HW5 - Do Not Post

0) $6a(n-1) + a(n+3) + 5a(n+1) = 0$

*$n = n-3$

$6a(n-3-1) + a(n-3+3) + 5a(n-3+1) = 0$

$6a(n-4) + a(n) + 5a(n-2) = 0$

$a(n) + 0a(n-1) + 5a(n-2) + 0a(n-3) + 6a(n-4) = 0$

→ rest done on Maple ←

```
> #Do Not post homework
  #Nikita John, September 20th, 2021, Assignment 5
> #Inputting M5 code into Worksheet
  with(LinearAlgebra) :

  Help5 :=proc( ) : print( `RecToSeq(INI,REC,N), GrowthC(INI,REC,K) , GrowthCe(REC)`) :
  print( ` LeslieMod(SUR,FER): e.g. LeslieMod([9/10,9/10],[0,1,1]);`) :
  print( ` LeslieMat(SUR,FER); e.g. LeslieMat([9/10,9/10],[0,1,1]);`) :
  end:




     #RecToSeq(INI,REC,N): Inputs two lists of numbers, INI and REC (of the same length, let's
     call it k) and a positive integer N larger than their length

     #outputs the list of the first N members of the sequence satisfying the linear recurrence with
     constants coefficients or order k
  #f(n)=REC[1]*f(n-1)+... +REC[k]*f(n-k)

  RecToSeq :=proc(INI, REC, N ) local i, k, L, newguy :
  if not (type(INI, list) and type(REC, list) and nops(INI) = nops(REC) and type(N, integer)
     and N ≥ nops(INI) ) then
   print( `bad innput`) :
   RETURN(FAIL) :
  fi:

  k := nops(INI) :

  L := INI :

  while nops(L) < N do
   newguy := add(REC[i] * L[-i], i = 1 ..k) :
   L := [op(L), newguy] :
  od:
  L :
  end:




     #GrowthC(INI,REC,K): The estimate of the growth constant of the sequence given by the
     inital conditions and recurrence pair [INI,REC] using K terms
  GrowthC :=proc(INI, REC, K) local L, a, b :
  L := RecToSeq(INI, REC, K) :

  a := L[-1]/L[-2] :
  b := L[-2]/L[-3] :

  if abs(a-b) < 1/10^(Digits + 3) then
  RETURN(evalf(a)) :
```

```
else
print(`make`, K, `bigger`) :
 RETURN(FAIL) :
fi:
end:



    #GrowthCe(REC): The  EXACT  growth constant of the recurrence REC using the
    characteristic equation
GrowthCe :=proc(REC) local x, i :
evalf ( [solve(1-add(REC[i]/x^i, i=1 ..nops(REC)) ) ]) [1] :

end:


#LeslieMod(SUR,FER):In a population with A age-groups and survival vector
#(following the notation in the book DMB, p.33

    #SUR=[p[0], ...p[A-1]] where p[i] is the probability of somebody of age i will still be alive
    the next year and

    #FER=[f[0],..., f[A]] where f[i] is the fertility factor (of course f[0]=0 in real life, and also in
    real life the
#younger ages can't have babies, so SUR has A components and FER has A+1 components
#Outputs the recurrence vector REC that enables the computation of the population growth. TRY:
#LelieMod([9/10,9/10],[0,1/2,1]]);
LeslieMod :=proc(SUR, FER) local i, L, A :

 if not (type(SUR, list) and type(FER, list) and nops(SUR) + 1 = nops(FER)) then
  print(`bad input`) :
 RETURN(FAIL) :
 fi:

A := nops(SUR) :

L[0] := 1 :

for i from 1 to A do
 L[i] := L[i-1] * SUR[i] :
od:

[seq(FER[i + 1] * L[i], i=0 ..A) ] :

end:
```

```
#LeslieMat(SUR,FER):In a population with A age-groups and survival vector
#(following the notation in the book DMB, p.33

    #SUR=[p[0], ...p[A-1]] where p[i] is the probability of somebody of age i will still be alive
    the next year and

    #FER=[f[0],..., f[A]] where f[i] is the fertility factor (of course f[0]=0 in real life, and also in
    real life the
#younger ages can't have babies, so SUR has A components and FER has A+1 components
#Outputs the Leslie A+1 by A+1 Leslie Matrix (DMB, p. 36, Eq. (2.17))
#LelieMat([9/10,9/10],[0,1/2,1]]);
LeslieMat := proc(SUR, FER) local i, A :

 if not (type(SUR, list) and type(FER, list) and nops(SUR) + 1 = nops(FER)) then
  print(`bad input`) :
 RETURN(FAIL) :
 fi:

A := nops(SUR) :

matrix([ FER, seq([0$(i-1), SUR[i], 0$(A + 1 - i)], i = 1 ..A)]) :
end:
```

```
> #0:
 L := RecToSeq([1, 2, 4, 11], [0, 5, 0, 6], 1000) :
 L[1000];
```

3683996880106022149977149682587150505723356918030336591301750861694696588362536504205178533310767791007606587990608250593032959373621150406207204854512813328871185841783578276539845684169221860138216640003144359501996893741538233993921528818293111738166473743309967618857396002120164560584834479930853955023620658624045097495586148633886614690490296885793838695258529580052 0887274913960521 **(1)**

```
> #1 : First way - Use GrowthC
GrowthC([1, 2, 4, 5, 3, 2, 7, 0, 0, 9], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], 100);
```

