MATH 151A PROJECT 2

Shiqi Liang and Yuchen Yao

February 2021

Function and Generating Data

The function we picked is $f(x) = x^2$ which is continuously defined on [-1,1] and easy to manage.

The pseudo code for our function would be

```
input x
output x*x
```

which is really straight forward.

The pseudo code to generate a list of interpolating x values to get a corresponding $f(x)$ values from is

```
given n
create empty list x to store all points
for j in 0 to n{
temp=(2j/n)-1
add temp to x
}
return x
```

The other set of interpolating x values are generated by a similar structure:

```
given n
create empty list x to store all points
for j in 0 to n{
temp=cos((pi/2)*((2j+1)/(n+1)))
add temp to x
}
return x
```

Similarly, pseudo code to generate a list of real points x_j given n is:

```
given n

create empty list c to store all results
```

```
for j in 0 to 5n{
    temp=(2j/5n)-1
    add temp to list c
}

return list c
```

We thus implement this via a function in matlab. Please see appendix for the actual matlab code.

Lagrange Interpolation Matlab Representation

From class, we know the general form of the Lagrange's interpolation polynomial is represented as

$$P(x) = \sum_{i=1}^{n+1} y_i (\prod_{j=1, j\neq 1}^{n+1} \frac{(x - x_j)}{(x_i - x_j)})$$

This is a combination of two general formula which is

$$P(x) = \sum_{i=1}^{n+1} y_i l_i(x)$$

and

$$l_i(x) = \prod_{j=1, j\neq 1}^{n+1} \frac{(x - x_j)}{(x_i - x_j)}$$

Thus, the pseudocode for this lagrange interpolation representation is

```
given x, y and xp
n=length(x)-1
sum=0
for i = 1: n+1
    pr=1
    for j = 1:n+1
        if j does not equal to i
        pr=pr*((xp-x(j))/(x(i)-x(j))

        end
    end
sum=sum+y(i)*pr
end
output sum
```

See the appendix for our lagrange interpolation function code.

An easy way of checking whether the function is working correctly is to plug in the original x values. Since we know that interpolating those value would yield $f(x)$, this is an easy way to check performance.

1 Results

1.1 Method (i)

Uniform interpolation nodes $x_j = \frac{2j}{n} - 1$ for $j = 0, 1, ..., n$

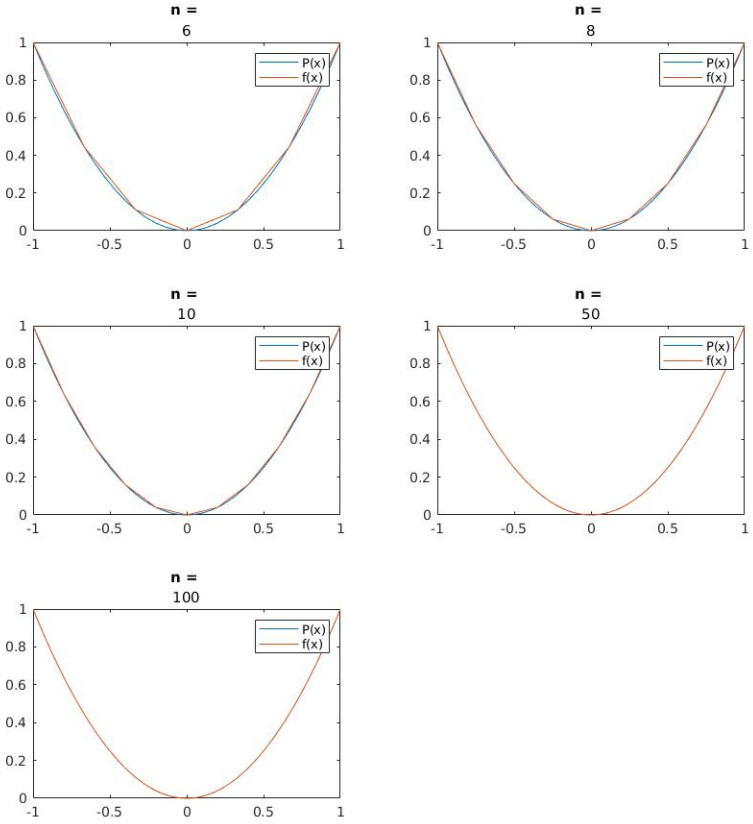

Figure 1: plots of P(x) vs. f(x)

