Intro Math Problem Solving
November 9

A Forest Fire Model

A Predator Prey Model

A Disease Model

A Linear Transition Model

Homework #10

Reference

Chapter 7, Section 1 of our textbook "Insight Through Computing" discusses the idea of a transition matrix, that represents how a vector of data changes from one step to the next, under a linear transformation.

.

Things That Change Over Time

All around us, things change with time. A forest fire starts, a seed is planted, a single body cell turns cancerous.

Observing the current conditions, we may be able to predict the future, if we understand the rules by which a system changes.

The standard mathematical technique for doing this involves differential equations, which is beyond the scope of this course.

We can use some simpler techniques to illustrate this prediction process, which starts with a model of a system, and then estimates the changes that occur in a series of small steps.

Forest Fires

Forest Fires

On a computer, of course, a forest becomes a rectangular array of trees.

We imagine that one tree has begun burning, perhaps because it was struck by lightning; it will burn for 1 time unit and then be consumed.

Depending on dryness and wind, neighboring trees have a certain probability p of catching fire from a burning neighbor.

We are interested in the typical percentage of the forest that will be destroyed in such a fire.

A Forest Model

Let T be an MxN array, with each entry (I,J) representing a single tree in the forest.

The value of T(I,J) will represent the current "state" of the tree:

0: untouched

1: burning

2: burnt

A Forest Change Model

How trees may change from one step to the next:

unburnt:

if a neighbor is burning, then

with probability p, this tree changes to burning

burning: changes to burnt

burnt: does not change

Initialization

Create the MxN array T.

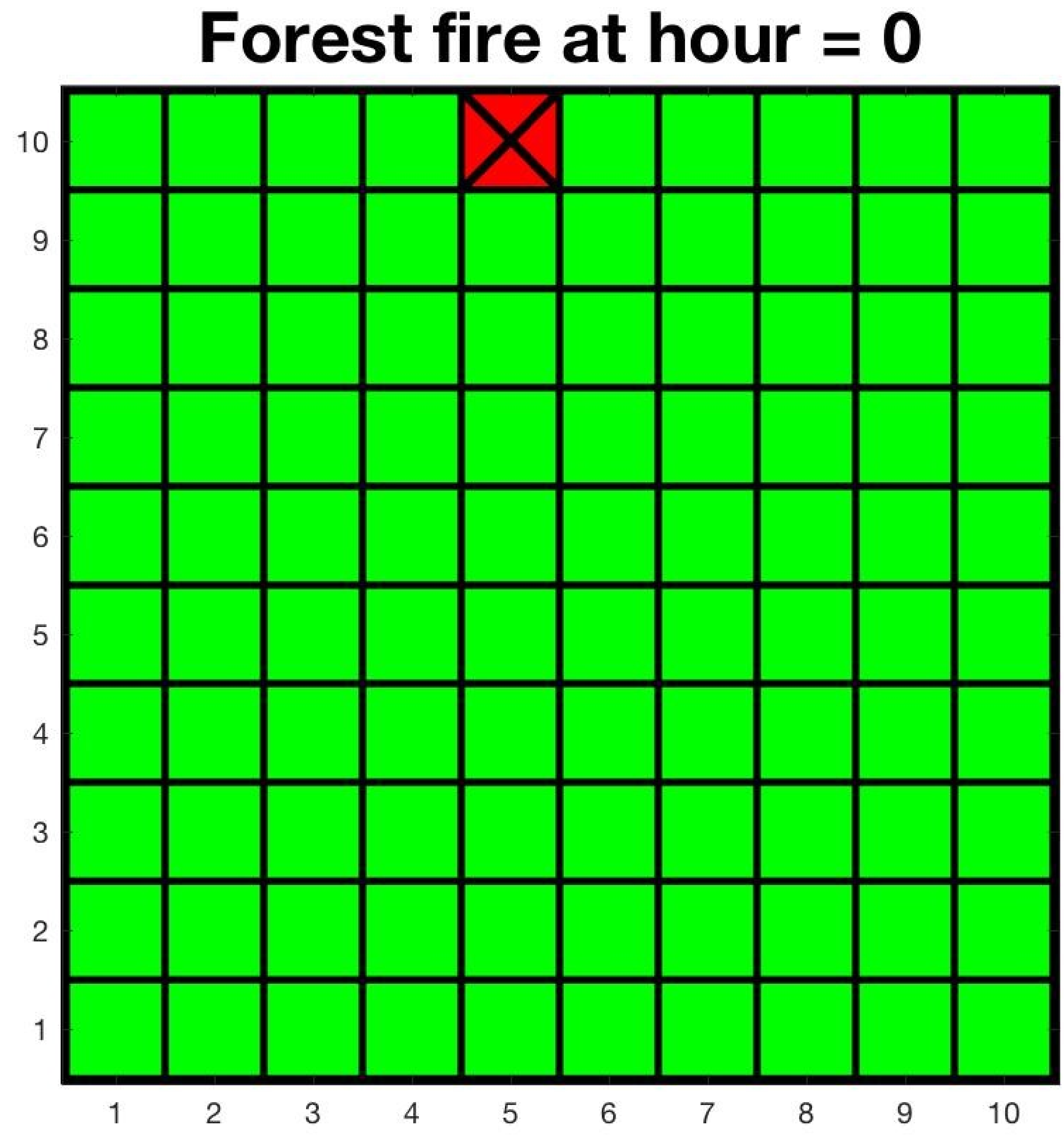
Set time step to 0.

Initialize all entries of T to “unburnt”.

Set one entry of T to “burning” (lightning hit?)

Display the current forest.

The Fire Starts at a Random Tree



Next Step

Let Told store a copy of T.

Set STEP = STEP + 1

For each tree Tij in the forest

  if Tij was UNBURNT then

    each BURNING neighbor has a probability P of setting this tree on fire

  else if Tij was BURNING

    it is now BURNT

We make decisions using the OLD values, in Told, and we carry out our decisions by updating values in T.

“Neighbors” of Tree Tij

```
        N              NW – N – NE
        |               |   |   |
 W  – C – E            W – C – E
        |               |   |   |
        S              SW – S – SE
```

We could use 4 neighbors or 8.

8 Neighbors Convenient

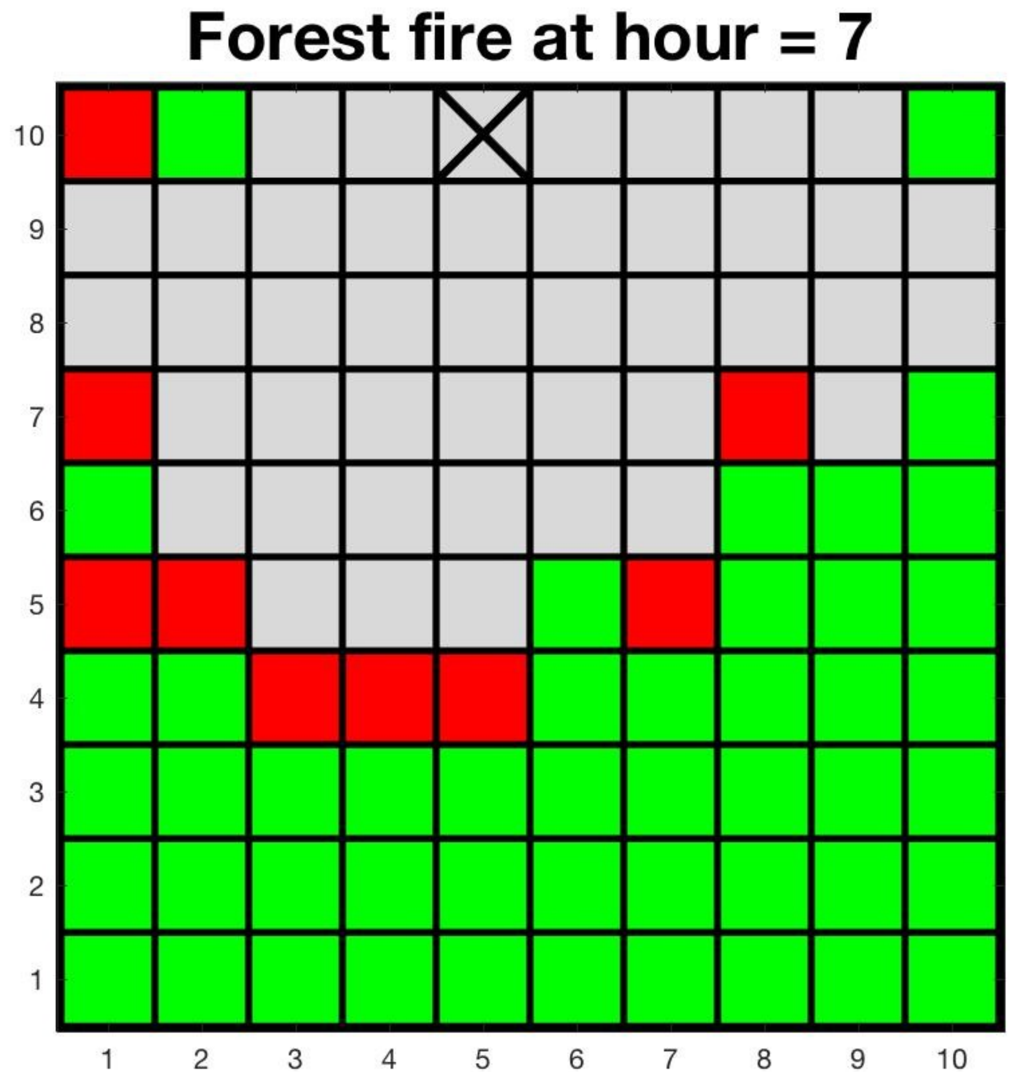
The 8 neighbor idea is convenient, because it means we can easily describe the neighborhood even when a tree is on the boundary:

```
im1 = max ( i – 1, 1 );
ip1 = min ( i + 1, m );
jm1 = max ( j – 1, 1 );
jp1 = min ( j + 1, n );
NABES = T(im1:ip1,jm1:jp1);
```

