

Solutions to Problem Set 6, PHYS350, Spring 2009

by Matt Craig and Juan Cabanela

As usual, we begin by wiping memory and loading what we need.

```
> restart; with(plots): with(orthopoly):
```

▼ Problem 1

Odd functions will only contain odd terms in their expansions in terms of Legendre polynomials and even functions will have only even terms in their expansions. A function that is neither odd nor even will have both odd and even terms. So we really need to figure out whether the functions below are odd or even.

As a reminder, a function is odd if $f(-x) = -f(x)$ and a function is even if $f(-x) = f(x)$. If neither holds the function is neither even nor odd.

▼ Part a

Here $f(x) = \sin(\pi \cdot x)$. Since this is an odd function its expansion would contain only odd terms.

▼ Part b

Now $f(x) = x^2 e^{-x^2}$ which is an even function, so its expansion would contain only even terms.

▼ Part c

The function $f(x) = x e^{-x}$ is neither even nor odd. To check, note that $f(-x) = -x e^x$ which is neither $f(x)$ nor $-f(x)$. The expansion of this function would include both even and odd terms.

▼ Part d

The function $(x^2 - 1)^2$ is even, so its expansion contains only even terms.

▼ Problem 2

▼ Definition of cosine and sine fourier series

From the Supplement to Lab 5: cosine series routines

There are two differences between this and what was in Lab 6.

```
> assume(n, integer); assume(m, integer):
```

In the definition of the cosine I've change the period to 2L

```
> c := (x, n) -> cos((2*Pi*n*x)/(2*L)):
```

and the integral below for A goes from -L to L, not from 0 to L:

```
> A:=proc(expr, var, n)
    evalf(simplify(int(expr*c(var, n), var=-L..L)/int(c
```

```

    (var,n)*c(var,n),var=-L..L));
end proc:
> cosineFP:=proc(expr,var,n)
    A(expr,var,0)+add(A(expr,var,m)*c(var,m),m=1..n);
end proc:

```

From the solutions to Lab 5: sine series routines

Again, we change the sine term so the period is 2L

```

> s := (x,n) -> sin((2*Pi*n*x)/(2*L));
and the integrals go from -L to L
> B:=proc(expr,var,n)
    evalf(simplify(int(expr*s(var,n),var=-L..L)/int(s
    (var,n)*s(var,n),var=-L..L)));
end proc:
> sineFP:=proc(expr,var,n)
    add(B(expr,var,m)*s(var,m),m=1..n);
end proc:

```

Finally, we define a full Fourier series that is the sum of the sine and cosine series

```

> fullFP := proc(expr,var,n)
>     cosineFP(expr,var,n)+sineFP(expr,var,n);
> end proc:

```

Legendre polynomial expansion

First a routine to calculate the coefficients, copied from the Lab 6 solutions.

```

> legcoeff := proc (f,x,n)
> local a;
> a := evalf(((2*n+1)/2)*int(f*P(n,x),x=-1..1));
> return(a);
> end:

```

Now the function that calculates the series expansion. Note that I've used add instead of sum.

```

> legseries := proc(f,x,nmax)
> local s;
> s := add(legcoeff(f,x,n)*P(n,x),n=0..nmax);
> return(collect(s,x));
> end:

```

Part a

First we define the function of interest,

```

> f := x-> exp(-x^2)*x^2*cos(sqrt(x));

```

Next, we find its Taylor series. I'll start by expanding to 10th order and see what I get.

```

> ftaylor10:=unapply(convert(taylor(f(x),x, 11),polynom),x);

