

# Tuesday Morning Homework

Mike May, S.J.

```
> with(plots);
```

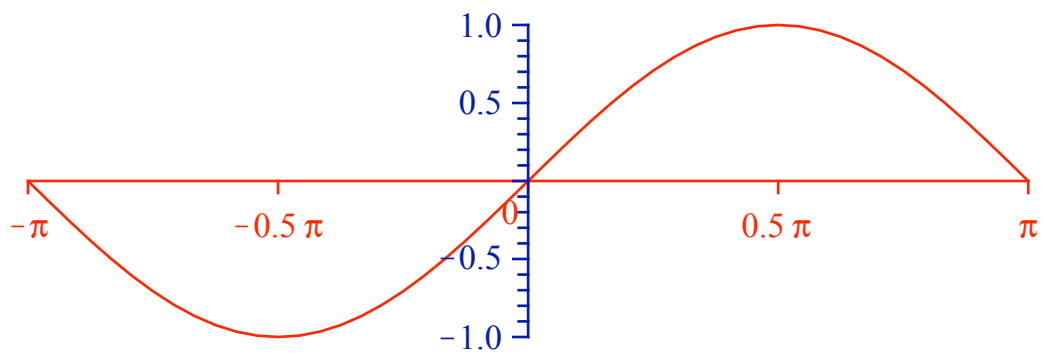
```
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d,  
conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot,  
display, fieldplot, fieldplot3d, gradplot, gradplot3d, graphplot3d, implicitplot,  
implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot,  
listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple,  
odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d,  
polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions,  
setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]
```

(1)

