```
> 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

---

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).

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 ] ] );
>
```

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 ] ] );
```

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

$$gradf := X \to \mathrm{vector}\left(\left[2\, X_1,\, 2\, a^2\, X_2\right]\right)$$

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

```
> a := 6:
>
```

Example 3: Problem 10 (p. 130 of Peressine, 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 ] ] );
>
```

Specify the initial point (x.0).

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

$$x0 := [.9863939240 \quad .02739983122]$$

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

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

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

Enter the number of iterations to be computed

```
> N := 15;
```

$$N := 15$$

```
>
```

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 ] ) );
```

$$0, [[.9863939240 \quad .02739983122], [.9863939240 \quad .02739983122], [.9863939240 \quad .02739983122]]$$

$$1, [[0 \quad 0], [.9330753335 \quad -.02591875925], [-.9863939240 \quad -1.945388017]]$$

$$2, [[0 \quad 0], [.8826388290 \quad .02451774524], [-.9330753340 \quad 1.840231907]]$$

$$3, [[0 \quad 0], [.8349286220 \quad -.02319246175], [.3287097900 \quad -.648288753]]$$

$$4, [[0 \quad 0], [.7897973452 \quad .02193881512], [.3\ 10^{-9} \quad -.3\ 10^{-9}]]$$

$$5, [[0 \quad 0], [.7471055968 \quad -.02075293325], [-.296059727\ 10^{-10} \quad .583895575\ 10^{-10}]]$$

$$6, [[0 \quad 0], [.7067215105 \quad .01963115305], [-.339266055\ 10^{-11} \quad .669108055\ 10^{-11}]]$$

$$7, [[0 \quad 0], [.6685203477 \quad -.01857000973], [0 \quad .2\ 10^{-20}]]$$

$$8, [[0 \quad 0], [.6323841128 \quad .01756622529], [-.1881147720\ 10^{-21} \quad .371004133\ 10^{-21}]]$$

$$9, [[0 \quad 0], [.5982011876 \quad -.01661669973], [.504863600\ 10^{-22} \quad -.995703212\ 10^{-22}]]$$

$$10, [[0 \quad 0], [.5658659884 \quad .01571849960], [.1\ 10^{-31} \quad -.7\ 10^{-31}]]$$

$$11, [[0 \quad 0], [.5352786375 \quad -.01486885112], [.3290339930\ 10^{-32} \quad -.648928153\ 10^{-32}]]$$

$$12, [[0 \quad 0], [.5063446573 \quad .01406512929], [-.392029520\ 10^{-33} \quad .773169334\ 10^{-33}]]$$

$$13, [[0 \quad 0], [.4789746757 \quad -.01330485220], [-.3\ 10^{-42} \quad .6\ 10^{-42}]]$$

$$14, [[0 \quad 0], [.4530841529 \quad .01258567082], [-.4762339\ 10^{-45} \quad .9392389\ 10^{-45}]]$$

$$15, [[0 \quad 0], [.4285931174 \quad -.01190536447], [.7499611\ 10^{-48} \quad -.14790896\ 10^{-47}]]$$

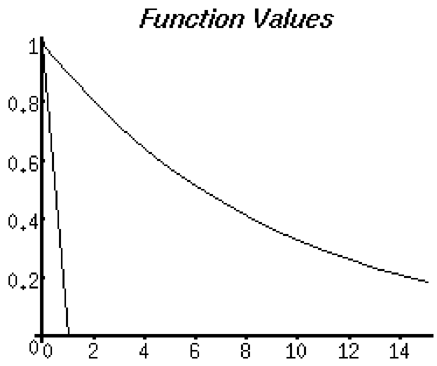
```
>
```

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 ):
  Sval := plot( [ seq( [k,f(s.k)], k=0..N ) ], color=green ):
  Bval := plot( [ seq( [k,f(b.k)], k=0..N ) ], color=blue ):
> Npt := plot( [ seq( n.k, k=0..N ) ], color=red ):
  Spt := plot( [ seq( s.k, k=0..N ) ], color=green ):
  Bpt := plot( [ seq( b.k, k=0..N ) ], color=blue ):
```

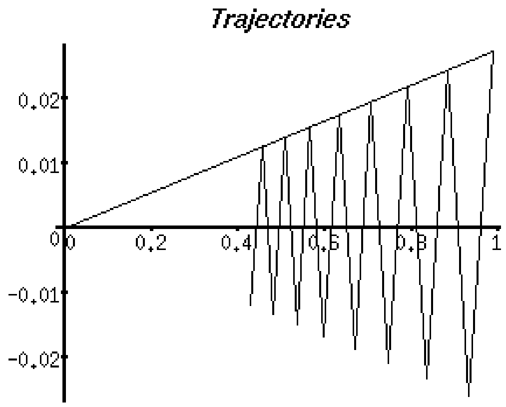
Plot the function values of the iterates of each method.

```
> 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' );
```



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' );
```



END OF WORKSHEET

>