Math 151A Homework #4

Nathan Solomon

November 1, 2024

Problem 0.1.

First, we can use $P_{0,1}$ and $P_{1,2}$ to find $P_{0,2}$:

$$P_{0,2}(x) = \frac{(x-0)P_{1,2}(x) - (x-2)P_{0,1}(x)}{2-0} = \frac{x(3x-1) - (x-2)(x+1)}{2} = \frac{2x^2+2}{2} = x^2+1.$$

Using the same recursive formula, we get

$$P_{0,3}(1.5) = \frac{(1.5-0)P_{1,3}(1.5) - (1.5-3)P_{0,2}(1.5)}{3} = \frac{P_{1,3}(1.5) + P_{0,2}(1.5)}{2} = \frac{4+3.25}{2} = 3.625$$

Problem 0.2.

$$f(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1)$$

Since $f(x_0) = 0$, we have $a_0 = 0$. Then $f(x_1) = \sin(\pi/4) = 1/\sqrt{2}$, so $a_1 = f(x_1)/(x_1 - x_2) = (1/\sqrt{2})/(\pi/4) = 2\sqrt{2}/\pi$.

Therefore $f(x_2) = 1 = (2\sqrt{2}/\pi) \cdot (\pi/2) + a_2(\pi^2/8)$, which simplifies to

$$a_2 = \frac{8}{\pi^2}\left(1-\sqrt{2}\right),$$

so the interpolating polynomial is

$$f(x) = \frac{2\sqrt{2}}{\pi}x + \frac{8-8\sqrt{2}}{\pi^2}\left(x-\frac{\pi}{4}\right)x.$$

Problem 0.3.

$$f[x_0,x_1,x_2] = -\frac{3}{2} = \frac{f[x_1,x_2] - f[x_0,x_1]}{x_2-x_0} = \frac{f[x_1,x_2]-5}{2-0},$$

so $f[x_1,x_2] = 2$.

$$f[x_0,x_1] = 5 = \frac{f[x_1]-f[x_0]}{x_1-x_0} = \frac{f[x_1]+1}{1},$$

so $f[x_1] = 4$.

$$f[x_1,x_2] = 2 = \frac{f[x_2]-f[x_1]}{x_2-x_1} = \frac{f[x_2]-4}{1},$$

so $f[x_2] = 6$.

Problem 0.4.

For the error bound to be less than 10^{-6}, we need at least $n+1=7$ nodes.

```
>>> from math import sin
>>> def err_bound(n):
...     M = 1 if n%2==0 else sin(1)
...     return M * n**(-1-n) / 4 / (n+1)
...
>>> for i in range(1, 10):
...     print(i, err_bound(i))
...
1 0.10518387310098706
2 0.010416666666666666
3 0.0006492831672900435
4 4.8828125e-05
5 2.2439226261543903e-06
6 1.2758018159252726e-07
7 4.5614702528754705e-09
8 2.06960572136773e-10
9 6.033288038733948e-12
```

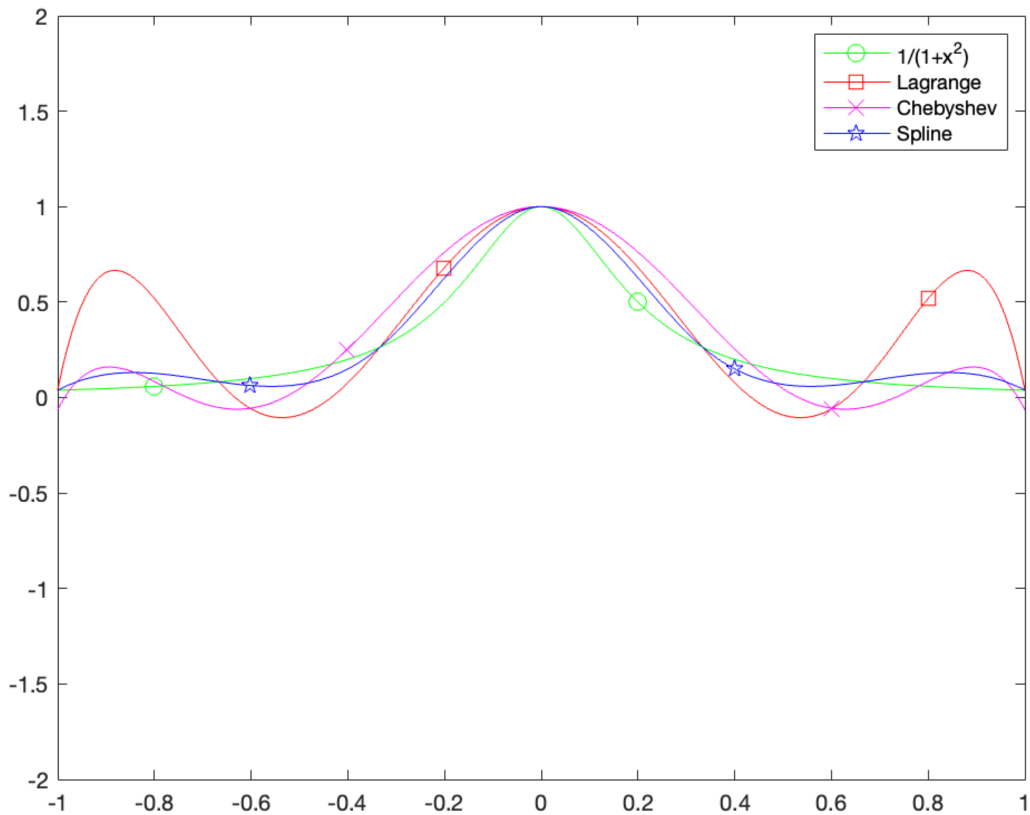
Problem 0.5.

