MSc Course Modelling and Simulation of Mectronic Systems - Francesco Biral - Enrico Bertolazzi

Summary DAE Part

Author: Simone Zamboni - 2019

1: Setup of the problem

1.1 Setup Maple

At the start of every file we need to call restart:

```
> restart:
```

We then need to import the libraries that we need.
Standard libraries that we usually use: plots and Linear Algebra

```
> with(plots):
  with(LinearAlgebra):
```

1.2 Write the equations

We need to write down the equations that we have to solve, for example:

```
> EQ1 := diff(x(t),t)-u(t);
  EQ2 := diff(y(t),t)-v(t);
  EQ3 := diff(u(t),t)*(1+4*x(t)^2)+2*x(t)*diff(v(t),t)+4*x(t)*u(t)^2+x(t)*g+2*x(t)*(1+2*(x(t)^2-y(t)
  ))*lambda(t);
  EQ4 := diff(v(t),t)+2*u(t)^2+2*x(t)*diff(u(t),t)-1/2*g+2*(y(t)-x(t)^2)*lambda(t);
  EQ5 := x(t)^2+(x(t)^2-y(t))^2-1;
```

$$EQ1 := \frac{\mathrm{d}}{\mathrm{d}t}\,x(t) - u(t)$$

$$EQ2 := \frac{\mathrm{d}}{\mathrm{d}t}\,y(t) - v(t)$$

$$EQ3 := \left(\frac{\mathrm{d}}{\mathrm{d}t}\,u(t)\right)(1+4x(t)^2) + 2x(t)\left(\frac{\mathrm{d}}{\mathrm{d}t}\,v(t)\right) + 4x(t)\,u(t)^2 + x(t)\,g + 2x(t)\,(1+2x(t)^2-2y(t))\,\lambda(t)$$

$$EQ4 := \frac{\mathrm{d}}{\mathrm{d}t}\,v(t) + 2u(t)^2 + 2x(t)\left(\frac{\mathrm{d}}{\mathrm{d}t}\,u(t)\right) - \frac{g}{2} + 2\,(y(t)-x(t)^2)\,\lambda(t)$$

$$EQ5 := x(t)^2 + (x(t)^2-y(t))^2 - 1 \qquad \textbf{(1.2.1)}$$

```
> dae := [EQ1,EQ2,EQ3,EQ4,EQ5]: <%>;
```

$$\begin{bmatrix} \frac{d}{dt}x(t) - u(t) \\ \frac{d}{dt}y(t) - v(t) \\ \left(\frac{d}{dt}u(t)\right)(1+4x(t)^2) + 2x(t)\left(\frac{d}{dt}v(t)\right) + 4x(t)\,u(t)^2 + x(t)\,g + 2x(t)\,(1+2x(t)^2 - 2y(t))\,\lambda(t) \\ \frac{d}{dt}v(t) + 2u(t)^2 + 2x(t)\left(\frac{d}{dt}u(t)\right) - \frac{g}{2} + 2\,(y(t) - x(t)^2)\,\lambda(t) \\ x(t)^2 + (x(t)^2 - y(t))^2 - 1 \end{bmatrix} \quad \textbf{(1.2.2)}$$

In this list of equations we have to extract the variables that we will need to compute (the number of this variables is the number of equations that we have):

> vars := [x(t),y(t),u(t),v(t),lambda(t)];

$$vars := [x(t), y(t), u(t), v(t), \lambda(t)] \quad \textbf{(1.2.3)}$$

Not all of these variable will be predent differentiated in the equations, and that is why this is a DAE.

1.3 Explicit form

Now it is time to put the equations in the form **Ey' + Ay + C = 0**

> E,G := GenerateMatrix([EQ||(1..5)],diff(vars,t));

$$E,\ G := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1+4x(t)^2 & 2x(t) & 0 \\ 0 & 0 & 2x(t) & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} u(t) \\ v(t) \\ -4x(t)\,u(t)^2 - x(t)\,g - 2x(t)\,(1+2x(t)^2 - 2y(t))\,\lambda(t) \\ -2u(t)^2 + \frac{g}{2} - 2\,(y(t) - x(t)^2)\,\lambda(t) \\ -x(t)^2 - (x(t)^2 - y(t))^2 + 1 \end{bmatrix} \quad \textbf{(1.3.1)}$$

> A,C := GenerateMatrix(convert(G,list), vars);

$$A,\ C := \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 4\lambda(t)\,x(t)^3 - 4\lambda(t)\,x(t)\,y(t) + 2\lambda(t)\,x(t) + 4x(t)\,u(t)^2 \\ -2\lambda(t)\,x(t)^2 + 2u(t)^2 - \frac{g}{2} + 2y(t)\,\lambda(t) \\ x(t)^4 + x(t)^2 - 2y(t)\,x(t)^2 + y(t)^2 - 1 \end{bmatrix} \quad \textbf{(1.3.2)}$$

> E.<diff(vars,t)> , A.<vars> , C;
E,A,C;

$$\begin{bmatrix} \frac{d}{dt}x(t) \\ \frac{d}{dt}y(t) \\ \left(\frac{d}{dt}u(t)\right)(1+4x(t)^2) + 2x(t)\left(\frac{d}{dt}v(t)\right) \\ \frac{d}{dt}v(t) + 2x(t)\left(\frac{d}{dt}u(t)\right) \\ 0 \end{bmatrix}, \begin{bmatrix} u(t) \\ v(t) \\ -x(t)\,g \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 \\ 0 \\ 4\lambda(t)\,x(t)^3 - 4\lambda(t)\,x(t)\,y(t) + 2\lambda(t)\,x(t) + 4x(t)\,u(t)^2 \\ -2\lambda(t)\,x(t)^2 + 2u(t)^2 - \frac{g}{2} + 2y(t)\,\lambda(t) \\ x(t)^4 + x(t)^2 - 2y(t)\,x(t)^2 + y(t)^2 - 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1+4x(t)^2 & 2x(t) & 0 \\ 0 & 0 & 2x(t) & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 4\lambda(t)\,x(t)^3 - 4\lambda(t)\,x(t)\,y(t) + 2\lambda(t)\,x(t) + 4x(t)\,u(t)^2 \\ -2\lambda(t)\,x(t)^2 + 2u(t)^2 - \frac{g}{2} + 2y(t)\,\lambda(t) \\ x(t)^4 + x(t)^2 - 2y(t)\,x(t)^2 + y(t)^2 - 1 \end{bmatrix} \quad \textbf{(1.3.3)}$$

We usually use the matrix E and G for our computation

2: Index of a DAE

2.1 LU decomposition Method (Library)

We define the function that given the E matrix returns the K matrix (Kernel of the E transpose matrix) and the L matrix (orthonormal matrix to complete the K matrix), both transposed, that will permit us to split between the algebric part and the differential part of the DAE:

```
> KtLtbuild := proc( E )
    local P, L, U, r, LtKt, Lt, Kt;          # definition of the local variables
    P,L,U := LUDecomposition( E );           # LU decomposition by Maple
    r := Rank( E );                          # the first r rows of the resulting matrix are the L
  matrix , the other is the K matrix
    LtKt := LinearSolve(L,Transpose(P));     # compute L^(-1).P
    Lt   := LtKt[1..r,1..-1];
    Kt   := LtKt[r+1..-1,1..-1];
    return Kt, Lt;
  end proc;
```

(2.1.1)

```
KtLtbuild := proc(E)
    local P, L, U, r, LtKt, Lt, Kt;
    P, L, U := LinearAlgebra:-LUDecomposition(E);
    r := LinearAlgebra:-Rank(E);
    LtKt := LinearAlgebra:-LinearSolve(L, LinearAlgebra:-Transpose(P));
    Lt := LtKt[1..r, 1..-1];
    Kt := LtKt[r+1..-1, 1..-1];
    return Kt, Lt
end proc
```

We can use the previous function to create a function that returns the DAE reduced by one index and the algebric part. To do this we call the function to compute Ltraspose and Ktranspose, the differential part will the Ltranspose.(DAE), and the algebric part will be Ktranspose.(DAE) . Then we differentiate the algebric part and return the differential part together with the differentiated algebric part, and also the old algebric part. Remember that this function takes as input the **DAE and the differentiate variables**.

```
> ReduceBy1TheIndex := proc( EQS, Dvars )
```

```
    local E, G, Kt, Lt, DPART, ALG, DALG;        # local variables
    E, G   := GenerateMatrix(EQS,Dvars);         # generate the matrix
    Kt, Lt := KtLtbuild( E );                    # get K and L (both transpose)
    # Separate Algebraic and Differential part
    DPART := simplify(Lt.(E.<Dvars>-G));         # simplify is important
    ALG   := simplify(Kt.(E.<Dvars>-G));
    DALG  := diff(ALG,t);
    # Build the modified DAE, by subtuting the algebraic equation(s) with the derivative of the
  algebraic equation(s)
    return <DPART,DALG>, ALG;
  end proc;
```

$$ReduceBy1TheIndex := \mathbf{proc}(EQS, Dvars) \qquad \textbf{(2.1.2)}$$

local $E, G, Kt, Lt, DPART, ALG, DALG;$

$E, G := LinearAlgebra:\text{-}GenerateMatrix(EQS, Dvars);$

$Kt, Lt := KtLtbuild(E);$

$DPART := simplify(`.`(Lt, `.`(E, <Dvars>) - G));$

$ALG := simplify(`.`(Kt, `.`(E, <Dvars>) - G));$

$DALG := diff(ALG, t);$

return $<DPART, DALG>, ALG$

end proc

2.2 Decomposition with Library

Using the previous functions, that we will have available with a library, we can compute the index of a DAE. We will call the function on the DAE, and we will get the new DAE and the algebric part. Then we will call the function on the new DAE **until the algebric part will be empty**.
When the algebric part will be empty we know that we went a step foreward, so the DAE before the last step is the ODE that we are searching for. The index is how many times we called the library function until we get the algebric part empty, minus 1.
We could also look at the DEA that exits from the library computation each time to know if we have finished. If all the variables are present differentiated we have and ODE, and the index is the number of times we called the library.

```
> dae1, alg1 := ReduceBy1TheIndex( dae, diff(vars,t) );
```

$$dae1, alg1 := \left[\begin{array}{c} \frac{d}{dt}x(t) - u(t) \\ \frac{d}{dt}y(t) - v(t) \\ \left(\frac{d}{dt}u(t)\right)(1+4x(t)^2) + \left(2\frac{d}{dt}v(t) + 4\lambda(t)x(t)^2 + (-4y(t)+2)\lambda(t) + 4u(t)^2 + g\right)x(t) \\ \frac{2\frac{d}{dt}v(t) - 32x(t)^4\lambda(t) + ((32y(t)-12)\lambda(t) - 8g)x(t)^2 + 4u(t)^2 + 4y(t)\lambda(t) - g}{8x(t)^2+2} \\ 2x(t)\left(\frac{d}{dt}x(t)\right) + 2(x(t)^2 - y(t))\left(2x(t)\left(\frac{d}{dt}x(t)\right) - \frac{d}{dt}y(t)\right) \end{array}\right], \qquad \textbf{(2.2.1)}$$

$$\left[\, x(t)^2 + (x(t)^2 - y(t))^2 - 1 \,\right]$$

```
> dae2, alg2 := ReduceBy1TheIndex( convert(dae1,list) ,diff(vars,t) );
  simplify(alg2);
```

$$dae2, alg2 := \left[\left[\, \frac{d}{dt}x(t) - u(t) \,\right],\right.$$

$$\left[\frac{d}{dt}\,y(t)-v(t)\right],$$

$$\left[\left(\frac{d}{dt}\,u(t)\right)(1+4x(t)^2)+\left(2\frac{d}{dt}\,v(t)+4\lambda(t)\,x(t)^2+(-4y(t)+2)\,\lambda(t)+4u(t)^2+g\right)x(t)\right],$$

$$\left[\frac{2\frac{d}{dt}\,v(t)-32x(t)^4\lambda(t)+((32y(t)-12)\,\lambda(t)-8g)\,x(t)^2+4u(t)^2+4y(t)\,\lambda(t)-g}{8x(t)^2+2}\right],$$

$$\left[12x(t)^2u(t)\left(\frac{d}{dt}\,x(t)\right)+4x(t)^3\left(\frac{d}{dt}\,u(t)\right)-2\left(\frac{d}{dt}\,v(t)\right)x(t)^2-4v(t)\,x(t)\left(\frac{d}{dt}\,x(t)\right)-4\left(\frac{d}{dt}\,y(t)\right)u(t)\,x(t)\right.$$
$$\left.+(-4y(t)+2)\left(\frac{d}{dt}\,u(t)\right)x(t)+(-4y(t)+2)\,u(t)\left(\frac{d}{dt}\,x(t)\right)+2\left(\frac{d}{dt}\,v(t)\right)y(t)+2v(t)\left(\frac{d}{dt}\,y(t)\right)\right]\Big],$$

$$\left[\,4x(t)^3u(t)-2v(t)\,x(t)^2+(-4y(t)+2)\,u(t)\,x(t)+2v(t)\,y(t)\,\right]$$
$$\left[\,4x(t)^3u(t)-2v(t)\,x(t)^2+(-4y(t)+2)\,u(t)\,x(t)+2v(t)\,y(t)\,\right] \qquad \textbf{(2.2.2)}$$

> dae3, alg3 := ReduceBy1TheIndex(convert(dae2,list) ,diff(vars,t));

$$dae3,\ alg3 := \left[\left[\frac{d}{dt}\,x(t)-u(t)\right], \right. \qquad \textbf{(2.2.3)}$$

$$\left[\frac{d}{dt}\,y(t)-v(t)\right],$$

$$\left[\left(\frac{d}{dt}\,u(t)\right)(1+4x(t)^2)+\left(2\frac{d}{dt}\,v(t)+4\lambda(t)\,x(t)^2+(-4y(t)+2)\,\lambda(t)+4u(t)^2+g\right)x(t)\right],$$

$$\left[\frac{2\frac{d}{dt}\,v(t)-32x(t)^4\lambda(t)+((32y(t)-12)\,\lambda(t)-8g)\,x(t)^2+4u(t)^2+4y(t)\,\lambda(t)-g}{8x(t)^2+2}\right],$$

$$\left[-64\left(\frac{d}{dt}\,\lambda(t)\right)x(t)^6-384\lambda(t)\,x(t)^5\left(\frac{d}{dt}\,x(t)\right)+\left(128\left(\frac{d}{dt}\,y(t)\right)\lambda(t)+(128y(t)-36)\left(\frac{d}{dt}\,\lambda(t)\right)\right)x(t)^4\right.$$
$$+4((128y(t)-36)\,\lambda(t)-16g)\,x(t)^3\left(\frac{d}{dt}\,x(t)\right)+\left(\left(-128y(t)\left(\frac{d}{dt}\,y(t)\right)+40\frac{d}{dt}\,y(t)\right)\lambda(t)+(-64y(t)^2\right.$$
$$\left.+40y(t)-4)\left(\frac{d}{dt}\,\lambda(t)\right)+16\left(\frac{d}{dt}\,y(t)\right)g+32u(t)\left(\frac{d}{dt}\,u(t)\right)\right)x(t)^2+2((-64y(t)^2+40y(t)-4)\,\lambda(t)$$
$$+16y(t)\,g+16u(t)^2-5g)\,x(t)\left(\frac{d}{dt}\,x(t)\right)-8\left(\frac{d}{dt}\,x(t)\right)u(t)\,v(t)-8x(t)\left(\frac{d}{dt}\,u(t)\right)v(t)-8x(t)\,u(t)\left(\frac{d}{dt}\right.$$
$$\left.v(t)\right)-4\left(\frac{d}{dt}\,\lambda(t)\right)y(t)^2-8\lambda(t)\,y(t)\left(\frac{d}{dt}\,y(t)\right)-16u(t)\left(\frac{d}{dt}\,u(t)\right)y(t)+(-8u(t)^2+g)\left(\frac{d}{dt}\,y(t)\right)$$
$$\left.+4u(t)\left(\frac{d}{dt}\,u(t)\right)+4v(t)\left(\frac{d}{dt}\,v(t)\right)\right]\Big], [-64\lambda(t)\,x(t)^6+((128y(t)-36)\,\lambda(t)-16g)\,x(t)^4+((-64y(t)^2+40y(t)-4)\,\lambda(t)+16y(t)\,g+16u(t)^2-5g)\,x(t)^2-8x(t)\,u(t)\,v(t)-4\lambda(t)\,y(t)^2+(-8u(t)^2+g)\,y(t)+2u(t)^2+2v(t)^2]$$

> dae4, alg4 := ReduceBy1TheIndex(convert(dae3,list) ,diff(vars,t));

$$dae4,\ alg4 := \left[\left[\frac{d}{dt}\,x(t)-u(t)\right], \right. \qquad \textbf{(2.2.4)}$$

$$\left[\frac{d}{dt}\,y(t)-v(t)\right],$$

$$\left[\left(\frac{d}{dt}\,u(t)\right)(1+4x(t)^2)+\left(2\frac{d}{dt}\,v(t)+4\lambda(t)\,x(t)^2+(-4y(t)+2)\,\lambda(t)+4u(t)^2+g\right)x(t)\right],$$

$$\left[\frac{2\frac{d}{dt}v(t) - 32x(t)^4\lambda(t) + ((32y(t) - 12)\lambda(t) - 8g)x(t)^2 + 4u(t)^2 + 4y(t)\lambda(t) - g}{8x(t)^2 + 2}\right],$$

$$\left[(-64x(t)^6 + (128y(t) - 36)x(t)^4 + (-64y(t)^2 + 40y(t) - 4)x(t)^2 - 4y(t)^2)\left(\frac{d}{dt}\lambda(t)\right) - 768\lambda(t)x(t)^5u(t)\right.$$
$$+ 256\lambda(t)x(t)^4v(t) - 160\left(\left(-\frac{32y(t)}{5} + \frac{9}{5}\right)\lambda(t) + g\right)u(t)x(t)^3 + 48v(t)\left(\left(-\frac{16y(t)}{3} + \frac{5}{3}\right)\lambda(t) + g\right)x(t)^2$$
$$+ 64\left(\frac{3u(t)^2}{4} + \left(-4y(t)^2 + \frac{5y(t)}{2} - \frac{1}{4}\right)\lambda(t) + g\left(y(t) - \frac{11}{32}\right)\right)u(t)x(t) + 3v(t)\left(-8u(t)^2 - \frac{16y(t)\lambda(t)}{3}\right.$$
$$\left.\left.+ g\right)\right], [\ ]$$

The resulting algebric part is empty, therefore the ODE is dae3 and this is an index-3 DAE. Because we called four times the reduction function before obtaining the algebric part empty.
We could have also noted at dae3 that it constains all the variable differentiated, and therefore is an ODE.

