4.3 Newton interpolation:

Recall Lagrange interpolation:

$$p(x) = \sum_{i=0}^{n} \alpha_i L_i^n(x)$$

such that $L_i^n(x_j) = \delta_{ij}$

consequence: collocation matrix $\Phi = Id$

thus $\alpha_i = p_i$

The idea of Newton interpolation is to use basis functions $P_0(x), \ldots, P_n(x)$ with $P_i(x)$ being of degree i with roots at $x_0, \ldots, x_{i-1}$:

I.e. $P_0(x) = 1$

$$P_i(x) = (x-x_0)(x-x_1)\cdots(x-x_{i-1})$$

Consequently, the collocation matrix is lower triangul[ar]

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \vdots & * & 0 & 0 & \vdots \\ \vdots & \vdots & * & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & * & * & \cdots & * \end{bmatrix}$$

Example 34: (cf. Example 33)

x_i	0	1	2
p_i	2	4	3

- 1 point interpolation:

$p(x_0) = p_0$, degree$(p) = 0$

then $p(x) = \alpha P_0(x) = \alpha 1 = \alpha \stackrel{!}{=} p_0$

so $\boxed{p(x_0) = p_0 = 2.}$

In Newton interpolation we introduce specia[l] notation: For α we write $p[x_0]$, so $p(x) = p[x_0]P_0$

Here: $p[u_0] = 2$, so $\boxed{p(u) = 2 \cdot 1 = 2}$.

- Now: 2 point interpolation:

The weights are denoted $p[u_0]$ and $p[u_0 u_1]$. So:

Find $p(u) = p[u_0] P_0(u) + p[u_0 u_1] P_1(u)$

such that $\left.\begin{array}{l} p(u_0) = p_0 \\ p(u_1) = p_1 \end{array}\right\}$ interpolation conditions.

The first condition yields

$$p_0 = p[u_0] P_0(u_0) + p[u_0 u_1] \underbrace{P_1(u_0)}_{=0 \text{ by construction}}$$

$$= p[u_0] P_0(u_0)$$

But this is fulfilled due to the 1-point interpol.

The 2^{nd} interpolation condition yields:

$$p_1 = p(u_1) = p[u_0] P_0(u_1) + p[u_0 u_1] P_1(u_1)$$

$$\Leftrightarrow \quad p_1 = p[u_0] \cdot 1 \quad + p[u_0 u_1](u_1 - u_0)$$

$$\Rightarrow \quad 4 = 2 \cdot 1 \quad + p[u_0 u_1](1-0)$$

$$\Leftrightarrow \quad p[u_0 u_1] = \frac{4-2}{1-0} = \underline{\underline{2}}$$

In summary:

$$p(u) = 2 + 2 P_1(u) = \boxed{2 + 2u}$$

is the 2 point interpolation polynomial.

- 3 point interpolation:

Find $p(u) = p[u_0] P_0(u) + p[u_0 u_1] P_1(u) + p[u_0 u_1 u_2] P_2(u)$

Here, the first and 2^{nd} term have been determined by the 1 point and 2 point interpol.

Then $p(u) = 2 + 2u + p[u_0 u_1 u_2] P_2(u)$

Thus, $p(u) = 2 + 2u + p \ldots$

$$= 2 + 2u + p[u_0 u_1 u_2](u-u_0)(u-u_1)$$

Use the 3^{rd} interpolation condition to solve f...

$p[u_0 u_1 u_2]$: $p(u_2) = p_2$

So:
$$p_2 = 2 + 2u_2 + p[u_0 u_1 u_2](u_2-u_0)(u_2-u_1)$$

$$\Rightarrow \quad 3 = 2 + 2\cdot 2 + p[u_0 u_1 u_2](2-0)(2-1)$$

$$\Leftrightarrow \quad \frac{3-6}{2} = p[u_0 u_1 u_2]$$

To summarize: We get the interpolation poly

$$p(u) = 2 + 2u - \frac{3}{2} u(u-1)$$

$$\Leftrightarrow \quad p(u) = 2P_0(u) + 2P_1(u) - \frac{3}{2} P_2(u)$$

The resulting polynomial is the same as for Lagrange interpolation and for the basis $1, x, x^2, \ldots$

