

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

Contents

1	Algebra	4
1.1	Polynomial tools (polynomials.mws)	4
1.1.1	Random number generator	4
1.1.2	Generate polynomial of selected degree	4
1.1.3	Plot	4
1.1.4	Solve, fsolve	4
1.1.5	Factor	4
1.1.6	Generate a polynomial with roots a[1], a[2], a[3]	4
1.1.7	Find an appropriate window to display all the roots of the polynomial	5
1.1.8	Generate a rational function with vertical asymptote at a[4]	5
1.1.9	Choose a window that shows all roots and vertical asymptote	5
1.2	Functions vs expressions (intersections.mws)	5
1.2.1	Functions	5
1.2.2	Expressions	5
1.2.3	Evaluations:	6
1.2.4	Intersections	6
1.2.5	Define the functions	6
1.2.6	Plot	6
1.2.7	Find intersections	6
1.2.8	Forcing more solutions	6
1.2.9	Question:	7
1.2.10	Question:	7
1.3	Simplify, assume (simplify.mws)	7
1.3.1	Absolute value, Piecewise function	7
1.4	Parametric Curves (parametric.mws)	8
1.4.1	Parametric representation of circle	8
1.4.2	Variations (speed, orientation, re-center)	8
1.4.3	Cycloid	8
1.4.4	cycloid.mws #link to another worksheet	8
1.4.5	Polar Curves	8
1.4.6	Polar Coords	8
1.4.7	Parametric Option	8
1.4.8	Exercise	9
1.4.9	Cycloids (cycloid.mws)	9

2	Calculus	10
2.1	Limits (limits.mws)	10
2.1.1	Diffence Quotients	10
2.1.2	Indeterminate Limits	10
2.1.3	Sequences and Series	10
2.1.4	Series	10
2.1.5	Exercise	11
2.2	Differentiate (derivatives.mws)	11
2.2.1	Derivatives	11
2.2.2	Expressions	11
2.2.3	Functions	11
2.2.4	Higher Order	11
2.2.5	Multivariable functions	11
2.3	Integrate (integration.mws)	11
2.3.1	Riemann sums	12
2.3.2	Animate	12
2.3.3	Exercise:	12
2.3.4	Integration commands	12
2.3.5	Tools in Student package	13
2.3.6	See also regionBetweenAndBothAxes.mws	13
2.3.7	User defines the functions and sets boundaries a,b,c,d.	13
2.3.8	Enter 2 functions:	13
2.3.9	Set colors	13
2.3.10	At the point the user can re-set any of the boundaries a,b,c,d provided that they are inside the previous window	14
2.3.11	Enter a value of k for the horizontal axis of revolution: $y = k$	14
2.3.12	Enter a value of j for the vertical axis of revolution: $x = j$	14
2.3.13	Display the region and the axes of revolution:	15
2.3.14	Create the solid of revolution about the horizontal axis:	15
2.3.15	Create the solid of revolution about the vertical axis:	16
2.3.16	Plot the top and bottom surfaces using cylindrical coordinates	16
2.3.17	Iterated Integrals	16
2.3.18	Indefinite integrals	16
2.3.19	Definite integrals	16
2.3.20	Cylindrical coordinates	16
2.3.21	Exercise: Can you show that the two surfaces are the same? Hint: color them, name them, and display them together.	16
2.4	3-D figures (3dFigures.mws)	17
2.4.1	Spacecurves	17
2.4.2	Surfaces	17
2.4.3	Plane	17
2.4.4	Line and plane	17
2.4.5	Point of intersection	17
2.4.6	Mailbox	17
2.4.7	Spheres and other surfaces that are NOT functions	17
2.4.8	Rectangular coordinates	17
2.4.9	Spherical coordinates	17
2.4.10	Plot the ellipsoid with shifted center:	17

3 Animations	18
3.1 Function variations (functionVariations.mws)	18
3.1.1 Choose a function to study	18
3.1.2 Vertical shift	18
3.1.3 Horizontal shift	18
3.1.4 Vertical scaling	18
3.1.5 Horizontal scaling	18
3.1.6 Exercise:	18
3.1.7 Reflections	19
3.1.8 Horizontal reflection	19
3.1.9 Vertical reflection	19
3.1.10 Absolute value effects	19
3.1.11 Absolute value applied to function	19
3.1.12 Absolute value applied to input variable	19
3.2 Trace tool (traceTool.mws)	19
3.2.1 User defines function and window to be plotted	19
3.2.2 Function (or parametric equations)	19
3.2.3 Window	20
3.2.4 Plot the curve	20
3.2.5 Add a "ticker" for coordinates	20
3.2.6 Add a point for the "cursor"	20
3.2.7 Insert option for user to request a point on graph	20
3.3 Secant line -> tangent line	21
3.4 Newton's method	21
3.5 Riemann sums -> area	21
3.6 Volume of Revolution	21

secantAnim_f01.mws Maple Workshop

December 18, 2002 Coreen Mett, Radford University

```
> restart;
```

1 Algebra

1.1 Polynomial tools (polynomials.mws)

Roots and Zeros of Polynomials

*(polynomials.mws) Coreen Mett, Radford University last modified
12/5/02*

```
> restart;  
> randomize(): #resets the seed by system clock
```

