AMSC 460 Notes

Dennis Chunikhin

December 29, 2024

Contents

1 Polynomial Interpolation

Assume the following for this entire section:

We have an arbitrary function $f(x)$.

- We pick $n+1$ distinct points

$$x_0 \neq x_1 \neq \ldots \neq x_n \in \mathbb{R}$$

- We evaluate the function at these points:

$$y_0 = f(x_0), y_1 = f(x_1), \ldots, y_n = f(x_n)$$

- We want to find n-th degree polynomial

$$p_n(x) = a_0 + a_1 x + \cdots + a_n x^n$$

  such that the polynomial equals $f(x)$ at all x_i:

$$\begin{aligned} p_n(x_0) &= y_0 = f(x_0) \\ p_n(x_1) &= y_1 = f(x_1) \\ &\vdots \\ p_n(x_n) &= y_n = f(x_n) \end{aligned}$$

1.1 Vandermonde Matrix

Notice that we can write the polynomial as a product of a row and column vector:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \begin{pmatrix} 1 & x & x^2 & \cdots & x^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \tag{1}$$

We want this to hold for each x_i and y_i, so we concatenate the row-vectors on the left hand side of equation 1 as rows in a **Vandermonde** matrix and set the product equal to a vector of the y's:

Definition 1.1: Polynomial interpolation using the Vandermonde matrix

To find the coefficients of the interpolating polynomial $p_n(x)$, we need to solve the following linear system of equations for the coefficient vector $\mathbf{a}$:

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

The matrix on the left-hand side of this equation is called the **Vandermonde matrix**.

Theorem 1.1: A solution to the system exists

A solution to the system of equations in definition 1.1 always exists.

Proof of theorem 1.1

If we can show that the Vandermonde matrix is invertible, we prove that a solution exists.

To show that the Vandermonde matrix is invertible, we need to show that its columns are linearly independent. That is, that the only solution to

$$c_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + c_1 \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n^n \end{pmatrix} + c_2 \begin{pmatrix} x_0^2 \\ x_1^2 \\ \vdots \\ x_n^n \end{pmatrix} + \cdots + c_n \begin{pmatrix} x_0^n \\ x_1^n \\ \vdots \\ x_n^n \end{pmatrix} = 0 \tag{3}$$

is $c_0 = c_1 = \cdots = c_n = 0$.

Notice that we can write equation 3 as

$$\begin{pmatrix} c_0 + c_1 x_0 + c_2 x_0^2 + \cdots + c_n x_0^n \\ c_0 + c_1 x_1 + c_2 x_1^2 + \cdots + c_n x_1^n \\ \vdots \\ c_n + c_1 x_n + c_2 x_n^2 + \cdots + c_n x_n^n \end{pmatrix} = 0 \tag{4}$$

Let's denote the polynomial $c_0 + c_1x + c_2x^2 + \cdots + c_nx^n = C(x)$.

Notice that each row of equation 4 is the polynomial C evaluated at x_i, and is equal to zero. This implies that each x_i is a root of the polynomial C. However C is an n-th degree polynomial, meaning it must have at most n distinct roots, or be identically zero ($C(x) = 0$). Since there are $n+1$ distinct x_i, we must have $C(x) = c_0+c_1x+\cdots+c_nx^n = 0 \implies c_0 = c_1 = \cdots = c_n = 0$.

Thus the columns are linearly independent, the Vandermonde matrix is invertible, and a solution to the linear equation in definition 1.1 exists!

1.2 Lagrange polynomials

Alternatively we can interpolate using **Lagrange polynomials:**

Definition 1.2: Lagrange Polynomials

The k-th Lagrange polynomial is:

$$l_k(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}$$

This polynomial is constructed to be zero at all $x_i, i \neq k$ and 1 at x_k. This makes it very useful for finding interpolating polynomials mathematically:

Theorem 1.2: Interpolation using Lagrange polynomials

The interpolating polynomial is given by:

$$p_n(x) = y_0l_0(x) + y_1l_1(x) + \cdots + y_nl_n(x)$$

This is inefficient to compute, but is mathematically useful.

2 Accuracy of Interpolation

Take the same setup as in section 1:

We are interpolating arbitrary function $f(x)$ on the interval $[a,b] \subseteq \mathbb{R}$ at $n+1$ points with an n-th degree polynomial $p(x)$.

We make the additional assumption that the function f is defined on the interval $[a,b]$ and that $f^{(n+1)}$ is well defined on $[a,b]$.

Construction 2.1: Setup

Pick an arbitrary point in the interval

$$\xi \in [a,b]$$

Define

$$L(x) = (x - x_0)(x - x_1)\dots(x - x_n) \tag{5}$$

This is the "simplest" polynomial that has the roots $x_0, \dots, x_n$.

Define

$$\alpha = \frac{f(\xi) - p(\xi)}{L(\xi)} \tag{6}$$

Finally, define

$$F(x) = f(x) - p(x) - \alpha L(x) \tag{7}$$

All of these constructions designed so that F has zeros at $x_0, \dots, x_n$ and ξ:

$$F(x_0) = F(x_1) = \dots = F(x_n) = F(\xi) = 0 \tag{8}$$

Notice that by equation 8, F has $n+2$ roots in $[a,b]$.

By Rolle's theorem,

$$\begin{aligned} F'(x) &\text{ has } n+1 \text{ roots} \\ F''(x) &\text{ has } n \text{ roots} \\ &\vdots \\ F^{(n+1)}(x) &\text{ has } 1 \text{ root} \end{aligned}$$

Construction 2.2: η

Let $\eta \in [a,b]$ be the root of $F^{(n+1)}$

That is,
$$F^{(n+1)}(\eta) = 0 \tag{9}$$

By the definition of F (equation 7),

$$F^{(n+1)}(\eta) = f^{(n+1)}(\eta) - p^{(n+1)}(\eta) - \alpha L^{(n+1)}(\eta) = 0 \tag{10}$$

Since the interpolating polynomial p is an n-th degree polynomial, we have $p^{(n+1)} = 0$.