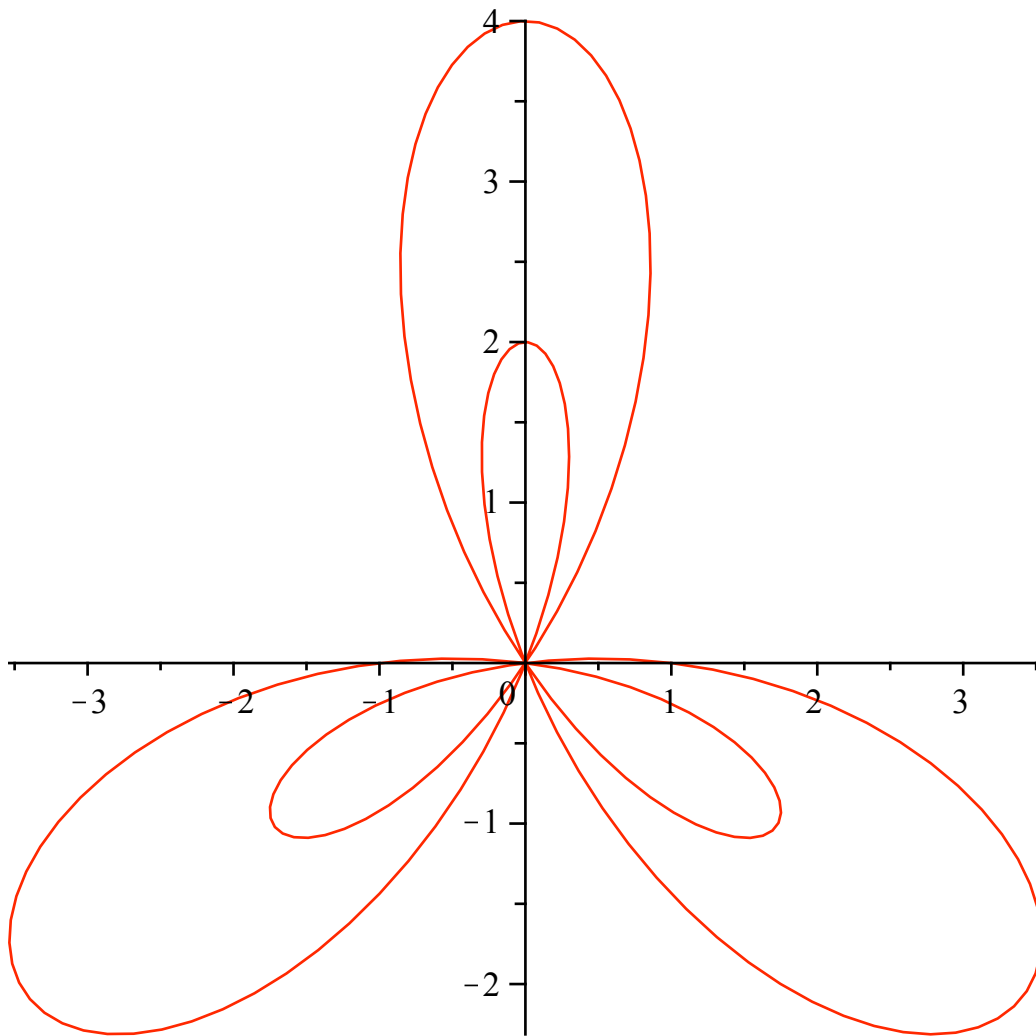
## ▼ Animating polar plots

This morning someone asked how to animate curves. This was asked in the context of doing polar graphing. When in doubt I go to help and figured it would be in the plots package. I found the `animate plots` command which is demonstrated below.

```
> animatecurve(sin(x), x=-Pi..Pi, frames=50,  
tickmarks=[spacing(Pi/2), default], scaling=constrained);
```

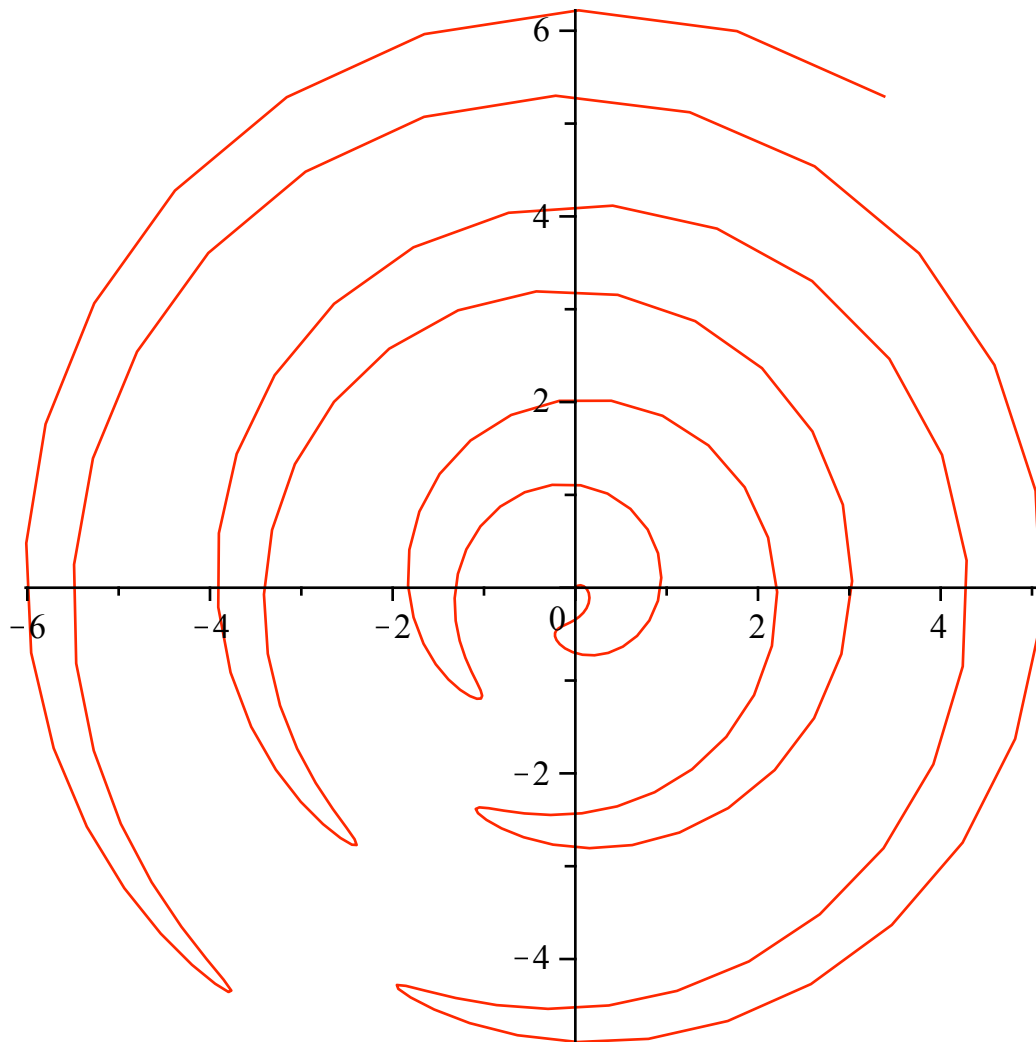


```
> animatecurve([1-3*sin(3*t),t,t=0..2*Pi],coords=polar, frames=50,numpoints=200);
```



