```
> read `C:/Users/cgrie/Dynam Models Bio/Homeworks/HW15/M15.txt`
> Help15();
```

HW3(u,v,w), HW2(u,v) , Dis1(F,y,y0,h,A), ToSys(k,z,f,INI) **(1)**

```
> f:= x(n) = (x(n-1)+2*x(n-2)+3*x(n-3)+11*x(n-4))/(x(n-1)+x(n-3));
```

$$f := x(n) = \frac{x(n-1) + 2\,x(n-2) + 3\,x(n-3) + 11\,x(n-4)}{x(n-1) + x(n-3)} \quad (2)$$

```
> ic := {x(0)=1,x(1)=5,x(2)=5,x(3)=2}
```

$$ic := \{x(0) = 1, x(1) = 5, x(2) = 5, x(3) = 2\} \quad (3)$$

```
> print(ToSys);
```

proc(k, z, f, INI) **local** i; $[f, seq(z[i-1], i=2..k)]$ **end proc** **(4)**

```
> f_sys:= ToSys(2,x,f,ic);
```

$$f\_sys := \left[x(n) = \frac{x(n-1) + 2\,x(n-2) + 3\,x(n-3) + 11\,x(n-4)}{x(n-1) + x(n-3)}, x_1\right] \quad (5)$$

```
> SFP2drz(f_sys,x,x__1);
Error, (in FP2drz) invalid input: numer expects its 1st argument, x, to be of type {list, set, algebraic}, but received x(n)-x = (x(n-1)+2*x(n-2)+3*x(n-3)+11*x(n-4))/(x(n-1)+x(n-3))-x |C:/Users/cgrie/Dynam Models Bio/Homeworks/HW15/M15.txt:261|
```

Was the problem here because i dont have it in the right form (left hand should not be an equation)?

```
> print(SFP2);
```

(6)

```
proc(F, x, y)
    local L, J, S, J0, i, pt, EV;
    L := evalf(FP2(F, x, y));
    J := Matrix(normal([[diff(F[1], x), diff(F[1], y)], [diff(F[2], x), diff(F[2], y)]]));
    S := [ ];
    for i to nops(L) do
        pt := L[i];
        J0 := subs({x=pt[1], y=pt[2]}, J);
        EV := LinearAlgebra:-Eigenvalues(J0);
        if abs(EV[1]) < 1 and abs(EV[2]) < 1 then S := [op(S), pt] end if
    end do;
    S
end proc
```

```
> Help13();
```

RT2(x,y,d,K), Orb2(F,x,y,pt0,K1,K2), FP2(F,x,y), SFP2(F,x,y), PlotOrb2(L), FP2drz(F,x,y), SFP2drz(F,x,y) **(7)**

```
> #Problem 3
```

