```
> restart;
  with( linalg ):
  with( plots ):
Warning, new definition for norm
Warning, new definition for trace
```

Comparison of Iterative Methods for Function Minimization

Prepared by Douglas B. Meade (meade@math.sc.edu)
13 November 1995

WWS URL: http://www.math.sc.edu/~meade/math524/worksheets/comp-iter.ms

Updated to Maple V, Release 4, by Douglas B. Meade (5 November 1996)

Maple Implementation of Three Iterative Minimization Procedures

Newton's Method for Function Minimization

```
> NewtonM := x -> convert( evalm( x - inverse( Hf(x) ) &*
  gradf(x) ), vector ):
```

Method of Steepest Descent

```
> Steep := proc(x)
    local phi, phiP, t, v;
    v := gradf(x);
    phi := f( evalm( x-t*v ) );
    phiP := -dotprod( gradf( evalm( x -t*v ) ), v );
    if phiP<>0 then
      t := op(1,[fsolve( phiP=0, t )])
    else
      t := 0
    fi;
    convert( evalm( x - t*gradf(x) ), vector );
  end:
```

Broyden's Method for Function Minimization

```
> Broyden := proc(x)
    local d, xx;
    global DD;
    xx := evalm( x - inverse( DD ) &* gradf( x ) );
    d := evalm( xx - x );
    if dotprod( d, d ) > 0 then
```

```
    DD := evalm( DD + ( gradf( xx ) &* transpose( d ) ) /
dotprod( d, d ) )
  fi;
  convert( evalm( xx ), vector );
end:
```

Definition of the function to be minimized (f), its gradient (gradf) and its Hessian (Hf) and the initial point (x.0)

Example 1: Problem #4 (p. 129 of Peressini, Sullivan, Uhl)

```
> f := x -> 2*x[1]^4+x[2]^2-4*x[1]*x[2]+5*x[2];
gradf := x -> vector( [ 8*x[1]^3-4*x[2], 2*x[2]-4*x[1]+5 ] );
Hf := x -> matrix( [ [ 24*x[1]^2, -4 ], [ -4, 2 ] ] );
```

$$f := x \to 2\,x_1^{\,4} + x_2^{\,2} - 4\,x_1\,x_2 + 5\,x_2$$

$$gradf := x \to vector([8\,x_1^{\,3} - 4\,x_2, 2\,x_2 - 4\,x_1 + 5])$$

$$Hf := x \to matrix([[24\,x_1^{\,2}, -4], [-4, 2]])$$

```
> x.0 := vector( [ 0, 0] );
```

$$x0 := [0, 0]$$

Example 2: Example 3.2.4 (p. 101 of Peressini, Sullivan, Uhl)

```
> f := X -> X[1]^2 + a^2 * X[2]^2;
gradf := X -> vector( [ 2*X[1], 2*a^2*X[2] ] );
Hf := X -> matrix( [ [ 2, 0 ], [ 0, 2*a^2 ] ] );
```

$$f := X \to X_1^{\,2} + a^2\,X_2^{\,2}$$

$$gradf := X \to vector([2\,X_1, 2\,a^2\,X_2])$$

$$Hf := X \to matrix([[2, 0], [0, 2\,a^2]])$$

```
> a := 6:
> x.0 := vector( [ evalf( a/sqrt(1+a^2) ), evalf( 1/a/sqrt(1+a^2)
) ] );
```

$$x0 := [.9863939240, .02739983122]$$

Example 3: Problem 10 (p. 130 of Peressini, Sullivan, Uhl)

```
> f := x -> x[1]^4/4 + x[2]^2;
gradf := x -> vector( [ x[1]^3, 2*x[2] ] );
Hf := x -> matrix( [ [ 3*x[1]^2, 0 ], [ 0, 2 ] ] );
```

$$f := x \to \frac{1}{4}\,x_1^{\,4} + x_2^{\,2}$$

$$gradf := x \to vector([x_1^{\,3}, 2\,x_2])$$

$$Hf := x \to matrix([[3\,x_1^{\,2}, 0], [0, 2]])$$

```
> x.0 := vector( [ 1, 0 ] );
```

$$x0 := [1, 0]$$

Chose one of the following matrices -- or enter your own -- for Broyden's Method

```
> DD := matrix( [ [ 1, 0 ], [ 0, 1 ] ] );
```

$$DD := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
> #DD := Hf( x.0 );
```

Enter the number of iterations to be computed

```
> N := 5:
```

Compute -- and display -- the first N iterates for each of the three methods.

DO NOT MODIFY THESE COMMANDS