```

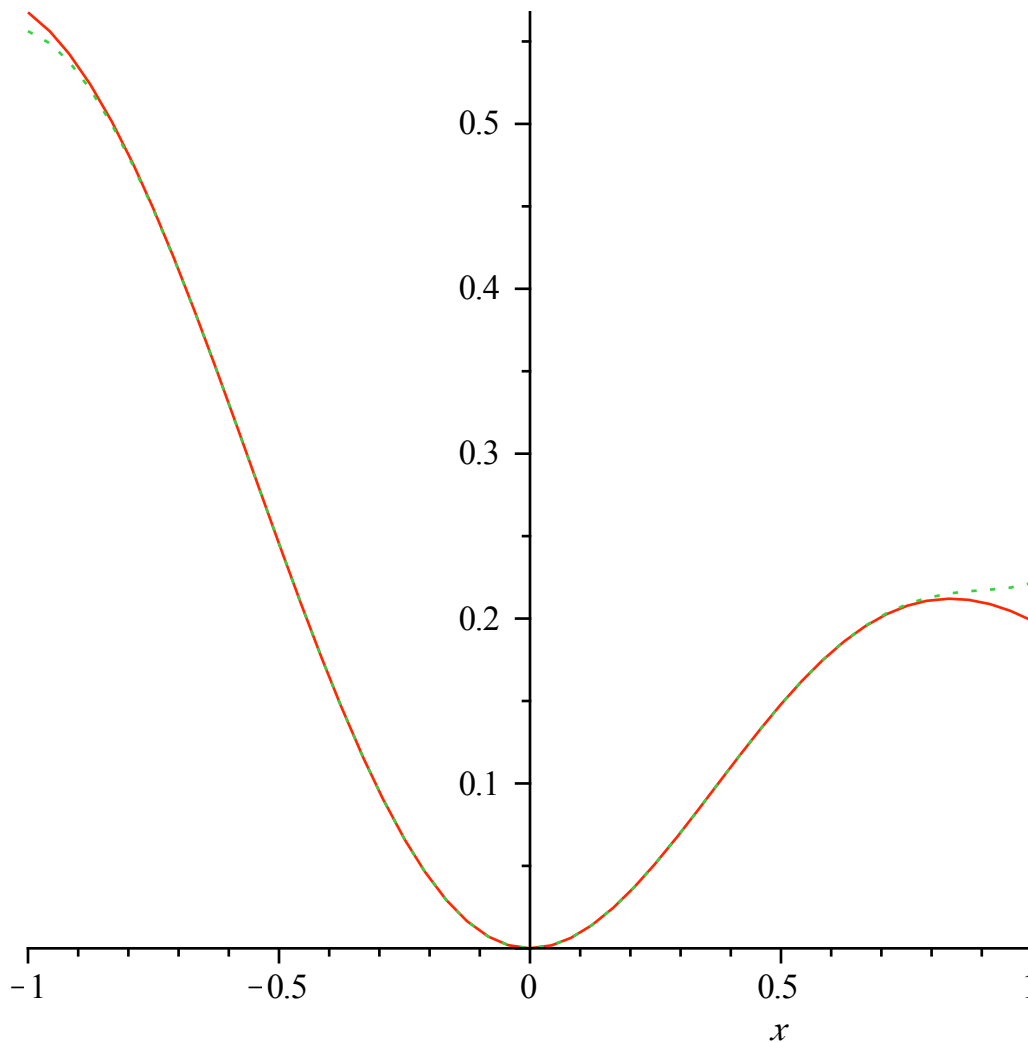
$$\begin{aligned}
 ftaylor10 := x \rightarrow & x^2 - \frac{1}{2}x^3 - \frac{23}{24}x^4 + \frac{359}{720}x^5 + \frac{18481}{40320}x^6 - \frac{902161}{3628800}x^7 \\
 & - \frac{9980897}{68428800}x^8 + \frac{7204341143}{87178291200}x^9 + \frac{726745175521}{20922789888000}x^{10}
 \end{aligned} \tag{2.3.1}$$

NOTE that to get the 10th order term I had to use "11" as the last argument to taylor.

```

> plot([f(x),ftaylor10(x)],x=-1..1,linestyle=[1,2]);

```



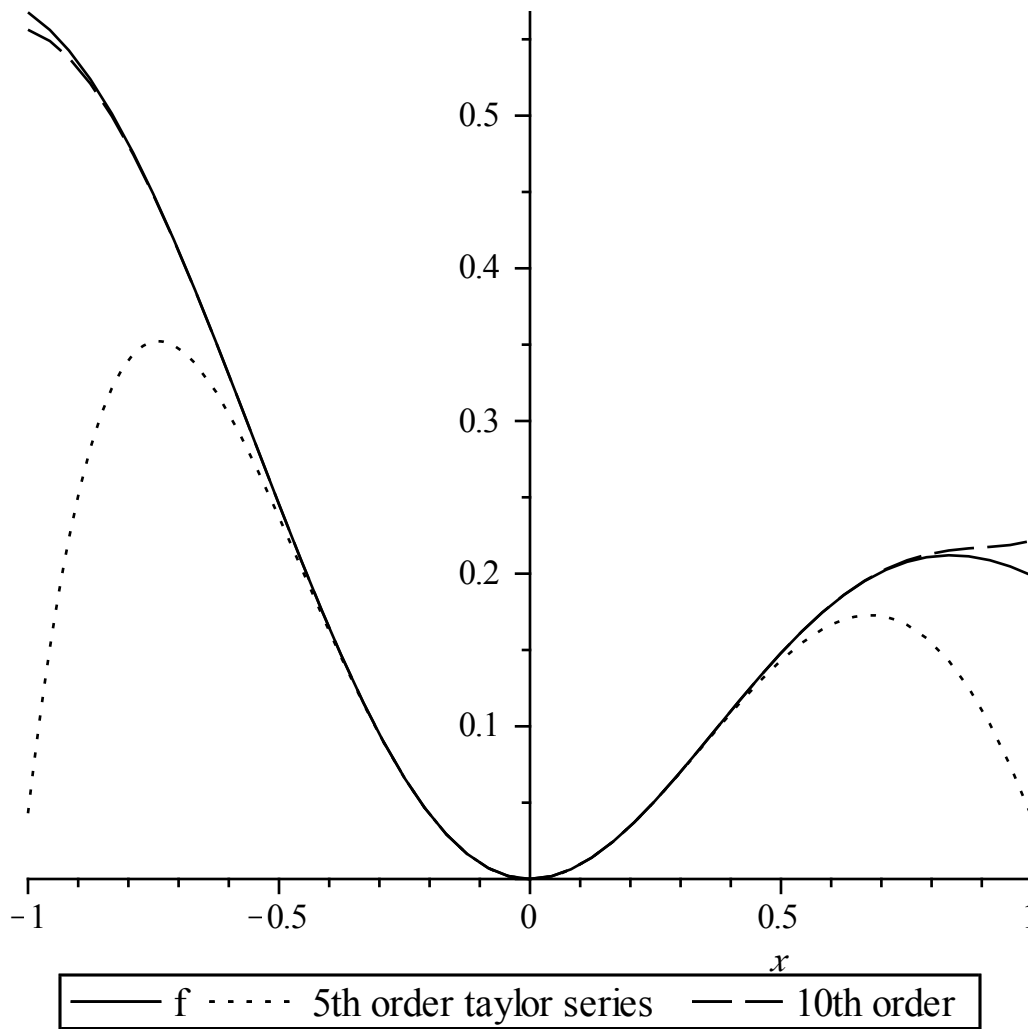
Wow! That fit really well. Let me try dropping down to 5 terms.

```
> ftaylor5:=unapply(convert(taylor(f(x),x,6),polynom),x);
```

