

<http://www.radford.edu/~cmett/ictcm2001/>

---

## Maple Code

### 1 Intro to Maple

#### 1.1 Define a function

Define the function  $f(x) = \sin(x^2)$

```
> f:=x->sin(x^2);
```

#### 1.2 Plot

Construct a graph of  $f(x)$  on the domain  $[-\pi, 2\pi]$

```
> plot(f(x),x=-Pi..2*Pi);
```

On the same axes plot the line  $y = \frac{1}{x}$

```
> plot([f(x),1/x],x=-Pi..2*Pi,y=-10..10,color=[red,blue]);
```

#### 1.3 Solve

Find the intersections on  $[0,3]$

```
> solve(f(x)=1/x,x);  
> r1:=fsolve(f(x)=1/x,x);  
> r2:=fsolve(f(x)=1/x,x=r1..2);
```

### 2 Animate

```
> restart;  
> #?animate  
> #get Maple help, including examples  
> with(plots):  
> #load the animate tool  
> animate(sin(x+h),x=-5..5,h=0..3);  
> #pause and play animation  
> animate({sin(x+h),sin(x)+h},x=-5..5,h=0..3);  
> #pause and play animation  
> animate3d(t*(sin(y)+cos(x)), x=-2*Pi..2*Pi, y=-2*Pi..2*Pi,  
> t=-3..3,frames=50);  
> #pause and play animation
```

### 3 Mix of Animation and display:

Keep a stationary background and overlay an animation:

See **slicesOfCone.mws** for simple overlay

See **cycloid.mws** for shifted overlay. NOTE: this animation also demonstrates the "pixel generation" in time sequence, emulating the way a graphing calculator displays parametric curves with direction.

## 4 Zoom

```
> restart; with(plots):
> a:=Pi/3; #center of attention
> N:=3; #number of frames
> Z:=2; #zoom factor (e.g. Z=2 will double magnification each
time)
> W:=3; #initial window size
> window:=x=a-W/(k*Z)..a+W/(k*Z);
> S:=seq(plot(sin(x),window),k=1..N):
> display(S,insequence=true);
> #not desirable because window is the union of all windows
> #pause and play animation
```

How can we truly get zoom?

```
> display(%);
> #display the array of frames:
```

See program **zoomAndReadStat.mws**.

## 5 Add a tracking point

Motivation: ICTCM: pre-calculus question on locus of point traced by vertex in family of parabolas (see **vertexAnimation.mws**)

```
> f:=x->5*x*(x-1)*(x+1);
> animate(t*f(x),x=-1.5..1.5,t=1..3); p3:=%:
> #pause and play animation
```

mis-Conclusions: "coefficient t has the effect of **rotating** the curve"

```
> animate([1.2,f(1.2)*t,x=-1.5..1.5],t=1..3,style=point,color=blue):
> p4:=%:
> display(p4,p3);
```

Leave a trail?

```
> animate([1.2,f(1.2)*t,x=-1.5..1.5],t=1..3,style=point,color=blue):
> p4:=%:
> N:=16; #default number of frames in animation
> b:=3;
> a:=1;
> delta:=(b-a)/N; #step size
> S4:=seq(plot([seq([1.2,f(1.2)*(a+k*delta)],k=1..j)],style=point,color
=blue),j=1..N):
> display(S4,insequence=true);
> #pause and play animation
> display(p3,S4);
> #pause and play animation
> display(p3,S4,insequence=true);
> #Why doesn't this work?
> #pause and play animation
```

Let's "marry" the two animations:

```
> S5:=seq(plot((a+k*delta)*f(x),x=-1.5..1.5),k=1..N):
> display(S4,S5,insequence=true);
> #pause and play animation
> #This still does not do the trick. Why?
> P:=seq(display(S4[j],S5[j]),j=1..N):
> display(P,insequence=true);
> #pause and play the animation
> # Success!
```

Add original curve?

```
> p0:=plot(f(x),x=-1.5..1.5,color=blue,linestyle=2):
> P:=seq(display(p0,S4[j],S5[j]),j=1..N):
> #add the original to EVERY frame
> display(P,insequence=true);
> #pause and play animation
```

## 6 Adding text

```
> #?plot[options]
> #display(P,insequence=true,title="c*f(x)");
> display(P,insequence=true,title="c*f(x)",titlefont=[TIMES,BOLD,22]);
```

Make the title reflect the current coefficient?

```
> #?sprintf
> S5:=seq(plot((a+k*delta)*f(x),x=-1.5..1.5,title=sprintf("
> %3.2f*f(x)",evalf(a+k*delta))),k=1..N):
> #display(S5,insequence=true); #test result
```

Now we re-construct the set of images to display:

```
> P:=seq(display(p0,S4[j],S5[j]),j=1..N):
> display(P,insequence=true);
```

See `secantAnim.mws` for another example of a data-ticker on animation.

## 7 Building flexibility

User can insert any function? any window? an number of frames? etc.