Use procedure OrbK to numerically find stable fixed point (if it exists) of the second order recurrence with given ICs

```
> ToSys(x(n
```
$x(n) = (1 - x(n-1)(1 - x(n-2)), x(0) = 2.5,\ x(1) = 2.7$

```
> f:= x(n)=(1-x(n-1))*(1-x(n-2));
```
$$f := x(n) = (1 - x(n-1))\ (1 - x(n-2)) \quad \textbf{(8)}$$

```
> ic2:= {x(0)=2.5,  x(1)=2.7};
```
$$ic2 := \{x(0) = 2.5, x(1) = 2.7\} \quad \textbf{(9)}$$

```
> ToSys(2,f,ic2);
```
$$[\{x(0) = 2.5, x(1) = 2.7\}, (x(n) = (1 - x(n-1))\ (1 - x(n-2)))_1] \quad \textbf{(10)}$$

```
> #M15.txt: Maple code for Lecture 15 of Dynamical Modesl in Biology, Fall 2021 (taught by Dr. Z.)

Help15 :=proc( ) : print(`HW3(u,v,w), HW2(u,v) , Dis1(F,y,y0,h,A), ToSys(k,z,f,INI) `) :end:



    #ToSys(k,z,f,INI): converts the kth order difference equation x(n)=f(x[n-1],x[n-2],...x[n-k]) to a first-order system
#x1(n)=F(x1(n-1),x2(n-1), ...,xk(n-1))
#x2(n)=x1(n-1)
#...

    #xk(n)=x[k-1](n-1). It gives the underlying transformation phrased in terms of z[1],...z[k], followed by the initial conditions. Try:
#ToSys:=proc(2,z,z[1]+z[2],[1,1])
ToSys :=proc(k, z, f, INI) local i :
[ f, seq(z[i-1], i=2..k) ], INI :
end:


    #HW3(u,v,w): The Hardy-Weinberg unerlying transformation witu (u,v,w), Eqs. (53a,53b, 53c) in Edelestein-Keshet Ch. 3
HW3 :=proc(u, v, w) : [u^2 + u*v + (1/4)*v^2,  u*v + 2*u*w + 1/2*v^2 + v*w, 1/4*v^2 + v*w + w^2] :end:


    #HW2(u,v): The Hardy-Weinberg unerlying transformation witu (u,v,w), Eqs. (53a,53b,53c) in Edelestein-Keshet Ch. 3 using the fact that u+v+w=1
HW2 :=proc(u, v) : expand([u^2 + u*v + (1/4)*v^2 , u*v + 2*u*(1-u-v) + 1/2*v^2 + v*(1-u-v)]) :end:




    #Dis1(F,y,y0,h,A): The approximate orbit of the Dynamical system approximating the 1D for
```

```
the autonomous  continuous dynamical  process dy/dt=F(y(t)) , y(0)=y0 with mesh size h
from t=0 to t=A
Dis1 :=proc(F, y, y0, h, A) local L, x, i :
L := Orb(x + h * subs(y=x, F), x, y0, 0, trunc(A/h)) :

L := [seq([i * h, L[i]], i=1 ..nops(L))] :
end:

##old stuff

#M13.txt: Maple code for Lecture 13 of Dynamical Modesl in Biology, Fall 2021 (taught by
Dr. Z.)
Help13 :=proc() :
 print(`RT2(x,y,d,K), Orb2(F,x,y,pt0,K1,K2), FP2(F,x,y), SFP2(F,x,y), PlotOrb2(L), FP2drz
(F,x,y), SFP2drz(F,x,y) `) :end:

with(LinearAlgebra) :


#RT2(x,y,d,K): A random rational transformation of degree d from R^2 to R^2 with postiive
integer coefficients from 1 to K The inputs are variables x and y and
#the output is a pair of expressions  of (x,y) representing functions. It is for generating examples
#Try:
#RT2(x,y,2,10);
RT2 :=proc(x, y, d, K) local ra, i, j, f, g :
ra := rand(1 ..K) : #random integer from -K to K
f := add(add(ra() * x^i * y^j, j=0 ..d-i), i=0 ..d) / add(add(ra() * x^i * y^j, j=0 ..d-i), i=0
 ..d) :
g := add(add(ra() * x^i * y^j, j=0 ..d-i), i=0 ..d) / add(add(ra() * x^i * y^j, j=0 ..d-i), i=0
 ..d) :
[f, g] :
end:



#Orb2(F,x,y,pt,K1,K2): Inputs a mapping F=[f,g] from R^2  to R^2 where f and g describe
functions of x and y, an initial point pt0=[x0,y0]
#outputs the orbit starting at discrete time K1 and ending in discrete time K2. Try
#F:=RT2(x,y,2,10);
#Orb2(F,x,y,[1.1,1.2],1000,1010);
Orb2 :=proc(F, x, y, pt0, K1, K2)  local pt, L, i :
pt := pt0 :

for i from 1 to K1-1 do
pt := subs({x=pt[1], y=pt[2]}, F) :
od:

L := [ ] :
for i from K1 to K2 do
```

```
L := [op(L), pt] :
pt := subs({x=pt[1], y=pt[2]}, F) :

od:
L :
end:

#FP2(F,x,y): The list of fixed points of the transformation [x,y]->F. Try
#FP2([x-y,x=y],x,y);
FP2 :=proc(F, x, y) local L, i :
L := [solve({F[1]=x, F[2]=y}, {x, y})] :

[seq(subs(L[i], [x, y]), i=1 ..nops(L))] :
end:


#SFP2(F,x,y): The list of Stable fixed points of the transformation [x,y]->F. Try
#SFP2([(1+x)/(1+y), (1+7*y)/(4+x)],x,y);