$$ftaylor5 := x \rightarrow x^2 - \frac{1}{2}x^3 - \frac{23}{24}x^4 + \frac{359}{720}x^5$$

(2.3.2)

```
> plot([f(x),ftaylor5(x),ftaylor10(x)],x=-1..1,linestyle=[1,2,3],color=black,thickness=[1,1],legend=["f","5th order taylor series","10th order"]);
```

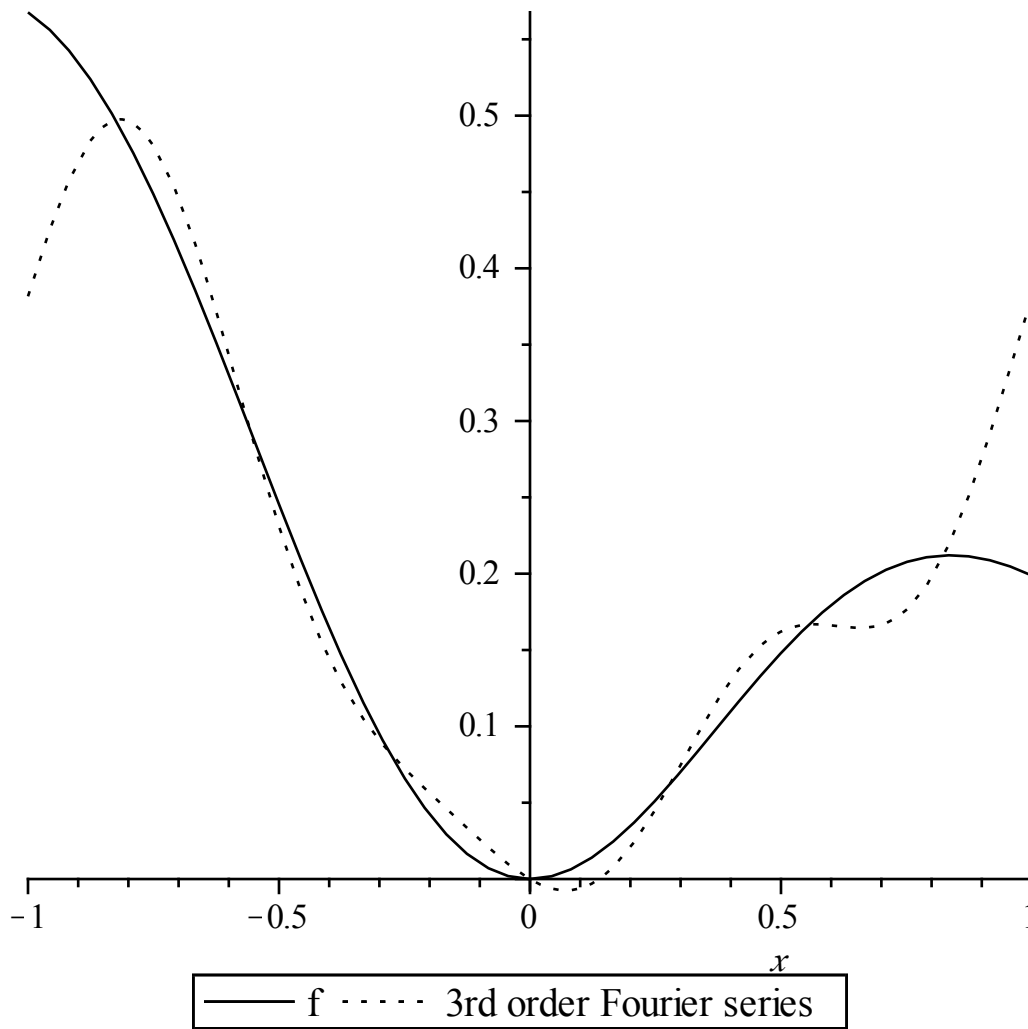


I would say 10th in the smallest order I need to get a decent fit.