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

### 7.1 User inserts values:

```
> f:=x->5*x*(x-1)*(x+1);
> #f:=x->sin(x); #any function
```

#### 7.1.1 Define a window [a,b] for the x-axis:

```
> a:=-1.5;
> b:=1.5;
> #a:=-2*Pi;
> #b:=2*Pi;
```

#### 7.1.2 Define a set of coefficients, c, for graphing

$$y = cf(x)$$

```
> c_start:=1;
> c_end:=3;
```

### 7.1.3 Select a point $x_0$ for tracking:

```
> x0:=.5; #user defines value in domain
> if (x0>b) then print ("Please go back pick a point 'x0' in domain")
> end if; #make sure point is in domain
> if (x0<a) then print ("Please go back pick a point 'x0' in domain")
> end if; #make sure point is in domain
> #if (evalf(x0)>evalf(b)) then print ("Please go back pick a point
> 'x0' in domain") end if; #make sure point is in domain
> #if (evalf(x0)<evalf(a)) then print ("Please go back pick a point
'x0'
> in domain") end if; #make sure point is in domain
> plot(f(x),x=a..b,color=blue,linestyle=2): p0:=%:
> plot([[x0,f(x0)]],color=blue,style=point):
> c0:=%:
> display(p0,c0);
> #make sure everything meets user's specifications and save for
later
> display
### Make sure this is the function you desire.
```

### 7.1.4 A number of frames desired for animation:

```
> N:=10;
```

### 7.2 Build the animation:

```
> delta:=(c_end-c_start)/N: #step size
> Pnts:=seq(plot([seq([x0,f(x0)*(c_start+k*delta)], k=0..j)],
> style=point, color=blue), j=1..N): #trail of points
> VarPlot:=seq(plot((c_start+k*delta)*f(x), x=a..b), k=0..N):#actual
> function
> P:=seq(display(p0,Pnts[j],VarPlot[j]),j=1..N):#assemble
> display(P,insequence=true);
> #pause and play the animation
```

### 7.3 Exercise:

1. Test effects of shrinking coefficient by changing just one value in the above "user-defined values"
2. Build an animation to demonstrate the effect of changing coefficients "c" in  $y = f(cx)$

## Conic Sections

*animation demo of intersections of plane and cone*

```
> restart;
> with(plots):
> A :=plot3d(r,theta=0..2*Pi,r=-10..10, coords=cylindrical,
> orientation=[134,82]):
> B :=animate3d(t/2*x+t/4*y-1,x=-5..5,
> y=-5..5,t=0..15,color=yellow):
> display(A,B,style=patchnogrid,view=[-10..10,-10..10,-10..10]);
> #pause to play animation
#interesting variations found by altering coefficients of x and y in above
plane, and changing vertical shift constant c= -1. For example, planes through
vertex of cone is obtained by c=0.
```

Change  $x$  to  $x - 3$  in the definition of  $\mathbf{B}$  above to get a parabola.

Can also change viewing angle in 3d figure with left mouse – click and hold to rotate.

---

Created by Coreen Mett, Radford University, December 1997, last modified October, 2001

---

## Generating a cycloid path

*Animate the particle on spoke of wheel to create the cycloid using parametric equations.*

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

### 8 User sets values

r = radius of wheel

b = radius of particle on spoke, the particle to be tracked

```
> r:=1;
> b:=1.6;
```

### 9 Define the parametric curves

#### 9.1 path of cycloid at time t

```
> x:=t->r*t-b*sin(t);
> generate the path of the cycloid
> y:=t->r-b*cos(t);
```

#### 9.2 circle at time t

```
> X:=s->r*cos(s)+r*t;
> generate the circle
> Y:=s->r*sin(s)+1;
```

### 10 Build the animation

```
> N:=20:
> #number of frames
> for t from 0 to N do
> path[t]:=plot([x(w),y(w),w=-0.01..t],color=red):
> spoke[t]:=plot([[r*t,1],[x(t),y(t)]],color=black):
> wheel[t]:=plot([X(s),Y(s),s=0..2*Pi],color=blue):
> p[t]:=display(path[t],spoke[t],wheel[t]):
> end do:
> display(p[j]$j=0..N, insequence=true,
> scaling=constrained,title=sprintf('b=%2.2f\n r=%2.2f',b,r));
> #pause and play animation
```

### 11 Display together

Plot  $x(t)$ ,  $y(t)$  and  $r(t)$  together so that user can watch the behavior of  $x(t)$  and  $y(t)$  as  $r(t)$  is generated.

```
> t:='t': with(plottools): #required for the translate tool
> N:=20: #number of frames
> R:=2: #number of wheel rotations desired
> R:=2*2*Pi:
> for k from 1 to N do
> T:=R/N*k;
> pt:=plot([T,s,s=-1..x(R)],color=blue): #vertical line at time
t
> ft:=plot([x(t),y(t),0],t=0..T,color=[gold,green,black]):
> ct:=plot([x(t),y(t),t=0..T]):
> B[k]:=plots[display](translate(pt,0,4),ct,translate(ft,0,4)):
> end do:
```

```
> plots[display](B[j]$j=1..N,insequence=true,title="x(t) is gold
\n
> y(t) = green \n cycloid is red");
> #pause and play animation
```

---

## Zoom and ReadStat

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

Before executing the commands below, set Options -> Plot Display -> Window

It is also probably a good idea to select Window -> Cascade

## 12 Zoom tool

```
> f:=x->sin(x);
> print('You will be asked to enter some values. ');
> print('Follow your value with a semicolon, and hit <ENTER>');
```

Prompt user for x-target of zoom:

```
> x0 := readstat("enter value of x to center your zoom:");
> 2;
```

User may alter then values of N and Z below:

```
> N:=5;
> #number of zooms, must be an integer
> Z:=2;
> # positive zoom factor (i.e. Z=2 ==> 2X magnification each time,
NOTE: Z=1/2 ==> zoom out)
```

Prompt user for window size:

```
> r0 := readstat("enter initial RADIUS of your window in x:");
> 1;
> if (type(evalf(r0),positive)=false) then
> print ('Please go back and pick positive value for r0')
> end if;
> #check that radius is positive
> rad:=r0: #initialize
> for k from 1 to N do
> #generate the zoom frames
> p[k]:=plot(sin(x),x=x0-rad..x0+rad):
> rad:=rad/Z:
> end do:
> for k from 1 to N do
> # reverse order and display so that initial frame is on top
> display(p[N+1-k]);
> end do;
> #pause and examine the windows
```