i by $y(t_i)$

$$\tau_{i+1}(h,i,y) = \frac{y(t_{i+1} - y(t_i))}{h} - f(t_i, y(t_i))$$

Definition 5.4.2. The order of accuracy is s.t. for $C > 0$

$$|\tau_{i+1}| \leq Ch^p, \quad t \in [0,T]$$

Consider forward euler using a taylor expansion we will get

$$y(t) = y(t_i) + (t-t_i)y'(t_i) + \frac{y''}{2}(t-t_i)^2 + \mathcal{O}((t-t_i)^3)$$

Plugging into the forward euler method for both points

$$\begin{aligned}\tau_i(h,i,y) &= \frac{y(t_{i+1} - y(t_i))}{h} - f(t_i, y(t_i)) \\ &= \frac{y(t_{i+1} - y(t_i))}{h} - y'(t_i) \\ &= \frac{y''(t_i)}{2}h + \mathcal{O}(h^2)\end{aligned}$$

5.5 Higher Order Methods

Consider IVP of ODEs with taylor expansions with y that are sufficiently regular. We will directly consider the nth order scheme given in taylor's method where

$$Y_{i+1} = Y_i + hT^{(n)}(t_I, Y_i), \quad Y_0 = y_a$$

$$T^{(n)}(t_i, Y_i) = f(t_i, Y_i) + \frac{h}{2}f'(t_i, Y_i) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i.Y_I)$$