Part b

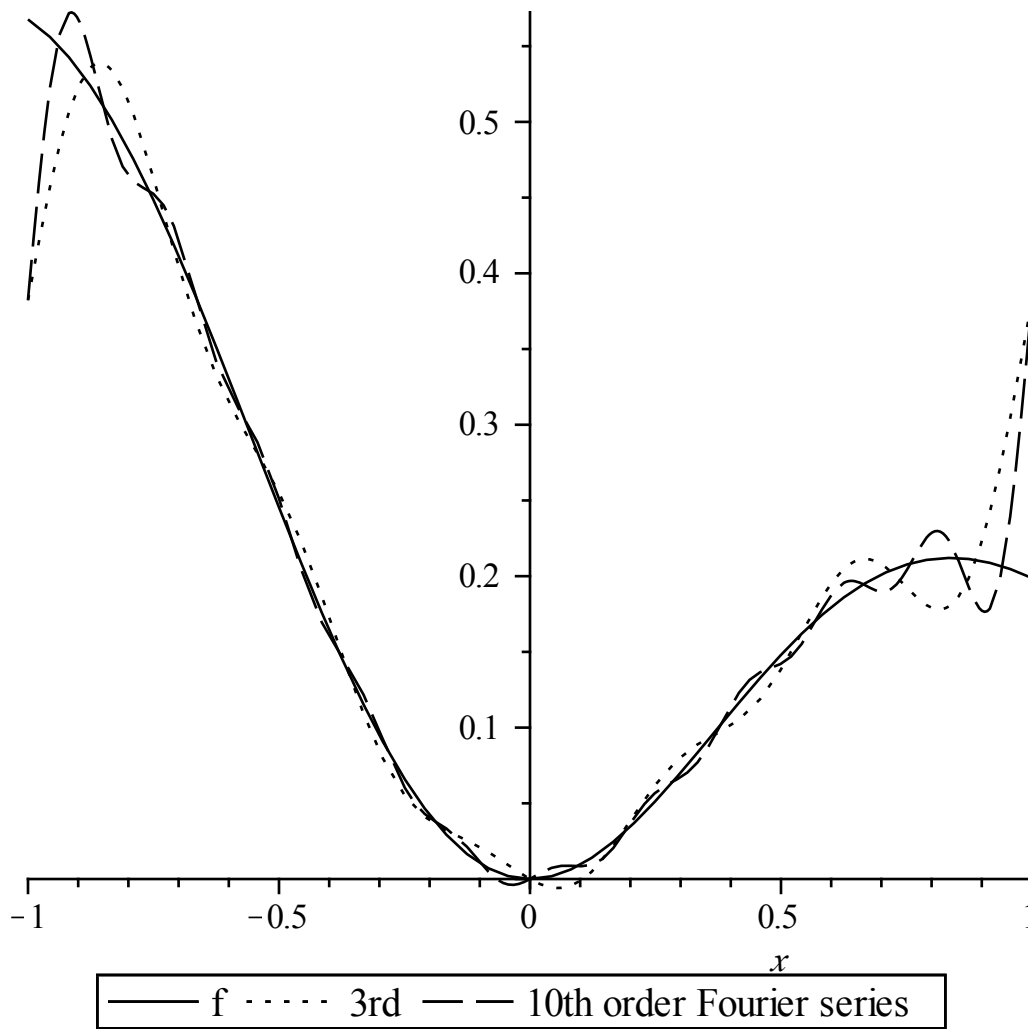
We'll try the Fourier series expansion to fifth order also to see how it looks.

```
> L:=1: ffourier5 := fullFP(f(x),x,5):
> plot([f(x),ffourier5],x=-1..1,linestyle=[1,2],color=black,
thickness=[1,1],legend=["f","5th order Fourier series"]);
```



