Section: Approximating functions - Lecture 3

[3] Interpolation error

When interpolating a given function f by a polynomial of degree n at the nodes $x_0, \cdots, x_n$, it can be shown that we get the error

$$f(x) - p(x) = f[x_0, \cdots, x_n, x] \prod_{i=0}^{n} (x - x_i),$$

where $f[x_0, \cdots, x_n, x]$ is the notation for divided differences.

So in principle, to have a "best" interpolation we should choose nodes such that

$$P(x) = \prod_{i=0}^{n} (x - x_i)$$

is minimized, i.e., such that $\|P(x)\|_\infty$ is smallest.

In the following we use some examples to demonstrate the relation between the chosen nodes and the function $P(x)$. You will see that the Chebyshev nodes is the best.

Remark:

We'll use packages LinearAlgebra for solving the polynomial and PyPlot for plotting. Run `using Pkg; Pkg.add("PyPlot")` if you don't have PyPlot installed.

```julia
using PyPlot
```

We first define a function that plots the polynomial $P(x)$ for given nodes for later usage.

```julia
# a function that plots the corresponding polynomial with given nodes
function PlotPolynomial(x_nodes,labelstring="P(x)",l=1000::Int)
    # x: the nodes
    # l: length of the linsapce
    # labelstring: the label of the polynomial

    n=length(x_nodes);

    # construct the linspace
    xl = range(-1,stop=1,length=1000);

    # construct the polynomial
    p_nodes = xl .-x_nodes[1];
    for ii=1:n-1
        p_nodes = p_nodes.*(xl.-x_nodes[ii+1]);
    end

    # plot the polynomial
    plot(xl, p_nodes,label=labelstring);
    plot(x_nodes, zeros(n,1), "or");
    plt.legend();
    println("max. norm of "*labelstring*"=",maximum(abs.(p_nodes)));
end
```

```
PlotPolynomial (generic function with 3 methods)
```

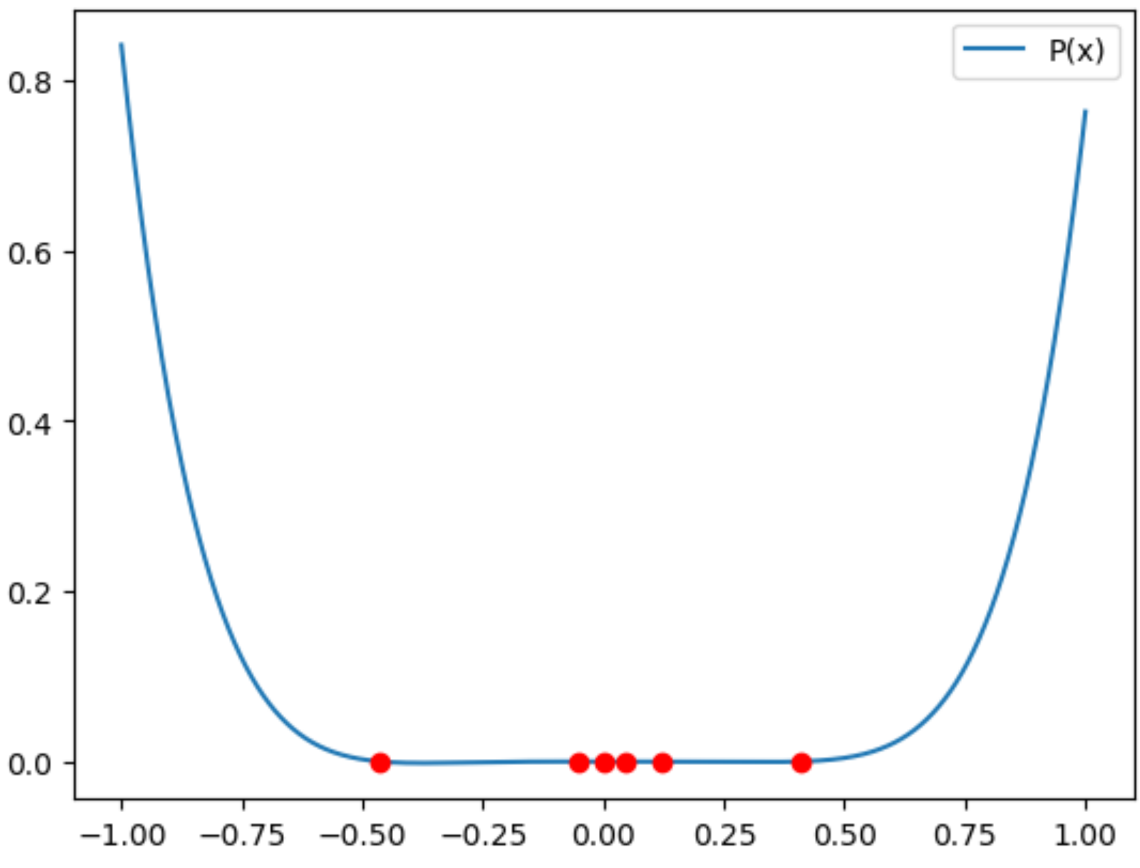
[3.1] Random Nodes

Example 1

We first illustrate what happened if we simply choose random nodes on $[-1, 1]$ to do the polynomial interpolation.

