

# Animating Taylor polynomials in several variables

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

Revised by Russell Blyth - blythrd@slu.edu

>

When working with Taylor polynomials there are two features that are worth demonstrating using animations to compare higher and higher degree Taylor polynomials with the original function. The first feature is that, as we would hope with something that is called an approximation, the Taylor polynomials morph into the function as the degree of the approximation is increased. The second feature is observing the radius of convergence of a Taylor series. If the Taylor series has a finite radius of convergence, it will only morph to the function in that region, and will never become a good approximation outside that region. As with all concepts in multi-variable calculus, we start by reviewing the one variable calculus concept.

The animations are memory intensive. Since this often causes memory problems the animation is done in this separate worksheet. Before working through this worksheet, you should have worked through the worksheet on Taylor polynomials in several variables.

First some technical details. Rather than working out the Taylor polynomials from the definition, we are going to call the Maple library for multivariate Taylor polynomials.

> **restart: with(plots): with(Student[MultivariateCalculus]):**

## One variable convergence demonstration

We start with the single variable case. We first demonstrate the idea of convergence. Click on the graph that results from executing the following code and use the control buttons on the tool bar at the top of the window to play the animation. You may step through the animation one frame at a time using the third button from the left, or slow down the animation using the FPS dialog.

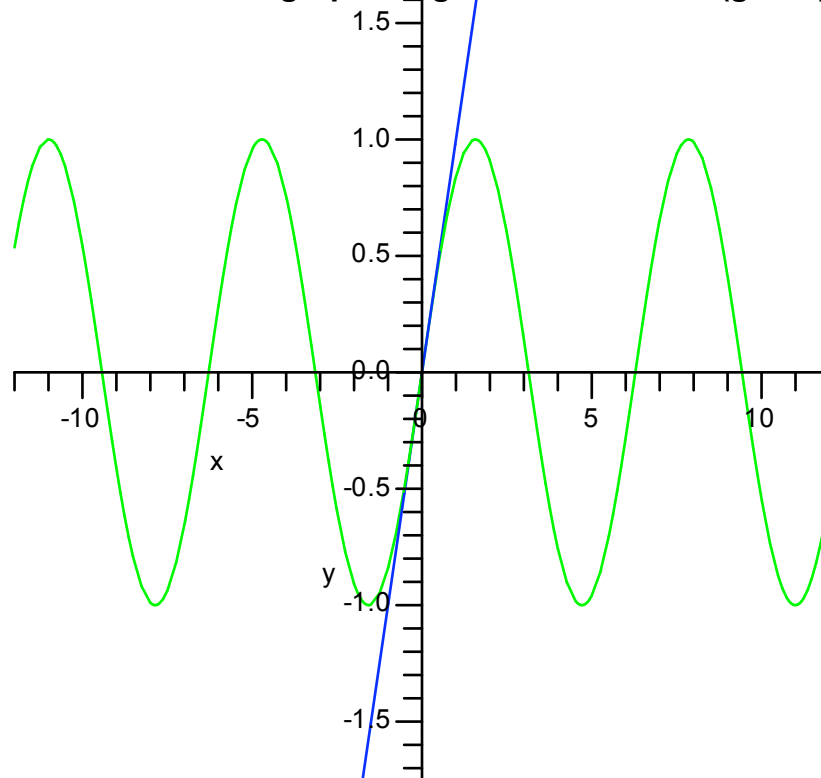
```
> func := sin(x);
aboutx := 0; width := 12: height:= 1.75:
yval := subs(x=aboutx, func);
minx := aboutx-width: maxx := aboutx+width:
miny := yval-height: maxy := yval+height:
mindeg := 2: degsteps := 10: bydeg :=4:
fudge:= (maxy-miny)/20:
framer2d := proc(i)
    local A, B, C, D, TaylorDeg:
    TaylorDeg := mindeg + bydeg*i:
    A := plot([func, TaylorApproximation(func, [x]=[aboutx],
        TaylorDeg)], x=minx..maxx, y=miny..maxy,
        color=[green,blue]):
    C := textplot([minx, maxy,
        `The Taylor polynomial of degree `||(mindeg + bydeg*i)||
        `(blue)`], align={ABOVE,RIGHT}, font = [HELVETICA, BOLD,
12],
        view=[minx..maxx, miny..maxy]):
    D := textplot([maxx, maxy-fudge,
        `graphed against the function (green)`],
```

```

align={ABOVE,LEFT}, font = [HELVETICA, BOLD, 12],
view=[minx..maxx,miny..maxy]:
display({A,C, D},view=[minx..maxx,miny..maxy]);
end:
display([seq(ramer2d(i), i=0..degsteps)], insequence = true);
      func := sin(x)
      aboutx := 0
      yval := sin(0)

