01 Apr 1991

Mathematica Demonstrations for Calculus

Bryan Krueger

MATHEMATICA DEMONSTRATIONS FOR CALCULUS

Bryan Krueger

Introduction

The objective of this research was to generate useful classroom demonstrations of fundamental Calculus concepts. Herein are the code listings of two *Mathematica* "programs." The first displays individual plots of two dimensional Taylor Series approximations. The second set of code creates a TNB frame on a three dimensional space curve. Folowing this introduction is a complete listing of each "program" along with an example of its function.

Each set of *Mathematica* commands follows a simple format. The special functions are loaded first, then an input routine, followed by the calculation/graphics output section. One thing to note about these demonstrations is the input section. Previously, new functions were put in a single block statement which utilized commands written in a format such that they could be loaded the same way as other special functions, i.e. the cross product (<<:LinearAlgebra:Cross.m). As the necessary input increases these block statements become lengthy and confusing. To avoid this, *Mathematica*'s built in user active input is used. This allows for better control over output such as the number of plots generated and their size. It may even speed usage as memorization and constant reference are not required to place the needed values in their exact location and format. Hence, fewer errrors and frustrations are created while interest is kept high.

Just these few examples are consistent with the great usefulness and versatility of *Mathematica*. The code lines can easily be taken apart, added to, and used in other student exercises. Each program generates a separate plot for every iteration. The plots can then be combined for colorful examples and high quality animations. *Mathematica* built-in organization and text functions are very useful in the production of reports such as this one and insure its capabilities will be used for other courses aside from matematics.

Taylor Series

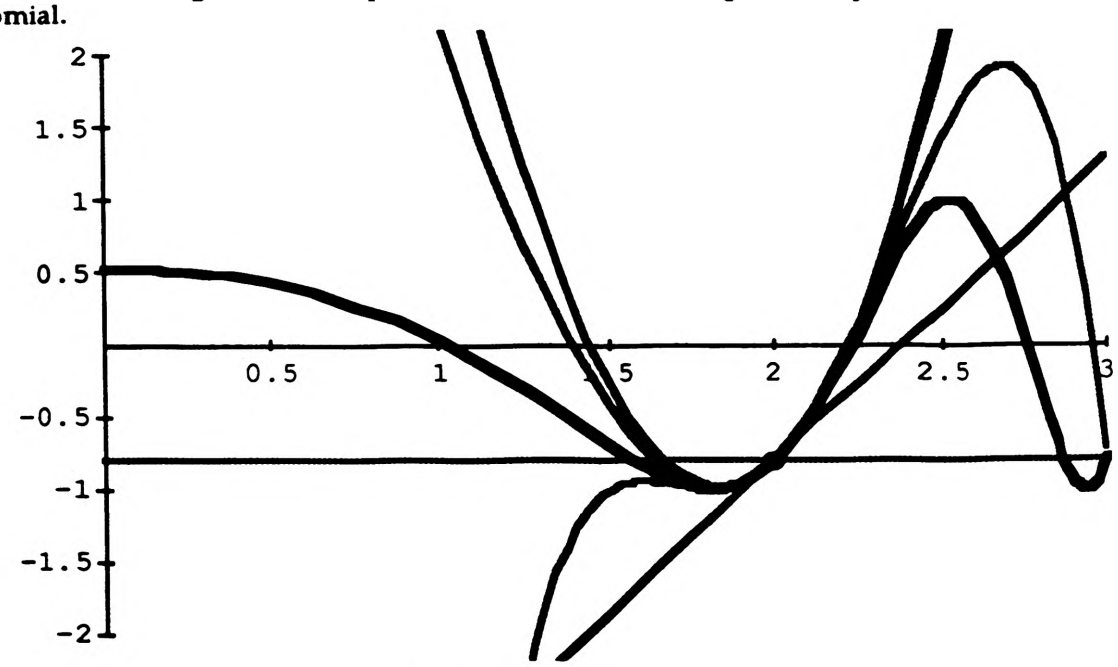
This code produces plots of a user-supplied function and a specified number of approximating Taylor polynomials about a user-given point.

```
(*Two-Dimensional Taylor Series Approximations*)

f[x_] =     Input["Enter a function in terms of x:"];
p     =     Input["Enter the point about which to approximate:"];
x1    =     Input["Enter the MINIMUM x value for viewing:"];
x2    =     Input["Enter the MAXIMUM x value for viewing:"];
y1    =     Input["Enter the MINIMUM y value for viewing:"];
y2    =     Input["Enter the MAXIMUM y value for viewing:"];
t     =     Input["Enter the number of iterations to perform:"];
t     =     t - 1;
ts[x_,a_] = 0;
pt    =     Graphics[{RGBColor[0,0,1], PointSize[.02], Point[{p,f[p]}]}];
Do[ts[x_,a_] = ts[x,a] + (D[f[a],{a,n}] (x - a)^n)/n!;
```

```
Show[Plot[{f[x],ts[x,p]},{x,x1,x2},
            PlotStyle->{Thickness[.0075], RGBColor[1,0,0]},
            DisplayFunction->Identity],
      pt, PlotRange->{{x1,x2},{y1,y2}},
        AspectRatio->1/GoldenRatio,
         Axes->{0,0}, DisplayFunction->$DisplayFunction],
{n, 0, t}]
```