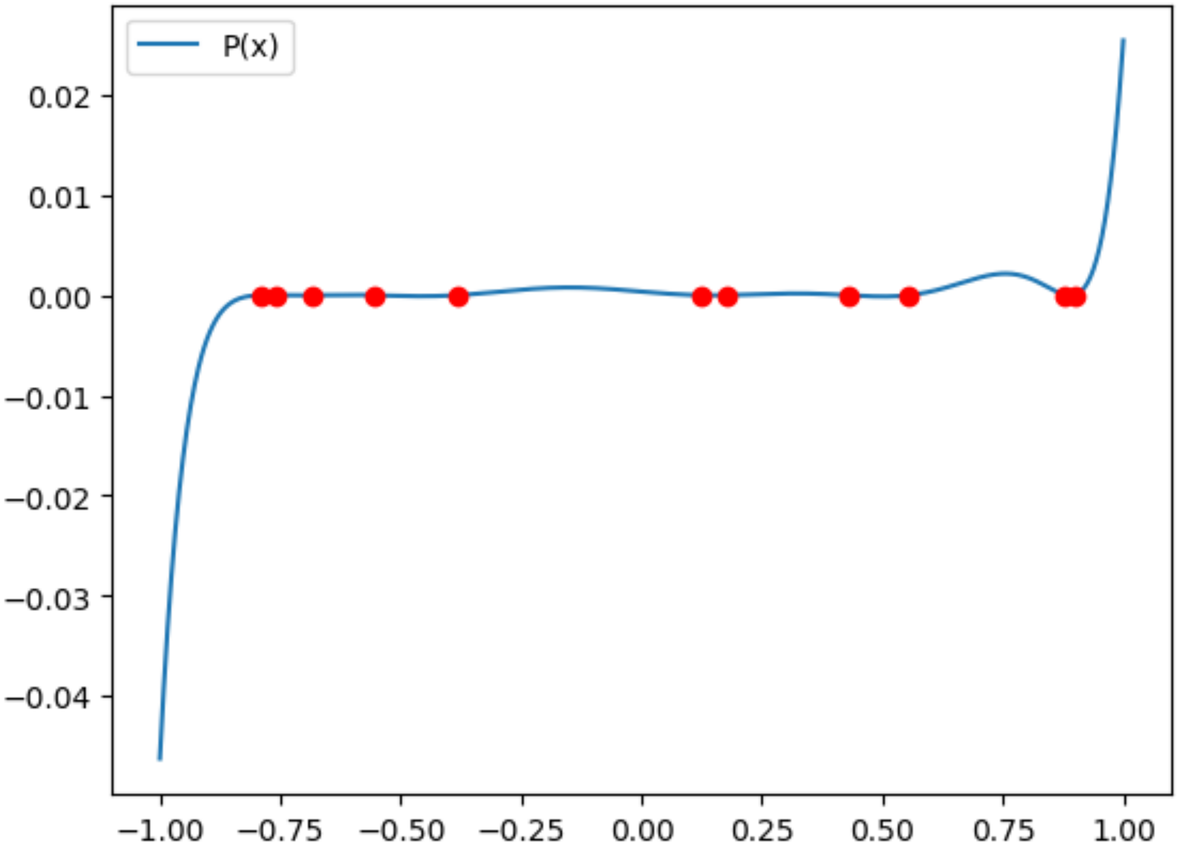
We plot the polynomial $P(x) = \prod_{i=0}^{n}(x - x_i)$ when choosing $n + 1$ random points on $[-1, 1]$ for $n = 5, 10$ and 20.

```julia
# 2*rand(n+1).-1 gives n+1 random nodes for integer n
n=5;
PlotPolynomial(2*rand(n+1).-1);
```



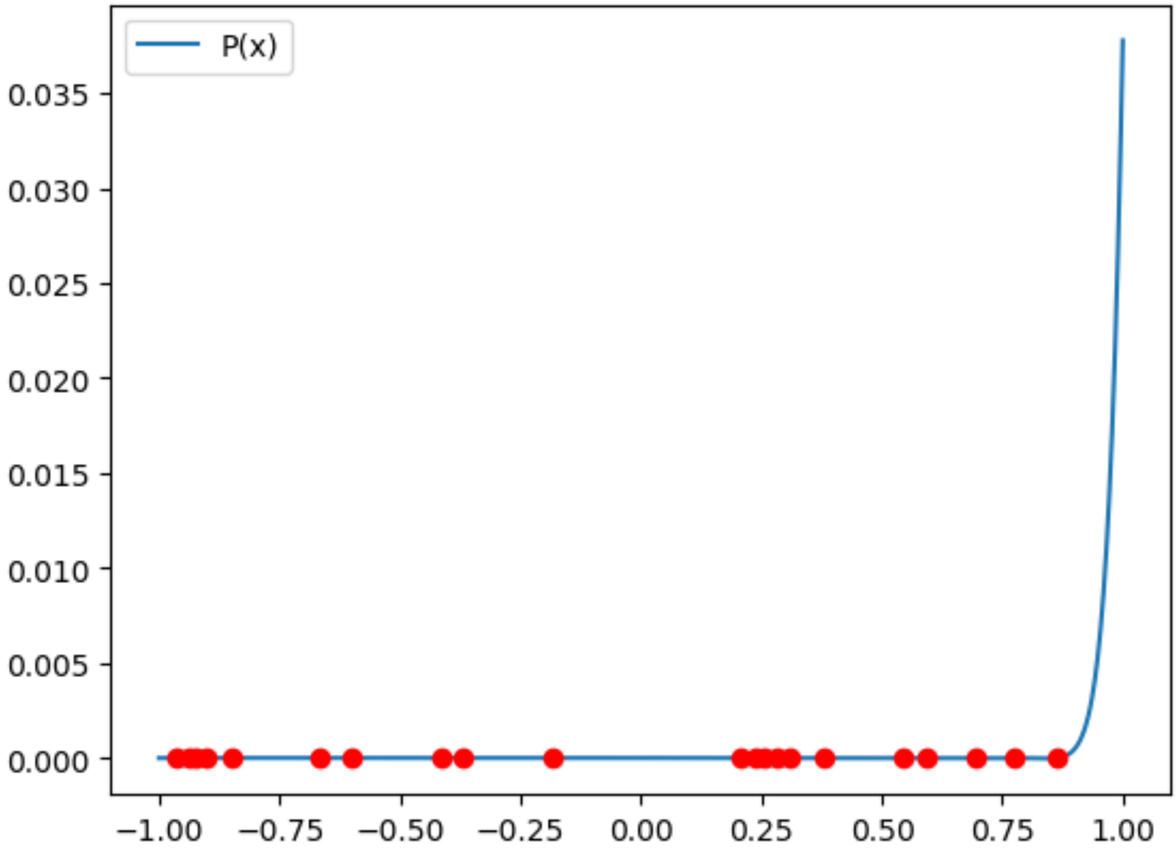

```
max. norm of P(x)=0.8415108366736999
```

```julia
n=10;
PlotPolynomial(2*rand(n+1).-1);
```




```
max. norm of P(x)=0.04628282859966022
```

```julia
n=20;
PlotPolynomial(2*rand(n+1).-1);
```




```
max. norm of P(x)=0.03777704716032644
```