Counting BURNING Neighbors

```
NABES = T(im1:ip1,jm1:jp1);
K = ( NABES == BURNING );  ← 3x3 array of 0 and 1
k_num = sum ( sum ( K ) );  ← sum(sum for arrays
%
%  Every burning neighbor gets a chance to light T(i,j).
%
for k = 1 : k_num
  if ( rand ( ) < prob )
    T(i,j) = BURNING;
  end
end
```

Fire Progresses Through Forest



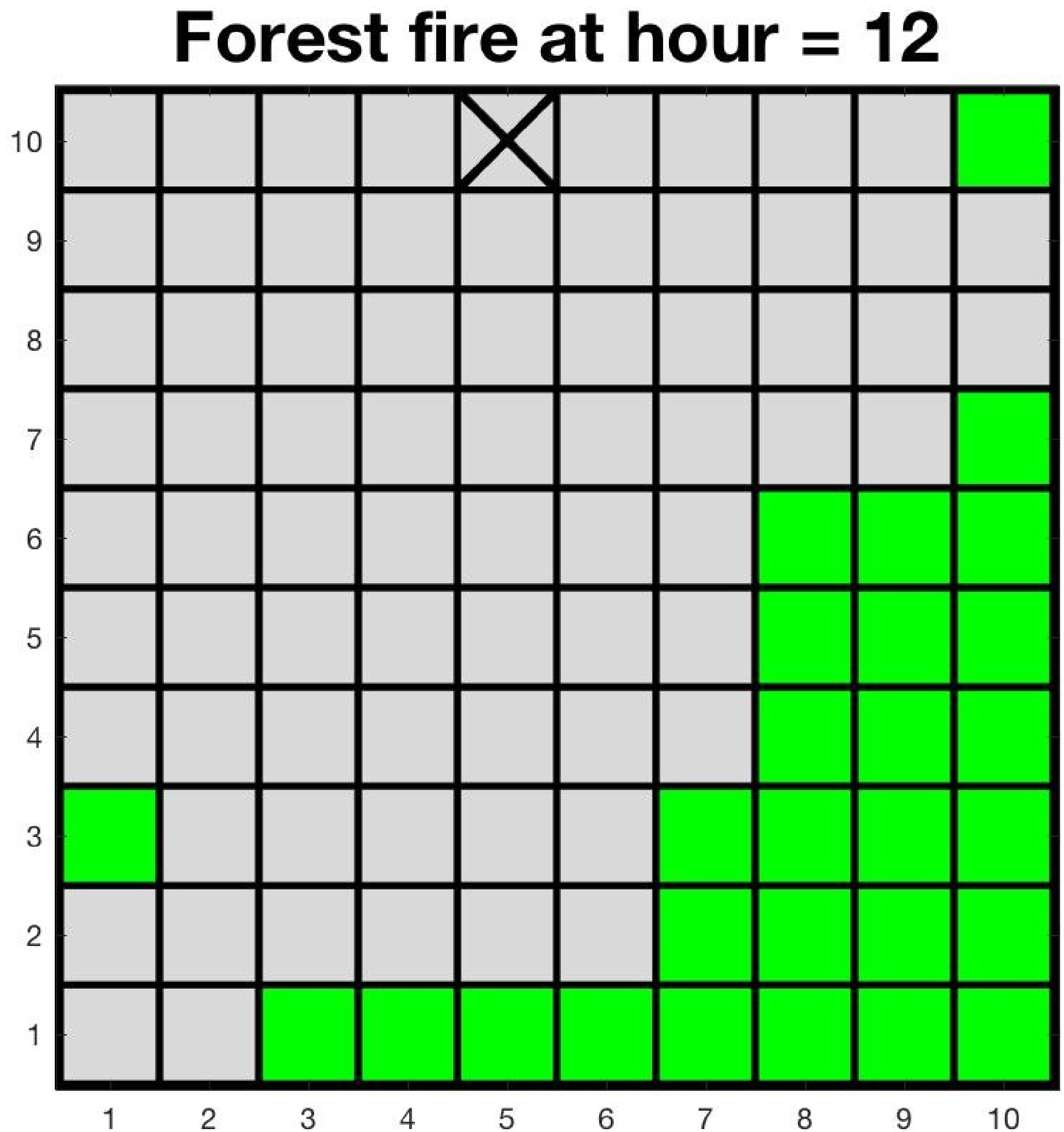
Stop When No Tree is Burning

The fire is out when no tree is burning.

This condition is easy to check.

```
FIRE = ( T == BURNING );
if ( sum ( sum ( FIRE ) ) == 0 )
  break
end
```

Stop When No Tree is Burning



Measurable Results

We can count the percent of the forest that was burned, or the length in hours of the fire.

We can count the number of separate areas of unburned trees.

We can look at how these measures of damage are related to the probability of the fire spreading from one tree to the next.

It's not hard to add features to this simple model that allow us to make a more realistic simulation.

Fancier Models

Wind has a strong influence on the direction and degree to which a fire spreads.

A forest often includes cleared areas intended to stop or delay the spread of fire. Where should they be? How wide should a cleared area be?

Fire prefers to move uphill, so topography matters.

Fire sends up burning cinders that can land far away, allowing the fire to make big jumps.

Forests are a combination of grassland, brush, and trees of different types and ages, which resist fire in different ways.

Predator/Prey Population Model

Tracking Two Populations

Suppose we are monitoring the number of foxes and rabbits in an isolated valley, by conducting a count each month.

We could imagine storing our data in two vectors or lists,

F = [F(1), F(2), F(3), ..., F(N)]

R = [F(1), R(2), R(3), ..., R(N)]

We'd like to see if a computer model can simulate this situation.

Simple Population Growth

Recall we looked at a model for population growth in Mexico. If "b" was the chance that 1 person would have 1 baby in 1 time period, then we had the model:

$$P(t+1) = P(t) + b * P(t) = (1 + b) * P(t).$$

Here "1+b" can be thought of as:

1 = everyone who was alive is still alive

b = some babies are added to the population

The result was a kind of exponential growth.

Complications

Keeping track of two populations wouldn't be much harder than one population, except that we know that foxes and rabbits "interact".

In particular, while rabbits can depend on a steady food supply of grass and clover, foxes need to eat rabbits, or they will starve.

So both populations have tendencies to grow or decrease, and the size of one population affects the survival chances of the other.

