ME2 Computing- Session 3: Numerical Interpolation

Learning outcomes:

- Being able to compute Lagrangian numerical interpolation
- Being able to compute Newton numerical interpolation
- Being able to interpolate with splines

Before you start:

In your H drive create a folder *H:\ME2MCP\Session3* and work within it.

Task A: Lagrangian polynomials and interpolation

1. Write a function, *Lagrangian*, to compute the Lagrangian polynomial *j* for a set of known nodes *xn*, at an interpolated point *xp*.
   The function receives the values *j, xp* and the array of nodes *xn*, and returns the value:
   $$L_j(x_p) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{(x_p - x_k)}{(x_j - x_k)}$$

2. Write a function, *LagrInterp*, that receives the sets of know values, *xn* and *yn*, the points to be interpolated *x*, and returns the interpolated values *y*, by using Lagrangian polynomials.

3. Test the two functions above with $f(x) = \sin(x)$ over the range *x = [0:3]* with step *0.05*, given the nodal values at:
   a) *xn = [1:2]* with 2 nodes: linear interpolation $p_1(x)$
   b) *xn = [1:2]* with 3 nodes: quadratic interpolation $p_2(x)$
   c) *xn = [1:2]* with 4 nodes: cubic interpolation $p_3(x)$

   Compare/plot the interpolating polynomials, $p_1(x), p_2(x), p_3(x)$ with/against those calculated manually in slides 110, 111 and 112, respectively. (You should end up with a plot like in slide 114).

4. **Error analysis**: compute the basic error (as defined in slide 116) for $p_1(x), p_2(x), \dots\dots, p_{13}(x), p_{14}(x)$ at x = π/2 (slide 123).

Task B: Newton interpolation

1. Write a recursive function, *NewtDivDiff,* to compute the value of the Newton's Divided Difference $f[x_0, x_1, x_2, \dots, x_N]$, as defined in slide 128. The function receives the two lists of nodal points *xn* and *yn* and returns the corresponding scalar value.

2. Write a function, *NewtonInterp*, that receives the sets of know values, *xn* and *yn*, the points to be interpolated *x*, and returns the interpolated values *y*, by using Newton's method.

3. Test the two functions above with $f(x) = \sin(x)$ over the range *x = [0:3]* with step *0.05*, given the nodal values at:
    a) *xn = [1:2]* with 2 nodes: linear interpolation $p_1(x)$
    b) *xn = [1:2]* with 3 nodes: quadratic interpolation $p_2(x)$
    c) *xn = [1:2]* with 4 nodes: cubic interpolation $p_3(x)$

    Compare/plot the interpolating polynomials, $p_1(x), p_2(x), p_3(x)$ with/against those calculated with Lagrangian interpolation.

4. Interpolate the function (slide 142):

$$f(x) = \frac{1}{1 + 25x^2}$$

    in the range $-1 \leq x \leq 1$, with Newton's interpolation of order n = 1, 2, 3, 4, 5, … 14 and plot the interpolating polynomials (Runge's phenomenon).

Task C: Splines

1. Write a function, *Splines*, that receives the sets of know values, *xn* and *yn*, the points to be interpolated *x*, the clamped boundary conditions $y'(a)$, $y'(b)$, and returns the interpolated values *y*, by using cubic splines, with

2. Test the function above with:

$$f(x) = \frac{1}{1 + 25x^2}$$

    with $a = -1, b = 1, y'(a) = 0.074$, $y'(b) = -0.074$, by using 3, 5 and 11 nodes.

    Note: to invert the matrix you can use the function *MyGauss* you wrote in Computing 1.

Task D: Two-dimensional interpolation

1. Read the image *Flower.jpg* and plot it.

2. Shrink the image into a new image, *n* time smaller, and save it into a new file, *Shrunk.jpg*.

3. Resize the image, *m* time larger, by using the bilinear interpolation (slide 158), starting from the shrunk image, *Shrunk.jpg*.