CSCC37

Topics

Floating Point Arithmetic

- Representing numbers on a computer

  E.g. None of the following can be represented exactly on a computer b/c binary representation does not terminate a string of bits

  $\pi = 3.14159\ldots$

  $\sqrt{2} = 1.4142\ldots$

  $1/10$

- Error propagation/rounding error

  E.g. 4 digit mantissa computer with rounding (following called subtractive cancellation)

$$\begin{aligned} 0.1234367 \quad & \leftarrow \text{stored as } 0.1234 \\ -0.1234216 \quad & \leftarrow \text{stored as } 0.1234 \\ ------ & \\ 0.0000151 \quad & \leftarrow \text{computer gives } 0 \end{aligned}$$

- Conditioning / stability

  Measures how small changes (perturbations) in the initial input affects the general result

  Conditioning refers to functions. Stability refers to algorithms.

  E.g. subtractive cancellation is an unstable algorithm.

  For functions $F : \mathbb{R} \to \mathbb{R}$, $cond(F)$ measures how changes in input x affects output $f(x)$. The smaller the better.

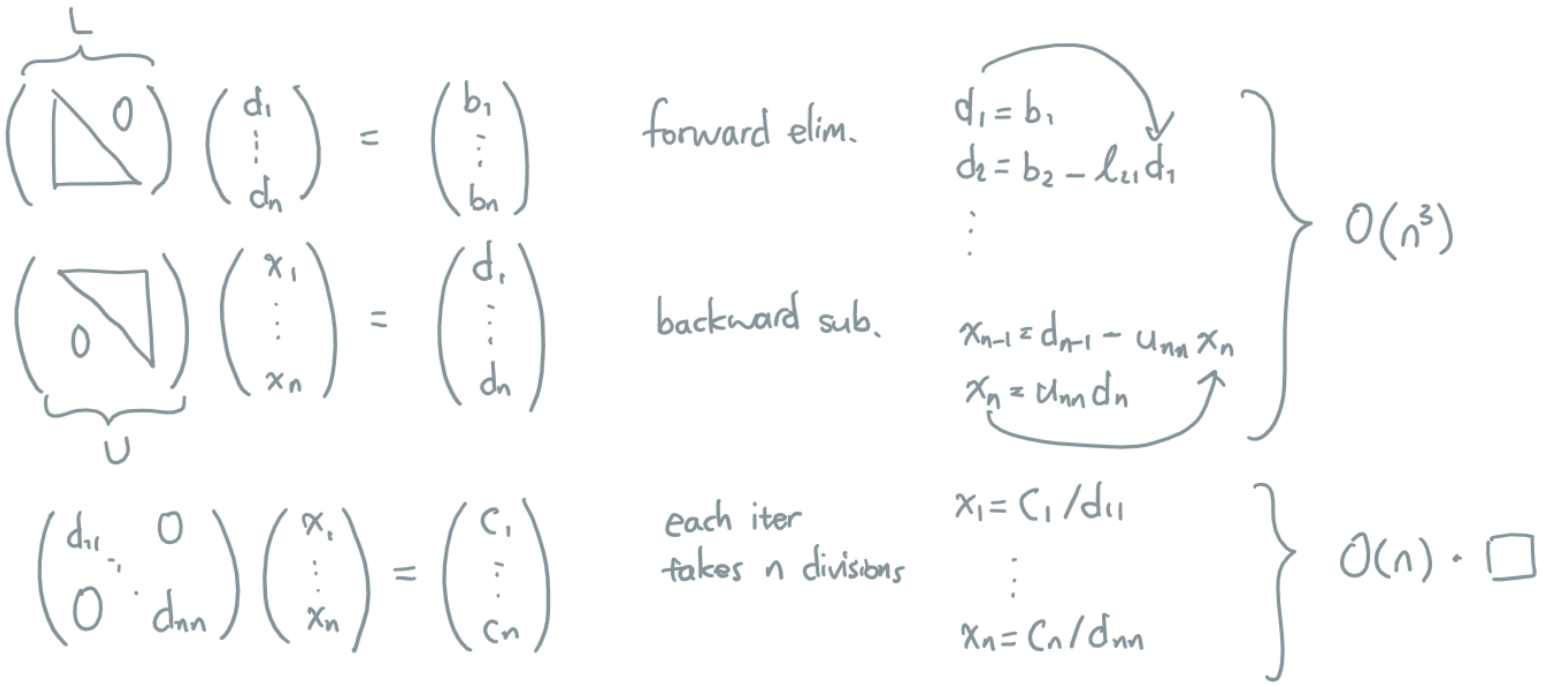
Linear Systems of Equations

Solve $A\vec{x} = \vec{b}$ where $A \in \mathbb{R}^{n \times n}, \vec{b} \in \mathbb{R}^n, \vec{x} \in \mathbb{R}^n$

Direct method - Gaussian elimination

- process: factor $A = LU$ where $L \in \mathbb{R}^{n \times n}$ is a unit lower triangular matrix, $U \in \mathbb{R}^{n \times n}$ is upper triangular

$$
\begin{aligned}
(\underbrace{LU}_{\mathbf{d}})\mathbf{x} &= \mathbf{b} \\
L\mathbf{d} &= \mathbf{b} \quad \leftarrow \text{solve for d via forward elimination} \\
U\mathbf{x} &= \mathbf{d} \quad \leftarrow \text{solve for x via backward substitution}
\end{aligned}
$$



Iterative method

- Write $A = L + D + U$ where L is lower triangular, D is diagonal, U is upper triangular

$$
\begin{aligned}
(L + D + U)\mathbf{x} &= \mathbf{b} \\
L\mathbf{x} + D\mathbf{x} + U\mathbf{x} &= \mathbf{b} \\
D\mathbf{x} &= -(L + U)\mathbf{x} + \mathbf{b} = \mathbf{c} \quad \leftarrow \text{diagonal system, which} \in O(n) \\
D\mathbf{x}^{(k+1)} &= -(L + U)\mathbf{x}^k + \mathbf{b} \quad \leftarrow \text{k} = 0, 1, \ldots \text{ until convergence}
\end{aligned}
$$

- Start with an initial guess, typically $x^{(0)} = 0$
- How do we measure convergence?

  X-test: $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| <$ tolerance

  F-test: $||\mathbf{b} - A\mathbf{x}^{(k)}|| <$ tolerance
- Which test is more reliable?