1.1.1 Random number generator

Used to generate random integer coefficients of a polynomial

```
> d:=9; #user selects bound for integer coefficients  
> Rand:=rand(-d..d); #defines a random number procedure selecting  
> integers between -d and d  
> a:=seq(Rand(),k=1..20);  
> #generate some coefficients
```

1.1.2 Generate polynomial of selected degree

User chooses the degree of polynomial to study

```
> deg:=3;  
> #user selects degree of polynomial to generate (odd is best)  
> p:=sum(a[k]*x^(k-1),k=1..deg+1); #generate polynomial
```

1.1.3 Plot

```
> plot(p,x=-5..5);  
> plot(p,x=-5..5,y=-10..10); #restrict y-axis window
```

1.1.4 Solve, fsolve

```
> solve(p=0,x);  
> evalf(%); #floating point format for previous output  
> fsolve(p=0,x); #find floating point solution(s)
```

1.1.5 Factor

```
> factor(p);
```

1.1.6 Generate a polynomial with roots a[1], a[2], a[3]

```
> p:=expand((x-a[1])*(x-a[2])*(x-a[3]));  
> solve(p=0,x);  
> plot(p,x=-5..5,y=-10..10);  
> factor(p);
```

1.1.7 Find an appropriate window to display all the roots of the polynomial

1.1.8 Generate a rational function with vertical asymptote at a[4]

```
> r:=p/(x-a[4]);
> plot(r,x=a[4]-5..a[4]+5);
> plot(r,x=a[4]-5..a[4]+5,y=-50..50,discont=true);
> factor(r);
```

1.1.9 Choose a window that shows all roots and vertical asymptote

```
> plot(r,x=-10..20,y=-50..500,discont=true);
> denom(r); #extract denominator of r
> numer(r);
> #numerator of r
```

1.2 Functions vs expressions (intersections.mws)

Functions and Expressions

(intersections.mws)

Coreen Mett, Radford University last modified 12/5/02

```
> restart;
```

1.2.1 Functions

```
> f:=x->sin(x) + exp(x) + Pi*x^2+ sqrt(5*x);
> f(3);
> evalf(f(3));
> f(3.);
> f(t);
> f(s);
> f(t+5);
> plot(f(x),x=-5..5);
> plot(f);
> #Maple does its best with defaults
> plot(f,x=0..5);
> #Maple doesn't understand
> plot(f,0..5);
> #correct version of above command
> plot(f(t),t=0..5);
> #another correct version of plot command. Note the "t" variable on the horizontal axis.
```

1.2.2 Expressions

```
> g:=sin(x) + exp(x) + Pi*x^2+ sqrt(5*x);
> #note that the x -> assignment was NOT used
> plot(g);
> plot(g,x=0..5);
> plot(g(x),x=0..5);
> #uses function notation for an EXPRESSION
```

1.2.3 Evaluations:

```
> g(3);
> subs(x=3,g);
> evalf(%);
> subs(x=3.,g);
```

1.2.4 Intersections

Find the intersection(s) of

$$y = \sin(px)$$

$$y = mx$$

where p is a random integer between 2 and 5, m is a random slope between 0 and 1.

```
> randomize();
> #set the random number seed by system clock
```

1.2.5 Define the functions

```
> r1:=rand(2..5);
> r2:=rand(1..9);
> p:=r1();
> P:=r2();
> Q:=r2();
> if (P = Q) then
>   Q := Q+1
> end if;
> S:=sort([P,Q]);
> m:=S[1]/S[2];
> #divides smaller integer by the larger
> f:=x->sin(p*x);
> g:=x->m*x;
```

1.2.6 Plot

Comment out the first 2 lines below if you use the random function generator above.