Interaction Model

A particular model has growth AND death parts:

A) Rabbits:

growth: = +0.001 * R(t) death: = - 0.000002 * F(t)*R(t)

B) Foxes:

death: = -0.005 * F(t) growth: +0.0000015 * R(t)*F(t)

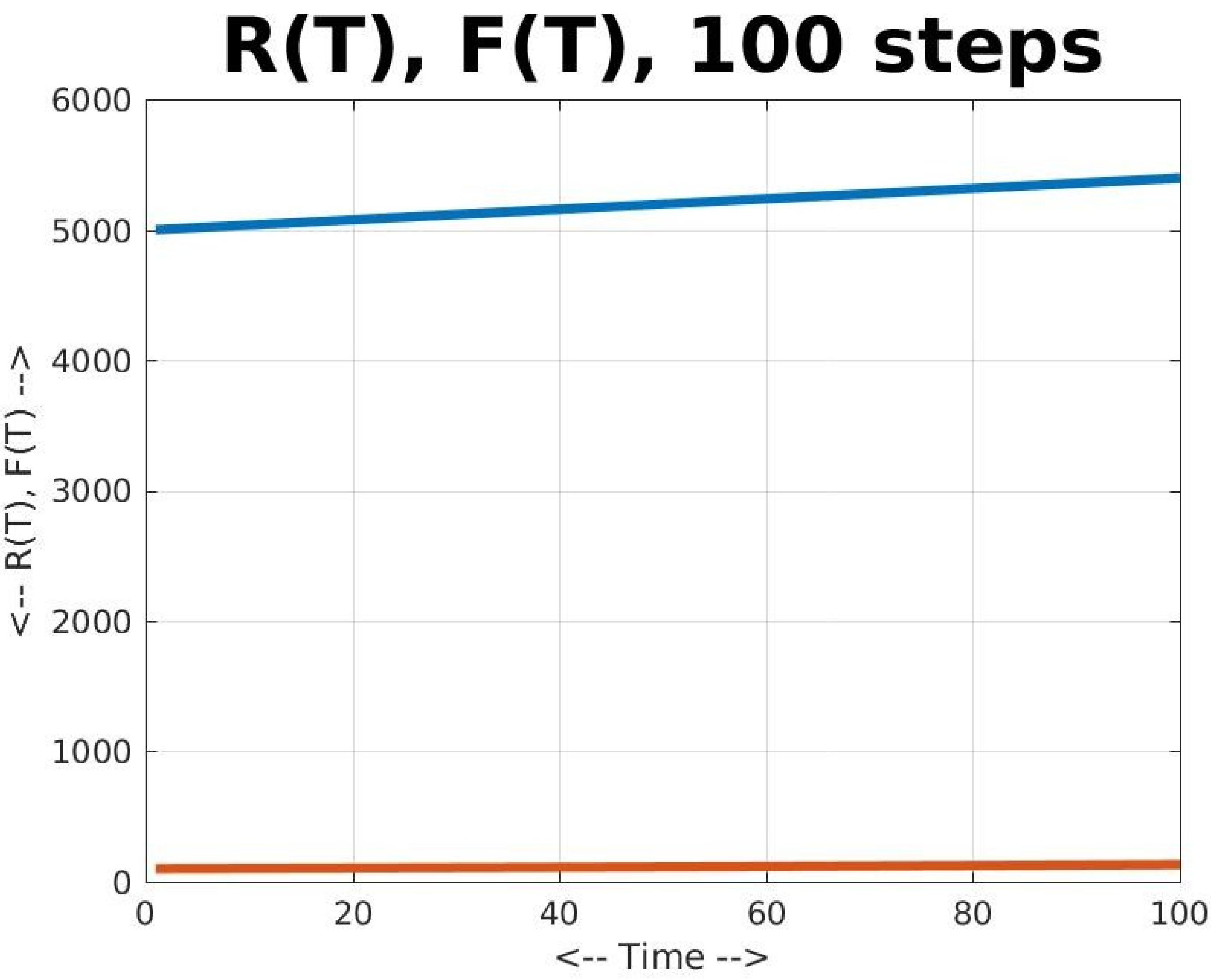
So in a month, a rabbit has 1/thousand chance of adding a baby to the population, and a fox has a 5/thousand chance of dying (by starvation).

F(t)*R(t) measures the chance that any fox will meet any rabbit in the month. There is a 2/million chance that any given rabbit will die because of such an encounter; there is a 1.5/million that a baby fox will be born because a rabbit was caught and eaten.

Follow Populations for M months

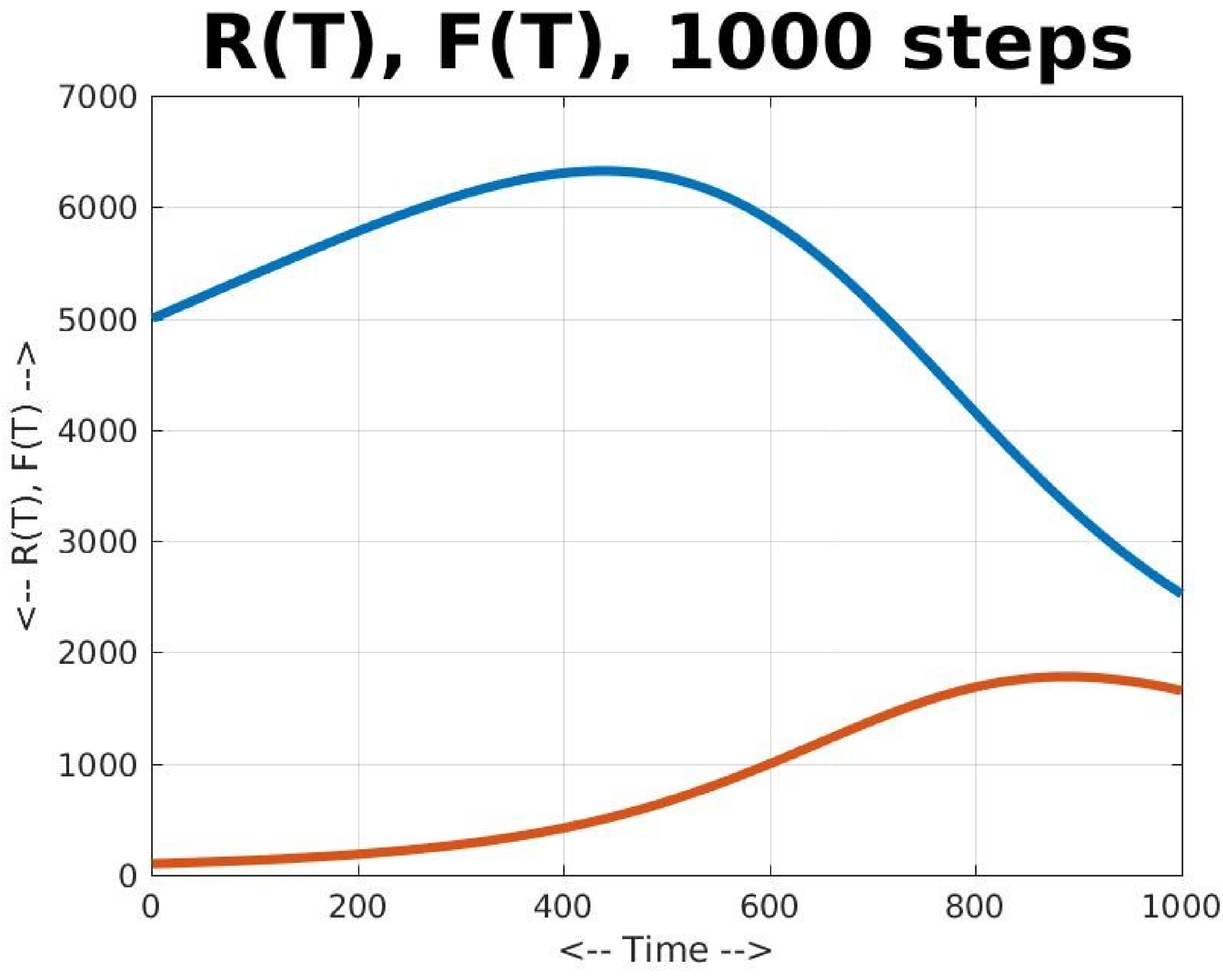
```
gr = 0.001; dr = 0.000002; df = -0.005; gf =0.0000015 ;
R = zeros ( 1, m ); F = zeros ( 1, m );
for t = 1 : m
  if ( t == 1 )
    R(1) = 5000;
    F(1) = 100;
  else
    R(t+1) = ( 1.0 + gr ) * R(t) – dr * F(t) * R(t);
    F(t+1) = ( 1.0 + df ) * F(t) + gf * F(t) * R(t);
  end
end
```

