Differensekvationer

Jan Snellman

<2023-12-11 Mon>

Contents

1 (Diskreta) Trajektorier

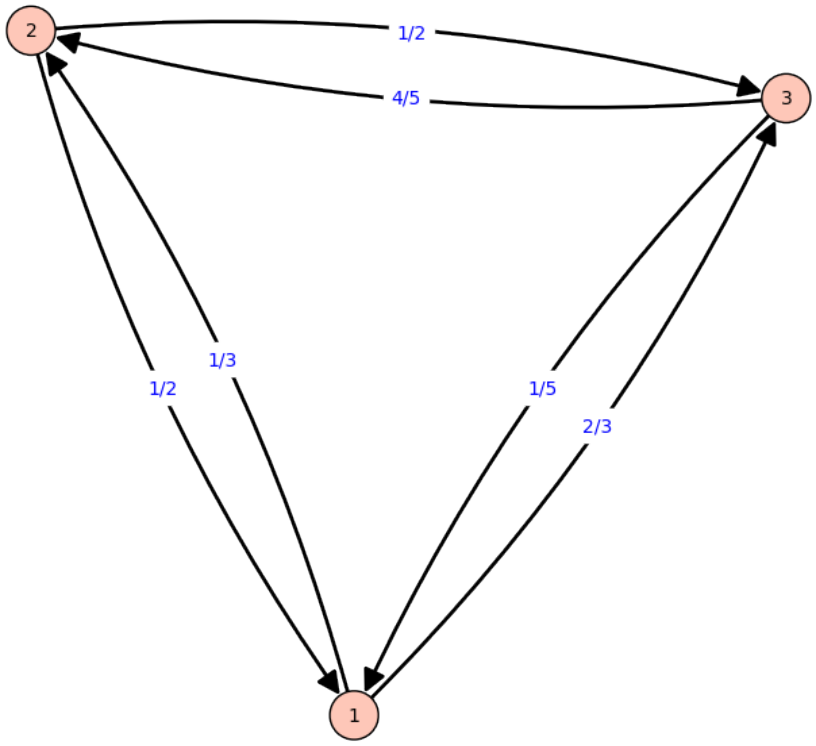
1.1 Exempel 1, enkel markovkedja

1.1.1 Riktad viktad graf

Varje tidsenhet så flyttar en viss andel av populationen från position 1 till position 2, eller blir kvar; likaså för position 2.

```
G = DiGraph({1: {1: 2/3, 2: 1/3}, 2: {1: 1/2, 2: 1/2}}, weighted=True)
```

```
import sage.graphs.graph_plot
sage.graphs.graph_plot.DEFAULT_PLOT_OPTIONS['figsize'] = 8
sage.graphs.graph_plot.DEFAULT_PLOT_OPTIONS['transparent'] = True
G.plot(edge_labels=True)
```



1.1.2 Linjär avbildning, uppdatera Xn

Vi uppdaterar fördelningen X_n vid tid n enligt $X_n = AX_{n-1}$.

```
<<Gmarkov1>>
A = G.weighted_adjacency_matrix().transpose().change_ring(QQbar)
X0 = matrix(QQbar,[[1/3,2/3]]).transpose()
```

```
<<AXdef>>
"adjacensmatris A:", A
"intialfördelning X:", X0
```

```
(
                     [2/3 1/2]
'adjacensmatris A:', [1/3 1/2]
)
(
                       [1/3]
'intialfördelning X:', [2/3]
)
```

1.1.3 Trajektorier, numeriskt

Trajektorierna X_n kommer att gå närma sig en stationär fördelning.

```
<<AXdef>>
[(k,numerical_approx((A**k*X0).transpose())) for k in range(N)]
```

```
0  [0.333333333333333 0.666666666666667]
1  [0.555555555555556 0.444444444444444]
2  [0.592592592592593 0.407407407407407]
3  [0.598765432098765 0.401234567901235]
4  [0.599794238683128 0.400205761316872]
5  [0.599965706447188 0.400034293552812]
6  [0.599994284407865 0.400005715592135]
7  [0.599999047401311 0.400000952598689]
8  [0.599999841233552 0.400000158766448]
9  [0.599999973538925 0.400000026461075]
```