1.999018633 **(2)**

```
> #1: Second Way - Use GrowthCe
 GrowthCe([1, 1, 1, 1, 1, 1, 1, 1, 1, 1]);
```

1.999018633 **(3)**

```
> #2: First Way - using LeslieMod and GrowthCe
 Fer := [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.25, 0.25, 0.25, 0.25,
    0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25] :
 Sur := [0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99,
    0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99] :
 LeslieMod(Sur, Fer);
```

[0.5, 0.495, 0.49005, 0.4851495, 0.480298005, 0.4754950250, 0.4707400747, 0.4660326740, 0.4613723472, 0.4567586238, 0.4521910375, 0.4476691271, 0.4431924358, 0.4387605115, 0.4343729064, 0.2150145887, 0.2128644428, 0.2107357984, 0.2086284404, 0.2065421560, 0.2044767344, 0.2024319671, 0.2004076474, 0.1984035710, 0.1964195352, 0.1944553399, 0.1925107865, 0.1905856786, **(4)**

```
0.1886798218, 0.1867930236]
> GrowthCe(%);
                                   1.489452967                                   (5)
> #2: Second Way - using LeslieMat and finding the largest eigenvalue (in absolute value) of the
      Leslie Matrix
  LeslieMat(Sur, Fer) :
> evalf(Eigenvectors(%));
      #The largest Eigenvector is the one at the top of the matrix on the left, 1.48945296740263,
      and matches the previous result!
```

$$\left[\begin{array}{c} 1.48945296740263 + 0.\,I \\ 0.919290135969103 + 0.203023441233320\,I \\ 0.919290135969103 - 0.203023441233320\,I \\ -0.908341203138505 + 0.\,I \\ 0.878000388157044 + 0.418038228436395\,I \\ 0.878000388157044 - 0.418038228436395\,I \\ 0.745229302808563 + 0.560079548963723\,I \\ 0.745229302808563 - 0.560079548963723\,I \\ 0.618720877336058 + 0.727705464482224\,I \\ 0.618720877336058 - 0.727705464482224\,I \\ \vdots \end{array}\right], \left[\begin{array}{cc} 0.747133899825791 + 0.\,I & 0.0737617262045268 + 0.(\\ 0.496600145835682 + 0.\,I & 0.0760795350819318 - 0. \\ 0.330076984729944 + 0.\,I & 0.0746753017259807 - 0. \\ 0.219393443119249 + 0.\,I & 0.0692283964372843 - 0. \\ 0.145825019951331 + 0.\,I & 0.0595950738933778 - 0. \\ 0.0969260345317051 + 0.\,I & 0.0458346864168076 - 0. \\ 0.0644241720191553 + 0.\,I & 0.0282280801598856 - 0. \\ 0.0428210434936969 + 0.\,I & 0.00728638772175782 - (\\ 0.0284620152408615 + 0.\,I & -0.0162512575317153 - \\ 0.0189179488746058 + 0.\,I & -0.0414321385284767 - \\ \vdots & \vdots \end{array}\right]$$

30 element Vector[column]

```
> #3: Find the growth rate of salmon using the information in the textbook
  GrowthCe([0, 0, 0, 0.46, 0.41]);
                                   0.9693601961                                  (7)
> #4: Coding Plant Growth
  PlantGseq := proc(alpha, beta, gamma, sigma, INI, K) local i, k, L, newguy, REC :
  if not (type(INI, list) and nops(INI) = 2 and type(K, integer) and K ≥ nops(INI)) then
   print(`bad innput`) :
   RETURN(FAIL) :
  fi:

  k := nops(INI) :
  L := INI:
   REC := [alpha·sigma·gamma, beta·sigma·(1 − alpha)·sigma·gamma] :
  while nops(L) < K do
   newguy := add(REC[i]*L[-i], i = 1 ..k) :
   L := [op(L), newguy] :
  od:
  L :
   end:
> PlantGseq(0.5, 0.25, 2, 0.8, [100, 80], 20);
[100, 80, 80.00000, 76.8000000, 74.24000000, 71.68000000, 69.22240000, 66.84672000,   (8)
```

```
    64.55296000, 62.33784320, 60.19874816, 58.13305344, 56.13824246, 54.21188252,
    52.35162481, 50.55520105, 48.82042081, 47.14516882, 45.52740239, 43.96514892]
> PlantGseq(0.6, 0.3, 2, 0.8, [100, 96], 20);
[100, 96, 107.5200, 117.964800, 129.7612800, 142.6902221, 156.9139458, 172.5546061,        (9)
    189.7544040, 208.6686153, 229.4681472, 252.3409206, 277.4935912, 305.1534130,
    335.5702921, 369.0190446, 405.8018797, 446.2511298, 490.7322533, 539.6471367]
> #5: Calculate growth constant of plants
  PlantGSeq := proc(alpha, beta, gamma, sigma) local x, i, REC :
  REC := [alpha·sigma·gamma, beta·sigma·(1 − alpha)·sigma·gamma];
  evalf([solve(1-add(REC[i]/x^i, i=1..nops(REC)))])[1] :

  end:
> PlantGSeq(0.2, 0.2, 4, 0.3); #Extinction
                                0.3883281573                                              (10)
> PlantGSeq(0.5, 0.5, 5, 0.75); #Explosion
                                2.195288237                                               (11)
> PlantGSeq(0.485, 0.515, 2, 0.73); #Stability (or very close to stability)
                                1.007087066                                               (12)
>
```