

Polynomial Approximation - II

Legendre vs Chebyshev Polynomials

Worksheet by Mike May, S.J., maymk@slu.edu

```
> restart;
```

Overview

In the last worksheet we looked at polynomial approximation using Legendre polynomials. These polynomials form an orthogonal set with a least squares fit definition for the norm. In this worksheet we will look at other norms and the corresponding sets of orthogonal polynomials.

We begin by loading Maple's package for orthogonal polynomials. We also load the plots package so that we can graph our results. We repeat the technical commands from the last worksheet.

```
> with(orthopoly); with(plots):  
assume('i',integer): assume('j',integer):  
interface(showassumed=0):
```

```
[G, H, L, P, T, U]
```

```
Warning, the name changecoords has been redefined
```

For the Legendre polynomials we saw that the norm is defined by $\langle f(x), g(x) \rangle = \int_{-1}^1 f(x) g(x) dx$

. The other inner products we will look at are defined by $\langle f(x), g(x) \rangle =$

$\int_{-1}^1 wfunc(x) f(x) g(x) dx$ where $wfunc(x)$ is a weight function. The use of different weight

functions gives us a way to make some parts of the domain more important than others. It also makes it possible to allow functions with poles into the set of functions we are trying to approximate.

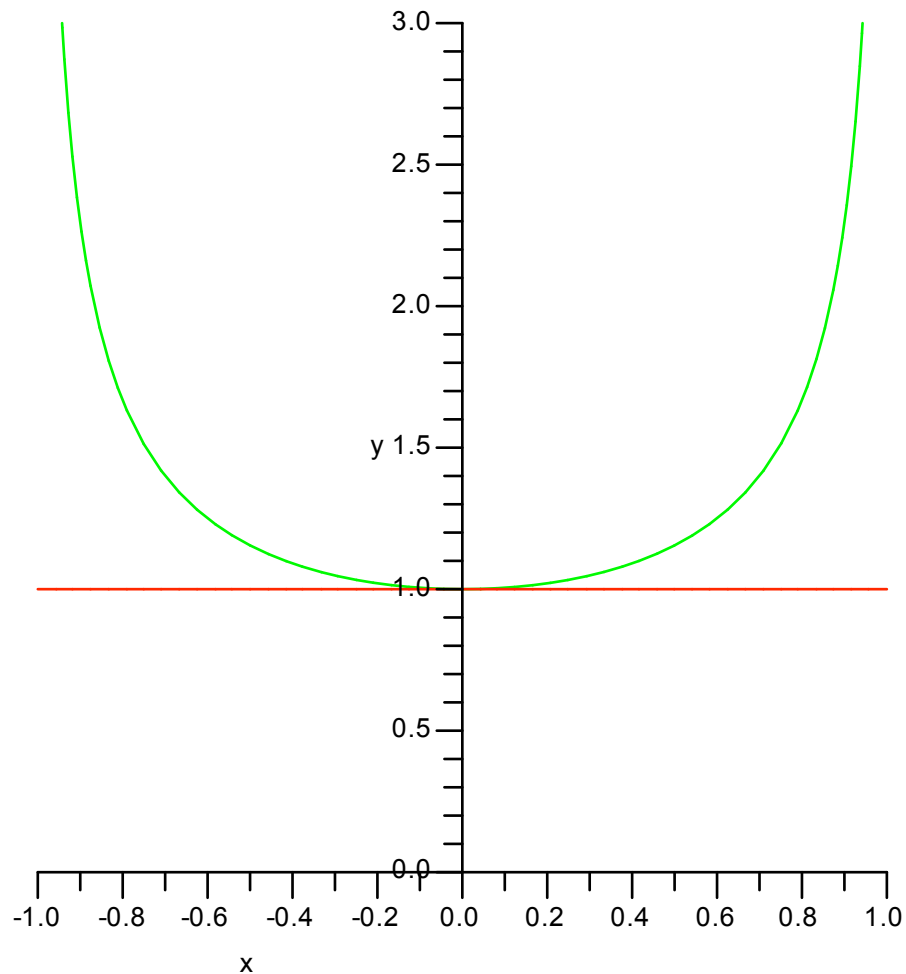
Chebyshev polynomials of the first kind

We will begin with Chebyshev polynomials of the first kind.

```
> ?orthopoly,T
```