  Often use both tests. Use **iterative refinement/improvement** to possibly improve the computed solution $\hat{x}$

Non-linear Equations

- Solve $F(x) = 0$ for x^*, which is the root of the equation

  Is the equation linear? Is there a solution? How many solutions?

  linear: $F(x) = 2x + 7$

  non-linear: $F(x) = 2x^2 + 7$

  non-linear: $F(x) = x - e^{-x}$

- Algorithms based on the fixed point problem

  $F(\underbrace{x}_{\text{root x*}}) = 0 \iff x = g(\underbrace{x}_{\text{fixed point x*}})$

  Can always use $\begin{matrix} g(x) = x = F(x) \\ g(x) = x - h(x) \cdot F(x) \end{matrix}$ where h(x) is the auxiliary function

- Fixed point iteration

  Initial guess: $x^{(0)} =$?, then iterate $x^{(k+1)} = g(x^{(k)})$

  Newton's method for $f(x) = 0$

  $x^{k+1} = x^k - \frac{F(x^k)}{\frac{dF}{dx}x^k}$

Approximation / Interpolation

Data fitting

- Find a line that goes through 2 points $(t_0, y_0), (t_1, y_1)$

  $a_0 + a_1 t_0 = y_0$
  $a_0 + a_1 t_1 = y_1$

  OR $Ax = b$

  $\begin{bmatrix} 1 & t_0 \\ 1 & t_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$

- Find a line that goes through 3 points $(t_0, y_0), (t_1, y_1), (t_2, y_2)$

  $a_0 + a_1 t_0 = y_0$
  $a_0 + a_1 t_1 = y_1$
  $a_0 + a_1 t_2 = y_2$

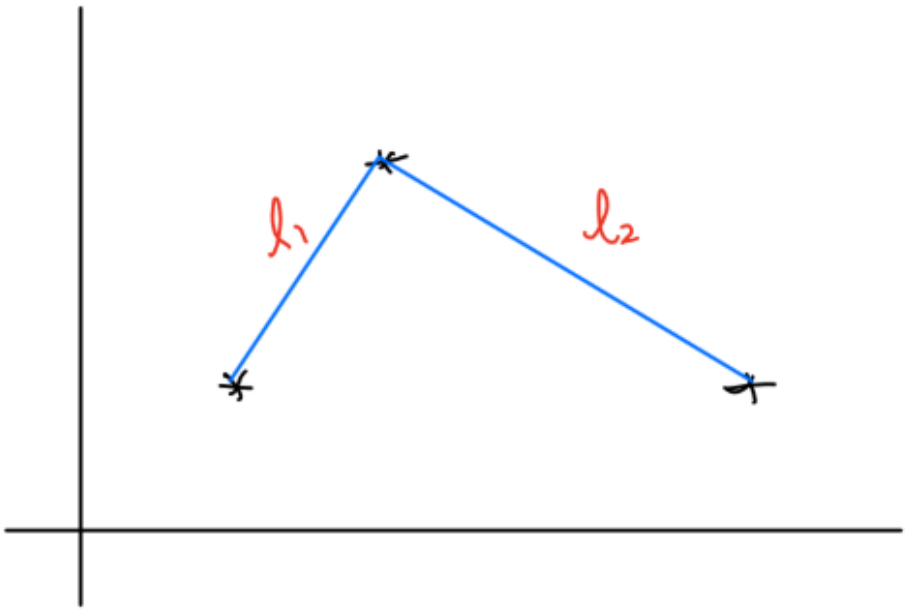
  OR $Ax = b$

$$\begin{bmatrix} 1 & t_0 \\ 1 & t_1 \\ 1 & t_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

- Non square
  - overdetermined
    - More rows than columns
    - In general, no unique sol'n
    - Look for best possible fit

      ex. find a_0, a_1 s.t. $||Ax - b||$ is minimized
  - underdetermined
    - More columns than rows
    - In general, infinitely many solutions
    - Look for a family of solutions

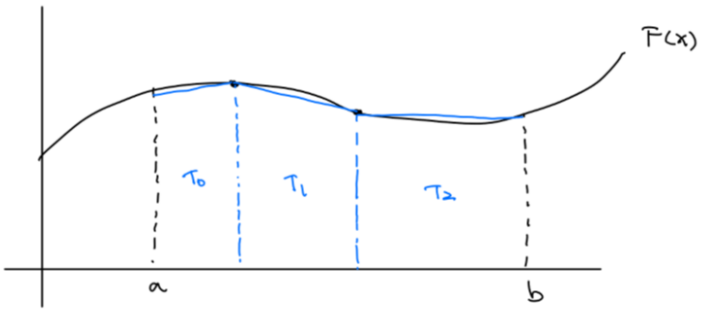
      Or use piecewise interpolants (potential problem: piecewise interpolants are not smooth)



Numerical integration / quadrature

Problem: estimate $\int_a^b f(x)dx$

Base idea: integrate piecewise and sum the areas



Then $\int_a^b F(x)\,dx \approx \sum_i T_i$

Linear Systems

Relative Residual vs Relative Error

Because of machine round off error,

- $PA = LU$ becomes $\hat{P}(A+E) = \hat{L}\hat{U}$, where $\hat{P}, \hat{L}, \hat{U}$ are the computed factors.
- Then solving $A\vec{x} = \vec{b}$ becomes solving $(A+E)\hat{x} = \vec{b}$, where $\hat{x}$ is the computed solution.
- Letting $E\hat{x} = \vec{r}$, $(A+E)\hat{x} = \vec{b}$ becomes $A\hat{x} + \vec{r} = \vec{b}$, where $\vec{r} = \vec{b} - A\hat{x}$ is the **residual**. (We would like this to be $\vec{0}$).

We can show that if row partial pivoting is used, the error and **relative residual** $\frac{\|\vec{r}\|}{\|\vec{b}\|}$ are bounded

- $\|E\| \lesssim k \cdot eps \cdot \|A\|$ where k is not too large and depends on n.
- $\|\vec{r}\| \lesssim k \cdot eps \cdot \|\vec{b}\| \iff \frac{\|\vec{r}\|}{\|\vec{b}\|} \lesssim k \cdot eps$

However, this does not mean that $\|\vec{x} - \hat{x}\|$ or **relative error** $\frac{\|\vec{x} - \hat{x}\|}{\|\vec{x}\|}$ is small.

What's the relationship between relative error and relative residual?

We have $\begin{cases} A\vec{x} = \vec{b} \\ A\hat{x} = \vec{b} - \vec{r} \end{cases} \iff A(\vec{x} - \hat{x}) = \vec{r}$

Derive upper bound:

- $\vec{x} - \hat{x} = A^{-1}\vec{r} \iff \|\vec{x} - \hat{x}\| = \left\|A^{-1}\vec{r}\right\| \leq \left\|A^{-1}\right\|\|\vec{r}\|$
- $\vec{b} = A\vec{x} \iff \|\vec{b}\| = \|A\vec{x}\| \leq \|A\|\|\vec{x}\|$

Combining the above: $\frac{\|\vec{x}-\hat{x}\|}{\|A\|\|\vec{x}\|} \leq \frac{\left\|A^{-1}\right\|\|\vec{r}\|}{\|\vec{b}\|} \iff \frac{\|\vec{x}-\hat{x}\|}{\|\vec{x}\|} \leq \frac{\|A\|\left\|A^{-1}\right\|\|\vec{r}\|}{\|\vec{b}\|}$ where $\|A\|\left\|A^{-1}\right\| = \operatorname{cond}(A)$

Derive a lower bound:

- $\vec{r} = A(\vec{x}-\hat{x}) \iff \|\vec{r}\| \leq \|A\|\|\vec{x}-\hat{x}\|$
- $\vec{x} = A^{-1}\vec{b} \iff \|\vec{x}\| \leq \left\|A^{-1}\right\|\|\vec{b}\|$

Combining the above: $\frac{\|A\|\|\vec{x}-\hat{x}\|}{\|\vec{x}\|} \geq \frac{\|\vec{r}\|}{\|A^{-1}\|\|\vec{b}\|} \iff \frac{\|\vec{x}-\hat{x}\|}{\|\vec{x}\|} \geq \frac{\|\vec{r}\|}{\|\vec{b}\|\|A\|\,\|A^{-1}\|}$

Combining:

we have $\frac{\|\vec{r}\|}{\|\vec{b}\|\operatorname{cond}(A)} \leq \frac{\|\vec{x}-\hat{x}\|}{\|\vec{x}\|} \leq \operatorname{cond}(A)\frac{\|\vec{r}\|}{\|\vec{b}\|}$

If $\operatorname{cond}(A)$ is very large, problem is poorly conditioned. Small relative residual does not mean small relative error (the bounds are large).

If the $\operatorname{cond}(A)$ is not too large, the problem is well conditioned and the small relative residual is a reliable indicator of small relative error.

Conditioning is a continuous spectrum, how large is "very large" depends on context.

Example

$\begin{bmatrix}0.780 & 0.563\\ 0.913 & 0.656\end{bmatrix}\vec{x} = \begin{bmatrix}0.217\\ 0.254\end{bmatrix}$, we know true solution is $\vec{x} = \begin{bmatrix}1\\ -1\end{bmatrix}$

Consider two computed solutions, $\hat{x}_1 = \begin{bmatrix}0.999\\ -1.001\end{bmatrix}, \hat{x}_2 = \begin{bmatrix}0.341\\ -0.087\end{bmatrix}$

$\vec{r}_1 = \vec{b} - A\hat{x}_1 = \begin{bmatrix}-0.001243\\ -0.001572\end{bmatrix}, \vec{r}_2 = \vec{b} - A\hat{x}_2 = \begin{bmatrix}-0.000001\\ 0\end{bmatrix}$

$\frac{\|\vec{x}-\hat{x}_1\|}{\|x\|}$ is much smaller than $\frac{\|\vec{x}-\hat{x}_2\|}{\|x\|}$ as expected (since $\hat{x}_2$ is a terrible solution) , but $\frac{\|\vec{r}_1\|}{\|\vec{b}\|}$ is much larger than $\frac{\|\vec{r}_2\|}{\|\vec{b}\|}$.

$\implies$ relative residual is not a reliable indicator of relative error.

Let's take a look at $cond(A)$.

$A^{-1} = 10^6\begin{bmatrix}0.656 & -0.565\\ -0.913 & 0.780\end{bmatrix}$, where $10^6 = \frac{1}{\det(A)}$

$\|A\|_\infty = 1.572, \left\|A^{-1}\right\|_\infty = 1.693 \cdot 10^6 \Rightarrow \operatorname{cond}_\infty(A) = \|A\|_\infty\left\|A^{-1}\right\|_\infty = 2.66 \cdot 10^6$

$\frac{\|\vec{x}-\hat{x}\|}{\|\vec{x}\|} \leq 2.66 \cdot 10^6\frac{\|r\|}{\|b\|}$, i.e. relative error in x could be as big as $2.66 \cdot 10^6$ times the relative residual.

Thus this A is a poorly conditioned matrix.

Iterative Improvement

Can we improve $\hat{x}$? One easy way is to improve mantissa length.

We want to solve $A\vec{x} = \vec{b}$, having already solved $(A+E)\hat{x} = \vec{b}$ or $A\hat{x} + \vec{r} = \vec{b}$

Subtracting the two equations yields $A(\vec{x} - \hat{x}) = \vec{r}$, so we let $\vec{z} = \vec{x} - \hat{x}$, and solve $A\vec{z} = \vec{r}$

Then $\vec{x} = \hat{x} + \vec{z}$, but this is a fallacy since we can not solve for $\vec{z}$ exactly. Thus we get $\hat{z}$, and not $\vec{z}$.

But if we were to get 2 leading digits correct in $\hat{x}$, then we will get 2 leading digits correct in $\hat{z}$, and then we will have 4 leading digits correct in $\hat{x} + \hat{z}$

The Algorithm

Compute $\hat{x}^{(0)}$ by solving $A\vec{x} = \vec{b}$ in a floating point system.

For $i = 0, 1, 2, \ldots$ until solution is good enough (or until convergence failure)

1. Compute the residual: $r^{(i)} = b - A\hat{x}^{(i)}$
2. Solve $A\vec{z}^{(i)} = r^{(i)}$ For some $\hat{z}^{(i)} = x - x_i$
3. Update $\hat{x}^{(i+1)} = \hat{x}^{(i)} + \hat{z}^{(i)}$

Why is x_{i+1} better than x_i?

x^1 is the first element of x where $Ax = b$ and $x^1 = .d_1d_2d_3d_4\ldots$

In the first iteration we get $x_0^1 = .d_1d_2\tilde{d}_3\tilde{d}_4\ldots$

z_0 is solved from $Az_0 = r_0$

$z_0 = 0.00z_3z_4\tilde{z}_5\tilde{z}_6$

$x_1 = x_0 + z_0 = 0.d_1d_2\tilde{d}_3\tilde{d}_4 + 0.00z_3z_4\tilde{z}_5\tilde{z}_6$

since $z_3 = d_3 - \tilde{d}_3$ and $z_4 = d_4 - \tilde{d}_4$

we have that $x_1 = .d_1d_2\underbrace{(\tilde{d}_3 + z_3)}_{d_3}\underbrace{(\tilde{d}_4 + z_4)}_{d_4}$

Non Linear Equations

Fixed Point Iteration

Problem: $F : \mathbb{R} \to \mathbb{R}$

We need to some $F(\tilde{x}) = 0$ for some $\tilde{x} \in \mathbb{R}$ (root finding problem)

Typically, we cannot find a "closed form" (analytic) solution to non linear F, which is why we need numerical techniques.

We can find iterative methods which generate an approximation $\hat{x_k}, k = 0, 1, 2, \ldots$ such that $\hat{x_k} \to \tilde{x}$ as $k \to \infty$

Fixed Point Methods

Given a root finding problem, $F(\tilde{x}) = 0$, this is equivalent to a fixed point problem $\tilde{x} = g(\tilde{x})$, where $\tilde{x}$ is a fixed under g

Example: $F(x) = x - e^{-x}$, this is equivalent to finding the fixed point of $x = e^{-x}$

We often use the algebraically equivalent fixed point problem to solve the root finding problem because there is an obvious iteration algorithm to do so.