Thus equation 10 becomes

$$f^{(n+1)}(\eta) - \alpha L^{(n+1)}(\eta) = 0 \tag{11}$$

Notice that by the definition of L (equation 5), L is an $n+1$ degree polynomial with leading term $x^{(n+1)}$, so

$$L^{(n+1)} = (n+1)! \tag{12}$$

Thus equation 11 becomes

$$f^{(n+1)}(\eta) - \alpha(n+1)! = 0 \tag{13}$$

which gives us

$$\begin{align} f^{(n+1)}(\eta) &= \alpha(n+1)! \tag{14}\\ &= \frac{f(\xi) - p(\xi)}{L(\xi)}(n+1)! \tag{15}\end{align}$$

This finally gives us the result

$$\begin{align} f(\xi) - p(\xi) &= \frac{f^{(n+1)}(\eta)}{(n+1)!}L(\xi) \tag{16}\\ &= \frac{f^{(n+1)}(\eta)}{(n+1)!}(\xi - x_0)(\xi - x_1)\ldots(\xi - x_n) \tag{17}\end{align}$$

Theorem 2.1: Polynomial interpolation error

The error of polynomial interpolation at any point ξ is given by

$$f(\xi) - p(\xi) = \frac{f^{(n+1)}(\eta)}{(n+1)!}(\xi - x_0)(\xi - x_1)\dots(\xi - x_n) \tag{18}$$

Corollarly 2.1: Polynomial interpolation error bound

The error of polynomial interpolation is bounded by

$$\left|f - p\right| \leq \left|\frac{f^{(n+1)}(\eta)}{(n+1)!}\right| \max_{\xi \in [a,b]} \left|(\xi - x_0)\dots(\xi - x_n)\right| \tag{19}$$

This is a direct consequence of theorem 2.1 (take the absolute value of the result).

Algorithm 2.1: Efficiently evaluating a polynomial

Evaluate a polynomial $p(x) = a_0 + a_1 x + \dots + a_n x^n$ at some point x.

We rewrite $p(x) = a_0 + x(a_1 + x(a_2 + \dots))$.

Algorithm 1 Evaluate a polynomial

```
1: $p = a_n$
2: for $i = n - 1$ to 0 do
3:     $p = a_i + xp$
4: end for
```

2.1 Piecewise Interpolation

Break interval into n segments and do polynomial interpolation on each of the segments.

Let $h = \max_{1 \leq i \leq n} \left|x_i - x_{i-1}\right|$ denote the maximum width of the segments.

Then, by adding up the error for each of the intervals as given by theorem 2, we get:

Theorem 2.2: Linear piecewise interpolation error

The error bound for linear (degree 1) piecewise interpolation $l(x)$ using n intervals with maximum interval width h is given by

$$\left|f - l\right| \leq \max_{\eta \in [x_0, x_n]} \frac{\left|f''(\eta)\right|}{2} \frac{h^2}{4} \tag{20}$$

Benefit: We reduce error as we add intervals (as $h \to 0$).

Con: The derivative is not continuous.

3 Least Squares Fitting

Consider data points consisting of a function f which, evaluated at points $x_0, x_1, \ldots, x_n$, yields $y_0, y_1, \ldots, y_n$.

We want to approximate f using these datapoints with polynomial $\hat{f}$ of degree $m < n$.

As in section 1, this problem can be represented with the Vandermonde matrix

Construction 3.1: Least squares fitting problem

Consider fitting polynomial

$$\hat{f}(x) = a_0 + a_1 x + \cdots + a_m x^m \tag{21}$$

Ideally, we want $\hat{f}(x_i) = y_i$ for all $i = 1, \ldots, n$, which we write down as the following matrix equation

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^m \\ 1 & x_1 & x_1^2 & \cdots & x_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{22}$$

We denote the big matrix A and write this equation more concisely as

$$Aa = y \tag{23}$$

Unless we get incredibly lucky, the system $Aa = y$ has no solutions. Instead we want to find coefficient vector a that minimizes the error.

Definition 3.1: l^2 norm

The l^2 norm is the typical distance metric we are used to. It is defined for a vector $\mathbf{x}$ as

$$\left\| \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \right\|_2 = \sqrt{x_1^2 + \cdots + x_n^2} \tag{24}$$

Note that

$$\mathbf{x}^T \mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = x_1^2 + \cdots + x_n^2 = \|\mathbf{x}\|_2^2 \tag{25}$$

Arguably this is a better (more generalizable) definition for l^2 norm.

We define the error using this distance metric:

$$\|y - Aa\|_2 \tag{26}$$

We want to minimize this. Note that minimizing this is equivalent to minimizing

$$\begin{aligned}\|y - Aa\|_2^2 &= (y - Aa)^T(y - Aa) && (27)\\ &= (y^T - (Aa)^T)(y - Aa) && (28)\\ &= y^Ty - (Aa)^Ty - y^T(Aa) + (Aa)^T(Aa) && (29)\end{aligned}$$

We note that for two vectors x, y, $x^Ty = y^Tx$ (this may look familiar, since $x \cdot y = x^Ty$ is a common way to define the dot product, and the dot product is commutative).

We also note that $(Aa)^T = a^TA^T$

Thus equation 29 becomes

$$y^Ty - 2y^TAa + a^TA^TAa = y^Ty - 2(A^Ty)^Ta + a^TA^TAa \tag{30}$$

To mimize this, we set its gradient (with respect to a) equal to 0:

$$\begin{aligned}\nabla(y^Ty - 2(A^Ty)^Ta + a^TA^TAa) &= \nabla(-2(A^Ty)^Ta + a^TA^TAa) && (31)\\ &= -2A^Ty + 2A^TAa = 0 && (32)\end{aligned}$$

The calculation of the gradient can be tedious to show but is an approachable exercise in multivariable calculus. Thus we state it without proof.

Equation 32 gives us

$$A^TAa = A^Ty \tag{33}$$

Theorem 3.1: The normal equations

The least squares polynomial fit for equation $Aa = y$ is given by the solution to

$$A^TAa = A^Ty \tag{34}$$

This system of equations is called the **normal equations**.