The help page shows us that weight function of Chebyshev polynomials of the first kind is $(1-x^2)^{-1/2}$. The effect of the weight function is to make the endpoints count more when defining what it means for function to be small.

```
> plot({(1-x^2)^(-1/2),1}, x=-1..1, y=0..3);
```



Once we have loaded the orthopoly package, we obtain the nth Chebychev polynomial with $T(n,x)$;

```
> T(3, x);
```

$$4x^3 - 3x \tag{2.1}$$

Exercise:

1) Find the 1st, second, and 5th Chebychev polynomials of the first kind. Verify that they are orthogonal under the appropriate inner product.

```
>
```

▼ An extended example, back to $f(x) = \sin(3\pi x)$

Time to look at an example to see how this works out in a particular case. Once again we will use $\sin(3\pi x)$ as our function.

```
> func := x -> sin(3*Pi*x);
```

$$func := x \rightarrow \sin(3\pi x) \tag{3.1}$$

▼ Computing coefficients and approximating polynomials

Next we start computing coefficients. We will find the coefficients for both the Legendre polynomials and for Chebyshev polynomials of the first kind. Recall that if $P(n,x)$ is the nth

Legendre polynomial, the projection coefficient of $f(x)$ onto $P(n,x)$ should be

$$\int_{-1}^1 f(x) P(0,x) dx \text{ divided by } \int_{-1}^1 P(n,x) P(n,x) dx .$$

Similarly, if $T(n,x)$ is the n th Chebyshev polynomial of the first kind, the projection coefficient of $f(x)$ onto $T(n,x)$ is

$$\int_{-1}^1 wtCheb(x) f(x) P(0,x) dx \text{ divided by } \int_{-1}^1 wtCheb(x) P(n,x) P(n,x) dx ,$$

where $wtCheb(x) = (1-x^2)^{-1/2}$.

```
> wtchev := (1-x^2)^(-1/2):
for i from 0 to 15 do
normP[i] := evalf(Int((P(i,x))^2, x=-1..1));
coeffP[i] := evalf(Int(simplify(P(i,x)*func(x)*1.0), x=-1..1)
);
normT[i] := evalf(Int(wtchev*(T(i,x))^2, x=-1..1));
coeffT[i] := evalf(Int(wtchev*T(i,x)*func(x), x=-1..1)):
od;
```

$normP_0 := 2.$

$coeffP_0 := 0.$

$normT_0 := 3.141592654$

$coeffT_0 := 0.$

$normP_1 := 0.6666666667$

$coeffP_1 := 0.2122065907$

$normT_1 := 1.570796327$

$coeffT_1 := 0.5551985872$

$normP_2 := 0.4000000000$

$coeffP_2 := 0.$

$normT_2 := 1.570796327$

$coeffT_2 := 0.$

$normP_3 := 0.2857142857$

$coeffP_3 := 0.1763715524$

$normT_3 := 1.570796327$

$coeffT_3 := 0.2635803132$

$normP_4 := 0.2222222222$

$coeffP_4 := 0.$

$normT_4 := 1.570796327$

$coeffT_4 := 0.$

$normP_5 := 0.1818181818$

$coeffP_5 := -0.01322273781$

$normT_5 := 1.570796327$

$coeffT_5 := -.4620896712$

$normP_6 := 0.1538461538$
 $coeffP_6 := 0.$
 $normT_6 := 1.570796327$
 $coeffT_6 := 0.$
 $normP_7 := 0.1333333333$
 $coeffP_7 := -.2657941222$
 $normT_7 := 1.570796327$
 $coeffT_7 := -.9263350154$
 $normP_8 := 0.1176470588$
 $coeffP_8 := 0.$
 $normT_8 := 1.570796327$
 $coeffT_8 := 0.$
 $normP_9 := 0.1052631579$
 $coeffP_9 := 0.1669531813$
 $normT_9 := 1.570796327$
 $coeffT_9 := 0.7906766020$
 $normP_{10} := 0.09523809524$
 $coeffP_{10} := 0.$
 $normT_{10} := 1.570796327$
 $coeffT_{10} := 0.$
 $normP_{11} := 0.08695652174$
 $coeffP_{11} := -0.04841458486$
 $normT_{11} := 1.570796327$
 $coeffT_{11} := -.2675504440$
 $normP_{12} := 0.08000000000$
 $coeffP_{12} := 0.$
 $normT_{12} := 1.570796327$
 $coeffT_{12} := 0.$
 $normP_{13} := 0.07407407407$
 $coeffP_{13} := 0.008597245012$
 $normT_{13} := 1.570796327$
 $coeffT_{13} := 0.05294405753$
 $normP_{14} := 0.06896551724$
 $coeffP_{14} := 0.$
 $normT_{14} := 1.570796327$
 $coeffT_{14} := 0.$
 $normP_{15} := 0.06451612903$