To obtain a working $g(x)$, we can use 2 forms:

- $g(x) = x - F(x)$ $\quad \leftarrow$ first form
  - $F(\tilde{x}) = 0 \Leftrightarrow \tilde{x} = g(\tilde{x})$
- $g(x) = x - h(x)F(x)$ $\quad \leftarrow$ second form
  - $F(\tilde{x}) = 0 \Rightarrow \tilde{x} = g(\tilde{x})$, however we can have a fixed point $\tilde{x} = g(\tilde{x})$ such that $h(\tilde{x}) = 0$ but $F(\tilde{x}) \neq 0$
  - Advantage: there is flexibility in designing g(x) to make iteration converge faster.

Start with an approximate solution $\hat{x_0}$ then iterate: $x_{\hat{k+1}} = g\left(\hat{x_k}\right), k = 0, 1, 2, \ldots$ until convergence/failure.

Example: Let $F(x) = x^2 + 2x - 3$ with exact roots $\tilde{x} = 1, -3$

Consider the fixed point iteration (FPI): $x_{k+1} = x_k + \frac{x_k^2 + 2x_k - 3}{x_x^2 - 5}$ for the fixed point problem $x = g(x) = x + \frac{x^2 + 2x - 3}{x^2 - 5}$

We see that this is the second form of the fixed point problem, where $h(x) = -\frac{1}{x^2 - 5}$

FPI of $\hat{x_0} = -5 \to \hat{x_k}$ converges to -3.

However, FPI of $\hat{x_0} = 5 \to \hat{x_k}$ does not converge. We can keep trying, but that would be a significant waste of time.

Fixed Point Theorem

If there is an interval $[a, b]$ such that $\forall\, x \in [a, b]$

- $g(x) \in [a, b]$
- $|g'(x)| \leq L < 1$

then $g(x)$ has a unique fixed point in $[a, b]$

Proof

> Recall the **Mean Value Theorem** (MVT)
>
> If f is continuous on $[a, b]$ and differentiable on (a, b), then $\exists\ c$ s.t. $f'(c) = \dfrac{f(b) - f(a)}{b - a}$
>
> $\iff f(b) - f(a) = f'(c)(b - a)$

Start with any $\hat{x_0} \in [a, b]$ and iterate with $\hat{x}_{k+1} = g\left(\hat{x_k}\right), k = 0, 1, 2 \ldots$, then all $\hat{x_k} \in [a, b]$

Moreover, by the MVT, we have that for some $\eta_k \in [x_{k-1}, x_k] \subset [a, b]$

$$\begin{aligned} x_{k+1} - x_k &= g\left(x_k\right) - g\left(x_{k-1}\right) \\ &= g'\left(\eta_k\right)\left(x_k - x_{k-1}\right) \\ \left|x_{k+1} - x_k\right| &\leq \left|g'\left(\eta_k\right)\right| \left|x_k - x_{k-1}\right| \\ &\leq L\left|x_k - x_{k-1}\right| \\ &\leq \cdots \\ &\leq L^k \left|x_1 - x_0\right| \end{aligned}$$

As $k \to \infty$, RHS $\to 0$ since $L < 1$, so LHS $|x_{k+1} - x_k| \to 0$

This means the fixed points are converging to some $\tilde{x} \in [a, b]$

To complete this proof, we must show:

1. $\tilde{x} = g(\tilde{x})$ (i.e., $\tilde{x}$ is a fixed point!)
2. $\tilde{x}$ is unique in $[a, b]$

Example How do we use FPT?

$F(x) = x - e^x = 0$ and $g(x) = e^{-x}$ for $x = g(x)$

What is the maximum interval of guaranteed convergence of x_k ?

Rate of Convergence

We have p-th order convergence to fixed point $\tilde{x}$ if $\lim_{x_k \to \tilde{x}} \dfrac{|\tilde{x} - x_{k+1}|}{|\tilde{x} - x_k|^p} = c \neq 0$

Example: Table of absolute error of iterates, $|\tilde{x} - x_k|$

k	$p = 1, c = 1/2$	$p = 2, c = 1$
0	10^{-1}	10^{-1}
1	$5 \cdot 10^{-2}$	10^{-2}
2	$2.5 \cdot 10^{-2}$	10^{-4}
3	$1.25 \cdot 10^{-2}$	10^{-8}
4	$6.125 \cdot 10^{-3}$	10^{-16}

We see that the second column converges much much faster.

Thm (FPI Convergence)

For the Fixed Point Iteration $x_{k+1} = g(x_k)$, we have p-th order convergence to $\hat{x}$ if $g'(\hat{x}) = g''(\hat{x}) \cdots = g^{(p-1)}(\hat{x}) = 0$ but $g^p(\hat{x}) \neq 0$

Proof

> Recall: the **Taylor series** of a function is an infinite sum of terms that are expressed in terms of the function's derivatives at a single point.
>
> $$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

So letting $x_{k+1} = g(x_k) = g(\tilde{x} + (x_k - \tilde{x}))$, its Taylor expansion is

$$x_{k+1} = g(\tilde{x}) + (x_k - \tilde{x})g'(\tilde{x}) + \frac{(x_k - \tilde{x})^2}{2!}g''(\tilde{x}) + \cdots + \frac{(x_k - \tilde{x})^{p-1}}{(p-1)!}g^{(p-1)}(\tilde{x}) + \frac{(x_k - \tilde{x})^p}{p!}g^{(p)}(\eta_k)$$

where $\eta_k \in [\tilde{x}, x_k]$

However, if $g'(\hat{x}) = g''(\hat{x}) = \cdots = g^{(p-1)}(\hat{x}) = 0$, the Taylor expansion yields

$$
\begin{aligned}
x_{k+1} &= g(\hat{x}) + \frac{\left(x_k - \hat{x}\right)^p}{p!} g^{(p)}\left(\eta_k\right) \\
\Leftrightarrow \frac{x_{k+1} - g(\hat{x})}{\left(x_k - \hat{x}\right)^p} &= \frac{1}{p!} g^{(p)}\left(\eta_k\right) \\
\Leftrightarrow \frac{x_{k+1} - \hat{x}}{\left(x_k - \hat{x}\right)^p} &= \frac{1}{p!} g^{(p)}\left(\eta_k\right)
\end{aligned}
$$

We see that as $k \to \infty, x_k \to \hat{x}, \eta_k \in \left[\hat{x}, x_k\right] \to \hat{x}$

We can rewrite this as

$$
\lim_{x_k \to \hat{x}} \frac{\left|x_{k+1} - \hat{x}\right|}{\left|x_k - \hat{x}\right|^p} = \frac{1}{p!} g^{(p)}(\hat{x}) \neq 0
$$

Thus we see that when the p-th derivative of g at $\hat{x}$ is not zero, we reach p-th order convergence.

We see that now we can use the second form of the fixed point iteration $g(x) = x - h(x)F(x)$ to accelerate iteration by picking a $h(x)$ such that the p-th derivative of g is not zero at $\hat{x}$.

