Math 4250 Notes

Tony Xu, tlx2

August 2023

Textbook: *An Introduction to Numerical Analysis* by Endre Süli and David Mayers, Cambridge University Press.

Instructor: Yunan Yang

Course Description: Introduction to the fundamentals of numerical analysis: error analysis, approximation, interpolation, numerical integration. In the second half of the course, the above are used to build approximate solvers for ordinary and partial differential equations. Strong emphasis is placed on understanding the advantages, disadvantages, and limits of applicability for all the covered techniques. Computer programming is required to test the theoretical concepts throughout the course.

Contents

1 Introduction to numerical computing (8/22/23)

Motivations of Numerical Methods.

Example 1.1. *Root finding*

Consider

$$ax^2 + bx + c = 0; \quad a \neq 0, b^2 - 4ax \geq 0.$$

We know the solution is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

What if our polynomial is more complicated? Can we still do the same thing?

Theorem 1.2. Abel-Ruffini Theorem *There is no algebraic formula (ie. $+, -, \times, /$, etc) for high degree polynomial roots ($n \geq 5$).*

Example 1.3. *Interpolation* We are given discrete measurements at different times (t_1, t_2, ..., t_n) and functions $(u(t_1), u(t_2, \dots)$. When we do interpolations we need to consider

- Known data
- What we think is best

Example 1.4. *Integration/Quadratrue*

Consider

$$\int_{-1}^{1} e^{-x}(\cos(6x))^8(\sin(15x))^9 dx,$$

which does not have an analytic formula for the anti-derivative of the integrand. Then, we can try to numerically approximate it if it is finite.

Trapezoid rule:

$$\int_a^b f(x)dx \approx \frac{b-a}{2n}\left[f(x_1) + 2\sum_{k=2}^{n} f(x_k) + f(x_{k+1})\right];$$

$$x_k = a + \frac{b-a}{n}(k-1), \ k = 1, \dots, n+1.$$

Example 1.5. *Differential equations*

Consider the nonlinear system for the flow of turbulence, Lozrenz-63:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$
$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z,$$

with some given initial condition. We will see that small perturbations to the initial conditions will lead to very different solution (ie. chaos).

When we say solve, we always mean *approximately solve.*

- **Error**:

$$e = \underbrace{x}_{\text{Approximate}} - \underbrace{x^*}_{\text{ground truth}}$$

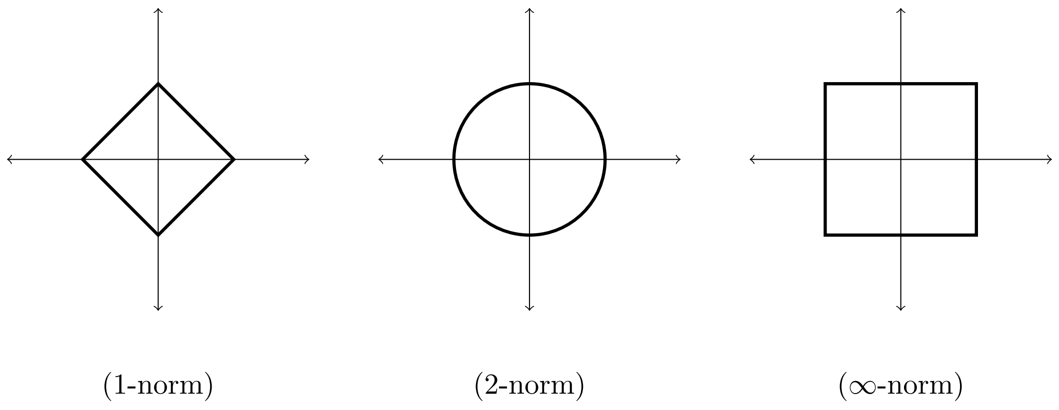


Figure 1: Inside the shapes represent $\|x\|_p < 1$

- **Absolute error**:
$$\hat{e} = \|x - x^*\|,\ x \in \mathbb{R}$$
Note that we choose the norm (ie. 1, 2, or sup norm).
- **Relative Error:** Difference in contrast to ground truth.
$$\tilde{e} = \frac{\hat{e}}{\|x^*\|} = \frac{\|x - x^*\|}{\|x^*\|}$$
We will use this most often. For example, when working with small numbers the absolute error will seem small.

Sources of errors

1. Blunders (human)
2. Modelling error
3. Data uncertainty
4. Discretization error (our focus)

2 Floating Point and Errors (8/24/23)

Recap: Some norms we typically use: Given $x \in \mathbb{R}^2, x = (x_1, x_2)^T$, some typical norms we will use are

$$\begin{aligned}\|x\|_1 &= |x_1| + |x_2|,\\ \|x\|_2 &= \sqrt{|x_1|^2 + |x_2|^2},\\ \|x\|_\infty &= \max |x_1|, |x_2|.\end{aligned}$$

From Figure 1, we see that as the p for the p-norm increases, we encompass more points. At higher dimensions, this can be quite significant.

Rounding errors: Say we are given π, where our machine can only store 4 digits:

$$\begin{aligned}\pi &= 3.1415926535\ldots,\\ &\approx 3.141, \quad \text{Truncation error}\\ &\approx 3.142, \quad \text{Rounding/Round off error}\end{aligned}$$

Modern computers typically use rounding.

Floating-point representation: A floating point number

$$x = \pm(d_0.d_1d_2\ldots d_{t-1})_\beta \times \beta^e$$

- $d_0 \neq 0$ (typical case)
- $d_0, \ldots, d_{t-1} \in \{0, 1, \ldots, \beta - 1\}$
- e: an integer such $L \leq e \leq U$

We can write out the number more explicitly as

$$x = \pm\left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_{t-1}}{\beta^{t-1}}\right) \times \beta^e.$$

Things that determine the representation: β (base), L (lower) , U (upper) , t (number of digits). So $\mathbb{F}_{[\beta,t,L,U]}$ is all numbers in this system. We represent 0 by all the $d_i = 0$.

Example 2.1. Given a system with $\beta = 2,\ t = 4,\ U = 1,\ L = -2$, draw all the positive numbers represented by this system.

	$e=-2$	$e=-1$	$e=1$	$e=1$
1.00	1/4	1/2	1	2
1.01	5/16	5/8	5/4	5/2
1.10	3/8	3/4	3/2	3
1.11	7/16	7/8	7/4	7/2

The distribution of the numbers represented are closer to smaller numbers around 0. So, the absolute error will be smaller for small numbers when dealing with roundoff. But not necessarily for relative error. This distribution is universally true.

Upper bound for absolute error, look at past the last digit possible from truncation.

$$\frac{\beta}{\beta^t}\beta^e, \quad \text{for truncation}$$

$$\frac{1}{2}\beta^{-(t-1)}\beta^e, \quad \text{for rounding}$$

Relative error

$$\frac{\frac{1}{2}\beta^{-(t-1)}\beta^e}{(d_0d_1\ldots d_{t-1})\beta^e} = \frac{\frac{1}{2}\beta^{-(t-1)}}{\underbrace{(d_0d_1\ldots d_{t-1})}_{\geq 1}} \leq \frac{1}{2}\beta^{-(t-1)}.$$

Truncation:

$$\frac{\beta^{-(t-1)}\beta^e}{(d_0d_1\ldots d_{t-1})\beta^e} \leq \beta^{-(t-1)}.$$

2.1 Binary

The IEEE 754 standard for floating point:

- 32-bit
  - first bit: sign, $s \in \{0,1\}$. 0 is positive, 1 is negative
  - next 8 bits are for exponent: $(00000000)_2 = 0 \to (11111111)_2 = 255$. But the actual range for exponents are $1 \le \text{exponent} \le 254$. Let $e = \text{exponent} - 127$, then $-126 \le e \le 127$.
  - rest 23 bits are for Mantissa/fraction. We know that $d_0 = 1$ always, so it is implied implicitly. It is actually store as $1.d_1 d_2 \dots d_{23}$, so $t = 24$ for this system.

exponents	fraction $= 0$	fraction $\neq 0$	number
(00000000)	zero	subnormal	
$1 \le \text{exponent} < 254$	normal	normal	
(11111111)	$\pm$ inf	NaN	

- 64-bit: similar concept as above just twice the amount of bits

Machine precision: upper bound for the relative error. In MATLAB,

```
>> eps
ans = 2.2204e-16
```

for 64-bit system with $t = 53$. This $\epsilon_{\text{machine}} = 2^{-52}$.

2.2 Loss of accuracy

Example 2.2. Consider the MATLAB code:

```
>> format long
>> a = 2^53;
>> b = 2^53 + 1;
>> b-a
ans = 0
```

- $+, -$: can shift decimal places and lose accuracy
- $\times$: n-digits $\times$ n-digits $\to$ $2n$-digits
- div: Example: $2/3 = 0.6666$ loses accuracy

Takeaway: *the more operations you do, the more accuracy you lose.*

Example 2.3. Consider $x + x^2 + x^3$. To reduce operations, we can rewrite it as $x(1 + x + x^2)$.

Cancellation Error: A particular case for subtraction of 2 numbers of similar magnitude.

$$1.111 \times 10^5 - 1.1109 \times 10^5 = 2 \times 10^0.$$

We lose a lot of precision here ($10^5 \to 10^0$).

3 Simple iteration (8/29/23)

3.1 Stability

Consider the ground truth

$$x^* \to f(x) \to y^*.$$

What really goes into the algorithm is

$$x^* + \Delta x \to f(x) \to y^* + \Delta y.$$

Now we want to measure stability (error in vs error out). The *condition number* is relative error in output over relative error in input

$$\begin{aligned}\kappa(f, x^*) &= \sup_{\|\Delta x\|>0} \frac{\|f(x^*+\Delta x) - f(x^*)\|/\|f(x^*)\|}{\|x^* + \Delta x - x^*\|/\|x^*\|}, \\ &= \sup_{\|\Delta x\|>0} \frac{\|x^*\|\|f'(x^*)\|}{\|f(x^*)\|}, \quad \text{under certain conditions}\end{aligned}$$

We can see that if $f(x^*)$, has very high slope, have bad condition number and problem is not stable/how error propogates.

3.2 Root-finding

Recall from Theorem 1.2, that we have no analytic solutions for polynomials for powers higher that 5. Today, we are going to look at problems like

$$\begin{aligned} f(x) &= 0, \quad x \in \mathbb{R}^n \\ g(X) &= 0, \quad X \in \mathbb{R}^{m\times n}\end{aligned}$$

Things we will consider are

1. If the solution exits
2. How can we find them numerically

Consider a function $f : [a, b] \to \mathbb{R},\ a \neq b$.

Theorem 3.1. *If f is continuous, and $f(a) \le 0, f(b) \ge 0$ (or vice versa), then $\exists \xi \in [a, b],\ f(\xi) = 0$.*

Proof.
- If $f(a)$ or $f(b) = 0$, then $\xi = a$ or b.
- If $f(a), f(b) \neq 0$, then either $f(a) < 0, f(b) > 0$ or $f(a) > 0, f(b) < 0$. Using the Intermediate Value Theorem, then ξ exists.

□

Theorem 3.2. (Brouwer fixed-point theorem) *Consider $g : [a, b] \to \mathbb{R}$, continuous, $\forall x \in [a, b], g(x) \in [a, b]$. Then $\exists$ a* fixed point *$\xi \in [a, b]$ such that $\xi = g(\xi)$.*

Proof. Define $f(x) = x - g(x)$. Since g is continuous, then $f(x)$ is also continuous. Next, check end points. $f(a) = a - g(a) \le 0$ since $a \le g(a) \le b$. $f(b) = b - g(b) \ge 0$. By Theorem 3.1, $\exists \xi, f(\xi) = 0 \implies \xi = g(\xi)$. □

Example 3.3.

$$f(x) = e^x - 2x - 1 \text{ on } [1, 2]$$

.

1. Verify $f(x)$ has at least 1 root in $[1, 2]$.

   Yes. f is continuous and $f(1) < 0$ and $f(2) > 0$. By Theorem 3.1, exists roots.

2. Can you find fixed-point equation $x = g(x)$ with x^* root satisfy $x^* = g(x^*)$

$$\begin{aligned} 0 &= e^{x^*} - 2x^* - 1 \\ x^* &= \underbrace{e^{x^*} - x^* - 1}_{g_1(x^*)} \\ &\text{OR} \\ 2x^* &= e^{x^*} - 1 \\ x^* &= \underbrace{\frac{e^{x^*} - 1}{2}}_{g_2(x^*)} \\ &\text{OR} \\ e^{x^*} &= 2x^* + 1 \\ x^* &= \underbrace{\log(2x^* + 1)}_{g_3(x^*)} \end{aligned}$$

Definition 3.4. *Simple fixed-point iteration (Picard iteration).* $g : [a, b] \to \mathbb{R}$, g continuous and $g(x) \in [a, b]$ $\forall x$. Then given $x_0 \in [a, b]$, then

$$x_{k+1} = g(x_k), \quad k = 0, 1, 2, \ldots$$

This looks like

$$x_0 \xrightarrow{g} g(x_0) = x_1 \xrightarrow{g} g(x_1) = x_2 \xrightarrow{g} \ldots$$

If $\{x_k\}$ converges. Then $\lim_{k\to\infty} x_k = \eta$. Then using the output of continuous functions are continuous

$$\eta = \lim_{k\to\infty} x_k = \lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} g(x_k) = g\left(\lim_{k\to\infty} x_k\right) = g(\eta).$$

Implies then that $g(\eta) = \eta$.

Definition 3.5. *(Contraction)* $g : [a, b] \to \mathbb{R}$. We call g a contraction if $\forall x, y \in [a, b]$

$$\|g(x) - g(y)\| \le L\|x - y\|, \quad 0 \le L < 1.$$

Theorem 3.6. (Contraction guarantees convergence). *If $g : [a, b] \to \mathbb{R}$, $g(x) \in [a, b]$ $\forall x$, g is continuous and contraction, then $\exists$ a* unique *fixed point ξ such that $\xi = g(\xi)$ and the simple iteration $\{x_k\}$ will converge $\forall x_0 \in [a, b]$.*

Proof. Separate into parts.

- *Existance:* by Theorem 3.2.

- *Uniqueness:* Suppose $\exists \xi \neq \eta$ such that $g(\xi) = \xi$ and $g(\eta) = \eta$. Then

$$\|\xi - \eta\| = \|g(\xi) - g(\eta) \leq L\|\xi - \eta\|, \quad L < 1.$$

  But $\|\xi - \eta\| \leq L\|\xi - \eta\|$ is a contradiction. Thus $\xi = \eta$.

- *Convergence:* Look at how each step is far from our fixed point:

$$\begin{aligned} \|x_k - \xi\| &= \|g(x_{k-1}) - g(\xi)\| \leq L\|x_{k-1} - \xi\|, \\ &\leq L^2\|x_{k-2} - \xi\| \leq \cdots \leq L^k\|x_o - \xi\|. \end{aligned}$$

$$\text{taking the limit then, } \lim_{k\to\infty} \|x_k - \xi\| \leq \underbrace{\left(\lim_{k\to\infty} L^k\right)}_{\to 0} \|x_0 - \xi\| = 0.$$

□

In practice, we want to look to an error tolerance $\epsilon > 0$. The question is, given I know the algorithm converges, what is a number of iterations to reach error $< \epsilon$.

Theorem 3.7. *g is continuous and a contraction. $g : [a, b] \to [a, b]$. For any $\epsilon > 0$ fixed, we will have $\|x_k - \xi\| \leq \epsilon$ for x_k from simple iteration and ξ fixed point*

$$\forall k \geq k_0(\epsilon) = \left[\frac{\ln\|x - x_0\| - \ln(\epsilon - \epsilon L)}{\ln(1/L)}\right].$$

Where ($[x] \leq x,\ [x]$ integer).

Proof. We know that

$$\|x_k - \xi\| \leq L^k\|x_0 - \xi\| \leq \epsilon.$$

Use triangle inequality

$$\begin{aligned} \|x_0 - \xi\| &= \|x_0 - x_1 + x_1 - \xi\| \\ &\leq \|x_0 - x_1\| + \|x_1 - \xi\| \\ &\leq \|x_0 - x_1\| + L\|x_0 - \xi\| \\ (1 - L)\|x_0 - \xi\| &\leq \|x_0 - x_1\| \end{aligned}$$

So then now we can say that

$$\|x_k - \xi\| \leq L^k\|x_0 - \xi\| \leq \frac{L^k}{1 - L}\|x_0 - x_1\| \leq \epsilon$$

Then

$$\begin{aligned} L^k &\leq \frac{\epsilon(1 - L)}{\|x_0 - x_1\|} \\ k \underbrace{\ln L}_{<0} &\leq \ln(\epsilon - \epsilon L) - \ln\|x_0 - x_1\| \\ k &\geq \frac{\ln(\epsilon - \epsilon L) - \ln\|x_0 - x_1\|}{\ln(L)} \end{aligned}$$

□

4 Newton and Bisection (8/31/23)

Example 4.1. $f(x) = e^x - 2x - 1$ on $x \in [1, 2]$. To find the root, we saw 3 ways to make this a fixed point equation

$$g_1(x) = \frac{e^x - 1}{2}, \quad g_2(x) = e^x - x - 1, \quad g_3 = \ln(2x + 1).$$

We can apply the mean value theorem $\|g(x) - g(y)\| = \|g'(\eta)\|\|x - y\|,\ \eta \in [x, y]$. We want to find the worst case for the derivative.

$$\begin{array}{ccc} g_1' = \frac{e^x}{2} & g_2' = e^x - 1 & g_3' = \frac{2}{2x+1} \\ g_1'(1) = \frac{e}{2} > 1 & g_2'(1) = e - 1 > 1 & g_3'(1) = \frac{2}{3} < 1 \\ g_1'(2) = \frac{e^2}{2} > 1 & g_2'(2) = e^2 - 1 > 1 & g_3'(2) = \frac{2}{5} < 1 \end{array}$$

g_3 is a contraction, so it is the one we choose. $L = 2/5$ in this domain.

Definition 4.2. $g : [a, b] \to \mathbb{R}$ is continuous, $g(\xi) = \xi$ (existence). We say ξ is *stable* if $\{x_k\} \to \xi$ if x_0 is close to ξ. ξ is *unstable* of $\{x_l\} \to \xi$ only if $x_0 = \xi$ (even if ϵ close).

Remark 4.3. There exist fixed points that are neither stable nor unstable.

To determine if a fixed point is stable, where g differentiable:

- If $\|g'(\xi)\| < 1$ then stable,
- If $\|g'(\xi)\| > 1$ then unstable,
- If $\|g'(\xi)\| = 1$ then inconclusive.

Definition 4.4. *Order of convergence.* Consider $\{x_k\} \to \xi$. If

$$\lim_{k\to\infty} \frac{\|x_{k+1} - \xi\|}{\|x_k - \xi\|^q} = \mu < \infty,$$

then we say $\{x_k\} \to \xi$ with order q.

From this, we see that fixed point is linear convergence.

4.1 Bisection

f continuous of $[a, b]$ such that $f(a)f(b) < 0$. Then $\exists \xi \in [a, b]$ such that $f(\xi) = 0$. The steps of the method are

1. Calculate the midpoint $c = \frac{a+b}{2}$.
2. Evaluate $f(c)$.
3. If $f(c)$ good enough, return
4. If not, examine the sign of $f(c)$ and replace either $(a, f(a))$ or $(b, f(b))$ with $(c, f(c))$ so that there is a zero crossing within the new interval.