Un-comment the lines for demo purposes.

```
> #f:=x->sin(3*x);
> #g:=x->x/3;
> plot([f(x),g(x)],x=-Pi..Pi);
> plot([f(x),g(x)],x=-Pi..Pi,color=[red,blue],thickness=3, title="f(x)
> in red \n g(x) in blue");
> #improved plot
```

1.2.7 Find intersections

```
> solve(f(x)=g(x),x);
> evalf(%);
> fsolve(f(x)=g(x),x);
```

1.2.8 Forcing more solutions

Likely these were not satisfactory, not complete. Here's how to FORCE Maple to find more solutions

```
> s1:=fsolve(f(x)=g(x),x=0..1);  
> s2:=fsolve(f(x)=g(x),x=s1..2);
```

No new solution. Try a larger domain:

```
> s2:=fsolve(f(x)=g(x),x=0..4);  
> s3:=fsolve(f(x)=g(x),x=s2..4);  
> #look for more?
```

1.2.9 Question:

How do we know that we have found all the nonnegative solutions?

1.2.10 Question:

How should the solve commands above be edited if f and g were EXPRESSIONS rather than functions?

1.3 Simplify, assume (simplify.mws)

Simplifying Expressions

*(simplify.mws) Coreen Mett, Radford University last modified
12-11-02*

```
> restart;  
> #?simplify #get help on the simplify command  
> f:=x->1-sin(x)^2;  
> g:=x->1-cos(x)^2;  
> simplify(f(x));  
> simplify(g(x));  
> convert(g(x),sin);  
> expand(%);  
> simplify(g(x)-sin(x)^2);
```

1.3.1 Absolute value, Piecewise function

```
> f:=x->abs(x);  
> g:=x->piecewise(x>=0,x,-x); #another way to define abs(x)  
> f(x-3);  
> g(x-3);  
> simplify(f(x)-g(x));  
> simplify(g(x)/f(x));  
> plot([f(x),g(x)],x=-3..3);  
> simplify(f(x),assume=positive);  
> simplify(g(x),assume=positive);  
> h:=x->sqrt(x^2); #yet another way to define abs(x)  
> simplify(h(x));  
> simplify(h(x),assume=positive);
```

1.4 Parametric Curves (parametric.mws)

Parametric Plots

(*parametric.mws*) Coreen Mett, Radford University last modified
12-10-02

```
> restart;
```

1.4.1 Parametric representation of circle

```
> xp:=t->r*cos(t);  
> yp:=t->r*sin(t);  
> r:=3;  
> plot([xp(t),yp(t),t=0..2*Pi]);  
> plots[animatecurve]([xp(t),yp(t),t=0..2*Pi]);
```

1.4.2 Variations (speed, orientation, re-center)

```
> plots[animatecurve]([xp(2*t),yp(2*t),t=0..2*Pi]); #double speed  
> plots[animatecurve]([xp(-t),yp(-t),t=0..2*Pi]); #reverse  
> orientation  
> plots[animatecurve]([1+xp(t),yp(t)-3,t=0..2*Pi]); #shift center  
to  
> (1,-3)
```

1.4.3 Cycloid

Study the path of a marker attached to an arm of a rolling wheel. Examine cases where length of arm is

- a) less than radius of wheel
- b) exactly radius of wheel
- c) greater than radius of wheel

1.4.4 *cycloid.mws* #link to another worksheet

1.4.5 Polar Curves

Plot the cardioid $r = 1 + 2 \cos(\theta)$ in polar coordinates

1.4.6 Polar Coords

```
> plot(1+2*cos(t),t=0..2*Pi,coords=polar);  
> plots[animatecurve]([1+2*cos(t),t,t=0..2*Pi],coords=polar);
```

1.4.7 Parametric Option

```
> r:='r'; #reset r as variable  
> xt:=r*cos(t);  
> yt:=r*sin(t);  
> r:=1+2*cos(t);  
> plot([xt,yt,t=0..2*Pi]);  
> plots[animatecurve]([xt,yt,t=0..2*Pi]);
```

1.4.8 Exercise

Plot the Lissajous curves given by

$$x(t) = \cos(pt)$$

$$y(t) = \sin(qt)$$

for various integers p, q . Formulate conjectures about values of p, q that generate closed curves, and the number of loops in each.