$$\begin{aligned}
 \text{coeff}P_{15} &:= -0.001053256856 \\
 \text{norm}T_{15} &:= 1.570796327 \\
 \text{coeff}T_{15} &:= -0.007062371922
 \end{aligned}
 \tag{3.1.1}$$

Now we define the approximating polynomials.

```

> Lapprox := proc(m, x)
  local count;
  sum(P(count, x) * coeffP[count] / normP[count], count=0..m) :
end;
Chebapprox := proc(m, x)
  local count;
  sum(T(count, x) * coeffT[count] / normT[count], count=0..m) :
end;
Lapprox := proc(m, x)
  local count;
  sum((P(count, x) * coeffP[count]) / normP[count], count=0..m)
end proc
Chebapprox := proc(m, x)
  local count;
  sum((T(count, x) * coeffT[count]) / normT[count], count=0..m)
end proc

```

(3.1.2)

Comparing the approximations

Since our function has 6 bumps and looks like it could be a 7th degree polynomial, we want to compare the 7th degree approximations. Note first that the polynomial approximations are significantly different.

```

> n:= 7:
  Lapprox(n, x);
  Chebapprox(n, x);
  3.616684570 x - 37.06656803 x3 + 85.76884958 x5 - 53.44953677 x7
  2.507234281 x - 26.46978826 x3 + 61.34219016 x5 - 37.74228394 x7

```

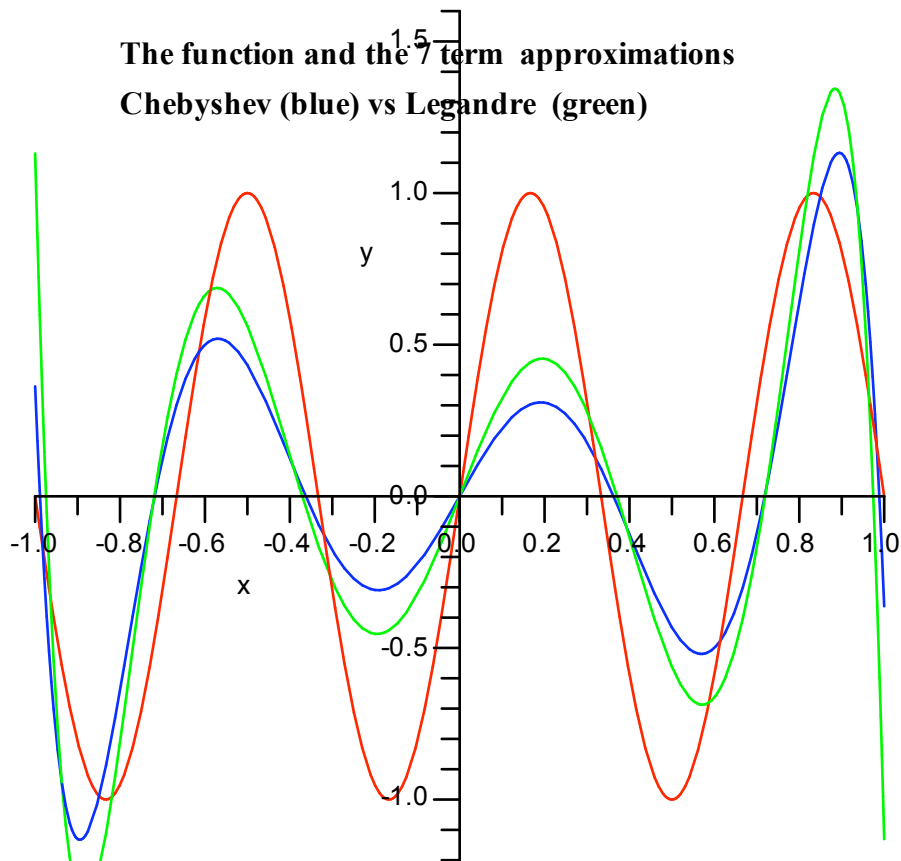
(3.2.1)