4 Numerical Integration

Given arbitrary function $f(x)$ defined on $[a, b] \subseteq \mathbb{R}$, we want to approximate

$$I(f) = \int_a^b f(x)\mathrm{d}x \tag{35}$$

We do this by interpolating the function and then taking the integral of the interpolating polynomial. The specific procedure (crucially, the points used for the interpolation) is called a **quadrature rule**.

Definition 4.1: Quadrature

We approximate

$$I(f) \approx Q(f)$$

using a **quadrature rule** Q, in general given by

$$Q(f) = (b - a)\sum_{k=1}^{m} w_k f(x_k) \tag{36}$$

for some weights w_k and some arbitrarily chosen points x_k at which we evaluate the function f.

The weights come about by integration as we will soon see.

4.1 General Quadrature Rule

We interpolate at points $x_0, x_1, \ldots, x_k$ with k-th degree interpolating $p_k(x)$.

Using Lagrange polynomials, we have:

$$p_k(x) = \sum_{i=0}^{k} f(x_i)l_i(x) \tag{37}$$

So the quadrature rule is:

$$Q(f) = \sum_{i=0}^{k} f(x_i)\int_a^b l_i(x)\mathrm{d}x \tag{38}$$

Theorem 4.1: General quadrature rule

The general degree k-th degree quadrature rule is given by

$$Q(f) = \sum_{i=0}^{k} f(x_i) \int_a^b l_i(x)\mathrm{d}x \tag{39}$$

Definition 4.2: Newton-Cotes rules

If the points of interpolation $x_0, \ldots, x_k$ are uniformly spaced, the quadrature rules are called **Newton-Cotes rules**.

4.2 Rectangle Rule

Take a 0-degree interpolant $p_0(x) = f(a)$ of $y_0 = f(a)$ at $x_0 = a$.

The quadrature rule is given by

$$Q_0(f) = \int_a^b p_0(x)\mathrm{d}x = \int_a^b f(a)\mathrm{d}x = (b-a)f(a) \tag{40}$$

Definition 4.3: Rectangle rule

The rectangle rule is the 0-th degree Newton-Cotes method interpolated at the point $x_0 = a$.

The quadrature rule is given by

$$Q_0(x) = (b-a)f(a) \tag{41}$$

4.3 Trapezoid Rule

Take a degree 1 Newton-Cotes interpolant $p_1(x)$ interpolating f at points $x_0 = a$ and $x_1 = b$.

$p_1(x)$ is a line which intersects f at $x = a$ and $x = b$. Thus,

$$p_1(x) = \frac{f(b) - f(a)}{b - a}(x - a) + f(a) \tag{42}$$

Therefore the quadrature rule is given by

$$\begin{aligned} Q_1(f) &= \int_a^b p_1(x)\mathrm{d}x && (43)\\ &= \frac{f(b) - f(a)}{2(b - a)}(x - a)^2 + f(a)x\Big|_a^b && (44)\\ &= (b - a)\frac{f(a) + f(b)}{2} && (45) \end{aligned}$$

Definition 4.4: Trapezoid rule

The trapezoid rule is the 1-st degree Newton-Cotes method interpolated at the points $x_0 = a$ and $x_1 = b$.

The quadrature rule is given by

$$Q_1(f) = (b - a)\frac{f(a) + f(b)}{2} \tag{46}$$

4.4 Simpson's Rule

Take a degree 2 Newton-Cotes interpolant $p_2(x)$ interpolating f at the points $x_0 = a$, $x_1 = \frac{a+b}{2}$, and $x_2 = b$.

We can follow the general quadrature rule procedure, wherein we first write down $p_2(x)$ using lagrange polynomials:

$$p_2(x) = f(a)l_0(x) + f(\frac{a+b}{2})l_1(x) + f(b)l_2(x) \tag{47}$$
$$= f(a)\frac{(x-\frac{a+b}{2})(x-b)}{(a-\frac{a+b}{2})(a-b)} \tag{48}$$
$$+ f(\frac{a+b}{2})\frac{(x-a)(x-b)}{(\frac{a+b}{2}-a)(\frac{a+b}{2}-b)} \tag{49}$$
$$+ f(b)\frac{(x-a)(x-\frac{a+b}{2})}{(b-a)(b-\frac{a+b}{2})} \tag{50}$$

The quadrature rule is given

$$Q_2(x) = \int_a^b p_2(x)\mathrm{d}x = f(a)\frac{1}{6} + f\left(\frac{a+b}{2}\right)\frac{2}{3} + f(b)\frac{1}{6} \tag{51}$$

The process of simplification and integration of $p_2(x)$ is tedious to typeset so it is omitted, but is in general not too difficult to do.

Definition 4.5: Simpson's rule

Simpson's rule is the 2-nd degree Newton-Cotes method interpolated at the points $x_0 = a$, $x_1 = \frac{a+b}{2}$, $x_2 = b$.

The quadrature rule is given by

$$Q_2(f) = f(a)\frac{1}{6} + f\left(\frac{a+b}{2}\right)\frac{2}{3} + f(b)\frac{1}{6} \tag{52}$$

4.5 Accuracy of Trapezoid Rule

The trapezoid rule uses piecewise linear interpolation. We know from theorem 2.1 that the error in piecewise linear interpolation is given by

$$f(x) - p_1(x) = \frac{f''(\eta_x)}{2}(x-a)(x-b) \tag{53}$$

The quadrature error is then

$$\begin{aligned}\left|\text{Error}\right| &= \left|\int_a^b f(x) - p_1(x)\mathrm{d}x\right| && (54)\\ &= \left|\int_a^b \frac{f''(\eta_x)}{2}(x-a)(x-b)\mathrm{d}x\right| && (55)\\ &\leq \int_a^b \left|\frac{f''(\eta_x)}{2}(x-a)(x-b)\right| \mathrm{d}x && (56)\\ &\leq \max_{\eta\in[a,b]} \frac{|f''(\eta)|}{2} \int_a^b \left|(x-a)(x-b)\right| \mathrm{d}x && (57)\end{aligned}$$