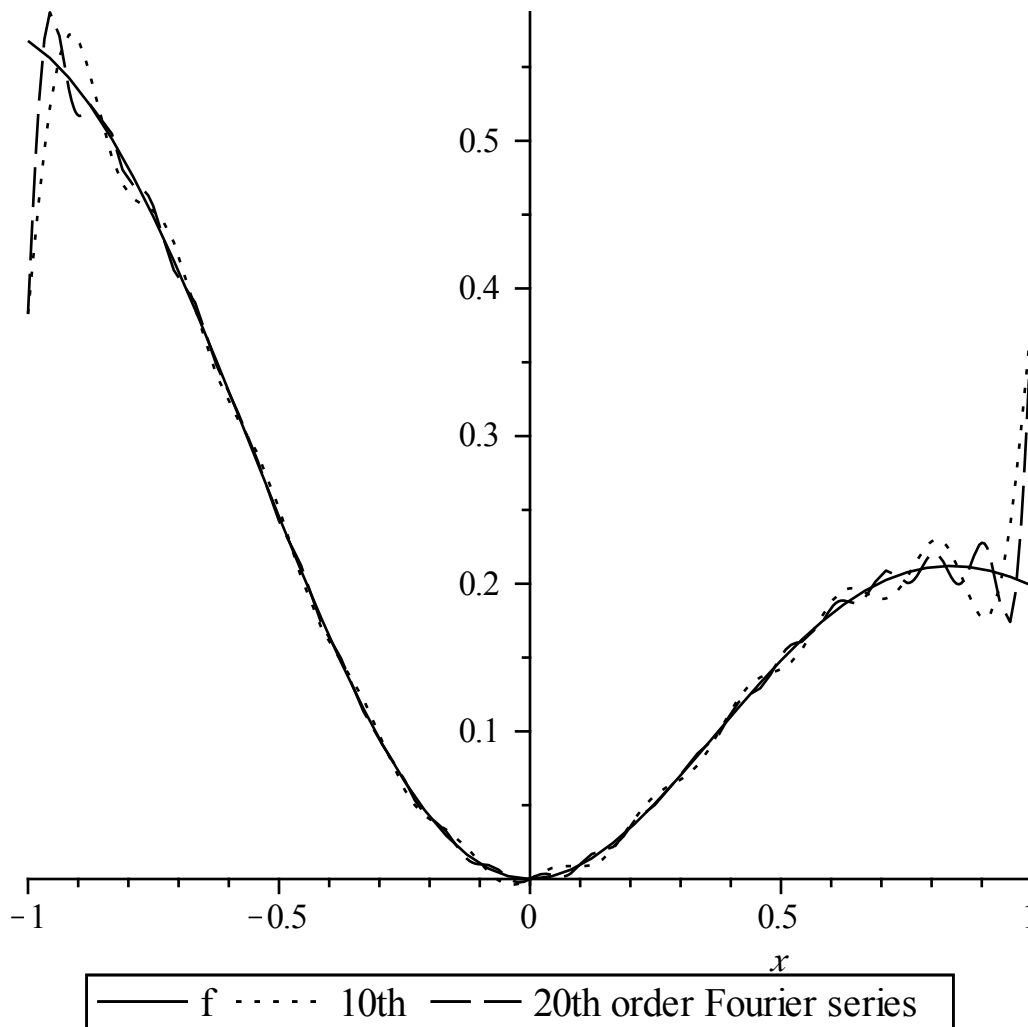
That doesn't look too good. Let's try ten terms.

```
> ffourier10 := fullFP(f(x),x,10):
> plot([f(x),ffourier5, ffourier10],x=-1..1,linestyle=[1,2,3],
color=black,thickness=[1,1,1],legend=["f","5th","10th order
Fourier series"]);
```



Not great, but not too bad...I'll try twenty terms just for fun.

```
> ffourier20 := fullFP(f(x),x,20):
> plot([f(x),ffourier10,ffourier20],x=-1..1,linestyle=[1,2,3],
color=black,thickness=[1,1,1],legend=["f","10th","20th order
Fourier series"]);
```