```
> n.0 := evalm( x.0 ):
  s.0 := evalm( x.0 ):
  b.0 := evalm( x.0 ):
> for k from 0 to N-1 do
    print( k, map( evalf@evalm, [ n.k, s.k, b.k ] ) );
    n.(k+1) := NewtonM( n.k );
    s.(k+1) := Steep( s.k );
    b.(k+1) := Broyden( b.k );
  od:
  print( N, map( evalf@evalm, [ n.N, s.N, b.N ] ) );
```

$$0, [[1., 0], [1., 0], [1., 0]]$$

$$1, [[.6666666667, 0], [0, 0], [0, 0]]$$

$$2, [[.4444444444, 0], [0, 0], [0, 0]]$$

$$3, [[.2962962963, 0], [0, 0], [0, 0]]$$

$$4, [[.1975308642, 0], [0, 0], [0, 0]]$$

$$5, [[.1316872428, 0], [0, 0], [0, 0]]$$

```
>
```

Assemble the lists of function values as well as the lists of points for each of the three methods.

```
> Nval := plot( [ seq( [k,f(n.k)], k=0..N ) ], color=red,
```

```
title='Value: Newton's Method' ):
Sval := plot( [ seq( [k,f(s.k)], k=0..N ) ], color=green,
title='Value: Steepest Descent' ):
Bval := plot( [ seq( [k,f(b.k)], k=0..N ) ], color=blue,
title='Value: Broyden's Method' ):
> Npt := plot( [ seq( convert(n.k,list), k=0..N ) ], color=red,
title='Path: Newton's Method' ):
Spt := plot( [ seq( convert(s.k,list), k=0..N ) ], color=green,
title='Path: Steepest Descent' ):
Bpt := plot( [ seq( convert(b.k,list), k=0..N ) ], color=blue,
title='Path: Broyden's Method' ):
```

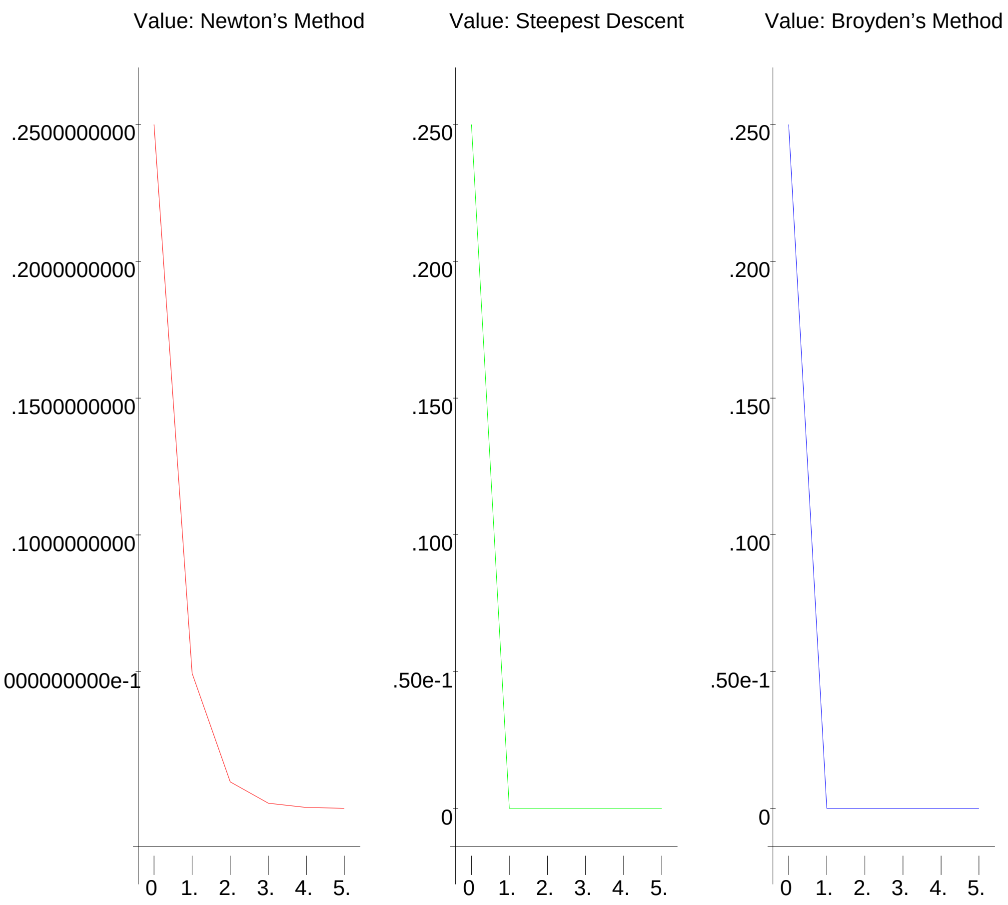
Plot the function values of the iterates of each method.

Release 4 Update

Side-by-side plots are now possible

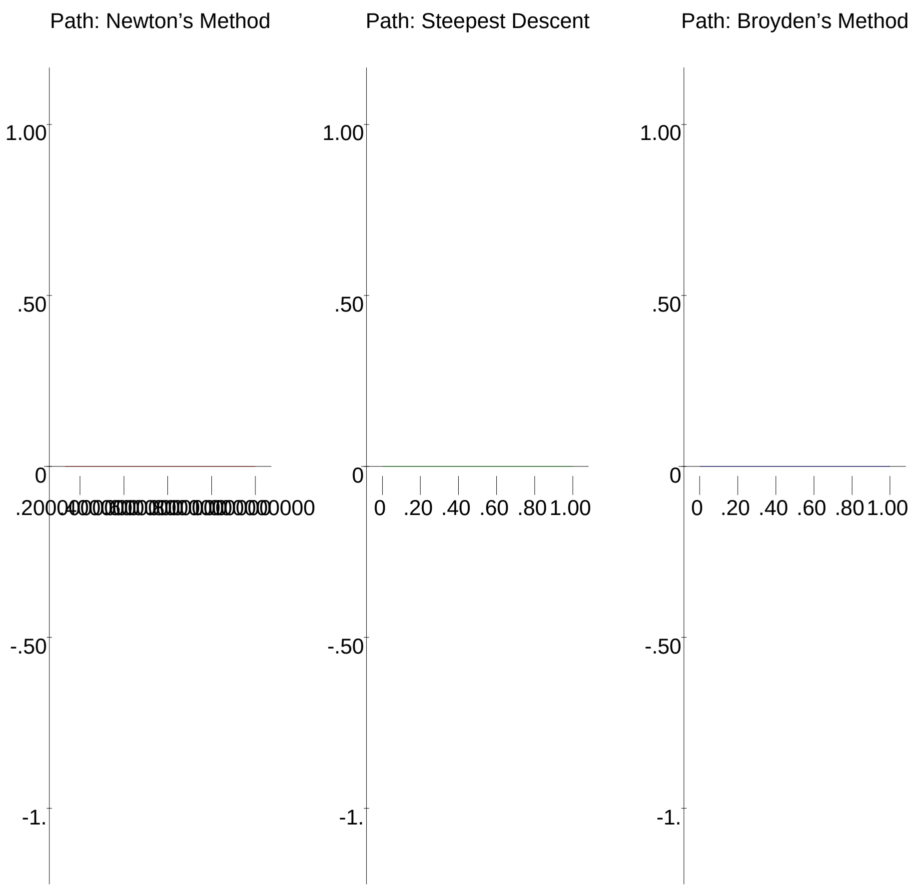
Plot the function values of the iterates of each method.