SFP2 :=proc(F, x, y) local L, J, S, J0, i, pt, EV :

L := evalf(FP2(F, x, y)) :
    #F is the list of ALL fixed points of the transformation [x,y]->F using the previous procedure
    FP2(F,x,y), but since we are interested in numbers we take the floating point version using
    evalf

J := Matrix(normal([[diff(F[1], x), diff(F[1], y)], [diff(F[2], x), diff(F[2], y)]])) :
    #J is the Jacobian matrix in general (in terms of the variables x and y). Note that J is a
    SYMBOLIC matrix featuring variables x and y

S := [ ] :  #S is the list of stable fixed points that starts out empty

for i from 1 to nops(L) do  #we examime it case by case
pt := L[i] : #pt is the current fixed point to be examined

J0 := subs({x=pt[1], y=pt[2]}, J) :
    #J0 is the NUMERICAL matrix obtained by plugging-in the examined fixed pt

EV := Eigenvalues(J0) :
    # We used Maple's command Eigenavalues to find the eigenvalues of this 2 by 2 matrix

if abs(EV[1]) < 1 and abs(EV[2]) < 1  then
 S := [op(S), pt] :
    #If both eigenvalues have absolute value less than 1 it means that they are stable, so we
    append the examined fixed point, pt, to the list of fixed points
fi:

od:
S : #the output is S
end:
```

```
###added Oct. 17, 20221
with(plots) :

PlotOrb1 :=proc(L) local i, d :

d := textplot([L[1], 0, 0]) :

for i from 2 to nops(L) do
d := d, textplot([L[i], 0, i-1]) :
od:
display(d) :
end:


PlotOrb2 :=proc(L) local i, d :

d := textplot([op(L[1]), 0]) :

for i from 2 to nops(L) do
d := d, textplot([op(L[i]), i-1]) :
od:
display(d) :
end:
###End added Oct. 17, 20221


###old stuff
#M11.txt: Maple code for Lecture 11 of Dynamical Models in Biology taught by Dr. Z.
Help11 :=proc( ) : print(` SFPe(f,x), Orbk(k,z,f,INI,K1,K2) `) :end:


    #SFPe(f,x): The set of fixed points of x->f(x) done exactly (and allowing symbolic
    parameters), followed by the condition of stability (if it is netween -1 and 1 it is stable)
#Try: FPe(k*x*(1-x),x);
#VERSION OF Oct. 12, 2021 (avoiding division by 0)
SFPe :=proc(f, x) local f1, L, i, M:
f1 := normal(diff(f, x)) :
L := [solve(numer(f-x), x)] :
M := [ ] :

for i from 1 to nops(L) do
 if subs(x=L[i], denom(f1) ) ≠ 0 then
  M := [op(M), [L[i], normal(subs(x=L[i], f1))]] :
 fi:
od:
M:

end:
```

```
#Added after class

    #Orbk(k,z,f,INI,K1,K2): Given a positive integer k, a letter (symbol), z, an expression f of z
    [1], ..., z[k] (representing a multi-variable function of the variables z[1],...,z[k]

    #a vector INI representing the initial values [x[1],..., x[k]], and (in applications) positive
    integres K1 and K2, outputs the

    #values of the sequence starting at n=K1 and ending at n=K2. of the sequence satisfying the
    difference equation
##x[n]=f(x[n-1],x[n-2],..., x[n-k+1]):

    #This is a generalization to  higher-order difference equation of procedure Orb(f,x,x0,K1,K2)
    . For example
#Orbk(1,z,5/2*z[1]*(1-z[1]),[0.5],1000,1010);  should be the same as
#Orb(5/2*z[1]*(1-z[1]),z[1],[0,5],1000,1010);
#Try:
#Orbk(2,z,(5/4)*z[1]-(3/8)*z[2],[1,2],1000,1010);
Orbk := proc(k, z, f, INI, K1, K2) local L, i, newguy :
L := INI : #We start out with the list of initial values

if not (type(k, integer) and type(z, symbol) and type(INI, list) and nops(INI) = k and type(K1,
    integer) and type(K2, integer) and K1 > 0 and K2 > K1) then
    #checking that the input is OK
 print(`bad input`) :
 RETURN(FAIL) :
fi:

while nops(L) < K2 do
 newguy := subs({seq(z[i] = L[-i], i = 1 ..k)}, f) :
    #Using what we know about the value yesterday, the day before yesterday, ... up to k days
    before yesterday we find the value of the sequence today
 L := [op(L), newguy] : #we append the new value to the running list of values of our sequence
od:

[op(K1 ..K2, L)] :

end:


####STAFT FROM M9.txt
#M9.txt: Maple Code for "Dynamical models in Biology" (Math 336) taught by Dr. Z., Lecture 9

Help9 :=proc( ) :
     print(` Orb(f,x,x0,K1,K2), Orb2D(f,x,x0,K) , FP(f,x) , SFP(f,x) , Comp(f,x)  `) :end:


    #Orb(f,x,x0,K1,K2): Inputs an expression f in x (desccribing) a function of x, an initial point,
    x0, and a positive integer K, outputs
```

```
#the values of x[n] from n=K1 to n=K2. Try: where x[n]=f(x[n-1]), . Try:
#Orb(2*x*(1-x),x,0.4,1000,2000);
Orb := proc(f, x, x0, K1, K2) local x1, i, L :
x1 := x0 :

for i from 1 to K1 do
 x1 := subs(x = x1, f) :
    #we don't record the first values of K1, since we are interested in the long-time behavior of
    the orbit
od:

L := [x1] :

for i from K1 to K2 do
 x1 := subs(x = x1, f) :    #we compute the next member of the orbit
 L := [op(L), x1] : #we append it to the list
od:

L : #that's the output

end:

#Orb2D(f,x,x0,K): 2D version of Orb(f,x,x0,0,K), just for illustration
Orb2D := proc(f, x, x0, K) local L, L1, i :
L := Orb(f, x, x0, 0, K) :
L1 := [[L[1], 0], [L[1], L[2]], [L[2], L[2]] ] :
for i from 3 to nops(L) do
 L1 := [op(L1), [L[i-1], L[i]], [L[i], L[i]]] :
od:
L1 :
end:


#FP(f,x): The list of fixed points of the map x->f where f is an expression in x. Try:
#FP(2*x*(1-x),x);
FP := proc(f, x)
evalf([solve(f = x, x)]) :
end:


#SFP(f,x): The list of stable fixed points of the map x->f where f is an expression in x. Try:
#SFP(2*x*(1-x),x);
SFP := proc(f, x) local L, i, f1, pt, Ls :
L := FP(f, x) : #The list of fixed points (including complex ones)

Ls := [ ] :       #Ls is the list of stable fixed points, that starts out as the empty list

f1 := diff(f, x) :  #The derivative of the function f w.r.t. x
```

```
for i from 1 to nops(L) do
pt := L[i] :

if abs(subs(x = pt, f1)) < 1 then

 Ls := [op(Ls), pt] :  # if pt,  is stable we add it to the list of stable points

fi:

od:

Ls :   #The last line is the output

end:

#Comp(f,x): f(f(x))
Comp :=proc(f, x) : normal(subs(x = f, f ) ) :end:




##added Oct. 17, 2021
#FP2drz(F,x,y): The list of fixed points of the transformation [x,y]->F. Dr. Z.'s way
#FP2([x-y,x+y],x,y);
FP2drz :=proc(F, x, y) local eq, i, L, S1 :
eq := [numer(F[1]-x), numer(F[2]-y)] :

L := Groebner[Basis](eq, plex(x, y)) :

S1 := evalf([solve(L[1], y)]) :
[seq([solve(subs(y = S1[i], L[2]), x), S1[i]], i = 1 ..nops(S1))] :
end:


#SFP2drz(F,x,y): The list of Stable fixed points of the transformation [x,y]->F. Try
#SFP2drz([(1+x)/(1+y), (1+7*y)/(4+x)],x,y);
SFP2drz :=proc(F, x, y) local L, J, S, J0, i, pt, EV :

L := FP2drz(F, x, y) :
    #F is the list of ALL fixed points of the transformation [x,y]->F using the previous procedure
    FP2(F,x,y), but since we are interested in numbers we take the floating point version using
    evalf

J := Matrix(normal([[diff(F[1], x), diff(F[2], x)], [diff(F[1], y), diff(F[2], y)]])) :
    #J is the Jacobian matrix in general (in terms of the variables x and y). Note that J is a
    SYMBOLIC matrix featuring variables x and y

S := [ ] :  #S is the list of stable fixed points that starts out empty

for i from 1 to nops(L) do  #we examime it case by case
```

```
pt := L[i] : #pt is the current fixed point to be examined

J0 := subs({x = pt[1], y = pt[2]}, J) :
    #J0 is the NUMERICAL matrix obtained by plugging-in the examined fixed pt

EV := Eigenvalues(J0) :
    # We used Maple's command Eigenavalues to find the eigenvalues of this 2 by 2 matrix

if abs(EV[1]) < 1 and abs(EV[2]) < 1 then
 S := [op(S), pt] :
    #If both eigenvalues have absolute value less than 1 it means that they are stable, so we
    append the examined fixed point, pt, to the list of fixed points
fi:

od:
S : #the output is S
end:
```