Theorem 4.5. (Convergence of bisection) *f continuous of $[a, b]$ such that $f(a)f(b) < 0$.*

$$\|x_n - \xi\| \le \frac{b - a}{2^n},\ n = 1, 2, 3, \ldots$$

Proof. The search domain at the n-th iteration is $[a_n, b_n]$. The size is

$$|b_n - a_n| = \frac{1}{n^2}|b-a|, \ n = 0, 1, \ldots, k.$$

We know $x_{n+1} = \frac{1}{2}(a_n + b_n), \ n = 0, 1, \ldots,$ then

$$|x_{n+1} - \xi| \leq \frac{1}{2}(b_n - a_n) = \frac{b-a}{2^{n+1}}$$

□

Bisection is linear convergence, because we only half the interval every time

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|} = \frac{1}{2}.$$

4.2 Newton's method

This requires $f'(x)$. Newton's method is

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \ f'(x_k) \neq 0.$$

We can see that Newton is a fixed point iteration. We are finding the root of the linear approximation of $f(x_k)$.

Quadratic convergence ($q = 2$). Taylor's theorem with remainder by expanding about x_k.

$$0 = f(\xi) = f(x_k) + (\xi - x_k)f'(x_k) + \frac{|\xi - x_k|^2}{2}f''(\eta), \ \eta \in [x_k, \xi] \text{ or } [\xi, x_k]$$

Assume $f'(x_k) \neq 0$.

$$0 = \frac{f(x_k)}{f'(x_k)} + \xi - x_k + \frac{|\xi - x_k|^2}{2}\frac{f''(\eta)}{f'(x_k)}$$

Then subtracting

$$|x_{k+1} - \xi| = \frac{|x_k - \xi|^2}{2}\left|\frac{f''(\eta)}{f'(x_k)}\right|$$

gives us the ratio

$$\frac{|x_{k+1} - \xi|}{|x_k - \xi|^2} = \frac{1}{2}\left|\frac{f''(\eta)}{f'(x_k)}\right|.$$

We see quadratic convergence.

4.3 Secant method

If gradient is expensive, can use secant. Without knowing $f'(x_k)$, we can estimate

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1}}{x_k - x_{k-1})}.$$

This gives us the iteration

$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}}$$

For secant method, $q = 1.6$, *superlinear* convergence.

5 Lagrange Interpolation (9/5/23)

Alex Townsend lecture.

Polynomial interpolation. Given $x_0, \dots, x_n \in \mathbb{R}$ (distinct) and values $y_0, \dots, y_n \in \mathbb{R}$, find a polynomial of degree $\leq n$ such that $y_j = p_n(x_j),\ j = 0, 1, \dots, n$. $p_n \in \mathcal{P}_n =$ space of polynomials of degree $\leq n$.

- Interpolation. There is an *unknown* function that has been evaluated. $(x_0, f(x_0)),\ \dots,\ (x_n, f(x_n))$ data. Find p_n, which will fill in the gaps.

  Example 5.1. Let $f : [0, T) \to \mathbb{R}$ (unknown) continuous. $f(t) =$ glucose level in patient's blood at time t. It has been samples at times $t_0 < t_1 < \cdots < t_n \leq T$ with $f(t_0), f(t_1), \dots, f(t_n)$. Task: Find $p_n \in \mathcal{P}_n$ such that $p_n(t_j) = f(t_j),\ j = 0, \dots, n$.

- Discretization. There is a *known* function that can be evaluated anywhere. But, I'll make a $p_n \mathcal{P}_n$ such that $\max_{x \in [a,b]} |f(x) - p_n| < 10^{-16}$. The idea is the computer cannot tell the difference between the functions.

  Example 5.2. Let's say I want

$$\underbrace{\int_a^b f(x)\,\mathrm{d}x}_{\text{symbolic}} \overbrace{\approx}^{10^{-16}} \underbrace{\int_a^b p_n(x)\,\mathrm{d}x}_{\text{numerically}}.$$

  p_n is a *proxy* of f.

5.1 Lagrange Polynomials

Solving polynomial interpolation.

$$\begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} = v_0 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + v_1 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + v_n \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.$$

The *Lagrange Polynomials* are

$$\ell_k(x) = \begin{cases} 1, & x = x_k \\ 0, & x \neq x_k\ (x = x_j,\ j \neq k) \end{cases}$$

with $\ell_k \in \mathcal{P}_n$. Then, we can write

$$\ell_k(x) = c(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n).$$

Constant c is determined by solve $\ell_k(x_k) = 1$. Then

$$\begin{aligned} \ell_k(x) &= \frac{(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}{(x_k - x_0) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)}, \text{ for } 0 \leq k \leq n \\ &= \frac{\prod_{j=0,\ j \neq k}^n (x - x_j)}{\prod_{j=0, j \neq k}^n (x_k - x_j)} \in \mathcal{P}_n. \end{aligned}$$

Definition 5.3. *Lagrange form of p_n.* Then we can write

$$p_n(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x) \in \mathcal{P}_n.$$

We can confirm that $p_n(x_j) = f(x_j)\ell_j(x_j) = f(x_j)$

Example 5.4. Find p_3 such that

NODES	5	-7	-6	0
SAMPLES	1	-23	-54	-954

.

Then

$$\begin{aligned}
\ell_0(x) &= \frac{(x+7)(x+6)x}{(5+7)(5+6)5} = \frac{1}{600}(x+7)(x+6)x \\
\ell_1(x) &= \frac{(x-5)(x+6)x}{(-7-5)(-7+6)(-7)} = -\frac{1}{84}(x+5)(x+6)x \\
\ell_2(x) &= \frac{(x-5)(x+7)x}{(6-5)(6+7)6} = \frac{1}{66}(x-5)(x+7)x \\
\ell_3(x) &= \frac{1}{210}(x+7)(x+6)(x-5) \\
\implies p_3(x) &= \ell_0(x) - 23\ell_1(x) - 53\ell_2(x) - 954\ell_3(x)
\end{aligned}$$

Example 5.5. Find p_5 such that

NODES	-1	-1/2	0	1/3	9/10	11/10
SAMPLES	1	1/4	0	1/9	81/100	121/100

.

$$p_5(x) = \ell_0(x) + \frac{1}{4}\ell_1(x) + \frac{1}{9}\ell_3(x) + \frac{81}{100}\ell_4(x) + \frac{121}{100}\ell_t(x) = x^2.$$

Theorem 5.6. *Let $f : [a,b] \to \mathbb{R}$ be a continuous function with $-\infty < a < b < \infty$ and $a \leq x_0 < x_1 < \cdots < x_n \leq b$ nodes. Then there* exists *a* unique *polynomial p_n of degree $\leq n$ such that $p_n(x_j) = f(x_j),\ j - 0, \ldots, n$.*

Proof. Existence is proven by construction. Uniqueness. Suppose there are two $p_n, q_n \in \mathcal{P}_n$ and $p_n(x_j) = f(x_j),\ q_n(x_j) - f(x_j)$ for $j = 0, \ldots, n$. Then $r_n(x) = p_n(x) - q_n(x) \in \mathcal{P}_n$. Where $r_n(x_j) = p_n(x_j) - q_n(x_j) = f(x_j) - f(x_j) = 0$ for $j = 0, 1, \ldots, n$. This means we have an n-degree polynomial with $n+1$ roots. Therefore $r_n(x) \equiv 0$. □

5.2 Error and Points

Suppose $f \in C^{n+1}$ and p_n is the polynomial interpolant at n points. $\xi \in [x_0, x_n]$. Then

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

Where should we pick x_k? Using the formula, we can try to minimize

$$\left| \prod_{k=0}^{n} (x - x_k) \right|.$$

Equally spaced nodes cases exponentially scaling error at the endpoints. Cheb points, throw electrons at a wire and how they space at equilibrium.

6 Polynomial Interpolation (9/7/23)

Alex Townsend lecture. **Continuing Error and Points.**

Typically, we set $M = \max_{x\in[x_0,x_n]} \left|f^{(n+1)}(x)\right|$, then

$$|f(x) - p_n(x)| \leq \frac{M}{(n+1)!} |\ell(x)|, \quad \ell(x) = \prod_{j=0}^{n} (x - x_j)$$

Equispaced points. n-equally spaced points $x_j = a + jh,\ h - \frac{b-a}{n},\ j = 0, \ldots, n$. We saw $n = 40,\ a = -1,\ b = 1$ gave us error up to 10^{10} at the edges. More analytically, we see that max is around the edge, so let's say $x^* = \frac{x_0+x_1}{2}$

$$\begin{aligned}\max_{x\in[a,b]} |\ell(x)| \geq |\ell(x^*)| = \prod_{j=0}^{n} |(x^* - x_j)| &\geq \underbrace{\frac{h}{2}}_{(j=0)} \cdot \underbrace{\frac{h}{2}}_{(j=1)} \cdot \underbrace{\frac{3h}{2}}_{(j=2)} \cdots \underbrace{(j - \frac{1}{2})h}_{j} \cdots \underbrace{(n - \frac{1}{2})h}_{j=n} \\ &= \frac{1}{2} \underbrace{\prod_{j=1}^{n} \left(j - \frac{1}{2}\right)}_{\text{ABSOLUTELY HUGE}} h^{n+1}\end{aligned}$$

grows exponentially with n.

Chebyshev points. The points

$$x_j = \cos\left(\frac{2j+1}{2n}\pi\right),\ a = -1,\ b = 1$$

are equally spaced on the complex unit semicircle, then take the projection onto the real axis. We have quadratic $\frac{1}{n^2}$ clustering at the endpoints such that the error grows only logarithmically. (Riesz, 1916). Runge 1909 also wrote this down in a Nature paper.

Example 6.1. *Runge's example.* Consider $f(x) = \frac{1}{1+25x^2}$ on $[-1, 1]$, which is infinitely differentiable. He proved that for the equally spaced interpolant:

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \text{ grows exponentially with } n.$$

Divergence at enpoints. Even though in reals, they are "feeling" the effects of the poles. For Chebyshev points, however,

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \text{ decays exponentially with } n.$$

Remark 6.2. *Comparing to Taylor.* Recall the Taylor expansion, where $f \in C^{n=1}([a, b])$.

$$f(x) - t_n(x) = f(x) - \sum_{j=0}^{n} \frac{f^{(j)}(x_0)}{j!}(x - x_0)^j = \frac{f^{(n+1)}(\eta_x)}{(n+1)!}(x - x_0)^{n+1},\ \eta_x \in [x_0, x_n].$$

We say polynomial interpolation is like a multi-point Taylor expansion.

Consider the case where we want to add points to "update" our interpolation, say in a time-dependent setting.

Definition 6.3. *Barycentric formula*

$$\ell(x) = \prod_{j=0}^{n}(x - x_j), \quad \ell_k(x) = \frac{\ell(x)}{x - x_k}\frac{1}{\ell'(x_k)}.$$

This is from all sums 0 except when $s = k$.

$$\ell'(x) = \sum_{s=0}^{n} \prod_{j=0,\ j\neq s}^{n} (x - x_j)$$

$$\ell'(x_k) = \prod_{j=0,\ j\neq k}^{n} (x_k - x_j)$$

Now we have weights, $\nu_k = \frac{1}{\ell'(x_k)}$, then the FIRST KIND formula $\ell_k(x) = \ell(x)\frac{\nu_k}{x-x_k}$.

Then

$$p_n(x) = \ell(x)\sum_{k=0}^{n}\frac{f(x_k)\nu_k}{x - x_k}.$$

The SECOND KIND is

$$p_n(x) = \frac{p_n(x)}{1} = \frac{\cancel{\ell(x)}\sum_{k=0}^{n}\frac{f(x_k)\nu_k}{x-x_k}}{\cancel{\ell(x)}\sum_{k=0}^{n}\frac{\nu_k}{x-x_k}} = \frac{\sum_{k=0}^{n}\frac{f(x_k)\nu_k}{x-x_k}}{\sum_{k=0}^{n}\frac{\nu_k}{x-x_k}}$$

If x close to x_k can get cancellation error. But since you do the same error in numerator and denominator, the error can cancel when they are divided, leading to stability.

Riesz, 1996 worked out how to choose the ν_k for Chebyshev points $x_j = \cos\left(\frac{2j+1}{2n}\pi\right)$ $v_0 = \frac{1}{2},\ v_1 = (-1)^j,\ j = 1, \ldots, n-1,\ v_n = \frac{1}{2}(-1)^n$.

7 Newton-Cotes formulae (9/12/23)

Recall that

$$p_n(x) = \sum_{k=0}^{n}\ell_k(x), \quad \ell_k(x) = \frac{\prod_{i\neq k,\ i=0}^{n}(x - x_i)}{\prod_{i\neq k,\ i=0}^{n}(x_k - x_i)} = \begin{cases}1, & x = x_k\\ 0, & x \neq x_k\end{cases}.$$

More on Lagrange. Recall that we have an exact error formula. Consider if $f^{(n+1)}(x) > 0$ on $[a, b]$ then it "oscillates" above and below each interpolated point. This is true since by looking at the error term, we see that as x goes from $x_k \to x_{k+1}$, $\ell(x)$ will flip a sign.

7.1 Hermite Interpolation

Given

x	x_0	x_1	$\cdots$	x_n
$f(x)$	y_0	y_1	$\cdots$	y_n
$f'(x)$	z_0	z_1	$\cdots$	z_n

We have $2n + 2$ points so can define $2n + 1$ degree polynomial exactly.

7.2 From Numerical Interpolation to Numerical Integration

The problem is

$$\int_a^b f(x)\,\mathrm{d}x.$$

What if we come across functions that don't have analytic anti-derivatives? This is the case many times. Cases where we want to use numerical integration:

- $f(x)$ does not have anti-derivative,
- Approximate $\int_a^b f(x)\,\mathrm{d}x$ quickly.

Idea: $f(x) \approx \mathcal{P}_n(x)$ since integrating polynomials is easy from the power rule ($\int x^n\,\mathrm{d}x = \left.\frac{x^{n+1}}{n+1}\right|_a^b$). Given f, we have to choose $\{(x_i, y_i = f(x_i))\}$. Where $\{x_i\}$ are the quadrature points/nodes. Different choices of points gives us different quadrature methods.

Newton-Cotes Method. Choose equally spaced points

$$x_i = a + \frac{b-a}{n} i, \quad i = 0, 1, \ldots, n,$$

which will also give us $f(x_i) = y_i$. This will give us a unique $p_n(x)$ from Lagrange interpolation. Then

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \int_a^b p_n(x)\,\mathrm{d}x,\\ &= \int_a^b \sum_{k=0}^n \ell_k(x) f(x_k)\,\mathrm{d}x,\\ &= \sum_{k=0}^n f(x_k) \underbrace{\int_a^b \ell_k(x)\,\mathrm{d}x,}_{w_k}\\ &= \sum_{k=0}^n w_k f(x_k)\end{aligned}$$

Note that w_k has nothing to do with $f(x)$, so this can be computed beforehand. We see that the set $(\{x_k\},\ \{w_k\})$ determines a method.

Consider the case $n = 1$. Then $x_0 = a,\ x_1 = b$ and $y_0 = f(a),\ y_1 = f(b)$. Now we can plug into Lagrange interpolation formula,

$$\begin{aligned}p_1(x) &= \ell_0(x) f(a) + \ell_1(x) f(b),\\ &= \frac{x-b}{a-b} f(a) + \frac{x-a}{b-a} f(b).\end{aligned}$$

Then the integral

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b p_1(x)\,\mathrm{d}x &= \left(\int_a^b \frac{x-b}{a-b}\,\mathrm{d}x\right) f(a) + \left(\int_a^b \frac{x-a}{b-a}\,\mathrm{d}x\right) f(b),\\ &= \frac{b-a}{2}(f(a) + f(b)).\end{aligned}$$

This is the trapezoid rule.

Consider $n = 2$. Then $x_0 = a,\ x_1 = \frac{a+b}{2},\ x_2 = b$ and the corresponding y_i by evaluating $f(x_i)$. Then the polynomial is

$$p_2(x) = \ell_0(x)f(a) + \ell_1(x)f\left(\frac{a+b}{2}\right) + \ell_2(x)f(b).$$

Writing out the integral is

$$\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x &\approx \int_a^b p_2(x)\,\mathrm{d}x,\\
&= f(a)\underbrace{\int_a^b \frac{\left(x-\frac{a+b}{2}\right)(x-b)}{\left(a-\frac{a+b}{2}\right)(a-b)}\,\mathrm{d}x}_{\frac{b-a}{6}} + f\left(\frac{a+b}{2}\right)\underbrace{\int_a^b \frac{(x-a)(x-b)}{a\left(\frac{a+b}{2}-a\right)\left(\frac{a+b}{2}-b\right)}\,\mathrm{d}x}_{\frac{4(b-a)}{6}}\\
&\quad + f(b)\underbrace{\int_a^b \frac{(x-a)\left(x-\frac{a+b}{2}\right)}{(b-a)\left(b-\frac{a+b}{2}\right)}\,\mathrm{d}x}_{\frac{b-a}{6}}\\
&= \frac{b-a}{6}f(a) + \frac{4(b-a)}{6}f\left(\frac{b+a}{2}\right) + \frac{b-a}{6}f(b).
\end{aligned}$$

This is Simpson's rule. For both trapezoid and Simpson's, we see the weights are such that n Newton-Cotes, $\sum w_i = b-a$. It integrates the n-th polynomial exactly. This integration is accurate for any polynomial up to degree n. So if we let our integrand $f = C$ a constant, then it makes sense that the sum of weights always has to be $b-a$.

8 Runge phenomenon and Composite formulae (9/14/23)

In numerical analysis, after developing an algorithm to do approximations, we always look at the error next. Let's look at the error of Newton-Cotes for f continuous and $\in C^{n+1}([a,b])$:

$$\begin{aligned}
E_n(f) &= \int_a^b p_n(x)\,\mathrm{d}x - \int_a^b f(x)\,\mathrm{d}x,\\
|E_n(f)| &= \left|\int_a^b (p_n - f(x))\,\mathrm{d}x\right|,\\
&= \left|\int_a^b \left(\frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)\right)\right|,\\
&\le \frac{M_{n+1}}{(n+1)!}\left|\int_a^b (x-x_0)(x-x_1)\cdots(x-x_2)\,\mathrm{d}x\right|,
\end{aligned}$$

where $M_{n+1} = \max_{x\in[a,b]}\left|f^{(n+1)}(x)\right|$. Observe that

- we can integrate up to n-th degree polynomial exactly because $M_{n+1} = 0$,
- $(x-x_0)(x-x_1)\cdots(x-x_n)$ term. Look at case $n=3$, where $(x-x_0)(x-x_1)(x-x_2)(x-x_3)$. We will see that (sign denotes sign of the product function)

$$\begin{array}{ccccccc} & - & & + & & - & \\ \hline x_0 & & x_1 & & x_2 & & x_3 \end{array}$$

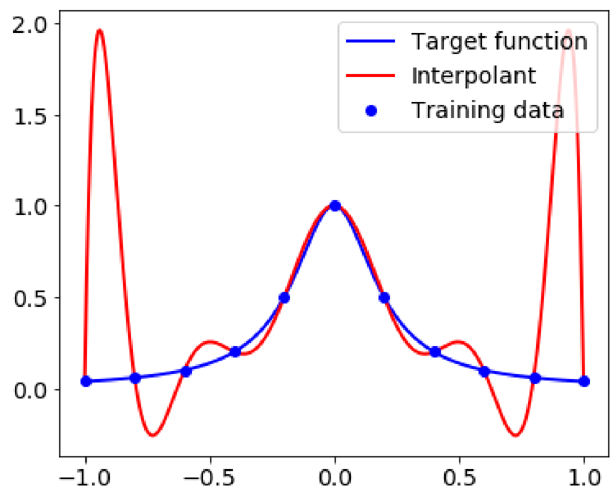
Consider the $n = 4$ case:

$$\begin{array}{ccccccccc} & + & & - & & + & & - & \\ \hline x_0 & & x_1 & & x_2 & & x_3 & & x_4 \end{array}$$

We see that for n even, we see that the integral over the product will give us 0!

$$\begin{cases} n \text{ odd}, & \text{exact at up to } n\text{-th polynomial}, \\ n \text{ even}, & \text{exact at up to the } (n+1)\text{-th polynomial} \end{cases}$$

Just as Lagrange interpolation suffers from Runge phenomenon, Newton-Cotes also suffers from the Runge phenomenon because of the product function in the error. Remember this is where we get exponential error at the edges. The problem is that the more points we use, the error becomes



greater, but with low points we can't approximate the function well. Solution?

