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

```
Warning: new definition for   norm
Warning: new definition for   trace
```

Solution of Homework Problems from Chapter 3 of Peressini, Sullivan, and Uhl

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

WWS URL: http://www.math.sc.edu/~meade/math524/worksheets/ch3hw2.ms

---

```
>
```

Problem #8

This problem can be solved using the Steepest Descent iteration defined in steep.ms

To use this, we have to enter the function to be minimized, f, and its gradient, df :

```
> f := x -> 2*x[1]^4+x[2]^2-4*x[1]*x[2]+5*x[2];
```

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

```
> df := x -> [ 8*x[1]^3-4*x[2], 2*x[2]-4*x[1]+5 ];
```

$$df := x \rightarrow \left[8\, x_1^{\,3} - 4\, x_2,\, 2\, x_2 - 4\, x_1 + 5 \right]$$

```
>
```

together with the intial point, x.0, and the number of iterates, N :

```
> x.0 := [0,0];
```

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

```
> N := 2;
```

$$N := 2$$

Compute the next N iterates by Steepest Descent.

```
> for k from 0 to N-1 do
    y.k := f(x.k);
    v.k := df(x.k);
    phi.k := f( evalm( x.k-t*v.k ) );
    phiP.k := -dotprod( df( evalm( x.k -t*v.k ) ), v.k );
```

```
  t.k := fsolve( phiP.k=0, t );
  x.(k+1) := convert( evalm( x.k - t.k*df(x.k) ), list );
od;
```

$$y0 := 0$$

$$v0 := [0, 5]$$

$$\phi 0 := 25\,t^2 - 25\,t$$

$$phiP0 := 50\,t - 25$$

$$t0 := .5000000000$$

$$x1 := [0, -2.500000000]$$

$$y1 := -6.250000000$$

$$v1 := [10.00000000, 0]$$

$$\phi 1 := 20000.00000\,t^4 - 6.250000000 - 100.0000000\,t$$

$$phiP1 := 80000.00000\,t^3 - 100.0000000$$

$$t1 := .1077217345$$

$$x2 := [-1.077217345, -2.500000000]$$

```
>
```

All relevant data can be obtained from the above output.

NOTE: For additional insight into this example, increase the value of N to, say, 10, then repeat the calculations. The output will be more difficult to read, but the following plots show that the function values are decreasing and the descent directions are orthogonal.

```
> vals := seq( evalf( [k, f(x.k) ]), k=0..N );
  plot( [ vals ], style=POINT, title=`Plot of k vs. f( x.k )` );
```

What do we expect to see in the list/plot of descent directions?

```
> pts := seq( evalf( x.k ), k=0..N );
  plot( [ pts ], title=`Steepest Descent Iterates` );
```

>

Problem #9a

```
> g := vector( 3, [ x^2+y^2+z^2-3, x^2+y^2-z-1, x+y+z-3 ] );
```

$$g := [x^2 + y^2 + z^2 - 3 \quad x^2 + y^2 - z - 1 \quad x + y + z - 3]$$

```
> G := 1/2 * dotprod( g, g );
```

$$G := \frac{1}{2}(x^2 + y^2 + z^2 - 3)^2 + \frac{1}{2}(x^2 + y^2 - z - 1)^2 + \frac{1}{2}(x + y + z - 3)^2$$

>

Problem #10

For this problem I will use the NewtonM and NewtonZ functions from the previous HW (ch3hw1.ms).

Note that NewtonZ is what is referred to, in this problem, as Newton's Method and NewtonM is modified Newton's Method.

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

Before we do any computations, let's think about this problem. The given function, f, is a scalar-valued function of two variables. The gradient of f will be a 2-dimensional vector. It makes no sense to talk about the inverse of the gradient of f; therefore, it makes no sense to use the (standard) Newton's Method to iterate towards a zero of this function. (Recall that Newton's Method is used to solve systems of n equations in n variables.)

```
>
```

For the modified Newton's Method we expect the iterates to converge to a minimizer of the function f

```
> f := x -> x[1]^4/4 + x[2]^2;
```

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

The gradient and Hessian of f are also needed

```
> gradf := x -> matrix( 2,1, [ x[1]^3, 2*x[2] ] );
```

$$gradf := x \to \mathrm{matrix}\left(2, 1, \left[x_1^3, 2\, x_2\right]\right)$$

```
> Hf := x -> matrix( 2,2,  [ [3*x[1]^2, 0], [0, 2] ] );
```

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

```
>
```

The initial guess is x[0] :

```
> x[0] := [ 1,0 ];
```

$$x_0 := [1, 0]$$

```
>
```

The first two iterates are thus:

```
> x[1] := NewtonM( x[0] );
```

$$x_1 := \begin{bmatrix} \frac{2}{3} & 0 \end{bmatrix}$$

```
> x[2] := NewtonM( x[1] );
```

$$x_2 := \begin{bmatrix} \frac{4}{9} & 0 \end{bmatrix}$$

This should be enough to show a general pattern for additional iterates.

```
>
```

```
>
```

Problem #11

For this problem we will use the same ideas/methods as in Problem #8.

```
> f := x -> 2*x[1]^2+x[2]^2-x[1]*x[2];
```

$$f := x \rightarrow 2\, x_1^{\,2} + x_2^{\,2} - x_1\, x_2$$

```
> df := x -> [ 4*x[1]-x[2], 2*x[2]-x[1] ];
```

$$df := x \rightarrow \left[4\, x_1 - x_2,\ 2\, x_2 - x_1 \right]$$

```
> x.0 := [1,4];
```

$$x0 := [1, 4]$$

```
> N := 2;
```

$$N := 2$$

Compute the next N iterates by Steepest Descent.

```
> for k from 0 to N-1 do
    y.k := f(x.k);
    v.k := df(x.k);
    phi.k := expand( f( evalm( x.k-t*v.k ) ) );
    phiP.k := -dotprod( df( evalm( x.k -t*v.k ) ), v.k );
    t.k := solve( phiP.k=0, t );
    x.(k+1) := convert( evalm( x.k - t.k*df(x.k) ), list );
  od;
```

$$y0 := 14$$

$$v0 := [0, 7]$$

$$\phi 0 := 14 - 49\, t + 49\, t^2$$

$$phiP0 := -49 + 98\, t$$

$$t0 := \frac{1}{2}$$

$$x1 := \left[1, \frac{1}{2} \right]$$

$$y1 := \frac{7}{4}$$

$$v1 := \left[\frac{7}{2}, 0\right]$$

$$\phi 1 := \frac{7}{4} - \frac{49}{4}\,t + \frac{49}{2}\,t^2$$

$$phiP1 := -\frac{49}{4} + 49\,t$$

$$t1 := \frac{1}{4}$$

$$x2 := \left[\frac{1}{8}, \frac{1}{2}\right]$$

END OF WORKSHEET

>