1.1.4 Exakt lösning via diagonalisering

Vi kan analysera exakt vad som sker genom att beräkna egenvärden och egenvektorer till A.

```
<<AXdef>>
eigve = A.eigenvectors_right()
la = [_[0] for _ in eigve ]
f =  [_[1][0] for _ in eigve ]
B = diagonal_matrix(la)
T = matrix(f).transpose()
Y0 = T.inverse()*X0
```

```
<<eigvedef>>
"Karakteristiskt polynom: ", A.charpoly()
"Egenvärden lambda: ", la
"Egenvektorer f: ", f
"Matris B: ", B
"Basbytesmatris T: ", T
"Intitialvektor X0: ", X0
"Initialvektor Y0=T^(-1)X0 : ", Y0
```

```
('Karakteristiskt polynom: ', x^2 - 7/6*x + 1/6)
('Egenvärden lambda: ', [1, 1/6])
('Egenvektorer f: ', [(1, 2/3), (1, -1)])
(
              [  1   0]
'Matris B: ', [  0 1/6]
)
(
                     [  1   1]
'Basbytesmatris T: ', [2/3  -1]
)
(
                      [1/3]
'Intitialvektor X0: ', [2/3]
)
(
                                [  3/5]
'Initialvektor Y0=T^(-1)X0 : ', [-4/15]
)
```

1.1.5 Tabulering via exakt formel

Bidraget från $\lambda_1^n f_1$ kommer att dominera när vi sätter

$$X_n = A^n X_0 = (TBT^{-1})^n X0 = TB^n T^{-1} X0 = TB^n Y_0 = (\lambda_1^n f_1, \lambda_2^2 f_2) Y_0$$

```
<<eigvedef>>
[(k,
  (la[0]^k*f[0]*Y0[0,0]).n(digits=4),
  (la[1]^k*f[1]*Y0[1,0]).n(digits=4),
  (la[0]^k*f[0]*Y0[0,0] + la[1]^k*f[1]*Y0[1,0]).n(digits=4)
  ) for k in range(N)]
```

0	(0.6000, 0.4000)	(-0.2667, 0.2667)	(0.3333, 0.6667)
1	(0.6000, 0.4000)	(-0.04444, 0.04444)	(0.5556, 0.4444)
2	(0.6000, 0.4000)	(-0.007407, 0.007407)	(0.5926, 0.4074)
3	(0.6000, 0.4000)	(-0.001235, 0.001235)	(0.5988, 0.4012)
4	(0.6000, 0.4000)	(-0.0002058, 0.0002058)	(0.5998, 0.4002)
5	(0.6000, 0.4000)	(-0.00003429, 0.00003429)	(0.6000, 0.4000)
6	(0.6000, 0.4000)	(-5.716e-6, 5.716e-6)	(0.6000, 0.4000)
7	(0.6000, 0.4000)	(-9.526e-7, 9.526e-7)	(0.6000, 0.4000)

1.1.6 Problemet uttryckt i Y-koordinater

Vi kan också titta på skeendet i Y-koordinater:

```
<<eigvedef>>
g = [vector([1,0]),vector([0,1])]
[(k,
  (la[0]^k*g[0]*Y0[0,0]).n(digits=4),
  (la[1]^k*g[1]*Y0[1,0]).n(digits=4),
  (la[0]^k*g[0]*Y0[0,0] + la[1]^k*g[1]*Y0[1,0]).n(digits=4)
  ) for k in range(N)]
```

0	(0.6000, 0.0000)	(0.0000, -0.2667)	(0.6000, -0.2667)
1	(0.6000, 0.0000)	(0.0000, -0.04444)	(0.6000, -0.04444)
2	(0.6000, 0.0000)	(0.0000, -0.007407)	(0.6000, -0.007407)
3	(0.6000, 0.0000)	(0.0000, -0.001235)	(0.6000, -0.001235)
4	(0.6000, 0.0000)	(0.0000, -0.0002058)	(0.6000, -0.0002058)
5	(0.6000, 0.0000)	(0.0000, -0.00003429)	(0.6000, -0.00003429)
6	(0.6000, 0.0000)	(0.0000, -5.716e-6)	(0.6000, -5.716e-6)
7	(0.6000, 0.0000)	(0.0000, -9.526e-7)	(0.6000, -9.526e-7)