```

The Taylor polynomial of degree 2 (blue)  
graphed against the function (green)



>

## One variable radius of convergence demonstration

Next we demonstrate the idea of radius of convergence. We look at an example that illustrates a finite radius of convergence. Once again, click on the graph and play the animation.

```

> func := 1/(1 + x^2);
  aboutx := .5; width := 3.5; height := 1.2;
  yval := eval(func, x=aboutx);
  minx := aboutx-width; maxx := aboutx+width;
  miny := yval-height; maxy := yval+height;
  mindeg := 2; degsteps := 20; bydeg := 15;
  fudge := (maxy-miny)/20;
  framer2d := proc(i)

```

```

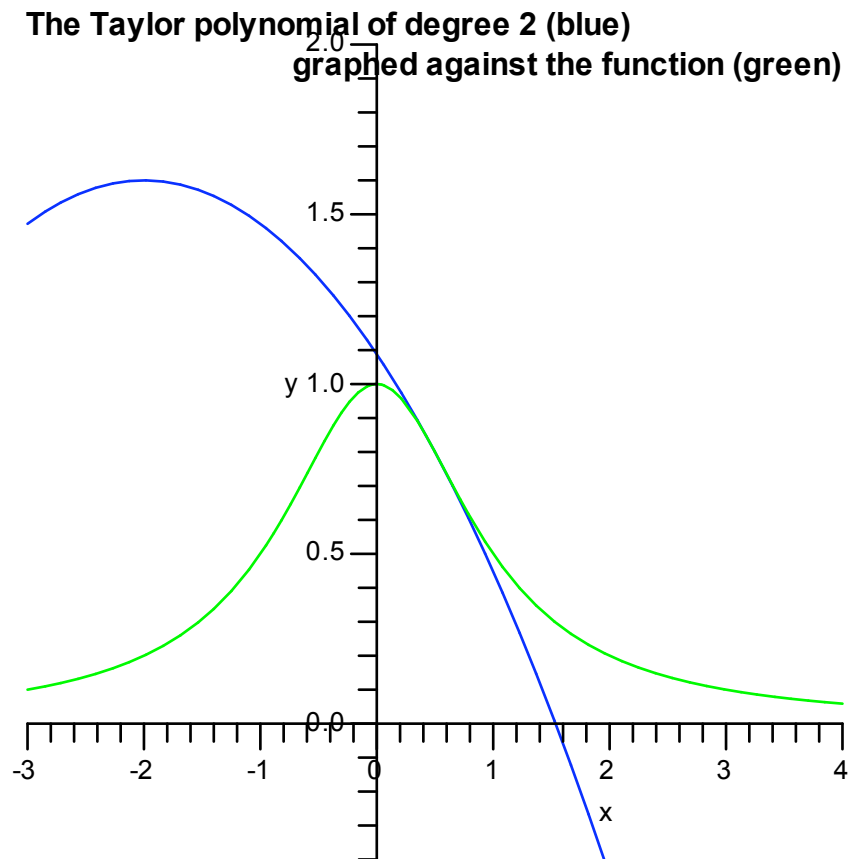
local A, C, D, TaylorDeg:
TaylorDeg := mindeg + bydeg*i:
A := plot([func, TaylorApproximation(func, [x]=[aboutx],
TaylorDeg)], x=minx..maxx, y=miny..maxy,
color=[green, blue]):
C := textplot([minx, maxy,
`The Taylor polynomial of degree `|(mindeg + bydeg*i)|
` (blue)`, align={ABOVE, RIGHT}, font = [HELVETICA, BOLD,
12],
view=[minx..maxx, miny..maxy]):
D := textplot([maxx, maxy-fudge,
`graphed against the function (green)`,
align={ABOVE, LEFT}, font = [HELVETICA, BOLD, 12],
view=[minx..maxx, miny..maxy]):
display({A, C, D}, view=[minx..maxx, miny..maxy]);
end:
display([seq(ramer2d(i), i=0..degsteps)], insequence = true);

```

$$func := \frac{1}{1+x^2}$$

$$aboutx := 0.5$$

$$yval := 0.8000000000$$



The approximation improves as the degree of the Taylor polynomial increases, but only within the

radius of convergence.