```
> display( array(1..3, [Nval, Sval, Bval] ) );
```



Plot the path defined by the points in the iteration sequences.

```
> display( array(1..3, [Npt, Spt, Bpt] ) );
```



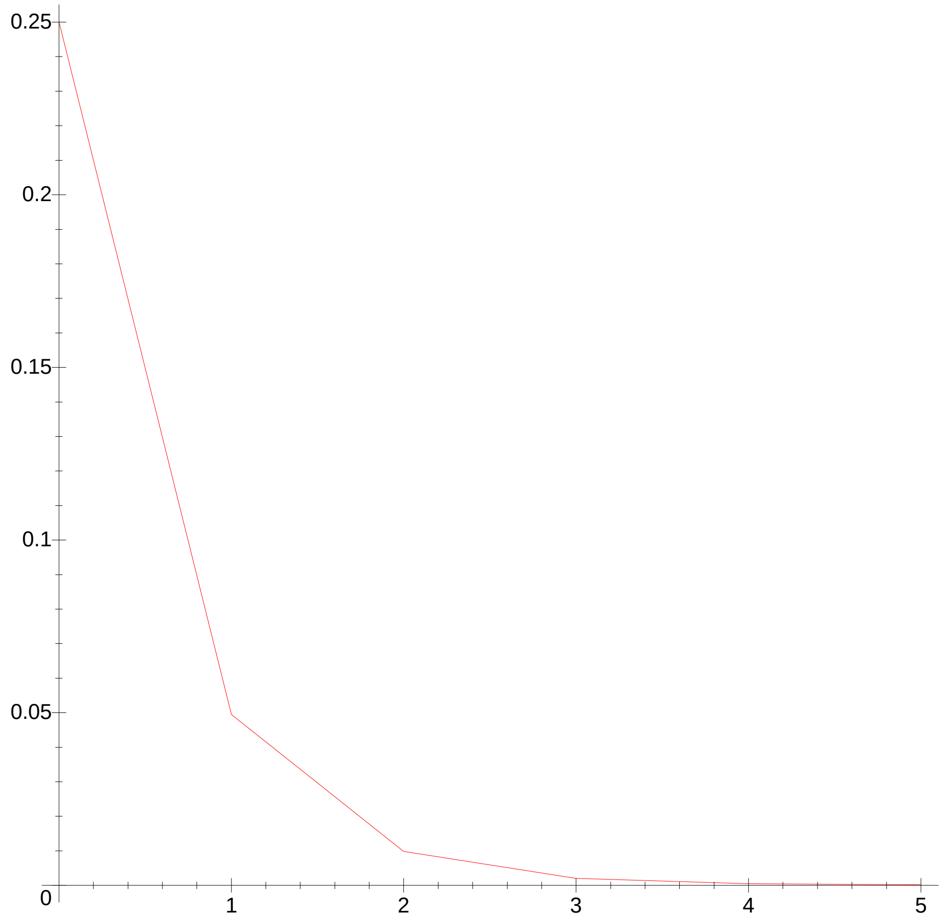
>

Old -- Release 3

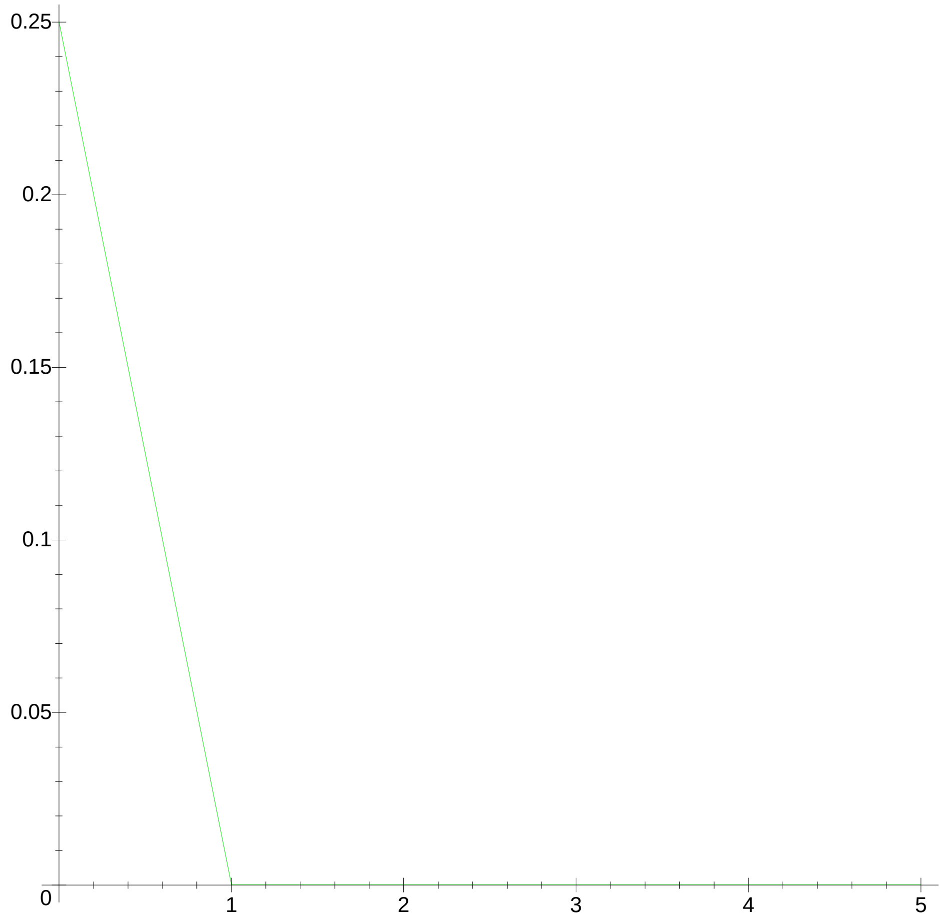
Plot the function values of the iterates of each method.