M=100 Steps



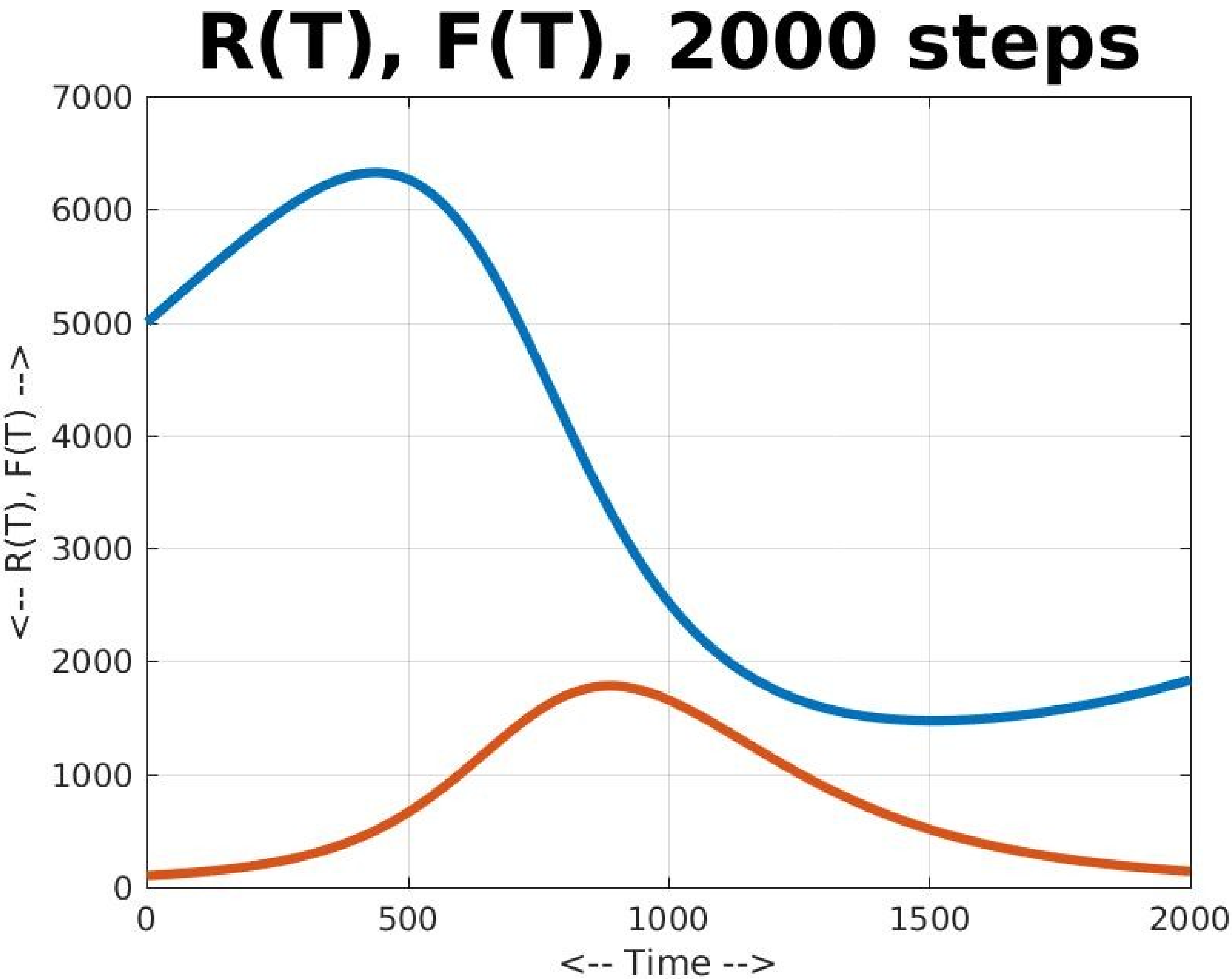
R(T), F(T), 100 steps
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
<-- R(T), F(T) -->
<-- Time -->

M=1000 Steps



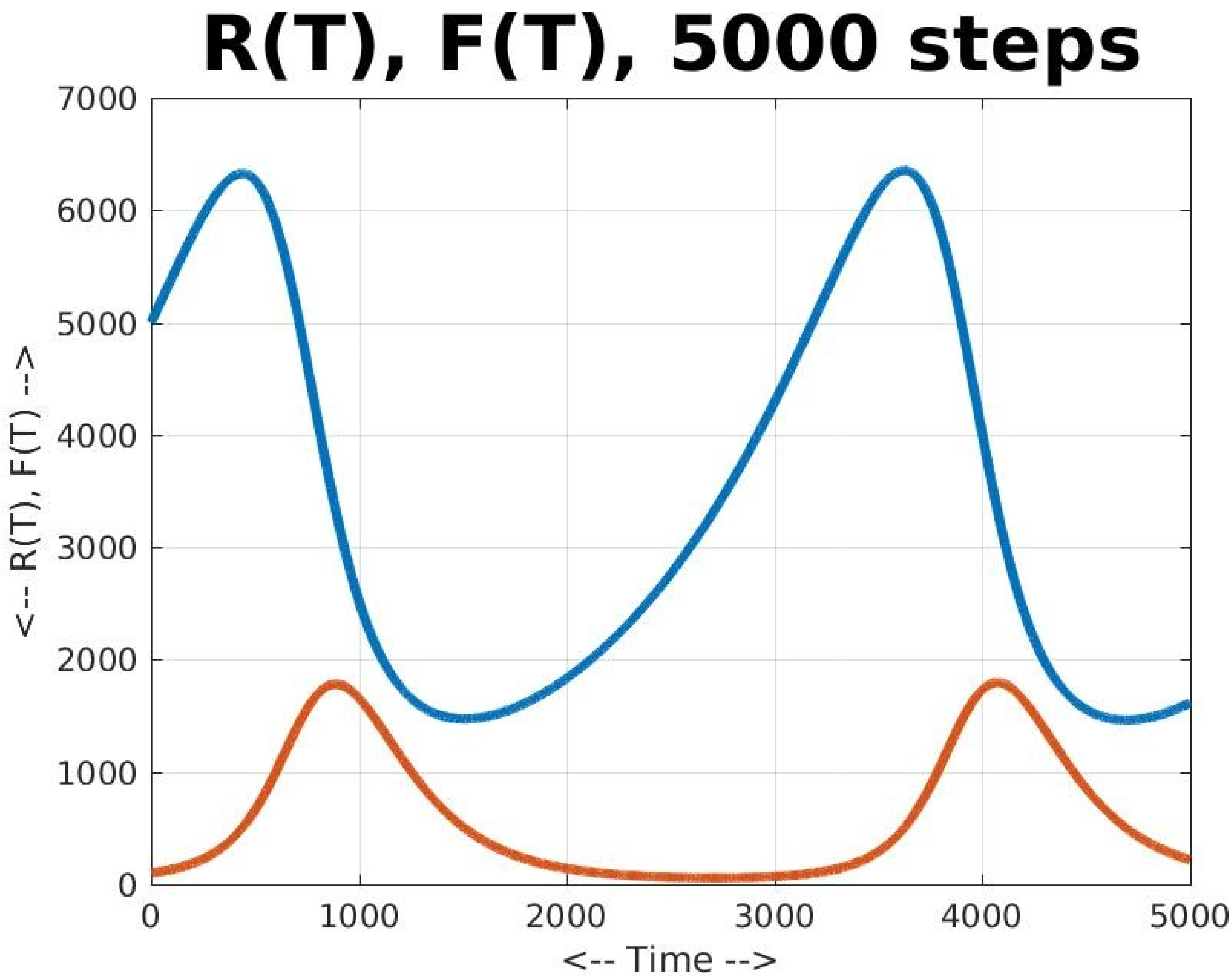
R(T), F(T), 1000 steps
7000
6000
5000
4000
3000
2000
1000
0
<-- R(T), F(T) -->
0
200
400
600
800
1000
<-- Time -->