1.4.9 Cycloids (cycloid.mws)

Cycloids

*(cycloid.mws) Coreen Mett, Radford University last modified
12-11-02*

Contains an animation showing how a cycloid is generated.

```
> restart;
> r:=1; b:=1.6;
> #user selects radius r of circle and radius b of marker arm
> x:=t->r*t-b*sin(t);
> generate the path of the cycloid
> y:=t->r-b*cos(t);
> X:=s->r*cos(s)+r*t;
> generate the circle
> Y:=s->r*sin(s)+1;
> N:=20;
> #user selects the number of frames in animation
> 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]:=plots[display](path[t],spoke[t],wheel[t]):
> end do:
> plots[display](p[k]$k=0..N,insequence=true,scaling=constrained,title=
> sprintf("b=%2.2f\n r=%2.2f",b,r));
> t:='t';
> plot([x(t),y(t),t=0..10]);
> plot([x(t),y(t)],t=0..10,title="x(t) in red\n y(t) in green");
> p1:=%:
> t1:=fsolve(y(t)=0,t=5..6);
> t2:=fsolve(y(t)=0,t=6..8);
> plot([t1,s,s=0..8],color=blue): pt1:=%:
> plot([t2,s,s=0..8],color=blue): pt2:=%:
> plots[display](p1,pt1,pt2,title="This is the interval of t on
which
> the point ab b moves backwards");
```

2 Calculus

2.1 Limits (limits.mws)

Limits

*(limits.mws) Coreen Mett, Radford University last modified
12-11-02*

```
> restart;
```

2.1.1 Diffence Quotients

```
> g:=x->sin(m*x);
> plots[animate]({g(x),m*x},x=-Pi..Pi,m=1..5,title="slope at (0,0)
> increases with m");
> limit(f(x),x=0);
> DiffQuotientG:=(g(0+h)-g(0))/h;
> limit(DiffQuotientG,h=0);
```

2.1.2 Indeterminate Limits

```
> f:=x->ln(x)/x;
> Limit(f(x),x=infinity);
> #inert form of limit
> value(%);
> limit(f(x),x=infinity);
> #accomplishes the same result, without the check of typing
> f:=x->sqrt(5+2*x^2)-sqrt(2*x^2+3*x);
> L:=limit(f(x),x=infinity);
> plot([f(x),L],x=1..200,color=[red,blue],title="visualize limit");
> limit(sin(x)/x,x=infinity);
> limit(exp(x)/x,x=infinity);
> limit((1+1/x)^(x),x=infinity);
> evalf(%);
```

2.1.3 Sequences and Series

```
> a:=n->3+(-1)^n/n;
> N:=10;
> #user selects numbe of points to plot
> S:=seq(a(n),n=1..N);
> Pts:=seq([n,a(n)],n=1..N);
> plot([Pts],style=point,symbol=circle,symbolsize=20); pSeq:=%:
> limit(a(n),n=infinity);
> pLim:=plot(3,x=1..N,color=blue):
> plots[display](pSeq,pLim,title="visualize limit of sequence");
```

2.1.4 Series

```
> Sum((1/3)^n,n=0..infinity); #geometric series, inert form of
Sum
> value(%);
> S:=N->sum((1/3)^k,k=0..N);
> Limit(S(n),n=infinity);
> #inert form of limit
```

```
> limit(S(n),n=infinity);  
> #same result as value(%) command
```

2.1.5 Exercise

Construct a visualization plot for the limit of the series as a limit of partial sums

2.2 Differentiate (derivatives.mws)

Derivatives

*(derivatives.mws) Coreen Mett, Radford University last modified
12-10-02*

```
> restart;
```

2.2.1 Derivatives

2.2.2 Expressions

```
> diff(sin(x^2+3)/exp(x^3+5),x);  
> simplify(%)  
> f1:=unapply(%,x);  
> f1(3);  
> evalf(%)
```

2.2.3 Functions

```
> f:=x->sin(x^2+3)/exp(x^3+5);  
> f1:=D(f);  
> f1(3);  
> evalf(%)
```

2.2.4 Higher Order

```
> f5:=(D@@5)(f); #fifth derivative  
> f5(3);  
> evalf(%)
```

2.2.5 Multivariable functions

```
> f:=(x,y)->sin(x^2+5*y)*exp(x*y^2);  
> fx:=D[1](f);  
> fx(1,0);  
> fy:=D[2](f);  
> VectorCalculus[Gradient](f(x,y),[x,y]);  
> grad_f:=%:  
> subs({x=1,y=0},grad_f);  
> fxy:=D[2,1](f);  
> #equivalent to D[2](D[1](f))
```

2.3 Integrate (integration.mws)

Integration

*(integration.mws) Coreen Mett, Radford University last modified
12-10-02*

```
> restart;
```

2.3.1 Riemann sums

```
> f:=x->sqrt(1-x^2);
> with(student):
> leftbox(f(x),x=0..1,8);
> rightbox(f(x),x=0..1,8);
```

2.3.2 Animate

```
> N:=20;
> #user selects the number of frames (maximum number of boxes)
> Plots:=seq(rightbox(f(x),x=0..1,k),k=1..N):
> plots[display](Plots,insequence=true);
```

2.3.3 Exercise:

Generate a sequence of "error" values displaying the difference between rectangular sum and actual area for n rectangles.