```
> display( Nval, title=`Function Values: Newton's Method` );
  display( Sval, title=`Function Values: Steepest Descent` );
  display( Bval, title=`Function Values: Broyden's Method` );
  display( {Nval,Sval,Bval}, title=`Function Values` );
```

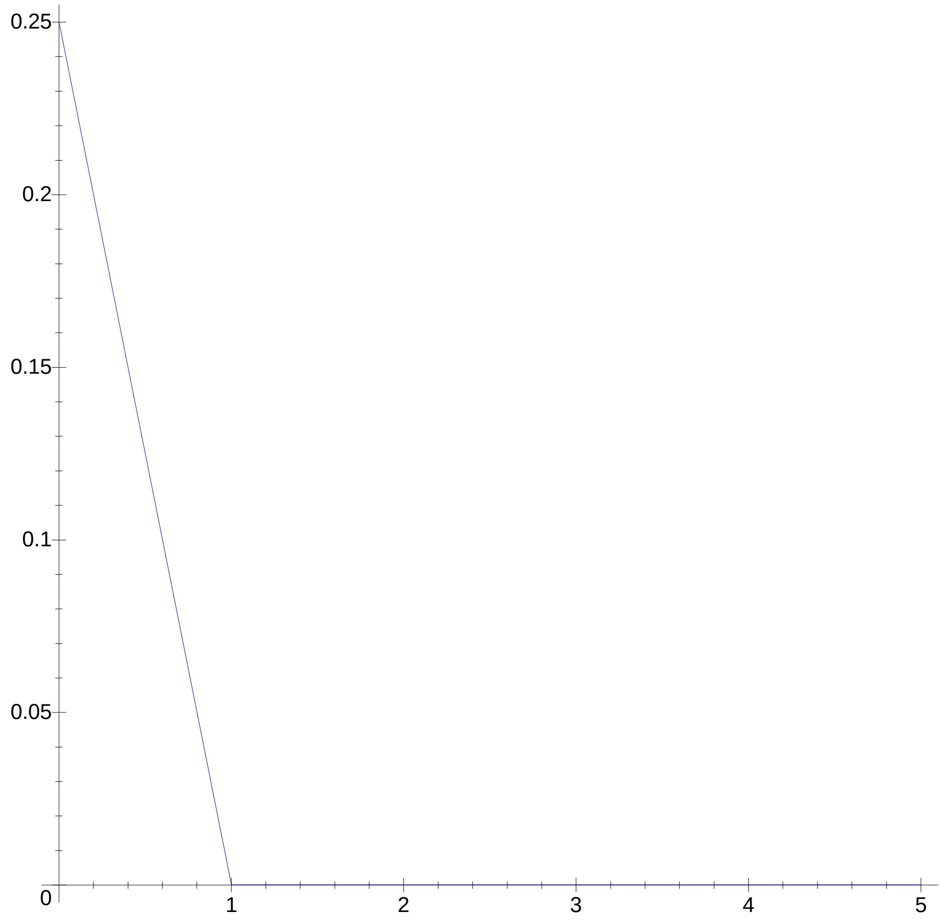
Function Values: Newton's Method



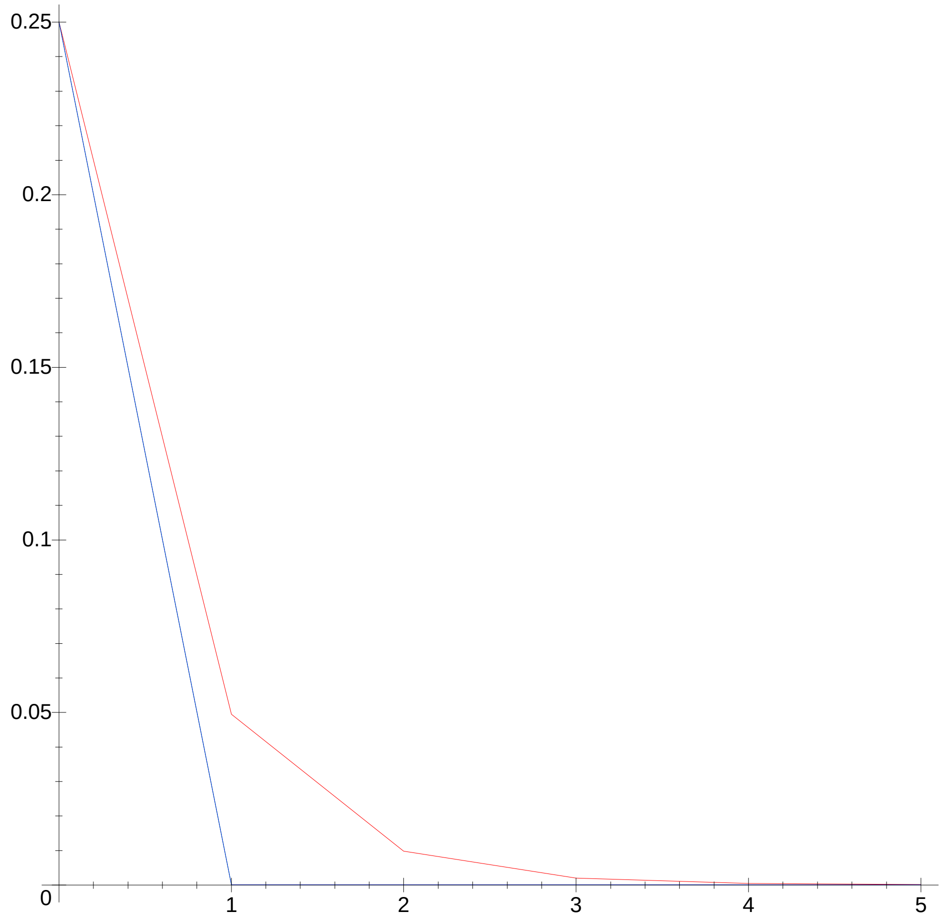
Function Values: Steepest Descent