Notice that the polynomial $(x-a)(x-b)$ is negative on the interval $x \in [a,b]$. Therefore equation 57 becomes

$$\left|\text{Error}\right| \leq -\max_{\eta\in[a,b]} \frac{|f''(\eta)|}{2} \int_a^b (x-a)(x-b)\mathrm{d}x \tag{58}$$

We integrate equation 58 by parts to get

$$\begin{aligned}\left|\text{Error}\right| &\leq -\max_{\eta\in[a,b]} \frac{|f''(\eta)|}{2}\left[\frac{1}{2}(x-a)^2(x-b)\Big|_a^b - \frac{1}{2}\int_a^b (x-a)^2\mathrm{d}x\right] && (59)\\ &= \max_{\eta\in[a,b]} \frac{|f''(\eta)|}{2}\frac{1}{6}(x-a)^3\Big|_a^b && (60)\\ &= \max_{\eta\in[a,b]} \frac{|f''(\eta)|}{12}(b-a)^3 && (61)\end{aligned}$$

Theorem 4.2: Trapezoid rule error bound

The Quadrature error of the trapezoid rule is bounded by

$$\max_{\eta\in[a,b]} \frac{|f''(\eta)|}{12}(b-a)^3 \tag{62}$$

4.6 More on General Quadrature

Theorem 4.3: Quadrature of even degree perfectly integrates polynomials of 1 degree higher

If k is even and f is a polynomial of degree $k+1$, then the k-th degree quadrature gives the exact integral:

$$Q_k(x) = I(x) \tag{63}$$

Sketch of proof of theorem 4.3 for the Simpson's rule

We note that I and Q are linear (it is a good exercise to verify this).

Consider $f = \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \alpha_0$.

$$\begin{aligned} I(f) &= \alpha_3 I(x^3) + I(\alpha_2 x^2 + \alpha_1 x + \alpha_0) && (64) \\ &= \alpha_3 I(x^3) + Q(\alpha_2 x^2 + \alpha_1 x + \alpha_0) && (65) \end{aligned}$$

and we can verify by integrating f and computing the quadrature rule that $I(x^3) = Q(x^3)$.

Thus we get $I(x) = Q(x)$.

5 Piecewise Quadrature Rules

We split the interval $[a, b]$ into sub-intervals and use a quadrature rule on each sub-interval and adding the results up.

Let's say we split the interval at points $\{x_i\}$ into sub-intervals $[x_i, x_{i+1}]$.

5.1 Composite Trapezoidal Rule

Let us apply the trapezoidal quadrature rule $Q_i(f)$ to f on each sub-interval $[x_i, x_{i+1}]$.

Call $I_i(f)$ the true integral of f on each sub-interval $[x_i, x_{i+1}]$, that is

$$I_i(f) = \int_{x_i}^{x_{i+1}} f(x)\mathrm{d}x \tag{66}$$

Since we can see that

$$I(f) = \sum_{i=1}^{n} I_i(f) \tag{67}$$

the composite trapezoidal quadrature rule is given by

$$Q(f) = \sum_{i=1}^{n} Q_i(f) \tag{68}$$

5.2 Piecewise Quadrature Error Bounds

5.2.1 Composite Trapezoidal Rule

We will now estimate the error bound for the composite trapezoidal rule.

$$\begin{aligned} \big|\text{Error}\big| &= |I(f) - Q(f)| && (69) \\ &= \left| \sum_{i=1}^{n} \big(I_i(f) - Q_i(f)\big) \right| && (70) \\ &\leq \sum_{i=1}^{n} |I_i(f) - Q_i(f)| && (71) \end{aligned}$$

We substitute the error bound for the trapezoidal quadrature rule (theorem 4.2) into equation 71 to get:

$$\begin{aligned} \big|\text{Error}\big| &\leq \sum_{i=1}^{n} \max_{\eta \in [x_i, x_{i+1}]} \frac{|f''(\eta)|}{12} (x_i - x_{i-1})^3 && (72) \\ &\leq \max_{\eta \in [a,b]} \frac{|f''(\eta)|}{12} \sum_{i=1}^{n} (x_i - x_{i-1})^3 && (73) \end{aligned}$$

Let's now assume that all of the sub-intervals are of size h, that is $\forall x_{i+1}, x_i = h$.

Note that since there are n intervals, $h = \frac{b-a}{n}$

Equation 73 then becomes:

$$\left|\text{Error}\right| \leq \max_{\eta\in[a,b]} \frac{|f''(\eta)|}{12} \sum_{i=1}^{n} h^3 \tag{74}$$

$$\leq \max_{\eta\in[a,b]} \frac{|f''(\eta)|}{12} n h^3 \tag{75}$$

$$= \max_{\eta\in[a,b]} \frac{|f''(\eta)|}{12} (b-a) h^2 \tag{76}$$

Theorem 5.1: Piecewise trapezoidal quadrature rule error

Given sub-intervals of equal width h, the error in the piecewise trapezoidal quadrature rule is bounded by

$$\max_{\eta\in[a,b]} \frac{|f''(\eta)|}{12} (b-a) h^2 \tag{77}$$

Error decreases as we decrease interval width h. Specifically, the error is $O(h^2)$.

5.2.2 Composite Simpson's Rule

The same analysis can be performed for piecewise Simpson's rule, though it is tedious and therefore omitted.

Theorem 5.2: Piecewise Simpson's quadrature rule error

Given sub-intervals of equal width h, the error in the piecewise Simpson's quadrature rule is bounded by

$$\max_{\xi\in[a,b]} \frac{\left|f^{(4)}(\xi)\right|}{2880} (b-a) h^4 \tag{78}$$

For composite Simpson's rule, the error reduces as $O(h^4)$!

5.3 Estimating Error in Piecewise Quadrature

Consider composite quadrature on interval $[\alpha, \beta]$

5.3.1 Composite Trapezoidal Rule

We will estimate the error bound of the trapezoid rule by performing it on different sub-intervals.

Let Q_1 be the composite trapezoidal quadrature rule on n sub-intervals of width h.