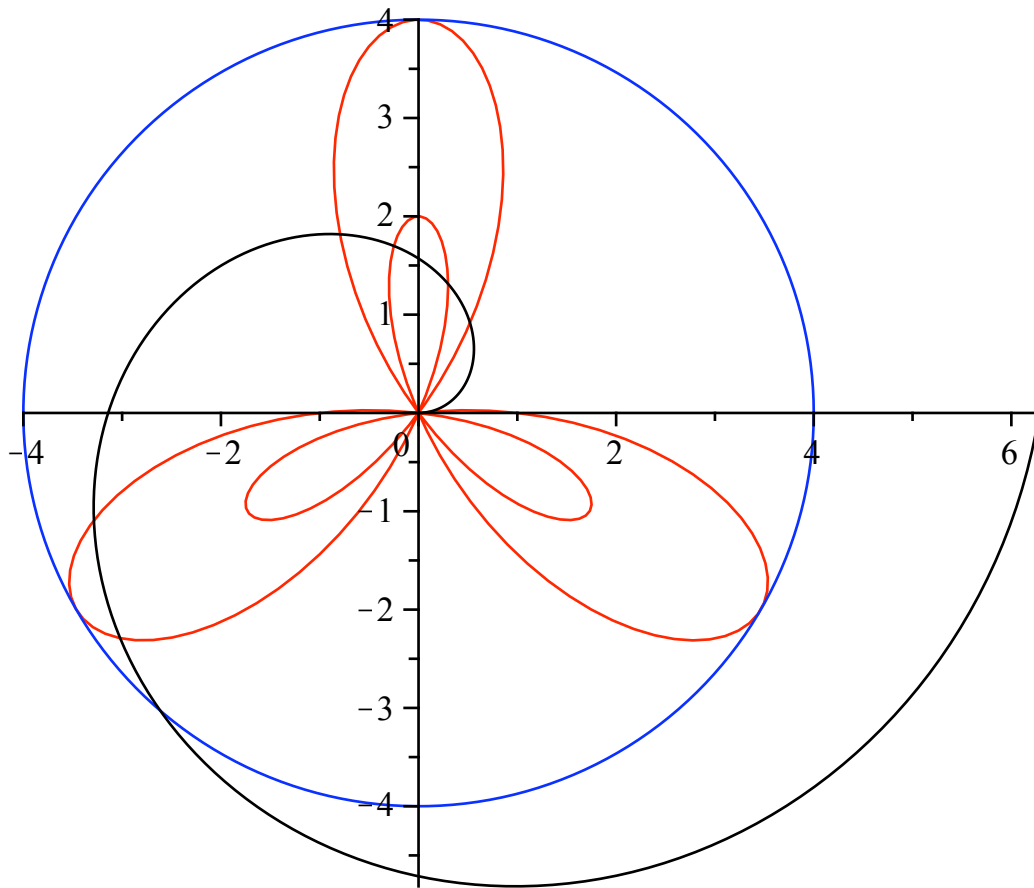
The reason I used the polar form is that Maple seems to do strange things in the standard form for polar plots. (It does theta as a function of r.)

```
> animatecurve(1-3*sin(3*t),t=0..2*Pi,coords=polar, frames=50,  
numpoints=200);
```