Now we could also put the ODE in the explicit form:

> ode := dae3:

> explicit_form := solve(convert(ode,list), diff(vars,t))[1]: <%>;

$$\left[\left[\frac{d}{dt}x(t) = u(t)\right], \right. \qquad \textbf{(2.2.5)}$$

$$\left[\frac{d}{dt}y(t) = v(t)\right],$$

$$\left[\frac{d}{dt}u(t) = -2(4\lambda(t)x(t)^2 - 4y(t)\lambda(t) + \lambda(t) + g)x(t)\right],$$

$$\left[\frac{d}{dt}v(t) = 16x(t)^4\lambda(t) - 16x(t)^2y(t)\lambda(t) + 6\lambda(t)x(t)^2 + 4x(t)^2g - 2y(t)\lambda(t) - 2u(t)^2 + \frac{g}{2}\right],$$

$$\left[\frac{d}{dt}\lambda(t) = -(768\lambda(t)x(t)^5u(t) - 256\lambda(t)x(t)^4v(t) - 1024\lambda(t)x(t)^3u(t)y(t) + 288\lambda(t)x(t)^3u(t)\right.$$
$$+ 256\lambda(t)x(t)^2v(t)y(t) + 256\lambda(t)x(t)u(t)y(t)^2 + 160x(t)^3u(t)g - 80\lambda(t)x(t)^2v(t) - 160\lambda(t)x(t)u(t)y(t)$$
$$- 48x(t)^2v(t)g - 48x(t)u(t)^3 - 64x(t)u(t)y(t)g + 16\lambda(t)x(t)u(t) + 16\lambda(t)v(t)y(t) + 22x(t)u(t)g$$
$$\left.+ 24u(t)^2v(t) - 3v(t)g)\right| (4(16x(t)^6 - 32x(t)^4y(t) + 9x(t)^4 + 16x(t)^2y(t)^2 - 10y(t)x(t)^2 + x(t)^2 + y(t)^2))\ ]$$

> E,G := GenerateMatrix(convert(ode,list),diff(vars,t));
E,G := GenerateMatrix(explicit_form,diff(vars,t));

$$E, G := \left[\left[1, 0, 0, 0, 0\right],\right.$$

$$\left[0, 1, 0, 0, 0\right],$$

$$\left[0, 0, 1 + 4x(t)^2, 2x(t), 0\right],$$

$$\left[0, 0, 0, \frac{2}{8x(t)^2 + 2}, 0\right],$$

$$[-384\lambda(t)x(t)^5 + 4((128y(t) - 36)\lambda(t) - 16g)x(t)^3 + 2((-64y(t)^2 + 40y(t) - 4)\lambda(t) + 16y(t)g + 16u(t)^2$$
$$- 5g)x(t) - 8u(t)v(t), 128x(t)^4\lambda(t) + ((-128y(t) + 40)\lambda(t) + 16g)x(t)^2 - 8y(t)\lambda(t) - 8u(t)^2 + g,$$
$$32x(t)^2u(t) - 8v(t)x(t) - 16u(t)y(t) + 4u(t), -8x(t)u(t) + 4v(t), -64x(t)^6 + (128y(t) - 36)x(t)^4 + ($$
$$-64y(t)^2 + 40y(t) - 4)x(t)^2 - 4y(t)^2]],$$

$$\begin{bmatrix} u(t) \\ v(t) \\ -(4\lambda(t)x(t)^2 + (-4y(t) + 2)\lambda(t) + 4u(t)^2 + g)x(t) \\ -\frac{-32x(t)^4\lambda(t) + ((32y(t) - 12)\lambda(t) - 8g)x(t)^2 + 4u(t)^2 + 4y(t)\lambda(t) - g}{8x(t)^2 + 2} \\ 0 \end{bmatrix}$$

$$E,\ G := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \left[\left[u(t) \right], \right. \tag{2.2.6}$$

$$\left[v(t) \right],$$

$$\left[-2\,(4\,\lambda(t)\,x(t)^2 - 4\,y(t)\,\lambda(t) + \lambda(t) + g)\,x(t) \right],$$

$$\left[16\,x(t)^4\,\lambda(t) - 16\,x(t)^2\,y(t)\,\lambda(t) + 6\,\lambda(t)\,x(t)^2 + 4\,x(t)^2\,g - 2\,y(t)\,\lambda(t) - 2\,u(t)^2 + \frac{g}{2} \right],$$

$$\left[-(768\,\lambda(t)\,x(t)^5\,u(t) - 256\,\lambda(t)\,x(t)^4\,v(t) - 1024\,\lambda(t)\,x(t)^3\,u(t)\,y(t) + 288\,\lambda(t)\,x(t)^3\,u(t) \right.$$
$$+ 256\,\lambda(t)\,x(t)^2\,v(t)\,y(t) + 256\,\lambda(t)\,x(t)\,u(t)\,y(t)^2 + 160\,x(t)^3\,u(t)\,g - 80\,\lambda(t)\,x(t)^2\,v(t) - 160\,\lambda(t)\,x(t)\,u(t)\,y(t)$$
$$- 48\,x(t)^2\,v(t)\,g - 48\,x(t)\,u(t)^3 - 64\,x(t)\,u(t)\,y(t)\,g + 16\,\lambda(t)\,x(t)\,u(t) + 16\,\lambda(t)\,v(t)\,y(t) + 22\,x(t)\,u(t)\,g$$
$$\left. + 24\,u(t)^2\,v(t) - 3\,v(t)\,g) \middle| (4\,(16\,x(t)^6 - 32\,x(t)^4\,y(t) + 9\,x(t)^4 + 16\,x(t)^2\,y(t)^2 - 10\,y(t)\,x(t)^2 + x(t)^2 + y(t)^2)) \right]\Big]$$

2.3 Manual decomposition

If we know exactly what is the algebric part in the starting DAE, we can manually compute the index reduction, taking the algebric part, differentiate it and substituting the expression of the already know differentiated variables in it, and continue to do that until we have the missing variable differentiated.

Let's start by explicitally solve the differential part of the DAE for the variables that we have differentiated

```
> solve(dae[1..4],[diff(x(t),t),diff(y(t),t),diff(u(t),t),diff(v(t),t)]): diff_part :=%[1]: <%>;
```

$$\begin{bmatrix} \frac{d}{dt}\,x(t) = u(t) \\ \frac{d}{dt}\,y(t) = v(t) \\ \frac{d}{dt}\,u(t) = -8\,\lambda(t)\,x(t)^3 + 8\,\lambda(t)\,x(t)\,y(t) - 2\,\lambda(t)\,x(t) - 2\,x(t)\,g \\ \frac{d}{dt}\,v(t) = 16\,x(t)^4\,\lambda(t) - 16\,x(t)^2\,y(t)\,\lambda(t) + 6\,\lambda(t)\,x(t)^2 + 4\,x(t)^2\,g - 2\,y(t)\,\lambda(t) - 2\,u(t)^2 + \frac{g}{2} \end{bmatrix} \tag{2.3.1}$$

Let's take the first algebric part from the DAE:

```
> alg1 := dae[5];
```

$$alg1 := x(t)^2 + (x(t)^2 - y(t))^2 - 1 \tag{2.3.2}$$

Now we will differentiate that part and substituting the solution for the other differentiated variables in it:

```
> alg2 := diff(alg1,t);
  alg2 := simplify(subs(diff_part,alg2));
```

$$alg2 := 2\,x(t)\left(\frac{d}{dt}\,x(t)\right) + 2\,(x(t)^2 - y(t))\left(2\,x(t)\left(\frac{d}{dt}\,x(t)\right) - \frac{d}{dt}\,y(t)\right)$$

$$alg2 := 2\,x(t)\,u(t) - 4\left(x(t)\,u(t) - \frac{v(t)}{2}\right)(y(t) - x(t)^2) \tag{2.3.3}$$

```
> alg3 := diff(alg2,t);
  alg3 := simplify(subs(diff_part,alg3));
```

$$alg3 := 2\left(\frac{d}{dt}\,x(t)\right)u(t) + 2\,x(t)\left(\frac{d}{dt}\,u(t)\right) - 4\left(\left(\frac{d}{dt}\,x(t)\right)u(t) + x(t)\left(\frac{d}{dt}\,u(t)\right) - \frac{\frac{d}{dt}\,v(t)}{2}\right)(y(t) - x(t)^2)$$
$$- 4\left(x(t)\,u(t) - \frac{v(t)}{2}\right)\left(\frac{d}{dt}\,y(t) - 2\,x(t)\left(\frac{d}{dt}\,x(t)\right)\right)$$

(2.3.4)

$$alg3 := -64\,\lambda(t)\,x(t)^6 + ((128\,y(t) - 36)\,\lambda(t) - 16\,g)\,x(t)^4 + ((-64\,y(t)^2 + 40\,y(t) - 4)\,\lambda(t) + 16\,y(t)\,g + 16\,u(t)^2 - 5\,g)\,x(t)^2 - 8\,x(t)\,u(t)\,v(t) - 4\,\lambda(t)\,y(t)^2 + (-8\,u(t)^2 + g)\,y(t) + 2\,u(t)^2 + 2\,v(t)^2 \quad \textbf{(2.3.4)}$$

> alg4 := diff(alg3,t):
alg4 := simplify(subs(diff_part,alg4));

$$\begin{aligned} alg4 := {} & (-64\,x(t)^6 + (128\,y(t) - 36)\,x(t)^4 + (-64\,y(t)^2 + 40\,y(t) - 4)\,x(t)^2 - 4\,y(t)^2)\left(\frac{d}{dt}\,\lambda(t)\right) - 768\,\lambda(t)\,x(t)^5\,u(t) \\ & + 256\,\lambda(t)\,x(t)^4\,v(t) - 160\left(\left(-\frac{32\,y(t)}{5} + \frac{9}{5}\right)\lambda(t) + g\right)u(t)\,x(t)^3 + 48\,v(t)\left(\left(-\frac{16\,y(t)}{3} + \frac{5}{3}\right)\lambda(t) + g\right)x(t)^2 \\ & + 64\left(\frac{3\,u(t)^2}{4} + \left(-4\,y(t)^2 + \frac{5\,y(t)}{2} - \frac{1}{4}\right)\lambda(t) + g\left(y(t) - \frac{11}{32}\right)\right)u(t)\,x(t) + 3\,v(t)\left(-8\,u(t)^2 - \frac{16\,y(t)\,\lambda(t)}{3} \right. \\ & \left. + g\right) \end{aligned} \quad \textbf{(2.3.5)}$$

As we can see in alg4 is present the derivative of the variable of wich the derivative was not present in the starting DAE. That is when you know that you reach the ODE stage. In this case, like with the library, we did this step four times and therefore the DAE is an index-3.

Now we can take our starting DAE, remove the algebric part and put the new differentiated part (alg4), and maybe also put the equation in the explicit form.

> ode := [dae[1], dae[2], dae[3], dae[4], alg4]; <%>;

$$\begin{aligned} ode := {} & \left[\frac{d}{dt}\,x(t) - u(t),\ \frac{d}{dt}\,y(t) - v(t),\ \left(\frac{d}{dt}\,u(t)\right)(1 + 4\,x(t)^2) + 2\,x(t)\left(\frac{d}{dt}\,v(t)\right) + 4\,x(t)\,u(t)^2 + x(t)\,g + 2\,x(t)\,(1 \right. \\ & + 2\,x(t)^2 - 2\,y(t))\,\lambda(t),\ \frac{d}{dt}\,v(t) + 2\,u(t)^2 + 2\,x(t)\left(\frac{d}{dt}\,u(t)\right) - \frac{g}{2} + 2\,(y(t) - x(t)^2)\,\lambda(t),\ (-64\,x(t)^6 + (128\,y(t) \\ & - 36)\,x(t)^4 + (-64\,y(t)^2 + 40\,y(t) - 4)\,x(t)^2 - 4\,y(t)^2)\left(\frac{d}{dt}\,\lambda(t)\right) - 768\,\lambda(t)\,x(t)^5\,u(t) + 256\,\lambda(t)\,x(t)^4\,v(t) \\ & - 160\left(\left(-\frac{32\,y(t)}{5} + \frac{9}{5}\right)\lambda(t) + g\right)u(t)\,x(t)^3 + 48\,v(t)\left(\left(-\frac{16\,y(t)}{3} + \frac{5}{3}\right)\lambda(t) + g\right)x(t)^2 + 64\left(\frac{3\,u(t)^2}{4} + \right. \\ & \left.\left. \left(-4\,y(t)^2 + \frac{5\,y(t)}{2} - \frac{1}{4}\right)\lambda(t) + g\left(y(t) - \frac{11}{32}\right)\right)u(t)\,x(t) + 3\,v(t)\left(-8\,u(t)^2 - \frac{16\,y(t)\,\lambda(t)}{3} + g\right)\right] \end{aligned}$$

$$\left[\left[\frac{d}{dt}\,x(t) - u(t)\right], \left[\frac{d}{dt}\,y(t) - v(t)\right], \left[\left(\frac{d}{dt}\,u(t)\right)(1 + 4\,x(t)^2) + 2\,x(t)\left(\frac{d}{dt}\,v(t)\right) + 4\,x(t)\,u(t)^2 + x(t)\,g + 2\,x(t)\,(1 + 2\,x(t)^2 - 2\,y(t))\,\lambda(t)\right], \left[\frac{d}{dt}\,v(t) + 2\,u(t)^2 + 2\,x(t)\left(\frac{d}{dt}\,u(t)\right) - \frac{g}{2} + 2\,(y(t) - x(t)^2)\,\lambda(t)\right], \left[(-64\,x(t)^6 + (128\,y(t) - 36)\,x(t)^4 + (-64\,y(t)^2 + 40\,y(t) - 4)\,x(t)^2 - 4\,y(t)^2)\left(\frac{d}{dt}\,\lambda(t)\right) - 768\,\lambda(t)\,x(t)^5\,u(t) + 256\,\lambda(t)\,x(t)^4\,v(t) - 160\left(\left(-\frac{32\,y(t)}{5} + \frac{9}{5}\right)\lambda(t) + g\right)u(t)\,x(t)^3 + 48\,v(t)\left(\left(-\frac{16\,y(t)}{3} + \frac{5}{3}\right)\lambda(t) + g\right)x(t)^2 + 64\left(\frac{3\,u(t)^2}{4} + \left(-4\,y(t)^2 + \frac{5\,y(t)}{2} - \frac{1}{4}\right)\lambda(t) + g\left(y(t) - \frac{11}{32}\right)\right)u(t)\,x(t) + 3\,v(t)\left(-8\,u(t)^2 - \frac{16\,y(t)\,\lambda(t)}{3} + g\right)\right], \right] \quad \textbf{(2.3.6)}$$

> explicit_form := solve(ode, diff(vars,t))[1]: <%>;

$$\left[\left[\frac{d}{dt}\,x(t) = u(t)\right], \left[\frac{d}{dt}\,y(t) = v(t)\right], \right. \quad \textbf{(2.3.7)}$$

$$\left[\frac{d}{dt}\,u(t) = -8\,\lambda(t)\,x(t)^3 + 8\,\lambda(t)\,x(t)\,y(t) - 2\,\lambda(t)\,x(t) - 2\,x(t)\,g\right],$$

$$\left[\frac{d}{dt}\,v(t) = 16\,x(t)^4\,\lambda(t) - 16\,x(t)^2\,y(t)\,\lambda(t) + 6\,\lambda(t)\,x(t)^2 + 4\,x(t)^2\,g - 2\,y(t)\,\lambda(t) - 2\,u(t)^2 + \frac{g}{2}\right],$$

$$\left[\frac{d}{dt}\,\lambda(t) = (-768\,\lambda(t)\,x(t)^5\,u(t) + 1024\,\lambda(t)\,x(t)^3\,u(t)\,y(t) + 256\,\lambda(t)\,x(t)^4\,v(t) - 256\,\lambda(t)\,x(t)\,u(t)\,y(t)^2 - 288\,\lambda(t)\,x(t)^3\,u(t) - 160\,x(t)^3\,u(t)\,g - 256\,\lambda(t)\,x(t)^2\,v(t)\,y(t) + 48\,x(t)\,u(t)^3 + 160\,\lambda(t)\,x(t)\,u(t)\,y(t) + 64\,x(t)\,u(t)\,y(t)\,g + 80\,\lambda(t)\,x(t)^2\,v(t) + 48\,x(t)^2\,v(t)\,g - 24\,u(t)^2\,v(t) - 16\,\lambda(t)\,x(t)\,u(t) - 22\,x(t)\,u(t)\,g - 16\,\lambda(t)\,v(t)\,y(t) + 3\,v(t)\,g)\Big/(4\,(16\,x(t)^6 - 32\,x(t)^4\,y(t) + 9\,x(t)^4 + 16\,x(t)^2\,y(t)^2 - 10\,y(t)\,x(t)^2 + x(t)^2 + y(t)^2))\right],$$

> E,G := GenerateMatrix(ode,diff(vars,t));
E,G := GenerateMatrix(explicit_form,diff(vars,t));

$$E,\,G := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1+4\,x(t)^2 & 2\,x(t) & 0 \\ 0 & 0 & 2\,x(t) & 1 & 0 \\ 0 & 0 & 0 & 0 & -64\,x(t)^6 + (128\,y(t) - 36)\,x(t)^4 + (-64\,y(t)^2 + 40\,y(t) - 4)\,x(t)^2 - 4\,y(t)^2 \end{bmatrix}, \left[\right.$$

$$\left[u(t)\right],$$

$$\left[v(t)\right],$$

$$\left[-4\,x(t)\,u(t)^2 - x(t)\,g - 2\,x(t)\,(1 + 2\,x(t)^2 - 2\,y(t))\,\lambda(t)\right],$$

$$\left[-2\,u(t)^2 + \frac{g}{2} - 2\,(y(t) - x(t)^2)\,\lambda(t)\right],$$

$$\left[768\,\lambda(t)\,x(t)^5\,u(t) - 256\,\lambda(t)\,x(t)^4\,v(t) + 160\left(\left(-\frac{32\,y(t)}{5} + \frac{9}{5}\right)\lambda(t) + g\right)u(t)\,x(t)^3 - 48\,v(t)\left(\left(-\frac{16\,y(t)}{3} + \frac{5}{3}\right)\lambda(t) + g\right)x(t)^2 - 64\left(\frac{3\,u(t)^2}{4} + \left(-4\,y(t)^2 + \frac{5\,y(t)}{2} - \frac{1}{4}\right)\lambda(t) + g\left(y(t) - \frac{11}{32}\right)\right)u(t)\,x(t) - 3\,v(t)\left(-8\,u(t)^2 - \frac{16\,y(t)\,\lambda(t)}{3} + g\right)\right]\right]$$

$$E,\,G := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \left[\left[u(t)\right], \right. \tag{2.3.8}$$

$$\left[v(t)\right],$$

$$\left[-8\,\lambda(t)\,x(t)^3 + 8\,\lambda(t)\,x(t)\,y(t) - 2\,\lambda(t)\,x(t) - 2\,x(t)\,g\right],$$

$$\left[16\,x(t)^4\,\lambda(t) - 16\,x(t)^2\,y(t)\,\lambda(t) + 6\,\lambda(t)\,x(t)^2 + 4\,x(t)^2\,g - 2\,y(t)\,\lambda(t) - 2\,u(t)^2 + \frac{g}{2}\right],$$

$$[(-768\,\lambda(t)\,x(t)^5\,u(t) + 1024\,\lambda(t)\,x(t)^3\,u(t)\,y(t) + 256\,\lambda(t)\,x(t)^4\,v(t) - 256\,\lambda(t)\,x(t)\,u(t)\,y(t)^2 - 288\,\lambda(t)\,x(t)^3\,u(t) - 160\,x(t)^3\,u(t)\,g - 256\,\lambda(t)\,x(t)^2\,v(t)\,y(t) + 48\,x(t)\,u(t)^3 + 160\,\lambda(t)\,x(t)\,u(t)\,y(t) + 64\,x(t)\,u(t)\,y(t)\,g + 80\,\lambda(t)\,x(t)^2\,v(t) + 48\,x(t)^2\,v(t)\,g - 24\,u(t)^2\,v(t) - 16\,\lambda(t)\,x(t)\,u(t) - 22\,x(t)\,u(t)\,g - 16\,\lambda(t)\,v(t)\,y(t) + 3\,v(t)\,g)\Big/(4\,(16\,x(t)^6 - 32\,x(t)^4\,y(t) + 9\,x(t)^4 + 16\,x(t)^2\,y(t)^2 - 10\,y(t)\,x(t)^2 + x(t)^2 + y(t)^2))]]$$

3: Initial conditions

Now it's time to calculate the initial conditions of the problem.
In order to do that we have to solve all the algebric parts together and put some initial conditions, maybe it's time=0 or maybe time=0 plus some variables.