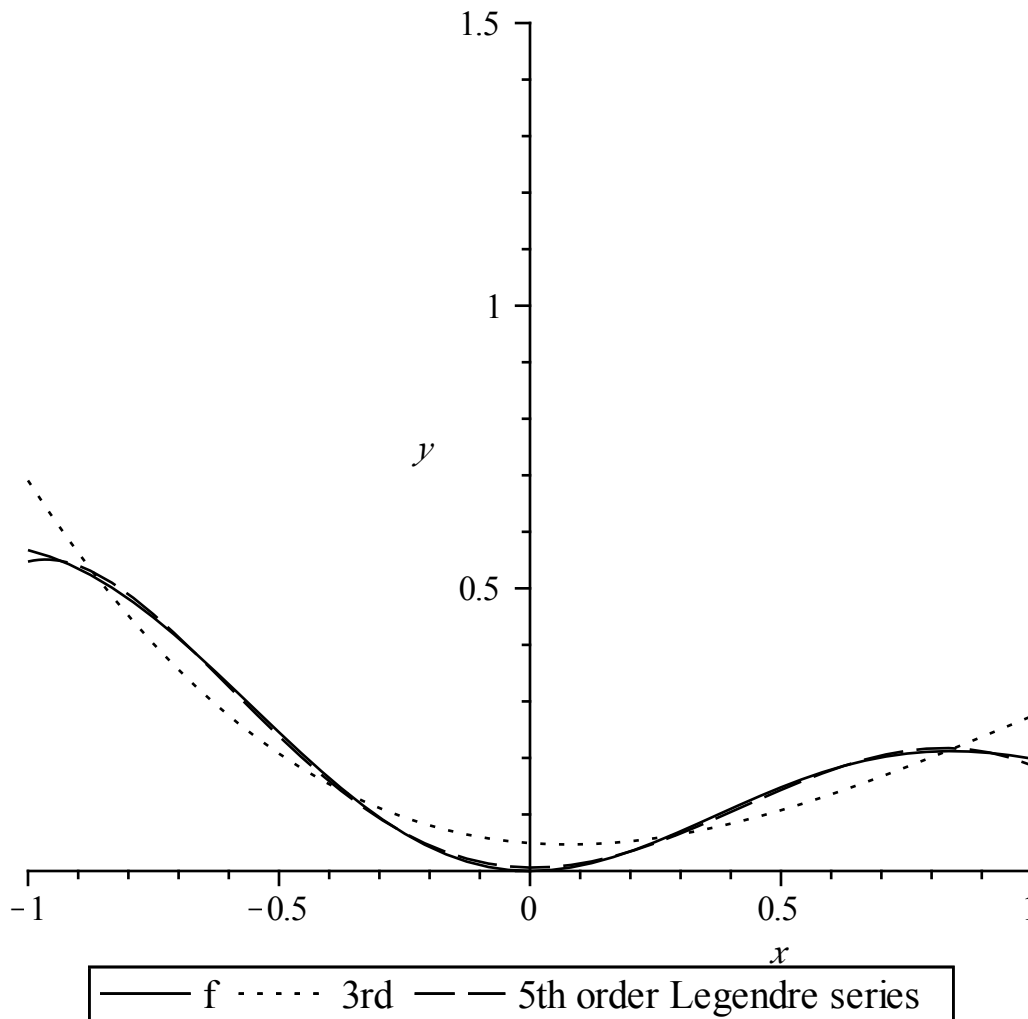


The 20th order really doesn't look a lot better than the 10th, so I'll say 10th is reasonably good.

Part C: Legendre polynomial expansion

I'll try expanding the function in Legendre polynomials to $n=3$ and to $n=5$. Why 3 and 5? Because I played around with the graphs for a bit and decided 3 isn't so good but 5 is very good.

```
> fleg3 := legseries(f(x),x,3): fleg5 := legseries(f(x),x,5):
> plot([f(x),fleg3,fleg5],x=-1..1,y=0..1.5,linestyle=[1,2,3],
color=black,thickness=[1,1],legend=["f","3rd","5th order
Legendre series"]);
```