Let Q_2 be the same quadrature rule but on $2n$ sub-intervals of width $\frac{h}{2}$.

The error bounds are

$$|I(f) - Q_1(f)| \leq \epsilon_1 = \frac{\beta - \alpha}{12} \max_{\xi \in [\alpha,\beta]} \left|f''(\xi)\right| h^2 \tag{79}$$

$$|I(f) - Q_2(f)| \leq \epsilon_2 = \frac{\beta - \alpha}{12} \max_{\xi \in [\alpha,\beta]} \left|f''(\xi)\right| \left(\frac{h}{2}\right)^2 \tag{80}$$

Notice that

$$\epsilon_2 = \frac{1}{4}\epsilon_1 \tag{81}$$

That is,

$$(I - Q_1) \approx 4(I - Q_2) \tag{82}$$

$$\implies (I - Q_1) - (I - Q_2) \approx 3(I - Q_2) \tag{83}$$

$$\implies (I - Q_2) \approx \frac{1}{3}(Q_2 - Q_1) \tag{84}$$

Theorem 5.3: Composite trapezoidal rule error estimate

Let Q_1 be the composite trapezoidal quadrature rule on n sub-intervals of width h.
Let Q_2 be the same quadrature rule but on $2n$ sub-intervals of width $\frac{h}{2}$.
The error bound of Q_2 is approximately

$$(I - Q_2) \approx \frac{1}{3}(Q_2 - Q_1) \tag{85}$$

5.3.2 Composite Simpson's Rule

The same analysis can be performed for the composite Simpson's quadrature rule.

Let Q_1 be the composite Simpson's quadrature rule on n sub-intervals of width h.

Let Q_2 be the same quadrature rule but on $2n$ sub-intervals of width $\frac{h}{2}$.

The error bounds are

$$|I(f) - Q_1(f)| \le \epsilon_1 = (\beta - \alpha) \max_{\xi\in[\alpha,\beta]} \frac{|f''(\xi)|}{2880} h^4 \tag{86}$$

$$|I(f) - Q_2(f)| \le \epsilon_2 = (\beta - \alpha) \max_{\xi\in[\alpha,\beta]} \frac{|f''(\xi)|}{2880} \left(\frac{h}{2}\right)^4 \tag{87}$$

Notice that

$$\epsilon_2 = \frac{1}{16}\epsilon_1 \tag{88}$$

That is,

$$(I - Q_1) \approx 16(I - Q_2) \tag{89}$$

$$\implies (I - Q_1) - (I - Q_2) \approx 15(I - Q_2) \tag{90}$$

$$\implies (I - Q_2) \approx \frac{1}{15}(Q_2 - Q_1) \tag{91}$$

Theorem 5.4: Composite Simpson's rule error estimate

Let Q_1 be the composite Simpson's quadrature rule on n sub-intervals of width h.
Let Q_2 be the same quadrature rule but on $2n$ sub-intervals of width $\frac{h}{2}$.
The error bound of Q_2 is approximately

$$(I - Q_2) \approx \frac{1}{15}(Q_2 - Q_1) \tag{92}$$

5.4 The Adaptive Quadrature Algorithm

We can use these estimated error bounds to create an adaptive algorithm that computes a quadrature rule to an arbitrary given tolerance.

Algorithm 5.1: Adaptive Quadrature

A recursive algorithm for performing quadrature of given function f on interval $[\alpha, \beta]$ to an arbitrary error tolerance τ.

Algorithm 2 Adaptive Quadrature

Input: α, β, f, τ

Output: Q

```
1:  function ADAPTQ($\alpha, \beta, f, \tau$)
2:      Compute $Q_1$                                  ▷ Intervals of width $h$
3:      Compute $Q_2$                                  ▷ Intervals of width $h/2$

4:      Compute $\epsilon = \frac{1}{3}(Q_2 - Q_1)$    ▷ for trapezoidal rule
5:      or
6:      Compute $\epsilon = \frac{1}{15}(Q_2 - Q_1)$   ▷ for Simpson's rule

7:      if $\epsilon \leq \tau$ then
8:          $Q = Q_1$
9:      else
10:         $Q = $ ADAPTQ($\alpha, \frac{\alpha+\beta}{2}, f, \frac{\tau}{2}$) + ADAPTQ($\frac{\alpha+\beta}{2}, \beta, f, \frac{\tau}{2}$)
11:     end if

12:     return $Q$
13: end function
```

6 Gaussian Quadrature

Recall that by theorem 4.3, a Newton-Cotes quadrature rule of degreen $m-1$ (using m points) produces the exact integral of a degree m polynomial.

By selecting points to interpolate more cleverly, we can do better.

6.1 An Orthonormal Basis for Polynomials

We must first discuss the vector space of degree m polynomials.

Definition 6.1: Polynomial inner product

Define the inner product of two functions $f(x)$ and $g(x)$ on interval $[a,b]$ by

$$\langle f,g\rangle = \int_a^b f(x)g(x)\mathrm{d}x \tag{93}$$

Definition 6.2: Orthogonality of polynomials

We say that two polynomials $p(x)$ and $q(x)$ are orthogonal if

$$\langle p,q\rangle = \int_a^b pq\mathrm{d}x = 0 \tag{94}$$

Consider the vector space of polynomials of degree m or less (often denoted $\mathcal{P}_m(\mathbb{R})$).

We begin with a standard basis of this vector space:

$$\{1, x, x^2, \dots, x^m\} \tag{95}$$

Then use the Gram-Schmidt process to create an orthonormal basis

$$\{\psi_0, \dots, \psi_m\} \tag{96}$$

Where each ψ_i is an i-th degree polynomial (note that if our interval is $[-1,1]$, these are called the Legendre polynomials).

Consider a k-th degree polynomial q. Note that since $\psi_0, \dots, \psi_m$ form a basis, we can write

$$q = c_0\psi_0 + \dots + c_m\psi_m \tag{97}$$

for some constants $c_0, \dots, c_m \in \mathbb{R}$.

Since for all $i > k$, ψ_i is a polynomial of order greater than k, meaning that $\forall i > k, c_i = 0$.