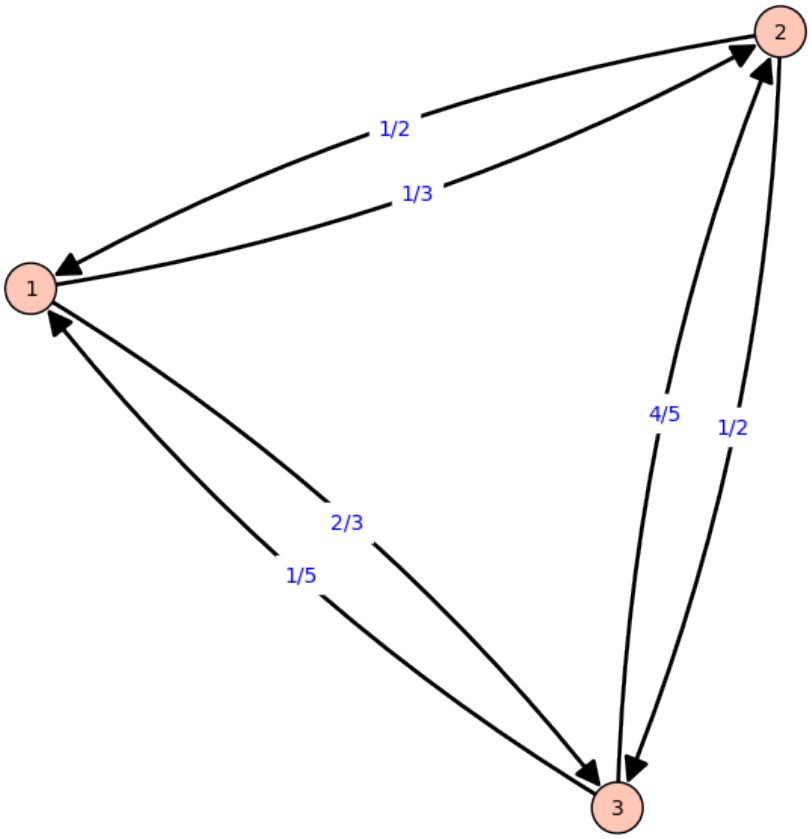
1.2 Exempel 2: markovkedja med komplexa egenvärden

1.2.1 Riktad viktad graf

Varje tidsenhet så flyttar en viss andel av populationen mellan position 1, 2, och 3.

```
G = DiGraph({1: {2: 1/3, 3: 2/3},
             2: {1: 1/2, 3: 1/2},
             3: {1: 1/5, 2:4/5}}, weighted=True)

G.plot(edge_labels=True)
```



1.2.2 Linjärt ekvationssystem

Vi uppdaterar fördelningen X_n vid tid n enligt $X_n = AX_{n-1}$.

```
<<Gmarkov2>>
A = G.weighted_adjacency_matrix().transpose().change_ring(QQbar)
X0 = matrix(QQbar,[[1/3,1/3,1/3]]).transpose()

<<AXdef2>>
```

```
"adjacensmatris A:", A
"intialfördelning X:", X0

(
                     [  0 1/2 1/5]
                     [1/3   0 4/5]
'adjacensmatris A:', [2/3 1/2   0]
)
(
                       [1/3]
                       [1/3]
'intialfördelning X:', [1/3]
)
```

1.2.3 Numerisk plot av trajektorier

Trajektorierna X_n kommer att gå närma sig en stationär fördelning.

```
<<AXdef2>>
[(k,numerical_approx((A**k*X0).transpose())) for k in range(N)]
```

0	[0.333333333333333 0.333333333333333 0.333333333333333]
1	[0.233333333333333 0.377777777777778 0.388888888888889]
2	[0.266666666666667 0.388888888888889 0.344444444444444]
3	[0.263333333333333 0.364444444444444 0.372222222222222]
4	[0.256666666666667 0.385555555555556 0.357777777777778]
5	[0.264333333333333 0.371777777777778 0.363888888888889]
6	[0.258666666666667 0.379222222222222 0.362111111111111]
7	[0.262033333333333 0.375911111111111 0.362055555555556]
8	[0.260366666666667 0.376988888888889 0.362644444444444]
9	[0.261023333333333 0.376904444444444 0.362072222222222]

