```
> #Title: PolarAnimate
  #Author: Gord Clement, August 2010
  #Description: For a given function and interval, this animates the function in polar coordinates.
   #                The radial arm is animated as the animation moves through the interval.

      #                When the radius is negative, the function is plotted with dots as if the radius
      were positive
  #Usage:
  #Call : PolarAnimate( function, interval)
  #function: polar function to be used for the animation
  #interval: interval to be used in the animation, entered
      in standard Maple notation, ie [a,b] would be entered θ = a ..b
   #           Note, the variable used in the interval will be interpretted as θ in the animation

  PolarAnimate := proc(expr, range)
  #local declarations
  local signOfFunction, var, minX, maxX, function, maxR;
  #function that is -1 is the given value is negative, 0 otherwise
  signOfFunction := x→piecewise(x < 0,-1);
  #extract variable and endpoints
   var := op( 1, range );
  minX := evalf(op(1, op(2, range ) ));
  maxX := evalf(op(2, op(2, range ) ));
  #calculate maximum radius length for length of radial arm
  maxR := maximize(abs(expr), range);
  function := var→expr;
  #create animation
  plots[animate](plot, [[[expr·cos(var), expr·sin(var), var = minX..s], [expr
      ·signOfFunction(expr)· cos(var), expr·signOfFunction(expr)·sin(var), var = minX..s],
      [maxR· t·cos(s), maxR· t·sin(s), t = 0..1]], color = [red, black, black], thickness = [2, 4,
      1], linestyle = [solid, dot, dash]], s = minX..maxX, scaling = constrained, frames = 100)
   end proc:
> PolarAnimate(3·sin(2·θ), θ = 0..2·Pi)

> PolarAnimate(1-2· cos(θ), θ = 0..2·Pi)

> PolarAnimate(1 + 2· sin(θ), θ = 0..2·Pi)

> PolarAnimate(sin(0.4·θ), θ = 0..10·Pi)

>

> PolarAnimate(cos(e·θ), θ = 0..20·Pi)

>
```