Math 270b: Numerical Approximation and Nonlinear Equations

Instructor: Randolph E. Bank

Winter Quarter 2020

Homework Assignment #8
Due Friday, February 28, 2020

Exercise 8.1. In this computational exercise, we will compare the following interpolation schemes:

- Lagrange Interpolation with uniform knots
- Lagrange Interpolation with Chebyshev knots
- $\mathcal{C}^0$ Piecewise Linear Interpolation with uniform knots
- $\mathcal{C}^1$ Piecewise Cubic Interpolation with uniform knots
- $\mathcal{C}^2$ Piecewise Cubic Interpolation with uniform knots

Use Runge's function

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

on the interval $-1 \leq x \leq 1$. Use $n = 5, 11, 21, 41$ knots. Approximate the $\mathcal{L}^\infty$ norm of the error using a uniform mesh of 201 points, i.e.

$$p_i = -1 + \frac{i-1}{100}$$

for $1 \leq i \leq 201$ and

$$||e||_\infty \approx \max_{1 \leq i \leq 201} |e(p_i)|$$

Graph some interesting examples. For the case of Lagrange interpolation with uniform knots, you should observe Runge's phenomenon. This should disappear when you use Chebyshev knots, although you may see some (nonsymmetrical) roundoff error for large values on n. You should be able to observe second order convergence for piecewise linear interpolation, and fourth order convergence for piecewise cubic interpolation.

You should write s simple procedure for computing a divided difference table and evaluating the corresponding polynomial for a given value of x. For the piecewise polynomial cases, you need a procedure for finding the interval containing a given value of x, and then you can evaluate the linear or cubic polynomial on that interval using your divided difference/polynomial evaluation procedures. Since we are using uniform meshes for the piecewise polynomial cases, the value of $(x - x_1)/h$, truncated to an integer, can be used to determine the interval containing x. For the case of the cubic spline, you can use algorithms *spline* and *seval* below, which take advantage of the uniformity of the mesh to simplify the construction and evaluation of the cubic spline.

Algorithm *spline* takes as input, x_{min} and x_{max}, the endpoints of the interval, n, the number of knots, and the vector y of function values. It returns the vectors z and y for use in *seval*. The vector d is a temporary array.

```
            algorithm spline
$h$       $\leftarrow$  $(x_{max} - x_{min})/(n-1)$
$z_1$     $\leftarrow$  $0$
$d_2$     $\leftarrow$  $4$
$z_2$     $\leftarrow$  $y_3 - 2y_2 + y_1$
$d_i$     $\leftarrow$  $4 - 1/d_{i-1}$                              for $i = 3, n-1$
$z_i$     $\leftarrow$  $y_{i+1} - 2y_i + y_{i-1} - z_{i-1}/d_{i-1}$   for $i = 3, n-1$
$z_{n-1}$ $\leftarrow$  $z_{n-1}/d_{n-1}$
$y_{n-1}$ $\leftarrow$  $y_{n-1} - z_{n-1}$
$z_i$     $\leftarrow$  $(z_i - z_{i+1})/d_i$                        for $i = n-2, 2, -1$
$y_i$     $\leftarrow$  $y_i - z_i$                                  for $i = n-2, 2, -1$
$z_n$     $\leftarrow$  $0$
```

Algorithm *seval* evaluates the spline at the point x, given the arrays y and z computed by algorithm *spline*. We assume here that $x_{min} \le x \le x_{max}$.

```
            algorithm seval
$h$        $\leftarrow$  $(x_{max} - x_{min})/(n-1)$
$i$        $=$  $\min[\text{int}\{(x - x_{min})/h\} + 1, n-1]$
$\theta$   $\leftarrow$  $(x - x_{min})/h - (i-1)$
$seval$    $=$  $z_i(1-\theta)^3 + z_{i+1}\theta^3 + y_i(1-\theta) + y_{i+1}\theta$
```