8.1 Composite formulation

Divide and conquer. Consider an interval $[a, b]$

$$\begin{array}{ccccc} \hline a = x_0 & x_1 & x_2 & x_3 & b = x_4 \end{array}$$

Then
$$\int_a^b f(x)\,\mathrm{d}x = \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \int_{x_1}^{x_2} f(x)\,\mathrm{d}x + \cdots + \int_{x_3}^{x_4} f(x)\,\mathrm{d}x.$$

We can just use trapezoidal on one of the terms. Let $h = \frac{b-a}{4}$

$$\int_{x_1}^{x_2} f(x)\,\mathrm{d}x \approx \frac{h}{2} f(x_1) + \frac{h}{2} f(x_2).$$

Then combining it all together,

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \frac{h}{2}(f(x_0)+f(x_1)) + \frac{h}{2}(f(x_1)+f(x_2)) + \frac{h}{2}(f(x_2)+f(x_3)) + \frac{h}{2}(f(x_3)+f(x_4)) \\ &= h\left(\frac{1}{2}f(x_0)+f(x_1)+f(x_2)+f(x_3)+\frac{1}{2}f(x_4)\right).\end{aligned}$$

This is the composite trapezoidal rule.

Now let's look at the error on $[x_{i-1}, x_i]$:

$$|E_n[f]| \le \frac{M_2}{2}\left|\int_{x_{i-1}}^{x_i}(x-x_i)(x-x_{i-1})\,\mathrm{d}x\right|,$$

where $M_2 = \max_{[a,b]}\left|f^{(2)}(x)\right|$. Evaluating the integral gives us

$$= \frac{M_2}{2}\frac{h^3}{6}.$$

This is the error on the subinterval, which is independent of i. So then the total error on $[a,b]$ is just the total of how many intervals we have (m intervals)

$$\begin{aligned}|E_n[f]| &\le m\frac{M_2}{2}\frac{h^3}{6}, \\ &= \frac{b-a}{\cancel{h}}\frac{M_2}{2}\frac{h^{\cancel{3}2}}{6}, \\ &= \frac{(b-a)M_2}{12}h^2.\end{aligned}$$

Here, we do not suffer from Runge phenomenon. So, trapezoidal rule is second order convergence. We can also apply the same thing for Simpson's rule.

9 Inner Product Spaces (9/19/23)

Remark 9.1. *On order of errors.* We said error for iterations $\lim_{n\to\infty}\dfrac{\|e_{n+1}\|}{\|e_n\|^q} = \mu < \infty$, the the method converges with order q. We saw for bisection, $q=1$ with $\mu = \frac{1}{2}$. Newton has $q=2$. In homework 1, exponential convergence is about $\lim_{n\to\infty}\|e_n\| \approx O(\frac{1}{n})$ or some other order. Now relating the two ideas:

$$\begin{aligned}|e_{n+1}| &= \mu|e_n|^q \\ \log|e_{n+1}| &= \underbrace{\log\mu}_{\to 0,\text{ let }\mu=1} + q\log|e_n| \\ \log|e_n| &= q^n\log|e_0| \\ |e_n| &= e^{qn}|e_0|.\end{aligned}$$

For some $x \in \mathbb{R}^n$, $\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$ for $1 \le p < \infty$. Consider $\lim_{p\to\infty}\|x\|_p = \max\{|x_1|,\ldots,|x_n|\}$. Given a matrix $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$, there is also a sense of norms (induced norms) see Matrix computations courses for more details.

We want to create some structure for functions to get the difference between functions.

Definition 9.2. Let V be a linear space ($\forall f \in V,\ \forall \lambda \in \mathbb{R},\ \lambda f \in V$ and $\forall f, g \in V,\ f+g \in V$) over the field $\mathbb{R}$. Consider a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$. It must satisfy

1. $\langle f+g, h\rangle = \langle f, h\rangle + \langle g, h\rangle$ (linearity)
2. $\langle \lambda f, h\rangle = \lambda \langle f, h\rangle,\ \lambda \in \mathbb{R}$ (homogeneity)
3. $\langle f, g\rangle = \langle g, f\rangle$ (symmetric)
4. $\langle f, f\rangle > 0$ as long as $f \neq 0$. (positive definite)

Then V is an inner product space.

Example 9.3. (Euclidean Space) $V = \mathbb{R}^n$, then $\forall x, y \in \mathbb{R}^n$ then $\langle x, y\rangle = x^T y$.

Example 9.4. $V = \mathbb{R}^n$ and let $A \in \mathbb{R}^{n\times n}$ such $\langle x, y\rangle = x^T A y$. To satisfy the properties of an inner product, we see that is true iff A is symmetric positive definite (spd). We see that we can define an infinite amount of inner products from this.

Definition 9.5. V is an inner product space. $f, g \in V$. If $\langle f, g\rangle = 0$ means that f, g are orthogonal.

Note that orthogonality depends on the inner product.

Example 9.6. Let $A = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$, $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. We see that x and y are orthogonal if we define $\langle x, y\rangle = x^T y = 0$. But using the inner product defines with A, $\langle x, y\rangle = x^T A y = 2 \neq 0$.

Definition 9.7. V is an inner product space. $\langle \cdot, \cdot \rangle$ induces a norm for elements in V as

$$\|f\| = \sqrt{\langle f, f\rangle}.$$

Lemma 9.8. Cauchy-Schwartz Inequality

$$|\langle f, g\rangle| \leq \|f\|\|g\|.$$

Proof. Consider for $f, g \neq 0$

$$\begin{aligned} 0 \leq \|\lambda f + g\|_2^2 &= \langle \lambda f + g, \lambda f + g\rangle \\ &= \langle \lambda f + g, \lambda f\rangle + \langle \lambda f + g, g\rangle \\ &= \lambda^2 \langle f,\ f\rangle + \lambda \langle g,\ f\rangle + \lambda \langle f,\ g\rangle + \langle g,\ g\rangle \\ &= \underbrace{\lambda^2 \|f\|^2 + 2\lambda \langle f,\ g\rangle + \|g\|^2}_{h(\lambda)} \geq 0 \end{aligned}$$

We have $h(\lambda) = a\lambda^2 + b\lambda + c \geq 0$. To have a quadratic be ≥ 0, then it has at most one root. Then $b^2 - 4ax \leq 0$. So,

$$\begin{aligned} (2\langle f,\ g\rangle)^2 - 4(\|f\|^2)(\|g\|^2) \leq 0 \\ |\langle f,\ g\rangle| \leq \|f\|\|g\|. \end{aligned}$$

□

Looking at the proof, consider when $\lambda = 1$. Then

$$\begin{aligned}\|f+g\|^2 &= \|f\|^2 + 2\langle f,\ g\rangle + \|g\|^2,\\ &\leq \|f\|^2 + 2|\langle f,\ g\rangle| + \|g\|^2,\\ &\leq \|f\|^2 + 2\|f\|\|g\| + \|g\|^2,\\ &= (\|f\| + \|g\|)^2.\end{aligned}$$

Taking the square root gives us the triangle inequality $\|f+g\| \leq \|f\| + \|g\|$.

Example 9.9. Let $V = \{$all continuous functions defined on $[a, b]$ $\}$. This is clearly a linear space since sum of continuous functions are continuous. Define

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

Since continuous functions are bounded on compact sets, we don't have to worry about integrability. We can always bound $|\langle f,\ g\rangle| \leq M_f M_g |b-a| < \infty$.
The norm is

$$\|f\| = \sqrt{\langle f,\ f\rangle} = \sqrt{\int_a^b f(x)^2\,\mathrm{d}x}.$$

Example 9.10. Another inner product:

$$\langle f,\ g\rangle_w = \int_a^b w(x)f(x)g(x)\,\mathrm{d}x,$$

where $w(x) \geq 0$ on (a, b). We typically call w a *weight function.* This gives a different norm as well.

$$\|f\|_w = \sqrt{\int_a^b w(x)|f(x)|^2\,\mathrm{d}x}$$

A general p-norm for $1 \leq p < \infty$ is

$$\|f\|_p = \left(\int_a^b |f(x)|^p\,\mathrm{d}x\right)^{1/p}.$$

The infinity norm is

$$\|f\|_\infty = \max_{x\in[a,b]} |f(x)|.$$

If $p \neq 2$ is also a norm, but is not associated with an inner product. Only 2-norm or weighted 2-norm have an inner product structure, which let's us talk about orthogonality.

10 Best Approximation in 2-norm (9/21/23)

Remark 10.1. Note that for a vector (linear) space, the $\mathbf{0}$ is always in it. This can be seen by either letting a scalar multiple be 0 or adding f with $-f$.

Let $C^n([a, b])$ be the the set of n-th order continuously differentiable on $[a, b]$. Let $L^p([a, b])$ we the space of p-integrable functions over $[a, b]$. From last time, we defined an inner product. Using the inner product of an element of the space with itself, gives us an induced norm.

Definition 10.2. A norm is a map $\|\cdot\| : V \to \mathbb{R}([0,\infty)$ that satisfies

1. $\|f+g\| \leq \|f\| + \|g\|$,
2. $\forall \lambda \in \mathbb{R}$, $\|\lambda f\| = |\lambda|\|f\|$,
3. $\|f\| \geq 0$, where $\|f\| = 0 \iff f = 0$.

If $f = g - h$, then the norm gives us a metric.

Last time we saw the norms $\|f\|_2, \|f\|_{2,w}$ are induced by an inner product, but $\|f\|_p,\ \|f\|_\infty$ are not induced a norm. This is because we need the even parity to get back to an inner product.

Now the question is, which norm do we choose? Recall from linear algebra, we preferred the canonical basis because of its orthogonality. In this same vein, we prefer inner product spaces because of *orthogonality.*

Example 10.3. Consider the interval $[-\pi, \pi]$. Consider a sequence of functions in this space.

$$p_0 = 1, p_1 = \sin(x), p_2 = \cos(x), \ldots, p_{2k-1}(x) = \sin(kx), p_{2k}(x) = \cos(kx),\ \forall k \in \mathbb{N}.$$

Here, there are infinitely many elements. Then

$$\langle p_i(x),\ p_j(x)\rangle = \int_{-\pi}^{\pi} p_i(x)p_j(x)\,\mathrm{d}x = \begin{cases} \|p_i(x)\|_2^2 = 2\pi, & i = j, \\ 0, & i \neq j \end{cases}$$

Example 10.4. *NN in ML.* Consider a neural network as $NN_\theta : x \to y$. The training process is given $\{(x_i, y_i)\}$ for $i = 1, \ldots, N$. We consider the outputs to be some underlying function $y_i = G(x_i)$. We want to find a neural network to approximate G $(NN_\theta(x) \approx G(x))$. The problem is

$$\min_\theta \|NN_\theta(x) - G(x)\|.$$

If we choose the weighted 2-norm, then the problem is

$$\min_\theta \int_\Omega |NN_\theta(x) - G(x)|^2 w(x)\,\mathrm{d}x \approx \min_\theta \sum_{i=1}^N |NN_\theta(x_i) - y_i|^2 w_i.$$

To solve on the computer, we have to discretize the continuous function, where given our node $\{x_i\}$, we have weights $\{w_i\}$. We choose 2 norm because we can take gradients of this quite easily. Consider taking the gradient of $|NN_\theta(x_1) - y(1)|^p$. The gradient is $p|NN_\theta(x_1) - y_1|^{p-1}\nabla_\theta|NN_\theta(x_1) - y_1|$. The problem is that the absolute value is not differentiable everywhere. But if $p = 2$, we can get rid of the absolute value and not worry about differentiating the absolute value.

To summarize, we choose 2-norm for orthogonality and easier gradients.

Question. Given a function $f(x) \in C([a,b])$ and $\in L_w^2([a,b])$. Can we find a polynomial of degree n such that

$$\|f - p_n(x)\|_2 = \inf_{q \in \mathbb{P}_n} \|f - q\|_2.$$

This p is the best approximation in the 2-norm. Assume existence. How can we find it? We always have $p_n(x) = c_0 + c_1 x + \cdots + c_n x^n$. We can then have a coefficient vector $\in \mathbb{R}^{n+1}$. Now our minimization problem is over the possible $c \in \mathbb{R}^{n+1}$

11 Orthogonal polynomials (9/26/23)

Basis. Recall from linear algevra, we could use the canonical basis to say

$$\mathbb{R}^3 = \operatorname{span}\left\{ \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} \right\}.$$

We can say the same for polynomials:

$$\mathcal{P}_n = \operatorname{span}\{x^0, x^1, x^2, \dots, x^2\}$$

is called the monomial basis. However, recall that a basis in *not* unique. So for polynomials, we can also consider another general basis as $\{\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)\}$. Necessary conditions are linearly independent and span the set. A linearly independent list that is the length of the dimension of the space will span, so we can just say we care about linear independence. So for a basis,

$$\gamma_0\varphi_0(x) + \cdots + \gamma_n\varphi_n(x) = 0 \implies \gamma_0 = \cdots = \gamma_n = 0.$$

Definition 11.1 (Orthogonal polynomials)**.** We say $\varphi_0, \varphi_1, \dots, \varphi_n, \dots$ is a sequence (system) of orthogonal polynomials defined on (a, b) with respect a weight function $w(x) > 0$ on (a, b) if

$$\langle \varphi_i(x),\ \varphi_j(x)\rangle_w = \int_a^b \varphi_i(x)\varphi_j(x)\,\mathrm{d}x = \begin{cases} 0, & i \neq j \\ \neq 0, & i = j \end{cases}.$$

For orthogonal polynomials, $\varphi_j(x)$ has to have degree j. Orthonormality is when it equals δ_{ij}.

Definition 11.2. If $\{\varphi_0(x), \dots, \varphi_n(x)\}$ are mutually orthogonal with respect to $w(x) > 0$ on (a, b) with degree $\leq n$, then $\{\varphi_0(x), \dots, \varphi_n(x)\}$ is an *orthogonal basis* for $\mathcal{P}_n(a, b)$.

Proof. WTS $\sum_{i=0}^n \gamma_i\varphi_i - 0 \iff \gamma_i = 0\ \forall i$. $\Longleftarrow$ is trivial. Consider $\Longrightarrow$. Take any φ_k from the basis,

$$\left\langle \sum_{i=0}^n \gamma_i\varphi_i,\ \varphi_k \right\rangle_w = \sum_{i=0}^n \gamma_i \langle \varphi_i,\ \varphi_k\rangle_w = \gamma_k \langle \varphi_k,\ \varphi_k\rangle.$$

Since we know $\langle \varphi_k,\ \varphi_k\rangle \neq 0$, then we know $\gamma_k = 0$. □

Now we go back to our problem for $f \in C([a, b])$:

$$\min_{p\in\mathcal{P}_n} \|f - p_n\|_{2,w}^2.$$

We can rewrite the square of the norm as

$$\begin{aligned}\langle f - p_n,\ f - p_n\rangle_w &= \langle f - p_n,\ f\rangle_w - \langle f - p_n,\ p_n\rangle_w \\ &= \langle f,\ f\rangle_w - \langle p_n,\ f\rangle_w - \langle f,\ p_n\rangle_w + \langle p_n,\ p_n\rangle_w \\ &= \underbrace{\langle f,\ f\rangle_w}_{\text{no role in minimizing } p_n} -2\langle p_n,\ f\rangle_w + \langle p_n,\ p_n\rangle_w.\end{aligned}$$

The problem now is

$$\min_{p\in\mathcal{P}_n} -2\langle p,\ f\rangle_w + \langle p,\ p\rangle_w.$$

If we choose some arbitrary basis, it is written as

$$\min_{\gamma_0,\ldots,\gamma_n\in\mathbb{R}} -2\left\langle \sum_{i=0}^{n}\gamma_i\varphi_i,\ f\right\rangle + \left\langle \sum_{j=0}^{n}\gamma_j\varphi_j,\ \sum_{i=0}^{n}\gamma_i\varphi_i\right\rangle = -2\sum_{i=0}^{n}\gamma_i\left\langle\varphi_i,\ f\right\rangle + \sum_{i,j=0}^{n}\gamma_i\gamma_j\left\langle\varphi_i,\ \varphi_j\right\rangle.$$

Define

$$b = \begin{bmatrix}\langle\varphi_0,\ f\rangle\\ \langle\varphi_1,\ f\rangle\\ \vdots\\ \langle\varphi_n,\ f\rangle\end{bmatrix}\in\mathbb{R}^{n+1},\quad \gamma = \begin{bmatrix}\gamma_0\\ \gamma_1\\ \vdots\\ \gamma_n\end{bmatrix}\in\mathbb{R}^{n+1}$$

and $M\in\mathbb{R}^{(n+1)\times(n+1)}$, where $M_{ij} = \langle\varphi_{i-1},\ \varphi_{j-1}\rangle$ as

$$\min_{\gamma} -2\gamma^T b + \gamma^T M\gamma = E(\gamma).$$

(Recall $a\cdot b = a^T b$.) We know

1. $E(\gamma)$ is quadratic.
2. M is symmetric positive definite.
3. We want to minimize $E(\gamma)$.

Let γ^* be the minimizer. Then $\nabla E(\gamma^*) = 0$.

$$\nabla E(\gamma^*) = -2b + 2M\gamma^* = 0.$$

Then the solution is $\gamma^* = M^{-1}b$. Inverting a matrix is $O(n^3)$ flops, so it is not ideal. But if our M is with respect to an orthogonal basis, then M will be diagonal, which is nice since inverting a diagonal matrix is $O(n)$. Going back to the sum expression

$$\begin{aligned}-2\sum_{i=0}^{n}\gamma_i\langle\varphi_i,\ f\rangle + \sum_{i=0}^{n}\gamma_i^2\langle\varphi_i,\ \varphi_j\rangle &= \sum_{i=0}^{n}\left(\gamma_i^2\langle\varphi_i,\ \varphi_i\rangle - 2\gamma_i\langle\varphi_i,\ f\rangle\right)\\ &= \sum_{i=0}^{n}\langle\varphi_i,\ \varphi_i\rangle\left(\left(\gamma_i - \frac{\langle\varphi_i,\ f\rangle}{\langle\varphi_i,\ \varphi_i\rangle}\right)^2 - \left|\frac{\langle\varphi_i,\ f\rangle}{\langle\varphi_i,\ \varphi_i\rangle}\right|^2\right)\end{aligned}$$

Since the coefficient is positive, and we can't really control the abs term, the solution becomes

$$\gamma_i^* = \frac{\langle\varphi_i,\ f\rangle}{\langle\varphi_i,\ \varphi_i\rangle}.$$

We claim that $0 = \langle f - p_n,\ \varphi_i\rangle\ \forall i$.