Newton's Method

> For $x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}$
>
> root finding problem of $F(x) = 0 \iff x = g(x)$ with $g(x) = x - \frac{F(x)}{F'(x)}$
>
> This is the second form of the fixed point problem with $h(x) = \frac{1}{F'(x)}$

Speed of Newton's Method

Suppose that $F'(\tilde{x}) = 0$ but $F''(\tilde{x}) \neq 0$

$g'(x) = 1 - \frac{F'(x)F'(x) - F(x)F''(x)}{\left(F'(x)\right)^2} = \frac{F(x)F''(x)}{\left(F'(x)\right)^2}$

so $g'(\tilde{x}) = 0$ for any function F.

By rate of convergence theorem, Newton's method has at least **quadratic convergence** (p=2), since $g'(\tilde{x}) = 0$ for any function F

Geometric Interpretation of Newton's Method

To solve $F(x)=0$ at an initial guess x_k , we can approximate (or model) $F(x)$ by a **linear polynomial** $p(x)$ that satisfies conditions: $p\left(x_k\right)=F\left(x_k\right)$ and $p'\left(x_k\right)=F'\left(x_k\right)$

Recall MVT: $f(b)-f(a)=f'(c)(b-a) \iff f(b)=f(a)+(b-a)f'(c)$

So we have the following polynomial

$$p_k(x)=F\left(x_k\right)+\left(x-x_k\right)F'\left(x_k\right)$$

x_{k+1} is the root of $p_k(x)$, i.e. $p_k\left(x_{k+1}\right)=0$

$$\begin{aligned}
&\Rightarrow F\left(x_k\right)+\left(x_{k+1}-x_k\right)F'\left(x_k\right)=0 \\
&\Rightarrow x_{k+1}=x_k-\frac{F\left(x_k\right)}{F'\left(x_k\right)}
\end{aligned}$$

Note Newton's Method does not always converge.

Example Newton's method on $F(x)=x-e^{-x}$

$$x_{k+1}=x_k-\frac{F\left(x_k\right)}{F'\left(x_k\right)}=x_k-\frac{x_k-e^{-x_k}}{1+e^{-x_k}}=\frac{(1+x_k)e^{-x_k}}{1+e^{-x_k}}$$

Secant Method

Recall NM: $x_{k+1}=x_k-\frac{F(x_k)}{F'(x_k)}$, notice that we have to supply the first derivative of F

We can approximate $F'\left(x_k\right)$ by $\frac{F(x_k)-F(x_{k-1})}{x_k-x_{k-1}}$, then

$$x_{k+1}=x_k-F\left(x_k\right)\frac{\left(x_k-x_{k-1}\right)}{F\left(x_k\right)-F\left(x_{k-1}\right)}$$

Another way to derive this is using the geometric interpretation.

$$\begin{aligned}
p_k(x)&=F\left(x_k\right)+\left(x-x_k\right)F'\left(x_k\right) \\
&=F(x_k)+(x-x_k)\frac{F(x_k)-F(x_{k-1})}{x_k-x_{k-1}}
\end{aligned}$$

Let x_{k+1} be the root of $p_k(x)$, i.e. $p_k(x_{k+1})=0=F(x_k)+(x_{k+1}-x_k)\frac{F(x_k)-F(x_{k-1})}{x_k-x_{k-1}}$

However, this is not a fixed point iteration of the form $x_{k+1}=g\left(x_k\right)$, as no function g could support two values.

We cannot directly use FPT or RCT to analyze this method. However, with some adjustments we can prove rate of convergence is $p=\frac{1+\sqrt{5}}{2}\approx 1.62$

Super Linear Convergence

Newton's Method: $x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)} \quad \leftarrow \text{p} = 2$

Secant Method: $x_{k+1} = x_k - \frac{F(x_k)}{\frac{F(x_k)-F(x_{k-1})}{x_k - x_{k-1}}} = x_k - F(x_k)\frac{(x_k - x_{k-1})}{F(x_k)-F(x_{k-1})} \quad \leftarrow \text{p} = 1.62$

But we see that Newton's method requires 2 Function evaluations per step as F, F' are not the same.

The secant method requires only 1 function evaluation. Although $F(x)$ needs to be evaluated, $F(x_{k-1})$ has been computed during the previous iteration!

Effective rate of convergence takes cost per iteration into account, as well as speed. Thus secant method is "effectively" faster than Newton's method.

Bisection Method

Often times, if we start Newton's/Secant method with the wrong initial guess, they will not converge (wrong side of the hill). Bisection Method however has guaranteed convergence, this is to say that it always finds a root, albeit slowly.

We need to find an $a \leq b$ such that $F(a) \leq 0 \leq F(b)$ or $F(b) \leq 0 \leq F(a)$. This way, there is at least one root of F in $[a, b]$

Assume $F(a) \leq 0 \leq F(b)$, we loop until $b - a$ is small enough:

Let $m = (a + b)/2$. If $F(m) \leq 0$ then set $a = m$, else $b = m$

Repeat for the interval $[m, b]$ or $[a, m]$

Then we have linear rate of convergence $p = 1, c = \frac{1}{2}$ with guaranteed convergence.

Hybrid Method

We can start off with slow reliable methods (e.g. bisection), then switch to a faster one that requires a more accurate initial guess.

Example: Combine Bisection and Newton's Method

System of Non-Linear Equations

We want to solve $F(\vec{x}) = \vec{0}$

where $F(\vec{x}) = \begin{bmatrix} F_1(x_1, x_2) \\ F_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} 4(x_1)^2 + 9(x_2)^2 - 16x_1 - 54x_2 + 61 \\ x_1x_2 - 2x_1 - 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Recall Newton's Method for a single non linear equation where $F : \mathbb{R} \to \mathbb{R}$ and $x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}$

Then for $F : \mathbb{R}^n \to \mathbb{R}^n$, $x_{k+1} = \vec{x_k} - \frac{F(\vec{x_k})}{F'(\vec{x_k})}$

Note F' is the Jacobian matrix of F, i.e. $F' \in \mathbb{R}^{n \times n}$

$$\vec{x_{k+1}} = \vec{x_k} - \left[F'\left(\vec{x_k}\right)\right]^{-1} F\left(\vec{x_k}\right)$$
$$\left[F'\left(\vec{x_k}\right)\right]\left(\vec{x_{k+1}} - \vec{x_k}\right) = -F\left(\vec{x_k}\right)$$

This is simply in the form of some $A\vec{x} = \vec{b}$, a linear system!

This is very expensive! We can use the Pseudo Newton Method by holding the Jacobian matrix fixed for a few iterations, so that we can reuse the $PA = LU$ factorization. This is alright so long as we are still converging.