[3.2] Equally spaced Nodes

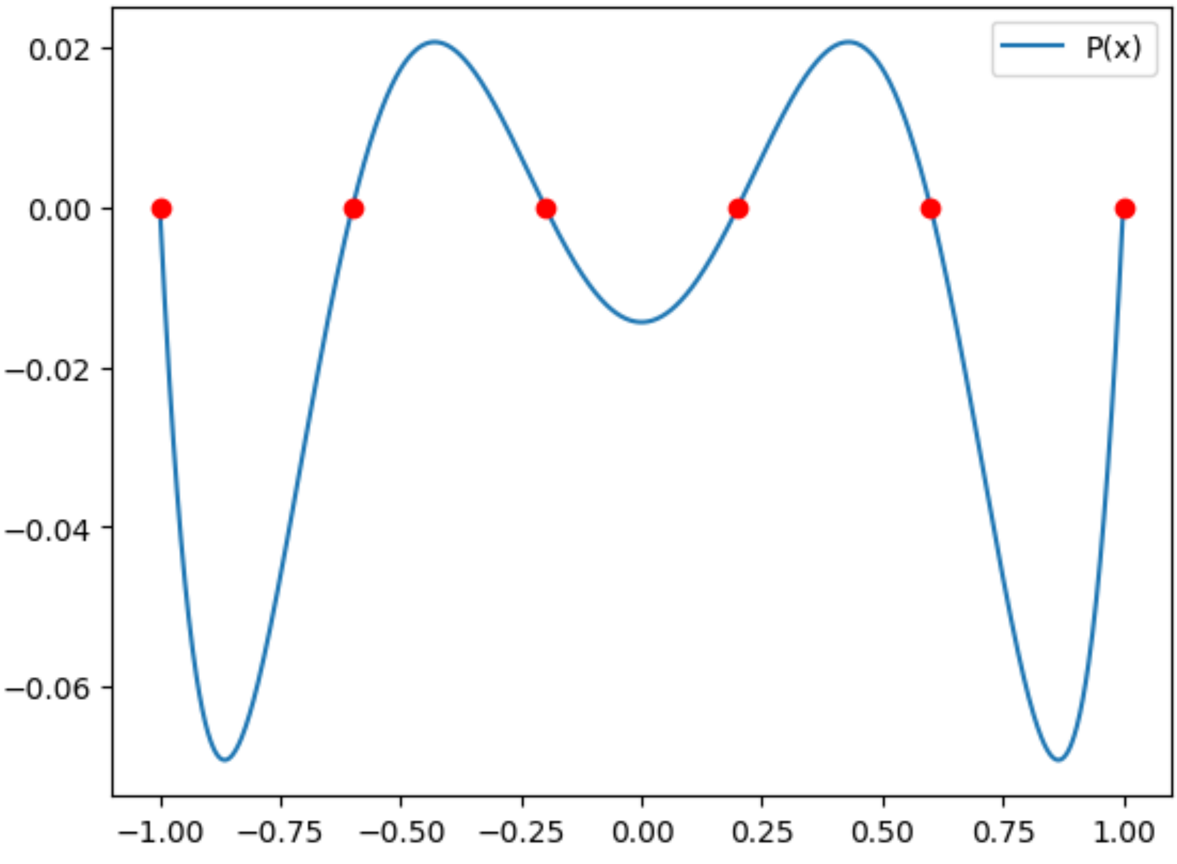
Now let's try uniformly distributed points on $[-1, 1]$:

$$x_k = -1 + 2\frac{k}{n}, \quad k = 0, 1, 2, \cdots, n,$$

Example 2

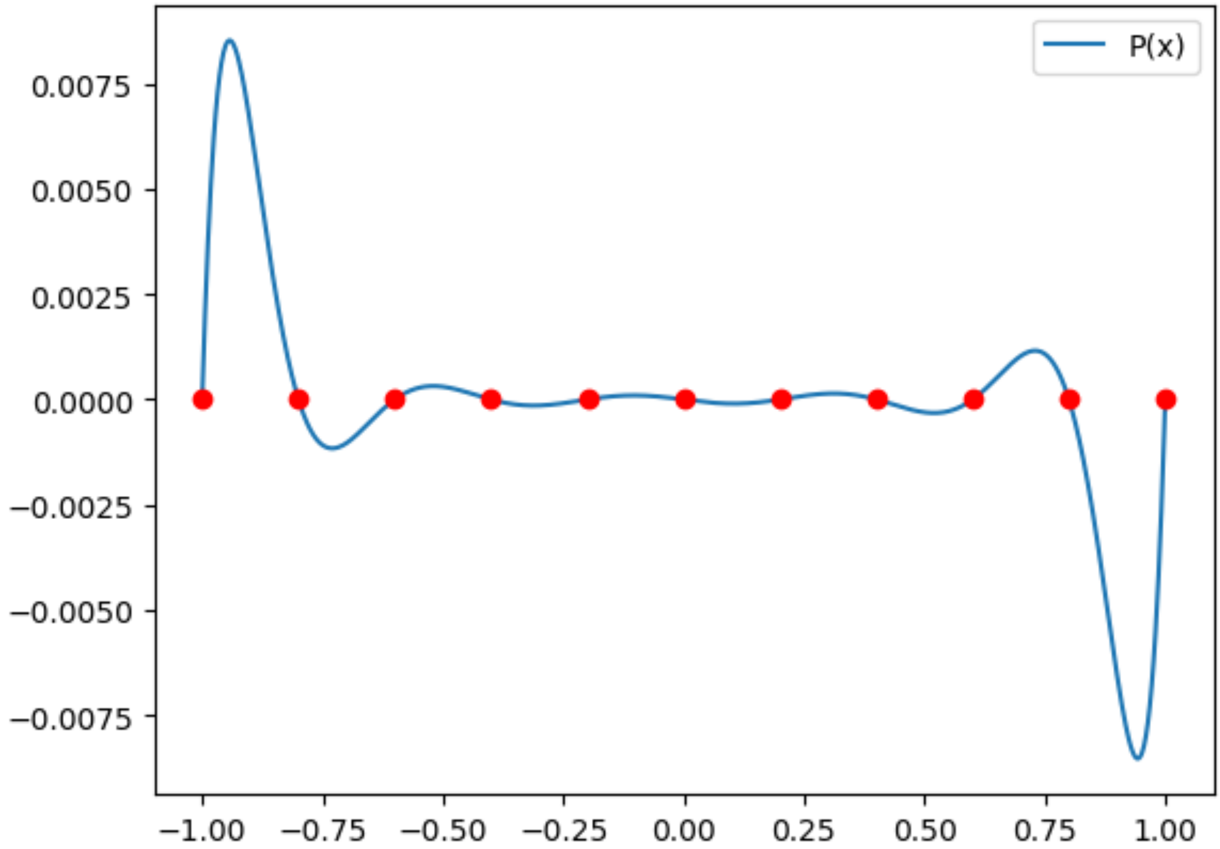
We would like to plot the polynomial $P(x) = \prod_{i=0}^{n}(x - x_i)$ when choose $n + 1$ equally spaced points on $[-1, 1]$ for $n = 5$, 10 and 20.

```julia
# 2*(0:n)/n .-1  gives n+1 equally spaced nodes for integer n
n=5;
PlotPolynomial(2*(0:n)/n .-1);
```



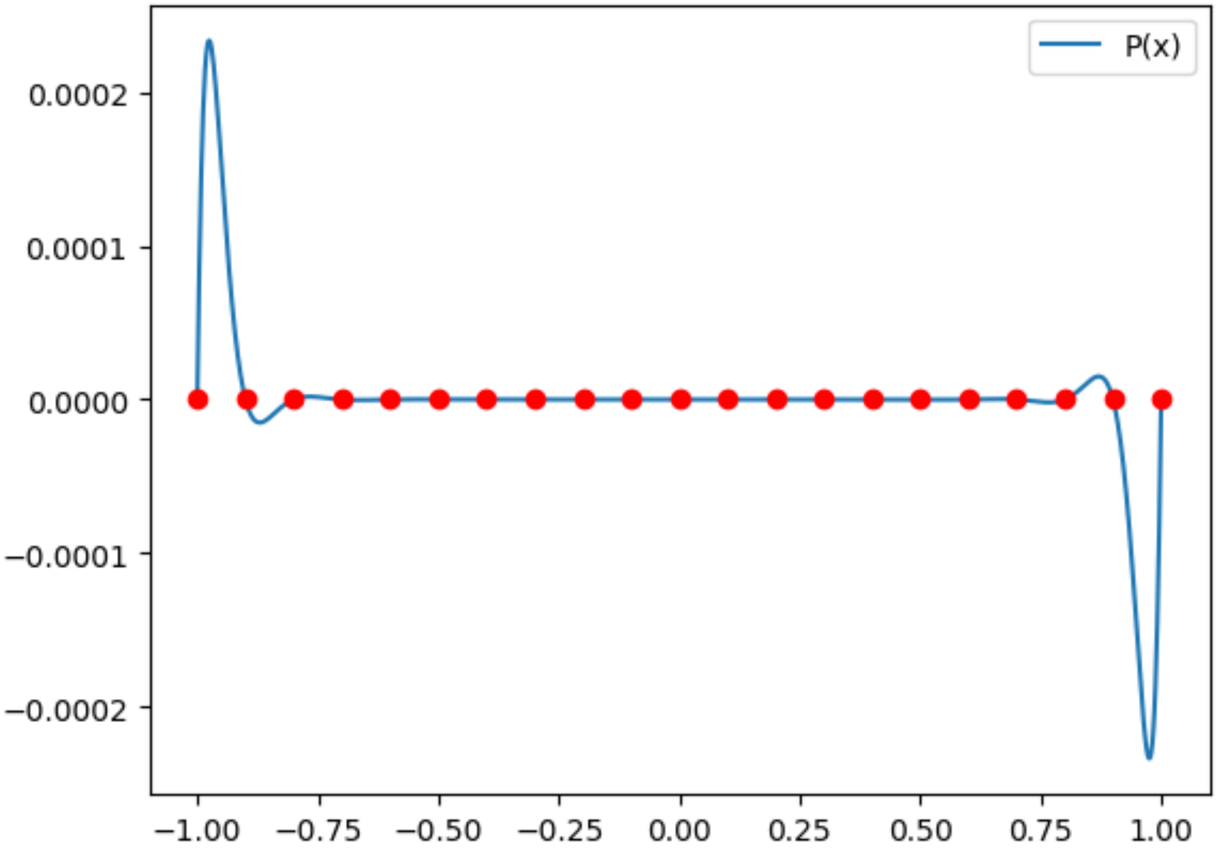

```
max. norm of P(x)=0.06922546010398747
```

```julia
n=10;
PlotPolynomial(2*(0:n)/n .-1);
```




```
max. norm of P(x)=0.008530768039898246
```

```julia
n=20;
PlotPolynomial(2*(0:n)/n .-1);
```




```
max. norm of P(x)=0.00023361682788821547
```