12 Construction of Gauss rules (9/28/23)

12.1 Orthogonal Polynomials Continued

Recap. We want to solve for

$$\min_{p\in\mathcal{P}_n}\|f-p\|_{2,w}$$

for f continuous on $[a, b]$, the solution of which is the best approximation in the 2-norm. The basis is $\operatorname{span}\{\varphi_0, \dots, \varphi_n\}$ is an orthogonal basis and any $p = \gamma_0\varphi_0 + \cdots + \gamma_n\varphi_n$. We the optimal result is

$$\gamma_j^* = \frac{\langle f,\ \varphi_j\rangle_w}{\langle \varphi_j,\ \varphi_j\rangle_w}.$$

So then

$$p^* = \sum_{j=0}^{n} \gamma_j^* \varphi_j.$$

We claim that $\langle f - p^*,\ \varphi_j\rangle = 0\ \forall j$. This means the residual $f - p^* \perp \mathcal{P}_n$.

$$\begin{aligned}
\langle f - p^*,\ \varphi_j\rangle &= \langle f,\ \varphi_j\rangle - \langle p^*,\ \varphi_j\rangle\,,\\
&= \langle f,\ \varphi_j\rangle - \left\langle \sum_{i=0}^{n} \gamma_i^*\varphi_i,\ \varphi_j \right\rangle,\\
&= \langle f,\ \varphi_j\rangle - \sum_{i=0}^{n} \gamma_i^* \langle \varphi_i,\ \varphi_j\rangle\,,\\
&= \langle f,\ \varphi_j\rangle - \varphi_j^* \langle \varphi_j,\ \varphi_j\rangle\,,\\
&= \langle f,\ \varphi_j\rangle - \frac{\langle f,\ \varphi_j\rangle}{\langle \varphi_j,\ \varphi_j\rangle} \langle \varphi_j,\ \varphi_j\rangle\,,\\
&= 0.
\end{aligned}$$

This then implies that

$$\|f - p^*\|_{2,w}^2 + \|p^*\|_{2,w}^2 = \|f\|_{2,w}^2\,.$$

We call p^* the orthogonal projection. It is unique.

12.2 Numerical Integration (revisited)

Using what we have learned, can we design a new algorithm for

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

a quadrature rule of $(n+1)$ nodes $\{x_i\}$ with $(n+1)$ weights $\{w_i\}$? We approximate the integrand $f(x) \approx p(x)$. We want our quadrature rule such that it is exact on our polynomial and p interpolates f at x_i:

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

Recall **Newton-Cotes:** $x_i = a + i\frac{b-a}{n}$ for $i = 0, 1, 2, \dots, n$ with weights $w_i = \int_a^b L_i(x)\,\mathrm{d}x$. Generally, Newton-Cotes with $(n+1)$ nodes is exact for $\mathcal{P}_n$.

Gauss quadrature with $(n+1)$ nodes is exact for $\mathcal{P}_{2n+1}$. We get this higher degree of accuracy by being more clever with the selection of our nodes.

Orthogonal polynomials. We say $\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_n, \dots$ forms a system of orthogonal polynomials with respect to inner product with weight $w(x) > 0$ on (a, b) if

1. The degree of φ_j is j.

2. $\langle \varphi_i,\ \varphi_j \rangle = \begin{cases} 0, & \text{if } i \neq j \\ > 0, & \text{if } i = j \end{cases}$

3. $\forall n$, $\varphi_{n+1} \perp \{\varphi_0, \varphi_1, \dots, \varphi_n\}$.

4. $\{\varphi_0, \dots, \varphi_n\}$ forms an orthogonal basis for $\mathcal{P}_n$.

5. $\varphi_{n+1} \perp \mathcal{P}_n$ $\forall n$.

6.

Theorem 12.1. *φ_j has distinct, real j roots that lie in (a, b).*

Theorem 12.2 (Polynomial Remainder Theorem)**.** *$p(x)$ divident. $g(x)$ divisor.*

$$p(x) = g(x)q(x) + r(x),$$

where degree$(r) <$ degree(g).

$\forall p \in \mathcal{P}_{2n+1}$. Let $w(x) = 1$. Pick φ_{n+1} as divisor with p as divident. Using the polynomial remainder theorem,

$$p = \varphi_{n+1}q + r.$$

We know degree$(p) \leq 2n+1$, degree$(r) \leq n$, and degree$(q) \leq n$. Now try to integrate p:

$$\begin{aligned} \int_a^b p(x)\,\mathrm{d}x &= \int_a^b \varphi_{n+1}q + r\,\mathrm{d}x, \\ &= \int_a^b \varphi_{n+1}q\,\mathrm{d}x + \int_a^b r\,\mathrm{d}x, \\ &= \underbrace{\langle \varphi_{n+1},\ q \rangle}_{=0,\ \varphi_{n+1} \perp \mathcal{P}_n} + \int_a^b r\,\mathrm{d}x \\ &= \int_a^b r\,\mathrm{d}x. \end{aligned}$$

Since $r(x)$ is at most degree n, then we can use $n+1$ nodes to integrate it exactly by interpolating $r(x)$. Recall Lagrange interpolation with $(n+1)$ nodes is exact for polynomials of degree $\leq n$. Then, we can replace r with its Lagrange interpolation

$$r(x) = \sum_{k=0}^{n} r(x_k)L_k(x), \quad L_k(x) = \prod_{i=0,\ i\neq k}^{n} \frac{(x - x_i)}{(x_k - x_i)}.$$

Then the original problem is

$$\begin{aligned} \int_a^b p(x)\,\mathrm{d}x &= \int_a^b \sum_{k=0}^{n} r(x_k)L_k(x)\,\mathrm{d}x, \\ &= \sum_{k=0}^{n} r(x_k) \underbrace{\int_a^b L_k(x)\,\mathrm{d}x}_{w_k}, \\ &= \sum_{k=0}^{n} w_k r(x_k). \end{aligned}$$

Only given a $p(x)$, do we want to go through polynomial remainder to find $r(x)$? Let's get in the form of $p(x)$ instead of $r(x)$. Try to cleverly choose nodes such that $r(x_k) = p(x_k)\ \forall k = 0, 1, \ldots, n$.

$$p(x_k) = \varphi_{n+1}(x_k)q(x_k) + r(x_k).$$

Can I find $\{x_0, x_1, \ldots, x_n\}$ such that $\varphi_{n+1} = 0,\ \forall k = 0, 1, 2 \ldots, n$, which is asking if I can find $(n+1)$ distinct roots for the polynomial $\varphi_{n+1}(x)$. Answer is yes using the theorem (12.1) stated earlier as $\varphi_{n+1}(x)$ has $n+1$ distinct roots on (a, b).

The Gauss quadrature rule selects

- $\{x_i\},\ i = 0, 1, \ldots, n$ to be the roots of $\varphi_{n+1}(x)$ orthogonal polynomial.
- $\{w_i\},\ i = 0, 1, \ldots, n$ for

$$\int_a^b p(x)\,\mathrm{d}x = \sum_{i=0}^n w_i p(x_i)\ \forall p \in \mathcal{P}_{2n+1}.$$

Gauss quadrature rule is not unique, depends on (a, b) and the weight. If we had a different weight,

$$\int_a^b p(x)w(x)\,\mathrm{d}x = \sum_{i=0}^n w_i p(x_i)w(x_i)\ \forall p \in \mathcal{P}_{2n+1}$$

where $\varphi_{n+1}(x)$ is orthogonal with respect to $w(x)$.

13 Error estimation for Gauss and composite Gauss (10/3/23)

Continuing from last time, we want to write our quadrature rule with evaluations of f, not p. Assume p interpolates f (i.e. $p(x_i) = f(x_i),\ i = 0, 1, 2, \ldots, n$).

Remark 13.1 (Interpolation recap). In Lagrange interpolation, we are given $\{(x_i, y_i)\},\ i = 0, 1, \ldots, n$. We can construct $p \in \mathcal{P}_n$, with $p(x_i) = y_i$ in the form $P(x) = \sum_{k=0}^n L_k(x)Y_k$ with $L_k(x) = \prod_{i=0,\ i\neq k}^n \frac{x - x_i}{x_k - x_i} = \delta_{ik}$.

Consider the case where we also know $f'(x_i)$ for $i = 1, 2, \ldots, n$. So we have $2n+2$ points of data, which can uniquely find a polynomial interpolant $p \in \mathcal{P}_{2n+1}$.

$$p(x) = \sum_{k=0}^n \left(H_k(x)f(x_k) + K_k(x)f'(x_k)\right),$$

with

$$H_k(x) = (L_k(x))^2(1 - 2L_k'(x_k)(x - x_k)) \in \mathcal{P}_{2n+1}, \quad K_k(x) = (L_k(x))^2(x - x_k) \in \mathcal{P}_{2n+1}.$$

We have the properties $H_k(x_i) = \delta_{ik}$, $H_k'(x_i) = 0\ \forall i$, $K_k(x_i) = 0\ \forall i$, and $K_k'(x_i) = \delta_{ki}$. Given these condition, $p_H(x_k) = f(x_k)$ and $p_H'(x_k) = f'(x_k),\ k = 0, 1, \ldots, n$.

Going back to the quadrature problem,

$$\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b p_H(x)\,\mathrm{d}x,& \\
&= \int_a^b \sum_{k=0}^n (H_k(x)f(x_k) + K_k(x)f'(x_k))\,\mathrm{d}x, \\
&= \sum_{k=0}^n f(x_k) \underbrace{\int_a^b H_k(x)\,\mathrm{d}x}_{w_k} + \sum_{k=0}^n f'(x_k) \underbrace{\int_a^b K_k(x)\,\mathrm{d}x}_{v_k}.
\end{aligned}$$

Now,

$$\begin{aligned} v_k &= \int_a^b (L_k(x))^2 (x - x_k)\,\mathrm{d}x, \\ &= \frac{1}{\prod_{i=0,\ i\neq k}^n (x_k - x_i)} \int_a^b \underbrace{(x-x_0)\cdots(x-x_k)\cdots(x-x_n)}_{\in\mathcal{P}_{n+1},\ \pi_{n+1}(x)} \underbrace{L_k(x)}_{\in\mathcal{P}_n}\,\mathrm{d}x. \end{aligned}$$

Consider $\varphi_{n+1} \perp \mathcal{P}$. $\varphi_{n+1}(x) = c(x-x_0)\cdots(x-x_n) = c\pi_{n+1}$ if $\{x_i\}$ are roots of φ_{n+1}. Then $\langle \varphi_{n+1},\ L_k\rangle = 0,\ \langle \pi_{n+1},\ L_k\rangle = 0$. Thus, $v_k = 0$.

Consider the other term now

$$\begin{aligned} w_k &= \int_a^b (L_k(x))^2 (1 - 2L_k'(x_k)(x-x_k))\,\mathrm{d}x \\ &= \int_a^b (L_k(x))^2\,\mathrm{d}x - 2L_k'(x_k) \underbrace{\int_a^b (L_k(x))^2 (x-x_k)\,\mathrm{d}x}_{=v_k=0} \\ &= \int_a^b (L_k(x))^2\,\mathrm{d}x. \end{aligned}$$

Remark 13.2. From last lecture of Gauss-Quadrature, $w_k = \int_a^b L_k(x)\,\mathrm{d}x$. That means $\int_a^b L_k(x)\,\mathrm{d}x = \int_a^b (L_k(x))^2\,\mathrm{d}x$ since the weights were both roots of the orthogonal polynomial.

Proof. We know Gauss-Quadrature is exact for any $2n+1$ polynomial. Each integrand is n and $2n$ degree, respectively, then

$$\begin{aligned} \int_a^b (L_k(x))^2\,\mathrm{d}x &= \sum_{i=0^n} (\underbrace{L_k(x_i)}_{\delta_{ik}})^2 w_i \\ &= w_k = \int_a^b L_k(x)\,\mathrm{d}x. \end{aligned}$$

□

This tells us that $w_k > 0$, so Gauss quadrature always has positive weights. This is a reason why it does not suffer from the Runge Phenomenon.

Here, we have to select nodes that are the roots of φ_{n+1}, which can be hard. We can do bisection, Newton's method, secant method, or other fixed point methods. Say ξ is one of the roots of φ_{n+1}. Then let $\psi(x) = \varphi_{n+1}(x)/(x-\xi) = 0 \in \mathcal{P}_n$. Continue this until all roots are found. If $n \to \infty$, then the roots cluster. I.e. consider a polynomial in $[-1, 1]$. So, it is bad for large n. (What about scaling domain?)

Usually, we use composite Gauss quadrature, where we apply Gauss quadrature to $[x_{i-1}, x_i]$.

13.1 Differential Equations

Consider

$$\det(D^2 u) = \frac{f(x)}{g(\nabla u(x))}.$$

Looks terrible to solve analytically. Or consider

$$\nabla \cdot (\sigma(x)\nabla u) = 1,$$

given $\sigma(x),\ x \in \Omega,\ u|_{\partial\omega} = 0$. Differential equations are an *implicit* way of writing down an equation. The goal is to find the *explicit* expression of the solution—at least numerically. What space to find solutions? We can use $\mathcal{P}_n$ to say $u \approx \sum_{i=0}^{n} x_i \varphi_i(x) \in \mathcal{P}_n$. These are called *spectral methods*—global polynmoial representation. Consider $\forall\varphi,\ \varphi = 0$ on $\partial\Omega$,

$$\int_\Omega \varphi \nabla \cdot (\sigma(x)\nabla u)\,\mathrm{d}x = \int_\Omega \nabla\varphi \cdot \nabla u \sigma(x)\,\mathrm{d}x.$$

If we write $\nabla\varphi$ and ∇u as a combination of orthogonal polynomials, then we can use orthogonality and such.

14 Review (10/5/23)

This lecture is a review of material covered so far, as they will be our tools to solve differential equations. I will not write down all things covered in this lecture if it is not new material.

Order of convergence.

$$\lim_{n\to\infty} \frac{|x_{n+1} - \xi|}{|x_n - \xi|^1} = \mu.$$

Suppose $q = 1$. If $\mu = 0$, then we have super-linear convergence. If $\mu = 1$, then it is sublinear convergence.

Example 14.1. Let

$$g(x) = \begin{cases} 2^{-(1+(\log_2(1/x))^{1/\alpha})^\alpha}, & x \in (0,1] \\ 0, & x = 0 \end{cases}.$$

$\xi = 0$ is a fixed point of $g(x)$. Then $x_k = 2^{-k^\alpha} \to 0$ as $k \to \infty$ and $\alpha > 0$.

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|} = \begin{cases} 1, & \alpha \in (0,1) \text{ sublinear} \\ \frac{1}{2}, & \alpha = 1 \text{ linear} \\ 0, & \alpha > 1 \text{ superlinear} \end{cases}$$

Orthogonal polynomials.

Definition 14.2 (Gram-Schmidt orthogonalization)**.** Given a linearly independent basis, make it orthogonal and linearly independent.

Example 14.3. Given $1, x, x^2, \ldots$. We have $\langle \cdot,\ \cdot \rangle$. Let $\varphi_0 = 1$. Consider $P_0 = \operatorname{span}\{\varphi_0\}$. Find the orthogonal component of x by subtracting the best 2-norm approximation of x in P_0. So, $\varphi_1 = x - \frac{\langle x,\ \varphi_0\rangle}{\langle \varphi_0,\ \varphi_0\rangle}\varphi_0 \implies \varphi_1 \perp \varphi_0$.

15 Numerical Solutions to ODEs (10/23/23)

Definition 15.1. An m-th order ODE has the form

$$y^{(m)} = f(t, y, y', \ldots, y^{(m-1)})$$

with $y(t_0) = \alpha_0,\ y'(t_0) = \alpha_1, \ldots, y^{(m-1)}(t_0) = \alpha_{m-1},\ t \in [t_0, T]$ and $y \in \mathbb{R}$.

We can transform it into a 1st-order ODE system. Let

$$y_0 = y, \quad y_1 = y', \ldots, y_{m-1} = y^{(m-1)}.$$

15.1 Initial Value Problems (IVP)

Consider problems in the form

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = f(t,x), \\ x(t=t_0) = x_0. \end{cases}$$

In in introductory course, recall we could plot some vector field in the x-t plane to represent $f(t,x)$. For any starting points (t_0, x_0), we can follow the flow of the vector field to obtain some curve, which is the solution $x(t)$. But if there is another initial condition $(t_0, \bar{x}_0)$, the curve will follow the vector field again and give us a different solution $\bar{x}(t)$. We can see the first line only determines the vector fields, and the second line specifies the unique solution.

There are certain conditions on $f(t,x)$ to make IVP a well-posed problem. We enforce f to be Lipschitz

Definition 15.2 (Lipschitz condition)**.** $\forall x, y$ in the domian of search

$$|f(t,x) - f(t,y)| < L|x-y|, \quad 0 < L < \infty.$$

Any function that is Lipschitz is also continuous.

Definition 15.3 (Domain of search)**.** Let $x \in \mathbb{R}$. The *domain of serach* $D = [t_0, T] \times [a,b]$. $x(t)$ as a curve in $\mathbb{R}$, we want every pair $(t, x(t)) \in D$.

So, we only want Lipschitz inside this domain of search. Many problems are not globally Lipschitz but locally Lipschitz. From now on, we will assume f is Lipschitz.

Numerical Representation. $x(t) \to$ discretization. Given an interval $[t_0, T]$. Divide it into n equal intervals. Now we have $t_0, t_1, \ldots, T = t_N$ with length $h = \frac{T-t_0}{N}$. Now we represent the solution as $x(t_0), x(t_1), \ldots, x(t_N)$. The question is *can I approximate* $\{x(t_i)\}_{i=0}^{N}$ *as accurate as possible* (in value-space)? We call $y_0 \approx x(t_0), y_1 \approx x(t_1), \ldots, y_N \approx x(t_N)$.

We can turn the differential equation into an integral equation by integrating both sides. Integrate over some interval

$$\begin{aligned} \int_{t_i}^{t_{i+1}} \frac{\mathrm{d}x}{\mathrm{d}t}\,\mathrm{d}t &= \int_{t_i}^{t_{i+1}} f(t,x)\,\mathrm{d}t, \\ x(t_{i+1}) - x(t_i) &= \int_{t_i}^{t_{i+1}} f(t,x)\,\mathrm{d}t \quad \text{FTC} \\ y_{i+1} - y_i &= f(t_i, y_i)h \quad \text{Explicit/Forward Euler} \\ &= f(t_{i+1}, y_{i+1})h \quad \text{Implicit/Backward Euler} \\ &= \frac{1}{2}\left(f(t_i, y_i + f(t_{i+1}, y_{i+1})\right)h \quad \text{Trapezoidal Rule} \end{aligned}$$

- For forward Euler,
$$y_{i+1} = y_i + hf(t_i, y_i)$$
we have an explicit evaluation of f.

- For backward Euler,
$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1})$$
we have an implicit function to solve for y_{i+1}. This becomes a root-finding problem, where we will often use Newton's method.

- For Trapezoidal Rule,
$$y_{i+1} = y_i + \frac{h}{2}(f(t_{i+1}, y_{i+1}) + f(t_i, y_i)).$$

Forward Euler is cheap to implement but has stability issues. Implicit method seems annoying to solve, but it is unconditionally stable.

Now we talk about error.

- *Global Error*: $e_n = x(t_n) - y_n$.
- *Local Truncation Error (LTE)*: For example for trapezoidal, the discrete solution satisfies $\frac{y_{n+1}-y_n}{2} = f(t_n, y_n) = \Phi(t,u)|_{t=t_n,u=y_n}$, but the true solution $x(t_n), x(t_{n+1})$ do not satisfy it.
$$L_n = \frac{x(t_{n+1}) - x(t_n)}{h} - \Phi(t_n, x(t_n)).$$

Error accumulates over time, and global error will tell us about this—the worst error will occur at the final time. Local error is more about error at each step.