1.2.4 Exakt formel via diagonalisering

Vi kan analysera exakt vad som sker genom att beräkna egenvärden och egenvektorer till A.

```
<<AXdef2>>
eigve = A.eigenvectors_right()
la = [_[0] for _ in eigve ]
f =  [_[1][0] for _ in eigve ]
```

```
B = diagonal_matrix(la)
T = matrix(f).transpose()
Y0 = T.inverse()*X0
```

```
<<eigvedef2>>
"Karakteristiskt polynom: ", A.charpoly()
"Egenvärden lambda: ", [_.n(digits=3) for _ in la]
"Egenvektorer f: ", [_.n(digits=3) for _ in f]
"Matris B: ", B.n(digits=2)
"Basbytesmatris T: ", T.n(digits=2)
"Intitialvektor X0: ", X0
"Initialvektor Y0=T^(-1)X0 : ", Y0.n(digits=3)
```

```
('Karakteristiskt polynom: ', x^3 - 7/10*x - 3/10)
('Egenvärden lambda: ', [1.00, -0.500 + 0.224*I, -0.500 - 0.224*I])
('Egenvektorer f: ',
 [(1.00, 1.44, 1.39),
  (1.00, -1.00 + 0.745*I, -8.50e-58 - 0.745*I),
  (1.00, -1.00 - 0.745*I, -8.50e-58 + 0.745*I)])
(
              [          1.0           0.00           0.00]
              [         0.00 -0.50 + 0.22*I           0.00]
'Matris B: ', [         0.00           0.00 -0.50 - 0.22*I]
)
(
'Basbytesmatris T: ',

[              1.0               1.0               1.0]
[              1.4     -1.0 + 0.75*I     -1.0 - 0.75*I]
[              1.4 -8.5e-58 - 0.75*I -8.5e-58 + 0.75*I]
)
(
                      [1/3]
                      [1/3]
'Intitialvektor X0: ', [1/3]
)
(
                                [           0.261]
                                [0.0362 - 0.0194*I]
'Initialvektor Y0=T^(-1)X0 : ', [0.0362 + 0.0194*I]
)
```

1.2.5 Tabulering via exakt formel

Bidraget från $\lambda_1^n f_1$ kommer att dominera när vi sätter

$$X_n = A^n X_0 = (TBT^{-1})^n X0 = TB^n T^{-1} X0 = TB^n Y_0 = (\lambda_1^n f_1, \lambda_2^2 f_2) Y_0$$

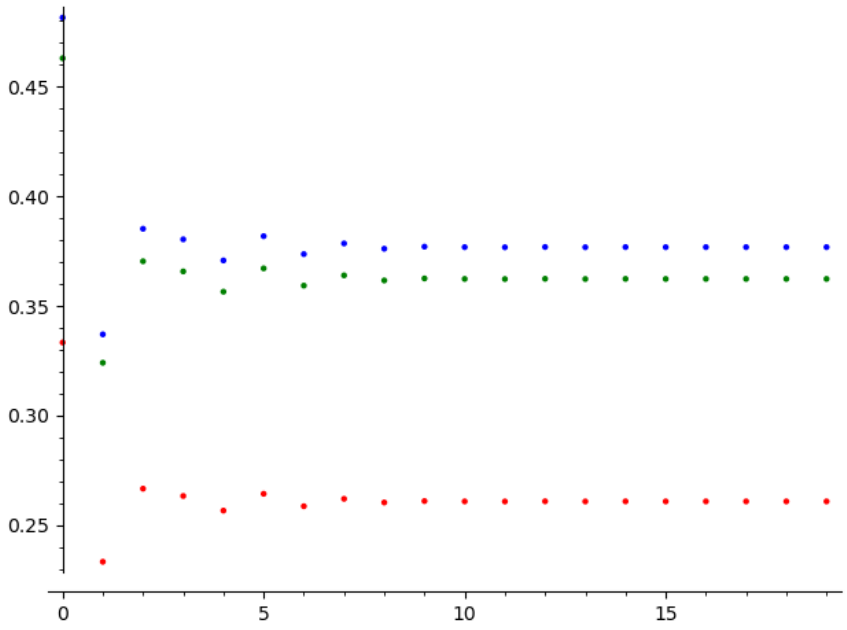
```
<<eigvedef2>>
[(k,
  (la[0]^k*f[0]*Y0[0,0]).n(digits=3),
  add([(la[j]^k*f[0]*Y0[j,0]).n(digits=3)
       for j in range(3)]).n(digits=3)
  ) for k in range(N)]

[(0,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.333, 0.482, 0.463)),
 (1,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.233, 0.337, 0.324)),
 (2,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.267, 0.385, 0.370)),
 (3,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.263, 0.380, 0.366)),
 (4,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.257, 0.371, 0.356)),
 (5,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.264, 0.382, 0.367)),
 (6,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.259, 0.374, 0.359)),
 (7,
  (0.261 - 2.54e-21*I, 0.377 - 3.39e-21*I, 0.362 - 3.39e-21*I),
  (0.262, 0.379, 0.364))]
```