Let's start by grouping all the algebric parts togheter:

```
> algs := [alg1, alg2, alg3]: <%>; # case manual reduction
  # algs := [ alg1[1], alg2[1], alg3[1] ]:<%>;: <%> # case library reduction
```

$$\left[\left[x(t)^2 + (x(t)^2 - y(t))^2 - 1\right],\right. \tag{3.1}$$

$$\left[2\,x(t)\,u(t) - 4\left(x(t)\,u(t) - \frac{v(t)}{2}\right)(y(t) - x(t)^2)\right],$$

$$\left.[-64\,\lambda(t)\,x(t)^6 + ((128\,y(t) - 36)\,\lambda(t) - 16\,g)\,x(t)^4 + ((-64\,y(t)^2 + 40\,y(t) - 4)\,\lambda(t) + 16\,y(t)\,g + 16\,u(t)^2 - 5\,g)\,x(t)^2 - 8\,x(t)\,u(t)\,v(t) - 4\,\lambda(t)\,y(t)^2 + (-8\,u(t)^2 + g)\,y(t) + 2\,u(t)^2 + 2\,v(t)^2]\right] ,$$

Surely at the start t=0 is always true. But it can be more complex than that. For example in this case we have in total three hidden constraints, but we have five variables. This means that on top of the t=0 at the start we will have to also put the initial conditions of the two indipendent variables, solve the hidden constraints for the dependent variables and then put the initial conditions of the inidpendent variable and find the initial condition of the dependent variables.

```
> # sol_initial_qD := solve (algs,qD);
> # initial_qD := subs(data, initial_qI, sol_initial_qD);
```

Another example could be:

```
> dae := [diff(x2(t),t) = q1(t) - x1(t),
  diff(x3(t),t) = q2(t) -(1+eta)*x2(t) -eta*t*(q1(t) -x1(t)),
  0=q3(t)-eta*t*x2(t)-x3(t)]: <%>;
  vars := [x1(t),x2(t),x3(t)];
```

$$\begin{bmatrix} \frac{d}{dt}\,x2(t) = q1(t) - x1(t) \\ \frac{d}{dt}\,x3(t) = q2(t) - (1+\eta)\,x2(t) - \eta\,t\,(q1(t) - x1(t)) \\ 0 = q3(t) - \eta\,t\,x2(t) - x3(t) \end{bmatrix}$$

$$vars := [x1(t),\, x2(t),\, x3(t)] \tag{3.2}$$

This is an index-3 DAE, let's see:

```
> dae1, alg1 := ReduceBy1TheIndex(dae, diff(vars,t));
```

$$dae1,\, alg1 := \begin{bmatrix} \frac{d}{dt}\,x2(t) - q1(t) + x1(t) \\ \frac{d}{dt}\,x3(t) - q2(t) + (1+\eta)\,x2(t) + \eta\,t\,(q1(t) - x1(t)) \\ -\frac{d}{dt}\,q3(t) + x2(t)\,\eta + \eta\,t\left(\frac{d}{dt}\,x2(t)\right) + \frac{d}{dt}\,x3(t) \end{bmatrix}, \begin{bmatrix} -q3(t) + \eta\,t\,x2(t) + x3(t) \end{bmatrix} \tag{3.3}$$

```
> dae2, alg2 := ReduceBy1TheIndex( convert(dae1,list), diff(vars,t));
```

$$dae2,\, alg2 := \begin{bmatrix} \frac{d}{dt}\,x2(t) - q1(t) + x1(t) \\ \frac{d}{dt}\,x3(t) - q2(t) + (1+\eta)\,x2(t) + \eta\,t\,(q1(t) - x1(t)) \\ \frac{d}{dt}\,q2(t) - \frac{d}{dt}\,x2(t) - \frac{d^2}{dt^2}\,q3(t) \end{bmatrix}, \begin{bmatrix} q2(t) - x2(t) - \frac{d}{dt}\,q3(t) \end{bmatrix} \tag{3.4}$$

```
> dae3, alg3 := ReduceBy1TheIndex( convert(dae2,list), diff(vars,t));
```

$$dae3,\ alg3 := \left[\begin{array}{c} \frac{d}{dt}\ x2(t) - q1(t) + x1(t) \\ \frac{d}{dt}\ x3(t) - q2(t) + (1+\eta)\ x2(t) + \eta\, t\,(q1(t) - x1(t)) \\ -\frac{d}{dt}\ q1(t) + \frac{d}{dt}\ x1(t) + \frac{d^2}{dt^2}\ q2(t) - \frac{d^3}{dt^3}\ q3(t) \end{array}\right], \left[-q1(t) + x1(t) + \frac{d}{dt}\ q2(t) - \frac{d^2}{dt^2}\ q3(t) \right] \quad (3.5)$$

> dae4, alg4 := ReduceBy1TheIndex(convert(dae3,list), diff(vars,t)); # empty alg4, is an index3 DAE and dae3 is the ODE

$$dae4,\ alg4 := \left[\begin{array}{c} -\frac{d}{dt}\ q1(t) + \frac{d}{dt}\ x1(t) + \frac{d^2}{dt^2}\ q2(t) - \frac{d^3}{dt^3}\ q3(t) \\ \frac{d}{dt}\ x2(t) - q1(t) + x1(t) \\ \frac{d}{dt}\ x3(t) - q2(t) + (1+\eta)\ x2(t) + \eta\, t\,(q1(t) - x1(t)) \end{array}\right], [\ ] \quad (3.6)$$

> ode := dae3;

$$ode := \left[\begin{array}{c} \frac{d}{dt}\ x2(t) - q1(t) + x1(t) \\ \frac{d}{dt}\ x3(t) - q2(t) + (1+\eta)\ x2(t) + \eta\, t\,(q1(t) - x1(t)) \\ -\frac{d}{dt}\ q1(t) + \frac{d}{dt}\ x1(t) + \frac{d^2}{dt^2}\ q2(t) - \frac{d^3}{dt^3}\ q3(t) \end{array}\right] \quad (3.7)$$

> algs := [alg1[1], alg2[1], alg3[1]]:<%>;

$$\left[\begin{array}{c} -q3(t) + \eta\, t\, x2(t) + x3(t) \\ q2(t) - x2(t) - \frac{d}{dt}\ q3(t) \\ -q1(t) + x1(t) + \frac{d}{dt}\ q2(t) - \frac{d^2}{dt^2}\ q3(t) \end{array}\right] \quad (3.8)$$

In this case we have three variables and three hidden constraints, which menas that we can solve analitically the hidden constraints for the variables and have the intial consitions depending only on time:

> sol_hidden := solve(algs , vars) [1] : <%>;

$$\left[\begin{array}{c} x1(t) = q1(t) - \frac{d}{dt}\ q2(t) + \frac{d^2}{dt^2}\ q3(t) \\ x2(t) = q2(t) - \frac{d}{dt}\ q3(t) \\ x3(t) = -\eta\, t\, q2(t) + \eta\, t\left(\frac{d}{dt}\ q3(t)\right) + q3(t) \end{array}\right] \quad (3.9)$$

> data := [q1(t) = 1, q2(t) = 1, q3(t) = 0,eta=1]; # compute the initial conditions substituting the data and t=0
eval(subs(data, sol_hidden)) ;
ics := evalf(subs(t=0,%));

$$data := [q1(t) = 1, q2(t) = 1, q3(t) = 0, \eta = 1]$$
$$[x1(t) = 1, x2(t) = 1, x3(t) = -t]$$
$$ics := [x1(0) = 1., x2(0) = 1., x3(0) = 0.] \quad (3.10)$$

4: Solve the DAE

To solve the DAE we need to have some things:
• The ODE Explicit form (for the numerical method, otherwise is ok the normal ODE)
• the list of the variables with initial conditions

We can either do it with the Maple command dsolve() or with our custom numerical method.

4.1 Maple dsolve()

With the dsolve() command Maple can solve the differential equations (and also the DAE if we want).
The command needs to be called with the ODE to solve united with the initial conditions in this for: x1(0) = 0, and if we need also the initial condition od the derivative of a function we have to write: D(x1)(0) = 0.
The command accepts some parameters, the most common that we use are two: numeric and implicit=true. With numeric Maple will try to solve the ODE in a numerical way, and we will have to use odeplot() to display the results, without numeric Maple will try to solve the ODE analitically and will return the analitical expression of the variables (if found). When we use a numerical method we can put implicit=true to say to Maple to solve the ODE with an implicit numerical method, we usually do this when the normal command (that uses the explicit methods) doesn't work.

```
> ode := [ diff(x2(t),t) - 1 + x1(t)=0, diff(x3(t),t) - 1 + 2*x2(t)+t*(1-x1(t))=0, diff(x1(t),t) =1]:<%>;
```

$$\left[\frac{d}{dt}x2(t) - 1 + x1(t) = 0,\ \frac{d}{dt}x3(t) - 1 + 2\,x2(t) + t\,(1 - x1(t)) = 0,\ \frac{d}{dt}x1(t) = 1\right] \quad \textbf{(4.1.1)}$$

```
> ics := {x1(0) = 1, x2(0) = 2, x3(0) = 3};
```

$$ics := \{x1(0) = 1,\ x2(0) = 2,\ x3(0) = 3\} \quad \textbf{(4.1.2)}$$

```
> sol_numerical := dsolve(convert(ode,set) union ics, numeric);
```

$$sol\_numerical := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc} \quad \textbf{(4.1.3)}$$

```
> sol_analitical := dsolve(convert(ode,set) union ics);
```

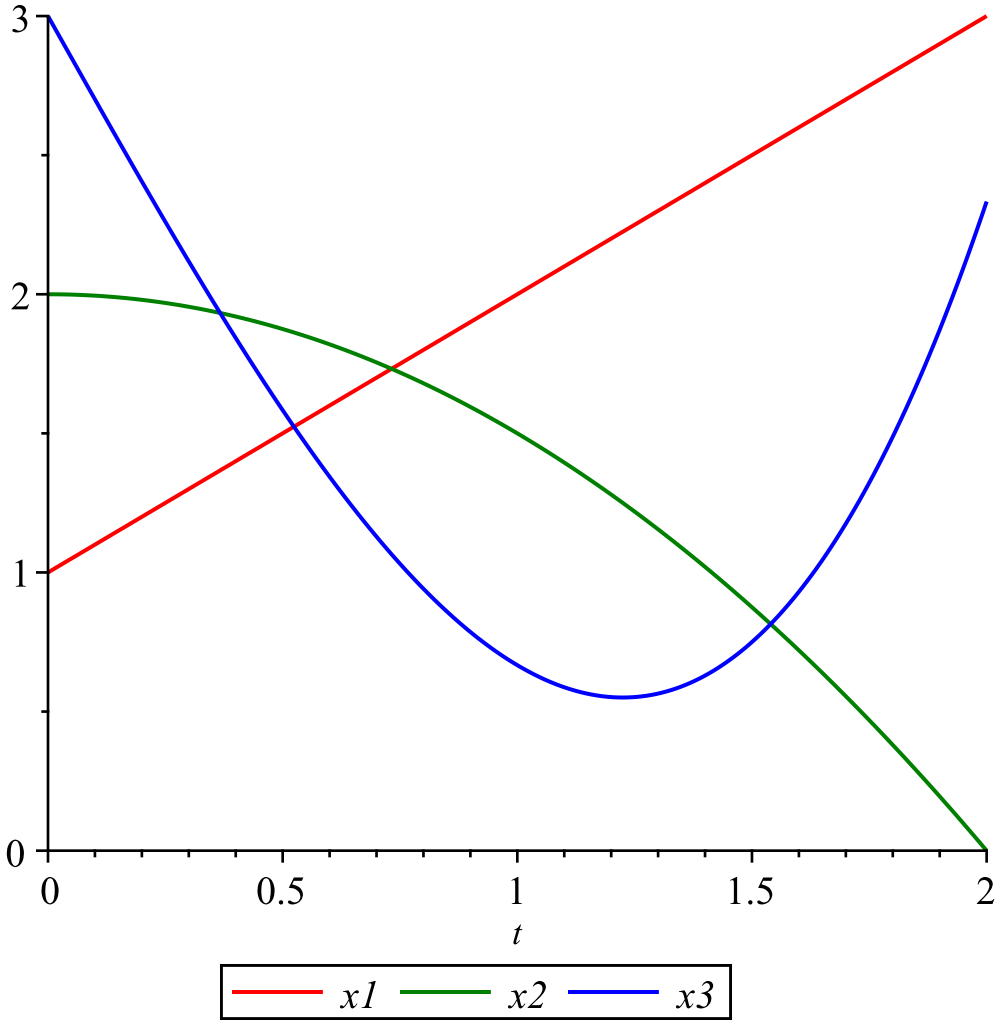
$$sol\_analitical := \left\{x1(t) = t + 1,\ x2(t) = -\frac{t^2}{2} + 2,\ x3(t) = \frac{2}{3}t^3 - 3\,t + 3\right\} \quad \textbf{(4.1.4)}$$

Now that we have the solution let's see how we can plot them.

4.1.1 Maple plot()

For tha analitical solution we will use the plot() command.
This comman is structured like this:
1. A function or a list of function to plot (mandatory) - only the function, not x1(t) = f(t) but only f(t), in case of x1(t)=f(t) use the command rhs()
2. The variable on the x-axis and its interval, in a form like t=a..b (mandatory)
3. color = "Red" or a list, like color = ["Red","Green"] (optional)
4. legend = x1 or a list, like legend = [x1,x2,x3] (optional)

```
> plot( [rhs(sol_analitical[1]), rhs(sol_analitical[2]), rhs(sol_analitical[3]) ] ,t=0..2 , color=
  ["Red","Green","Blue"], legend=[x1,x2,x3] );
```

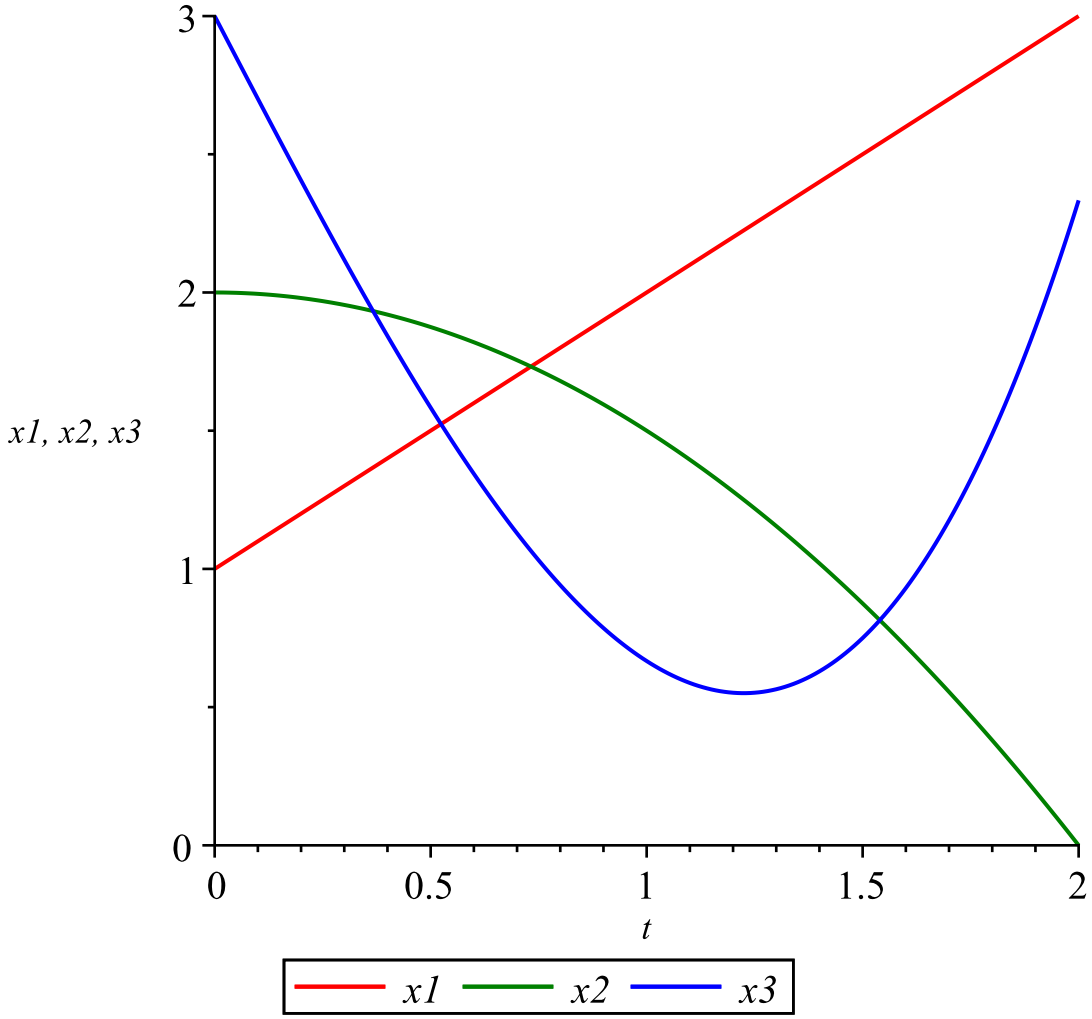


4.1.2 Maple odeplot()

For the numeric solution we have to use the command odeplot() to plot the solution.
This command is structured like this:

1. The variable containing the solution (mandatory)
2. A list or a list of list, with this structure: [variable on x-axis, function to plot, color="Red" (optional)
3. The variable on the x-axis and its interval, in a form like t=a..b
4. legend = x1 or a list, like legend = [x1,x2,x3] (optional)

```
> odeplot(sol_numerical, [ [t,x1(t),color="Red" ] , [t,x2(t),color="Green" ] , [t,x3(t),color=
  "Blue" ] ], t=0..2, legend = [x1,x2,x3]);
```



4.2 Numerical methods

We can also use our custom numerical method to solve the ODE, but it will require some preparation.