Assume that Φ is Lipschitz,
$$|\Phi(t,u) - \Phi(t,v)| < L_\Phi |u - v|.$$

1. Based on LTE, L_n,
$$x(t_{n+1}) = x(t_n) + h\Phi(t_n, x(t_n)) + hL_n.$$
2. Our scheme:
$$y_{n+1} = y_n + h\Phi(t_n, y_n).$$

Subtract the 2 equations gives us
$$\begin{aligned}
e_{n+1} &= e_n + h(\Phi(t_n, x(t_n)) - \Phi(t_n, y_n)) + hL_n, \\
|e_{n+1}| &\le |e_n| + h|\Phi(t_n, x(t_n)) - |\Phi(t_n, y_n)| + h|L_n|, \\
&\le |e_n| + hL_\Phi |e_n| + h|L_n|, \\
&= (1 + hL_\Phi)|e_n| + h|L_n|.
\end{aligned}$$

We can see the structure of new error related to past error and new error at step. Now we can induct all the way down.
$$\begin{aligned}
|e_n| &\le (1 + hL_\Phi)|e_{n-1}| + h|L_{n-1}|, \\
|e_{n+1}| &\le (1 + hL_\Phi)^2|e_{n-1}| + h(1 + hL_\Phi)|L_{n-1}| + h|L_n|, \\
&\vdots \\
&\le \underbrace{(1 + hL_\Phi)^{n+1}|e_0|}_{=0 \text{ bc IVP}} + \underbrace{\sum_{j=0}^{n}(1 + hL_\Phi)^j}_{\frac{(1+hL_\Phi)^{n+1}-1}{1+hL_\Phi-1}}(hL_{\max}), \quad L_{\max} = \max_{0\le j\le N} |L_j| \\
&\le \frac{L_{\max}}{L_\Phi}((1 + hL_\Phi)^{n+1} - 1), \\
&\le \frac{L_{\max}}{L_\Phi}(e^{hL_\Phi(n+1)} - 1), \\
&\le \frac{L_{\max}}{L_\Phi}(e^{L_\Phi(T-t_0)} - 1).
\end{aligned}$$

We know $1 + hL_\Phi \leq e^{hL_\Phi}$ from the linearization/Taylor series of exp. We can control $L_{\max}$ to control this bound on the global error.

16 (10/17/23)

Local truncation errors for the methods we described last time are

$$\begin{array}{cc} \text{Forward} & \frac{1}{h}(x(t_{n+1}) - x(t_n) - hf(t_n, x(t_n)) \\ \text{Backward} & \frac{1}{h}(x(t_{n+1}) - x(t_n) - hf(t_{n+1}, x(t_{n+1})) \\ \text{Trapezoidal} & \frac{1}{h}\left(x(t_{n+1} - x(t_n) - \frac{h}{2}f(t_n, x(t_n)) - \frac{h}{2}f(t_{n+1}, x(t_{n+1})\right) \end{array}.$$

Remember

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(t, x(t)) = x'(t),$$

so these can be subbed into the expressions above. Then, use Taylor series with remainder on $x(t)$. Let L_n denote the LTE.

- For Forward Euler,

$$x(t_{n+1}) = x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(\xi),\ t_n < \xi < t_{n+1}$$
$$\implies L_n = \frac{1}{h}\left(\frac{h^2}{2}x''(\xi)\right) = \frac{h}{2}x''(\xi).$$

  We have $L_{\max} \leq \frac{h}{2}M_2$ for $M_w = \max_{t\in[t_0,T]} |x''(t)|$ if $x(t) \in C^2([t_0, T])$. Then the error is $O(h)$, first order.

- For Backward Euler, use $t_n = t_{n+1} - h$

$$x(t_n) = x(t_{n+1}) - hx'(t_{n+1}) + \frac{h^2}{2}x''(\eta),\ \eta \in (t_n, t_{n+1}),$$
$$\implies L_n = \frac{1}{h}\left(-\frac{h^2}{2}x''(\eta)\right) = -\frac{h}{2}x''(\eta).$$

  Assume $L_{\max} \leq \frac{h}{2}M_2 = O(h)$. Then $e_n = O(h)$.

- For Trapezoidal, take one more term in Taylor

$$\begin{aligned} x(t_{n+1}) &= x(t_n) + hx'(t_{n+1}) + \frac{h^2}{2}x''(t_N) + \frac{h^3}{6}x'''(\tau),\ \tau \in (t_n, t_{n+1}) \\ x'(t_{n+1}) &= x'(t_n) + x''(t_n)h + \frac{h^2}{2}x''(\rho),\ \rho \in (t_n, t_{n+1}) \\ hL_n &= \frac{h^2}{2}x''(t_n) + \frac{h^3}{6}x''(\tau) - \frac{h}{2}\left(x''(t_n)h + \frac{h^2}{2}x'''(\rho)\right) \\ &= \frac{h^3}{6}x'''(\tau) - \frac{h^3}{4}x'''(\rho) \\ \implies L_{max} &\leq \frac{5h^2}{12}M_3, \quad M_3 = \max_{t\in[t_0,T]} |x''(t)|. \end{aligned}$$

  Thus, the local truncation error is $O(h^2)$. This comes from the cancellation of the second order terms. With more work, this can also be shown for the global error as well. So, second order accuracy for trapezoidal rule.

Reverse thinking: *if we wanted to create a higher order method, write out Taylor expansions and find linear combinations that cancel the lower order terms.*

16.1 Runge-Kutta methods

Recall the problem we have is

$$x(t_{n+1}) - x(t_n) = \int_{t_n}^{t_{n+1}} f(t, x(t))\,\mathrm{d}t,$$

and we are finding ways to approximate the RHS. For Runge-Kutta, a 2-stage method is

$$y_{n+1} - y_n = h(ak_1 + bk_2); \quad k_1 = f(t_n, y_n), \quad k_2 = f(t_n + \alpha h, y_n + \beta h k_1),\ 0 \le \alpha \le 1.$$

For higher values of i for k_i, there is a recursive definition that follows from above. The local truncation error for this 2-stage Runge-Kutta method is

$$\frac{1}{h}\left(x(t_{n+1}) - x(t_n) - ahf(t_n, x(t_n)) - bhf(t_n + \alpha h, x(t_n) + \beta h f(t_n, x(t_n)))\right).$$

Replace $f(t_n, x(t_n)) = x'(t_n)$.

$$\frac{1}{h}\left(x(t_{n+1}) - x(t_n) - ahx'(t_n) - bhf(t_n + \alpha h, x(t_n) + \beta h x'(t_n)\right).$$

We know

1. $x''(t_n) = \left.\dfrac{\mathrm{d}f(t, x(t))}{\mathrm{d}t}\right|_{t_n} = f_t + f_x \dfrac{\mathrm{d}x}{\mathrm{d}t} = f_t + f_x f|_{t_n}$.
2. $x'(t_n) = f|_{t_n}$

Then (for any f or derivatives, it is evaluated at (t_n, xt_n).)

$$\begin{aligned}
h(LTE) &= hx'(t_n) + \frac{h^2}{2}x''(t_n) + \frac{h^3}{6}x'''(\tau) - ahx'(t-n) \\
&\quad - bh\left(f + \alpha h f_t + \beta h f_x f + \frac{1}{2}(\alpha h)^2 f_{tt} + \frac{1}{2}(\beta h)^2 f_{xx} + (\alpha h)(\beta h) f_{xx} + O(h^3)\right) \\
&= hf + \frac{h^2}{2}(f_t + f_x f) + O(h^3) - ahf \\
&\quad - bh\left(f + \alpha h f_t + \beta h f_x f + \frac{1}{2}(\alpha h)^2 f_{tt} + \frac{1}{2}(\beta h)^2 f_{xx} + (\alpha h)(\beta h) f_{xx} + O(h^3)\right).
\end{aligned}$$

We want all the $O(h)$ terms and $O(h^2)$ cancel:

$$1 - a - b = 0,$$

$$h^2\left(\frac{1}{2}f_t + \frac{1}{2}f_x f - b\alpha f_t - \beta b f_x f\right) = 0 \implies \alpha b = \frac{1}{2}, \quad \beta b = \frac{1}{2}$$

Then LTE is $O(h^2)$. 2-stage Runge-Kutta returns the Trapezoidal rule when $\alpha = 1,\ b = \frac{1}{2},\ \beta = -1$.

17 (10/19/23)

17.1 Runge-Kutta Continued

A more commonly used form is analogous to Simpson's rule (4-stage).

$$\begin{aligned}
x(t_{n+1}) - x(t_n) &= \int_{t_n}^{t_{n+1}} f(t, x(t))\,\mathrm{d}t, \\
&\approx \frac{1}{6}\left(f(t_n, x(t_n)) + 4f\left(t_n + \frac{h}{2}\right), x\left(t_n + \frac{h}{2}\right) + f(t_{n+1}, x(t_{n+1})\right).
\end{aligned}$$

We estimate these quantities as

$$\begin{aligned}
f(t_n, x(t_n)) &\approx f(t_n, y_n) = A,\\
2f\left(t_n + \frac{h}{2}, x\left(t_n + \frac{h}{2}\right)\right) &\approx 2f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}A\right) = 2B,\\
2f\left(t_n + \frac{h}{2}, x\left(t_n + \frac{h}{2}\right)\right) &\approx 2f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}B\right) = 2C,\\
f(t_n + h, x(t_n + h)) &\approx f(t_n + h, y_n + hC) = D.
\end{aligned}$$

Then, the Runge-Kutta method is written as

$$y_{n+1} = y_n + \frac{h}{6}(A + 2B + 2C + D).$$

The local truncation error (LTE) is $O(h^4)$. The price we pay here is that to move one time-step, we need 4 function evaluations. This does not scale linearly (i.e. $O(h^9)$ would require a lot more than 9 function evaluations). Genearlly, 4th order Runge-Kutta is considered the "sweet spot."

17.2 Multi-step methods

So far, we have looked at *one-step methods.* This means $y_{n+1} \leftarrow y_n$. There is another class of methods called *Linear multi-step methods.* For example, $y_{n+1} \leftarrow y_n, y_{n-1}, y_{n-(k-1)}$. Consider the example now

$$x(t_{n+1}) - x(t_{n-1}) = \int_{t_{n-1}}^{t_{n+1}} f(t, x(t))\,\mathrm{d}t.$$

What quadrature method to use to estimate the RHS? Newton-Cotes with $n = 2$ since $n = 1$ just give backward/forward Euler with larger step size $2h$.

$$\begin{aligned}
x(t_{n+1}) - x(t_{n-1}) &\approx \frac{2h}{6}(f(t_{n-1}, x(t_{n-1})) + 4f(t_n, x(t_n)) + f(t_{n+1}, x(t_{n+1})),\\
y_{n+1} - y_{n-1} &= \frac{2h}{6}(f(t_{n-1}, y_{n-1}) + 4f(t_n, y_n) + f(t_{n+1}, y_{n+1})).
\end{aligned}$$

We can see this is a two-step method since y_{n+1} relies on the two previous steps. This method is implicit since we have y_{n+1} on both sides of the equation. To get the initial conditions, we typically use high order Runge-Kutta method to get the necessary k initial values for a k-step process, then we can start the multi-step.

Remark 17.1. We don't use Gauss Quadrature because the nodes are based on where the roots of the orthogonal roots are, so then that will affect our t_n spacing. Then, the approximation of x evaluated at that point is hard. We have to repeat it for each interval.

Suppose I have two points

$$(t_n, \underbrace{y_n}_{\approx x(t_n)}), \quad (t_{n-1}, \underbrace{y_{n-1}}_{\approx x(t_n - 1)}).$$

Let's interpolate between these points. $f(t, x(t)) = \bar{f}(t)$,

$$z_1 = (t_n, \underbrace{f(t_n, y_n)}_{\approx \bar{f}(t_n)}), z_0 = \quad (t_{n-1}, \underbrace{f(t_n, y_{n-1})}_{\approx \bar{f}(t_{n-1})}).$$

We have 2 points, so the best we can do is linear interpolation.

$$\hat{f}(t) = f(t_n, y_n) + \frac{t - t_n}{h}(f(t_n, y_n) - f(t_n, y_{n-1}))$$

is a linear interpolations of (z_1, z_0). Then,

$$\int_{t_n}^{t_{n+1}} \frac{f(t, x(t))}{\bar{f}(t)} \,\mathrm{d}t \approx \int_{t_{n-1}}^{t_n} \hat{f}(t) \,\mathrm{d}t,$$

$$y_{n+1} = y_n + \frac{h}{2}(3f(t_n, y_n) - f(t_{n-1}, y_{n-1}))$$

is an explicit multi-step method. This can be generalized to get more points and create a higher order interpolation. No matter what interpolation, it will be a Lagrange polynomial, so it is easy to integrate.

Generalized. A general k-step method is

$$\sum_{j=0}^{k} a_j y_{n+j} = h \sum_{j=0}^{k} b_j f(t_{n+j}, t_{n+j}), \ a_k = 1.$$

If $b_k = 0$, then the method is explicit. If not, then it is an implicit method and we have to find the root.

Errors. LTE is $\frac{1}{h}(\text{LHS} - \text{RHS})$ with $y_n, \dots y_{n+k}$ replaced by $x(t_n), \dots, x(t_{n+k})$. If we want $e_n = x(t_n) - y_n \to 0 \ \forall n$. The most important equation

$$\boxed{\text{convergence} = \text{consistency} + \text{stability.}}$$

- *Consistency*: LTE $\to 0$ as $h \to 0$. Then just need LTE $O(h^0)$.
- *zero-Stability*: Accumulations of error is small.

Definition 17.2. A linear k-step is *zero-stable* if $\exists C > 0$ such that sequences $(y_n), (z_n)$ based on $(y_0, \dots, y_{k-1}), \ (z_0, \dots, z_{k-1})$ such that

$$|y_n - z_n| \leq C \max_{0 \leq i \leq k-1} |y_i - z_i| \text{ as } h \to 0,$$

where $n = 0, \dots, \frac{T - t_0}{h}$.

What's an easier way to check stability instead of just running it and seeing if it converges? Use a *characteristic polynomial.* From the LHS of the formula for general k-step method, which is

$$a_0 y_0 + a_1 y_{n+1} + a_2 y_{n+1} + \cdots + a_k y_{n+k},$$

define

$$\rho(z) = a_0 + a_1 z + a_2 z^2 + \cdots + a_k z^k.$$

Theorem 17.3 (Root condition). *If $\rho(z)$ has all roots inside $\{z \in \mathbb{C} : |z| \leq 1\}$. For roots on $\{z \in \mathbb{C}, |z| = 1\}$, they need to be simple (no multiplicity), then the method is zero-stable. (This is an iff).*

All one-step methods are zero-stable because the characteristic polynomial is $\rho(z) = z - 1$ with a simple root.

Example 17.4. Consider a two-step method (LIAF)

$$y_{n+1} = -4y_n + 5y_{n-1} + h(4f(t_n, y_n) + 2f(t_{n-1}, y_{n-1})).$$

Then

$$\rho(z) = z^2 + 4z - 5 = (z+5)(z-1).$$

From the root condition, LIAF is not zero-stable. But this method is consistent, which we can see from the local truncation error. The LTE for this method is (with $x'(t_n) = f(t_n, x(t_n))$)

$$\frac{1}{h}(x(t_{n+1}) + 4xt(_n) - 5x(t_{n-1}) - 4hx'(t_n) - 2hx'(t_{n-1})$$

Taylor expand end terms about the center term.

$$\begin{aligned} x(t_{n+1}) &= x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) + \frac{h^3}{6}x'''(t_n) + \frac{h^4}{12}x^{(4)}(\xi), \\ x(t_{n-1}) &= x(t_n) - hx"(t_n) + \frac{h^2}{2}x''(t_n) - \frac{h^3}{6}x'''(t_n) + \frac{h^4}{12}x^{(4)}(\eta), \\ x'(t_{n-1}) &= x'(t_n) - hx''(t_n) + \frac{h^2}{2}x'''(t_n) - \frac{h^3}{6}x^{(4)}(\zeta), \\ hf(t_n, x(t_n)) &= hx'(t_n), \\ hf(t_{n-1}, x(t_{n-1})) &= hx'(t_n - h) = hx'(t_{n-1}) \\ &= hx'(t_n) - h^2x''(t_n) + \frac{h^3}{2}x'''(t_n) - \frac{h^4}{6}x'''(\zeta) \end{aligned}$$

18 (10/24/23)

Continuing from last time, we can make a table

	$O(1)$	$O(h)$	$O(h^2)$	$O(h^3)$	$O(h^4)$
$x(t_{n+1})$	x	x	$\frac{1}{2}x''$	$\frac{1}{6}x'''$	$\frac{1}{12}x^{(4)}(\xi)$
$4x(t_n)$	$4x$				
$-5x(t_{n-1})$	$-5x$	$5x'$	$-\frac{5}{2}x''$	$\frac{5}{6}x'''$	$-\frac{5}{12}x^{(4)}(\eta)$
$-4hx'(t_n)$		$-4x'$			
$-2hx'(t_{n-1})$		$-2x'$	$2x''$	$-x'''$	$\frac{1}{3}x^{(4)}(\xi)$
$+$	0	0	0	0	$\neq 0$

So, $h\text{LTE} = hL_n = O(h^4)$, so $L_n = O(h^3)$. The method has $\rho(z) = z^2+4z-5 = 0 = (z+5)(z-1)$. Since one of these roots is outside the unit circle, this method is not stable.

18.1 System of Equations

Instead of dealing with a scalar, we are now with vectors

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0.$$

Here, $f(t, \cdot) : \mathbb{R}^d \to \mathbb{R}^d,\ \mathbf{x} \in \mathbb{R}^d$. Assume $\|f(t, y_1) - f(t, y_2)\| \leq L\|y_1 - y_2\|$. All discrete vector norms are equivalent, meaning $\alpha\|\cdot\|_a \leq \|\cdot\|_b \leq \beta\|\cdot\|_a$. Now, we have existence and uniqueness.

- $\|e_{n+1}\| \leq \|e_n\| + hL_\Phi \|e_n\| + h\,\|L_n\|$

- $\mathbf{k}_1 \to \mathbf{k}_2 \to \mathbf{k}_3$ (Rk)
- $\mathbf{y}_{n+1} = \mathbf{y}_n + hf(t_{n+1}, \mathbf{y}_{n-1})$
- $g(z) = z - \mathbf{y}_n - hf(t_{n+1}, \mathbf{y}_{n-1}) = \mathbf{0}$
- Newton's method uses Jacobian matrix. Let $y_{n+1}^{(0)}$. Then $y_{n+1}^{(k+1)} = y_{n+1}^{(k)} - J_{y_{n+1}^{(k)}}^{-1} g(y_{n+1}^{(k)})$

18.2 Absolute Stability

Definition 18.1. A *stiff problem* is an ODE where certain numerical methods for solving it are not numerically stable (numerical solution does not make sense) unless the mesh size h is extremely small.

Even with a method that converges, a stiff problem can prevent things from going well.

Example 18.2. Solve $\frac{\mathrm{d}x}{\mathrm{d}t} = \lambda x$ for $\lambda < 0$ and $x(0) = 1$.

- Analytical solution $x(t) = e^{\lambda t}$
- Forward Euler scheme is $x_{n+1} = x_n + h\lambda x_n = (1+\lambda h)x_n$. Then, $x_n = (1+\lambda h)^n x_0 = (1+\lambda h)^n$. Only converges when $|1 + \lambda h| < 1$. Then we have
$$\begin{aligned} -1 &< 1 + \lambda h < 1 \\ -2 &< \lambda h < 0 \\ -\frac{2}{\lambda} &> h > 0 \end{aligned}$$
a condition on our h.
- Backward Euler scheme is $x_{n+1} = x_n + h\lambda x_{n+1} = \frac{1}{1-h\lambda}x_n$. We have $x_n = \left(\frac{1}{1-h\lambda}\right)^{n+1}$, since $\lambda < 0$, this always converges for all h. This is the benefit of the implicit method.

Then for general Linear Multistep Method (LMM): Test the method using the example just shown
$$\begin{aligned} \sum_{j=0}^{k} a_j y_{n+j} = h\sum_{j=0}^{k} \beta_j f(t_{n+j}, y_{n+j}) &= h\sum_{j=0}^{k} \beta_j \lambda y_{n+j} \\ \sum_{j=0}^{k}(a_j - \lambda h \beta_j) y_{n+j} &= 0 \\ \pi(z; \lambda h) = \sum_{j=0}^{k}(a_j - \lambda h \beta_j) z^j &= 0 \end{aligned}$$

Root-condition (zero stability) with $\dot{x} = 0$. Root test (A stability) is $\dot{x} = \lambda x$. Let $\{z_r(\lambda h)\}$ be the roots. Look at the region
$$D = \{s = \lambda h \in \mathbb{C} : |z_r(s)| < 1\}.$$
If D_{LMM} covers the entire negative (left) complex half-plane, then LMM is A-stable.