2.3.4 Integration commands

```
> f:=x->(3*x-5)/(5*x^2+7*x+9);
> int(f(x),x);
> Int(f(x),x);
> F:=value(%);
> #two-step process that achieves the same result, but displays integral as
well as solution
> dsolve(diff(y(x),x)=f(x));
> #adds constant of integration to antiderivative
> F:=rhs(%);
> F:=unapply(F,x);
> #convert from expression to function
> F(0);
```

Find the antiderivative of $f(x)$ that passes through $(0, 1)$:

```
> eqn:=F(0)=1;
> solve(eqn);
> _C1:=%;
> F(x);
```

Note that the same result could be obtained by using the DEFINITE INTEGRAL command:

```
> Int(f(t),t=0..x)+1;
> value(%);
> newF:=unapply(%,x);
> #find integral and make it a function
> newF(0);
> #check that initial condition holds
> simplify(%);
> newF(x);
> #same result as that obtained above.
```

2.3.5 Tools in Student package

```
> with(Student[Calculus1]);
> VolumeOfRevolution(cos(x)+1, x=0..4*Pi);
> VolumeOfRevolution(cos(x)+1, x=0..4*Pi,output=integral);
> VolumeOfRevolution(cos(x) + 1, x=0..4*Pi, output=plot);
```

2.3.6 See also regionBetweenAndBothAxes.mws

Revolve any region

between 2 curves about any axis, either vertical or horizontal, outside the region.

Coreen Mett, Radford University

last modified 12-11-02

Given a region bounded by two functions $f(x)$ and $g(x)$ that intersect at $A = [a,c]$ and $B = [b,d]$, revolve the region about any horizontal axis $y = k$ (where $k > d$ or $k < c$), and revolve the region about any vertical axis $x = j$ (where $j > b$ or $j < a$).

NOTE: It is required that $f(x)$ and $g(x)$ have **exactly** two points of intersection. Otherwise, modify boundaries.

```
> restart;
```

2.3.7 User defines the functions and sets boundaries a,b,c,d.

2.3.8 Enter 2 functions:

```
> f:=x->x^2;
> g:=x->x+2;
> #plot([f(x),g(x)],x=-10..10,y=-10..10); #check
```

2.3.9 Set colors

```
> fColor:=red;
> gColor:=green;
> hColor:=blue; #set color of horizontal axis
> vColor:=magenta; #set color of vertical axis
> shadeColor:=gray; #fill of region to revolve
> sx:=solve(f(x)=g(x),x); #find intersections
> evalf(sx);
> a:=min(evalf(sx));
> b:=max(evalf(sx));
> plot([f(x),g(x)],x=a..b,thickness=3,color=[fColor,gColor],title=
> sprintf("f is %s, g is %s",fColor,gColor)); p1:=%:
> fMin:=minimize(f(x),x=a..b):
> fMax:=maximize(f(x),x=a..b):
> gMin:=minimize(g(x),x=a..b):
> gMax:=maximize(g(x),x=a..b):
> low:=min(gMin,fMin):
> hi:=max(gMax,fMax):
> c:=low;
> d:=hi;
```

2.3.10 At the point the user can re-set any of the boundaries a,b,c,d provided that they are inside the previous window

```

> #a_new:=??;
> if (a_new > a and a_new < b) then
> a := a_new:
> else
> printf ("your choice must be between %2.2f and %2.2f", a,b):
> end if;
> #b_new:=??;
> if (b_new > a and b_new < b) then
> b := b_new:
> else
> printf ("your choice must be between %2.2f and %2.2f", a,b):
> end if;
> #c_new:=??;
> if (c_new > c and c_new < d) then
> c := c_new:
> else
> printf ("your choice must be between %2.2f and %2.2f", c,d):
> end if;
> #d_new:=??;
> if (d_new > c and d_new < d) then
> d := d_new:
> else
> printf ("your choice must be between %2.2f and %2.2f", c,d):
> end if;
> #plot([f(x),g(x),c,d],x=a..b,thickness=3,color=[fColor,gColor,black,b
> lack],title= printf("f is %s, g is %s \n horizontal lines at
x = c
> = %2.2f and x = d = %2.2f",fColor,gColor,c,d)); p1:=%:
> p1:=plot([f(x),g(x)],x=a..b,thickness=3,color=[fColor,gColor]):