Here is f[x] = Cos[Cosh[x]] and its first five Taylor polynimials expanded about x = 2. These have been merged into one plot - the command above gives a separate cell for each polynomial.



The TNB Frame

The following command creates frames for an animation showing the unit tangent, unit normal, and unit binormal vectors for a given space curve. On a color monitor these can be assigned different colors for easy identification. The output shows some frames when the space curve is

$$\{x[t], y[t], z[t]\} = \{\text{Cos}[t], \text{Sin}[t], \text{Sin}[2t]\}.$$

```
<<:Graphics:ParametricPlot3D.m
<<:LinearAlgebra:Cross.m
rvx[t_]  =  Input["Enter the x-component for a position vector:"];
rvy[t_]  =  Input["Enter the y-component for a position vector:"];
rvz[t_]  =  Input["Enter the z-component for a position vector:"];
t0       =  Input["Enter the initial value of t:"];
t1       =  Input["Enter the final value of t:"];
dt       =  Input["Enter the stepsize for t (used when drawing the curve):"];
tvm      =  Input["Enter the scalar multiplier for the tnb frame:"];
min      =  Input["Enter the largest NEGATIVE number any function achieves:"];
max      =  Input["Enter the largest POSITIVE number any function achieves:"];
m        =  Input["Enter the number of frames in the animation:"];
rv[t_]={rvx[t], rvy[t], rvz[t]}
```

```
tv[t_]=D[rv[t],t]/Sqrt[D[rv[t],t].D[rv[t],t]]
nv[t_]=D[tv[t],t]/Sqrt[D[tv[t],t].D[tv[t],t]]
bv[t_]=Cross[tv[t], nv[t]];
tstep=(t1-t0)/nn;
sc=SpaceCurve[{rvx[t], rvy[t], rvz[t]}, {t, t0, t1, dt}, DisplayFunction->Identity];
Do[Show[Graphics3D[{PointSize[.03], RGBColor[.5,.5,.5], Point[rv[n]]}],
      Graphics3D[{Thickness[.01], RGBColor[1,0,0],
          Line[{rv[n], rv[n]+tvm tv[n]}]}],
      Graphics3D[{Thickness[.01], RGBColor[0,1,0],
          Line[{rv[n], rv[n]+tvm nv[n]}]}],
      Graphics3D[{Thickness[.01], RGBColor[0,0,1],
          Line[{rv[n], rv[n]+tvm bv[n]}]}],
      Graphics3D[Thickness[.01], RGBColor[0,0,0]], sc,
BoxRatios->{1,1,1},
          PlotRange->{{min, max}, {min, max},{min, max}},
      DisplayFunction->$DisplayFunction],
{n, t0, t1, tstep}]
```