1.2.6 Informativ plot

Går det mot något? Vi plottar!

```
<<eigvedef2>>
klista = [add([(la[j]^k*f[0]*Y0[j,0]) for j in range(3)]) for k in range(N)]
P0 = list_plot([_[0] for _ in klista], color = 'red')
P1 = list_plot([_[1] for _ in klista], color = 'blue')
P2 = list_plot([_[2] for _ in klista], color = 'green')
P0 + P1 + P2
```



2 Fibonacci

2.1 Definition

Fibonacciföljden definieras av rekursionen

$$\forall n \geq 0 : a_{n+2} = a_{n+1} + a_n$$

samt begynnelsevärden för a_0, a_1. Dessa är $a_0 = a_1 = 1$.

2.1.1 Dum pythonimplementering

```
def fib(k):
    if k < 2:
        return 1
```

```
    else:
        return fib(k-1) + fib(k-2)

<<nomemofib>>
[(k,fib(k)) for k in range(15)]
```

0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89
11	144
12	233
13	377
14	610

2.2 Matrisformulering

Vi inför $b_n = a_{n+1}$ och sätter $X_n = (a_n, b_n)^t$, då gäller

$$X_{n+1} = \begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} b_n \\ a_{n+2} \end{pmatrix} = \begin{pmatrix} b_n \\ a_{n+1} + a_n \end{pmatrix} = \begin{pmatrix} b_n \\ b_n + a_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} X_n$$

```
A = matrix(QQbar,[[0,1],[1,1]])
bas1 = matrix([[1,0]]).transpose()
bas2 = matrix([[0,1]]).transpose()
```

Genom att läsa av första koordinaten borde vi få samma följd:

```
<<fib>>
A
bas1,bas2
[A**k*(bas1+bas2) for k in range(10)]

[0 1]
[1 1]
```

```
(
[1]  [0]
[0], [1]
)
[
[1]  [1]  [2]  [3]  [5]  [ 8]  [13]  [21]  [34]  [55]
[1], [2], [3], [5], [8], [13], [21], [34], [55], [89]
]
```