First we have to make the ode in the **explicit form** (we also did this in chapter 2), and then take only the right end side (so the function) of the ODE:

```
> ode: <%>;
  vars := [x1(t),x2(t),x3(t)];
  solve(ode,diff(vars,t)): ex_ode := %[1];
```

$$\begin{bmatrix} \frac{d}{dt}\,x2(t) - 1 + x1(t) = 0 \\ \frac{d}{dt}\,x3(t) - 1 + 2\,x2(t) + t\,(1 - x1(t)) = 0 \\ \frac{d}{dt}\,x1(t) = 1 \end{bmatrix}$$

$$vars := [x1(t), x2(t), x3(t)]$$

$$ex\_ode := \left[\frac{d}{dt}\,x1(t) = 1,\ \frac{d}{dt}\,x2(t) = 1 - x1(t),\ \frac{d}{dt}\,x3(t) = t\,x1(t) - 2\,x2(t) - t + 1\right] \qquad \textbf{(4.2.1)}$$

```
> rhs_ode := [ rhs(ex_ode[1]),rhs(ex_ode[2]),rhs(ex_ode[3])]:<%>;
```

$$\begin{bmatrix} 1 \\ 1 - x1(t) \\ t\,x1(t) - 2\,x2(t) - t + 1 \end{bmatrix} \tag{4.2.2}$$

Then we have to decide the **step size h** and the **number of steps** we will take. We could) also already compute the value of the time at each timestep

```
> h := 0.1:
  nsteps := 10:
> t_step := [ seq(h*i, i=0..nsteps)];
```

$$t\_step := [0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] \tag{4.2.3}$$

Then we create a matrix big enought to have all our variables for each step, in order to contain the results. Then we put on the first row the initial conditions of our system

```
> results := Matrix(nsteps , nops(vars));
```

$$results := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.2.4}$$

```
> results[1,1] := 1:
  results[1,2] := 2:
  results[1,3] := 3:
  results;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.2.5}$$

Now we have to create our own numerical method.

4.2.1 Numerical method procedure

A numerical method in Maple is simply a procedure that takes as input the time, step size, all the variables at the current iteration and the ode in the right end side form.
It then returns the values of all the variable at the next iteration.
How it is implemented inside dependes by the numerical method chosen. We will see some examples:

4.2.1.1 Explicit Euler method

The explicit Euler method follow a very simple structure:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

In our case we have three variables, so y is a vector of dimension three, and f is the right hand side of the ODE.

```
> EulerStep := proc(tk, h, x1k, x2k, x3k, ode)
    local x1k1, x2k1, x3k1 :
    x1k1 := eval(x1k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, x3(tk)=x3k, ode[1]) ) :
    x2k1 := eval(x2k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, x3(tk)=x3k, ode[2]) ) :
    x3k1 := eval(x3k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, x3(tk)=x3k, ode[3]) ) :
    return < x1k1, x2k1, x3k1> :
  end proc;
```

EulerStep := **proc**(*tk*, *h*, *x1k*, *x2k*, *x3k*, *ode*) **(4.2.1.1.1)**

local *x1k1*, *x2k1*, *x3k1*;

x1k1 := *eval*(*x1k* + *h***subs*(*t* = *tk*, *x1*(*tk*) = *x1k*, *x2*(*tk*) = *x2k*, *x3*(*tk*) = *x3k*, *ode*[1]));

x2k1 := *eval*(*x2k* + *h***subs*(*t* = *tk*, *x1*(*tk*) = *x1k*, *x2*(*tk*) = *x2k*, *x3*(*tk*) = *x3k*, *ode*[2]));

x3k1 := *eval*(*x3k* + *h***subs*(*t* = *tk*, *x1*(*tk*) = *x1k*, *x2*(*tk*) = *x2k*, *x3*(*tk*) = *x3k*, *ode*[3]));

return < *x1k1*, *x2k1*, *x3k1* >

end proc

4.2.1.2 Collaz scheme

The Collaz scheme follow this structure:

$$x_{k+\frac{1}{2}} = x_k + \frac{h}{2} f\left(t_{k'}, x_k\right)$$

$$x_{k+1} = x_k + h f\left(t_k + \frac{h}{2}, x_{k+\frac{1}{2}}\right)$$

In this case the implementation is for a function that accepts two variables.

```
> collatzStep := proc(tk,h,yk,zk,ode)
    local yk12,zk12,yk1,zk1;
    yk12 := yk + h/2 * subs( t=tk, y(tk)=yk, z(tk)= zk, ode[1]);
```

```
    zk12 := zk + h/2 * subs( t=tk, y(tk)=yk, z(tk)= zk, ode[2]);

    yk1 := evalf(yk + h * subs(t = tk+h/2, y(tk+h/2) = yk12, z(tk+h/2) = zk12, ode[1]));
    zk1 := evalf(zk + h * subs(t = tk+h/2, y(tk+h/2) = yk12, z(tk+h/2) = zk12, ode[2]));


    return <yk1,zk1>;
  end proc;
```

```
collatzStep := proc(tk, h, yk, zk, ode)
    local yk12, zk12, yk1, zk1;
    yk12 := yk + 1/2*h*subs(t = tk, y(tk) = yk, z(tk) = zk, ode[1]);
    zk12 := zk + 1/2*h*subs(t = tk, y(tk) = yk, z(tk) = zk, ode[2]);
    yk1 := evalf(yk + h*subs(t = tk + 1/2*h, y(tk + 1/2*h) = yk12, z(tk + 1/2*h) = zk12, ode[1]));
    zk1 := evalf(zk + h*subs(t = tk + 1/2*h, y(tk + 1/2*h) = yk12, z(tk + 1/2*h) = zk12, ode[2]));
    return <yk1, zk1>
end proc
```

(4.2.1.2.1)

4.2.1.3 Heun method

The Heun method is the following:

$$Yp = Y_n + h\,f\left(t_n, Y_n\right)$$

$$Y_{n+1} = Y_n + \frac{h\left(f\left(t_n, Y_n\right) + f\left(t_{n+1}, Yp\right)\right)}{2}$$

```
> heunStep := proc(h,tk,x1k,x2k,yk,ode)
    local x1k1, x2k1, yk1 ,x1p, x2p, yp :
    x1p := x1k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, y(tk)=yk, ode[1]) :
    x2p := x2k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, y(tk)=yk, ode[2]) :
    yp := yk + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, y(tk)=yk, ode[3]) :

    x1k1 := x1k + h*( subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, y(tk)=yk, ode[1]) + subs(t=tk+h, x1
  (tk+h)=x1p, x2(tk+h)=x2p, y(tk+h)=yp, ode[1])) /2;
    x2k1 := x2k + h*( subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, y(tk)=yk, ode[2]) + subs(t=tk+h, x1
  (tk+h)=x1p, x2(tk+h)=x2p, y(tk+h)=yp, ode[2])) /2;
    yk1 := yk + h*( subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, y(tk)=yk, ode[3]) + subs(t=tk+h, x1
  (tk+h)=x1p, x2(tk+h)=x2p, y(tk+h)=yp, ode[3])) /2;


    return < evalf(x1k1), evalf(x2k1), evalf(yk1)> :
  end proc;
```

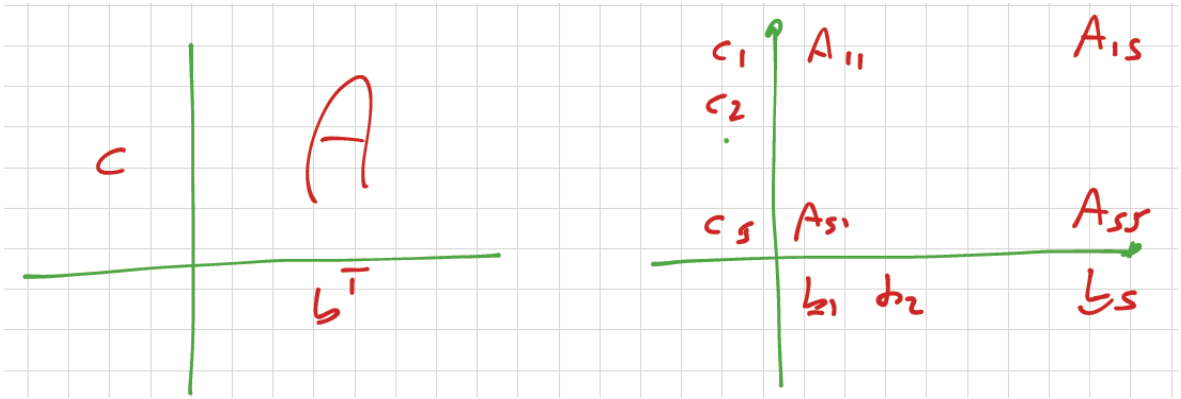
```
heunStep := proc(h, tk, x1k, x2k, yk, ode)
    local x1k1, x2k1, yk1, x1p, x2p, yp;
    x1p := x1k + h*subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, y(tk) = yk, ode[1]);
```

(4.2.1.3.1)

```
    x2p := x2k + h*subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, y(tk) = yk, ode[2]);
    yp := yk + h*subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, y(tk) = yk, ode[3]);
    x1k1 := x1k + 1/2*h*(subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, y(tk) = yk, ode[1]) + subs(t = tk + h,
    x1(tk + h) = x1p, x2(tk + h) = x2p, y(tk + h) = yp, ode[1]));
    x2k1 := x2k + 1/2*h*(subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, y(tk) = yk, ode[2]) + subs(t = tk + h,
    x1(tk + h) = x1p, x2(tk + h) = x2p, y(tk + h) = yp, ode[2]));
    yk1 := yk + 1/2*h*(subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, y(tk) = yk, ode[3]) + subs(t = tk + h, x1(tk
     + h) = x1p, x2(tk + h) = x2p, y(tk + h) = yp, ode[3]));
    return < evalf(x1k1), evalf(x2k1), evalf(yk1) >
end proc
```

4.2.1.4 Runge–Kutta methods

Runge-Kutta methods are a family of methods that can be represented witht the following tableau:



That corresponds to the formula:

THE FORM

number of stages

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i$$

$$\begin{cases} k_1 = h f\left(x_n + c_1 h,\ y_n + \sum_{j=1}^{s} A_{1j} k_j\right) \\ k_2 = h f\left(x_n + c_2 h,\ y_n + \sum_{j=1}^{s} A_{2j} k_j\right) \\ \vdots \\ k_s = h f\left(x_n + c_s h,\ y_n + \sum_{j=1}^{s} A_{sj} k_j\right) \end{cases}$$

An example could be:

0	0	0	0	0
1/3	1/3	0	0	0
2/3	-1/3	1	0	0
1	1	-1	1	0
	1/8	3/8	3/8	1/8

correspond to the method

$$y_{k+1} = y_k + \frac{1}{8}K_1 + \frac{3}{8}K_2 + \frac{3}{8}K_3 + \frac{1}{8}K_4$$

$$K_1 = hf(x_k, y_k)$$

$$K_2 = hf\left(x_k + \frac{1}{3}h, y_k + \frac{1}{3}K_1\right)$$

$$K_3 = hf\left(x_k + \frac{2}{3}h, y_k - \frac{1}{3}K_1 + K_2\right)$$

$$K_4 = hf\left(x_k + h, y_k + K_1 - K_2 + K_3\right)$$

Using the Runge-Kutta formulation we can represent various other numerical methods:

Explicit Euler:

0	0
	1

$$y_{k+1} = y_k + 1 \cdot K_1$$

$$K_1 = hf(x_k + 0h, y_k + 0K_1) = hf(x_k, y_k)$$

$$\Rightarrow y_{k+1} = y_k + hf(x_k, y_k)$$

Implicit Euler:

1	1
	1

$$y_{k+1} = y_k + 1\,k_2 = y_k + k_1$$

$$k_1 = h\,f(x_k + 1\cdot h,\ y_k + 1\,k_1)$$

$$= h\,f(x_k,\ y_k + k_1)$$

$$y'_{k+1}$$

$$\Rightarrow y_{k+1} = y_k + h\,f(x_k,\ y_{k+1})$$

The Heun method:

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$

$$y_{k+1} = y_k + \frac{1}{2}k_1 + \frac{1}{2}k_2$$

$$k_1 = h\,f(x_k + 0\cdot h,\ y_k)$$

$$k_2 = h\,f(x_k + 1\cdot h,\ y_k + 1\,k_1)$$

$$k_1 = h\,f(x_k, y_k)$$

$$k_2 = h\,f(x_{k+1},\ y_k + k_1) = h\,f(x_{k+1},\ y_k + h\,f(x_k, y_k))$$

$$\tilde{y}_{k+1}$$

$$\tilde{y}_{k+1} = y_k + h\,f(x_k, y_k) \qquad (P)$$

$$y_{k+1} = y_k + \frac{h}{2}\left(f(x_k, y_k) + f(x_{k+1},\ \tilde{y}_{k+1})\right) \qquad (C)$$

The Collaz method:

0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0
	0	1

$$y_{n+1} = y_n + 1 \cdot k_2$$

$$k_1 = h f(x_n, y_n)$$

$$k_2 = h f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2} k_1\right)$$

$$k_2 = h f\left(\underbrace{x_n + \frac{h}{2}}_{x_{n+\frac{1}{2}}}, \underbrace{y_n + \frac{h}{2} f(x_n, y_n)}_{y_{n+\frac{1}{2}}}\right)$$

$$\begin{cases} y_{n+\frac{1}{2}} = y_n + \frac{h}{2} f(x_n, y_n) \\ y_{n+1} = y_n + h f\left(x_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}\right) \end{cases}$$

And then the **Classical Runge-Kutta 4 Explicit:**

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

$$y_{n+1} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

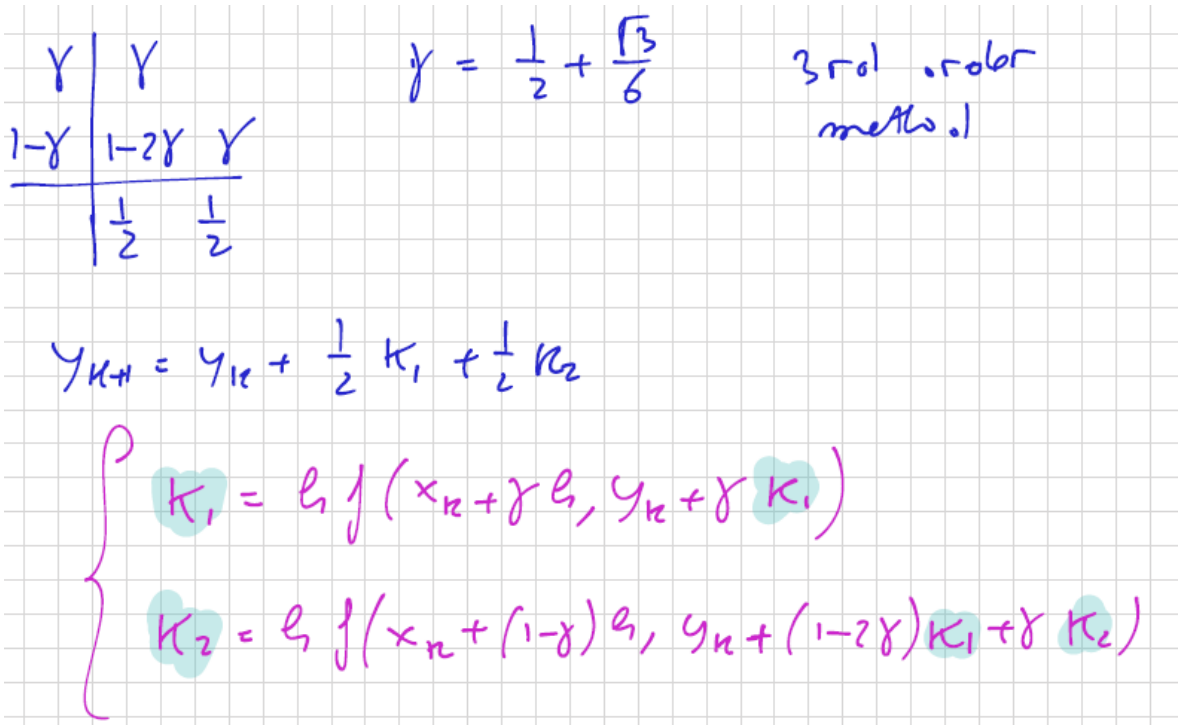
$$k_1 = h f(x_n, y_n)$$

$$k_2 = h f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2} k_1\right)$$

$$k_3 = h f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2} k_2\right)$$

$$k_4 = h f(x_{n+1}, y_n + k_3)$$

And the **Classical Runge-Kutta Implicit** :



Now that we have our way to compute a method step, it's time to create a loop to advance with the steps and fill out results matrix:

```
> for i from 2 to nsteps do
    tmp := EulerStep(h*(i-1), h, results[i-1,1], results[i-1,2], results[i-1,3], rhs_ode):
    results[i,1] := tmp[1]:
    results[i,2] := tmp[2]:
    results[i,3] := tmp[3]:
  end do:
> results; #this contains the results of our numerical method, each column is a variable and each
  row a step
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 1.1 & 2. & 2.70 \\ 1.2 & 1.99 & 2.402 \\ 1.3 & 1.97 & 2.110 \\ 1.4 & 1.94 & 1.828 \\ 1.5 & 1.90 & 1.560 \\ 1.6 & 1.85 & 1.310 \\ 1.7 & 1.79 & 1.082 \\ 1.8 & 1.72 & 0.880 \\ 1.9 & 1.64 & 0.708 \end{bmatrix} \tag{4.2.6}$$

We now need to find a way to plot these numbers in an easy way.