M=2000 Steps



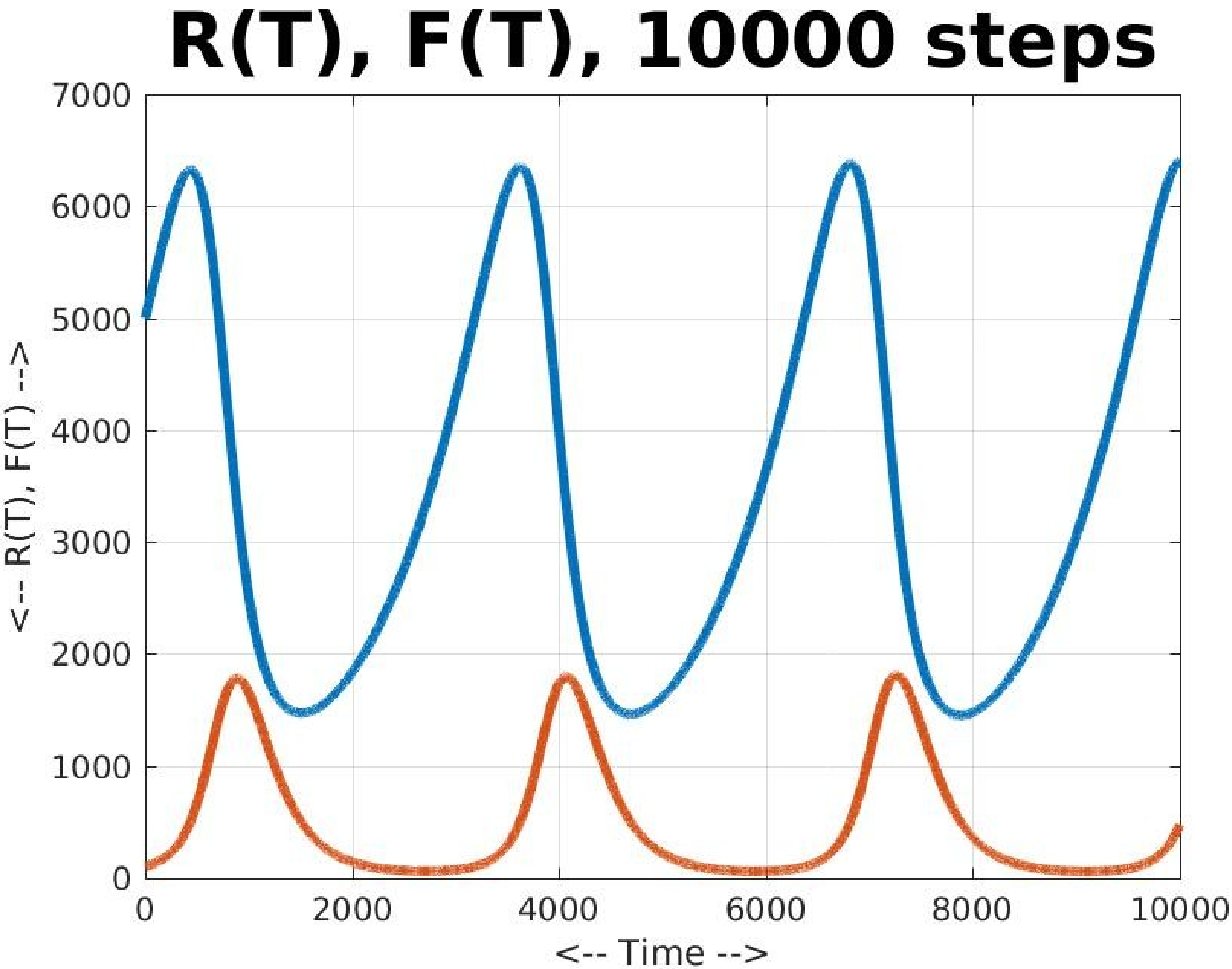
R(T), F(T), 2000 steps
7000
6000
5000
4000
3000
2000
1000
0
<-- R(T), F(T) -->
0
500
1000
1500
2000
<-- Time -->

M=5000 Steps



R(T), F(T), 5000 steps
7000
6000
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000
<-- Time -->
<-- R(T), F(T) -->

M=10,000 Steps



Plot Comments

M=100: we see very little change. The system does changes, but over a longer time than this.

M=1000: rabbits are disappearing, foxes rule!

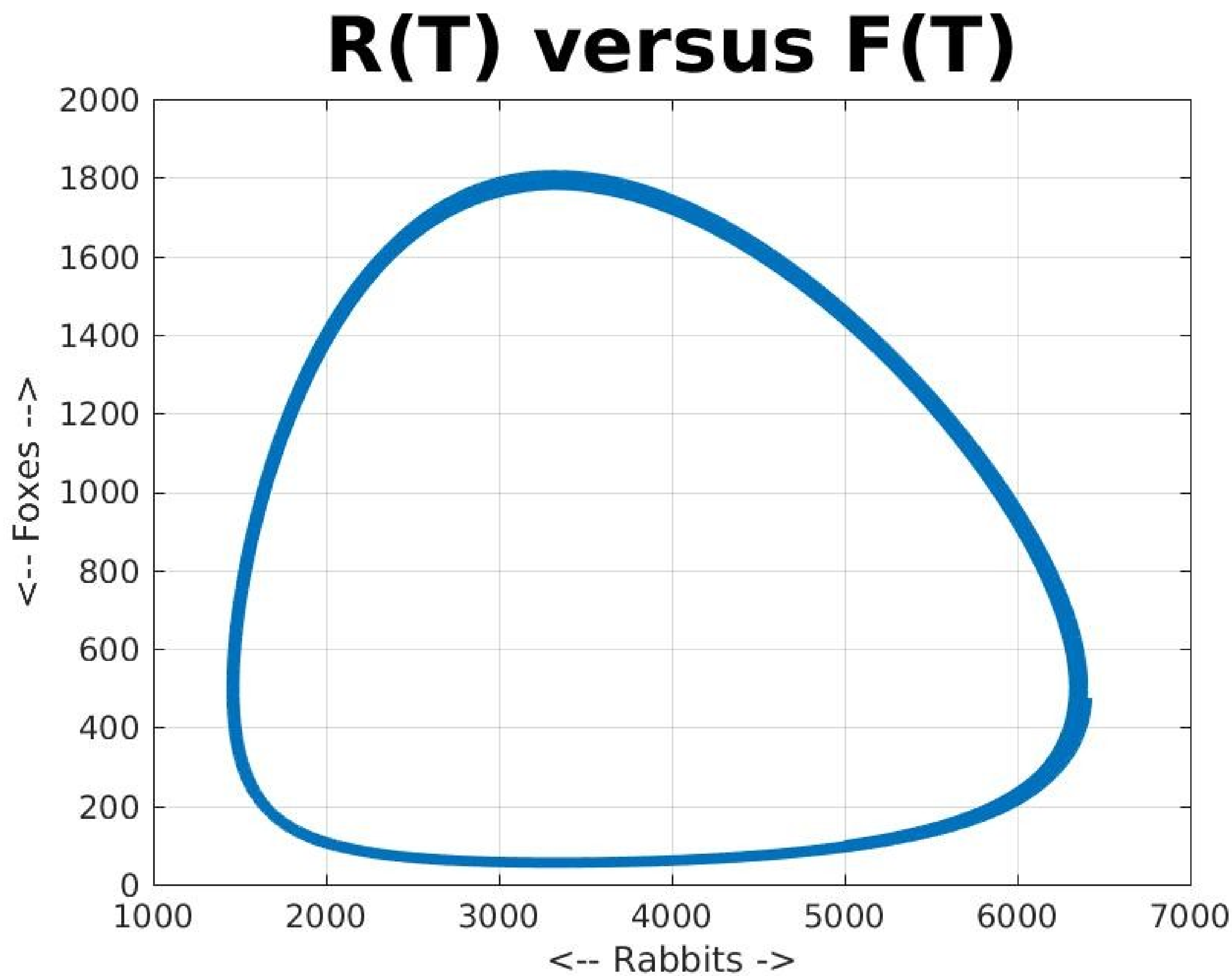
M=2000: rabbits recover a bit, foxes die out.

M=5000: both populations have a double peak

M=10000: the systems are periodic!

For a pair of periodic variables, we can plot the “phase plane”, which here is R(T) versus F(T).

The “Phase Plane”



Phase Plane Clues

The phase plane plot emphasizes the idea that, at least in our model, the fox and rabbit populations rise and fall over a large range, but follow a cycle that ensures that neither group dies out.

Because this corresponds to what we see in real life, we have a feeling that our predator-prey model has successfully imitated some features of this living example.

A Disease Model

b = the rate at which susceptible people become infectious
r = the rate at which infectious people recover/develop immunity

The SIR Disease Model

Many diseases are spread from one person to another; often, if a person contracts the disease, when they recover, they are immune and can't get the disease again.

The SIR disease model tries to simulate this kind of epidemic; it estimates the likelihood of transmission and the duration of the disease.

It asks why sometimes a disease outbreak is very limited, and other times becames an epidemic.

This is another case that is best handled by using differential equations; we will use a simplified approach that will still allow us to explore this idea.

Three Kinds of People

To put together our model, we will suppose that we have N people in a population.

Each person is in one of three classes:

S: “susceptible”: could get the disease

I: “infectious”: has the disease and can transmit it

R: “recovered”: no longer has the disease, can’t transmit it, and can’t get it again.

Two Changes

At any time t, our population is divided into S(t), I(t) and R(t) people. Over a short time step, each person encounters another person.

S-> I: Each susceptible person meets someone. I/N of these people are infectious. “b” or “beta” is the chance that an encounter with an infectious person will transmit the disease.