Example 18.3. Forward Euler: $y_{n+1} = y_n + \lambda h y_n$. The corresponding polynomial is $\pi(z; \lambda h) = z + (-1 - \lambda h)$ and the roots $z_r = 1 + \lambda h = z_r(\lambda h)$. Then, $z_r(s) = 1 + s$. Looking at $|1 + s| < 1$ on $\mathbb{C}$, it does not cover the whole half-plane.

Backward Euler: $y_{n+1} = y_n + \lambda h y_{n+1}$. Then $\pi(z; \lambda h) = (1-\lambda h)z - 1 = 0$. $z_r(\lambda h) = \frac{1}{1-\lambda h}$, $z_r(s) = \frac{1}{1-s}$. Then $D_{\text{BWD}} = \{s \in \mathbb{C} : \left|\frac{1}{1-s}\right| < 1\} = \mathbb{C}^-$.

19 Boundary value problems (10/26/23)

Alex Townsend Lecture.

BVPs are similar to IVPs, except they impose constraints on the solution at "boundaries." The simplest BVP looks like

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

This BVP can be solved by hand, i.e. writing the solution as $y(x) = a\cos(x) + b\sin(x)$ and then plugging in the BCs to get a, b explicitly.

Most BVPs need to be solved numerically. I.e let's solve the example

$$\begin{cases} y''(x) + x^2 y(x) = 0 \\ y(-1) = 0, y(1) = 2 \end{cases}.$$

A BVP is made up of 3 ingredients:

1. Differentiation (e.g. derivatives like $y''(x)$).
2. Multiplication (e.g. multiplications like $x^2 \cdot y(x)$.
3. BCs (e.g. boundary conditions $y(1) = 2$).

If we can discretize 1-3, then we can discretize BVPs.

19.1 Finite differences

The idea of finite differences is to replace $y(x)$ by values on a grid, and then discretize 1-3 using the grid. In the end, you solve for the values of $y(x)$ at the grid. Let $y_j = y(x_j)$ for $j = 0, \ldots, n+1$. $h = 2/(n+1)$.

Discretize differentiation

We want a matrix D_2 such that

$$D_2 \begin{bmatrix} y_0 \\ \vdots \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} z_0 \\ \vdots \\ z_{n+1} \end{bmatrix}, \quad z_j \approx y''(x_j).$$

Note that

$$y''(x_j) \approx \frac{y(x_{j-1} - 2y(x_j) + y(x_{j+1})}{h^2}.$$

So,

$$D_2 = \frac{1}{h^2} \begin{bmatrix} -1 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -1 \end{bmatrix}.$$

Discretizing multiplication

We want a matrix Ma so that

$$M_a \begin{bmatrix} y_0 \\ \vdots \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} z_0 \\ \vdots \\ z_{n+1} \end{bmatrix}, \quad z_j \approx a(x_j)y(x_j).$$

So,

$$M_a = \begin{bmatrix} a(x_0) & & & \\ & a(x_1) & & \\ & & \ddots & \\ & & & a(x_{n+1}) \end{bmatrix}.$$

Discretizing boundary conditions

$y(-1) = 0$ is $y_0 = 0$ and $y(1) = 2$ is $y_{n+1} = 2$. We can write these as

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_{n+1} \end{bmatrix} = 0, \quad \begin{bmatrix} 0 & \cdots 0 & 1 \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_{n+1} \end{bmatrix} = 2.$$

Putting these together gives us

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \frac{1}{h^2} & \frac{-2}{h^2} + x_1^2 & \frac{1}{h^2} & & & \\ & \frac{-1}{h^2} & -\frac{-2}{h^2} + x_2^2 & -\frac{1}{h^2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{h^2} & \frac{-2}{h^2} + x_n^2 & \frac{1}{h^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ \vdots \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 2 \end{bmatrix}.$$

This is a finite difference discretization of $y''(x) + x^2 y(x) = 0$ with $y(-1) = 0, y(1) = 2$.

19.2 Spectral methods

Another way to discretize BVPs is using Legendre polynomials, i.e.,

$$y(x) \approx \sum_{j=0}^{N} a_j P_j(x).$$

Discretizing differentiation

$$P_n'(x) = \frac{2P_{n-1}(x)}{\|P_{n-1}\|^2} + \frac{2P_{n-3}(x)}{\|P_{n-3}\|} + \cdots = (2(n-1)+1)P_{n-1}(x) + (2(n-3)+1)P_{n-3}(x) + \cdots.$$

We want a

$$D_2 \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}, \quad \text{where } y'(x) = \sum_{j=0}^{N} b_j P_j(x).$$

So, where we indicate the n-th column

$$D_n = \begin{bmatrix} 0 & 1 & \vdots & 0 \\ \vdots & 0 & 0 & \vdots \\ & \vdots & 2(n-3)+1 & \\ & & 0 & \\ & & 2(n-1)+1 & \\ & & 0 & 0 \\ & & \vdots & 2(N-3)+1 \\ & & & 0 \\ 0 & 0 & 0 & 2(N-1)+1 \end{bmatrix}$$

Discretizing multiplication

Recall

$$xP_n(x) = \frac{(n+1)P_{n+1}(x)}{2n+1} + \frac{nP_{n-1}(x)}{2n+1}, \ n \geq 1, \quad xP_0(x) = P_1(x).$$

So we want

$$M_x \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_N \end{bmatrix}, \text{ where } xy(x) \approx \sum_{j=0}^{N} b_j P_j(x),$$

Then (n used for n-th column),

$$M_x = \begin{bmatrix} 0 & \frac{1}{3} & & & \\ 1 & 0 & & & \\ 0 & \frac{2}{3} & \ddots & \frac{n}{2n+1} & \\ \vdots & 0 & & 0 & \\ 0 & 0 & & \frac{n+1}{2n+1} & \end{bmatrix}.$$

Then $x^2y(x)$ will be represented by M_x^2.

Discretizing boundary conditions

$$\sum_{j=0}^{N} a_j \underbrace{P_j(-1)}_{(-1)^j} = 0 = \begin{bmatrix} P_0(-1) & \cdots & P_N(-1) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix},$$

$$\sum_{j=0}^{N} a_j P_j(1) = 2 = \begin{bmatrix} P_0(1) & \cdots & P_N(1) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix}.$$

This means $y''(x) + x^2y(x) = 0$, $y(-1) = 0$, $y(1) = 2$ is discretized as

$$(D_2 + M_x^2)\begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$\text{subject to } \begin{bmatrix} 1 & -1 & 1\cdots & (-1)^N \end{bmatrix}\begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} = 0,\ \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}\begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} = 2.$$

We usually remove the last two rows of the linear system to replace with boundary conditions.

20 BVP continued (10/21/23)

Definition 20.1 (Boundary value problem (BVP)).

$$\begin{cases} y'' = F(x, y, y') & \text{on } (a, b) \\ \text{Boundary conditions} & \text{at } x = a, x = b. \end{cases}$$

Let us consider a simple case (from last lecture).

Example 20.2.

$$\begin{cases} y''(x) + x^2y(x) = 0 \\ y(-1) = 0,\ y(1) = 2 \text{ (Dirichlet BC)} \end{cases}$$

We will use 4 methods to numerically solve this:

1. Finite difference method
2. Collocation method
3. Shooting method
4. Galerkin method (Finite element)

20.1 Finite difference method

Discretize the domain. $a = x_0$, $b = x_N$, with $x_i = a + ih,\ i = 1, \dots, N$ and $h = \frac{b-a}{N}$. We have that

$$\begin{aligned} y(x_i) &\approx y_i, \\ y'(x_i) &\approx \frac{y(x_i + h) - y(x_i)}{h} \approx \frac{y_{i+1} - y_i}{h}, \text{ or} \\ &\approx \frac{y(x_i - h) - y(x_i)}{-h} \approx \frac{y_i - y_{i-1}}{h}, \text{ or} \\ &\approx \frac{y(x_i + h) - y(x_i - h)}{2h} \approx \frac{y_{i+1} - y_{i-1}}{2h}, \\ y''(x_i) &\approx \frac{y'(x_i) - y'(x_{i-1})}{h} \approx \frac{\frac{y_{i+1}-y_i}{h} - \frac{y_i - y_{i-1}}{h}}{h} = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}. \end{aligned}$$

The finite difference approximations for $y'(x_i)$ and $y''(x_i)$ are not unique. Different schemes have different local truncation error (LTE).

$$\underbrace{\begin{bmatrix} 1 & & & & & \\ \frac{1}{h^2} & -\frac{2}{h^2}+x_1^2 & \frac{1}{h^2} & & & \\ & \frac{1}{h^2} & -\frac{-2}{h^2}+x_2^2 & \frac{1}{h^2} & & \\ & & \ddots & \ddots & \ddots & \\ & & \frac{1}{h^2} & -\frac{2}{h^2}+x_{N-1}^2 & \frac{1}{h^2} & \\ & & & & 1 & \end{bmatrix}}_{A \text{ (sparse)}} \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \\ y_N \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{b}.$$

Solve for vector y through fast numerical linear algebra.

Numerical analysis. LTE

$$\begin{aligned} L_n &= \frac{y_{n-1}-2y_n+y_{n+1}}{h^2} + x_n^2 y_n, \text{ with } y_n \text{ replaced by true solution } y(x) \\ &= \frac{y(x_n-h)-2y(x_n)+y(x_n+h)}{h^2} + x_n^2 y(x_n). \end{aligned}$$

$$\begin{aligned} y(x_n-h) &= y(x_n) - hy'(x_n) + \frac{h^2}{2}y''(x_n) - \frac{h^3}{6}y'''(x_n) + \frac{h^4}{24}y^{(4)}(\xi), \\ y(x_n+h) &= y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{6}y'''(x_n) + \frac{h^4}{24}y^{(4)}(\zeta). \end{aligned}$$

Shoving it all in,

$$\begin{aligned} L_n &= \frac{1}{h^2}\left(h^2 y''(x_n) + \frac{h^4}{24}y^{(4)}(\xi) + \frac{h^4}{24}y^{(4)}(\zeta)\right) + x_N^2 y(x_n) \\ &= \underbrace{(y''(x_n) + x_n^2 y(x_n))}_{=0 \text{ bc BVP}} + h^2\left(\frac{1}{24}y^{(4)}(\xi) + \frac{1}{24}y^{(4)}(\zeta)\right) \\ &= O(h^2),\ n = 1, 2, \dots, N-1. \end{aligned}$$

Define $w_n = y(x_n) - y_n$ (Global error) as

$$A \begin{bmatrix} y(x_0) \\ y(x_1) \\ \vdots \\ \vdots \\ y(x_N) \end{bmatrix} = \begin{bmatrix} 0 \\ L_1 \\ L_2 \\ \vdots \\ L_{N-1} \\ 0 \end{bmatrix},$$

and then subtract $Ay = b$, to get

$$A \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ \vdots \\ e_{N-1} \\ e_N \end{bmatrix} = \begin{bmatrix} 0 \\ L_1 \\ L_2 \\ \vdots \\ L_{N-1} \end{bmatrix} \implies e_0 = 0, e_N = 0.$$

How about $\{e_n\}_{n=1}^{N-1}$? Remember that convergence requires stability and consistency. Here, stability comes from properties of A and consistency is $L_n = O(h^2)$. A has maximum principle (Theorem 13.3-13.4 in book), so $e_n = O(h^2)$.

What if we have other BCs?

- $y'(a) = \alpha,\ y'(b) = \beta$ (Neumann BC)
- $y''(a) = \alpha,\ y''(b) = \beta$.

We introduce *ghost points* y_{N+1} and y_{-1}. Then,

$$y'(a) \approx \frac{y_1 - y_{-1}}{2h}, \quad y'(b) = \frac{y_{N+1} - y_N}{2h}$$

Then the system looks like

$$\begin{bmatrix} -\frac{1}{2h} & 0 & \frac{1}{2h} & & \\ \frac{1}{h^2} & -\frac{2}{h^2} + x_0^2 & \frac{1}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{h^2} & -\frac{2}{h^2} + x_N^2 & \frac{1}{h^2} \\ & & -\frac{1}{2h} & 0 & \frac{1}{2h} \end{bmatrix} \begin{bmatrix} y_1 \\ y_0 \\ \vdots \\ y_N \\ y_{N+1} \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{bmatrix}.$$

We can rewrite it to get a tridiagonal matrix still which will give us

$$\tilde{A} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix} = \tilde{b}.$$

Similarly, for $y''(a) = \alpha,\ y''(b) = \beta$, then

$$\frac{y_{-1} - 2y_0 + y_1}{h^2} = \alpha, \quad \frac{y_{N-1} - 2y_N + y_{N+1}}{h^2} = \beta.$$

You get different matrix $Ay = b$.

20.2 Collocation method

This is approximating $y(x)$ by a finite linear combination of basis functions.

$$y(x) \approx \sum_{j=0}^{N} a_j \phi_j(x),$$

where $\{\phi_j\}$ can be polynomials, Fourier basis (these 2 are spectral methods), or piecewise-polynomials (finite element method).

1. $y'(x) \approx \sum_{j=0}^{N} b_j \phi_j(x)$, where $\begin{bmatrix} b_0 \\ \vdots \\ b_N \end{bmatrix} = D \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix}$.
2. $xy(x) \approx \sum_{j=0}^{N} c_j \phi_j(x)$, where $\begin{bmatrix} c_0 \\ \vdots \\ c_N \end{bmatrix} = M_x \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix}$.

3. $\sum_{j=0}^{N} a_j \phi_j(x=a) = \alpha, \quad \sum_{j=0}^{N} a_j \phi_j(x=b) = \beta.$

- **Method 1.** (Spectral method) We find a set of $(x_0, x_1, \ldots, x_{N-1})$, a *collocation* in the domain. Then we have BC (3 from above) and
$$\sum_{j=0}^{N} a_j \phi''(x_i) + x_i \left(\sum_{j=0}^{N} a_j \phi_j(x_i) \right) = 0, \ i = 0, 1, \ldots, N-1.$$
- **Method 2.** (spectral method)
$$(D^2 + M_x^2) \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$
subject to 3 from above.

21 (11/2/23)

Recap. Solve
$$\begin{cases} y'' = F(x, y, y') & \text{on } (a,b) \\ \text{BC} & \text{at } x = a, \ x = b. \end{cases}$$

1. Finite difference mothod approximates y'', y' by divided difference. Tackle BC using ghost points if not Dirichlet.
2. Collocation Method.
   (a) Find a linear space $V = \operatorname{span}\{\phi_0, \phi_1, \ldots, \phi_n\}$, $\{\phi_i\}$ are functions (could be nonlinear space as well).
   (b) Approximate $y(x) \approx \sum_{j=0}^{n} a_j \phi_j(x)$.
   (c) Plug in $y(x)$ into BVP. $(n+1)$ unknowns. I.e. $y'' + x^2 y' = 0, \ y(1) = 2, y(-1) = 0$
$$\sum_{j=0}^{n} a_j \phi_j''(x) + x^2 \sum_{j=0}^{n} a_j \phi_j(x) = 0.$$
   (d) Find N collocation points $\{x_k\}$ (not equally spaced)
   (e) Find $\{a_k\}_{j=0}^{n}$ such that
$$\begin{cases} \sum_{j=0}^{n} a_j \phi_j''(x_k) + x_k^2 \sum_{j=0}^{n} a_j \phi_j(x_k) = 0, \quad k = 1, 2, \ldots, N \\ \sum_{j=0}^{n} a_j \phi_j''(-1) + \sum_{j=0}^{n} a_j \phi_j(-1) = 0, \quad \sum_{j=0}^{n} a_j \phi_j''(1) + \sum_{j=0}^{n} a_j \phi_j(1) = 2 \end{cases},$$
   gives the system
$$\underbrace{\begin{bmatrix} \phi_0''(-1) + \phi_0(-1) & \phi_1''(-1) + \phi_1(-1) & \cdots & \phi_n''(-1) + \phi_n(-1) \\ \phi_0''(x_1) + \phi_0(x_1) & \phi_1''(x_1) + \phi_1(x_1) & \cdots & \phi_n''(x_1) + \phi_n(x_1) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0''(1) + \phi_0(1) & \phi_1''(1) + \phi_1(1) & \cdots & \phi_n''(1) + \phi_n(1) \end{bmatrix}}_{(N+2)\times(n+1)} = \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}}_{(n+1)\times 1} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 2 \end{bmatrix}}_{(N+2)\times 1} \quad /$$
   This is dense, which is bad for linear systems.

Recall for spectral method

$$y(x) \approx \sum_{j=0}^{n} a_j \phi_j(x),$$
$$\phi_j(x) = \sum_{k=0}^{j-1} b_k \phi_k(x),$$
$$\phi_j''(x) = \sum_{k=0}^{j-2} c_k \phi_k(x),$$

where $\phi_j(x)$ are Legendre polynomials, and the derivatives have one or two degree less. We do note need collocation points here. Only the relationship among the basis funcionts. This gives us a sparse system.

21.1 Shooting method

Suppose we want to solve the BVP

$$\begin{cases} y'' + x^2 y = 0, \\ y(a) = \alpha, \ y(b) = \beta. \end{cases}$$

Consider instead

$$\begin{cases} y'' + x^2 y = 0, \\ y'(a) = \gamma, \ y(a) = \alpha. \end{cases}$$

Let $u_1 = y, \ u_2 = y', \ u_1(a) = \alpha, \ u_2(a) = \gamma$, then the IVP is written as

$$\begin{cases} u_1' = u_2, \\ u_2' = -x^2 u_1. \end{cases}$$

The *shooting method:*

1. Step 1. Guess γ
2. Step 2. Solve IVP for $\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$
3. Step 3. Check $u_1(b) = y(b)$ if it is β.

The process of solving IVP using a numerical method yields a map $F(\gamma) = u_1(b)$. We want $F(\gamma) = \beta$!!! The problem then becomes a root-finding problem: $F(\gamma) = \beta$. This can be done through bisection or fixed-point iteration.