The weights $p[u_0 \ldots u_i]$ are obtained by back substitut... in the lower triangular matrix.
We obtain (proof: no credit exercise)

$$p[u_0 \ldots u_i] = \sum_{k=0}^{i} \frac{p_k}{\prod_{\substack{j=0 \\ j \neq k}}^{i} (u_k - u_j)}$$

which can be computed by the following recursion

$$p[u_0] = p_0$$

$$p[u_0 u_1] = \frac{p_0}{u_0 - u_1} + \frac{p_1}{u_1 - u_0}$$

$$\ldots \quad p[u_1] - p[u_0] \ldots$$

$$= \frac{p_1 - p_0}{u_1 - u_0} = \frac{p[u_1] - p[u_0]}{u_1 - u_0}$$

and in general

$$\boxed{p[u_0 \dots u_i] = \frac{p[u_1, \dots, u_i] - p[u_0 \dots u_{i-1}]}{u_i - u_0}}$$

The coefficients are called <u>divided differences</u> of order i at nodes $u_0 \dots u_i$.

<u>Example 35:</u> (cf. Example 34)

u_i	p_i	$p[u_i]$	$p[u_i u_{i+1}]$	$p[u_i u_{i+1} u_{i+2}]$
0	2	$\boxed{2}$	$\frac{4-2}{1-0} = \boxed{2}$	$\frac{-1-2}{2-0} = \boxed{-\frac{3}{2}}$
1	4	4	$\frac{3-4}{2-1} = -1$	
2	3	3		

The resulting coefficients result as the top row of this scheme, so $2, 2, -3/2$.

Note that the final divided differences are invariant under permutations of the nodes.

4.4 Error analysis:

All of the previous approaches lead to the same polynomial. Just the algorithm differs.

<u>Theorem 36:</u> Let $f \in C^{n+1}([a,b])$ and $p(u)$ a polynomial of degree $\leq n$ that interpolates f at $n+1$ distinct nodes $u_0, \dots, u_n \in [a,b]$. Then for any $u \in [a,b]$ there exists a $\xi \in (a,b)$ such that

$$f(u) - p(u) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (u - u_i)$$

$(n+1)!$ $\quad i=0$

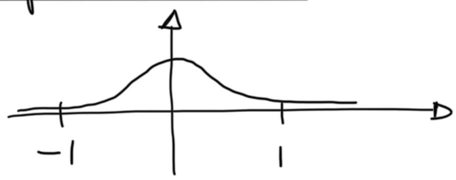
For equidistant nodes $x_i = a + i \cdot \frac{b-a}{n}$ we have

$$|f(x) - p(x)| \le \frac{1}{4(n+1)} M \left(\frac{b-a}{n}\right)^{n+1}$$

where $M = \max_{[a,b]} |f^{(n+1)}(x)|$

Example 37: Runge's phenomenon

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



then, for equidistant nodes:

$$\lim_{n \to \infty} \max_{[-1,1]} |f(x) - p(x)| = \infty$$

The interpolation polynomials oscillate and don't provide meaningful interpolations.

- Higher order polynomial interpolation on equidistant nodes can be problematic
- Use non-equidistant points, e.g.

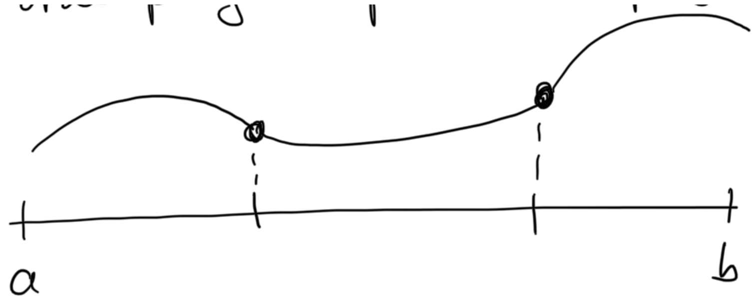
  Chebyshev nodes $\cos\left[\left(\frac{i}{n}\right)\pi\right]$, $i = 0, \dots, n$

  over $[-1, 1]$.

4.5 Piecewise polynomial interpolation

Another remedy of the problems of high order poly interpolation is piecewise polynomial interpolation.

Subdivide $[a,b]$ into smaller pieces on which lower-order poly interpolation is performed.



<u>Problem encountered</u>: Pieces of interpolation fit together continuously but not differentiable.

<u>Solution</u>: Use Hermite interpolation in which additional constraints for continuity of derivatives is used (see homework #4)