Approximation/Interpolation

Approximation Methods

Truncated Taylor Series

Find Taylor expansion for $F(x)$

$$F(x) = F(a) + \frac{F'(a)}{1!}(x-a) + \frac{F''(a)}{2!}(x-a)^2 + \frac{F'''(a)}{3!}(x-a)^3 + \cdots$$

Approximate it by truncating at n

$$p(x) = F(a) + F'(a)(x-a) + \cdots + \frac{F^{(n)}(a)}{n!}(x-a)^n$$

then the error can be expressed as

$$e(x) = p(x) - F(x) = \frac{F^{(n+1)}(\eta)}{(n+1)!}(x-a)^{n+1}$$

Interpolation

Find a polynomial p such that $p(x_i) = F(x_i), i = 0, 1, 2, \ldots$

F is the function we are trying to approximate, could simply be a set of data.

Least Squares

Find a polynomial p such that $p(x)$ minimizes $\|F-p\|_2 = \left(\int_a^b (F(x)-p(x))^2 dx\right)^{1/2}$

Other norms we can use for Least Squares approximation:

$\|F-p\|_\infty = \max_{a\le x\le b} |F(x)-p(x)|$

$\|F-p\|_1 = \int_a^b |F(x)-p(x)|dx$

Polynomial Interpolation

Consider $\mathcal{P}_n$: the set of all polynomials of degree $\le n$

It is a function space of dimension $n+1$, i.e., it requires a basis of $n+1$ functions.

The most common basis is the monomial basis $\left\{x^i\right\}_{i=0}^n$

i.e. it spans $\mathcal{P}_n$ and the x^i are linearly independent ($\sum_{i=0}^n a_i x^i = 0 \;\forall\; x \implies a_i = 0 \;\forall\; i$)

Note we can shift the basis $\left\{(x_\alpha)^i : i = 0,1,\dots,n\right\}$

Weierstrass' Theorem (Existence)

If a function F is continuous on an interval $[a,b]$, then for any $\epsilon > 0$, $\exists\, p_\epsilon$ s.t. $\|F-p_\epsilon\|_\infty < \epsilon$

I.e. for any continuous function on a closed interval $[a,b]$, there exists some polynomial that is as close to it as can be.

Numerical Methods for Polynomial Interpolation

3 bases - Monomial, Lagrange, and Newton

Monomial Basis

a.k.a Vandermonde Method or Method of Undetermined Coefficients

Theorem

For any sets: $\{x_i : i = 0,1,\dots n\}, \{y_i : i = 0,1,\dots,n\}$ for distinct x_i s and indistinct y_i s

$\nexists\, p(x) \in \mathcal{P}_n$ s.t. $p\left(x_i\right) = y_i, i = 0,1,\dots,n$

Proof

If such $p(x)$ exists, it must be possible to write it as a monomial space polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$

Then we have $n+1$ unknowns, $\{a_i\}, i = 0, 1, \ldots, n$

We need $n+1$ equations to solve for the a_i's

We can convert it into a matrix problem with $p(x_i) = y_i, i = 0, 1, \ldots n$

Vandermonde Matrix

$$Va = y$$

$$\begin{bmatrix} (x_0)^0 & (x_0)^1 & (x_0)^2 & \ldots & (x_0)^n \\ (x_1)^0 & (x_1)^1 & (x_1)^2 & \ldots & (x_1)^n \\ \vdots & & & & \\ (x_n)^0 & (x_n)^1 & (x_n)^2 & \ldots & (x_n)^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

The question now reduces to checking whether V is invertible/non-singular.

V is singular iff $\exists\, \vec{a} \neq 0$ s.t. $V\vec{a} = 0$

We can return to polynomial problem, we are asking if it's possible for $p \in \mathcal{P}_n$ s.t. $p(x_i) = 0, i = 0, 1, 2, \ldots, n$

If p has $n+1$ roots $\implies p(x_i) = 0 \; \forall \; x \implies \vec{a} = 0 \implies V$ is singular

But this is impossible as no polynomial in $\mathcal{P}_n$ can have $n+1$ roots

Alternatively, since $\{x_i\}$ are distinct, we have $\det(V) = (-1)^{n+1} \prod_{i=1}^{n} \prod_{j=0}^{i-1} (x_i - x_j) \neq 0 \implies V$ is non-singular

The Vandermonde matrix proves existence, but it does not lead to the best algorithm - it can be poorly conditioned, and the $PV = LU$ factorization plus forward/backward solve will be costly.

Lagrange Basis

Consider a simple interpolation problem $p(x_i) = y_i, i = 0, 1, \ldots, n$

Consider the basis $\{l_i(x)\}$ where $l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$ for $i = 0, 1, \ldots, n$

Note, $l_i(x) \in \mathcal{P}_n, \forall i$ and $l_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

With this basis function, we can write out the interpolating polynomial for free, but it is prohibitively expensive to evaluate at non-interpolation points

$p(x) = \sum_{i=0}^{n} l_i(x) y_i \implies p(x) \in \mathcal{P}_n$ and $p(x_i) = y_i, i = 0, 1, \ldots, n$

Newton Basis

For simple interpolation $p(x_i) = y_i, i = 0, 1, \ldots, n$, we look for an interpolating $p(x)$ of the form:

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

Converting into a matrix, we have

$$\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & (x_1 - x_0) & 0 & \cdots & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & & & & \\ 1 & (x_n - x_0) & \ldots & & \prod_{i=0}^{n-1}(x_n - x_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

This is a lower triangular system, which means there is no factorization involved.

$$\begin{aligned} a_0 &= y_0 \\ a_1 &= \frac{y_1 - y_0}{x_1 - x_0} \\ a_2 &= \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} \end{aligned}$$

We notice a pattern, defined below.

Divided Difference

$$\begin{aligned} y[x_1] &= y(x_1) = y_1 \\ y[x_{i+k}, \ldots, x_i] &= \frac{y[x_{i+k}, \ldots, x_{i+1}] - y[x_{i+k-1}, \ldots, x_i]}{x_{i+k} - x_i} \\ \text{e.g.} \quad y[x_2, x_1, x_0] &= \frac{y[x_2, x_1] - y[x_1, x_0]}{x_2 - x_0} \end{aligned}$$

The polynomial is thus given by

$$\begin{aligned} p(x) = y[x_0] + \\ (x - x_0)y[x_1, x_0] + \\ (x - x_0)(x - x_1)y[x_2, x_1, x_0] + \cdots + \\ (x - x_0)\ldots(x - x_{n-2})y[x_{n-1}, \ldots, x_0] + \\ (x - x_0)\ldots(x - x_{n-1})y[x_n, \ldots, x_0] \end{aligned}$$

Then, $p(x) \in \mathcal{P}_n$ and $p(x_i) = y_i, i = 0, 1, \ldots, n$

Proof by Induction let $p(x) \equiv p_n(x)$

Basis: $n = 0$

$p_0(x) = y[x_0]$

$p_0(x_0) = y[x_0] = y_0$

Basis: $n = 1$

$p_1(x) = y[x_0] + (x - x_0)y[x_1, x_0]$

$p_1(x_0) = y[x_0] = y_0$

$p_1(x_1) = y[x_0] + (x_1 - x_0)\left(\frac{y_1 - y_0}{x_1 - x_0}\right) = y[x_0] + y_1 - y_0 = y_1$