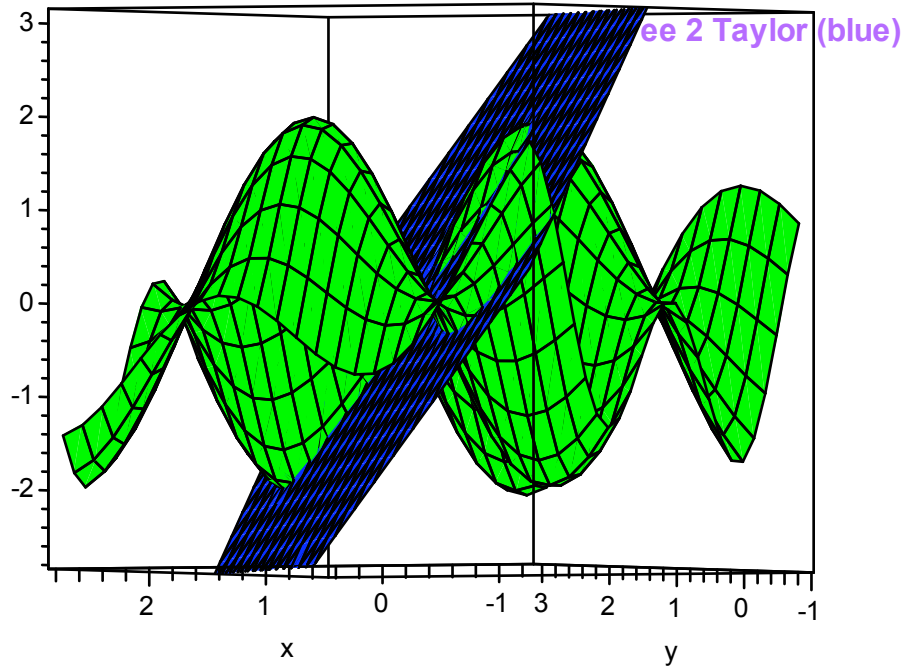
>

## Two variable convergence demonstration

Now we want to show the same idea with a function of two variables. We start with the same function we used in the previous worksheet, but expand about a different point.

Now we do a sample animation.

```
> g := (x,y) -> sin(x+3*y)+cos(2*x-y):
a:= Pi/4:          b:=Pi/3:
del := 2:          height := 3;
mindeg := 2:      degsteps := 7:      bydeg :=2:
lowx := a-del:    highx := a+del:
lowy := b-del:    highy := b+del:
zval := evalf(g(a, b));
lowz := zval-height:    highz := zval+height:
xrange:= lowx..highx:  yrange := lowy..highy:
zrange := lowz..highz:
framer3d := proc(i)
  local A, B, T1, deg:
  deg := mindeg + bydeg*i:
  A := plot3d(mttaylor(g(x,y),[x=a, y=b], mindeg + bydeg*i),
    x=xrange, y=yrange, view= zrange, color=blue):
  B := plot3d(g(x,y),x=xrange, y=yrange,
    view= zrange, color=green):
  T1 := textplot3d([(highx+lowx)/2,lowy,highz,
    `Degree `||deg||` Taylor (blue)`],
    align={BELOW,RIGHT}, font=[HELVETICA, BOLD, 12]):
  display3d({A,B,T1},axes=boxed);
end:
display([seq(framer3d(i), i=0..degsteps)], insequence = true,
  style=patch, view=zrange);
  height := 3
  zval := 0.1589186230
```



>

Step through the animation one step at a time using the ->| button. At each step, rotate the graph and note how the convergence changes with the degree of the approximation.

(The block of code above is designed so that all the variables to be changed for different examples are in the first 5 lines.)