Thus q is a linear combination of $\psi_0, \dots, \psi_k$, meaning that

$$\forall j > k, \langle \psi_j, q\rangle = 0 \tag{98}$$

6.2 Zeros of the Inner Product

Lemma 6.1: The zeros of ψ_m lie in (a, b)

All roots of ψ_m are real and lie in (a, b).

Proof of lemma 6.1

Suppose ψ_m has k roots in (a, b).

Denote these roots $x_1, \ldots, x_k$.

Suppose $k < m$.

Let $q(x) = (x - x_1)(x - x_2) \cdots (x - x_k)$.

q is a degree k polynomial, and since by assumption $k < m$, ψ_m is orthogonal to q. That is,

$$\int_a^b \psi_m(x) q(x) \mathrm{d}x = 0 \tag{99}$$

Next we use a neat parity argument. Note that since $x_1, \ldots, x_k$ are the only roots of ψ_m on (a, b), ψ_m changes sign at and only these points in (a, b). Note that these are also the only roots of q, so q also changes sign at and only at these points. Because of this, we can write

$$\psi_m(x) = q(x) s(x) \tag{100}$$

where $s(x)$ does not change sign in (a, b).

Consider

$$\langle \psi_m(x), q(x) \rangle = \int_a^b \psi_m(x) q(x) \mathrm{d}x \tag{101}$$

$$= \int_a^b q(x)^2 s(x) \mathrm{d}x \tag{102}$$

Since neither $q(x)^2$ nor $s(x)$ change sign on (a, b), and since neither are identically zero,

$$\langle \psi_m(x), q(x) \rangle = \int_a^b q(x)^2 s(x) \mathrm{d}x \neq 0 \tag{103}$$

but this is in contradiction with equation 99.

Thus, we must have $k = m$. That is, ψ_m must have all of its m roots in (a, b).

6.3 Gaussian Quadrature

Definition 6.3: Gaussian Quadrature

We want to interpolate f at m points using a polynomial of degree $m-1$.
Start with ψ_m (degree m).
By lemma 6.1, all roots of ψ_m lie in $[a, b]$.
Denote the roots of ψ_m by $x_1, \dots, x_m$.
These are the points we use for interpolation.
That is, take p_{m-1} the $m-1$ degree interpolating polynomial that interpolates f at $x_1, \dots, x_m$.
The quadrature rule is given by

$$Q(f) = \int_a^b p_{m-1}(x)\mathrm{d}x \tag{104}$$

$$= w_1 f(x_1) + w_2 f(x_2) + \cdots + w_m f(x_m) \tag{105}$$

where

$$w_j = \int_a^b l_j(x)\mathrm{d}x \tag{106}$$

Theorem 6.1: Gaussian quadrature is perfect for polynomials of degree $2m-1$

If f is a polynomial of degree $2m-1$, then $Q(f) = I(f)$

Proof of theorem 6.1

Take f a polynomial of degree $2m-1$.

We can write

$$f(x) = \psi_m(x)g(x) + r(x) \tag{107}$$

where g is of degree no more than $(2m-1)-m = m-1$ and r is of degreeno more than $m-1$.

Then

$$\begin{aligned} Q(f) &= Q(\psi_m q + r) && (108)\\ &= Q(\psi_m q) + Q(r) && (109)\\ &= w_1\psi_m(x_1)q(x_1) + \cdots + w_m\psi_m(x_m)q(x_m) + Q(r) && (110)\\ &= w_1 \cdot 0 \cdot q(x_1) + \cdots + w_m \cdot 0 \cdot q(x_m) + Q(r) && (111)\\ &= Q(r) && (112)\end{aligned}$$

Since r is of degree no more than $m-1$, $Q(r) = I(r)$. Thus,

$$Q(f) = I(r) \tag{113}$$

Now consider

$$I(f) = I(\psi_m q + r) = I(\psi_m q) + I(r) \tag{114}$$

By ortogonality $I(\psi_m q) = 0$, giving us

$$I(f) = I(r) \tag{115}$$

So by equations 113 and 115,

$$Q(f) = I(f) \tag{116}$$

Theorem 6.2: Perfectly interpolating a $2m-1$ degree polynomial is the best we can do

There is no quadrature rule Q such that $Q(f) = I(f)$ for all polynomials of degree $2m$.

Proof of theorem 6.2

Let $x_1, \ldots, x_m$ be the m distinct points at which the polynomial is interpolated.

Consider the $2m$ degree polynomial

$$f(x) = (x - x_1)^2 \cdots (x - x_m)^2 \tag{117}$$

Since $\forall x \in \mathbb{R}, f(x) \geq 0$ and is not identically 0.

$$I(f) = \int_a^b f(x)\mathrm{d}x > 0 \tag{118}$$

Let $p_{m-1}(x)$ be the degree $m-1$ polynomial interpolating f at $x_1, \ldots, x_m$.

$$p_{m-1} = f(x_1)l_1(x) + \cdots + f(x_m)l_m(x) \tag{119}$$

$$= 0 \cdot l_1(x) + \cdots + 0 \cdot l_m(x) = 0 \tag{120}$$

Therefore

$$Q(f) = \int_a^b p_{m-1}\mathrm{d}x = \int_a^b 0\mathrm{d}x = 0 \tag{121}$$

But since by equation 118, $I(f) > 0$,

$$Q(f) \neq I(f) \tag{122}$$

7 Quadrature in Higher Dimensions

Consider as an example quadrature in 2 dimensions.

We want to approximate

$$\int_a^b \int_c^d f(x,y)\mathrm{d}x\mathrm{d}y \tag{123}$$

Define

$$I_x(f) = \int_a^b f(x,y)\mathrm{d}y \approx Q(x) = \sum_{j=1}^{m} w_j f(x, y_j) \tag{124}$$

Then

$$I(f) = \int_a^b I_x \mathrm{d}x \approx \sum_{j=1}^{m} \sum_{k=1}^{m'} w_j v_k f(x_k, y_j) \tag{125}$$

Definition 7.1: 2 dimensional quadrature

To estimate

$$I(f) = \int_a^b \int_c^d f(x, y) \mathrm{d}x \mathrm{d}y \tag{126}$$

we use quadrature

$$Q(f) = \sum_{j=1}^{m} \sum_{k=1}^{m'} w_j v_k f(x_k, y_j) \tag{127}$$

Any 1 dimensional quadrature rule can be used for the weights w_j and v_k.