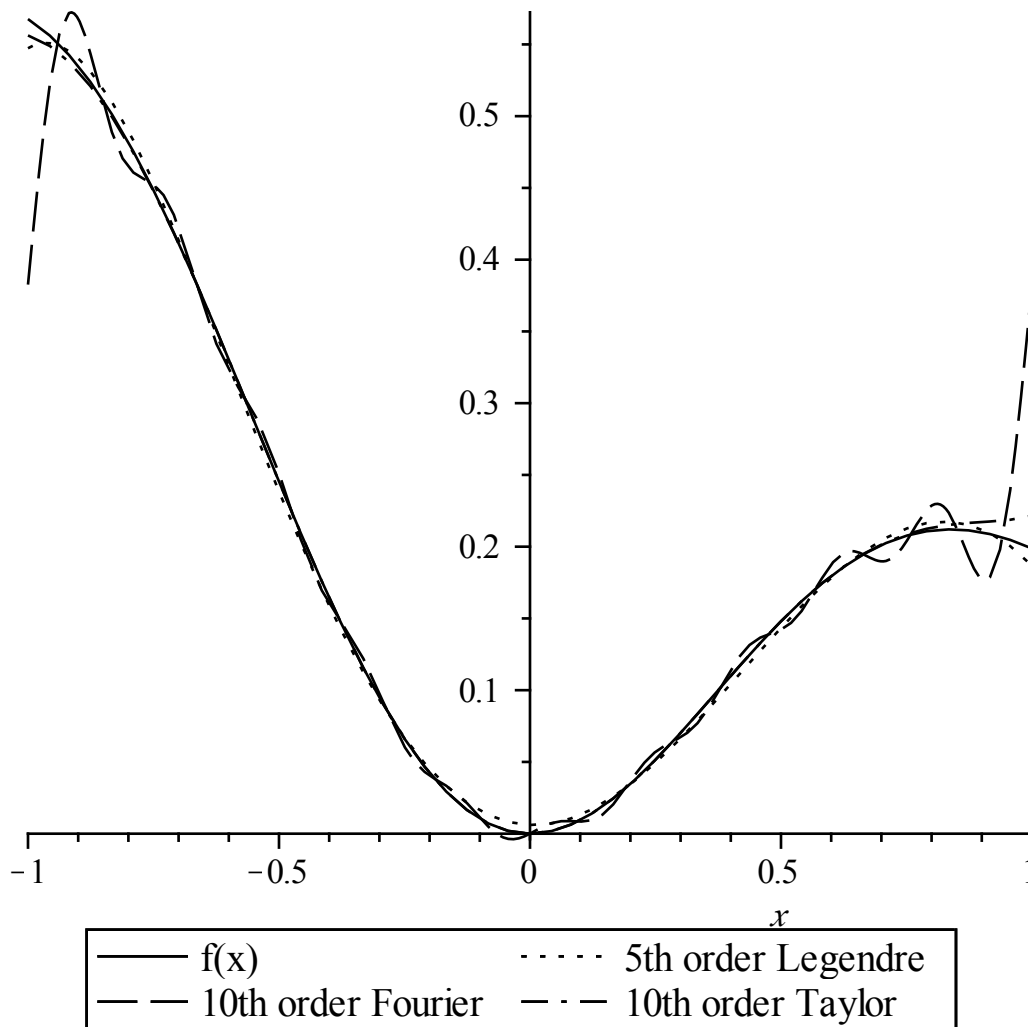


WOW! The difference between the exact function and the 5th order Legendre polynomial expansion is tiny.

Part d

I would say the best Taylor series was 10th order, the best Fourier series was 10th order, and the best Legendre series was 5th order. Take a look for yourself:

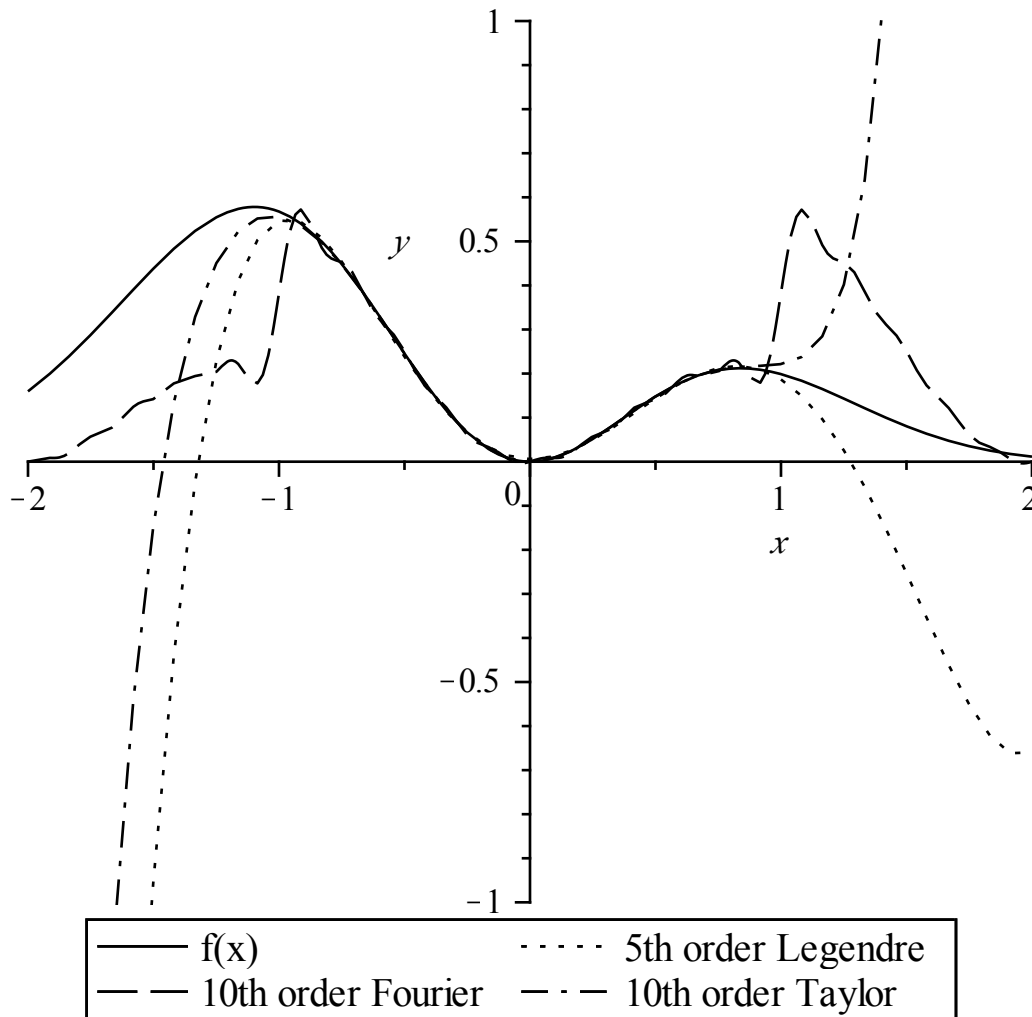
```
> plot([f(x), fleg5, ffourier10, ftaylor10(x)], x=-1..1, linestyle=[1,2,3,4], color=black, legend=["f(x)", "5th order Legendre", "10th order Fourier", "10th order Taylor"]);
```



It is a close call between the Taylor series and the Legendre polynomial, but I think the Legendre polynomial does a bit better, and with only terms up to x^5 . I'm not sure this is a general rule. The reason the Fourier series did so bad was that it assumes a periodic function, and the fact that $f(1)$ doesn't equal $f(-1)$ in our problem means the Fourier expansion is fitting a discontinuity.