```

2.3.11 Enter a value of k for the horizontal axis of revolution: $y = k$

```

> k:=5;
> if (k < hi and k > low) then
> print ("Your value of k is not acceptable.      It lies inside
the
> region.      Please try again.");
> end if;

```

2.3.12 Enter a value of j for the vertical axis of revolution: $x = j$

```

> j:=3;
> if (j < b and j > a) then
> print ("Your value of j is not acceptable.      It lies inside
the
> region.      Please try again.");
> end if;

```

2.3.13 Display the region and the axes of revolution:

```

> if (f((a+b)/2)>g((a+b)/2)) then
> T:=x->f(x);
> B:=x->g(x);
> TColor:=fColor;
> BColor:=gColor;
> else
> T:=x->g(x);
> B:=x->f(x);
> TColor:=gColor;
> BColor:=fColor;
> end if; #establish top and bottom functions
> left:=min(a,b,j):
> rt:=max(a,b,j):
> yMin:=min(c,d,k,fMin):
> yMax:=max(c,d,k,fMax):
> window:=left..rt,yMin..yMax:
> pBot:=plot(B(x),x=a..b,filled=true,color=white):
> pTop:=plot(T(x),x=a..b,filled=true,color=shadeColor):
> axh:=plot(k,x=a..b,color=hColor,thickness=3,linestyle=2):
> axv:=plot([j,t,t=low..hi],color=vColor,thickness=3,linestyle=2):
> plots[display](axh,axv,pBot,pTop,title="region to
> revolve",view=[window],axes=none): p2:=%:
to
> plots[display](p1,p2,title= sprintf("f is %s, g is %s\n Region
to
> revolve about displayed axes",fColor,gColor));
> #pause to observe region and axes

```

2.3.14 Create the solid of revolution about the horizontal axis:

Note: We are using parametric equations, and swapping the traditional positions of axes, so that the plane of the screen is the xy plane, and z points out toward the user.

```

> if k < a then
> R := x->T(x) - k;
> r := x-> B(x) - k;
> RColor:=TColor;
> rColor:=BColor;
> else
> R :=x-> B(x) - k;
> r :=x-> T(x) - k;
> RColor:=BColor;
> rColor:=TColor;
> end if; #set outer and inner radius
> plot3d([R(s)*sin(t),s,R(s)*(cos(t))+k],t=0..0.9*2*Pi,s=a..b,color=RCo
> lor): pOut:=%:
> plot3d([r(s)*sin(t),s,r(s)*(cos(t))+k],t=0..0.9*2*Pi,s=a..b,color=rCo
> lor): pIn:=%:
> hFace:=
> plot3d([0,s,t*T(s)+(1-t)*B(s)],s=a..b,t=0..1,color=shadeColor):
> with(plots):
> pHAx:=spacecurve([0,s,k],s=a..b,thickness=3,color=hColor):
> plots[display](hFace,pHAx,pOut,pIn,axes=normal,title=sprintf("region
> revolved about horizontal axis \n y = %2.2f",k));

> #use style=wireframe for a better view inside the solid

```

2.3.15 Create the solid of revolution about the vertical axis:

We first shift axes so that the axis of revolution is the y-axis. Then we use cylindrical coordinates to construct N faces. We also orient the axes so that the plane of the screen is the xy-plane.

```
> a1:=a-j;
> b1:=b-j;
> T1:=x->T(x+j);
> B1:=x->B(x+j);
> #plot([T1(x),B1(x)],x=a1..b1); #check
> N:=5; #number of faces desired
> delta:=2*Pi/N;
> for i from 0 to N do
> Face[i] := plot3d([r,delta*i,t*T1(r)+(1-t)*B1(r)], r=a1..b1,t=0..1,
> color=shadeColor, coords=cylindrical):
> end do;
> #plots[display](Face[n]$n=0..N);
> #optional display of several faces
> #plots[display](Face[n]$n=0..N,insequence=true);
> #animate
```

2.3.16 Plot the top and bottom surfaces using cylindrical coordinates

```
> p3:=plot3d([r,t,T1(r)],r=a1..b1,t=0..0.9*2*Pi,coords=cylindrical,colo
> r = TColor):
> p4:=plot3d([r,t,B1(r)],r=a1..b1,t=0..0.9*2*Pi,coords=cylindrical,colo
> r = BColor):
> pVax:=spacecurve([0,0,s],s=low..hi,thickness=3,color=vColor):
> plots[display](p3,p4,pVax,Face[0],title=sprintf("region revolved
> about vertical axis \n x = %2.2f",j));
> # Use style = wireframe for a better view
```