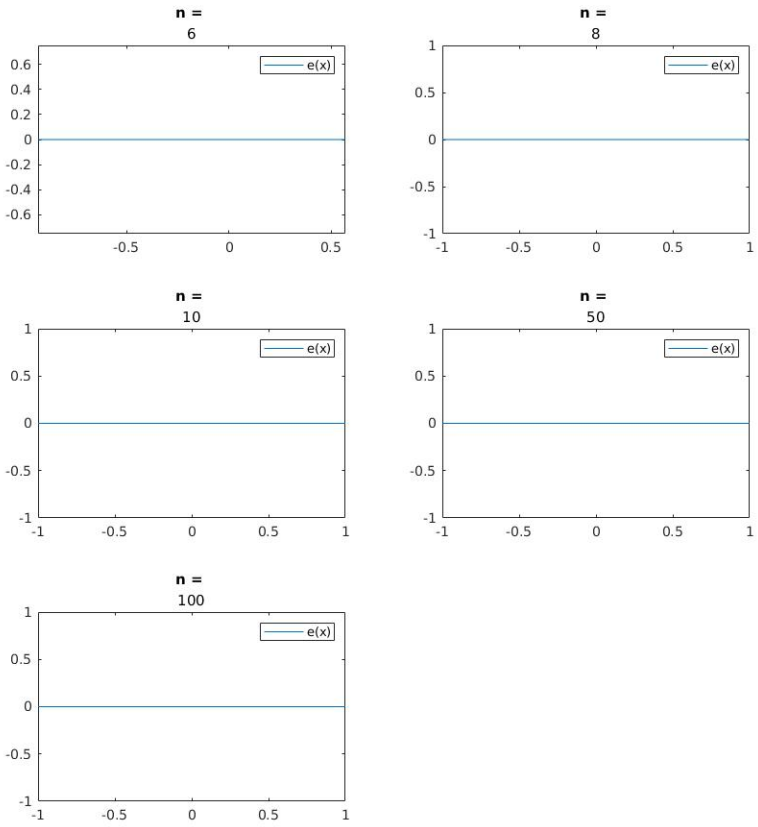

Figure 2: plots of errors $e(x) = \left| f(x) - P_n(x) \right|$

1.2 Method (ii)

Polynomial $x_j = cos(\frac{\pi}{2}\frac{2j+1}{n+1})$ for $j = 0, 1, ..., n$

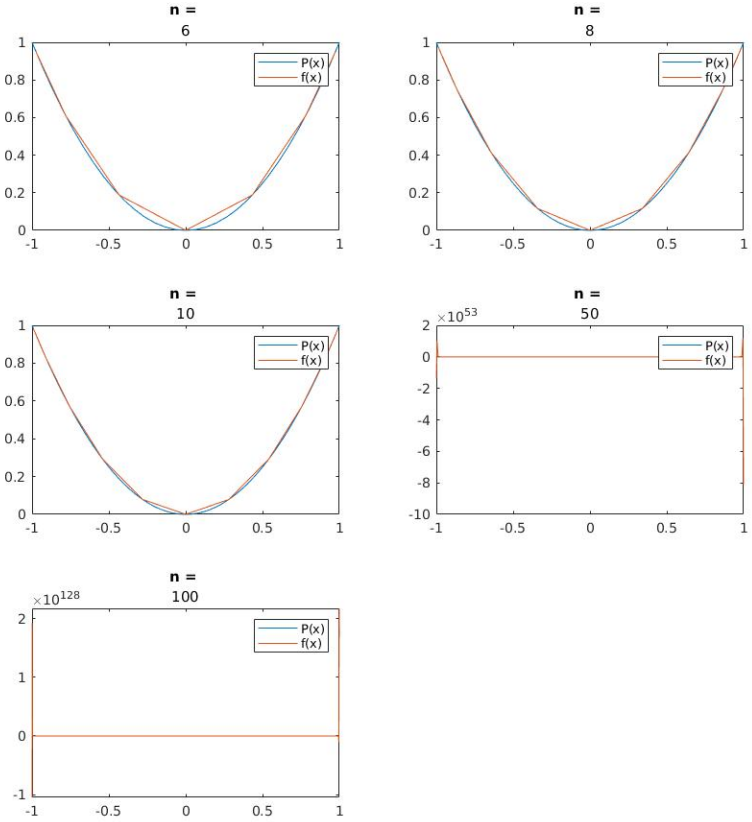

Figure 3: plots of P(x) vs. f(x)