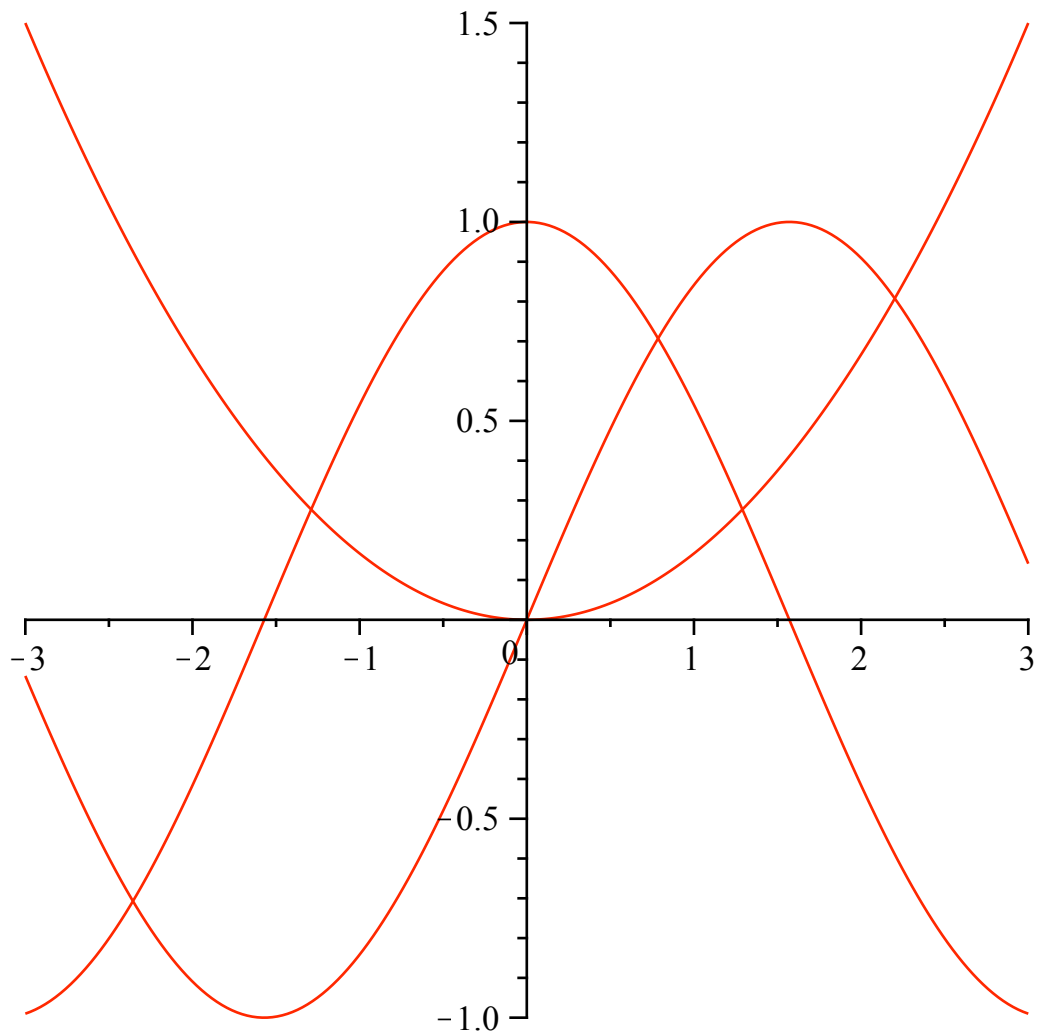
To animate several curves together I also bring in the display command.

```
> curvel := animatecurve([1-3*sin(3*t),t,t=0..2*Pi],
  coords=polar, frames=50,numpoints=200,color=red):
  spiral := animatecurve([t,t,t=0..2*Pi],
  coords=polar, frames=50,numpoints=200,color=black):
  circ := animatecurve([4,t,t=0..2*Pi],
  coords=polar, frames=50,numpoints=200,color=blue):
  display({curvel, spiral, circ},scaling=constrained);
```



If I am only doing cartesian plots, life is easier.