Function Values: Broyden's Method



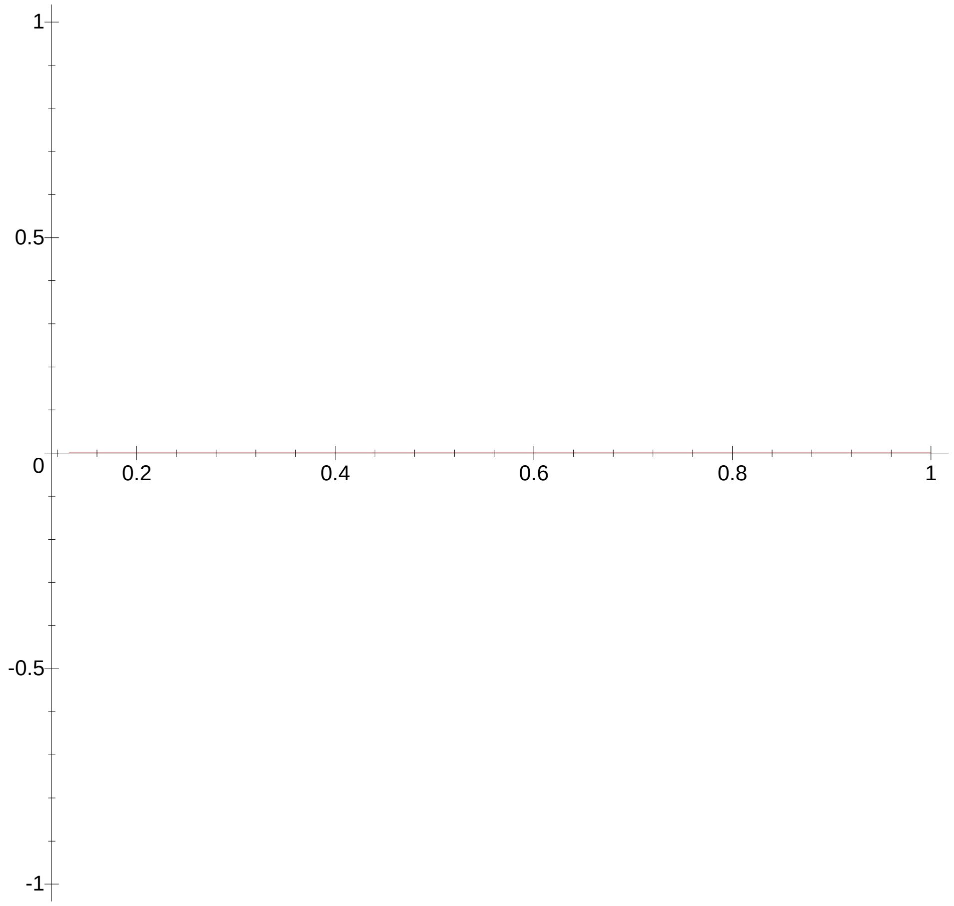
Function Values



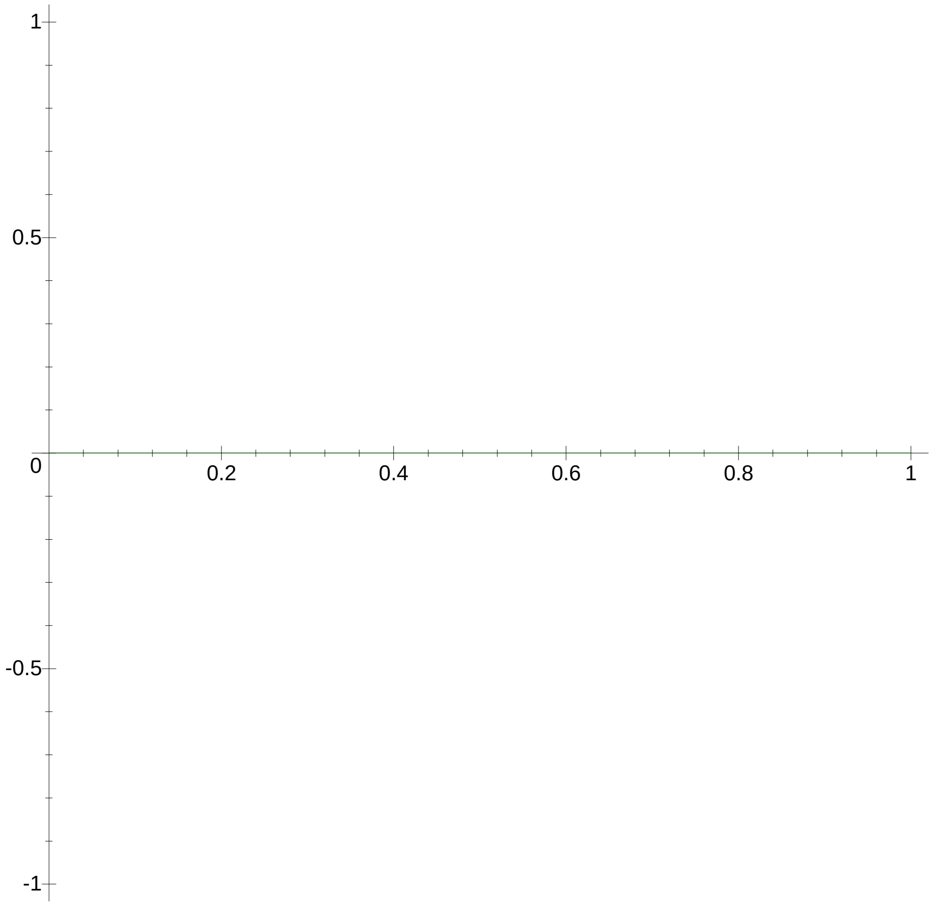
Plot the path defined by the points in the iteration sequences.