[3.3] Chebyshev Nodes of the first kind

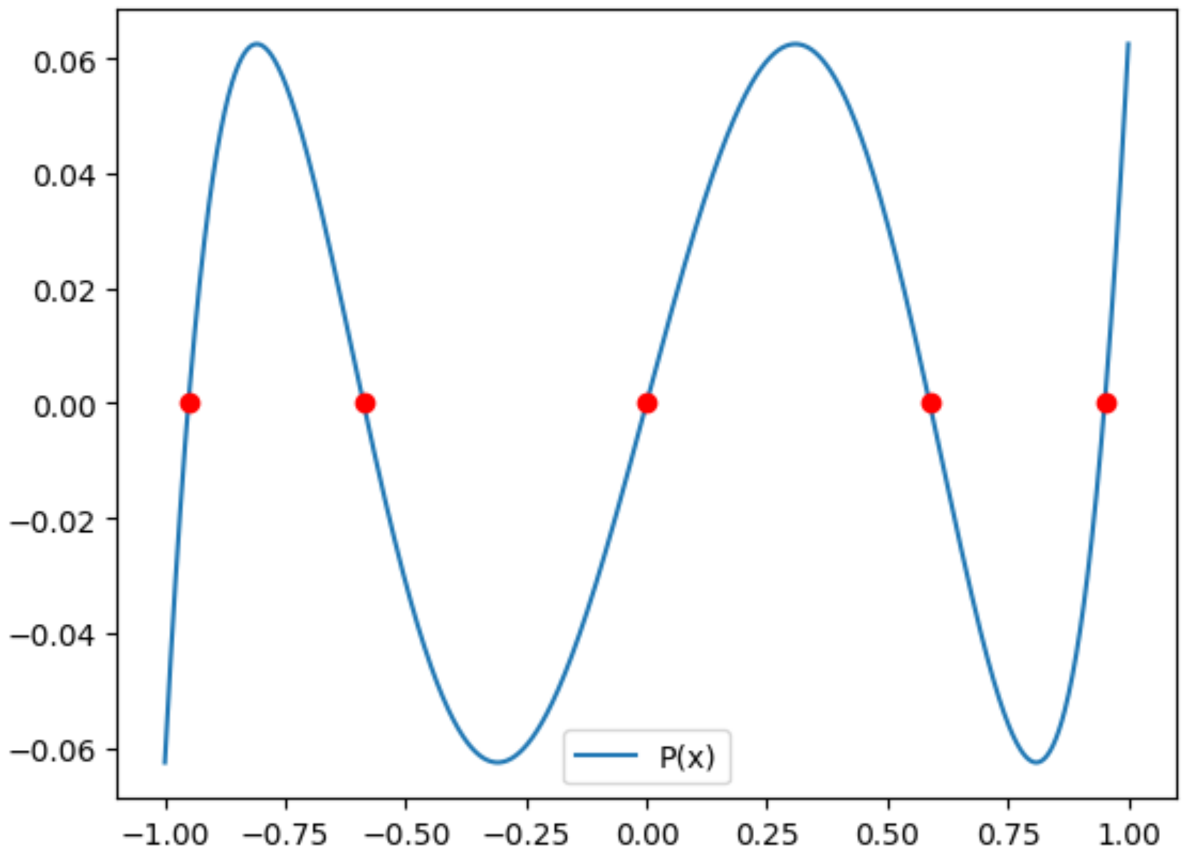
Consider chebychev nodes

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1, 2, \cdots, n.$$

Example 3

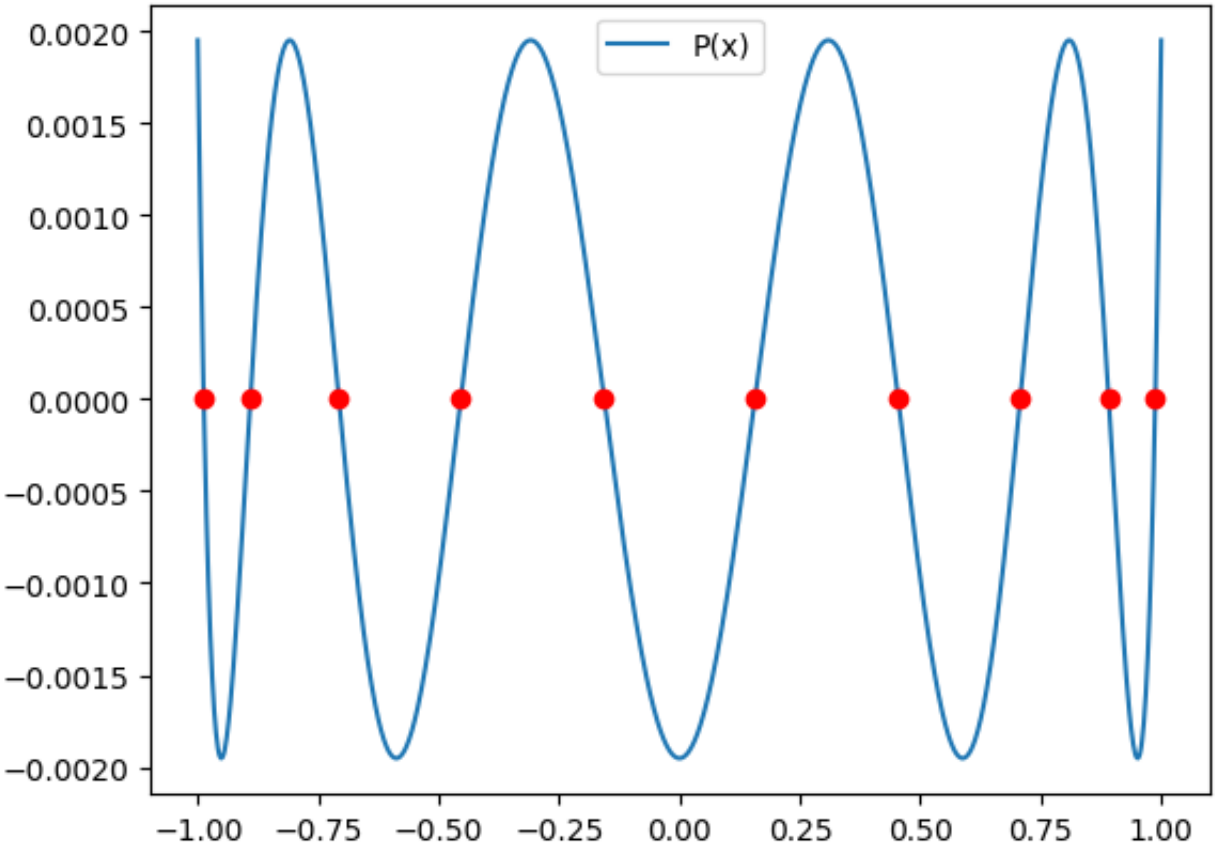
We would like to plot the polynomial $P(x) = \prod_{i=1}^{n}(x - x_i)$ when choose n Chebyshev Nodes of the first kind for $n = 5$, 10 and 20.

```julia
# cos.(((1:n).-0.5)*pi/n) gives n Chebyshev nodes of the first kind for integrer n
n=5;
PlotPolynomial(cos.(((1:n).-0.5)*pi/n));
```