21.2 Galerkin method

$$y(x) \approx \sum_{j=0}^{n} a_j \phi_j(x) + \psi(x),$$

where (on domain $[a, b] = [-1, 1]$)

$$\begin{aligned}
\psi(a) &= \psi(-1) = 0,\ \psi(b) = \psi(1) = 2,\\
\psi(x) &= \frac{2-0}{b-a}(x-a) + 0 = x + 1\\
\phi_j(a) &= 0,\ \phi_j(b) = 0,\ \forall j = 0, 1, 2, \dots, n
\end{aligned}$$

Therefore, $y(x)$ always satisfies $y(a) = 0,\ y(b) = 2$. Then

$$\begin{aligned}
y''(x) + x^2 y(x) &= 0 \quad \text{multiply both sides by } w(x),\\
w(x)y''(x) + x^2 y(x) w(x) &= 0 \quad \text{integrate over } \int_{-1}^{1},\\
\int_{-1}^{1} w(x)y''(x)\,\mathrm{d}x + \int_{-1}^{1} x^2 y(x) w(x) &= 0 \quad \text{int by parts,}\\
-\int_{-1}^{1} w'(x)y'(x)\,\mathrm{d}x + w(x)y'(x)|_{-1}^{1} + \int_{-1}^{1} x^2 y(x) w(x) &= 0
\end{aligned}$$

Assume $w(x) = 0$ on points $x = -1, 1$. Then

$$\boxed{-\int_{-1}^{1} w'(x)y'(x)\,\mathrm{d}x + \int_{-1}^{1} x^2 y(x) w(x)\,\mathrm{d}x = 0}$$

The Galerkin method is to consider $w(x) = \phi_j(x)$ for $j = 0, 1, \dots, n$ and $y(x) \approx \sum_{j=0}^{n} a_j \phi_j(x) + \psi(x)$ and find $\{a_j\}$. Then,

$$\int_{-1}^{1} \phi_j'(x) \sum_{k=0}^{n} a_k \phi_k'(x)\,\mathrm{d}x + \int_{-1}^{1} \phi_j'(x)\psi'(x)\,\mathrm{d}x - \int_{-1}^{1} \phi_j(x) x^2 \left(\sum_{k=0}^{n} a_k \phi_k(x) + \psi(x) \right) \mathrm{d}x = 0.$$

Consolidating,

$$\begin{aligned}
\sum_{k=0}^{n} a_k \left(\int_{-1}^{1} \phi_j'(x)\phi_k'(x)\,\mathrm{d}x - \int_{-1}^{1} x^2 \phi_j(x)\phi_k(x)\,\mathrm{d}x \right) &= \int_{-1}^{1} \phi_j(x)\psi(x)x^2\,\mathrm{d}x\\
&\quad - \int_{-1}^{1} \phi_j'(x)\psi'(x)\,\mathrm{d}x,\ \forall j = 0, \dots, n.\\
\underbrace{M}_{(n+1)\times(n+1)}\ \underbrace{a}_{(n+1)\times 1} &= \underbrace{b}_{(n+1)\times 1}.
\end{aligned}$$

We can write $M = M_1 + M_2$, where M_1 is the mass matrix and M_2 is the stiff matrix defined by

$$\begin{aligned}
(M_1)_{kj} = (M_1)_{jk} &= \int_{-1}^{1} \phi_j'(x)\phi_k'(x)\,\mathrm{d}x,\\
(M_2)_{kj} = (M_2)_{jk} &= -\int_{-1}^{1} x^2 \phi_j(x)\phi_k(x)\,\mathrm{d}x,\ k, j = 0, 1, \dots, n.
\end{aligned}$$

Also,

$$b_j = \int_{-1}^{1} \phi_j(x)\psi(x)x^2\,\mathrm{d}x - \int_{-1}^{1} \phi_j'(x)\psi'(x)\,\mathrm{d}x.$$

If $\{\phi_j\}$ are polynomials, M is dense. If $\{\phi_j\}$ are piecewise-polynomials, M is sparse. This often looks like a local support that makes triangular piecewise linear functions.

22 (11/7/23)

More on Galerkin. Consider the BVP

$$\begin{cases} y''(x) + x^2 y(x) = 0 \\ y(-1) = 0,\ y(1) = 2/ \end{cases}$$

We can write $y(x) = y_0(x) + (x+1)$ with $y_0(-1) = y_0(1) = 0$. Then plugging in,

$$(y_0 + x + 1)'' + x^2(y_0 + x + 1) = 0,$$
$$y_0'' + x^2 y_0 = -x^2(x+1) = -f(x), \quad y_0(-1) = y_0(1) = 0.$$

Is an equivalent formulation of the BVP to give easier boundary conditions. Just choose basis to be 0 at those points. We can do this because underlying ODE/PDE is linear.

- Finite difference and shooting methods are the same category for IVP, by approximating solution on a grid.
- Spectral, collocation, and Galerkin methods are are in another category by approximating the solution as a sum of basis functions. $y_0(x) \approx \sum_{j=1}^n a_j \phi_j(x)$. The coefficients a_j are the unknowns. Functions are infinite dimensional, but this problem is now finite dimensional for finding coefficients.

1. **Collocation method.**
$$y_0(x) \approx \sum_{j=1}^n a_j \phi_j(x) = \bar{y}_0(x).$$
Residual $R(x) = \bar{y}_0'' + x^2 \bar{y}_0 + f(x)$ since we don't know what y_0 is. Residual is 0 if $\bar{y}_0$ is y_0. For collocation method, enforce $R(x) = 0$ at $\{x_i\}$, so
$$0 = R(x_i) = \int_{-1}^{1} R(x)\delta(x - x_i)\,\mathrm{d}x.$$

2. **Least-squares method.**
$$\min_{a_1,\dots,a_n} \int_{-1}^{1} |R(x)|^2\,\mathrm{d}x.$$
We like 2-norm because optimality condition gives linear system and we can also differentiate it. (A lot harder if just abs$\hat{1}$).

3. **Subdomain method.**
$$0 = \int_{x_{i-1}}^{x_i} R(x)\,\mathrm{d}x = 0,\ \{[x_{i-1}, x_i)\}.$$
This allows for parallelization.

4. **Galerkin method.**
$$0 = \int_{-1}^{1} R(x)\phi_j(x) = 0,\ j = 1, \dots, n.$$

They all want $\int_{-1}^{1} R(x)w(x)\,\mathrm{d}x = 0$ or small. Collocation has $w(x) = \delta(x - x_i)$, least squares has $w(x) = R(x)$, subdomain has $w(x) = \mathbb{I}_{[x_{i-1},x_i)}$, Galerkin has $w(x) = \phi_j(x)$, $j = 1, \ldots, n$. So we have $F(\bar{y}_0) \to F(y_0)$ from $\bar{y}_0 \to y_0$.

More on Galerkin. Another interpretation of the residual is that R is orthogonal to ϕ_j, so R is orthogonal to the entire linear space. So, $R(x) \perp V$, where $V = \operatorname{span}\{\phi_1, \ldots, \phi_n\}$. $\psi(x)$ is a test function. Want $\psi(x) = 0$ at -1 and 1. Then

$$\begin{aligned}
\int_{-1}^{1} y_0''\psi + \int_{-1}^{1} x^2 y_0 \psi &= \int_{-1}^{1} -f\psi, \\
\underbrace{-\int_{-1}^{1} y_0'\psi' + y_0'\psi|_{-1}^{1}}_{=0} - \int_{-1}^{1} y_0'\psi' + \int_{-1}^{1} x^2 y_o \psi &= -\int_{-1}^{1} f\psi \\
\underbrace{\int_{-1}^{1} y_0'\psi' - \int_{-1}^{1} x^2 y_o \psi}_{A(y_0,\psi)} &= \int_{-1}^{1} f\psi.
\end{aligned}$$

Notice that A is

1. Symmetric $A(y_0, \psi) = A(\psi, y_0)$,
2. Bilinear.

Define

$$J(\psi) = \frac{1}{2} A(\psi, \psi) - \int_{-1}^{1} f\psi.$$

We want to minimize $J(\psi)$ with $\psi \in V$. So, $\psi = \sum a_j \phi_j(x)$. Optimality condition will be

$$M \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}.$$

This optimal ψ is what is obtained from the Galerkin method. This is called *Rayleigh-Ritz principle* (Ritz method). Optimum ($\psi^* = \sum a_j \phi_j(x) = \bar{y}_0^*$) satisfies

$$A(\bar{y}_0^*, \phi_j) = \int_{-1}^{1} f\phi_j.$$

The true solution also satisfies

$$A(y_0, \phi_j) = \int_{-1}^{1} f\phi_j.$$

Subtracting the two and using bilinearity,

$$A(\bar{y}_0^* - y_0, \phi_j) = 0.$$

Then A defines the orthogonality as $\bar{y}_0^* - y_0 \perp_A V$. Best 2-norm approximation induced by the PDE/ODE and BVP, which is A.

$$\|\psi\|_A = \sqrt{A(\psi, \psi)}.$$

Finally, the Galerkin method is

$$\|y_0 - \bar{y}_0^*\|_A = \min_{\psi \in V} \|y_0 - \psi\|_A^2.$$

Rewrite our previous equation defining A with $\psi \to \phi_j$ and $y_0 \to \sum a_j\phi_j$ gies

$$\int_{-1}^{1} \phi_j' \left(\sum_{k=1}^{n} a_k\phi_k'\right) + \int_{-1}^{1} (-x^2)\phi_j \left(\sum_{k=1}^{n} a_k\phi_k\right) = \int_{-1}^{1} \phi_j f \, \mathrm{d}x$$

$$(M_1 + M_2)a = b;$$

$$(M_1)_{jk} = \int_{-1}^{1} \phi_j'\phi_k' \quad \text{(stiffness matrix)}, \quad (M_2)_{jk} = \int_{-1}^{1} (-x)^2\phi_j\phi_k \, \mathrm{d}x \quad \text{(mass matrix)}.$$

We want to choose ϕ_j to give a sparse matrix.

22.1 Linear Splines

Piecewise polynomials (splines). In interpolation given nodes $x_0, \dots, x_n$ and samples $f(x_0), \dots, f(x_n)$. Piecewise linear polynomials will just connect the nodes with a line.

Given x_k, x_{k+1} and $f(x_k), f(x_{k+1})$, the line connecting them is

$$\frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}(x - x_k) + f(x_k), \ x \in [x_k, x_{k+1}].$$

We can rearrange it as

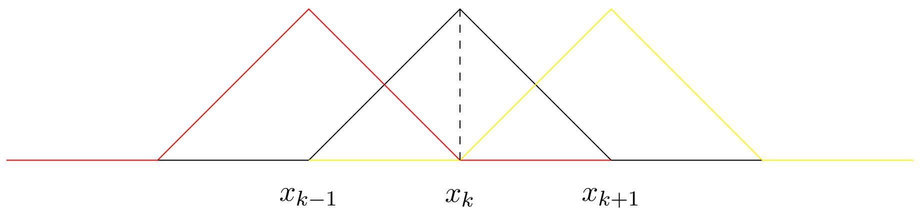
$$f(x_k)\left(1 - \frac{x - x_k}{x_{k+1} - x_k}\right) + f(x_{k+1})\left(\frac{x - x_k}{x_{k+1} - x_k}\right), \ [x_k, x_{k+1}]$$

$$f(x_{k-1})\left(1 - \frac{x - x_{k-1}}{x_k - x_{k-1}}\right) + f(x_k)\left(\frac{x - x_{k-1}}{x_k - x_{k-1}}\right), \ [x_{k-1}, x_k].$$

$f(x_k)$ only shows up in 2 segments. $f(x_k)\psi_k(x)$,

$$\psi_k(x) = \begin{cases} 0 & \\ 1 - \frac{x - x_k}{x_{k+1} - x_k} & x \in [x_k, x_{k+1}) \\ \frac{x - x_{k-1}}{x_k - x_{k-1}} & x \in [x_{k-1}, x_k)] \end{cases}.$$

So then the entire spline is $\sum_{k=0}^{n} f(x_k)\psi_k(x)$. $\psi_k(x)$ looks like a triangular function with compact support with height 1 at x_k over x_{k-1} to x_{k+1}.



23 (11/9/23)

Continuing on Galerkin, draw analogue to Lagrange interpolation $p(x) = \sum_{k=0}^{N} f(x_k)L_k(x)$ is an n-th degree polynomial. So, then

$$(M_1)_{jk} = \begin{cases} 0 & \text{if } |j-k| \geq 2, \\ \neq 0 & \text{if } j = k+1, k-1, \\ \neq 0 & \text{if } j = k. \end{cases}$$

So, M_1 is tridiagonal. Same for M_2.

Note in real finite element, do not have to use linear, can use higher order splines. In real finite element methods, the main difficulty is to build the elements for different geometries.

23.1 Cubic Splines

Any

$$p_j = a_j + b_j + (x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3.$$

We need in total $4N$ degrees of freedom. We enforce

1. We have is $N + 1$ data,
2. g, g', g'' continuous on interval $[x_0, x_N]$. This gives $3(N - 1)$ data.
3. Natural cubic spline enforces $0 = g''(x_0) = g''(x_N)$.

Hermite Cubic Splines require

1. $g(x_k) = f(x_k)$, $g'(x_k) = f'(x_k)$ gives $2N + 2$ data.
2. $g,\ g'$ continuous gives $2N - 2$ constraints.

Theorem 23.1. *Let $f \in C^2([a, b])$. Let $a = x_0, x_1, \ldots, x_N = b$. y is a natural cubic spline. Then*

$$\|y''\|_{L^2} \leq \|v''\|_{L^2}\ \forall v \in C^2([a, b]),\ v(x_k) = f(x_k)\ k = 0, \ldots, N.$$

This is saying that y is the smoothest possible interpolation. The variational principle

$$y = \text{argmin}_{v \in C^2([a,b]),\ v(x_k) = f(x)k)} \int_a^b \frac{|v''|^2}{1 + |v|^3}\,\mathrm{d}x$$

The basis functions for natural cubic splines, "B-spline."

$$S_{(n)}(x) = \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} (x - kh)_+^n$$

- h is spacing between x_{k+1}, x_k (equally spaced).
- $$(a)_+ = \text{ReLu(a)} = \begin{cases} a, & a \geq 0, \\ 0, & \text{otherwise,} \end{cases}$$

For linear spline,

$$\phi_k(x) = \frac{1}{h} S_{(1)}(x - x_{k-1}).$$

For natural cubic spline,

$$\phi_k(x) = \frac{1}{4h^3} S_{(3)}(x - x_{k-2}).$$

Looking at practice problems. Given heat equation

$$\begin{cases} u_t = k(x)u_{xx}, \ x \in (0,1), t \in (0,b) \\ u(t=0,x) = \sin(\pi x) \\ u(t,0) = u(t,1) = 0. \end{cases}$$

How to start?

1. Discretize or approximation—turn an infinite dimensional method into a finite dimensional problem.
2.
   - FDM. Let's discretize the spatial domain first. So, $u(t, x_i) \approx u_i(t)$ for $i = 0, \ldots, N_x$.
   - FEM. $u(t,x) \approx \sum_{k=1}^{N} a_j(t)\phi_j(x)$.

For today, we do FDM. Plug $u_i(t)$ into the PDE. then

$$u_t \to \begin{bmatrix} u_0(t) \\ \vdots \\ u_{N_x}(t) \end{bmatrix}_t, \quad k(x)u_{xx}|_{x_i} \approx k(x_i)\frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{\Delta x^2}$$

Then the PDE becomes (WLOG assume $k(x) = k$)

$$\begin{bmatrix} u_0(t) \\ \vdots \\ u_{N_x}(t) \end{bmatrix}_t = kA \begin{bmatrix} u_0(t) \\ \vdots \\ u_{N_x}(t) \end{bmatrix}.$$

A is tridiagonal. This is now a system of IVP's. If we do forward Euler, then

$$\begin{bmatrix} u_0(t_{k+1}) \\ \vdots \\ u_{N_x}(t_{k+1}) \end{bmatrix} = \begin{bmatrix} u_0(t_k) \\ \vdots \\ u_{N_x}(t_k) \end{bmatrix} + \Delta t k A \begin{bmatrix} u_0(t_k) \\ \vdots \\ u_{N_x}(t_k) \end{bmatrix}.$$

If we did FEM then, then apply the Galerkin principle, then $b = b(t)$. Note u_{xx} is not Lipschitz, but after discretization it can be "fake" Lipschitz. For explicit schemes, the CFL condition is $\Delta t < \frac{\Delta x^2}{2k}$.

24 (11/14/23)

Discretization is regularization.

Example 24.1 (Integral differential equation).

$$u_t(x,t) = \underbrace{\int_x u \, \mathrm{d}x}_{(t)} - u(x,t) \quad IC/BC.$$

1. Can discretize in x and use right riemann sum,

$$\begin{bmatrix} y_0(t) \\ \vdots \\ y_N(t) \end{bmatrix}_t = \frac{b-a}{N} \begin{bmatrix} 1 & 1 & & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} y_0(t) \\ \vdots \\ y_n(t) \end{bmatrix}$$

2. Solve IVP. If we use Rk4, then 1st order in space and 4th order in time. Error is $O(\Delta x) + O(\Delta t^4)$.

FEM would be approximating $u \approx \sum_{i=1}^{N} a_i(t)\varphi_i(x)$. Then,

$$\begin{bmatrix} a_0(t) & \cdots & a_N(t) \end{bmatrix}_t \begin{bmatrix} \varphi_1(x) \\ \vdots \\ \varphi_N(x) \end{bmatrix} = \begin{bmatrix} a_0 \int \varphi_0(x)\,\mathrm{d}x \\ \vdots \\ a_N \int \varphi_N(x)\,\mathrm{d}x \end{bmatrix} - \begin{bmatrix} a_0\varphi_1 \\ \vdots \\ a_N\varphi_N \end{bmatrix}.$$

General principle is that we will always get an IVP.

Definition 24.2 (Transport equation)**.**

$$u_t + c(x,t)u_x = 0.$$

c is the velocity of the flow.

Definition 24.3 (Continuity equation)**.**

$$u_t + (c(x,t)u)_x = 0.$$

Also known as conservation law.

One of them is the adjoint of the other. Requires special BC. For continuity, $u(0,t) = u(1,t) = 0$. Consider $y(x,t)$ where y solves the continuity equation.

$$\begin{aligned} 0 &= \int\int y(u_t + cu_x)\,\mathrm{d}x\,\mathrm{d}t \\ &= -y_t u + \int_x yu|_{t=0}^{t+T} = \int\int u(cy)_x + \underbrace{\int cuy|_a^b}_{=0} \\ &= -\int\int (y_t + (cy)_x)u\,\mathrm{d}x\,\mathrm{d}t + \int_x y(T,x)u(t,x)\,\mathrm{d}x - \int_x y(0,x)u(0,x)\,\mathrm{d}x, \\ \langle y, L_1 u\rangle &= \langle L_2 y, u\rangle, \end{aligned}$$

where L_1, L_2 are transport and continuity differential operators.

Conseration law

$$\begin{cases} \partial_t u + \nabla \cdot \mathbf{c}u = 0 \\ u(x\cdot \mathbf{n}, t) = 0 \text{ on } \partial\Omega \end{cases}$$

Most methods do not preserve these things that we would want physically, such as positivity or preserve conservation. Some solvers build these in as constraints. Other things such as wanting entropy to decay over time.

25 (11/16/23)

25.1 Review

Differential equations.

$$F(u) = 0, \quad F(u, x, t, \partial_x, \partial_t, \partial_{xy}, \partial_{xx}) = 0.$$

We look for well-posedness:

1. Existence,
2. Uniqueness,
3. Continuous dependence of data (stability).

Discretize solution u as $\hat{u}_N$. Then we also need to discretize the PDE operator to make it $F_N(\hat{u}_N) = 0$.

1. Discretize: $\infty \to \mathbb{R}^N$
   (a) Pointwise representation (grid). (FDM, pseudo-spectral)
   (b) Linear combination of basis of a linear space V (FEM, DG, Spectral)
   $$u_N = \sum_{i=1}^{N} a_i \psi_i(x).$$
   (c) (Today) local average (Finite Volume Method). I.e.
   $$Q_i \approx \int_{x_i}^{x_{i+1}} u(x)\,\mathrm{d}x.$$
   (d) (Today and next time) Particle representation (particle method, Monte Carlo). Start with some particles $x_1^0, x_2^0, x_3^0, \ldots$ at $t = 0$. As the PDE evolves, the x_i location changes to some x_1^1, x_2^1, x_3^1 which represent at $t = 1$.
2. $F_N(\hat{u}_N) = 0$. Solve using linear algebra, fixed-point iteration, or optimization.