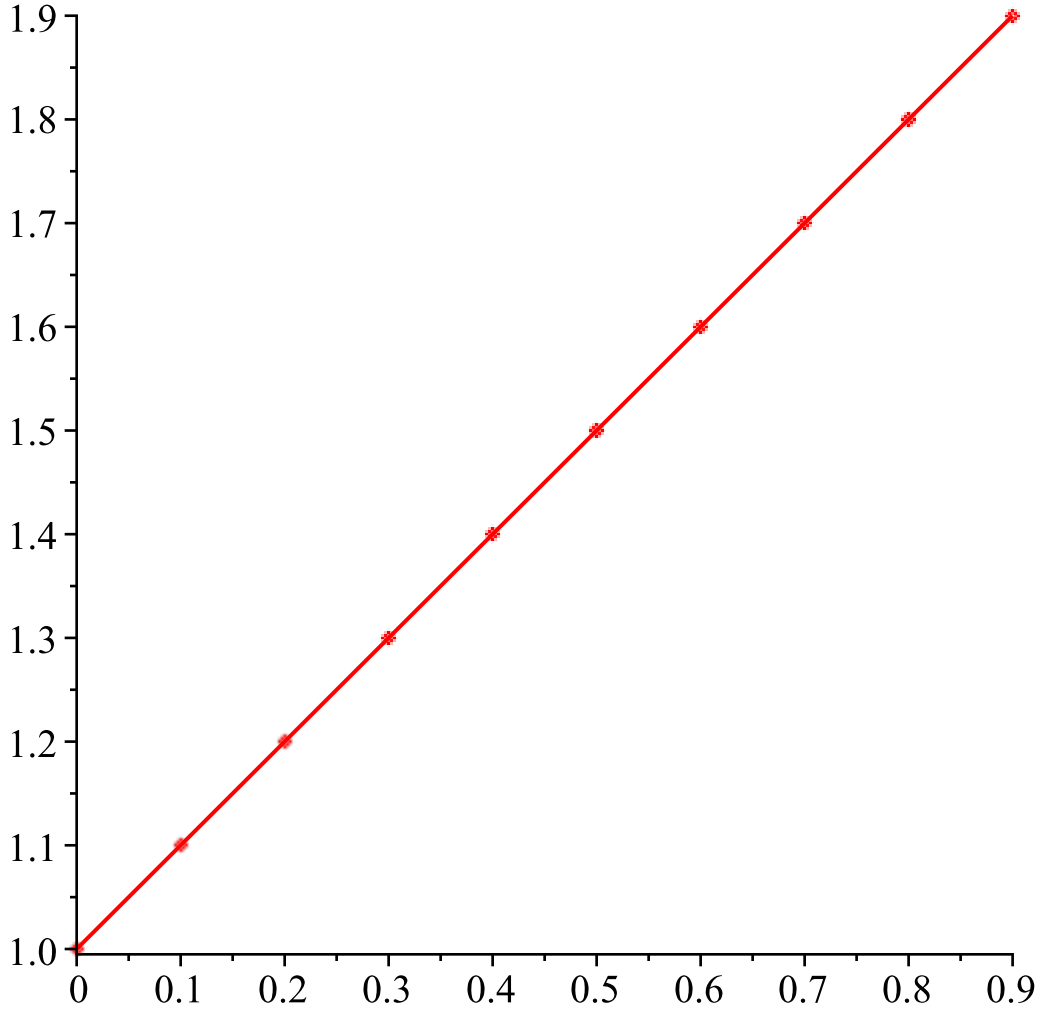
4.2.2 Maple pointplot

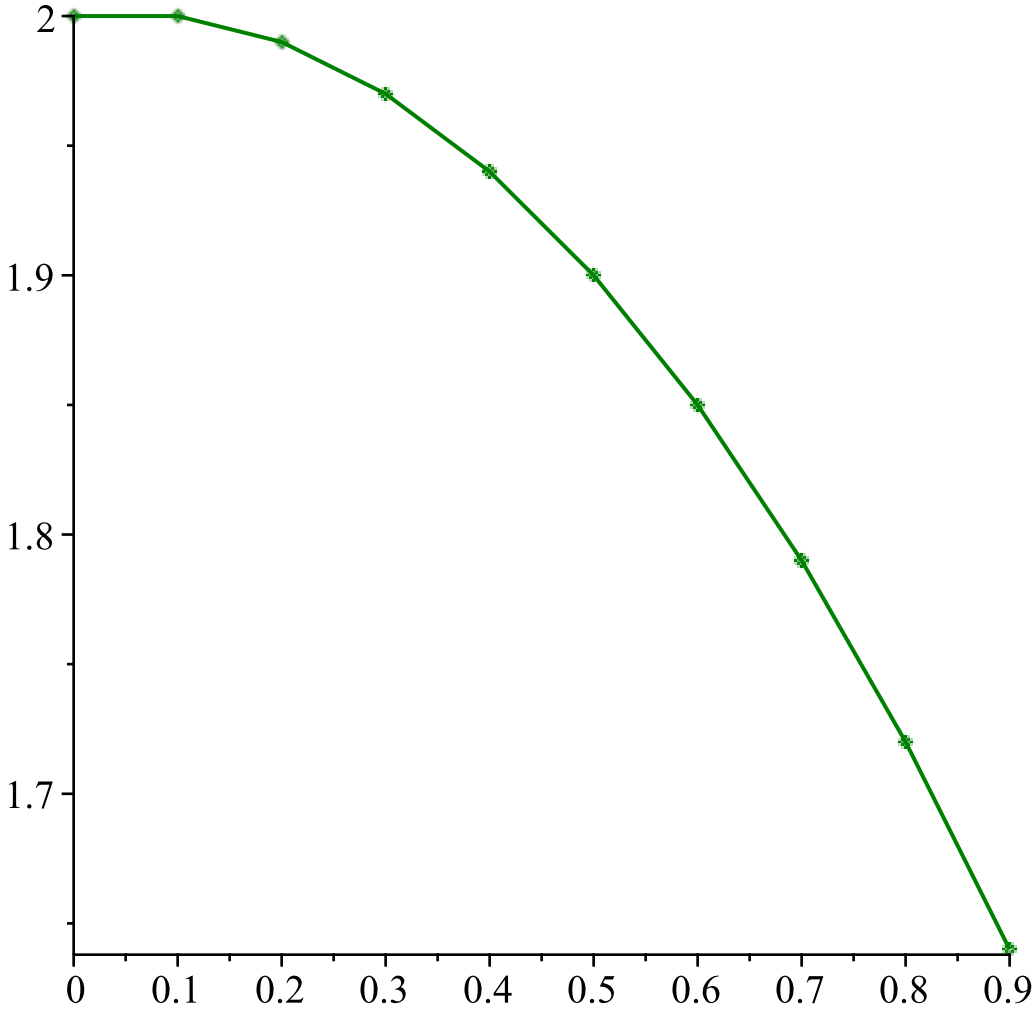
Maple pointplot permit us to plot and connect the points, but we need first to do some passages.
We basically have to plot matrices of nsteps rows and two columns, and the first column is the value of time, and the second is the value of the function at that time.

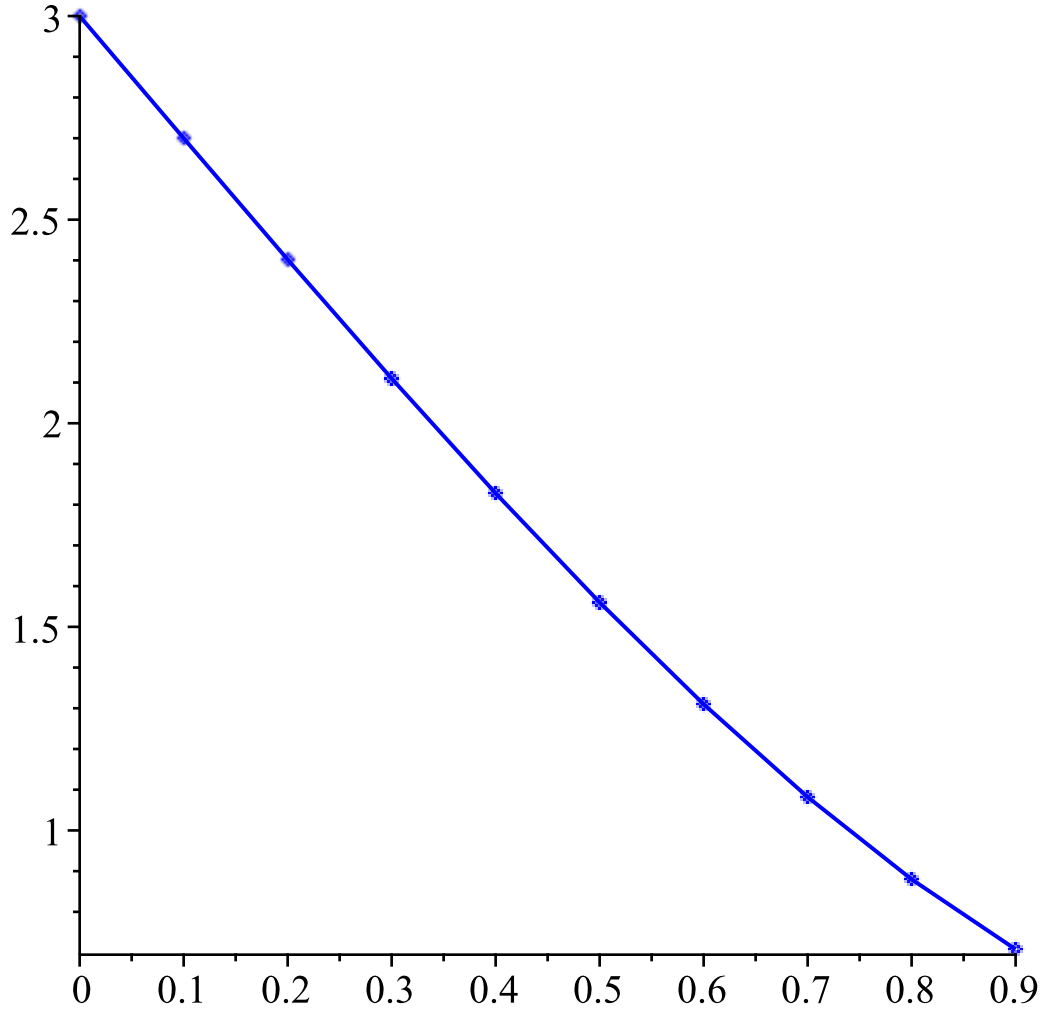
```
> <<t_step[1..nsteps]>|results[1..nsteps,1]>;
```

$$\begin{bmatrix} 0. & 1 \\ 0.1 & 1.1 \\ 0.2 & 1.2 \\ 0.3 & 1.3 \\ 0.4 & 1.4 \\ 0.5 & 1.5 \\ 0.6 & 1.6 \\ 0.7 & 1.7 \\ 0.8 & 1.8 \\ 0.9 & 1.9 \end{bmatrix} \quad \textbf{(4.2.2.1)}$$

```
> plots[pointplot](<<t_step[1..nsteps]>|results[1..nsteps,1]>,color="Red",symbol=diamond,
  connect=true,style=pointline );
  plots[pointplot](<<t_step[1..nsteps]>|results[1..nsteps,2]>,color="Green",symbol=diamond,
  connect=true,style=pointline );
  plots[pointplot](<<t_step[1..nsteps]>|results[1..nsteps,3]>,color="Blue",symbol=diamond,
  connect=true,style=pointline );
```




2
1.9
1.8
1.7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9



4.3 Baumgarte Stabilization

Instead of directly solving the constraint equations we can subtitute them with the Baumgarten stabilization, which for the index-3 dae has the form:

$$\frac{d^2}{dt^2}\mathbf{\Phi}(q(t),t) + 2\zeta\omega\frac{d}{dt}\mathbf{\Phi}(q(t),t) + \omega^2\mathbf{\Phi}(q(t),t) = \mathbf{0}$$

```
> eq_stab := alg3[1] + 2*zeta*omega__n*alg2[1] + omega__2^2*alg1[1];
```

$$eq\_stab := 2\,\zeta\,\omega_n\left(q2(t) - x2(t) - \frac{d}{dt}\,q3(t)\right) + \omega_2^{\,2}\left(-q3(t) + \eta\,tx2(t) + x3(t)\right) - q1(t) + x1(t) + \frac{d}{dt}\,q2(t) - \frac{d^2}{dt^2}\,q3(t) \quad \textbf{(4.3.1)}$$

```
> # or # Phi_Baum := <diff(Phi,t,t)> + zeta*omega__n*<diff(Phi,t)> + omega__n^2*<Phi>;
> # eq_stab := expand(subs(dae[1..2],diff(dae[3],t,t))) + 2*zeta*omega__n*expand(subs(dae[1..2],
```

diff(dae[3],t)))+ omega__n^2*dae[3]; # or we can compute it directly from the dae

We can do various things now.
If the DAE has index 2 we can substitute the stabilized equation in the place of the algebric part in the DAE, we would have a ODE and we can solve it with less drift.
If the DEA has index 3 we can substitute the stabilized equation in the place of the algebric part in the DAE, and do one or two things:

- Reduce again the DAE to have an ODE and then solve the ODE;
- We know that in MultiBodySystems the reaction forces / lagrange multiplier appear linearly. So we could derive two times the constraint equations and have a system that depends on the accellerations and on the multipliers. We could solve the accellerations as an ODE and then put the results in the multipliers to have them, but this solution would drift a lot. Or we can substitute the constraint equations with their Baumgarten stabilization, and still have something that depends on the accellerations and on the multipliers. So we could solve the accellerations as an ODE and then put the results in the multipliers to have them and this solution would drift less because there is the Baumgarte stabilization.

If we solve the stabilized differential equation and we **plot the invariants** we will be able to see that in the stabilized solution the invariats are very close to 0 and they drift very little, instead in the non-stabilizad solution we would see the invariants not always close to 0 and more importantly they would drift greatly.

Hint for the parameters from Biral: **Wn = 20** and **eta = 0.95**. Usually Bertolazzi set them like: **Wn = 20** and **eta = 1**.

If we have an index-2 DAE and we apply Baumgarte as before we will have an ODE, instead if we have an index-3 DAE and we apply Baumgarte as brfore we will obtain an index-1 DAE. This is not a problem, but if we want to apply Baumgarte and have an ODE we need to do this:

> # eq_stab := alg4 + 6*omega*alg3 + 11*omega^2*alg2 + 6*omega^3*alg1;

4.4 Projection Method

The projection method is a method to stabilize better the numerical solution of a DAE. In practice after every step of a numerical method we perform a step of projection.
In projection we try to project the solution of a step of a numerical method on the constraints, in this way we avoid drift because our solution is always very close to respecting the constraints.
We do this in this way (q0 is the solution of the numerical step, and q are the variables of the DAE):

MINIMIZE || q - q0||^2

SUBJECT TO: Phi(q,t) = 0

The result of this process is qk+1.

In practice to do this we have to use the Minimize function of Maple. First we import the optimization library:

> with(Optimization):

Then we learn how to use the minimize function:
Minimize (objective_function, {constraints});
as a results we obtain the value of the objective function (closer to 0 is better) and the value of the variables in the objective function in this form:
[objective_function_value, [variables values]]

As an example:

> (x1 - x10)^2;
Minimize(subs(x10 = 5,%),{x1=5});

$$(x1 - x10)^2$$
$$[0., [x1 = 5.]] \qquad \textbf{(4.4.1)}$$

In our case we want to minimize something similar:

```
> to_minimize := (x1-x10)^2 + (x2-x20)^2 + (x3-x30)^2;
```

$$to\_minimize := (x1 - x10)^2 + (x2 - x20)^2 + (x3 - x30)^2 \tag{4.4.2}$$

With x10, x20 and x30 the results of the numerical method.
And the contraints will be (substitute the x2(t) with x2 and the algs should be alg[1] =0, alg2 = 0 ecc...) :

```
> algs:<%>;
  data := [q1(t) = 1, q2(t) = 1, q3(t) = 0,eta=1]:
  eval(subs(data,algs)):
   [%[1] = 0, %[2] = 0, %[3] = 0]:
  constr := subs(x1(t)=x1, x2(t)=x2, x3(t)=x3,%):<%>;
```

$$\begin{bmatrix} -q3(t) + \eta\, t\, x2(t) + x3(t) \\ q2(t) - x2(t) - \frac{d}{dt}\, q3(t) \\ -q1(t) + x1(t) + \frac{d}{dt}\, q2(t) - \frac{d^2}{dt^2}\, q3(t) \end{bmatrix}$$

$$\begin{bmatrix} t\, x2 + x3 = 0 \\ 1 - x2 = 0 \\ -1 + x1 = 0 \end{bmatrix} \tag{4.4.3}$$

So:

```
> Minimize( subs(x10=0,x20=0,x30=0, to_minimize), subs(t=0,constr));
```

$$[2., [x1 = 1., x2 = 1., x3 = 0.]] \tag{4.4.4}$$

Therefore the Projection step after the numerical step should look something like this:

```
> ProjectionStep := proc( q0, tk, constr)
    local res,to_minimize;
    to_minimize := (x1-x10)^2+(x2-x20)^2+(x3-x30)^2;
    res := Minimize ( subs(x10=q0[1],x20=q0[2],x30=q0[3],to_minimize), subs(t=tk,constr));
    return <rhs(res[2,1]),rhs(res[2,2]),rhs(res[2,3])>;
  end proc;
```

```
ProjectionStep := proc(q0, tk, constr)
    local res, to_minimize;
    to_minimize := (x1 - x10)^2 + (x2 - x20)^2 + (x3 - x30)^2;
    res := Optimization:-Minimize(subs(x10 = q0[1], x20 = q0[2], x30 = q0[3], to_minimize), subs(t = tk, constr));
    return <rhs(res[2, 1]), rhs(res[2, 2]), rhs(res[2, 3])>
end proc
```

(4.4.5)

5 Full Example

Let's do a full example.
Initialize Maple:

```
> restart:
> with(LinearAlgebra):
  with(plots):
```

This is the DAE we have to solve, with the following variables:

```
> dae := [diff(x2(t),t) = sin(t) - x1(t),
```

```
diff(x3(t),t) = 1 -(1+1)*x2(t) -t*(sin(t) -x1(t)),
2-t*x2(t) -x3(t)=0]: <%>;
```

$$\begin{bmatrix} \frac{d}{dt}\,x2(t) = \sin(t) - x1(t) \\ \frac{d}{dt}\,x3(t) = 1 - 2\,x2(t) - t\,(\sin(t) - x1(t)) \\ 2 - t\,x2(t) - x3(t) = 0 \end{bmatrix} \tag{5.1}$$

```
> vars := [x1(t), x2(t), x3(t)];
```

$$vars := [x1(t), x2(t), x3(t)] \tag{5.2}$$

Put the DAE in explicit form:

```
> E,G := GenerateMatrix(dae, diff(vars,t) ):
  E,<diff(vars,t)>,G;
```

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} \frac{d}{dt}\,x1(t) \\ \frac{d}{dt}\,x2(t) \\ \frac{d}{dt}\,x3(t) \end{bmatrix}, \begin{bmatrix} \sin(t) - x1(t) \\ 1 - 2\,x2(t) - t\,(\sin(t) - x1(t)) \\ -2 + t\,x2(t) + x3(t) \end{bmatrix} \tag{5.3}$$

Compute the index with the library:

```
> KtLtbuild := proc( E )
    local P, L, U, r, LtKt, Lt, Kt;          # definition of the local variables
    P,L,U := LUDecomposition( E );           # LU decomposition by Maple
    r := Rank( E );                          # the first r rows of the resulting matrix are the L matrix
  , the other is the K matrix
    LtKt := LinearSolve(L,Transpose(P));     # compute L^(-1).P
    Lt   := LtKt[1..r,1..-1];
    Kt   := LtKt[r+1..-1,1..-1];
    return Kt, Lt;
  end proc;
```

```
KtLtbuild := proc(E)
    local P, L, U, r, LtKt, Lt, Kt;
    P, L, U := LinearAlgebra:-LUDecomposition(E);
    r := LinearAlgebra:-Rank(E);
    LtKt := LinearAlgebra:-LinearSolve(L, LinearAlgebra:-Transpose(P));
    Lt := LtKt[1..r, 1..-1];
    Kt := LtKt[r+1..-1, 1..-1];
    return Kt, Lt
end proc
```