2.3.17 Iterated Integrals

2.3.18 Indefinite integrals

```
> Int(Int(x^2*y+5*tan(y),x),y);
> value(%);
```

2.3.19 Definite integrals

Find volume beneath the surface $z = 9 - x^2 - y^2$ and above the xy-plane.

```
> f:=(x,y)->9-x^2-y^2;
> plot3d(f(x,y),x=-3..3,y=-3..3,axes=boxed);
> V:=Int(Int(f(x,y),x=-sqrt(9-y^2)..sqrt(9-y^2)),y=-3..3);
> V:=value(%);
```

2.3.20 Cylindrical coordinates

```
> f:=(r,theta)->9-r^2;
> plot3d([r,t,f(r,t)],t=0..2*Pi,r=0..3,coords=cylindrical,axes=boxed);
```

2.3.21 Exercise: Can you show that the two surfaces are the same?

Hint: color them, name them, and display them together.

```
> Int(Int(f(r,t)*r,r=0..3),t=0..2*Pi);
> V2:=value(%);
```

Note that the answer is the same.

2.4 3-D figures (3dFigures.mws)

3-dimensional Figures

*(3dFigures.mws) Coreen Mett, Radford University last modified
12-10-02*

```
> restart; with(plots):
> #load the library that includes spacecurve command
```

2.4.1 Spacecurves

```
> r:=t->[sin(t),cos(t),t];
> #vector valued function whose trace is a curve in 3-D
> spacecurve(r(t),t=0..4*Pi);
> tubeplot(r(t),t=0..4*Pi);
> #plots the same curve, just thicker
> tubeplot(r(t),t=0..4*Pi,radius=.05);
> #somewhere in between
```

2.4.2 Surfaces

2.4.3 Plane

```
> plot3d(12-3*x-4*y,x=-4..4,y=-3..3,axes=normal); plane:=%:
```

2.4.4 Line and plane

```
> line:=spacecurve([4*t-5,3-t,2*t-1],t=-1..3,color=blue,thickness=3):
> display(line,plane);
```

2.4.5 Point of intersection

```
> P:=plottools[sphere]([7/5,7/5,11/5],.5):
> #claimed point of intersection
> display(P,plane,line);
```

2.4.6 Mailbox

```
> f:=(x,y)->sqrt(9-x^2);
> #construct a mailbox roof
> plot3d(f(x,y),x=-3..3,y=-5..5);
```

2.4.7 Spheres and other surfaces that are NOT functions

Plot the sphere $x^2 + y^2 + z^2 = 9$

2.4.8 Rectangular coordinates

```
> fTop:=(x,y)->sqrt(9-x^2-y^2);
> #top hemisphere
> fBottom:=(x,y)->-sqrt(9-x^2-y^2);
> plot3d({fTop(x,y),fBottom(x,y)},x=-3..3,y=-3..3,axes=boxed);
```

2.4.9 Spherical coordinates

```
> plot3d(3,theta=0..2*Pi,phi=0..Pi,coords=spherical,axes=boxed);
```

2.4.10 Plot the ellipsoid with shifted center:

Plot the surface defined by $\frac{(x-3)^2}{4} + \frac{(y+2)^2}{9} + (z-1)^2 = 1$

which represents an ellipsoid with center (3,-2,1) and axes of semi-length 2,3,1 respectively.

```

> xp:=2*sin(u)*cos(v)+3;
> yp:=3*sin(u)*sin(v)-2;
> zp:=cos(u)+1;
> #defines the parametric equations for the ellipse
> eqn:=(xp-3)^2/4 + (yp+2)^2/9+(zp-1)^2 = 1;
> #substitute parametric expressions into equation in order to check
> simplify(eqn);
> #verify that equation holds for all u,v
> plot3d([xp,yp,zp],u=0..2*Pi,v=0..2*Pi,axes=boxed,scaling=constrained)
> ;
> #pause and verify claims for center and lengths of axes.

```

3 Animations

3.1 Function variations (functionVariations.mws)

Animating Function Properties

(functionVariations.mws) Coreen Mett, Radford University last modified 12-10-02

```

> restart;with(plots):
> #load the library that contains the animate command

```

3.1.1 Choose a function to study

User defines the function.

```

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

```

3.1.2 Vertical shift

```

> animate(f(x)+t,x=-3..3,t=0..5);
> #pause and play animation

```

3.1.3 Horizontal shift

```

> animate(f(x+t),x=-10..3,t=0..5);
> #pause and play animation

```

3.1.4 Vertical scaling

```

> animate(t*f(x),x=-3..3,t=1..5);
> #pause and play animation

```

3.1.5 Horizontal scaling

```

> animate(f(t*x),x=-3..3,t=1..5);
> #pause and play animation

```

3.1.6 Exercise:

Add a "marker" to trace a point as the animation proceeds. (See vertex_animation.mws)