25.2 Solving transport equation

The transport equation is

$$\begin{cases} u_t + c(x,t) u_x = 0, & \text{on } (0, L) \times (0, T] \\ u(x, 0) = f(x) & \text{(IC) } , \forall x \in (0, L] \\ u(0, t) = u(L, t) & \text{(BC) } , \forall t \in [0, T] \end{cases} . \tag{25.1}$$

Currently, x, t are independent variables. What if we introduce $X(t) : [0, t] \to [0, L)$. We want it to satisfy

$$\dot{X}(t) = \frac{\mathrm{d}X}{\mathrm{d}t} = c(X, t).$$

X is a curve in x, t-space. Now we restrict the solution u to be only on the curve X. So,

$$u(x = X(t), t) = v(t).$$

Now looking at the dynamics of v,

$$\frac{\mathrm{d}v}{\mathrm{d}t} = u_x \frac{\mathrm{d}X}{\mathrm{d}t} + u_t = u_x(X(t),t)c(X(t),t) + u_t(X(t),t) = 0,$$

we see that $v(t)$ is constant in time.

$$v(T) = \cdot = \underbrace{v(t^*)}_{u(X(t^*),t^*)} = \cdots = \underbrace{v(0)}_{v(X(0),0)}$$

Now to get $u(x^*,t^*)$? We can construct an IVP. Let

$$\begin{cases} \dot{X}(t) = c(X,t) \\ X(t^*) = x^*, \qquad \text{backward in time} \end{cases}.$$

Find $X(0)$ by solving the ODE. Then solving backward,

$$u(x^*,t^*) = u(X(0),0) = f(X(0)).$$

This is known as the *method of characteristics* (or Lagrangian method). So, in the domain $[0,L]$, we have many points at t^* and each flows back to where it goes to $t=0$. Along each curve, the solution is constant.

Forward. An alternative way is to go forward in time. With $u(x,0) = f(x)$.

$$\begin{cases} \dot{X}(t) = c(X,t) \\ X(0) = x_k \end{cases}.$$

To see what solution is doing at a time t_1, then where the characteristic values are at that time at corresponding x gives the solution. Define $G^{t_1} : x_k \to X(t_1)$ with x_k as IC

$$u(x, t=t_1) = \{(G^{t_1}(x_k), f(x_k) = u(G^{t_1}(x_k), t_1))\}_{k=0}^{N_x} \approx \{(X(x_k), f(x_k))\}_{k=0}^{N_x}.$$

PDE is in parts, then we can interpolate. c Lipschitz makes things nice in that we don't have CFL. Can take large steps in x.

Now consider the modified transport equation as

$$\begin{cases} u_t + c(x,t)u_x = u_{xx}, & \text{on } (0,L)\times(0,T] \\ u(x,0) = f(x) & \text{(IC) }, \forall x \in (0,L] \\ u(0,t) = u(L,t) & \text{(BC) }, \forall t \in [0,T] \end{cases}. \tag{25.2}$$

This has advection and diffusion. This will make the solver more complicated. The characteristics we derived won't be constant anymore. We can employ *splitting*

$$\begin{cases} u_t + c(x,t)u_x = 0 \\ \text{IC, BC} u(x,0) = f(x) \end{cases}, \quad \begin{cases} \bar{u}_t = \bar{u}_{xx} \\ \text{IC, BC} \bar{u}(x,0) = u(x,\Delta t) \end{cases}$$

for small Δt. Solve first one in Δt time step then give output to the second system for another Δt time. As long as Δt is small, the order of error $O(\Delta t)$. But the output of the first system is very irregular from method of characteristics, so then we have to interpolate the first system to use it in a grid. This is called a *semi-Lagrangian* method. Now the second half of the solve is subject to CFL. (In Lagrangian, we do not have CFL).

25.3 Solving continuity equation

The continuity equation is

$$\begin{cases} y_t + (c(x,t)y)_x = 0, & \text{on } (0,L) \\ y(x,0) = g(x) & \text{(IC) } \forall x \in [0,L] \\ y(0,t) = y(L,t) = 0 & \text{(BC) },\forall t \in [0,T] \end{cases} . \tag{25.3}$$

$$\int_0^L y(x,t)\,\mathrm{d}x = \text{constant} = \int_0^L g(x)\,\mathrm{d}x.$$

Finite Volume Method. Let $0 = x_0, x_1, \ldots, L = x_4$ be the grid. We also keep track of the center of the cells $x_{1/2}, x_{3/2}, ...x_{7/2}$. We also have ghost points $x_{-1/2}$ and $x_{9/2}$. We are trying to approximate in between the centers. I.e.

$$\int_{x_{1/2}}^{x_{3/2}} y(x,t)\,\mathrm{d}x = \rho_1(t).$$

Plugging into the PDE,

$$\int_{x_{1/2}}^{x_{3/2}} y_t\,\mathrm{d}x + \int_{x_{1/2}}^{x_{3/2}} (cy)_x\,\mathrm{d}x = \frac{\mathrm{d}\rho_1}{\mathrm{d}t} + \underbrace{cy|_{x=3/2}}_{\approx F^k_{3/2}} - \underbrace{cy|_{x=1/2}}_{\approx F^k_{1/2}} = 0$$

We see that cy is the flux. Need a way to eliminate one of ρ or y. Use

$$Q_1^k \approx \frac{\rho_1(x_1,t_k)}{\Delta x}$$

as the cell average (mean mass). Then, we get the scheme

$$\frac{Q_1^{k+1} - Q_1^k}{\Delta t} = -\frac{1}{\Delta x}(F^k_{3/2} - F^k_{1/2}).$$

There is also an *upwind scheme* to approximate the flux F,

$$F^k_{i=1/2} = (c(x_{i-1/2},t_k))_+ Q^k_{i-1} + (c(x_{i-1/2},t_k))_- Q^k_i,$$

where subscript $()_-, ()_+$ denote the negative part and positive part, respectively.

26 Finite Volume and Monte Carlo (11/21/23)

Recap. We have the PDE

$$\begin{cases} \partial_t y + \partial_x(c(x,t)y) = 0, \ x \in (a,b), t > 0 \\ y = 0 \text{ at } x = a,b, \ y(x,0) = g(x) \end{cases}$$

Properties of this equation are

1. $\int_a^b y\,\mathrm{d}x = \rho(t)$ is constant in t,
2. If $g(x) \geq 0$, then $y(x,t) \geq 0$ for all t, x.

We want to design numerical methods with these properties as well.

26.1 Finite Volume Method

Let $a = x_0, x_1, \dots, b = x_N$ be the grid setup with spacing Δx. We also keep track of the center of the cells $x_{1/2}, x_{3/2}, ...x_{N-1/2}$. We also have ghost points $x_{-1/2}$ and $x_{N+1/2}$. Looking at the densities,

$$\begin{aligned}
\rho_0(t) &= \int_{x_{-1/2}}^{x_{1/2}} y(x,t)\,\mathrm{d}x \approx \Delta x y(x_0,t) = y(a,t) \equiv 0,\\
\rho_N(t) &= \int_{x_{N-1/2}}^{x_{N+1/2}} y(x,t)\,\mathrm{d}x \approx \Delta x y(x_N,t) = y(b,t) \equiv 0,\\
\rho_i(t) &= \int_{x_{i-1/2}}^{x_{i+1/2}} y(x,t)\,\mathrm{d}x = \text{ total mass in cell } [x_{i-1/2}, x_{i+1/2}).
\end{aligned}$$

Where the first two approximations were from the mid-point rule for integration. We can then use Q_i^k to approximate its average at time $t = t_k$.

$$Q_i^k \approx \frac{1}{\Delta x}\rho_i(t_k) = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} y(x,t_k)\,\mathrm{d}x \approx y(x_i,t_k).$$

Integrating our original PDE over this interval,

$$\begin{aligned}
\int_{t_k}^{t_{k+1}}\int_{x_{i-1/2}}^{x_{i+1/2}} \partial_t y\,\mathrm{d}x &= -\int_{t_k}^{t_{k+1}}\int_{x_{i-1/2}}^{x_{i+1/2}} \partial_x(c(x,t)y(x,t))\,\mathrm{d}x\\
\rho_i(t_{k+1}) - \rho_i(t_k) &= -\int_{t_k}^{t_{k+1}} c(x_{i+1/2},t)y(x_{i+1/2},t)\,\mathrm{d}t + \int_{t_k}^{t_{k+1}} c(x_{i-1/2},t)y(x_{i-1/2},t)\,\mathrm{d}t.
\end{aligned}$$

If we approximate $c(x_{i+1/2},t_k)y(x_{i+1/2},t_k)$ by $F_{i+1/2}^k$, and use forward Euler, we get

$$Q_i^{k+1} - Q_i^k = -\frac{\Delta t}{\Delta x}(F_{i+1/2}^k - F_{i-1/2}^k).$$

The next question: how to relate Q_i^k with $F_{i+1/2}^k$? Idea: $Q_i^k \approx y(x_i,t_k)$ at *cell center* and $F_{i+1/2}^k \approx c(x_{i+1/2},t_k)y(x_{i+1/2},t_k)$ at *cell edge*. We can try these methods:

1. $F_{i+1/2}^k = Q_{i+1}^k$, $F_{i-1/2}^k = Q_i^k$. This is *very unstable.*
2. (Lax-Friedricks). $F_{i+1/2}^k = \frac{1}{2}(c(x_{i+1/2},t_k)Q_{i+1}^k + c(x_{i-1/2},t_k)Q_i^k) - \frac{\Delta x}{2\Delta t}(Q_{i+1}^k - Q_i^k)$. This is stable with $\frac{\Delta t}{\Delta x}$ sufficiently small.
3. (Upwind). $F_{i+1/2}^k = (c(x_{i+1/2},t_k))_+ Q_i^k + (c(x_{i+1/2},t_k))_- Q_{i+1}^k$.

   (a) If $c_{i-1/2}^k > 0$, then flux at $x_{i-1/2}$ is better approximated by $c_{i-1/2}^k Q_{i-1}^k = F_{i-1/2}^k$ (positive).

   (b) If $c_{i-/12}^k < 0$, then flux at $x_{i-1/2}$ is better approximate by $c_{i-1/2}^k Q_i^k = F_{i-1/2}^k$ (negative).

   To sum up, $F_{i-1/2}^k = (c_{i-1/2}^k)_+ Q_{i-1}^k + (c_{i-1/2}^k)_- Q_i^k$. Where $(g)_+ = \max(g,0)$, $(g)_- = \min(g,0)$. Using 1st order explicit time scheme (forward Euler),

$$Q_i^{k+1} - Q_i^k = -\frac{\Delta t}{\Delta x}(F_{i+1/2}^k - F_{i-1/2}^k); \quad F_{i-1/2}^k = (c_{i-1/2}^k)_+ Q_{i-1}^k + (c_{i-1/2}^k)_- Q_i^k.$$

Remark 26.1.

1. If $y(x,t)$ is smooth, $y(x_i,t_k)$ agrees with Q_i^k up to $O(\Delta x^2)$.
2. Since we have zero-flux BCs, $F_{-1/2}^k \equiv 0,\ \forall k$, and $F_{N_x+1/2}^k \equiv 0,\ \forall k$.
3. FVM: enforces conservation of quantities at discretized level. Takes advanteage of arbitrary meshes (Integrals).
4. For FVM with 1st-order unpaired scheme (intorduced earlier),

$$
\begin{aligned}
Q_i^{k+1} &= Q_i^k - \frac{\Delta t}{\Delta x}(F_{i+1/2}^k - F_{i-1/2}^k),\\
&= Q_i^k - \frac{\Delta t}{\Delta x}((c_{i+1/2}^k)_+ Q_i^k + (c_{i+1/2}^k)_- Q_{i+1}^k - (c_{i+1/2}^k)_+ Q_{i-1}^k - (c_{i-1/2}^k)_- Q_i^k),
\end{aligned}
$$

$$
\begin{bmatrix} Q_0^{k+1} \\ Q_1^{k+1} \\ \vdots \\ Q_{Nx}^{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1-\frac{\Delta t}{\Delta x}(c_{1/2}^k)_+ & -\frac{\Delta t}{\Delta x}(c_{1/2}^k)_- & \\ \frac{\Delta t}{\Delta x}(c_{1/2}^k)_+ & 1-\frac{\Delta t}{\Delta x}(c_{3/2}^k)_+ + \frac{\Delta t}{\Delta x}(c_{1/2}^k)_- & -\frac{\Delta t}{\Delta x}(c_{3/2}^k)_- \\ 0 & \frac{\Delta t}{\Delta x}(c_{3/2}^k)_+ & \\ \vdots & 0 & \\ & \ddots & \\ 0 & & \end{bmatrix}}_{M\in\mathbb{R}^{(N_x+1)\times(N_x+1)}} \begin{bmatrix} Q_o^k \\ Q_1^k \\ \vdots \\ Q_{N_x}^k \end{bmatrix},
$$

$$\mathbf{Q}^{k+1} = M\mathbf{Q}^k.$$

   (a) Columns of M sum to 1 $\implies \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}\mathbf{Q}^{k+1} = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}\mathbf{Q}^k$.

   (b) If $2\frac{\Delta t}{\Delta x}(\max_{x,t} c(x,t)) < 1$, then every entry of M is ≥ 0. That is, together with 1, M is a Markov Matrix (column stochastic).

26.2 Monte Carlo Method (Integration)

Consider some arbitrary domain $\Omega \subseteq [0,1]^2$, what is its area?

$$\text{Area} = \int_\Omega 1\,\mathrm{d}x = \int\int_{[0,1]^2} \mathbb{I}_{\{x\in\Omega\}}\,\mathrm{d}x.$$

If we randomly drop a needle in $[0,1]^2$ (with no bias),

$$\mathbb{P}(\text{needle} \in \Omega) = \frac{\text{Area of }\Omega}{\text{Area of }[0,1]^2 = 1}.$$

To approximate this probablity more accurately, we drop 10^3 pins/needles,

$$\text{Area of }\Omega \approx \frac{\text{sum of pins landed in }\Omega}{10^3}.$$

Each "pin/needle" is a delta function $\delta(x-x_i)$. All 10^3 pins $\{x_i\}$ form a function $\frac{1}{10^3}\sum_{i=1}^{10^3}\delta(x-x_i)$.

A more complicated case is $A = \int \varphi(x)f(x)\,\mathrm{d}x$, where $f(x)$ is a probability function, ie, $f \geq 0$, $\int f(x)\,\mathrm{d}x = 1$. If we find N pins following distribution of $f(x)$, $\{x_i\}$ approx sampling $f(x)$, $\frac{1}{N}\sum_{i=1}^N \delta(x-x_i) \approx f(x)$. Then, $A = \int\varphi(x)f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^N \varphi(x_i)$. Only sampling is needed.

Based on the law of large numbers (almost sure of convergence),

$$\begin{aligned}\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\varphi(x_i) &= \int \varphi(x)f(x)\,\mathrm{d}x, \text{ with probability 1}\\ \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\varphi(x_i)\right] &= \int \varphi(x)f(x)\,\mathrm{d}x, \text{ (unbiased)}\\ \text{error } e_N[\varphi] &= \int \varphi f - \frac{1}{N}\sum_{i=1}^{N}\varphi(x_i) \approx \frac{\sigma}{\sqrt{N}}\nu,\end{aligned}$$

where $\nu \sim N(0,1)$ standard normal, and variate $\sigma = \int(\int \varphi f - \varphi(x))^2 f(x)\,\mathrm{d}x$. Therefore, Monte Carlo method has $O\left(1/\sqrt{N}\right)$ order of error. (A half order, not so good).

In terms of quadrature rule notations, **nodes** $\{x_i\}$ randomly chosen iid follor $f(x)$, **weights** $1/N$ for all nodes.

Back to Conservation law

$$\begin{cases}\partial_t y + \partial_x(c(x,t)y) = 0, \; x\in(a,b), t>0\\ y = 0 \text{ at } x = a,b, \; y(x,0) = g(x)\end{cases}.$$

One can multiply both sides by $\varphi(x)$ and integrate over x

$$\partial_t(\int_x \varphi y) - \int \varphi_x c y = 0.$$

If $y = \delta(x-\hat{x}(t)$, then $\partial_t\varphi(\hat{x}(t)) - \varphi_x(\hat{x}(t))c(\hat{x}(t),t) = 0$. The solution is

$$\frac{\mathrm{d}\hat{x}(t)}{\mathrm{d}t} = c(\hat{x}(t),t) \implies y(x,t) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(x-\hat{x}_i(t)).$$

27 Course Review (11/28/23)

27.1 Tools

1. Iterative methods. $x_{n+1} = G(x_n)$.
   (a) fixed-point iterations
   (b) bisection method
   (c) Newton's method and generalizations

   Application: root-finding, solving PDEs
2. Interpolation method: $\{(x_i, y_i = f(x_i))\}_{i=0}^{N}$
   (a) Lagrange, Hermite interpolation
   (b) Spline (piecewise-polynomial) interpolation
   (c) linear: Fouier basis, wavelet basis. nonlinear: neural network

   Applications: analyzing data, generation, prediction, semi-Lagrangian method

3. Integration $\int_\Omega f(x)\,\mathrm{d}x$
   (a) Newton-Cotes (equally-spaced)
   (b) Composite Newton-Cotes (trapezoid $O(1/N^2)$, Simpson $(O(1/N^4))$
   (c) Gauss-Quadrature $(n+1 \implies 2n+1)$ nodes weights to roots of orthogonal polynomial
   (d) Monte Carlo integration $w_i = 1/N$. $\int_\Omega f(x)|\Omega|\frac{1}{|\Omega|}\,\mathrm{d}x$. $\frac{1}{|\Omega|}\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^N \delta(x-x_i)$. The integral is approximated as $\frac{|\Omega|}{N}\sum_{i=1}^N f(x_i)$ with $x_1,\ldots,x_N$ uniformly distributed over Ω. The error is $O(1/\sqrt{N})$. The sqrt is from the large of numbers.

4. Linear space V.
   (a) Inner product $\implies$ norms.
   (b) Best 2-norm approximation. Optimality condition $Ax = b$.
   (c) Orthogonal basis of V, (wrt which inner product)
   (d) System of orthogonal polynomials. Any system wrt any weight can form a basis for $\mathcal{P}_n$.
   (e) Orthogonal projection.

27.2 Problems

(solving PDEs numerically—this is 50% of computational math)

1. IVP (ODE) $\dot{x} = f(x,t),\ x(0) = x_0$. for $x \in \mathbb{R}^d$.
   (a) f is Lipschitz continuous in x for uniqueness.
   (b) Higher-order ODEs can always be transformed into a system of 1st-order ODEs.
   (c) Explicit vs implicit, one-step vs linear multi-step method (LMM), and Runge-Kutta methods (RK4 most popular) is a one-step method and can be both explicit or implicit.
   (d) Analyze Local Truncation Error (LTE) vs relationship to Global Error. LTE $\to 0$ is consistency. Stability + consistency $\implies$ convergence. (This stability is zero-stability). Convergence means as mesh size $h \to 0$, the answer will converge to the true solution. A-stability (domain of absolute stability) cares about h not so small, does the solution make sense (forward-Euler is not A-stable if not $h < 1/L_\Phi$).

2. BVP $L[u] = 0$.

$$\text{Dirichlet}\begin{cases} u = \alpha & \text{at } x = a \\ u = \beta & \text{at } x = b \end{cases}, \quad \text{Neumann}\begin{cases} u_x = \alpha & \text{at } x = a \\ u_x = \beta & \text{at } x = b \end{cases}$$

   and other possible mixed conditions. We have FDM, Galerkin (FEM, galerkin with linear spline), Spectral method, collocation method (BVP to IVP to root-finding), shooting method. Can do LTE for FDM.

$$\text{LTE} = \text{LHS} - \text{RHS}|_{y_i \approx y(x_i)}$$

   replace with true solution.

3. PDEs (time-dependent) Heat equation, wave equation, transport equation, continuity equation. Usually,

(a) discretize spacial domain 1st, to become an IVP problem for a vector $x(t) \in \mathbb{R}^d$.

(b) Solve IVP $\implies$ Forward Euler (multiplication), Backward Euler (backward solve). Remember forward has CFL condition but backward does not.

For heat eqn with constant coeff, with $u(t = 0, x) = f(x)$, then $u(t, x) = f(x - ct)$. If have nonconstant coefficients, can either start at $t = 0$ and go forward or strat at t and go backwards. Starting position is regular grid, but where you end up is not regular (from method of characteristics). This then requires interpolation.

Conservation law. Retains positivity if IC $g(x) > 0$. Finite Volume Method, upwind scheme.