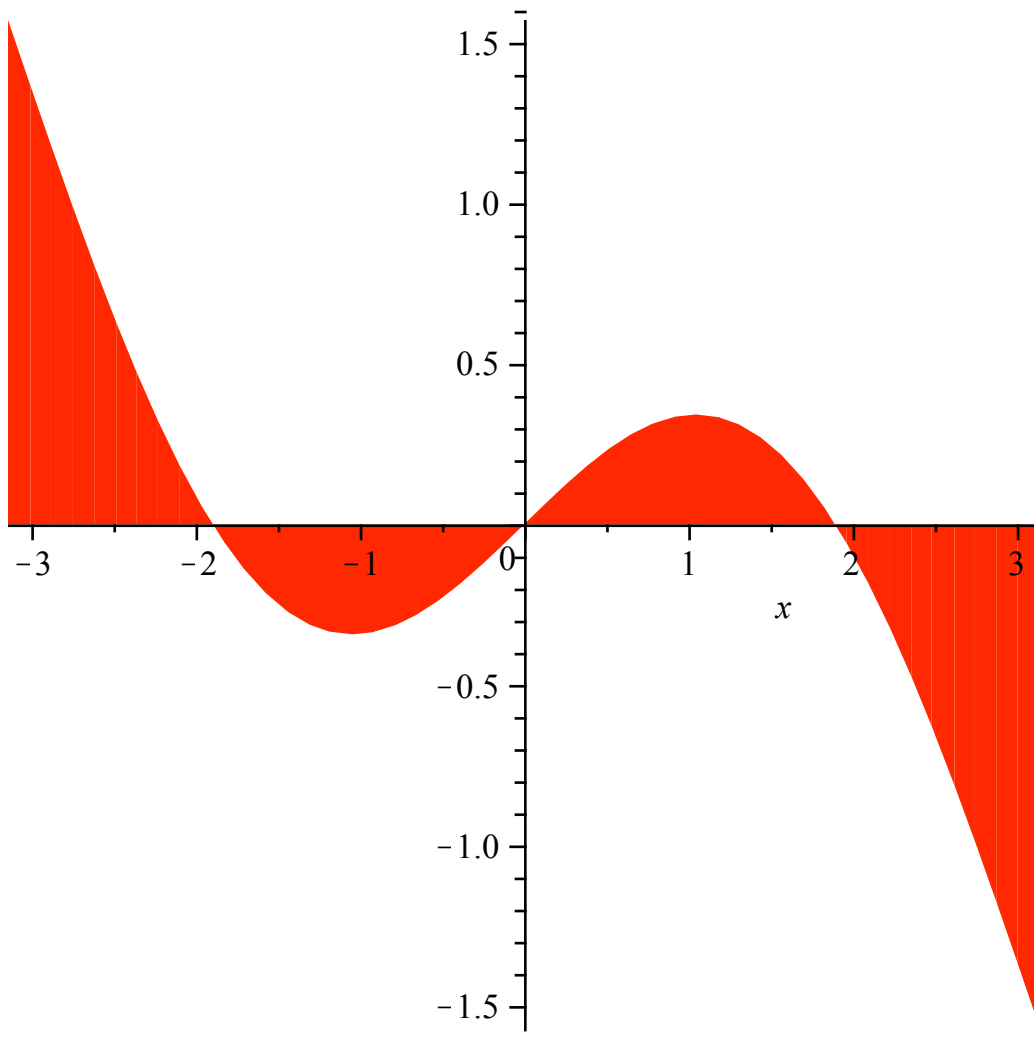
```
> animatecurve({sin(x), cos(x), x^2/6}, x=-3..3, frames=50,  
numpoints=200);
```



## Shading regions

The plot command has a filled parameter which is defaulted to false. It fills the region between the curve and the x axis.

```
> plot(sin(x)-x/2,x=-Pi..Pi, filled=true);
```