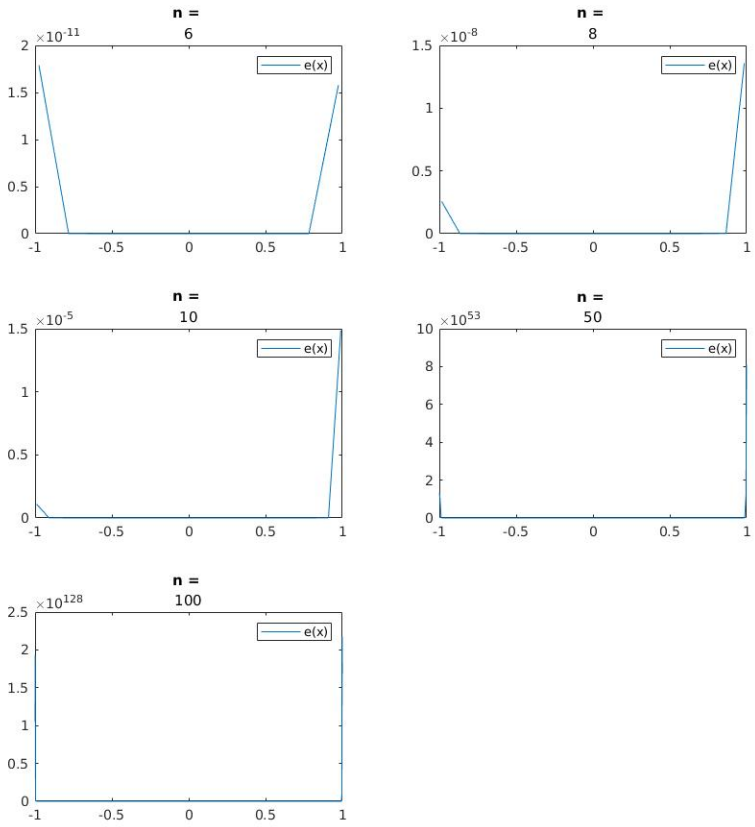

Figure 4: plots of errors $e(x) = \left|f(x) - P_n(x)\right|$

Discussion

From the above plots, we can see that method (i) and method (ii) performs similarly when x is near 0. Nevertheless, method (ii) performs poorly at end points.This indicates that the generation of points has an effect on the accuracy of the results. So generally speaking Lagrange interpolation method works well on our function, but for future research we could try out functions that are more complex.

Conclusion

We can see that Lagrange interpolation generally yields fairly accurate results. The accuracy of Lagrange interpolation also depends on how the reference points are generated. Method (i) is better when we want overall performance, method (ii) is good when we only care about values around $x = 0$, as it performs poorly on the edge cases.

Appendix: Matlab codes

code for real x generator

```
function [v] = genx(n)
    for j=0:(5*n)
        temp = ((2*j)/(5*n))-1;
        v(j+1)=temp;
    end
end
```

code for interpolating x generator

```
function [v] = geninterx(n)
    for j=0:n
        temp = ((2*j) / n) - 1;
        v(j+1)=temp;
    end
end
```

code for polynomial interpolating x generator

```
function [v] = gennewx(n)
    for j = 0:n
        temp = cos((pi/2)*((2*j+1)/(n+1)));
        v(j+1)=temp;
    end
end
```

code for the function f(x)

```
function [o] = sqx(x)
    o=x.* x;
end
```

Lagrange interpolation code

```
function [yp]=lagrange(xp,x,y)
    n=length(x)-1;
    sm=0;
    for i = 1:n+1
        pr=1;
        for j = 1:n+1
            if j ~=i
                pr=pr.*(xp-x(j))/(x(i)-x(j));
            end
        end
        sm=sm+y(i)*pr;
    end
    yp=sm;
end
```