I->R: By the next time step, some I people can recover; the likelihood of this depends on a recovery rate “g” or “gamma”.

A Peek at Differential Equations

$$
\begin{aligned}
\frac{dS}{dt} &= -\beta\frac{SI}{N} \\
\frac{dI}{dt} &= \beta\frac{SI}{N} - \gamma I \\
\frac{dR}{dt} &= \gamma I \\
N &= S + I + R
\end{aligned}
$$

Interpret Differential Equations

These differential equations have a meaning:

The change in S over time is a decrease, proportional to beta, the number of S people, and the proportion of infected people (I/N).

The change in I over time is the newly infected S people, minus the I people who recover.

The change in R over time is the I people who have recovered.

Simplified Differential Equations

```
(S(t+dt)-S(t)) / dt = -beta * S(t) * I(t) / N

(I(t+dt)-I(t)) / dt = +beta * S(t) * I(t) / N – gamma * I(t)

(R(t+dt)-R(t)) / dt =                          + gamma * I(t)


S(t+dt) = S(t) + dt * ( -beta * S(t) * I(t) / N )

I(t+dt) = I(t) + dt * ( +beta * S(t) * I(t) / N – gamma * I(t) )

R(t+dt) = R(t) + dt * (                          + gamma * I(t) )


S(i+1) = S(i) - BETA * S(i) * I(i) / N

I(i+1) = I(i) + BETA * S(i) * I(i) / N – GAMMA * I(i)

R(i+1) = R(i)                          + GAMMA * I(i)
```

sir.m

```
function [ S, I, R ] = sir ( m, beta, gamma )

  S = zeros ( 1, m );
  I = zeros ( 1, m );
  R = zeros ( 1, m );

  for i = 1 : m

    if ( i == 1 )
      S(i) = 99;
      I(i) = 1;
      R(i) = 0;
      N = S(i) + I(i) + R(i);
    else
      S(i) = S(i-1) - beta * S(i-1) * I(i-1) / N;
      I(i) = I(i-1) + beta * S(i-1) * I(i-1) / N - gamma * I(i-1);
      R(i) = R(i-1)                              + gamma * I(i-1);
    end

  end
```

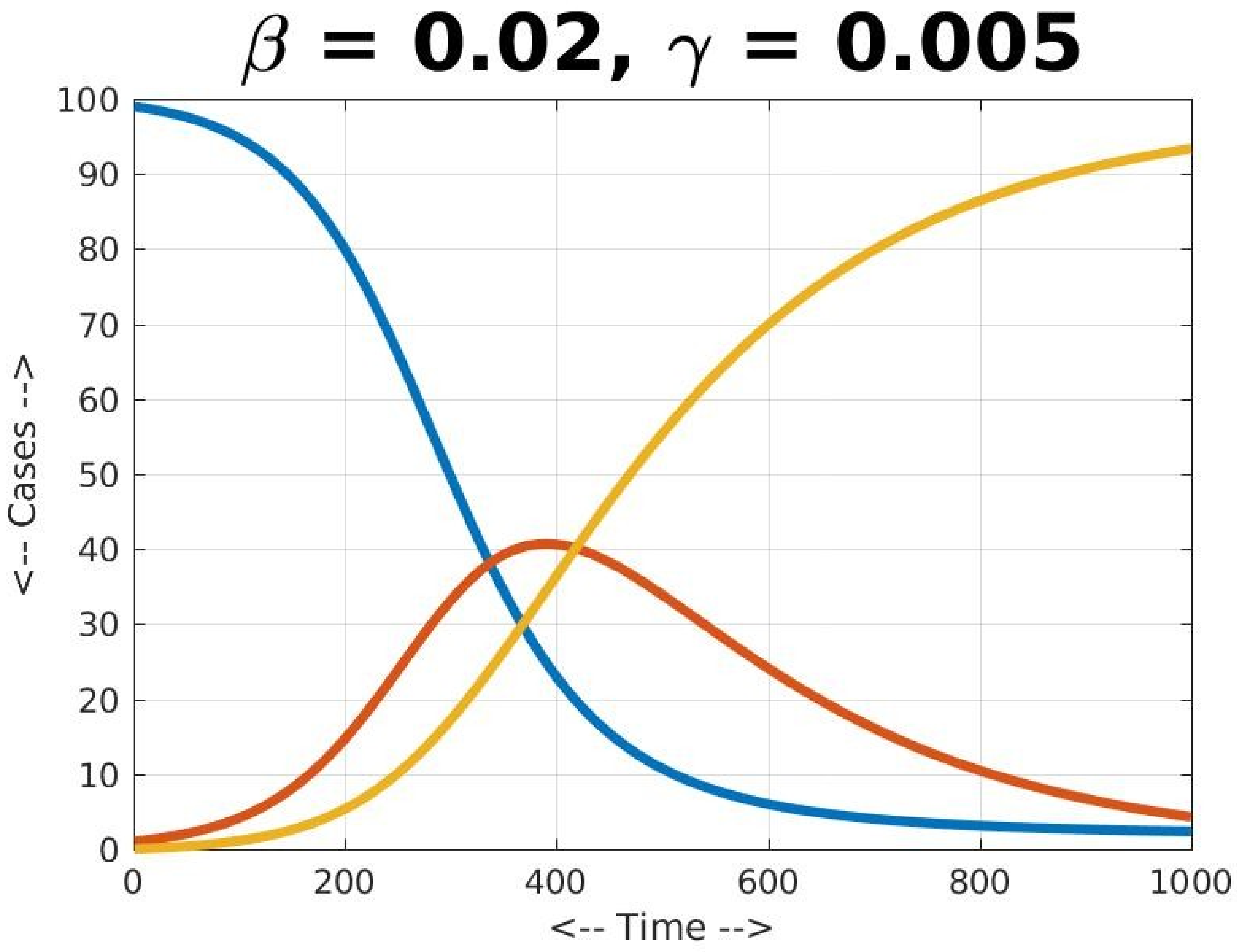
Study Infectious Rate Beta

Beta measures how easy it is to catch the disease.

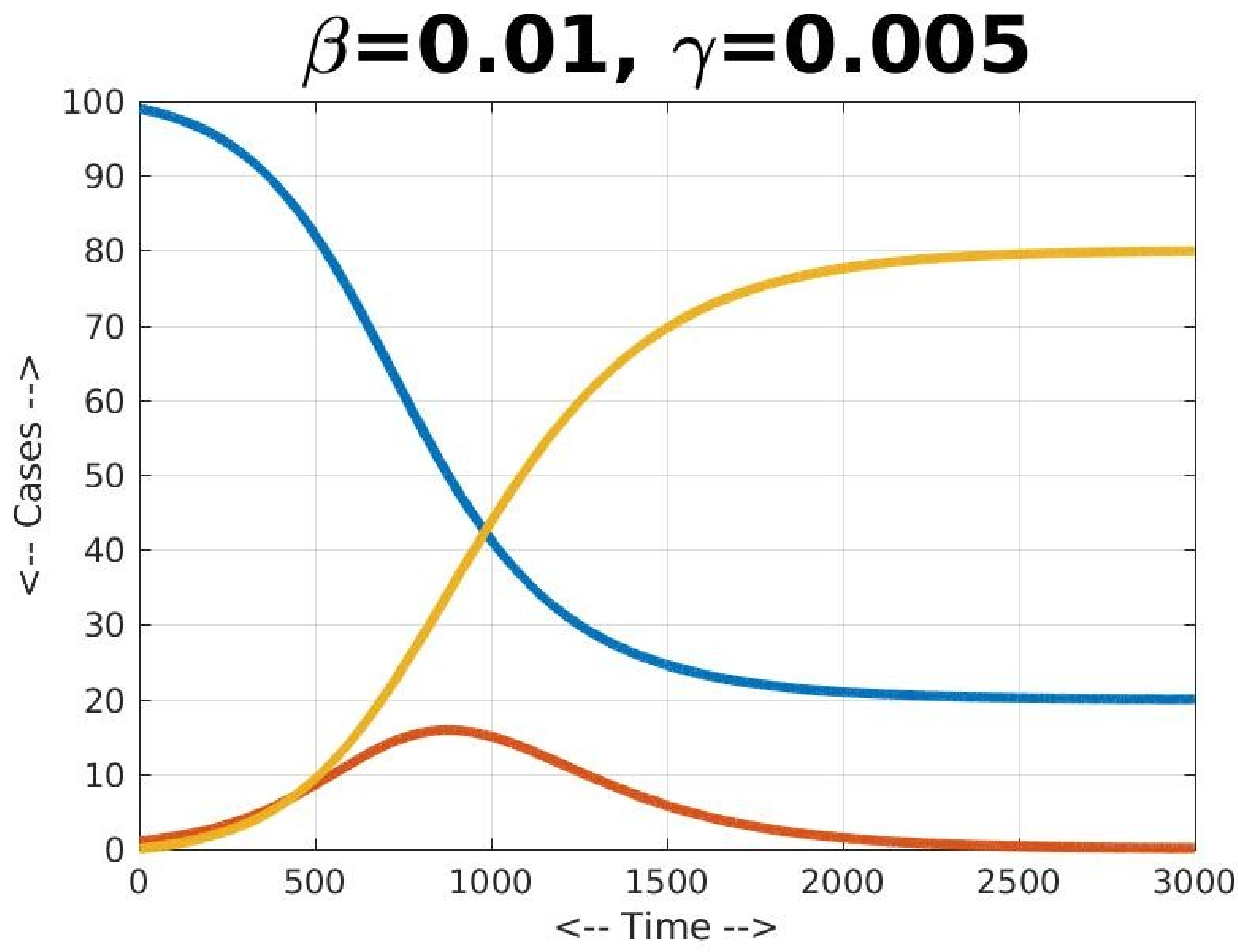
We'll fix gamma at 0.005, and look at what happens as we decrease Beta.