Next compare the graphs of the approximations and of the errors.

```

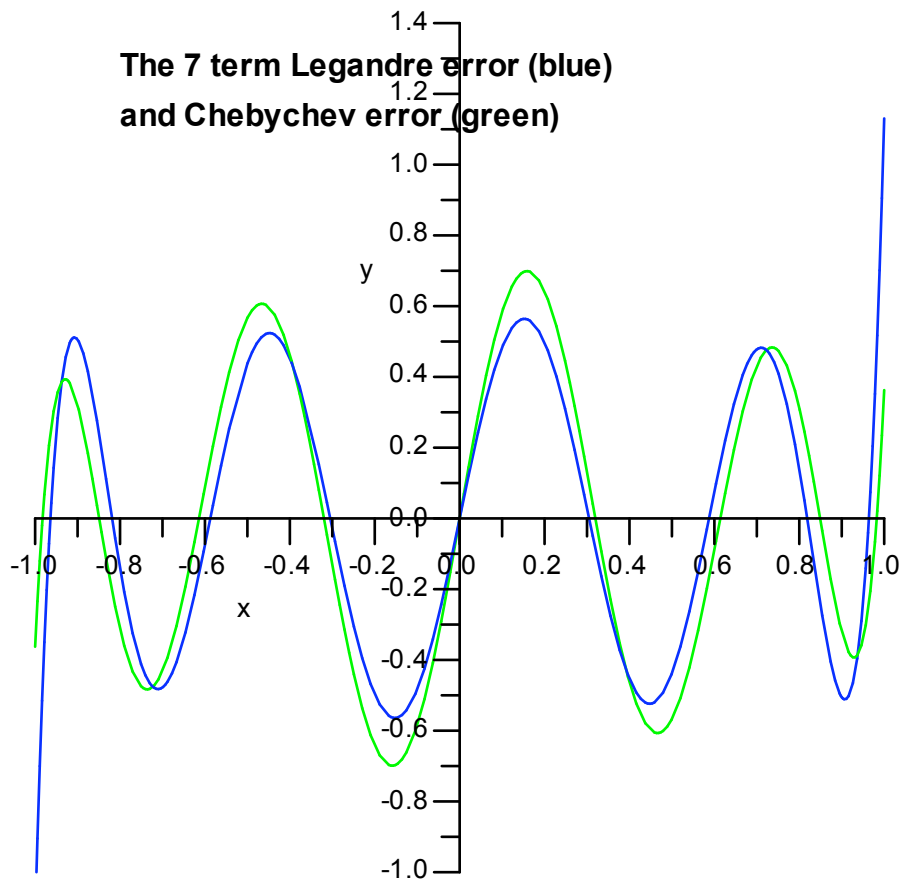
> lowy := -1.2: highy := 1.6: n:= 7:
  p11 := plot(func(x), x=-1..1, y=lowy..highy, color=red):
  p12 := plot(Lapprox(n, x), x=-1..1, y=lowy..highy, color=green)
  :
  p13 := plot(Chebapprox(n, x), x=-1..1, y=lowy..highy, color=
  blue):
  B := textplot({[-0.8, highy - (highy - lowy) / 20,
  `The function and the `|n||` term approximations`]
  ,
  [-0.8, highy - (highy - lowy) / 9, `Chebyshev (blue) vs
  Legendre (green)`]},
  align=RIGHT, font = [TIMES, BOLD, 12] ):
  display({p11, p12, p13, B}, view=[-1..1, lowy..highy]);

```



As expected, the Legendre approximation tends to be better in the center of the graph, while the Chebyshev approximation tends to be better on the ends. This becomes even clearer when we look at the graphs of the errors.

```
> lowy := -1.0: highy := 1.4: n:= 7:
  pl1 := plot(func(x) - Chebapprox(n,x), x=-1..1,y=lowy..highy,
  color=green):
  pl2 := plot(func(x) - Lapprox(n,x), x=-1..1,y=lowy..highy,
  color=blue):
  B := textplot({[-0.8,highy-(highy-lowy)/20,
    `The`||n||` term Legendre error (blue)`],
    [-0.8,highy-(highy-lowy)/9, `and Chebychev error`],
    (green)`}],
    align=RIGHT, font = [HELVETICA, BOLD, 12] ):
  display({pl1, pl2, B});
```



It is clear that the Chebyshev approximation is better at the endpoints, for degree 7.