(5.4)

```
> ReduceBy1TheIndex := proc( EQS, Dvars )
    local E, G, Kt, Lt, DPART, ALG, DALG;     # local variables
    E, G   := GenerateMatrix(EQS,Dvars);      # generate the matrix
    Kt, Lt := KtLtbuild( E );                 # get K and L (both transpose)
    # Separate Algebraic and Differential part
    DPART := simplify(Lt.(E.<Dvars>-G));      # simplify is important
    ALG   := simplify(Kt.(E.<Dvars>-G));
    DALG  := diff(ALG,t);
    # Build the modified DAE, by subtuting the algebraic equation(s) with the derivative of the
  algebraic equation(s)
    return <DPART,DALG>, ALG;
```

```
  end proc;
ReduceBy1TheIndex := proc(EQS, Dvars)
    local E, G, Kt, Lt, DPART, ALG, DALG;
    E, G := LinearAlgebra:-GenerateMatrix(EQS, Dvars);
    Kt, Lt := KtLtbuild(E);
    DPART := simplify( `.`(Lt, `.`(E, < Dvars > ) - G));
    ALG := simplify( `.`(Kt, `.`(E, < Dvars > ) - G));
    DALG := diff(ALG, t);
    return < DPART, DALG > , ALG
end proc
```

(5.5)

```
> dae1,alg1 := ReduceBy1TheIndex(dae,diff(vars,t));
```

$$dae1,\ alg1 := \left[\begin{array}{c} \frac{d}{dt}\ x2(t) - \sin(t) + x1(t) \\ t\sin(t) - t\,x1(t) + \frac{d}{dt}\ x3(t) + 2\,x2(t) - 1 \\ -x2(t) - t\left(\frac{d}{dt}\ x2(t)\right) - \frac{d}{dt}\ x3(t) \end{array}\right], \left[\ 2 - t\,x2(t) - x3(t)\ \right] \tag{5.6}$$

```
> dae2,alg2 := ReduceBy1TheIndex( convert(dae1,list) ,diff(vars,t));
```

$$dae2,\ alg2 := \left[\begin{array}{c} \frac{d}{dt}\ x2(t) - \sin(t) + x1(t) \\ t\sin(t) - t\,x1(t) + \frac{d}{dt}\ x3(t) + 2\,x2(t) - 1 \\ \frac{d}{dt}\ x2(t) \end{array}\right], \left[\ x2(t) - 1\ \right] \tag{5.7}$$

```
> dae3,alg3 := ReduceBy1TheIndex( convert(dae2,list) ,diff(vars,t));
```

$$dae3,\ alg3 := \left[\begin{array}{c} \frac{d}{dt}\ x2(t) - \sin(t) + x1(t) \\ t\sin(t) - t\,x1(t) + \frac{d}{dt}\ x3(t) + 2\,x2(t) - 1 \\ \cos(t) - \frac{d}{dt}\ x1(t) \end{array}\right], \left[\ \sin(t) - x1(t)\ \right] \tag{5.8}$$

```
> dae4, alg4  := ReduceBy1TheIndex( convert(dae3,list) ,diff(vars,t));
  # the algebric part is empty, therefore the previous dae was the ode and the previous algebraic part
  was the last one
```

$$dae4,\ alg4 := \left[\begin{array}{c} \cos(t) - \frac{d}{dt}\ x1(t) \\ \frac{d}{dt}\ x2(t) - \sin(t) + x1(t) \\ t\sin(t) - t\,x1(t) + \frac{d}{dt}\ x3(t) + 2\,x2(t) - 1 \end{array}\right], [\ \ ] \tag{5.9}$$

```
> ode := dae3;
```

$$ode := \left[\begin{array}{c} \frac{d}{dt}\ x2(t) - \sin(t) + x1(t) \\ t\sin(t) - t\,x1(t) + \frac{d}{dt}\ x3(t) + 2\,x2(t) - 1 \\ \cos(t) - \frac{d}{dt}\ x1(t) \end{array}\right] \tag{5.10}$$

```
> algs := [ alg1[1],  alg2[1], alg3[1] ]:<%>;
```

$$\begin{bmatrix} 2 - t\,x2(t) - x3(t) \\ x2(t) - 1 \\ \sin(t) - x1(t) \end{bmatrix} \tag{5.11}$$

Compute the index by manual substitution

> al1 := dae[3];

$$al1 := 2 - t\,x2(t) - x3(t) = 0 \tag{5.12}$$

> al2 := diff(al1, t);
al2 := subs(dae[1], dae[2], al2);

$$al2 := -x2(t) - t\left(\frac{\mathrm{d}}{\mathrm{d}t}\,x2(t)\right) - \frac{\mathrm{d}}{\mathrm{d}t}\,x3(t) = 0$$
$$al2 := x2(t) - 1 = 0 \tag{5.13}$$

> al3 := diff(al2, t);
al3 := subs(dae[1], dae[2], al3);

$$al3 := \frac{\mathrm{d}}{\mathrm{d}t}\,x2(t) = 0$$
$$al3 := \sin(t) - x1(t) = 0 \tag{5.14}$$

> al4 := diff(al3, t); # this is the differential part
al4 := subs(dae[1], dae[2], al4);

$$al4 := \cos(t) - \frac{\mathrm{d}}{\mathrm{d}t}\,x1(t) = 0$$
$$al4 := \cos(t) - \frac{\mathrm{d}}{\mathrm{d}t}\,x1(t) = 0 \tag{5.15}$$

> # ode := [dae[1], dae[2], al4];
> # algs := [al1, al2, al3]: <%>;

Create the ODE and put it in the explicit form

> E,G := GenerateMatrix(convert(ode,list), diff(vars,t));

$$E,\ G := \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} \sin(t) - x1(t) \\ -t\sin(t) + t\,x1(t) - 2\,x2(t) + 1 \\ -\cos(t) \end{bmatrix} \tag{5.16}$$

> ex_ode := solve(convert(ode,list), diff(vars,t))[1]:<%>;

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}\,x1(t) = \cos(t) \\ \frac{\mathrm{d}}{\mathrm{d}t}\,x2(t) = \sin(t) - x1(t) \\ \frac{\mathrm{d}}{\mathrm{d}t}\,x3(t) = -t\sin(t) + t\,x1(t) - 2\,x2(t) + 1 \end{bmatrix} \tag{5.17}$$

> E,G := GenerateMatrix(convert(ex_ode,list), diff(vars,t));

$$E,\ G := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \cos(t) \\ \sin(t) - x1(t) \\ -t\sin(t) + t\,x1(t) - 2\,x2(t) + 1 \end{bmatrix} \tag{5.18}$$

Find the initial conditions, in this case we have three invariants and three variables, so we only need to put t=0 and solve the invariants, if we wold have two invariants and three equations variables we would have need an initial condition for a variable.

> <algs>;

(5.19)

$$\begin{bmatrix} 2 - t\,x2(t) - x3(t) \\ x2(t) - 1 \\ \sin(t) - x1(t) \end{bmatrix} \tag{5.19}$$

```
> subs( x1(t) = x1, x2(t) = x2, x3(t) = x3, algs);
  solve(subs( t=0, %), [x1,x2,x3]);
  evalf(%[1]);
  ics := subs(x1 = x1(t), x2= x2(t), x3= x3(t), %);
```

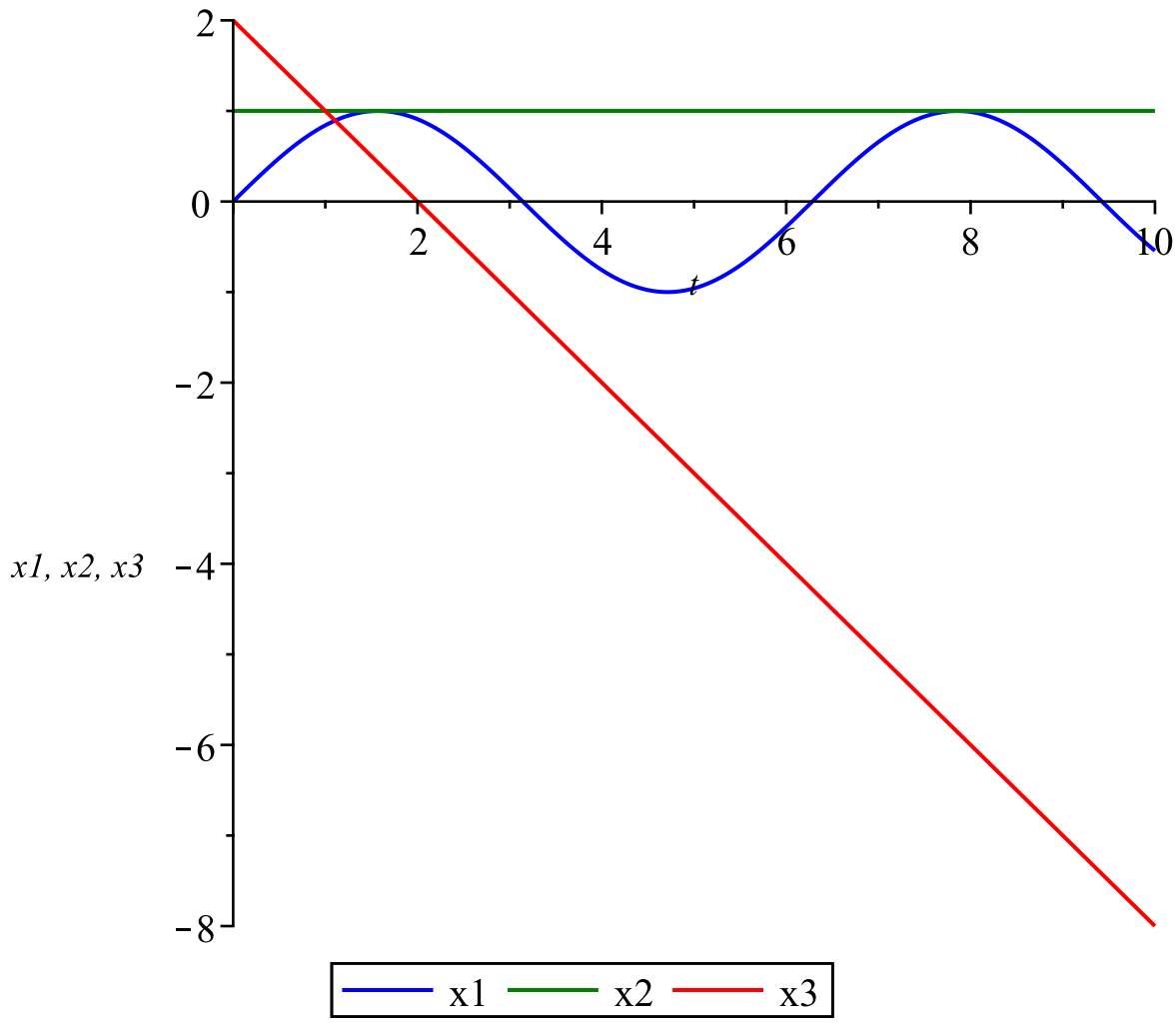
$$[-t\,x2 - x3 + 2,\ x2 - 1,\ \sin(t) - x1]$$
$$[[x1 = \sin(0),\ x2 = 1,\ x3 = 2]]$$
$$[x1 = 0.,\ x2 = 1.,\ x3 = 2.]$$
$$ics := [x1(t) = 0.,\ x2(t) = 1.,\ x3(t) = 2.] \tag{5.20}$$

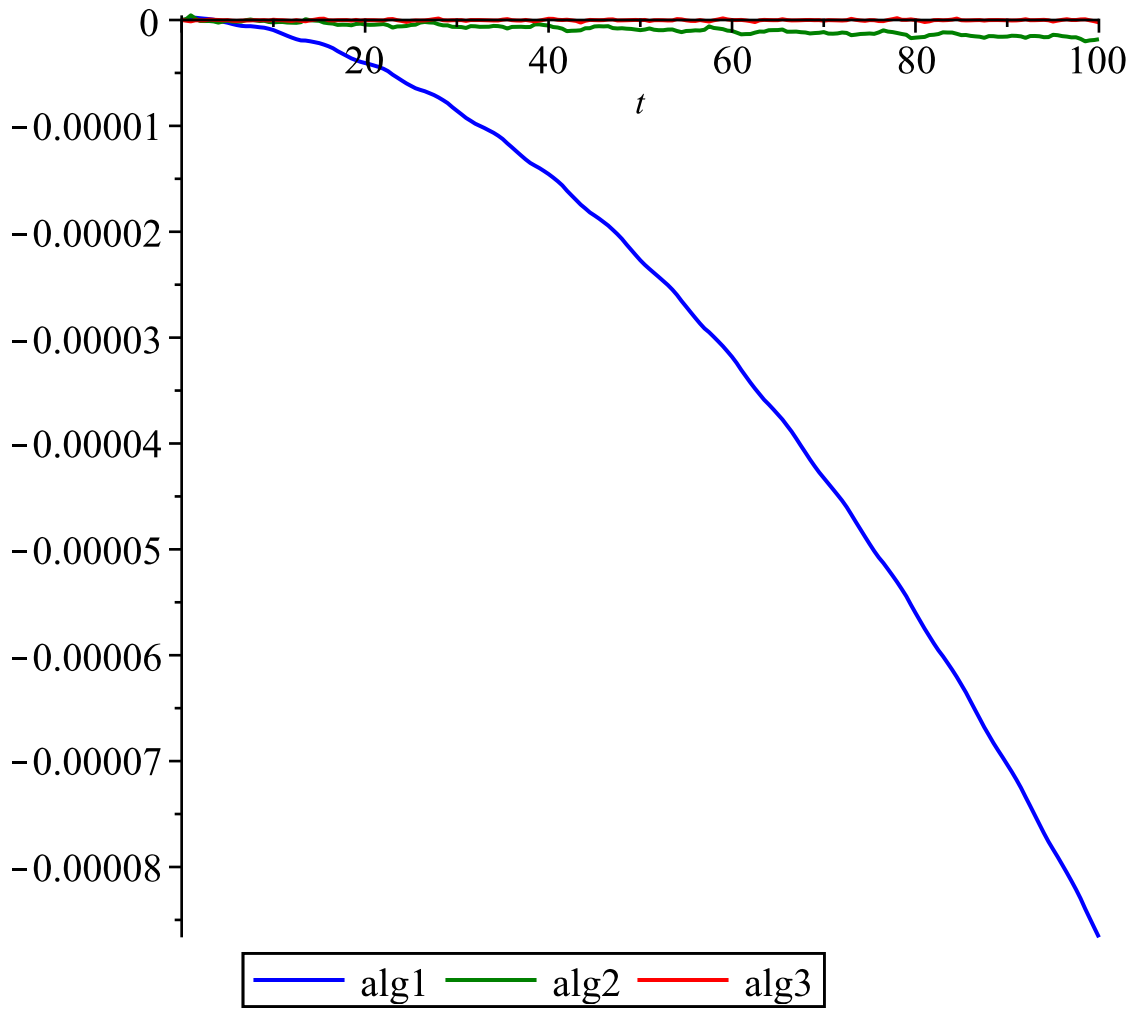
Solve the ODE with Maple:

```
> sol_ode_maple := dsolve( convert(ode,set) union convert(subs(t=0,ics),set) , numeric);
```

$$sol\_ode\_maple := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc} \tag{5.21}$$

```
> odeplot(sol_ode_maple,[[t,x1(t),color="Blue"],[t,x2(t),color="Green"],[t,x3(t),color="Red"]], t=0.
  .10,legend=["x1","x2","x3"]);
  odeplot(sol_ode_maple,[[t,alg1[1],color="Blue"],[t,alg2[1],color="Green"],[t,alg3[1],color="Red"]],
  t=0..100,legend=["alg1","alg2","alg3"]);
```





Stabilize with Baumgarte, solve and confront the invariants. As you can see the invariants in the non-stabilized ODE arrive to 8*10^(-5) after 100 seconds and we can see there is a trend towards drifting, while with the stabilized with Baumgarte wa arrive to 6*10^(-7) and we don't have a drift attitude after 100 seconds.