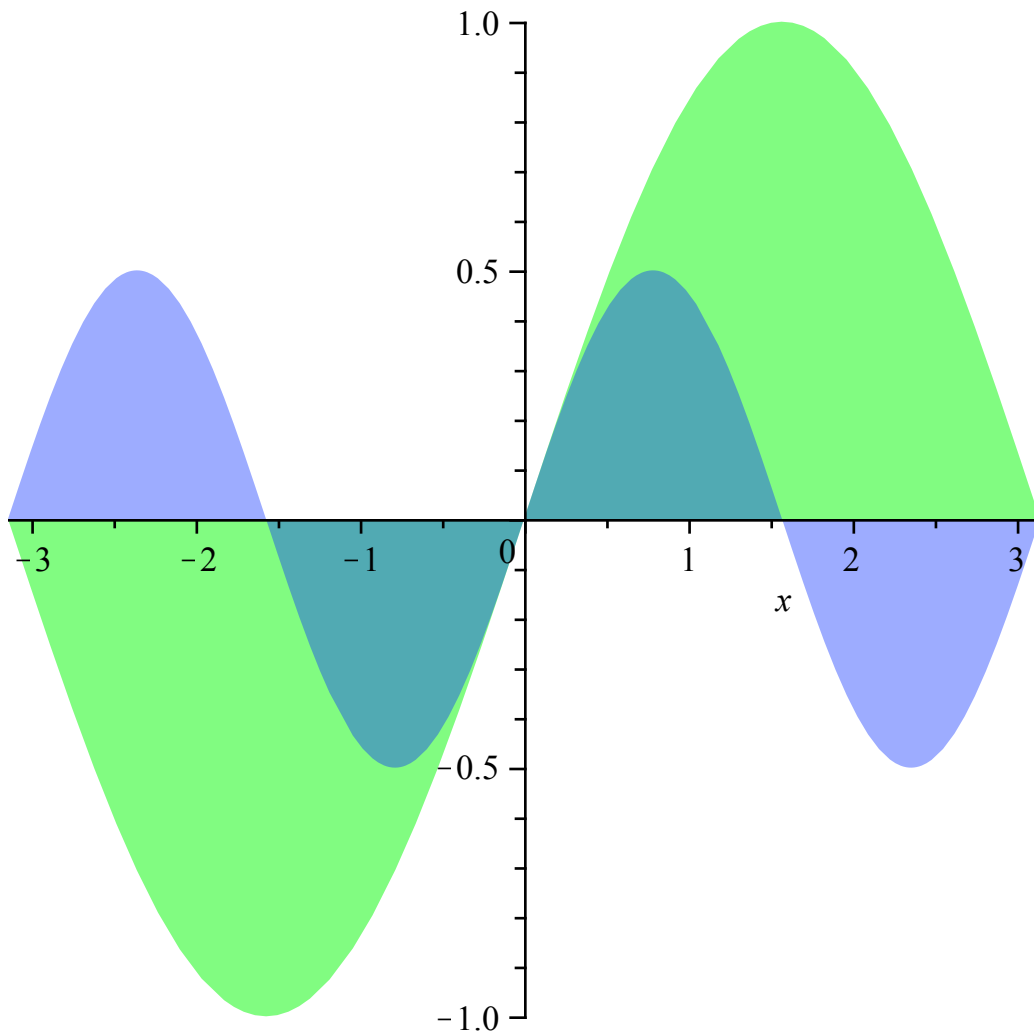


There does not seem to be a good command for the area between curves. You can do a good approximation with display.

```
> p1 := plot(sin(x), x=-Pi..Pi,
  filled=true, color=green, transparency=.5);
p2 := plot(sin(2*x)/2, x=-Pi..Pi,
  filled=true, color=blue, transparency=.6);
display({p1, p2});
```

*p1 := PLOT(...)*

*p2 := PLOT(...)*



We also can get some interesting stuff with the Student[Calculus1] package;

```
> with(Student[Calculus1]);
[AntiderivativePlot, AntiderivativeTutor, ApproximateInt, ApproximateIntTutor, ArcLength, (2.1)
  ArcLengthTutor, Asymptotes, Clear, CriticalPoints, CurveAnalysisTutor, DerivativePlot,
  DerivativeTutor, DiffTutor, ExtremePoints, FunctionAverage, FunctionAverageTutor,
  FunctionChart, FunctionPlot, GetMessage, GetNumProblems, GetProblem, Hint,
  InflectionPoints, IntTutor, Integrand, InversePlot, InverseTutor, LimitTutor,
  MeanValueTheorem, MeanValueTheoremTutor, NewtonQuotient, NewtonsMethod,
  NewtonsMethodTutor, PointInterpolation, RiemannSum, RollesTheorem, Roots, Rule,
  Show, ShowIncomplete, ShowSteps, Summand, SurfaceOfRevolution,
  SurfaceOfRevolutionTutor, Tangent, TangentSecantTutor, TangentTutor,
  TaylorApproximation, TaylorApproximationTutor, Understand, Undo,
  VolumeOfRevolution, VolumeOfRevolutionTutor, WhatProblem ]
```

```
> FunctionChart(sin(x) - x/2, x = -Pi..Pi);
```

The Chart of  
 $f(x) = \sin(x) - 1/2 * x$   
on the Interval  $[-\pi, \pi]$