```
> display( Npt, title='Trajectory: Newton's Method' );
  display( Spt, title='Trajectory: Steepest Descent' );
  display( Bpt, title='Trajectory: Broyden's Method' );
  display( { Npt, Spt, Bpt }, title='Trajectories' );
```

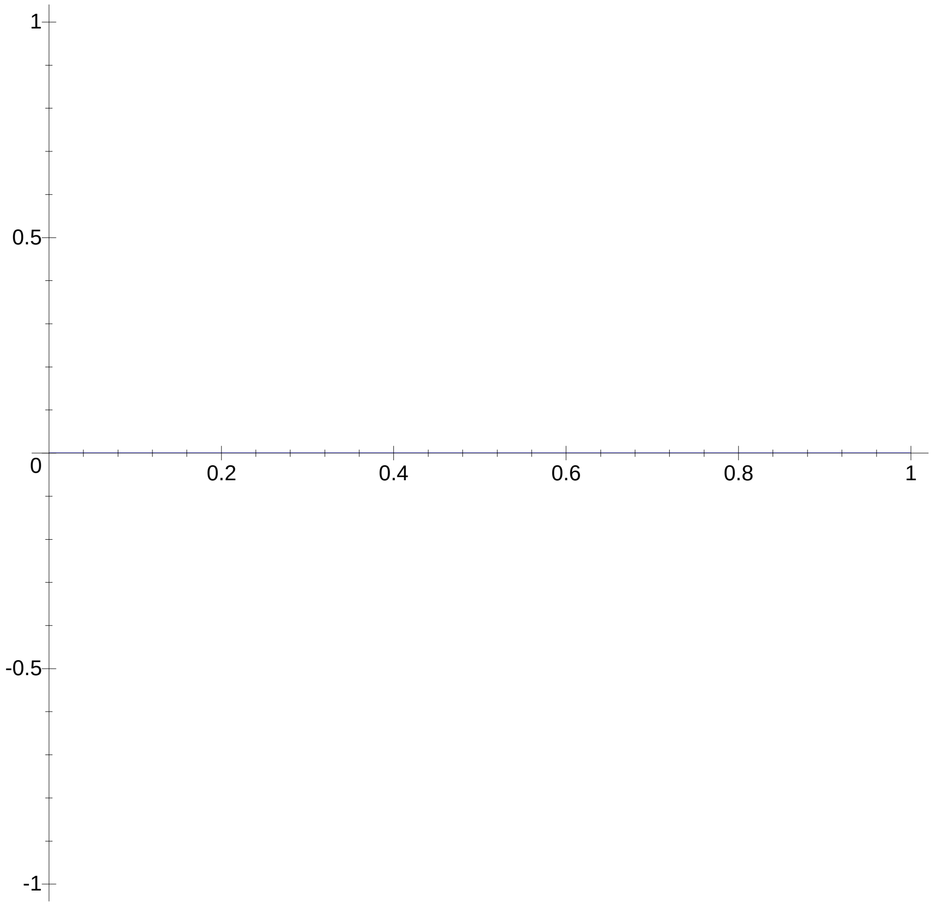
Trajectory: Newton's Method



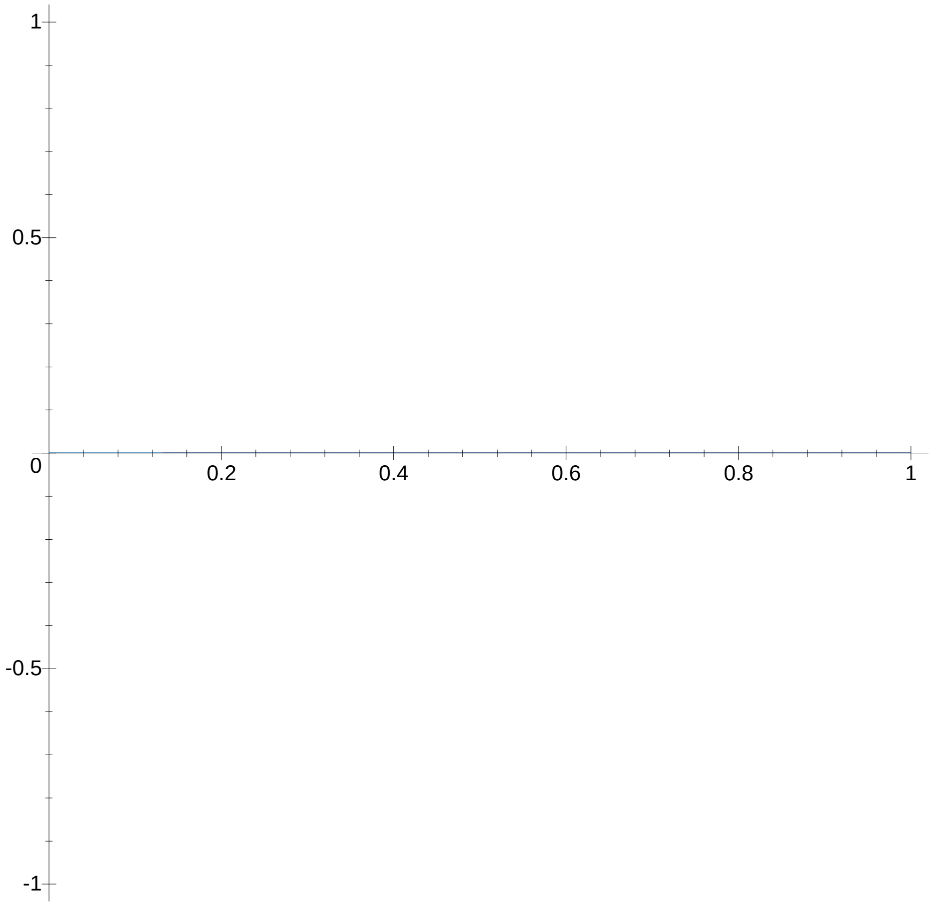
Trajectory: Steepest Descent



Trajectory: Broyden's Method



Trajectories



>

END OF WORKSHEET

>

>