> baum_part := alg3[1] + eta*omega__n*alg2[1] + omega__n*alg3[1];

$$baum\_part := \eta\,\omega_n\,(x2(t) - 1) + \omega_n\,(\sin(t) - x1(t)) + \sin(t) - x1(t) \tag{5.22}$$

> baum_dae := [dae[1], dae[2], baum_part]:<%>;

$$\begin{bmatrix} \frac{d}{dt}\,x2(t) = \sin(t) - x1(t) \\ \frac{d}{dt}\,x3(t) = 1 - 2\,x2(t) - t\,(\sin(t) - x1(t)) \\ \eta\,\omega_n\,(x2(t) - 1) + \omega_n\,(\sin(t) - x1(t)) + \sin(t) - x1(t) \end{bmatrix} \tag{5.23}$$

> baum_ode, useless := ReduceBy1TheIndex(baum_dae, diff(vars,t));

$$baum\_ode,\ useless := \begin{bmatrix} \frac{d}{dt}\,x2(t) - \sin(t) + x1(t) \\ t\sin(t) - t\,x1(t) + \frac{d}{dt}\,x3(t) + 2\,x2(t) - 1 \\ (\omega_n + 1)\cos(t) + (-\omega_n - 1)\left(\frac{d}{dt}\,x1(t)\right) + \eta\,\omega_n\left(\frac{d}{dt}\,x2(t)\right) \end{bmatrix}, \tag{5.24}$$

$$\left[\ (\omega_n + 1)\sin(t) + (-\omega_n - 1)\,x1(t) + \eta\,\omega_n\,(x2(t) - 1)\ \right]$$

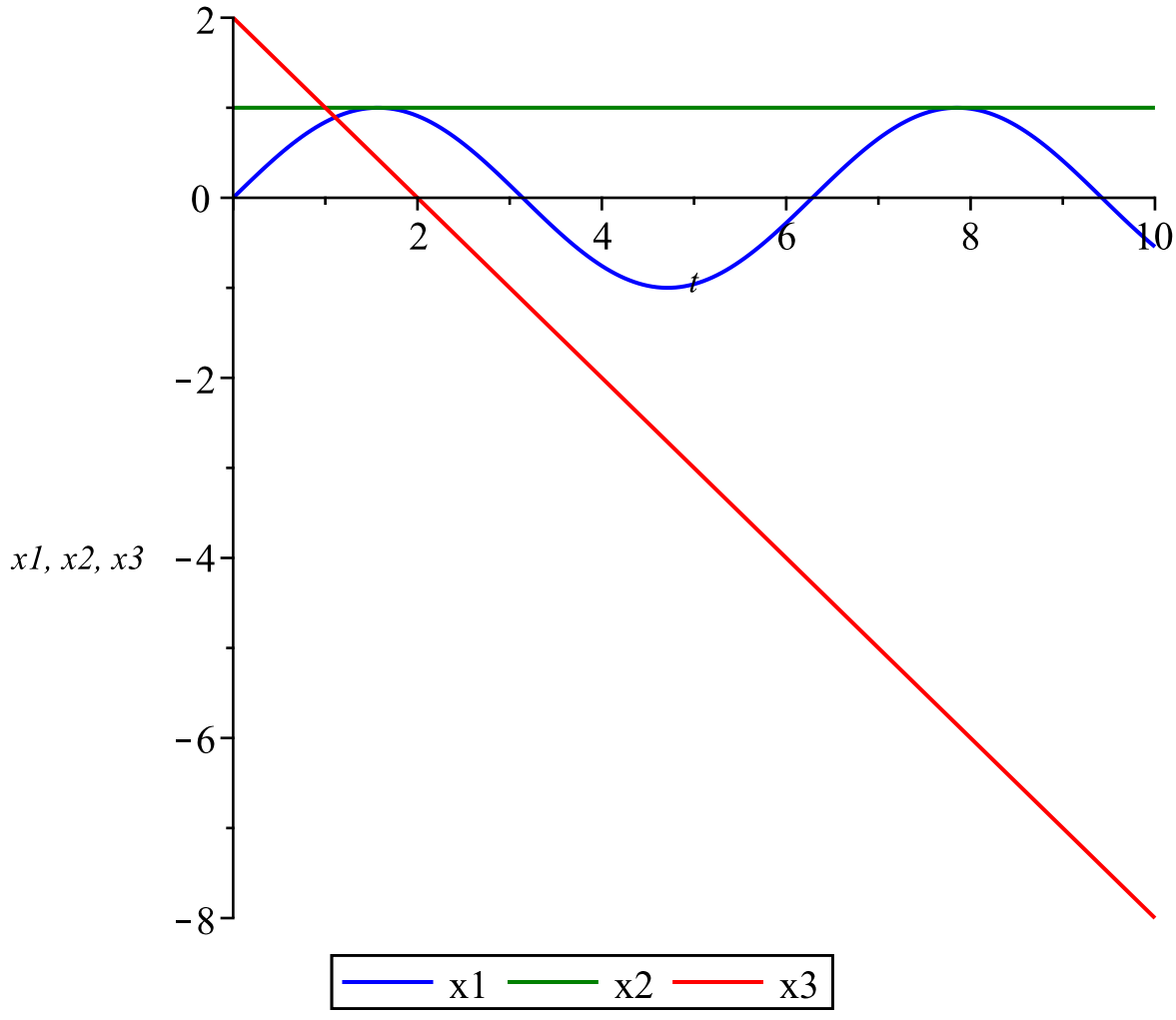
```
> ex_baum := solve(convert(baum_ode,list), diff(vars,t))[1]:<%>;
```

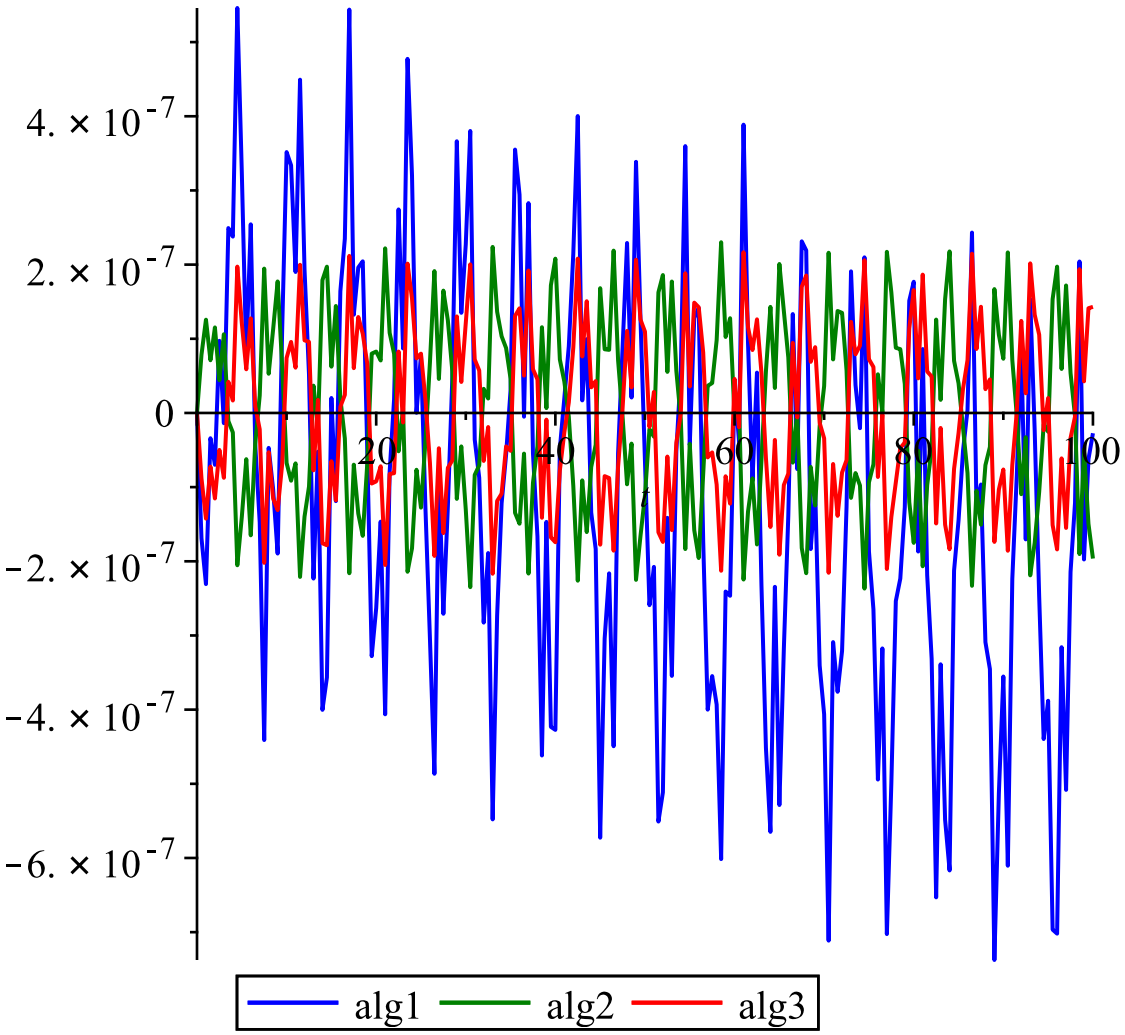
$$\left[\frac{d}{dt}x1(t) = \frac{\eta\,\omega_n \sin(t) - \eta\,\omega_n x1(t) + \cos(t)\,\omega_n + \cos(t)}{\omega_n + 1},\ \frac{d}{dt}x2(t) = \sin(t) - x1(t),\ \frac{d}{dt}x3(t) = -t\sin(t) + t\,x1(t) - 2\,x2(t) + 1\right] \quad \textbf{(5.25)}$$

```
> sol_stab_ode_maple := dsolve ( convert( subs( eta=1, omega__n = 20, ex_baum) ,set) union convert(
  subs( t=0, ics), set) , numeric);
```

$$sol\_stab\_ode\_maple := \textbf{proc}(x\_rkf45)\ \ldots\ \textbf{end proc} \quad \textbf{(5.26)}$$

```
> odeplot(sol_stab_ode_maple,[[t,x1(t),color="Blue"],[t,x2(t),color="Green"],[t,x3(t),color="Red"]],
  t=0..10,legend=["x1","x2","x3"]);
  odeplot(sol_stab_ode_maple,[[t,alg1[1],color="Blue"],[t,alg2[1],color="Green"],[t,alg3[1],color=
  "Red"]], t=0..100,legend=["alg1","alg2","alg3"]);
```





Create the Euler Numerical Method

```
> EulerStep := proc(tk, h, x1k, x2k, x3k, ode)
    local x1k1, x2k1, x3k1 :
    x1k1 := eval(x1k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, x3(tk)=x3k, ode[1]) ) :
    x2k1 := eval(x2k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, x3(tk)=x3k, ode[2]) ) :
    x3k1 := eval(x3k + h*subs(t=tk, x1(tk)=x1k, x2(tk)=x2k, x3(tk)=x3k, ode[3]) ) :
    return < x1k1, x2k1, x3k1> :
  end proc;
```

$$EulerStep := \mathbf{proc}(tk, h, x1k, x2k, x3k, ode) \qquad \textbf{(5.27)}$$

$$\mathbf{local}\ x1k1, x2k1, x3k1;$$

$$x1k1 := eval(x1k + h*subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, x3(tk) = x3k, ode[1]));$$

$$x2k1 := eval(x2k + h*subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, x3(tk) = x3k, ode[2]));$$

$$x3k1 := eval(x3k + h*subs(t = tk, x1(tk) = x1k, x2(tk) = x2k, x3(tk) = x3k, ode[3]));$$

$$\mathbf{return}\ \ < x1k1, x2k1, x3k1 >$$

end proc

Use the Numerical Method: 100 steps with step size (h) at 0.1, therefore we will integrate the system for 10 seconds. As we can see the solution drifts away from the real one, the step size is too big, we should do 1000 steps with step size 0.01 for 10 seconds to have better results.

```
> h := 0.1;
  nsteps := 100;
```

$$h := 0.1$$
$$nsteps := 100 \quad \textbf{(5.28)}$$

```
> t_step := [seq( h*i, i=1..nsteps)];
```

$$t\_step := [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0, 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 6.0, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 7.0, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 8.0, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 9.0, 9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 9.7, 9.8, 9.9, 10.0] \quad \textbf{(5.29)}$$

```
> results := Matrix(nsteps,3);
```

$$results := \begin{bmatrix} 100\ x\ 3\ Matrix \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix} \quad \textbf{(5.30)}$$

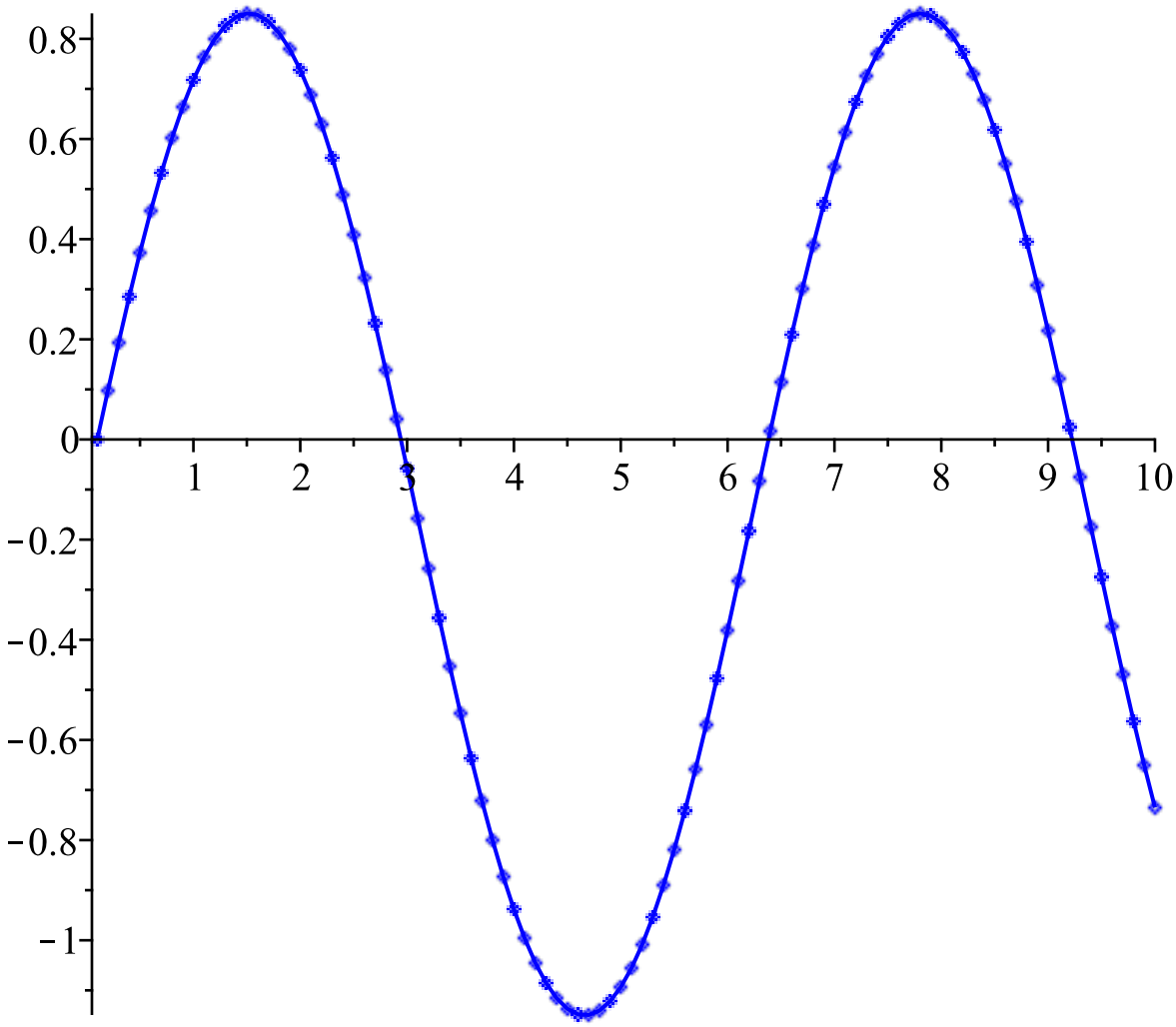
```
> results[1,1] := rhs(ics[1]);
  results[1,2] := rhs(ics[2]);
  results[1,3] := rhs(ics[3]);
  results[1,1..3];
```

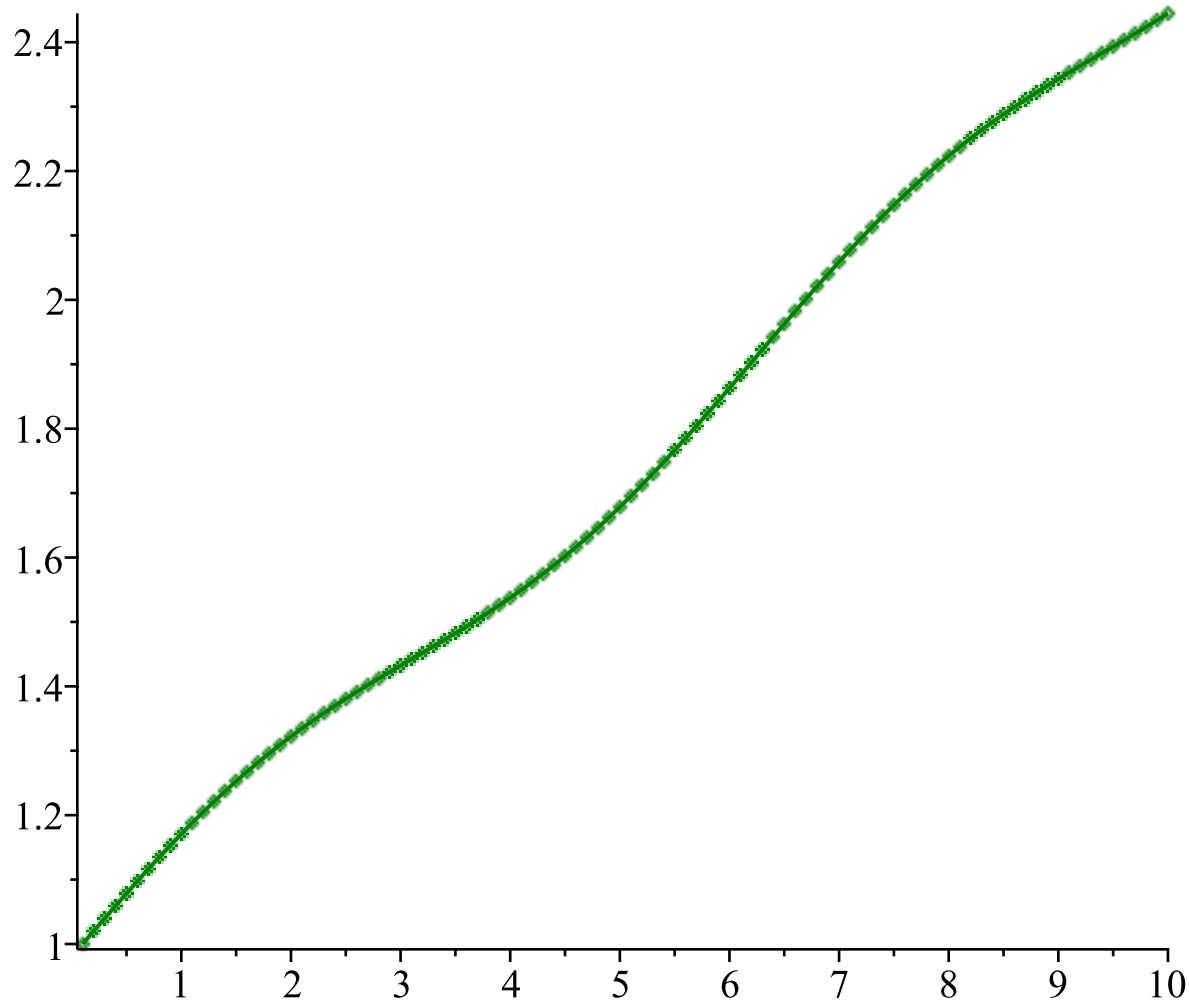
$$results_{1,1} := 0.$$
$$results_{1,2} := 1.$$
$$results_{1,3} := 2.$$
$$\begin{bmatrix} 0. & 1. & 2. \end{bmatrix} \quad \textbf{(5.31)}$$