2.3 Analys med egenvärden, egenvektorer

```
<<fib>>
"karpoly A: ", A.charpoly()
eig= A.eigenvalues()
l1 = eig[0]
l2 = eig[1]
"eigenvalues: ", l1,l2
eiv =A.eigenvectors_right()
f1 = eiv[0][1][0]
f2 = eiv[1][1][0]
"eigenvectors: ",f1,f2
T = matrix([f1,f2]).transpose()
Y0 =T.inverse()*bas1
"basyting T: ", T
"Initialvillkor i ny bas Y0: ",  Y0
"Fib sequence from bas1: "
[Y0[0,0]*l1^k*f1 + Y0[1,0]*l2^k*f2 for k in range(10)]

('karpoly A: ', x^2 - x - 1)
('eigenvalues: ', 1.618033988749895?, -0.618033988749895?)
('eigenvectors: ', (1, 1.618033988749895?), (1, -0.618033988749895?))
(
                [                   1                    1]
'basyting T: ', [ 1.618033988749895? -0.618033988749895?]
)
(
                                  [0.276393202250021?]
'Initialvillkor i ny bas Y0: ', [0.723606797749979?]
)
'Fib sequence from bas1: '
```

```
[(1.0000000000000000?, 0.?e-18),
 (0.?e-18, 1.0000000000000000?),
 (1.0000000000000000?, 1.0000000000000000?),
 (1.0000000000000000?, 2.0000000000000000?),
 (2.0000000000000000?, 3.0000000000000000?),
 (3.0000000000000000?, 5.0000000000000000?),
 (5.0000000000000000?, 8.00000000000000?),
 (8.00000000000000?, 13.00000000000000?),
 (13.00000000000000?, 21.00000000000000?),
 (21.00000000000000?, 34.00000000000000?)]
```

Vi kan också ansätta $a_n = c_1\lambda_1^n f_1 + c_2\lambda_2^n f_2$ och bestämma c_1, c_2 från begynnelsevärdena. För stora n så är a_n ungefär $c_1\lambda_1^n$.

3 System av linjära differentialekvationer, kontinuerliga trajektorier.

- $X'(t) = AX(t)$
- Byt till egenvektorbas
- $Y'(t) = BY(t)$
- B diagonal
- Kan lösa detta system
- Byt tillbaka via $X = AT$
- Eller ansätt $X = \sum_j c_j f_j e^{\lambda_j t}$

3.1 Exempel, system av två första ordningens ode

Vi löser

$$X(t)' = AX(t), \qquad A(0) = X_0$$

med

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \qquad X_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

3.1.1 Matrisformulering, diagonalisering, lösning, allt i ett svep

```
var('t,c1,c2')
A = matrix([[2,-1],[-1,2]])
X0 = matrix([[1,0]]).transpose()
"A: ",A
"X0: ",X0
"karpol: ", A.charpoly(), " = ", A.charpoly().factor()
eigve = A.eigenvectors_right()
la = [_[0] for _ in eigve ]
f =  [_[1][0] for _ in eigve ]
B = diagonal_matrix(la)
T = matrix(f).transpose()
Y0 = T.inverse()*X0
"lambdas: ", la
"f: ", f
"B: ",B
"T: ",T
"Y0: ",Y0
X = c1*f[0]*exp(la[0]*t) + c2*f[1]*exp(la[1]*t)
"X(t): ", X
X0t = X.substitute(t==0)
"Begynnelsevärdena ger: ", X0t, " == ", X0
soln = solve([X0t[0] == X0[0,0], X0t[1] == X0[1,0]],[c1,c2])
"lösning :", soln
"Så X blir: ", X.substitute(soln)
```

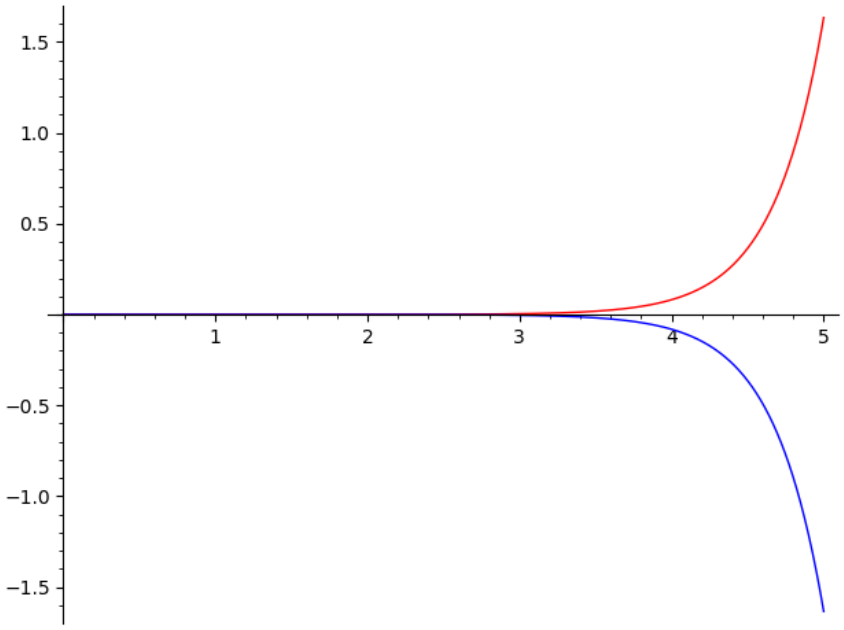
```
(t, c1, c2)
(
       [ 2 -1]
'A: ', [-1  2]
)
(
        [1]
'X0: ', [0]
)
('karpol: ', x^2 - 4*x + 3, ' = ', (x - 3) * (x - 1))
('lambdas: ', [3, 1])
('f: ', [(1, -1), (1, 1)])
(
       [3 0]
```

```
'B: ', [0 1]
)
(
       [ 1  1]
'T: ', [-1  1]
)
(
        [1/2]
'Y0: ', [1/2]
)
('X(t): ', (c1*e^(3*t) + c2*e^t, -c1*e^(3*t) + c2*e^t))
(
                                                              [1]
'Begynnelsevärdena ger: ', (c1 + c2, -c1 + c2), ' == ', [0]
)
('lösning :', [[c1 == (1/2), c2 == (1/2)]])
('Så X blir: ', (1/2*e^(3*t) + 1/2*e^t, -1/2*e^(3*t) + 1/2*e^t))
```