Induction hypothesis

1. $p_{n-1}(x) \in P_{n-1}$ is the unique polynomial of degree $\leq n-1$ that satisfies $p_{n-1}(x_i) = y_i, i = 0, \ldots, n-1$

   $p_{n-1}(x) = y[x_0] + (x - x_0)y[x_1, x_0] + (x - x_0)(x - x_1)y[x_2, x_1, x_0] + \ldots + (x - x_0)(x - x_1)\ldots(x - x_{n-2})y[x_{n-1}, \ldots, x_0]$

2. $q(x) \in P_{n-1}$ is the unique polynomial of degree $\leq n-1$ that satisfies $q(x_i) = y_i, i = 1, \ldots, n$

   $q(x) = y[x_1] + (x - x_1)y[x_2, x_1] + (x - x_1)(x - x_2)y[x_3, x_2, x_1] + \ldots + (x - x_1)(x - x_2)\ldots(x - x_{n-1})y[x_n, \ldots, x_1]$

   Note $q(x)$ is just $p_{n-1}(x)$ with the data shifted $x_i \to x_{i+1}$

Induction step

Consider $p_n(x) = p_{n-1}(x) + (x - x_0)(x - x_1)\ldots(x - x_{n-1})a_n$

Clearly, $p_n(x_i) = p_{n-1}(x_i) = y_i$ for $i = 0, 1, \ldots, n-1$

Need to choose a_n s.t. $p_n(x_n) = y_n$

Consider the auxiliary polynomial $r(x) = \dfrac{(x - x_0)q(x) - (x - x_n)p_{n-1}(x)}{x_n - x_0}$

Then, $r(x_0) = p_{n-1}(x_0) = y_0$ by IH1, and $r(x_n) = q(x_n) = y_n$ by IH2

and for $i = 1, 2, \ldots, n-1$, $r(x_i) = \dfrac{(x_i - x_0)y_i - (x_i - x_n)y_i}{x_n - x_0} = y_i$

Also $r(x) \in P_n$, so $r(x)$ is the unique polynomial of degree $\leq n$ that satisfies $r(x_i) = y_i, i = 0, \ldots, n$

If $p_n(x) = p_{n-1}(x) + (x - x_0)(x - x_1)\ldots(x - x_{n-1})a_n$ also satisfies $p_n(x_i) = y_i, i = 0, \ldots, n$, then we must have that $p_n(x) = r(x)$

Coefficient of x^n in $p_n(x)$ is a_n

Coefficient of x^n in $r(x)$ is $\dfrac{y[x_n \ldots x_1] - y[x_{n-1} \ldots x_0]}{x_n - x_0} = y[x_n \ldots x_0]$

Hence $a_n = y[x_n \dots x_0]$ and $p_n(x) = p_{n-1}(x) + (x - x_0)(x - x_1)\dots(x - x_{n-1})y[x_n \dots x_0]$

E.g. Find a $p \in P_3$ s.t. $p(0) = 1, p(1) = 3, p(2) = 9, p(3) = 25$

Divided difference table

x	$y[x_i]$	$y[x_{i+1}, x_i]$	$y[x_{i+2} \dots x_i]$	$y[x_{i+3} \dots x_i]$
0	1	$\frac{3-1}{1-0} = 2$	$\frac{6-2}{2-0} = 2$	$\frac{5-2}{3-0} = 1$
1	3	$\frac{9-3}{2-1} = 6$	$\frac{16-6}{3-1} = 5$	
2	9	$\frac{25-9}{3-2} = 16$		
3	25			

$$\begin{aligned} p(x) &= y[x_0] + (x - x_0)y[x_1, x_0] + (x - x_0)(x - x_1)y[x_2, x_1, x_0] + (x - x_0)(x - x_1)y[x_3, x_2, x_1, x_0] \\ &= 1 + 2x + 2(x)(x - 1) + x(x - 1)(x - 2) \end{aligned}$$

Osculatory Interpolation (interpolation with derivatives)

How are divided differences and derivatives related?

For $y\left[x_1, x_0\right] = \frac{y(x_1) - y(x_0)}{x_1 - x_0}$, we have

$$\lim_{x_1 \to x_0} y\left[x_1, x_0\right] = \lim_{x_1 \to x_0} \frac{y\left(x_1\right) - y\left(x_0\right)}{x_1 - x_0} = y'\left(x_0\right)$$

For $y\left[x_2, x_1, x_0\right] = \frac{y[x_2, x_1] - y[x_1, x_0]}{x_2 - x_0}$, we have

$$\lim_{\substack{x_2 \to x_0 \\ x_1 \to x_0}} y\left[x_2, x_1, x_0\right] = \lim_{x_1 \to x_0} \frac{y\left[x_2, x_1\right] - y\left[x_1, x_0\right]}{x_2 - x_0} = \frac{y''\left(x_0\right)}{2!}$$

We can show that in general

$$\lim_{\substack{x_k \to x_0 \\ x_{k-1} \to x_0 \\ \vdots \\ x_1 \to x_0}} y\left[x_k, x_{k-1}, \dots x_0\right] = \frac{y^{(k)}\left(x_0\right)}{k!}$$

Example: Find $p \in \mathcal{P}_4$ such that

$p(0) = 0$

$p(1) = 1, p'(1) = 1, p''(1) = 2$

$p(2) = 6$

x	$y[x_i]$	$y[x_{i+1}, x_i]$	$y[x_{i+2} \dots x_i]$	$y[x_{i+3} \dots x_i]$	$y[x_{i+4} \dots x_i]$
0	0	$\frac{1-0}{1-0} = 1$	$\frac{1-1}{1-0} = 0$	$\frac{1-0}{1-0} = 1$	$\frac{3-1}{2-0} = 1$
1	1	$\frac{y'(1)}{1!} = 1$	$\frac{y''(1)}{2!} = 1$	$\frac{4-1}{1} = 3$	
1	1	$\frac{y'(1)}{1!} = 1$	$\frac{5-1}{2-1} = 4$		
1	1	$\frac{6-1}{2-1} = 5$			
2	6				

$$\begin{aligned} p(x) &= y[0] + xy[1,0] + x(x-1)y[1,1,0] + x(x-1)^2 y[1,1,1,0] + x(x-1)^3 y[2,1,1,1,0] \\ &= 0 + x + x(x-1)^2 + x(x-1)^3 \end{aligned}$$

Error in Polynomial Interpolation

$E(x) = y(x) - p(x)$

For simple interpolation, $p(x_i) = y_i, i = 0, 1, \dots, n$, we can show that

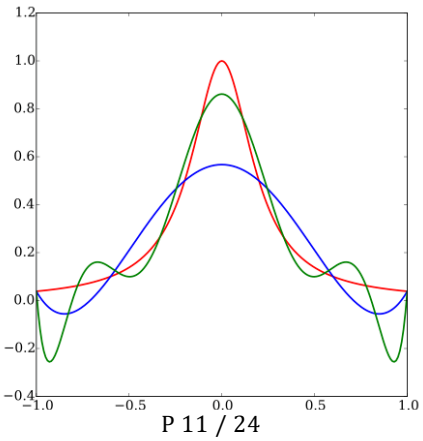
$$E(x) = \frac{y^{(n+1)}(\eta)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

where $\eta \in sp\{x_0, \dots, x_n, x\} = [\min\{x_0, \dots, x_n, x\}, \max\{x_0, \dots, x_n, x\}]$

