```
> # This example is example 4 in Chapter 5 of the book Modelling population dynamics:  model
      formulation, fitting and assessment  using state−space methods
  # Press enter to activate each line of code (or use !!! button above to active all the code)
> restart;
> with(LinearAlgebra) :
> Dmat :=proc(kappa, pars)
  local DD1, i, j;
  description "This proceedure form the derivative matrix given a vector of exhaustive summary
      terms, kappa, and a vector of parameters, pars";
  with(LinearAlgebra) :
  DD1 := Matrix(1 ..Dimension(pars), 1 ..Dimension(kappa)) :
  for i from 1 to Dimension(pars)  do
      for j from 1 to Dimension(kappa) do
          DD1[i, j] := diff(kappa[j], pars[i])
      end do
   end do;
  DD1;
   end proc:
> Estpars :=proc(DD1, pars)
  local r, d, alphapre, alpha, PDE, FF, i, ans;
  description "Finds the estimable set of parameters for derivative matrix DD1 and vector of
      parameters pars";
  with(LinearAlgebra) :
  r := Rank(DD1);
  d := Dimension(pars) - r :
  alphapre := NullSpace(Transpose(DD1)) :
  α := Matrix(d, Dimension(pars)) : PDE := Vector(d) :
  FF := f(seq(pars[i], i = 1 ..Dimension(pars))) :
  for i from 1 to d do
        α[i, 1 ..Dimension(pars)] := alphapre[i] :
        PDE[i] := add(diff(FF, pars[j]) · α[i, j], j = 1 ..Dimension(pars)) :
   end do:
   ans := pdsolve({seq(PDE[i] = 0, i = 1 ..d)});
  end proc:
> Expan :=proc(A, C, x0, n)
   local i, x, y, kappa, tt;
  description "Finds the exhaustive summary for the expansion method with n terms";
   y := eval(Multiply(C, x0), t = 0);
   x := eval(Multiply(A, x0), t = 1);
   tt := 1 :
   kappa := ⟨ ⟩ :
    for i from 1 to n do
         y := Multiply(eval(C, t = tt), x);
          tt := tt + 1 :
         x := Multiply(eval(A, t = tt), x);
         kappa := ⟨kappa, y⟩;
   end do:
   kappa := convert(kappa, Vector)
   end proc:
> Formnum :=proc(D1)
```

```
local results, j, numpars, D1rand :

    description  "This procedure finds the rank and alpha for the hybrid-symbolic-numeric
    method";
results := Matrix(5, 2) :
for j from 1 to 5 do
    numpars := seq(indets(D1)[i] = evalf(rand( )/1000000000000), i = 1 ..nops(indets(D1))) :
    D1rand := eval(D1, {numpars});
    results[j, 1] := Rank( D1rand);
   results[j, 2] := NullSpace(Transpose(D1rand)) :
end do:
results :
end proc:
```

```
> PLUR :=proc(D1)
local pp, ll, u1, r1, DetU;
description  "This procedure finds a PLUR or turing factorisation of the matrix D1." :
(pp, ll, u1, r1) := LUDecomposition( D1, output = ['P','L','U1','R'] ) :
DetU := Determinant(u1);
⟨DetU = 0, {P = pp, L = ll, U = u1, R = r1}⟩ :
end proc:
```

```
> #The observation and transition matrices:
```

```
> OO := ⟨⟨1|0|0|0⟩, ⟨0|1|0|0⟩, ⟨0|0|1|0⟩, ⟨0|0|0|1⟩⟩; L := ⟨⟨λ·(1 − μ12)·ϕ1|λ·(1 − μ12)·ϕ2|λ
    ·μ21·ϕ3|λ·μ21·ϕ4⟩, ⟨(1 − μ12)·ϕ1|(1 − μ12)·ϕ2|μ21·ϕ3|μ21·ϕ4⟩, ⟨λ·μ12·ϕ1|λ·ϕ2|λ·(1 − μ21)
    ·ϕ3|λ·(1 − μ21·ϕ4)⟩, ⟨μ12·ϕ1|μ12·ϕ2|(1 − μ21)·ϕ3|(1 − μ21)·ϕ4⟩⟩;
```

$$OO := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L := \begin{bmatrix} \lambda\left(1-\mu_{12}\right)\phi_1 & \lambda\left(1-\mu_{12}\right)\phi_2 & \lambda\,\mu_{21}\,\phi_3 & \lambda\,\mu_{21}\,\phi_4 \\ \left(1-\mu_{12}\right)\phi_1 & \left(1-\mu_{12}\right)\phi_2 & \mu_{21}\,\phi_3 & \mu_{21}\,\phi_4 \\ \lambda\,\mu_{12}\,\phi_1 & \lambda\,\phi_2 & \lambda\left(1-\mu_{21}\right)\phi_3 & \lambda\left(1-\mu_{21}\,\phi_4\right) \\ \mu_{12}\,\phi_1 & \mu_{12}\,\phi_2 & \left(1-\mu_{21}\right)\phi_3 & \left(1-\mu_{21}\right)\phi_4 \end{bmatrix} \quad \textbf{(1)}$$

```
> #The values of the state-equations at time zero are assumed to be the known constants:
> ints := ⟨n0, 1, n0, 2, n0, 3, n0, 4⟩ :
>
    #The exhaustive summary is found using the proceedure Expan. In this case we include
    terms E(y0),..., E(y2)
> kappa := (Expan(L, OO, ints, 2)) :
> pars := ⟨λ, μ12, μ21, ϕ1, ϕ2, ϕ3, ϕ4⟩ :
> D1 := Dmat(kappa, pars) :
```

```
> r := Rank(D1); d := Dimension(pars) − r;
```

$$r := 7$$

$$d := 0 \tag{2}$$

```
> #The model is therefore full rank so in theory all the parameters can be estimated.
>
```