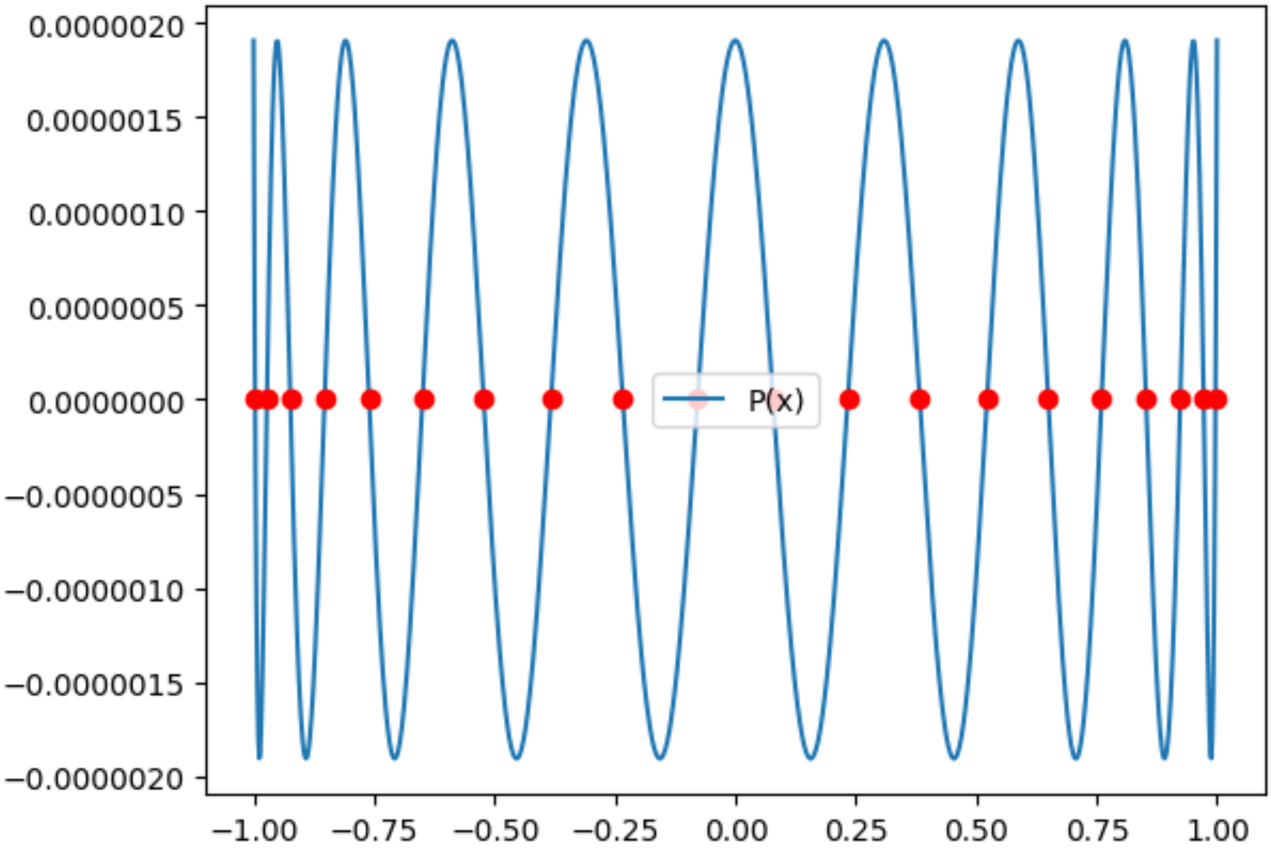

```
max. norm of P(x)=0.06250000000000008
```

```julia
n=10;
PlotPolynomial(cos.(((1:n).-0.5)*pi/n));
```




```
max. norm of P(x)=0.0019531250000000126
```

```julia
n=20;
PlotPolynomial(cos.(((1:n).-0.5)*pi/n));
```




```
max. norm of P(x)=1.907348632812514e-6
```

[3.35] Notes on Chebyshev Nodes of the first kind

Consider polynomial interpolation of some function f on $[-1, 1]$ using Chebyshev Nodes of the first kind:

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right) \quad k = 1, 2, \cdots, n.$$

Then, construct $p(x)$ by determinating the $n-1$ degree polynomial that has value $f(x_k)$ at x_k for each k. One can prove that

$$|f(x) - p(x)| \leq \frac{1}{2^{n-1} n!} \max_{\xi \in [-1,1]} |f^n(\xi)|,$$

which gives us small error as n become large.