The points x_0, x_1 are equally spaced in the interval $[x_0, x_1]$, so we can use the theorem for the error bound when nodes are equally spaced. Let $n = 1$, and let M be the maximum value of $\left|f^{(n+1)}(x)\right| = |f''(x)|$ on the interval $[x_0, x_1]$. Then on that same interval,

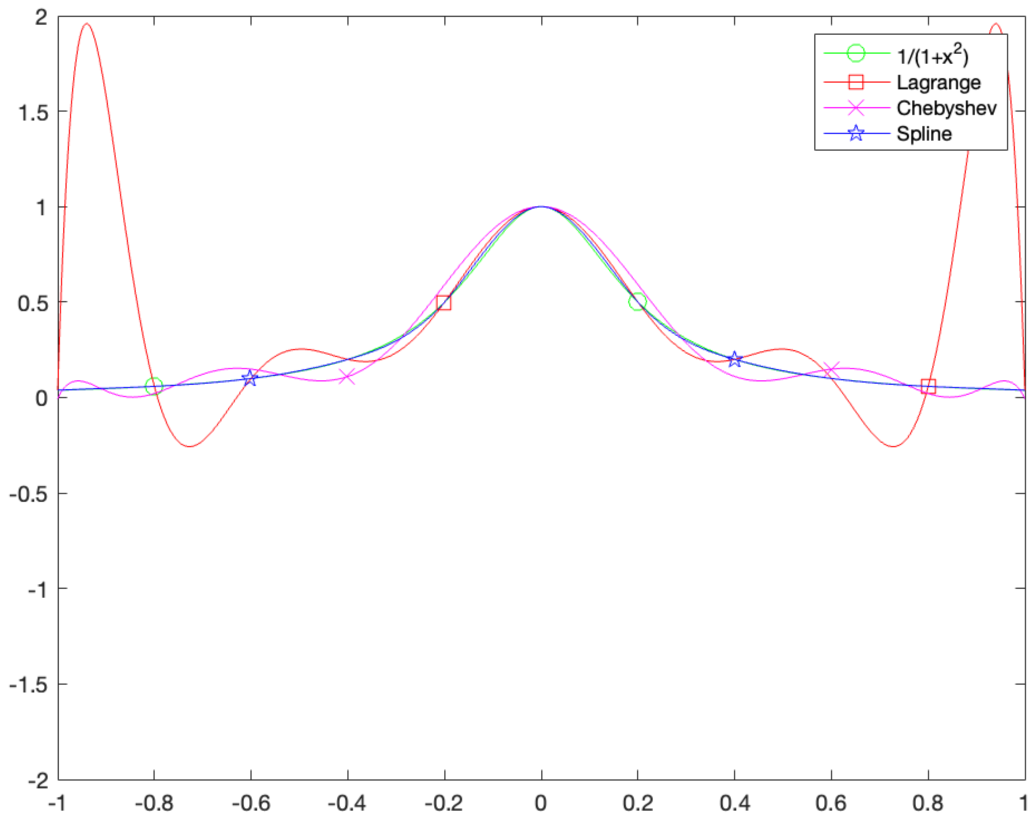
$$|f(x) - P(x)| \leq \frac{1}{4} \cdot \frac{M}{n+1} \cdot h^{n+1} = \frac{Mh^2}{8} = \frac{h^2}{8} \max_{x \in [x_0, x_1]} |f''(x)| \, .$$

Problem 0.6.