Each time, the disease takes longer to "settle down", and the number of people who never get the disease increases.

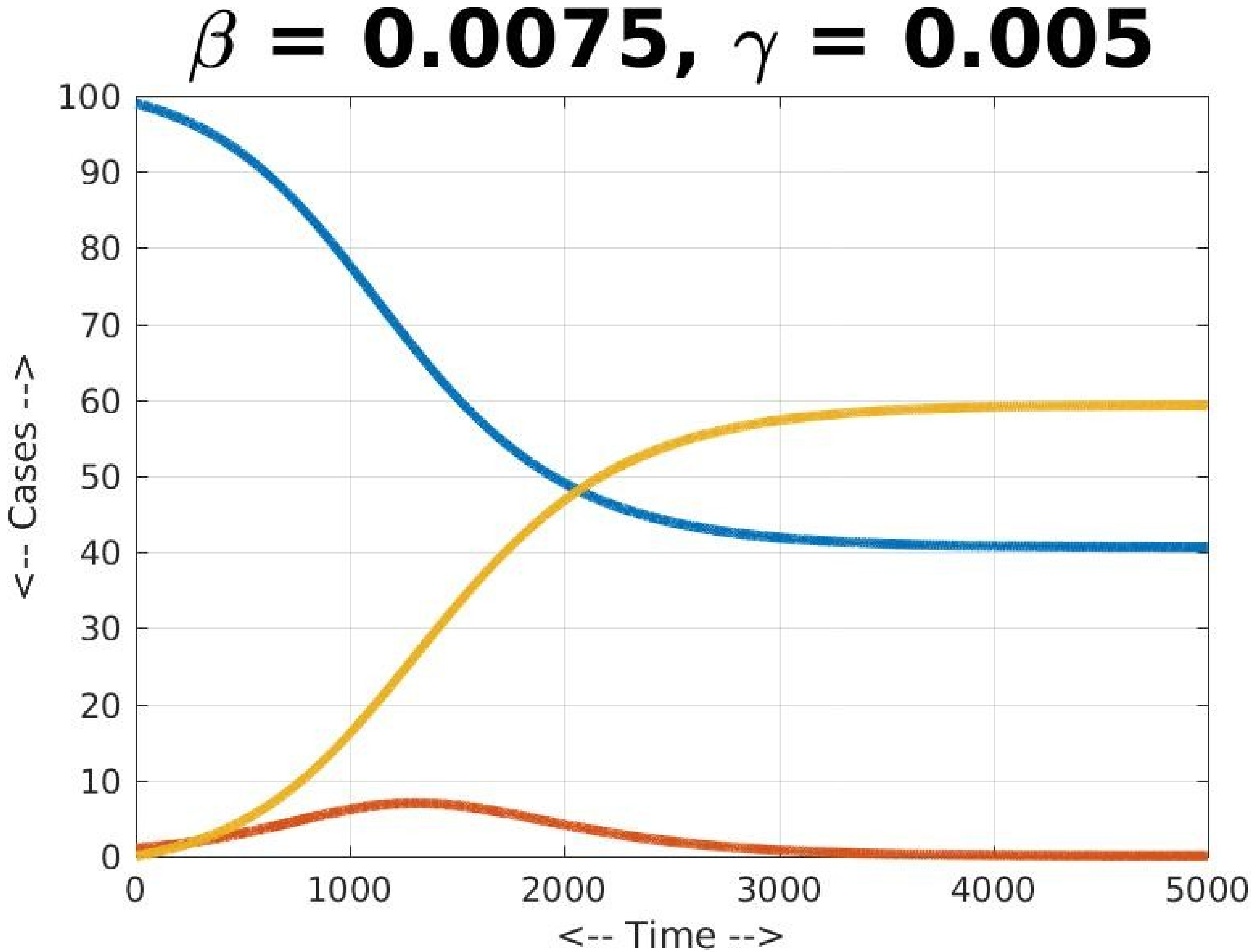
Beta = 0.02, 0.01, 0.0075



Beta = 0.02, 0.01, 0.0075



Beta = 0.02, 0.01, 0.0075



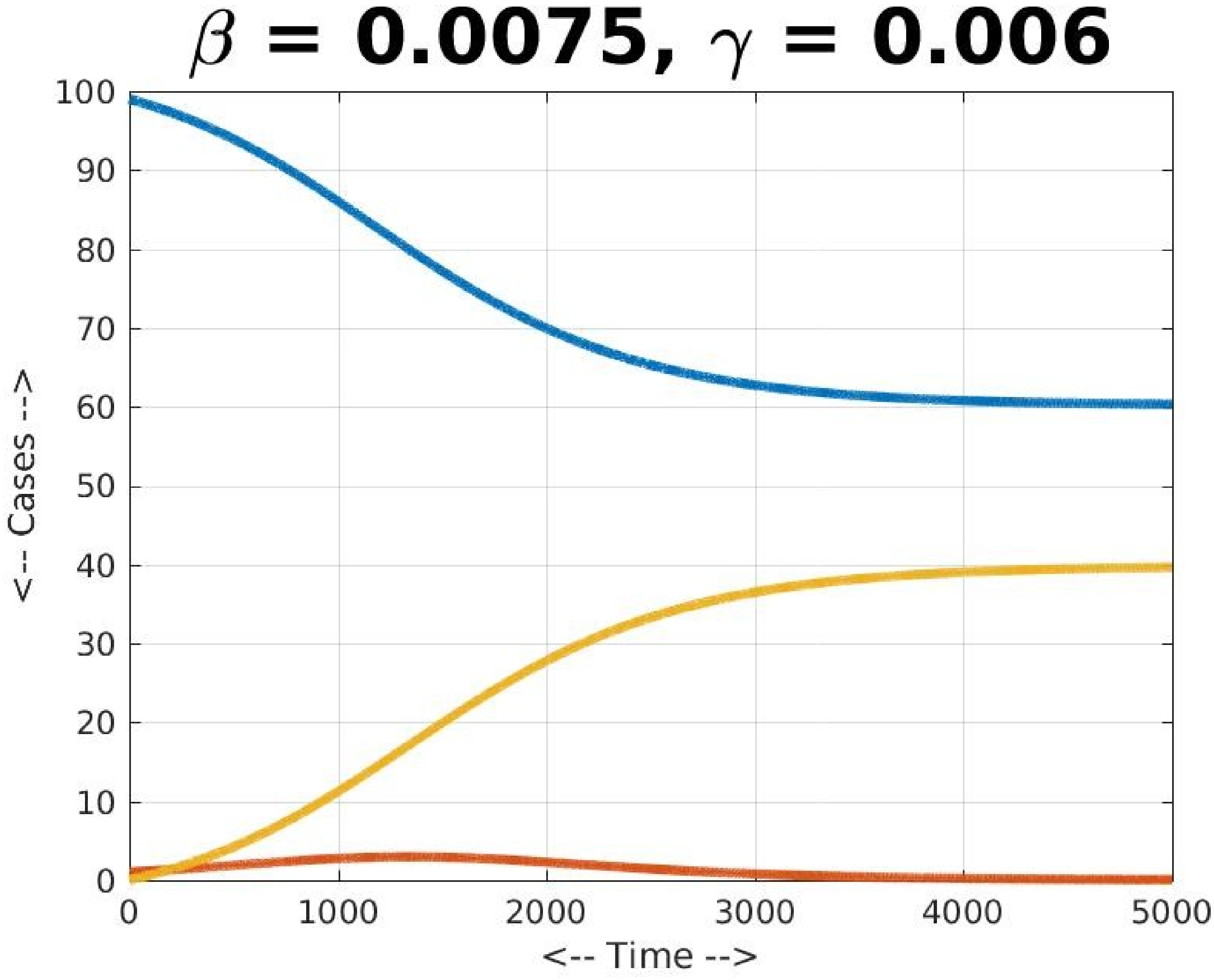
Study Recovery Rate Gamma

Gamma measures how quickly one recovers from the disease.

