MA 214 - Introduction to Numerical Analysis

Instructor: *Prof. Saikat Mazumdar*

Last updated February 23, 2021

Om Prabhu

Undergraduate, Mechanical Engineering

Indian Institute of Technology Bombay

Note To Reader

This document is a compilation of the notes I made while taking the course MA 214 (Introduction to Numerical Analysis) in my 4^{th} semester at IIT Bombay. It is not meant to serve as a replacement for any formal textbook or lecture on the subject, since I sometimes overlook the theory parts.

There will probably be many instances where I use certain symbols without explicitly mentioning what they mean. It is to be assumed that they carry their usual meanings. I may also change the order of notes compared to those in the slides if I find it more convenient.

If you have any suggestions and/or spot any errors, you know where to contact me.

Contents

Notation

$\mathbb{P}_n$	set of all polynomials of degree $\leqslant n$
$\mathcal{C}[a,b]$	set of all continuous functions on $[a,b]$ (an infinite dimensional vector space)
$\mathcal{C}^n[a,b]$	set of all n^{th}-order continuously differentiable functions on $[a,b]$

---

1 Interpolation Theory

Suppose $(n+1)$ real points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ are known. Further these **interpolation points** x_i are spread out over the interval $[a, b]$. Then the problem of approximating a function over the interval $[a, b]$ passing through these points is called interpolation.

There are infinite such functions. We mainly consider polynomial interpolation in this section i.e. we approximate the **interpolant** f by an **interpolating polynomial** $p_n \in \mathbb{P}_n$.

1.1 Some existence theorems

1. The **Joseph-Louis Lagrange Theorem** states that given a set of $n+1$ real, unique data points $S = \{(x_i, y_i) \mid i = 0, 1, \ldots, n\}$, there exists a unique polynomial $p_n \in \mathbb{P}_n$ such that

$$p(x_i) = y_i \text{ for } i = 0, 1, \ldots, n$$

We define the norm on $\mathcal{C}[a.b]$ as: $||f|| = \max_{x \in [a,b]} |f(x)|$. To define the 'closeness' of 2 functions formally, we consider the quantity $||f - g|| = \max_{x \in [a,b]} |f(x) - g(x)|$.

2. Take a function $f \in \mathcal{C}[a, b]$. The **Weierstrass Approximation Theorem** states that given any real number $\varepsilon > 0$, there exists a polynomial p such that

$$||f - p|| < \varepsilon \implies |f(x) - p(x)| < \varepsilon \quad \forall x \in [a, b]$$

1.2 Lagrange interpolation formula

Given $n+1$ distinct real points $x_0, x_1, \ldots, x_n$ and a function f whose values are known at these points, there exists a unique polynomial $p_n \in \mathbb{P}_n$ such that $p_n(x_i) = f(x_i)$ for $i = 0, 1, \ldots, n$. Construct n^{th} degree polynomials $L_0^n(x), L_1^n(x), \ldots, L_n^n(x)$ such that

$$L_k^n(x_i) = \delta_{ki} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases} \implies \boxed{p_n(x) := \sum_{k=0}^{n} f(x_k) L_k^n(x)}$$

The **lagrange polynomials** L_k^n can be found using the following algorithm

$$\boxed{L_k^n(x) := \prod_{j=0, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)}}$$

NOTE: As will be seen later, the method of divided differences can also be used for polynomial interpolation. A little bit of manipulation on the Lagrange interpolation formula gives us an alternative way to calculate the divided difference $f[x_0, x_1, \ldots, x_n]$, given by

$$f[x_0, x_1, \ldots, x_n] := \sum_{k=0}^{n} f(x_k) \prod_{j=0, j\neq k}^{n} \frac{1}{x - x_j}$$

1.3 Newton's divided differences

Let $x_0, x_1, \dots, x_n$ be $n+1$ real distinct points in $[a, b]$. Let $f : [a, b] \to \mathbb{R}$ be a function whose values are known at these points. We want to find a polynomial $p_n(x) \in \mathbb{P}_n$ such that $p_n(x_i) = f(x_i)$ for $i = 0, 1, \dots, n$.