>

Exercises:

>

2) For the example above, approximately what is the maximum error for over the domain $[-1, 1]$ for the 11th degree Legendre and Chebyshev approximations?

>

3) For the example above, approximately what is the maximum error for over the domain $[-.5, .5]$ for the 11th degree Legendre and Chebyshev approximations?

>

>

▼ Viewing the pattern of approximations

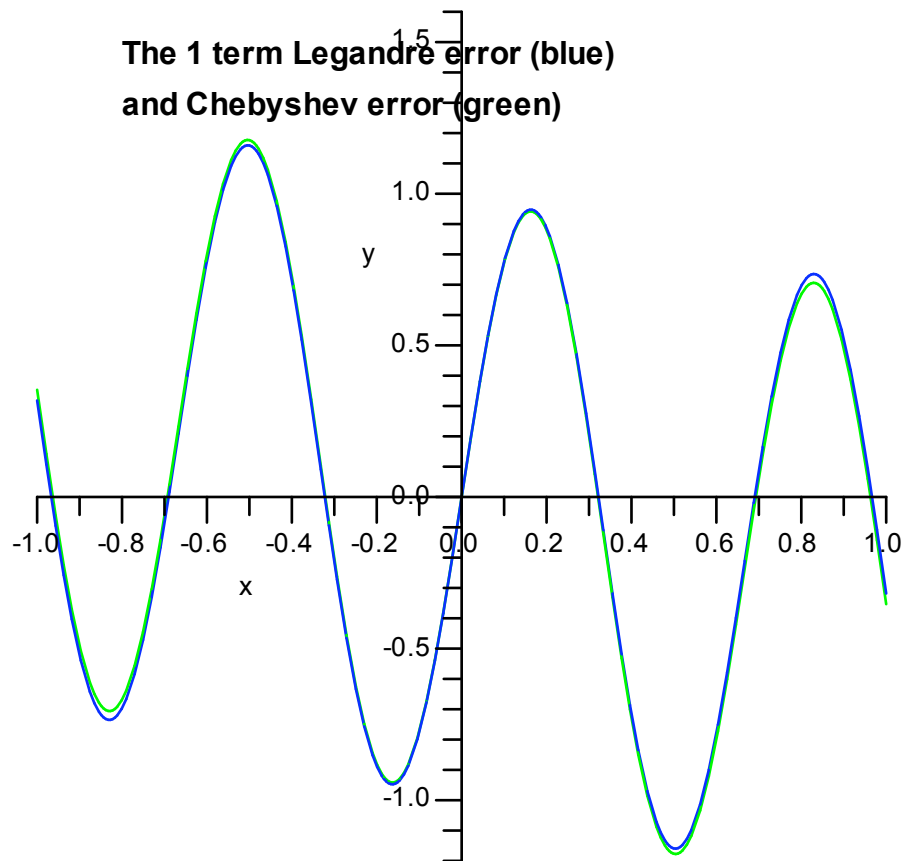
To illustrate the pattern we can do an animation of the errors produced by the polynomials.

```
> framererrCheb := proc(n,x, lowy, highy)
  local p11, p12, B:
  p11 := plot(func(x) - Chebapprox(n,x),
    x = -1..1, y = lowy..highy, color = green):
  p12 := plot(func(x) - Lapprox(n,x),
```

```

    x = -1..1, y = lowy..highy, color = blue):
    B := textplot({[-0.8,highy-(highy-lowy)/20,
        `The `|n|` term Legandre error (blue)`],
        [-0.8,highy-(highy-lowy)/9, `and Chebyshev error
(green)`]},
        align=RIGHT, font = [HELVETICA, BOLD, 12] ):
    display({p11, p12, B});
end:
> display([seq(ramererrCheb(counta,x, -1.2, 1.6), counta=1..15)
], insequence = true);

```



>

Exercise:

4) Explain why, with our example, the error only changes with odd degree approximations.

>

More exercises:

5) Find the 10th degree Legendre and Chebyshev polynomial approximations to $\sin(\pi x) + \cos(2\pi x)$. Approximately what is the maximum error for each of these polynomials? Approximately what is the error at the endpoints of the domain for each of the approximations.

>

6) Give a function that has a norm under either the Legendre or Chebyshev inner product but not under the other.

>