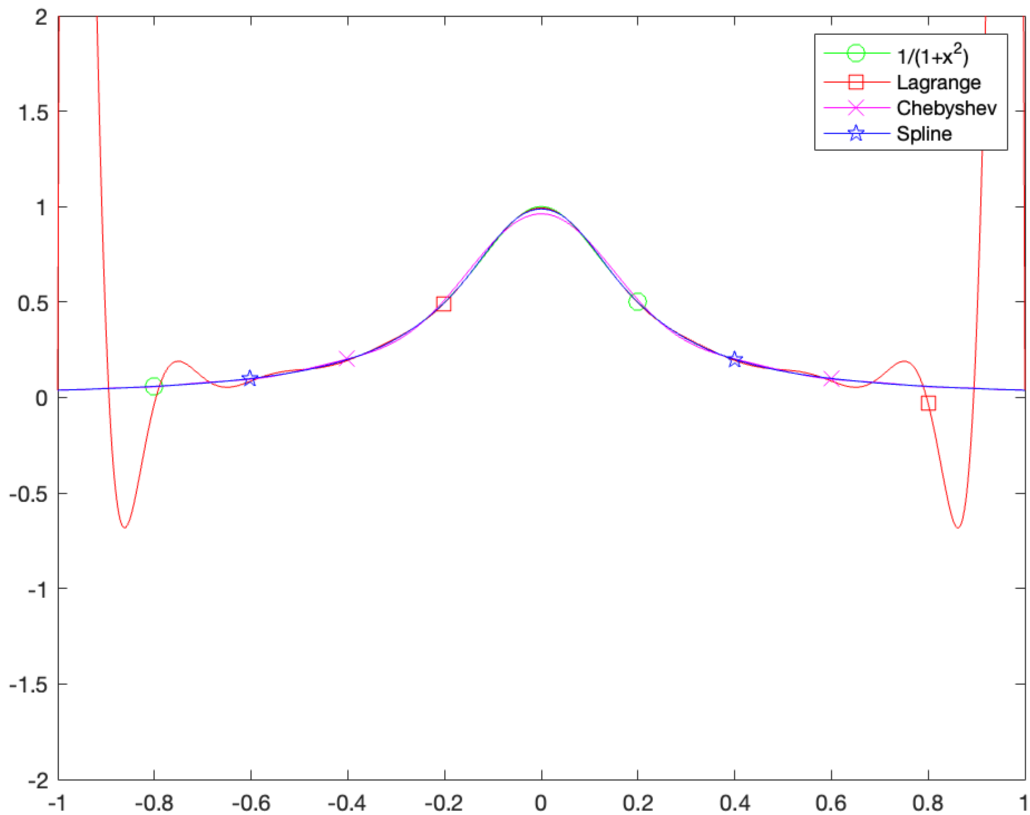
Here's $N = 7$. The spline is the best fit, although the Chebyshev interpolator isn't bad either.



Below is the graph for $N = 11$. Once again, the spline is the best fit. The Lagrange polynomial has gotten worse, and has pretty bad Runge phenomena.



Below is the graph for $N = 20$. The Chebyshev interpolator and spline fits are both very good – the spline looks slightly better, but I can't really tell. The Lagrange polynomial is good for x near zero, but very bad near $x = \pm 1$.




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Note: If the plot below does not contain all four functions,
%%%        the command window will spit out a warning: "Ignoring
%%%        extra legend entries".
%%%
%%%        The code will still run, but the legend label might not
%%%        correspond to the curve that's actually shown.
%%%
%%%        It's recommended
%%%        you comment out the 'legend' command below unless plotting
%%%        all four curves at once.
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N = 11;        % number of interpolation points
a = -1;        % left endpoint of interval
b = 1;         % right endpoint of interval


f = @(x) 1./(1.+25.*x.^2);      % create function 'handle'
                                % note the '.' after x in the quadratic
                                % term. If this is removed, code will break
```

```
xinterp = linspace(-1,1,N);       % equispaced interpolation points
yinterp = f(xinterp);             % values of f(x) at these points


Nplot = 1000;                     % number of evaluation/plotting points
xplot = linspace(-1,1,Nplot);     % xvals at which each function will be
                                  %    evaluated

ytrue = f(xplot);                 % values of the true f(x) to be plotted

%%%% get lagrange interpolant
ylagrange = lagrange(xplot,xinterp,yinterp);

%%%% get chebyshev interpolant
c = chebyshev_coefficients(a, b, N, f);
ycheb = chebyshev_interpolant(a, b, N, c, Nplot, xplot);

%%% get cubic spline interpolant
yspline = spline(xinterp, yinterp, xplot);
%                                     [this function is native to matlab and
%                                          doesn't require an extra .m file]


%%% plot each function from -1 to 1
plot(xplot,ytrue,'g-o','MarkerIndices', 100:500:length(xplot),'MarkerSize',8); hold on;
plot(xplot,ylagrange,'r-s', 'MarkerIndices', 400:500:length(xplot),'MarkerSize',8); hold o
plot(xplot,ycheb, 'm-x','MarkerIndices', 300:500:length(xplot),'MarkerSize',8); hold on;
plot(xplot,yspline,'b-p','MarkerIndices',200:500:length(xplot),'MarkerSize',8); hold on;
axis([-1,1,-2,2])
legend('1/(1+x^2)', 'Lagrange', 'Chebyshev', 'Spline') % only call me if all four curves a
```