We define the **divided differences** (independant of order of points) using the recursive relation:

$$f[x_0] := f(x_0)$$

$$f[x_0, x_1, \dots, x_{m+1}] := \frac{f[x_1, \dots, x_{m+1}] - f[x_0, \dots, x_m]}{x_{m+1} - x_0}$$

Then the polynomial $p_n(x)$ can be written as:

$$\boxed{p_n(x) := f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \dots, x_n] \prod_{k=0}^{n-1} (x - x_k)}$$

1.4 Matrix representation

The problem of interpolation can also be expressed as a system of linear equations and solved for the coefficients. A matrix similar to the Vandermonde matrix is generated.

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & \ddots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

1.5 Error estimation

Take $f \in \mathcal{C}^{n+1}[a, b]$. Let $x_0, x_1, \dots, x_n$ be $n + 1$ distinct points in $[a, b]$. Let $p \in \mathbb{P}_n$ such that $p(x_i) = f(x_i)$ for $i = 1, 2, \dots, n$. Then for all $x \in [a, b]$, there exists $\xi = \xi(x) \in (a, b)$ such that

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{k=0}^{n} (x - x_k)$$

Taking maximum over $x \in [a, b]$, we can see that our choice of interpolation points influences the error significantly.

$$\max_{x \in [a,b]} |f(x) - p(x)| \leqslant \frac{1}{(n+1)!} ||f^{(n+1)}|| \max_{x \in [a,b]} \prod_{k=0}^{n} |(x - x_k)|$$

This invokes the concept of **Chebyshev's interpolation points**. These are essentially the vertical projections of equally spaced points on a half-circle with center $\frac{a+b}{2}$ and radius $\frac{b-a}{2}$, given by

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{k\pi}{n}\right)$$

1.6 Piecewise interpolation

A function $\varphi \in \mathcal{C}[a,b]$ is a **piecewise polynomial** on $[a,b]$ if

- there exist points $\{x_i\}_{i=0}^{n}$ such that $a = x_0 < x_1 < \cdots < x_n = b$
- $\varphi \in \mathbb{P}_m$ is defined in each interval $[x_{i-1}, x_i]$ but not necessarily on the entire domain
- $m \leqslant n$ and $m \geqslant 0$

Piecewise interpolation involves building a function $\varphi \in \mathcal{C}[a,b]$ such that $\varphi \in \mathbb{P}_n$ on $[x_{i-1}, x_i]$ and $\varphi(x_{i-1}) = f_{i-1}$ and $\psi(x_i) = f_i$. The general algorithm for piecewise interpolation is:

- pick data points $\{(x_i, f_i) \mid i = 0, 1, \ldots, n\}$ such that $a = x_0 < x_1 < \cdots < x_n = b$
- build $\varphi \in \mathcal{C}[a,b]$ on each $[x_{i-1}, x_i]$ such that $\varphi \in \mathbb{P}_m[x_{i-1}, x_i]$ and $\varphi(x_{i-1}) = f_{i-1}$

$$\varphi(x_i) = f(x_i) = f_i \text{ for } i = 0, 1, \ldots, n \to (n+1) \text{ conditions}$$

$$\varphi(x) = \left\{ \begin{array}{cl} a_0^{(1)} + a_1^{(1)}x + \cdots + a_m^{(1)}x^m & \text{on } [x_0, x_1] \\ a_0^{(2)} + a_1^{(2)}x + \cdots + a_m^{(2)}x^m & \text{on } [x_1, x_2] \\ & \vdots \\ a_0^{(n)} + a_1^{(n)}x + \cdots + a_m^{(n)}x^m & \text{on } [x_{n-1}, x_n] \end{array} \right\} n(m+1) \text{ coefficients}$$