3.1.2 Plot av lösning

$x_t(t)$ i rött, $x_2(t)$ i blått:

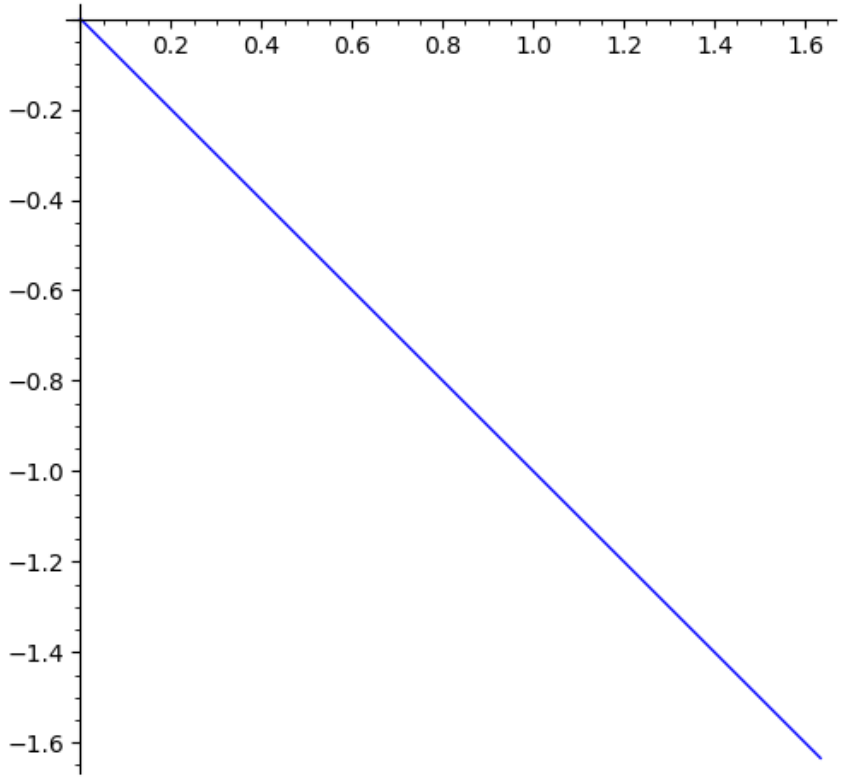
```
<<enkelODE>>
X1p = plot(X.substitute(soln)[0],(t,0,5), color = 'red')
X2p = plot(X.substitute(soln)[1],(t,0,5), color = 'blue')
X1p + X2p
```



3.1.3 xy-plot

Vi plottar punkterna $(x_1(t), x_2(t))$ för $-t \leq t \leq 5$:

```
<<enkelODE>>
parametric_plot(X.substitute(soln),(t,-5,5))
```


0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
−1.6