We can also see that this procedure generalizes to arbitrary dimensions.

7.1 Cost of 1D Quadrature

Definition 7.2: Problem size

Define the **problem size**, denoted N, of a quadrature rule as the number of function evaluations needed.

For quadrature using m points in the interval, we need m function evauations (1 at each point), so the problem size is $N = m$.

Thus the total cost of a quadrature rule is $O(m) = O(N)$.

We wan tot find the total cost to achieve a given error tolerace:

$$|I(f) - Q(f)| \leq \tau \tag{128}$$

For the composite trapezoidal rule, $\tau \propto h^2 \propto \frac{1}{m^2} = \frac{1}{N^2}$

We want

$$\frac{1}{N^2} \leq \tau \tag{129}$$

$$\implies N \geq \frac{1}{\sqrt{\tau}} \tag{130}$$

Thus

$$O(N) = \frac{1}{\sqrt{\tau}} \tag{131}$$

7.2 Cost of 2D Quadrature

Consider the composite trapezoidal rule with m intervals in each direction.

We need m^2 function evalulations, so the problem size is $N = m^2$.

The width of each interval is proportional to h, so the area of each box on the domain is proportional to h^2.

The error on the whole domain is proportional to h^2.

We want $h^2 \leq \tau$.

Since $N = m^2 \propto \frac{1}{h^2}$, the condition becomes

$$\frac{1}{N} \leq \tau \tag{132}$$

$$\implies N \geq \frac{1}{\tau} \tag{133}$$

So the cost is

$$O(N) = \frac{1}{\tau} \tag{134}$$

We can see that this logic generalizes to any dimension.

Theorem 7.1: Cost of composite trapezoidal quadrature in d dimensions

The cost of quadrature in d dimensions with given error tolerance τ with respect to problem size N is

$$O(N) = \frac{1}{\tau^{d/2}} \tag{135}$$

7.3 Integration via Sampling Methods

Notice that we can think of a vector of the points we are using for quadrature as a random variable

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \tag{136}$$

Notice that the sample mean/expectation value is a quadrature rule!

Let's denote the sample mean quadrature rule by $\hat{E}(f)$.

From statistics we know that that the error given n_s samples is

$$I(f) - \hat{E}(f) \propto \frac{1}{\sqrt{n_s}} \tag{137}$$

Thereofore if we want the error to be below tolerance τ, we require

$$\tau \leq \sqrt{n_s} \tag{138}$$

It is interesting to ask when the composite trapezoidal rule is more efficient than the statistical method. This equates to asking for what d is the following inequality satisfied

$$N^{2/d} \geq \sqrt{n_s} \tag{139}$$

$$\implies N^{4/d} \geq n_s \tag{140}$$

This shows us that the composite trapezoidal rule will be more efficient when $d \leq 4$.

Theorem 7.2: Statistical method more efficient for high dimensions

The composite trapezoidal quadrature rule is more efficient than the statistical rule for dimensions $d \leq 4$. Otherwise the statistical rule is more efficient.

The composite Simpson's quadrature rule is more efficient than the statistical rule for dimensions $d \leq 8$. Otherwise the statistical rule is more efficient.

8 Gaussian Elimination

Given a linear system of equations $Ax = b$, where A is a matrix and x, b are vectors, we want to compute x.

We can do this using Gaussian elimination (also known as row-reduction), which we represent as a series of matrix multiplications.

Given matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{12n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \tag{141}$$

we represent the first step of Gaussian elimination as multiplication by the following matrix:

Construction 8.1: L_1

We construct the matrix

$$L_1 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -a_{21}/a_{11} & 1 & & & \\ -a_{31}/a_{11} & & 1 & & \\ \vdots & & & \ddots & \\ -a_{n1}/a_{11} & & & & 1 \end{pmatrix} \tag{142}$$

We can alternatively view this as a block matrix

$$L_1 = \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{l_1} & I_{n-1} \end{array}\right) \tag{143}$$

where

$$\mathbf{l_1} = \begin{pmatrix} -a_{21}/a_{11} \\ -a_{31}/a_{11} \\ \vdots \\ -a_{n1}/a_{11} \end{pmatrix} \tag{144}$$

and I_{n-1} is the $n-1 \times n-1$ identity matrix.

The effect of this construction is to eliminate all entries below a_{11}.

Theorem 8.1: L_1^{-1}

We can check that the inverse of L_1 is

$$L_1^{-1} = \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline -\mathbf{l_1} & I_{n-1} \end{array}\right) \tag{145}$$

$$= \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_{21}/a_{11} & 1 & & & \\ a_{31}/a_{11} & & 1 & & \\ \vdots & & & \ddots & \\ a_{n1}/a_{11} & & & & 1 \end{pmatrix} \tag{146}$$

Theorem 8.2: Gaussian Elimination Step

The first step in Gaussian elimination yields

$$L_1 A = \left(\begin{array}{c|ccc} 1 & a_{12} & \cdots & a_{1n} \\ \hline 0 & & & \\ \vdots & & \hat{A}_1 & \\ 0 & & & \end{array}\right) \tag{147}$$

where $\hat{A}_1$ is some arbitrary sub-matrix resulting from the multiplication.

Having done this, we can apply the same Gaussian elimination procedure to the $n-1 \times n-1$ sub-matrix $\hat{A}_1$. This corresponds to multiplication by $n-1 \times n-1$ matrix L_2':

$$L_2' \hat{A}_1 \tag{148}$$

But we want to represent all Gaussian elimination steps as multiplications performed on A:

$$L_2 L_1 A \tag{149}$$

We can get L_2 from L_2':

Construction 8.2: L_2

Given the matrix L_2 from the Gaussian elimination step on $\hat{A}_1$, we have

$$L_2 = \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{0} & L_2' \end{array}\right) \tag{150}$$

This gaussian elimination procedure and the associated constructions continue recursively, each time operating on the increasingly small sub-matrix $\hat{A}$.

