Newton set up to work for N functions

```
> restart;
> with(LinearAlgebra);
  v := Vector(2);
  w := Vector(2);
```

[*&x*, *Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, *BilinearForm*, *CARE*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*, *ColumnDimension*, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*, *ConditionNumber*, *ConstantMatrix*, *ConstantVector*, *Copy*, *CreatePermutation*, *CrossProduct*, *DARE*, *DeleteColumn*, *DeleteRow*, *Determinant*, *Diagonal*, *DiagonalMatrix*, *Dimension*, *Dimensions*, *DotProduct*, *EigenConditionNumbers*, *Eigenvalues*, *Eigenvectors*, *Equal*, *ForwardSubstitute*, *FrobeniusForm*, *GaussianElimination*, *GenerateEquations*, *GenerateMatrix*, *Generic*, *GetResultDataType*, *GetResultShape*, *GivensRotationMatrix*, *GramSchmidt*, *HankelMatrix*, *HermiteForm*, *HermitianTranspose*, *HessenbergForm*, *HilbertMatrix*, *HouseholderMatrix*, *IdentityMatrix*, *IntersectionBasis*, *IsDefinite*, *IsOrthogonal*, *IsSimilar*, *IsUnitary*, *JordanBlockMatrix*, *JordanForm*, *KroneckerProduct*, *LA_Main*, *LUDecomposition*, *LeastSquares*, *LinearSolve*, *LyapunovSolve*, *Map*, *Map2*, *MatrixAdd*, *MatrixExponential*, *MatrixFunction*, *MatrixInverse*, *MatrixMatrixMultiply*, *MatrixNorm*, *MatrixPower*, *MatrixScalarMultiply*, *MatrixVectorMultiply*, *MinimalPolynomial*, *Minor*, *Modular*, *Multiply*, *NoUserValue*, *Norm*, *Normalize*, *NullSpace*, *OuterProductMatrix*, *Permanent*, *Pivot*, *PopovForm*, *QRDecomposition*, *RandomMatrix*, *RandomVector*, *Rank*, *RationalCanonicalForm*, *ReducedRowEchelonForm*, *Row*, *RowDimension*, *RowOperation*, *RowSpace*, *ScalarMatrix*, *ScalarMultiply*, *ScalarVector*, *SchurForm*, *SingularValues*, *SmithForm*, *StronglyConnectedBlocks*, *SubMatrix*, *SubVector*, *SumBasis*, *SylvesterMatrix*, *SylvesterSolve*, *ToeplitzMatrix*, *Trace*, *Transpose*, *TridiagonalForm*, *UnitVector*, *VandermondeMatrix*, *VectorAdd*, *VectorAngle*, *VectorMatrixMultiply*, *VectorNorm*, *VectorScalarMultiply*, *ZeroMatrix*, *ZeroVector*, *Zip*]

$$v := \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w := \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textbf{(1)}$$

```
> f1:=x1^3/9+x2^3/4-1:
  f2:=x1^3/4+x2^3/9-1:
> F := proc(f,N,v)
  with(LinearAlgebra);
  w:= Vector(N);
  for k from 1 to N do
  w(k):=f||k;
  for j from 1 to N do
  w(k):= subs(x||j=v(j),w(k)):
  od:
  od:
  w;
  end proc;
```

```
Warning, `w` is implicitly declared local to procedure `F`
Warning, `k` is implicitly declared local to procedure `F`
Warning, `j` is implicitly declared local to procedure `F`
```

$$F := \mathbf{proc}(f, N, v) \quad (2)$$

```
    local w, k, j;
    with(LinearAlgebra);
    w := Vector(N);
    for k to N do w(k) := f||k; for j to N do w(k) := subs(x||j = v(j), w(k)) end do end do;
    w
end proc
```

```
> F(f,2,v);
```

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \quad (3)$$

```
> JF := proc(f,N,v)
  with(LinearAlgebra);
  Jac:= Matrix(N);
  for a from 1 to N do
  for b from 1 to N do
  Jac(a,b):= diff(f||a,x||b):
  od:
  od:
  #print(Jac);
  for a from 1 to N do
  for b from 1 to N do
  for j from 1 to N do
  Jac(a,b):= subs(x||j=v(j),Jac(a,b)):
  od:
  od:
  od:
  Jac;
  end proc;
```

```
Warning, `Jac` is implicitly declared local to procedure `JF`
Warning, `a` is implicitly declared local to procedure `JF`
Warning, `b` is implicitly declared local to procedure `JF`
Warning, `j` is implicitly declared local to procedure `JF`
```

$$JF := \mathbf{proc}(f, N, v) \quad (4)$$

```
    local Jac, a, b, j;
    with(LinearAlgebra);
    Jac := Matrix(N);
    for a to N do for b to N do Jac(a, b) := diff(f||a, x||b) end do end do;
    for a to N do
        for b to N do for j to N do Jac(a, b) := subs(x||j = v(j), Jac(a, b)) end do end do
    end do;
    Jac
```

```
end proc
```

```
> v(1):=1;
  v(2):=2;
  JF(f,2,v);
```

$$v := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$v := \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{3} & 3 \\ \frac{3}{4} & \frac{4}{3} \end{bmatrix} \quad \textbf{(5)}$$

```
> Newton2:= proc (v,f,N)
  with(LinearAlgebra):
  w:= Vector(N);
  w:= v-MatrixInverse(JF(f,2,v)) . F(f,2,v);
  w;
  end proc;
```

Warning, `w` is implicitly declared local to procedure `Newton2`

```
Newton2 := proc(v, f, N)
    local w;
    with(LinearAlgebra);
    w := Vector(N);
    w := v - `.`(LinearAlgebra:-MatrixInverse(JF(f, 2, v)), F(f, 2, v));
    w
end proc
```

(6)

```
> for j from 1 to 3 do
  v:= evalf(Newton2(v,f,2));
  od;
```

$$v := \begin{bmatrix} 1.589743590 \\ 1.564102564 \end{bmatrix}$$

$$v := \begin{bmatrix} 1.42507359700000 \\ 1.42005296800000 \end{bmatrix}$$

$$v := \begin{bmatrix} 1.40457985300000 \\ 1.40445245600000 \end{bmatrix} \quad \textbf{(7)}$$

```
> v(1)^3/9+v(2)^3/4;
```

$$1.00045680651692 \quad \textbf{(8)}$$

Griewank's Example--Newton diverges for any point (except the origin) in a ball around the origin.

```
> f1:= 29/16 * x1^3 -2*x1*x2;
  f2:=  x2-x1^2;
```

$$f1 := \frac{29}{16}\, x1^3 - 2\, x1\, x2$$

$$f2 := x2 - x1^2 \tag{9}$$

```
> v:= Vector(2);
  v(1) := 1;
  v(2) := 2;
  v;
```

$$v := \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$v := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$v := \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{10}$$

```
> for j from 1 to 3 do
  v:= evalf(Newton2(v,f,2));
  od;
```

$$v := \begin{bmatrix} 0.9268292683 \\ 0.8536585366 \end{bmatrix}$$

$$v := \begin{bmatrix} 0.610884635700000 \\ 0.273359027100000 \end{bmatrix}$$

$$v := \begin{bmatrix} -3.55046747100000 \\ -4.71103209400000 \end{bmatrix} \tag{11}$$