

Later we will turn this into a proc with n arbitrary.  
 Note # comments out the rest of a line.

```
> restart:
with(LinearAlgebra):
>
> n := 2;
# number of variables and equations
xx:= Vector(2):
                                n := 2
```

You can comment out a system to be solved or put in a different one

```
> #f1 := (x,y) -> x^2+y^2-1;
#f2 := (x,y) -> x^2/16+y^2/9-1;
f1 := (x,y) -> x*y;
f2 := (x,y) -> (x-1)*(y-1);
```

$$f1 := (x, y) \rightarrow xy$$

$$f2 := (x, y) \rightarrow (x-1)(y-1)$$

```
> d1:= degree (f1(x,y));
d2:= degree (f2(x,y));
```

```
RandGamma := stats[random, uniform] () + stats[random, uniform] () * I;
Eq1 := (t,x,y) -> (1-t)*f1(x,y) + RandGamma*t*(x^d1-1);
Eq2 := (t,x,y) -> (1-t)*f2(x,y) + RandGamma*t*(y^d2-1);
```

$$d1 := 2$$

$$d2 := 2$$

$$RandGamma := 0.3957188605 + 0.1931398164 I$$

$$Eq1 := (t, x, y) \rightarrow (1-t) f1(x, y) + RandGamma t (x^{d1} - 1)$$

$$Eq2 := (t, x, y) \rightarrow (1-t) f2(x, y) + RandGamma t (y^{d2} - 1)$$

```
> dxEq1 := unapply(diff(Eq1(t,x,y), x), t, x, y);
dyEq1 := unapply(diff(Eq1(t,x,y), y), t, x, y);
dtEq1 := unapply(diff(Eq1(t,x,y), t), t, x, y);
dxEq2 := unapply(diff(Eq2(t,x,y), x), t, x, y);
dyEq2 := unapply(diff(Eq2(t,x,y), y), t, x, y);
dtEq2 := unapply(diff(Eq2(t,x,y), t), t, x, y);
```

$$dxEq1 := (t, x, y) \rightarrow (1-t)y + (0.7914377210 + 0.3862796328 I)tx$$

$$dyEq1 := (t, x, y) \rightarrow (1-t)x$$

$$dtEq1 := (t, x, y) \rightarrow -xy + (0.3957188605 + 0.1931398164 I)(x^2 - 1)$$

$$dxEq2 := (t, x, y) \rightarrow (1-t)(y-1)$$

$$dyEq2 := (t, x, y) \rightarrow (1-t)(x-1) + (0.7914377210 + 0.3862796328 I) t y$$

$$dtEq2 := (t, x, y) \rightarrow -(x-1)(y-1) + (0.3957188605 + 0.1931398164 I)(y^2 - 1)$$

>

>

```
H := proc(t, xx)
local F := Vector(n);
local i;
  for i from 1 to n do
    F[i] := Eq[i](t, xx[1], xx[2]);
  od;
return(F);
end proc;
```

```
JH := proc(t, xx)
local F := Matrix(n);
F[1,1] := dxEq1(t, xx[1], xx[2]);
F[1,2] := dyEq1(t, xx[1], xx[2]);
F[2,1] := dxEq2(t, xx[1], xx[2]);
F[2,2] := dyEq2(t, xx[1], xx[2]);
return(F);
end proc;
```

```
dtH := proc(t, xx)
local F := Vector(n);
F[1] := dtEq1(t, xx[1], xx[2]);
F[2] := dtEq2(t, xx[1], xx[2]);
return(F);
end proc;
```

>

>

**H := proc(t, xx)**

**local F, i;**

**F := Vector(n); for i to n do F[i] := Eq[i](t, xx[1], xx[2]) end do; return F**

**end proc**

**JH := proc(t, xx)**

**local F;**

**F := Matrix(n);**

**F[1,1] := dxEq1(t, xx[1], xx[2]); end proc**

**F[1,2] := dyEq1(t, xx[1], xx[2]);**

**F[2,1] := dxEq2(t, xx[1], xx[2]);**

**F[2,2] := dyEq2(t, xx[1], xx[2]);**

**return F**

```

dtH := proc(t, xx)
local F;
F := Vector(n);
F[1] := dtEq1(t, xx[1], xx[2]);
F[2] := dtEq2(t, xx[1], xx[2]);
return F;
end proc

```

```

> X := Vector(2);
N := 100;
h := 1.0/N;

```

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

$N := 100$

$h := 0.010000000000$

```

> v := Vector(n);
for i from 1 to d1 do
for j from 1 to d2 do
X[1] := cos(2*Pi*i/d1) + I * sin(2*Pi*i/d1);
X[2] := cos(2*Pi*j/d2) + I * sin(2*Pi*j/d2);
for k from 1 to N do
v := LinearSolve(JH(1.0-(k-1)*h,X), dtH(1.0-(k-1)*h,X));
X := X + h . v;
v := LinearSolve(JH(1.0-k*h,X), H(1.0-k*h,X));
X := X - v;
od;
print(X);
# print(" and the residual is ", H(0,X));
od;
od;

```

$$\begin{bmatrix} -0.967014551180669258 & 10^{-5} - 0.0000124112081287258748 & I \\ 1.00000958510219107 & + 0.0000124163948953537329 & I \end{bmatrix}$$

$$\begin{bmatrix} -0.358953684553853236 & 10^{-7} - 0.390997456634067930 & 10^{-9} I \\ & 1. + 0. I & \end{bmatrix}$$

$$\begin{bmatrix} 1.00000000076474050 & + 0.179329536866394972 & 10^{-8} I \\ -0.364061796951275964 & 10^{-9} - 0.179330075110770798 & 10^{-8} I \end{bmatrix}$$

$$\begin{bmatrix} -0.251184571189652650 & 10^{-14} - 0.482005115074596313 & 10^{-14} I \\ & 1. + 0. I & \end{bmatrix}$$

Notice that we got the root (0,1) three times!  
The trouble is that paths are going to infinity and in this case Newton causes a path jump. Let's run the loop without Newton!

```

> v := Vector(n);

```

```

for i from 1 to d1 do
  for j from 1 to d2 do
    X[1] := cos(2*Pi*i/d1)+ I * sin(2*Pi*i/d1);
    X[2] := cos(2*Pi*j/d2)+ I * sin(2*Pi*j/d2);
    for k from 1 to N do
      v:= LinearSolve(JH(1.0-(k-1)*h,X),dtH(1.0-(k-1)*h,X));
      X := X + h . v ;
      #v:= LinearSolve(JH(1.0-k*h,X),H(1.0-k*h,X));
      #X := X - v ;
      od;
      print(X);
    od;
  od;
od;

```

```

[0.0000244480532394773412 + 0.000101140858141047613 I]
[ 409.153631650140312 - 199.244297172043588 I]

```

```

[-408.218501125724686 + 199.206352219410518 I]
[ 1. + 0. I]

```

```

[ 0.993750524949161829 - 0.00585557620623754564 I]
[ 0.0101807092245746866 - 0.00110637408404875518 I]

```

```

[-0.00313777320501406818 + 0.00231472223037341562 I]
[ 1. + 0. I]

```

>

the roots (1,0) and (0,1) plus two paths are heading to infinity