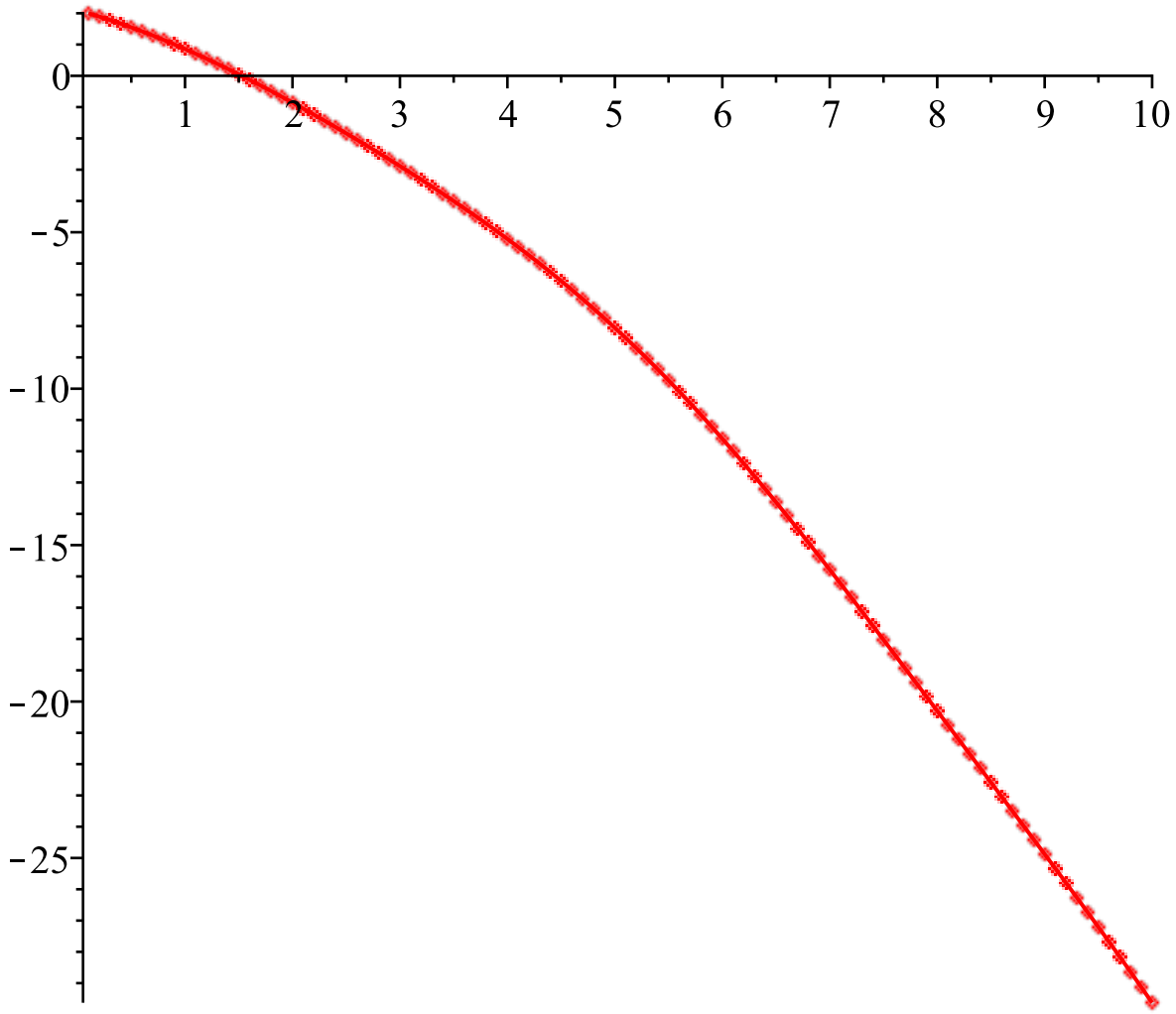
```
> rhs_ode := [ rhs( ex_ode[1]) , rhs( ex_ode[2]) ,rhs( ex_ode[3])]: <%>;
```

$$\begin{bmatrix} \cos(t) \\ \sin(t) - x1(t) \\ -t\sin(t) + t\,x1(t) - 2\,x2(t) + 1 \end{bmatrix} \quad \textbf{(5.32)}$$

```
> # Loop to perform the numerical method
> for i from 2 to nsteps do:
    tmp := EulerStep(t_step[i], h , results[i-1,1], results[i-1,2], results[i-1,3], rhs_ode);
    results[i,1] := tmp[1];
    results[i,2] := tmp[2];
    results[i,3] := tmp[3];
  end do:
> # plot the results:
  plots[pointplot](<<t_step[1..nsteps]>|<results[1..nsteps,1]>>,color="Blue",symbol=diamond,
  connect=true,style=pointline);
  plots[pointplot](<<t_step[1..nsteps]>|<results[1..nsteps,2]>>,color="Green",symbol=diamond,
  connect=true,style=pointline);
  plots[pointplot](<<t_step[1..nsteps]>|<results[1..nsteps,3]>>,color="Red",symbol=diamond,
  connect=true,style=pointline);
```


0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
2
3
4
5
6
7
8
9
10


2.4
2.2
2
1.8
1.6
1.4
1.2
1
1
2
3
4
5
6
7
8
9
10



Or to improve the solution we could create the projection step (that take as input the numerical current solution, the time and the constraints):

```
> with(Optimization):                    # remember to incluse the library
> subs(x1(t)=x1,x2(t)=x2,x3(t)=x3,algs);
  constr :=  [%[1] = 0, %[2] = 0, %[3] = 0] :<%>;
```

$$[-t\,x2 - x3 + 2,\ x2 - 1,\ \sin(t) - x1]$$

$$\begin{bmatrix} -t\,x2 - x3 + 2 = 0 \\ x2 - 1 = 0 \\ \sin(t) - x1 = 0 \end{bmatrix} \tag{5.33}$$

```
> ProjectionStep := proc(q0,tk,constr)
    local res,to_minimize;
    to_minimize := (x1-x10)^2+(x2-x20)^2+(x3-x30)^2;
    res := Minimize ( subs(x10=q0[1],x20=q0[2],x30=q0[3],to_minimize) , subs(t=tk,constr) );
    return <rhs(res[2,1]),rhs(res[2,2]),rhs(res[2,3])>;
  end proc;
```

$$\begin{aligned} &ProjectionStep := \mathbf{proc}(q0,\ tk,\ constr) \\ &\quad \mathbf{local}\ res,\ to\_minimize; \\ &\quad to\_minimize := (x1 - x10)\,\hat{}\,2 + (x2 - x20)\,\hat{}\,2 + (x3 - x30)\,\hat{}\,2; \\ &\quad res := Optimization{:}\text{-}Minimize(subs(x10 = q0[1],\ x20 = q0[2],\ x30 = q0[3],\ to\_minimize),\ subs(t = tk,\ constr)); \\ &\quad \mathbf{return}\ \ <rhs(res[2,\ 1]),\ rhs(res[2,\ 2]),\ rhs(res[2,\ 3])> \end{aligned} \tag{5.34}$$

end proc

Compute and plot the results: now the solution is very close to the real one, because the projection greatly limits the drift.

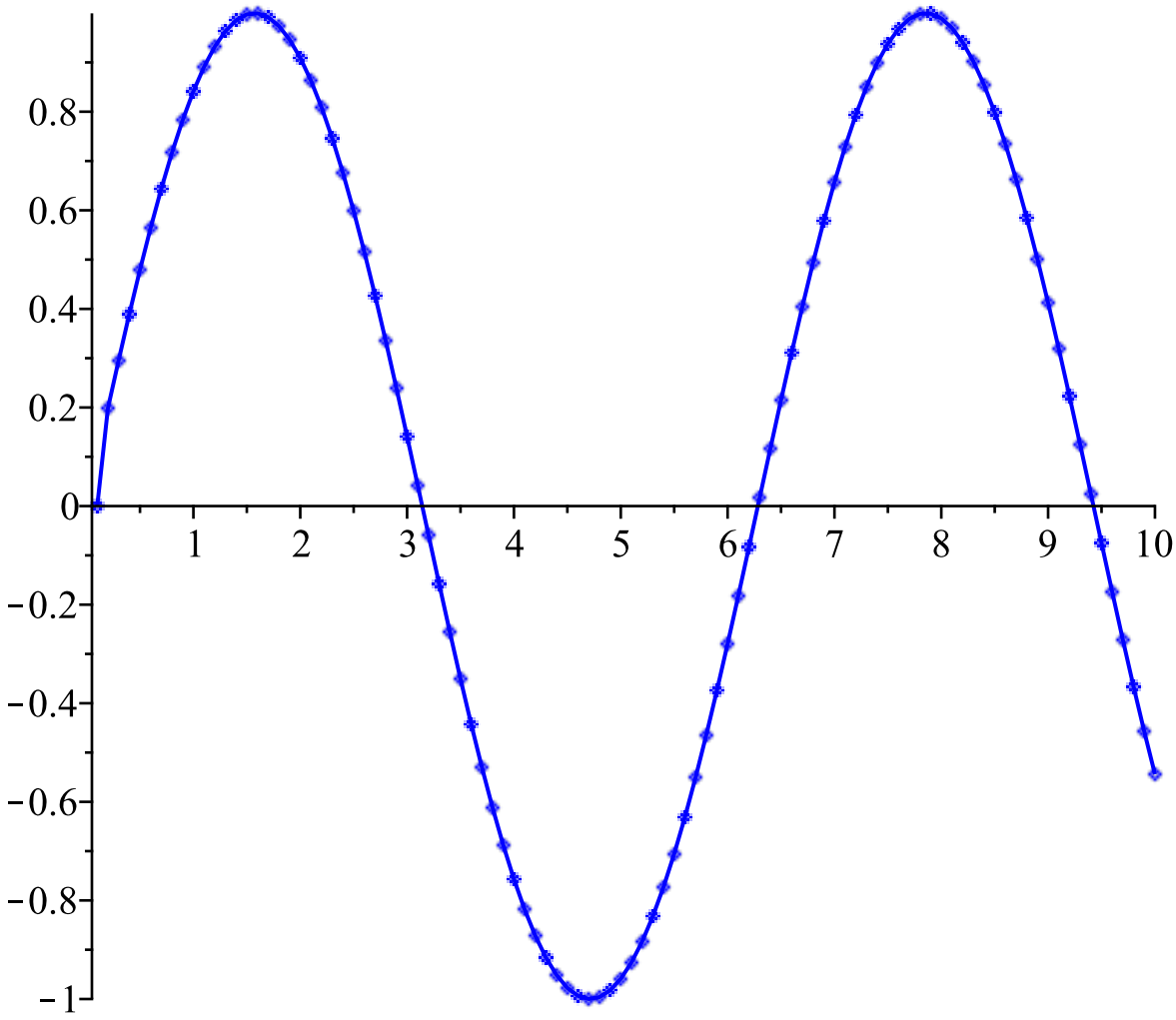
```
> results_proj := Matrix(nsteps,3);
```

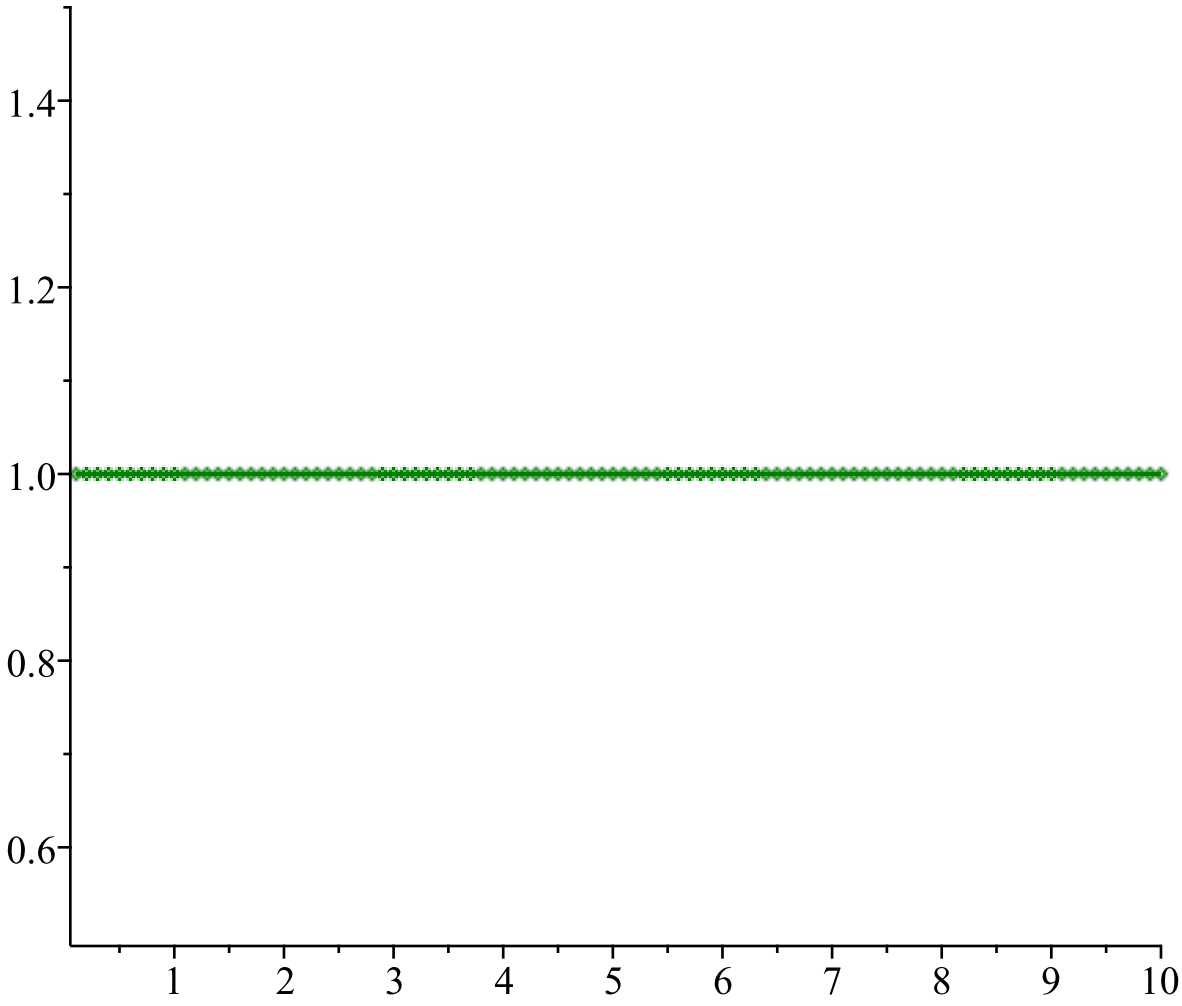
$$results\_proj := \begin{bmatrix} 100 \text{ x } 3 \text{ Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix} \tag{5.35}$$

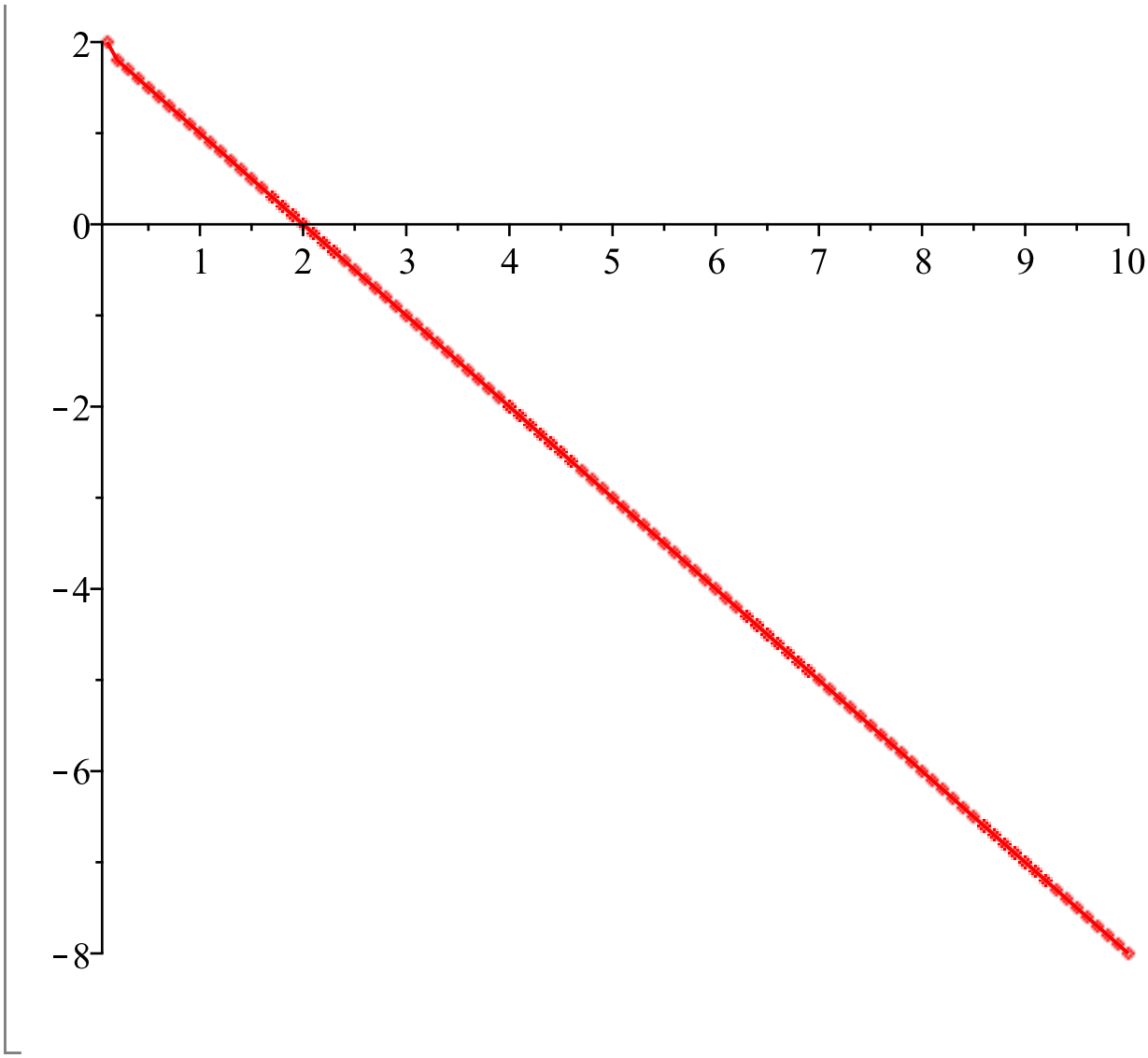
```
> results_proj[1,1] := rhs(ics[1]);
  results_proj[1,2] := rhs(ics[2]);
  results_proj[1,3] := rhs(ics[3]);
  results_proj[1,1..3];
```

$$results\_proj_{1,1} := 0.$$
$$results\_proj_{1,2} := 1.$$
$$results\_proj_{1,3} := 2.$$
$$\begin{bmatrix} 0. & 1. & 2. \end{bmatrix} \tag{5.36}$$

```
> for i from 2 to nsteps do:
    tmp1 := EulerStep(h,t_step[i],results_proj[i-1,1],results_proj[i-1,2],results_proj[i-1,3],rhs_ode);
    tmp2 := ProjectionStep(tmp1,t_step[i],constr);
    results_proj[i,1] := tmp2[1];
    results_proj[i,2] := tmp2[2];
    results_proj[i,3] := tmp2[3];
  end do:
> plots[pointplot](<<t_step[1..nsteps]>|<results_proj[1..nsteps,1]>>,color="Blue",symbol=diamond,
  connect=true,style=pointline);
  plots[pointplot](<<t_step[1..nsteps]>|<results_proj[1..nsteps,2]>>,color="Green",symbol=diamond,
  connect=true,style=pointline);
  plots[pointplot](<<t_step[1..nsteps]>|<results_proj[1..nsteps,3]>>,color="Red",symbol=diamond,
  connect=true,style=pointline);
```


0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
2
3
4
5
6
7
8
9
10


1.4
1.2
1.0
0.8
0.6
1
2
3
4
5
6
7
8
9
10


2
0
1
2
3
4
5
6
7
8
9
10
-2
-4
-6
-8