#ToSys(k,z,f,INI): converts the kth order difference equation x(n)=f(x[n-1],x[n-2],...x[n-k]) to a first-order system
#x1(n)=F(x1(n-1),x2(n-1), ...,xk(n-1))
#x2(n)=x1(n-1)
#...

#xk(n)=x[k-1](n-1). It gives the underlying transformation phrased in terms of z[1],...z[k], followed by the initial conditions. Try:
#ToSys:=proc(2,z,z[1]+z[2],[1,1])

Look above ^^^ see how this works.

Dynam Models Bio Homework 15

Fully convert by hand the fourth order recurrence

$$X(n) = \frac{x(n-1) + 2x(n-2) + 3x(n-3) + 11x(n-4)}{x(n-1) + x(n-3)}$$

where $x(0)=1$, $x(1)=5$, $x(2)=5$, $x(3)=2$

INTO A FIRST-ORDER SYSTEM with four sequences $x_1(n), x_2(n), x_3(n), x_4(n)$

Where $x_1(n) = \frac{x(n-1) + 2x(n-2) + 3x(n-3) + 11x(n-4)}{x(n-1) + x(n-3)}$

$x_2(n) = x(n-1)$

$x_3(n) = x(n-2)$

$x_4(n) = x(n-3)$

Why do we need 4 sequences instead of 5?

Can we plug in some values

As an extension to turning a 2nd Order Recurrence equation into a 1st order system, we can say

$x_4(n) = 0 + 0 + \dots + x(n-3)$

But do x_2 and x_3 need to agree?

Maybe they don't and everything might still work out.

Dynam Models Bio HW 15

x_1	whatever mess that is		
x_2	1	0	0
x_3	0	1	0
x_4	0	0	1

Maybe a characteristic Polynomial could be better

But the recurrence looks nonlinear because

$$x(n)\left[x(n-1)+x(n-3)\right]$$

$$\parallel$$

$$x(n-1)+2x(n-2)+3x(n-3)+11x(n-4)$$

Homework 15

4. Understand Section 3.6 in Reshet
Fill in all the blanks of all tables

Mating table 3.1

	genotype →	Father AA	Aa	aa
	%	u	v	w
mother AA	u	u^2	uv	uw
Aa	V	vu	v^2	vw
aa	w	wu	wv	w^2

Offspring table 3.2

Parent Combination	Frequency %	Offspring Genotype frequencies AA %	aA	aa
AA × AA	u^2	u^2	0	0
AA × Aa	$2uv$	uv	uv	0
AA × aa	$2uw$	0	$2uw$	0
Aa × Aa	v^2	$v^2/4$	$v^2/2$	$v^2/4$
Aa × aa	$2vw$	0	vw	vw
aa × aa	w^2	0	0	w^2
Total		$\left(u^2 + uv + \frac{v^2}{4}\right)$	$\left(\frac{v^2}{2} + vw + 2uw + uv\right)$	$\left(w^2 + vw + \frac{v^2}{4}\right)$

Problem Finished!