The end result is that A will be reduced to an upper-diagonal matrix U.

Definition 8.1: U

Given $n \times n$ matrix A, the Gaussian elimination procedure (repeated elimination for each of the $n-1$ columns, excluding the last column) generates the upper-triangular matrix U:

$$L_{n-1}L_{n-2}\cdots L_2L_1A = U \tag{151}$$

Notice that by this definition

$$\begin{aligned} A &= (L_{n-1}L_{n-2}\cdots L_2L_1)^{-1}U && (152) \\ &= L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}U && (153) \end{aligned}$$

Definition 8.2: L

We define

$$L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1} = L \tag{154}$$

Theorem 8.3: L is lower-triangular

We can check by multiplication that L is lower-triangular.
Moreover, we can see that L takes the form:

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ \mathbf{l_1} & \mathbf{l_2} & \cdots & \mathbf{l_{n-1}} & 1 \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{pmatrix} \tag{155}$$

where sub-column-matrices $\mathbf{l_1}, \ldots, \mathbf{l_n}$ are as defined in construction 8.1.

Definition 8.3: LU decomposition

The LU decomposition of A is given by

$$A = LU \tag{156}$$

where L and U are determined by Gaussian elimination as defined above.
L is a lower-triangular matrix and U is an upper-triangular matrix.

Let's again consider the linear system of equations

$$Ax = b \tag{157}$$

Performing Gaussian elimination, we get

$$L_{n-1} \cdots L_2 L_1 A x = L_{n-1} \cdots L_2 L_1 b \tag{158}$$

The right hand side contains our definition of U:

$$Ux = L_{n-1} \cdots L_2 L_1 b \tag{159}$$

Definition 8.4: y

Define

$$y = L_{n-1} \cdots L_2 L_1 b \tag{160}$$

Equivalently,

$$L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} y = b \tag{161}$$

The left hand side contains our definition of L:

$$Ly = b \tag{162}$$

Using this definition, our system of linear equations becomes

$$Ux = y \tag{163}$$

where

$$Ly = b \tag{164}$$

Algorithm 8.1: Solve linear system of equations

Given a linear system of equations $Ax = b$, we want to solve for x.

Algorithm 3 Solve $Ax = b$ for x

```
1: Compute $L, U$          ▷ $LU$ decomposition of $A$
2: Solve $Ly = b$          ▷ Forward substitution
3: Solve $Ux = y$          ▷ Back substitution
```

The advantage of this algorithm is that when solving systems of equations involving the same matrix A, L and U can be reused.

Algorithm 8.2: Solve $Ly = b$

Given triangular system of equation $Ly = b$, solve for y
We are given

$$\begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} \tag{165}$$

The solution is given by:

$$y_1 = \frac{b_1}{l_{11}} \tag{166}$$

$$y_2 = \frac{b_2 - l_{21}y_1}{l_{22}} \tag{167}$$

$$\vdots \tag{168}$$

$$y_n = \frac{b_n - l_{n1}y_1 - l_{n2}y_2 - \cdots - l_{n,n-1}y_{n-1}}{l_{nn}} \tag{169}$$

The same exact procedure in the reverse order can be used to solve the upper-triagonal system of equations $Ux = y$.

8.1 Costs

Here we will consider the computational cost of the Gaussian elimination procedure–namely the number of floating point additions and multiplications (floating point operations, also known as FLOPs) required.

When computing the first Gaussian elimination step (multiplication of A by L_1), for each row i below the 1st row, we multiply the row i by $-\frac{a_{i1}}{a_{11}}$ and add the result to the row.

This means 1 division per row and $n-1$ multiplications and additions per row.

Since there are $n-1$ rows below the 1st row, this results in $n-1$ divisions and $(n-1)^2$ multiplication/additions overall.

The Gaussian elimination is then repeated for every resulting sub-matrix $\hat{A}$.

Thus the overall number of divisions is

$$\sum_{k=1}^{n-1} k \tag{170}$$

which is $O(n^2)$.

The overall number of multiplications is

$$\sum_{k=1}^{n-1} k^2 \tag{171}$$

This sum has a closed form solution that can be computed with the use of generating functions. It can also be bounded above and below by a Riemann integral approximation. Either way, we get that the sum is $\Theta\left(\frac{n^3}{3}\right)$.

Finally, having found L and U, to solve $Ly = b$ and $Ux = y$ using algorithm 8.2, we need

$$\sum_{k=1}^{n-1} k \tag{172}$$

FLOPs each, which is $O(n^2)$.

Theorem 8.4: Cost of Gaussian elimination

Given $n \times n$ matrix A, to solve the system of equations $Ax = b$ using Gaussian elimination requires $\Theta\left(\frac{n^3}{3}\right)$ FLOPs.

8.2 Floating Point Error

Definition 8.5: Machine precision

Given a real number a, in machine computation it is approximated by $\mathrm{fl}(a)$ which has the property

$$\frac{|a - \mathrm{fl}(a)|}{|a|} \leq \mu \tag{173}$$

where μ is the **machine precision**, also called the **machine epsilon**.

Corollarly 8.1: Addition of numbers below machine precision has no effect

For any ϵ such that $|\epsilon| < \mu$,

$$\mathrm{fl}(1 + \epsilon) = 1 \tag{174}$$

Corollarly 8.2: μ is the smallest number whose addition has an effect

The machine precision μ is the smallest number such that

$$\mathrm{fl}(1 + \mu) > 1 \tag{175}$$

Corollarly 8.3: Addition to large numbers has no effect

For any ϵ such that $|\epsilon| < \mu$,

$$\mathrm{fl}\left(\frac{1}{\epsilon} \pm 1\right) = \frac{1}{\epsilon} \tag{176}$$

If the first entry in the process of Gaussian elimination is smaller in absolute value than the machine precision, floating point error can blow up significantly.