MATH-131 (Numerical Methods for Scientists and Engineers) — Worksheet 9

Semester: Spring 2019, Instructor: Nicholas Knight

Due Apr. 9 at 2359. *Please remember to cite your sources, including collaborators.*

Deliverable: Submit a Live script titled `worksheet9.mlx` via CatCourses (under Assignments). Divide this file into sections, one for each of the following questions, plus an extra (final) section containing all the function definitions. Document each function definition to explain the input and output arguments. Also document key portions of the algorithm to make it clear you understand how your code works.

1. Approximate the derivative of each of the following functions using the forward, backward, and centered difference formulas on the grid `linspace(-5,5,100)`.

$$f'(x) \approx \begin{cases} \frac{f(x+h)-f(x)}{h} & \text{forward,} \\ \frac{f(x)-f(x-h)}{h} & \text{backward,} \\ \frac{f(x+h)-f(x-h)}{2h} & \text{centered.} \end{cases}$$

   For each part, make a single plot (with three curves) showing the absolute error at each grid point. (Note that the approximations are undefined at one or both endpoints.) Also state which approximations are exact (within roundoff error).

   (a) $f : x \mapsto x$.

   (b) $f : x \mapsto x^2$.

   (c) $f : x \mapsto x^3$.

   (d) $f : x \mapsto \sin(x)$.

2. Use the centered difference formula to approximate $f'(x)$ for $f = \sin$ and $x = 1$, using `h` $= 1, 1/10, 1/100, \ldots, 1/10^{15}$. Plot the absolute error. Explain the behavior as h decreases. (*Hint:* Read the last subsection of §4.1 concerning roundoff error.) Why does this instability not arise with $x = 0$?

3. For any function f and $n+1$ distinct nodes $x_0, \ldots, x_n$, Lagrange interpolation factors $f = P + R$ where

$$P\colon x \mapsto \sum_{i=0}^{n} f(x_i) \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

and R is the approximation error. The following Matlab function,

```
function a = linterp_poly(X, Y)
a = zeros(1,numel(X));
for i = 1:numel(X)
    aa = poly(X([1:i-1 i+1:end]));
    a = a + Y(i) * aa / polyval(aa, X(i));
end
end
```

represents $P(x) = \sum_{k=0}^{n} a_k x^k$ as its coefficients `a` $= [a_n, a_{n-1}, \ldots, a_1, a_0]$ (this solves Worksheet 7 Q2.)

(a) Write a Matlab function that uses P to approximate $f^{(d)}(x)$ for any positive integer d and real number x. The function signature should be

```
function y = num_diff(X, Y, x, d)
```

and should invoke both `linterp_poly` (see above) and Matlab's `polyder`. Test your code with by constructing a problem where you expect the answer to be exact (up to rounding errors). What happens when d equals or exceeds the number of given interpolation nodes $(n+1)$?

(b) *Bonus (not graded):* Reimplement `linterp_poly` using your own versions of `poly` and `polyval`.

(c) *Bonus (not graded):* Repeat Part (a) but avoid converting P to its "monomial basis" representation, $[a_n, a_{n-1}, \ldots, a_1, a_0]$. Instead, take advantage of the recursive formulas known for the derivatives of Lagrange polynomials; see (e.g.) `https://en.wikipedia.org/wiki/Lagrange_polynomial#Derivatives`.

4. *Bonus; not graded):* Write a Matlab function that constructs finite difference formulas for a given stencil and derivative order. That is, given points $x_0, x_1, \ldots, x_n$ and a positive integer d, your code should compute $c_0, c_1, \ldots, c_n$ such that

$$f^{(d)}(x) \approx \sum_{i=0}^{n} c_i f(x + x_i).$$

The function signature should be

```
function C = fdcoeff(X, d)
```

where `X` is a numeric array containing the (distinct) stencil points, `d` is the desired derivative order, and `C` is an array of stencil coefficients. For example, `fdcoeff([-1, 0, 1], 1)` should return `[-0.5, 0, 0.5]` (the 3-point midpoint formula). Note that the textbook does not give you a formula: you will have to research it.

Hint: You will need to construct and solve a linear system. You are encouraged to follow the example at `http://web.media.mit.edu/~crtaylor/calculator.html`.