For more details one can read "Chebyshev nodes" in wiki.

[3.4] Chebyshev Nodes of the second kind

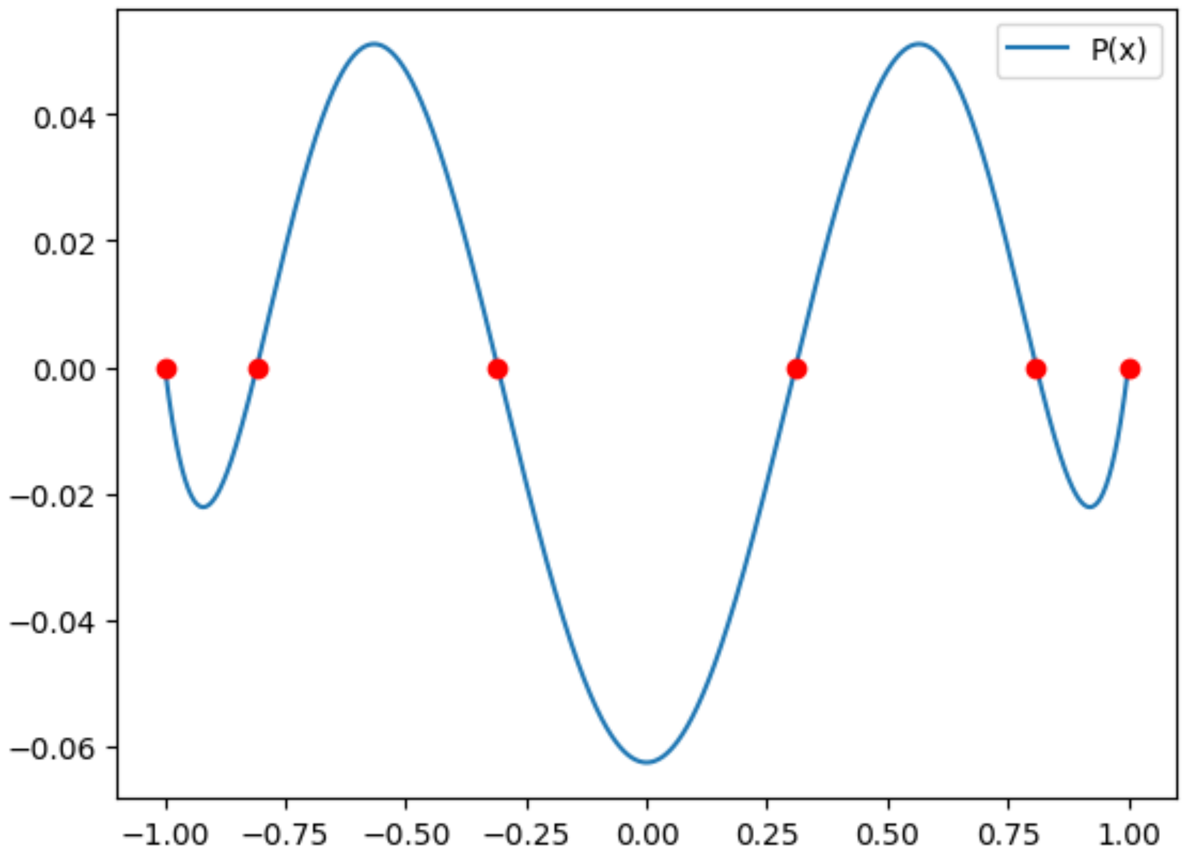
Consider chebychev nodes of the second kind:

$$x_k = \cos\left(\frac{k}{n}\pi\right), \qquad k = 0, 1, 2, \cdots, n.$$

Example 4

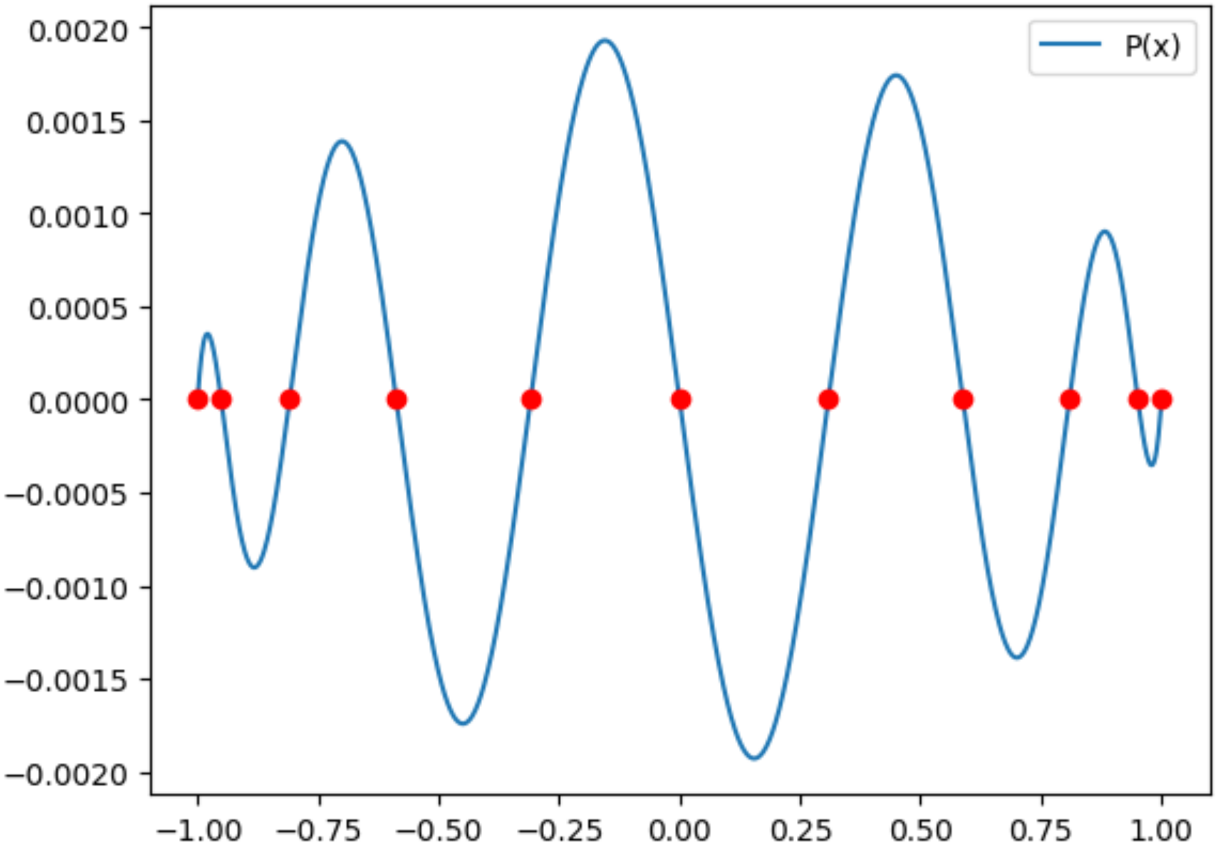
We would like to plot the polynomial $P(x) = \prod_{i=0}^{n}(x - x_i)$ when choose $n + 1$ Chebyshev Nodes of the second kind for $n = 5$, 10 and 20.

```julia
# cos.((0:n)*pi/n) gives the n+1 Chebyshev nodes of the second kind for integer n
n=5;
PlotPolynomial(cos.((0:n)*pi/n));
```