Part e

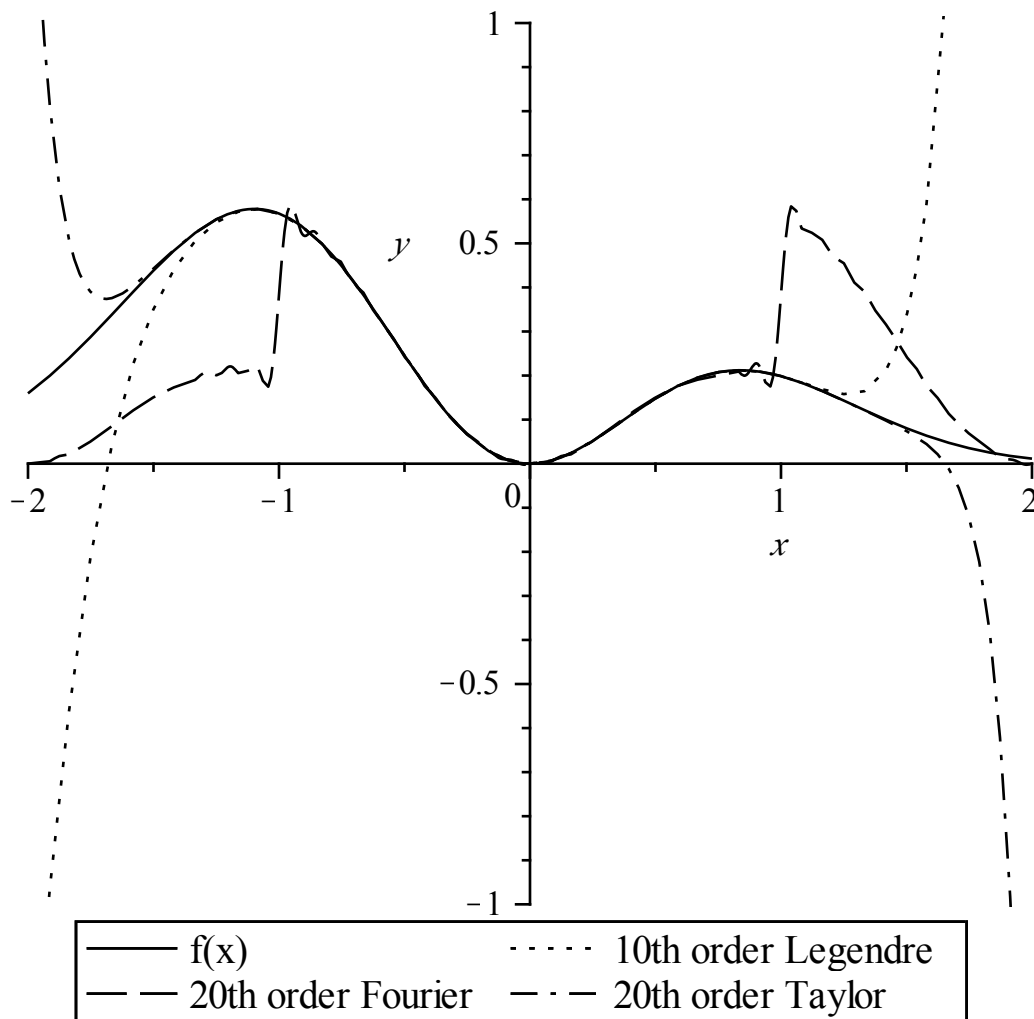
```
> plot([f(x), fleg5, ffourier10, ftaylor10(x)], x=-2..2, y=-1..1,
linestyle=[1,2,3,4], color=black, legend=["f(x)", "5th order
Legendre", "10th order Fourier", "10th order Taylor"]);
```



The Fourier series and Legendre series don't do well outside of the range -1 to 1 for two very different reasons. For the Fourier series it is because we chose a period of 2 (since $1 - (-1)$ is 2) and our function isn't periodic. The Legendre polynomials are orthogonal only over the range -1 to 1. Outside of that range you cannot expect them to fit well.

Only the Taylor series would get better by adding more terms. Don't believe me? In the plot below I've doubled the number of terms in each series.

```
> ftaylor20:=unapply(convert(taylor(f(x),x, 21),polynom),x):
fleg10 := legseries(f(x),x,10): plot([f(x),fleg10,ffourier20,
ftaylor20(x)],x=-2..2,y=-1..1,linestyle=[1,2,3,4],color=
black, legend=["f(x)", "10th order Legendre", "20th order
Fourier", "20th order Taylor"]);
```



Now the Taylor series is doing quite a bit better. The Legendre polynomial is also somewhat better but not as good as the Taylor series.

Part f

Fourier series are best at representing periodic functions. Legendre functions work very well if you need an approximation over only a limited range. Taylor series will work in any situation but require a large number of terms to accurately represent a function over a large range.