- continuity of derivatives on interior points $\{x_i \mid i = 1, 2, \ldots, n-1\}$

$$\left. \begin{array}{rl} \lim_{h\to 0^+} \varphi(x_i - h) = & \lim_{h\to 0^+} \varphi(x_i + h) \\ \lim_{h\to 0^+} \varphi^1(x_i - h) = & \lim_{h\to 0^+} \varphi^1(x_i + h) \\ \vdots & \\ \lim_{h\to 0^+} \varphi^{m-1}(x_i - h) = & \lim_{h\to 0^+} \varphi^{m-1}(x_i + h) \end{array} \right\} m(n-1) \text{ more conditions}$$

- still need $(m-1)$ more conditions

1.7 Linear interpolating splines

Take $n+1$ points such that $a = x_0 < x_1 < \cdots < x_n = b$ and a function $f \in \mathcal{C}[a,b]$. The linear interpolating spline $s_L(x)$ is

$$\boxed{s_L(x) = \left(\frac{x_i - x}{x_i - x_{i-1}}\right) f_{i-1} + \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right) f_i}$$

This is nothing different from connecting each pair of consecutive points with a straight line. Clearly, there will be some error in interpolation since we are approximating f by a set of polynomials in $\mathbb{P}_1$. The error bound can be quantified as

$$||f - s_L|| \leqslant \frac{h^2}{8}||f''|| \qquad \text{where } h = \max_{1\leqslant i\leqslant n} h_i = \max_{1\leqslant i\leqslant n}(x_i - x_{i-1})$$

The proof relies on the error equation introduced in Section 1.5. Substitute $n = 1$ and note how $\max|(x - x_{i-1})(x - x_i)| = h_i^2/4$ where $h_i = x_i - x_{i-1}$. Finally take a maximum over all the i's.

1.8 Cubic splines

This is another case of spline interpolation where $s \in \mathcal{C}^2[x_0, x_n]$ such that $s \in \mathbb{P}_3$ on each $[x_i, x_{i+1}]$.

- interpolation conditions:

$$\begin{aligned}
\text{function value} \rightarrow & \begin{cases} s_i(x_i) = f_i & \text{for } i = 0, 1, \ldots, n-1 \\ s_{n-1}(x_n) = f_n & \end{cases} \\
\text{continuity of } s \rightarrow & \quad s_i(x_{i+1}) = s_{i+1}(x_{i+1}) \quad \text{for } i = 0, 1, \ldots, n-2 \\
\text{continuity of } s' \rightarrow & \quad s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}) \quad \text{for } i = 0, 1, \ldots, n-2 \\
\text{continuity of } s'' \rightarrow & \quad s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}) \quad \text{for } i = 0, 1, \ldots, n-2
\end{aligned}$$

- take polynomials of the form $s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$ for $x \in [x_i, x_{i+1}]$ and $i = 0, 1, \ldots, n-1$
- $4n$ coefficients & $4n - 2$ conditions, need 2 more conditions $\rightarrow s_0''(x_0) = s_{n-1}''(x_n) = 0$

Instead of solving a $4n \times 4n$ matrix, we can make our life a little easier. Take equally spaced knots $h = |x_{i+1} - x_i|$ for $i = 0, 1, \ldots, n-1$. Using the general form for $s_i(x)$, we get

$$s_i(x_i) = f_i \implies \boxed{d_i = f_i} \quad \text{for } i = 0, 1, \ldots, n-1$$

We further define new variables as $\sigma_i = s''(x_i)$ for $i = 0, 1, \ldots, n$. We already know $\sigma_0 = \sigma_n = 0$, thus we have $n - 1$ unknown quantities. We have

$$s_i''(x) = 6a_i(x - x_i) + 2b_i \implies \sigma_i = s_i''(x_i) = 2b_i \implies \boxed{b_i = \frac{\sigma_i}{2}} \tag{1}$$

Using the condition that $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})$, we have

$$6a_i(x_{i+1} - x_i) + 2b_i = \sigma_{i+1} \implies \boxed{a_i = \frac{\sigma_{i+1} - \sigma_i}{6h}} \tag{2}$$