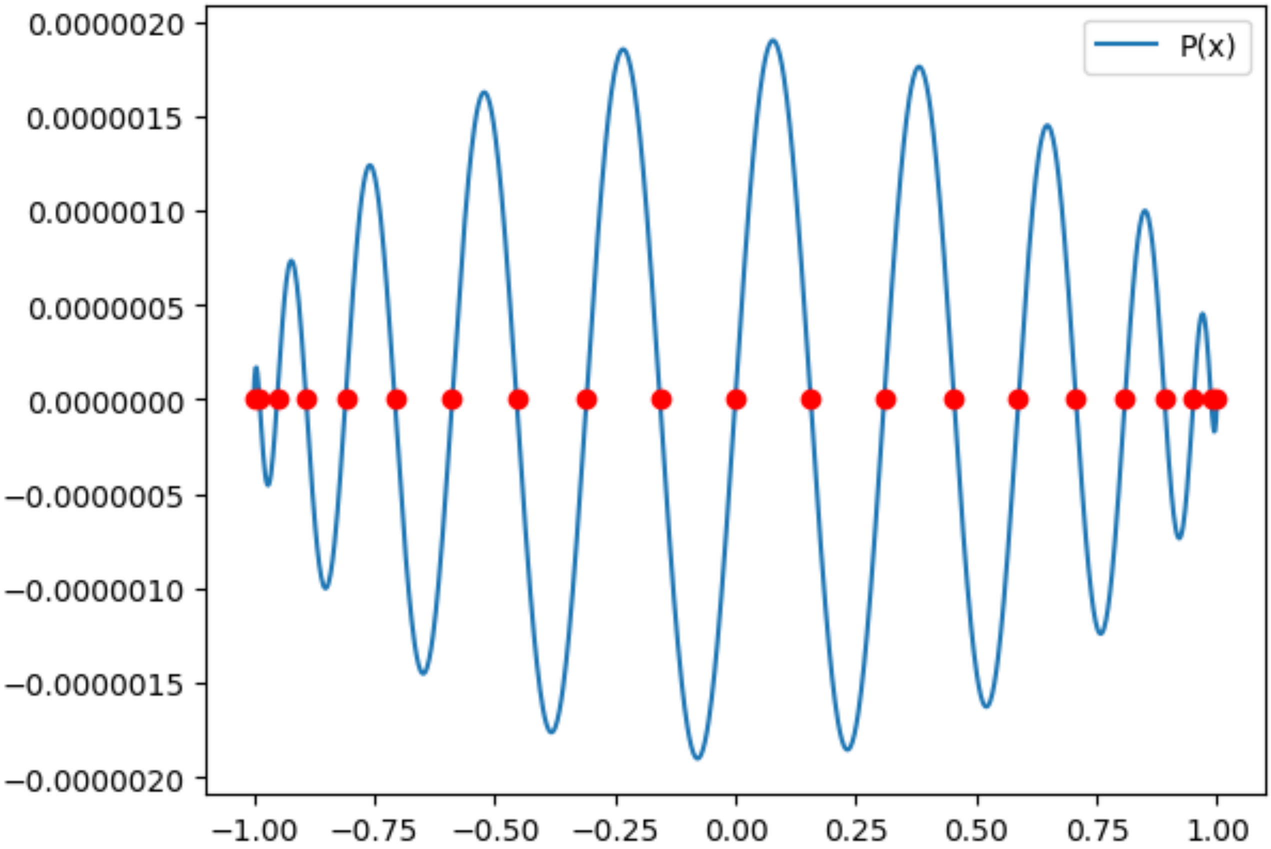

```
max. norm of P(x)=0.06249918587431625
```

```julia
n=10;
PlotPolynomial(cos.((0:n)*pi/n));
```




```
max. norm of P(x)=0.0019293110619113985
```

```julia
n=20;
PlotPolynomial(cos.((0:n)*pi/n));
```




```
max. norm of P(x)=1.9012283962444363e-6
```

Conclusion

To summarize, we have seen that among all the strategies of choosing nodes, the Chebyshev nodes of the first kind seems perform best, and the Chebyshev nodes of the second kind performs OK as well, in the sense that as the number of points n gets bigger, the maximum norm of $P(x)$ indeed decreases. The two other choices, random nodes and equally spaced nodes,