Runge Phenomenon

Problem of oscillation at the edges of an interval that occurs when interpolating with polynomials of high degree over a set of **equispaced** points - going to higher degrees does not always improve accuracy! (ML app: overfitting)

Below is the plot of the Runge function $f(x) = \frac{1}{1+25x^2}$ interpolated at equidistant $x_i \in [-1, 1]$ s.t. $x_i = \frac{2i}{n} - 1, i \in \{0, 1, \dots, n\}$



Solution: piecewise interpolation

Numerical Integration

Quadrature Rule

Quadrature is a historical term which refers to the process of determining area.

Problem: Approximate some $I(F) = \int_a^b F(x)dx$

Recall: Riemann Sums of a function F

$$R(F) = \sum_{i=0}^{u-1} F(\eta_i)(x_{i+1} - x_i)$$

Composite Rule: Apply a simple formula like above on many short integrals, i.e. $\int_{x_1}^{x_2} F(x)dx$

Basic Formulas: based on interpolating quadrature (instead of integrating we determine area using the interpolated polynomial)

We approximate $F(x)$ by some $p(x)$ on $[a, b]$ (or $[x_i, x_{i+1}]$). In Lagrange form, $p(x) = \sum_0^n F(x_i)l_i(x)$

$$\begin{aligned} I(F) \approx Q(F) &= I(p) \\ &= \int_a^b p(x)dx \\ &= \int_a^b \left[\sum_{i=0}^n F(x_i)l_i(x)\right] \\ &= \sum_{i=0}^n F(x_i) \underbrace{\int_a^b l_i(x)dx}_{A_i} \\ &= \sum_{i=0}^n A_i F(x_i) \end{aligned}$$

This is the standard form of **quadrature rule**. The A_i are weights, and x_i are nodes.

$Q(F)$ is a linear operator. I.e. $Q(\alpha f + g) = \alpha Q(F) + Q(g)$ for functions f and g

If $Q(F)$ integrates $1, x, x^2, \ldots, x^n$ exactly, then $Q(F)$ integrates all polynomials of degree $\leq n$ exactly.

Theorem: Existence of Interpolatory Quadrature

Given any set of $n+1$ distinct nodes, we can choose the weights A_i such that the quadrature rule is exact for all polynomials of degree $\leq n$, moreover A_i 's are unique.

Proof Since $Q(F)$ is linear, we need only show Q is exact for $\{x_i\}_{i=0}^n$. I.e., for each x^k, we need to show

$$Q\left(x^k\right)=\sum_{i=0}^n A_i(x_i)^k=\frac{1}{k+1}\left(b^{k+1}-a^{k+1}\right)$$

Refer to https://www.colorado.edu/amath/sites/default/files/attached-files/quadrature.pdf

This results in $n+1$ equations with $n+1$ unknowns.

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_0 & x_1 & x_2 & \dots & x_n \\ (x_0)^2 & (x_1)^2 & (x_2)^2 & \cdots & (x_n)^2 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ (x_0)^n & (x_1)^n & (x_2)^n & \cdots & (x_n)^n \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} (b-a)/1 \\ \left(b^2-a^2\right)/2 \\ \left(b^3-a^3\right)/3 \\ \vdots \\ \left(b^{n+1}-a^{n+1}\right)/(n+1) \end{bmatrix}$$

We see that the first matrix is simply the transposed Vandermonde Matrix, thus by invertibility, weights are unique.

Definition of Precision

If m is the highest natural number such that Q integrates all polynomials of degree $\leq m$, we say the **precision** of Q is m.

Notation: I for integral

Midpoint Rule: $I(F) \approx Q(F)=M(F)=I\left(p_0\right)=(b-a)F\left(\frac{a+b}{2}\right)$

Note The degree of interpolating polynomial n = 0, but precision is higher (m = 1)

$M(x^0)=M(1)=(b-a)(1)=I(1)$

$M(x^1)=(b-a)(\frac{a+b}{2})=\frac{b^2-a^2}{2}=I(x)$

$M(x^2)=(b-a)(\frac{a+b}{2})^2 \neq \frac{b^3-a^3}{3}=I(x^2)$

Trapezoid Rule: $I(F) \approx Q(F)=T(F)=I\left(p_1\right)=\frac{b-a}{2}[F(a)+F(b)]$

Note The degree of interpolating polynomial n = 1, precision m = 1

$T(x^0)=\frac{b-a}{2}(1+1)=b-a=I(1)$

$T(x^1) = \frac{b-a}{2}(a+b) = \frac{b^2-a^2}{2} = I(x)$

$T(x^2) = \frac{b-a}{2}(a^2+b^2) \neq \frac{b^3-a^3}{3} = I(x^2)$

Simpson's Rule: $I(F) \approx Q(F) = S(F) = I\left(p_2\right) = \frac{b-a}{6}\left[F(a) + 4F\left(\frac{b+a}{2}\right) + F(b)\right]$

Exercise: Write this in standard form (weight times node)

Note The degree of interpolating polynomial n = 2, precision m = 1

$S(x^0) = \frac{b-a}{6}(1+4+1) = b-a = I(1)$

$S(x^1) = \frac{b-a}{6}(a+4(\frac{b+a}{2})+b) = \frac{(b+a)(b-a)}{2} = \frac{b^2-a^2}{2} = I(x)$

$S(x^2) = \frac{b-a}{6}(a^2+4(\frac{b+a}{2})^2+b^2) = \frac{b-a}{6}(a^2+b^2+(b+a)^2) \neq I(x^2)$

Derivation: $\frac{2M+T}{3}$

Error in Interpolatory Quadrature

$$\begin{aligned} E_n &= I(F) - Q_n(F) \\ &= \int_a^b [F(x) - p_n(x)]dx \\ &= \int_a^b \left(\frac{F^{(n+1)}(\eta)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\right)dx \quad \leftarrow \text{sub in polynomial interpolation error} \end{aligned}$$

Will interpolatory quadrature rules converge to $I(F)$ as $n \to \infty$?

Theorem Let F be continuous on $[a,b]$ and $Q_n(F) = \sum_{i=0}^{n} A_i^{(n)} F\left(\left(x_i\right)^{(n)}\right)$

Then $\lim_{n\to\infty} Q_n(F) = I(F) \iff \exists k \in \mathbb{R}, \ni: \sum_{i=0}^{n}\left|A_i^{(n)}\right| \leq k, \forall n \in \mathbb{N}$

Question: what about the Runge Phenomenon?