By the time we get to the degree 16 approximation, the approximation matches the surface very closely over a reasonably large region.

## ▼ A visualization of "Radius of Convergence"

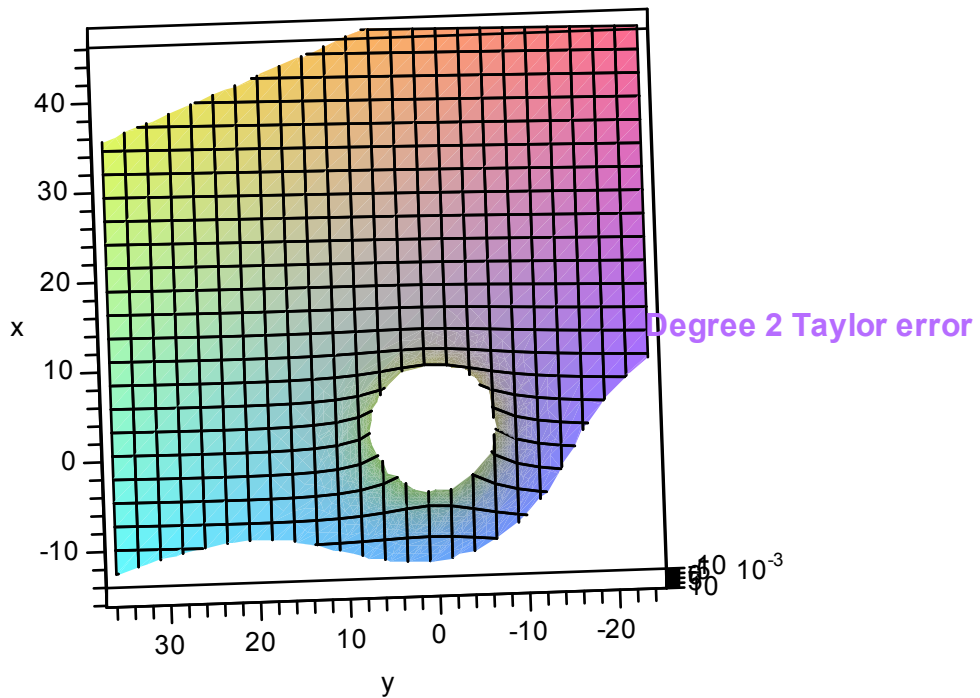
When we studied single variable Taylor polynomials we learned that the Taylor series of some functions have a finite radius of convergence. It is instructive to look at what happens in the 2 variable case. Consider the function  $h(x,y) = 1/(x^2+y^2)$  approximated by Taylor polynomials centered at (2,1). It is clear that we will never have a good approximation that includes the origin. What we get is an approximation that is good in a circular region with a radius small enough to exclude the trouble spot at the origin.

The following animation shows the error between a function and its Taylor approximation as the degree of the Taylor polynomial grows.

```

> h := (x,y) -> 1/(x^2+y^2):
a := 15:          b := 6:
del := 30:          errortolerance := .01;
mindeg := 2:      degsteps := 10:      bydeg :=4:
lowx := a-del:    highx := a+del:
lowy := b-del:    highy := b+del:
lowz := -errortolerance:  highz := errortolerance:
xrange:= lowx..highx:  yrange := lowy..highy:
zrange := lowz..highz:
framer3de := proc(i)
  local A, T1, deg:
  deg := mindeg + bydeg*i:
  A := plot3d(
    mtaylor(h(x,y),[x=a, y=b], deg)-h(x,y),
    x=lowx..highx, y=lowy..highy,
    view= lowz..highz, orientation=[135,45]):
  T1 := textplot3d([(highx+lowx)/2,lowy,highz,
    `Degree `||deg||` Taylor error`],
    align={BELOW,RIGHT}, font=[HELVETICA, BOLD, 12]):
  display3d({A,T1},axes=boxed);
end:
display([seq(framer3de(i), i=0..degsteps)], insequence = true,
  style=patch, view=lowz..highz);
      errortolerance := 0.01

```



Note that increasing the degree does not increase the region where the series converges to the function; it is instructive to look down on the error surface from above. Inside the region of convergence the approximation becomes very good; outside it remains poor.