Next, we evaluate $s_i(x)$ at $x = x_{i+1}$ to get

$$f_{i+1} = s_i(x_{i+1}) = a_i h^3 + b_i h^2 + c_i h + d_i \implies \boxed{c_i = \frac{f_{i+1} - f_i}{h} - \frac{h}{6}(2\sigma_i + \sigma_{i+1})} \tag{3}$$

Finally using the continuity of s' i.e. $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$, we get

$$\left.\begin{aligned} s_i'(x) &= 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \\ s_{i+1}'(x) &= 3a_{i+1}(x - x_i)^2 + 2b_{i+1}(x - x_i) + c_{i+1} \end{aligned}\right\} \implies 3a_i h^2 + 2b_i h + c_i = c_{i+1}$$

A little bit of careful manipulation using equations (1), (2) and (3) yields us the recursive relation for $i = 1, \ldots, n-1$:

$$\sigma_{i-1} + 4\sigma_i + \sigma_{i+1} = \frac{6}{h^2}(f_{i-1} - 2f_i + f_{i+1}) \begin{cases} \sigma_0 + 4\sigma_1 + \sigma_2 = \dfrac{6}{h^2}(f_0 - 2f_1 + f_2) \\ \sigma_1 + 4\sigma_2 + \sigma_3 = \dfrac{6}{h^2}(f_1 - 2f_2 + f_3) \\ \vdots \\ \sigma_{n-2} + 4\sigma_{n-1} + \sigma_n = \dfrac{6}{h^2}(f_{n-2} - 2f_{n-1} + f_n) \end{cases}$$

This system of equations can be expressed as a matrix equation which is more convenient to solve:

$$\begin{bmatrix} 4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_{n-3} \\ \sigma_{n-2} \\ \sigma_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} f_0 - 2f_1 + f_2 \\ f_1 - 2f_2 + f_3 \\ f_2 - 2f_3 + f_4 \\ \vdots \\ f_{n-4} - 2f_{n-3} + f_{n-2} \\ f_{n-3} - 2f_{n-2} + f_{n-1} \\ f_{n-2} - 2f_{n-1} + f_n \end{bmatrix}$$

As with linear splines, there is also an error bound associated with cubic splines. This is given by

$$||f - s|| \leqslant Ch^4 ||f^{(iv)}|| \quad \text{where } h = \max_{1 \leqslant i \leqslant n} h_i = \max_{1 \leqslant i \leqslant n} (x_i - x_{i-1}) \text{ and } C = \text{constant}$$

2 Numerical Integration

Given a real function f, we want to compute the integral $\int_a^b f(x)\mathrm{d}x$. While it might seem straightforward, finding an antiderivative is not always easy. Hence, we resort to approximating it.

2.1 Newton-Cotes formula

Let $f : [a, b] \to \mathbb{R}$ and $p \in \mathbb{P}_n$ be the interpolating polynomial. Define the **quadrature points** as $a = x_0 < \cdots < x_n = b$. Then $\int_a^b f(x)\mathrm{d}x$ can be approximated as

$$\int_a^b f(x)\mathrm{d}x \approx \int_a^b p(x)\mathrm{d}x \implies \int_a^b f(x)\mathrm{d}x \approx \int_a^b \sum_{i=0}^n f(x_i)L_i(x)\mathrm{d}x = \sum_{i=0}^n f(x_i) \int_a^b L_i(x)\mathrm{d}x$$

Assume equally spaced intervals such that $x_i = a + ih$. Further let $x = a + th$ for $t \in [0, n]$. We can then express the lagrange polynomials in terms of t.

$$L_i(x) = \prod_{k=0,k\neq i}^n \frac{(x - x_k)}{(x_i - x_k)} = \prod_{k=0,k\neq i}^n \frac{(t - k)}{(i - k)} = \varphi_i(t) \implies \int_a^b L_i(x)\mathrm{d}x = h \int_0^n \varphi_i(t)\mathrm{d}t$$

Defining the quadrature weights as $w_i = \int_0^n \varphi_i(t)\mathrm{d}t$ for $i = 0, \ldots, n$, we get

$$\boxed{\int_a^b f(x)\mathrm{d}x \approx h \sum_{i=0}^n w_i f(x_i)}$$

Note: The weights w_i are dependent only on n and are independent of f, a, b and h. Further, all the w_i's are symmetric i.e. $w_k = w_{n-k}$. Finally, all the weights add up to n i.e. $\sum_{i=0}^n w_i = n$.