3.1.7 Reflections

```
> f:=x->sin(x+1)+1/2;
```

3.1.8 Horizontal reflection

```
> plot([f(x),f(-x)],x=-3..3,color=[red,blue]);
```

3.1.9 Vertical reflection

```
> plot([f(x),-f(x)],x=-3..3,color=[red,blue]);
```

3.1.10 Absolute value effects

3.1.11 Absolute value applied to function

```
> plot([f(x),abs(f(x))],x=-3..3,color=[red,blue]);
```

3.1.12 Absolute value applied to input variable

```
> plot([f(x),f(abs(x))],x=-3..3,color=[red,blue],thickness=[1,2]);
```

3.2 Trace tool (traceTool.mws)

Trace Tool

*(traceTool.mws) Coreen Mett, Radford University last modified
12/16/02*

The purpose of this animation is to emulate the tools on a graphic calculator, tracing a curve and displaying coordinates as the pixels are generated. The key tools are **animatecurve** and **sprintf**.

User defines the function (parametrically) to be graphed, and chooses the number N of frames to display.

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

3.2.1 User defines function and window to be plotted

3.2.2 Function (or parametric equations)

```
> xp:=t->sin(t);
> yp:=t->cos(t);
```

For the plot of a function $y = f(x)$, set

```
f := x -> ??
```

```
xp := t -> t
```

```
yp := t -> f(t)
```

as shown in the example below.

```
> #f:= x-> sin(x);
> #xp:=t->t;
> #yp:=t->f(t);
```

3.2.3 Window

Enter the interval $[a,b]$ desired for independent variable. The other variable window will be determined by a best fit.

```
> a:=0;
> b:=2*Pi;
> window:=t=a..b;
```

3.2.4 Plot the curve

```
> animatecurve([xp(t),yp(t),window]); #parametric plot
> #pause and play animation
```

slow it down

```
> animatecurve([xp(t),yp(t),window],frames=50);
> #pause and play animation
```

3.2.5 Add a "ticker" for coordinates

```
> N:=20; #number of frames desired
> B:=b*k/N;
> #adjust window for N steps
> anim_window:=t=0..B:
> p:=seq(plot([xp(t),yp(t),anim_window],title=sprintf("(%f,
> %f)",evalf(xp(B)),evalf(yp(B))),k=1..N):
> display(p[j]$j=1..N,insequence=true);
> #pause and play animation
```

Different format for printed values:

```
> q:=seq(plot([xp(t),yp(t),anim_window],title=sprintf("(%a,
> %a)",(xp(B)),(yp(B))),k=1..N):
> display(q[j]$j=1..N,insequence=true);
> #pause and play animation
```

3.2.6 Add a point for the "cursor"

```
> r:=seq(plot([[xp(B),yp(B)]],style=point,symbol=circle,color=blue,symb
> olsize=15),k=1..N):
> q:=seq(plot([xp(t),yp(t),anim_window],title=sprintf("(%a,
> %a)",(xp(B)),(yp(B))),k=1..N):
> s:=seq(display(r[k],q[k]),k=1..N):
> display(s[j]$j=1..N,insequence=true);
> #pause and play animation
```

3.2.7 Insert option for user to request a point on graph

```
> xv:=readstat("Enter value, followed by ',' and ENTER:"):
> 10;
> if (evalf(xv)<evalf(a) or evalf(xv)>evalf(b)) then
> sprintf("Please re-enter a value between %f and %f",a,b);
> xv:=readstat("Enter value again:");
> end if;
> ### pause and make sure xv is entered
> Pt:=plot([[xp(xv),yp(xv)]],style=point,symbol=circle,color=blue,symbo
> lsize=15):
> display(Pt,q[N],title=sprintf("value=%f\n(%f,
> %f)",xv,evalf(xp(xv)),evalf(yp(xv))));
```

```
> xv:=2;  
> #if a new value is entered here, just go back and re-execute the Pt:=...  
command.
```

3.3 Secant line -> tangent line

(see http://www.radford.edu/~cmett/ictcm2001/animations/secantAnim_f01.mws)

3.4 Newton's method

(see <http://www.radford.edu/~cmett/ictcm2001/animations/newtonZoom.mws>)

3.5 Riemann sums -> area

(see material in integration.mws above)

3.6 Volume of Revolution

(see http://www.radford.edu/~cmett/ictcm2001/animations/volOfRev_s01.mws)