2.2 Special cases of the Newton-Cotes formula

Let $\mathcal{I}_f$ be the desired integral and $\mathcal{I}_p = \int_a^b p(x)\mathrm{d}x$ be the approximated integral. We can substitute values of n in the Newton-Cotes formula to get the following cases:

1. **Trapezium rule** $(n = 1)$: $\mathcal{I}_{p_1} = \frac{h}{2}(f(a) + f(b))$
2. **Simpson's $\frac{1}{3}$ rule** $(n = 2)$: $\mathcal{I}_{p_2} = \frac{h}{3}(f(a) + 4f(a+h) + f(b))$
3. Simpson's $\frac{3}{8}$ rule $(n = 3)$: $\mathcal{I}_{p_3} = \frac{3h}{8}(f(a) + 3f(a+h) + 3f(a+2h) + f(b))$
4. Milne's rule $(n = 4)$: $\mathcal{I}_{p_4} = \frac{h}{45}(14f(a) + 64f(a+h) + 24f(a+2h) + 64f(a+3h) + 14f(b))$

2.3 Error in the Newton-Cotes formula

Recall the error equation from Section 1.5. We use it to find the error in the Newton-Cotes formula as follows:

$$
\begin{aligned}
|\mathcal{I}_f - \mathcal{I}_{p_n}| &= \left|\int_a^b (f(x) - p_n(x))\mathrm{d}x\right| \leqslant \int_a^b |f(x) - p_n(x)|\,\mathrm{d}x \\
\therefore |\mathcal{I}_f - \mathcal{I}_{p_n}| &\leqslant \left[\frac{1}{(n+1)!}\max_{\eta\in[a,b]} |f^{(n+1)}(\eta)|\right] \int_a^b \left|\prod_{i=0}^{n}(x - x_i)\right| \mathrm{d}x \\
&= \frac{1}{(n+1)!}||f^{(n+1)}|| \int_a^b \prod_{i=0}^{n} \mid x - x_i \mid \mathrm{d}x
\end{aligned}
$$

– Trapezium rule: $|\mathcal{I}_f - \mathcal{I}_{p_1}| \leqslant \frac{1}{12} \mid\mid f'' \mid\mid (b-a)^3$

– Simpson's rule: $|\mathcal{I}_f - \mathcal{I}_{p_2}| \leqslant \frac{1}{192} \mid\mid f''' \mid\mid (b-a)^4$

As we increase n, some of the weights take negative values. As a result, the error does not converge to zero with n.

2.4 Gaussian quadrature

In order for the error to converge to 0, we must ensure that the weights are all positive. We define the Gaussian quadrature of order n as follows:

$$
\mathcal{G}_n(f) = \sum_{i=0}^{n} W_i f(x_i) \text{ where } W_i = \int_a^b [L_i(x)]^2 \mathrm{d}x = \int_a^b \prod_{k=0,k\neq i}^{n} \left(\frac{x - x_k}{x_i - x_k}\right)^2
$$

The quadrature points are not equally spaced, and are roots of certain polynomials.

$$
\lim_{n\to\infty} |\mathcal{G}_n(f) - \mathcal{I}_f| = 0
$$

2.5 Composite rules

This is very similar to spline interpolation, where we interpolated f by a piecewise cubic over each sub-interval. Here, we divide $[a, b]$ into m sub-intervals of equal length and apply Newton-Cotes on each set of quadrature points.