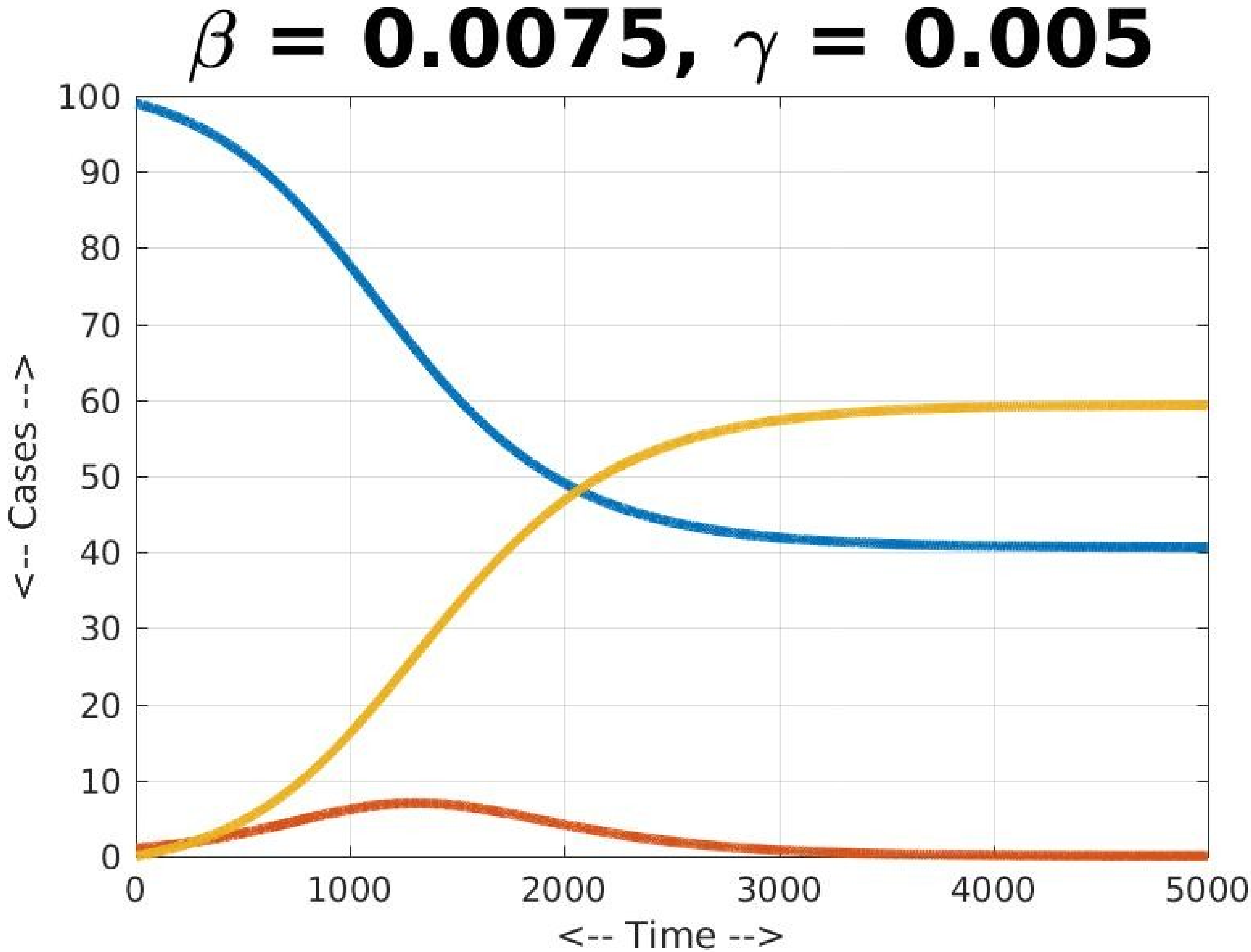
We’ll fix beta at 0.0075, and look at what happens as we decrease gamma from 0.006, 0.005, 0.004.

By slowing down the recovery rate, an infectious person has more time to infect more people, and the disease becomes more widespread.

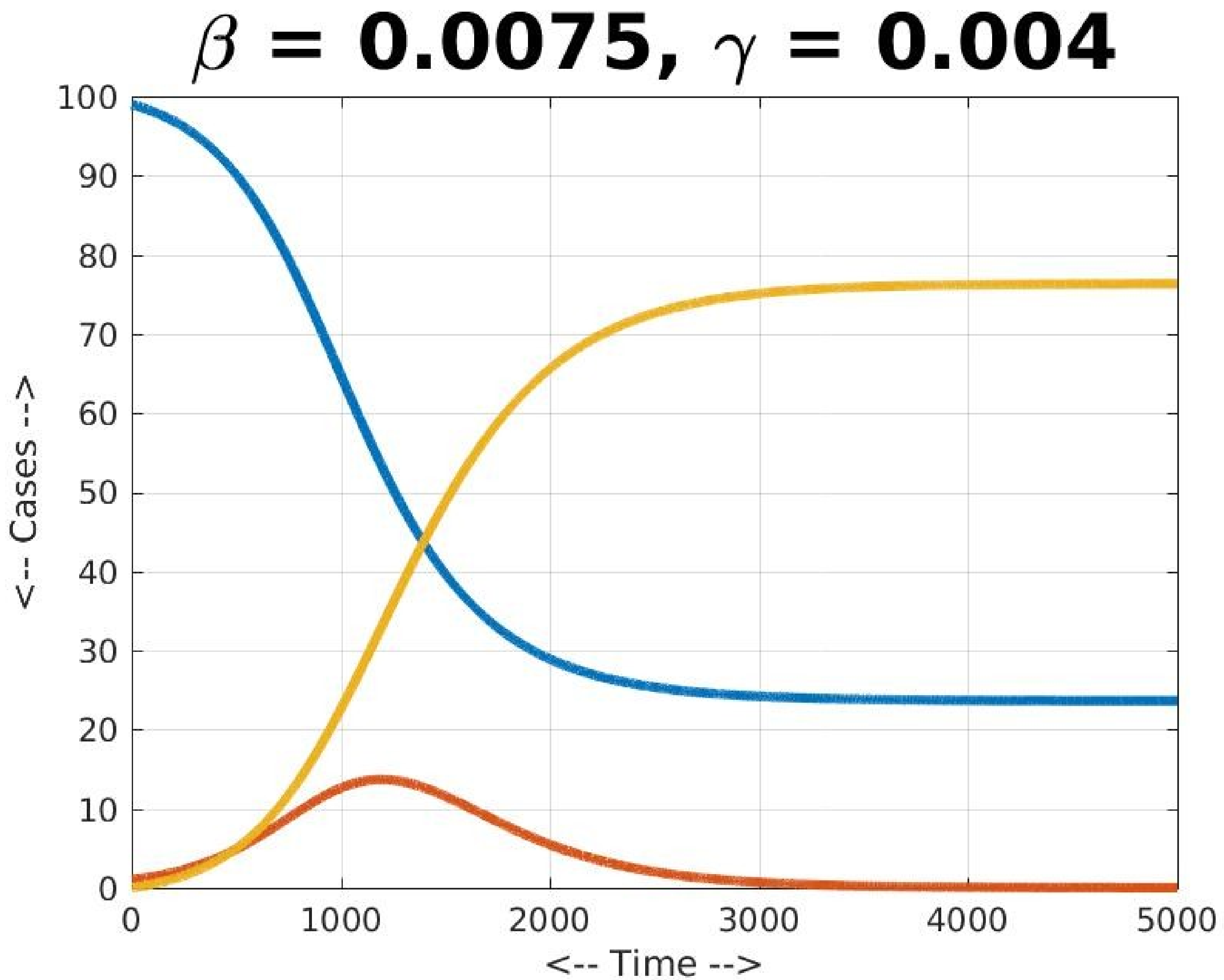
Gamma = 0.006, 0.005, 0.004



Gamma = 0.006, 0.005, 0.004



Gamma = 0.006, 0.005, 0.004

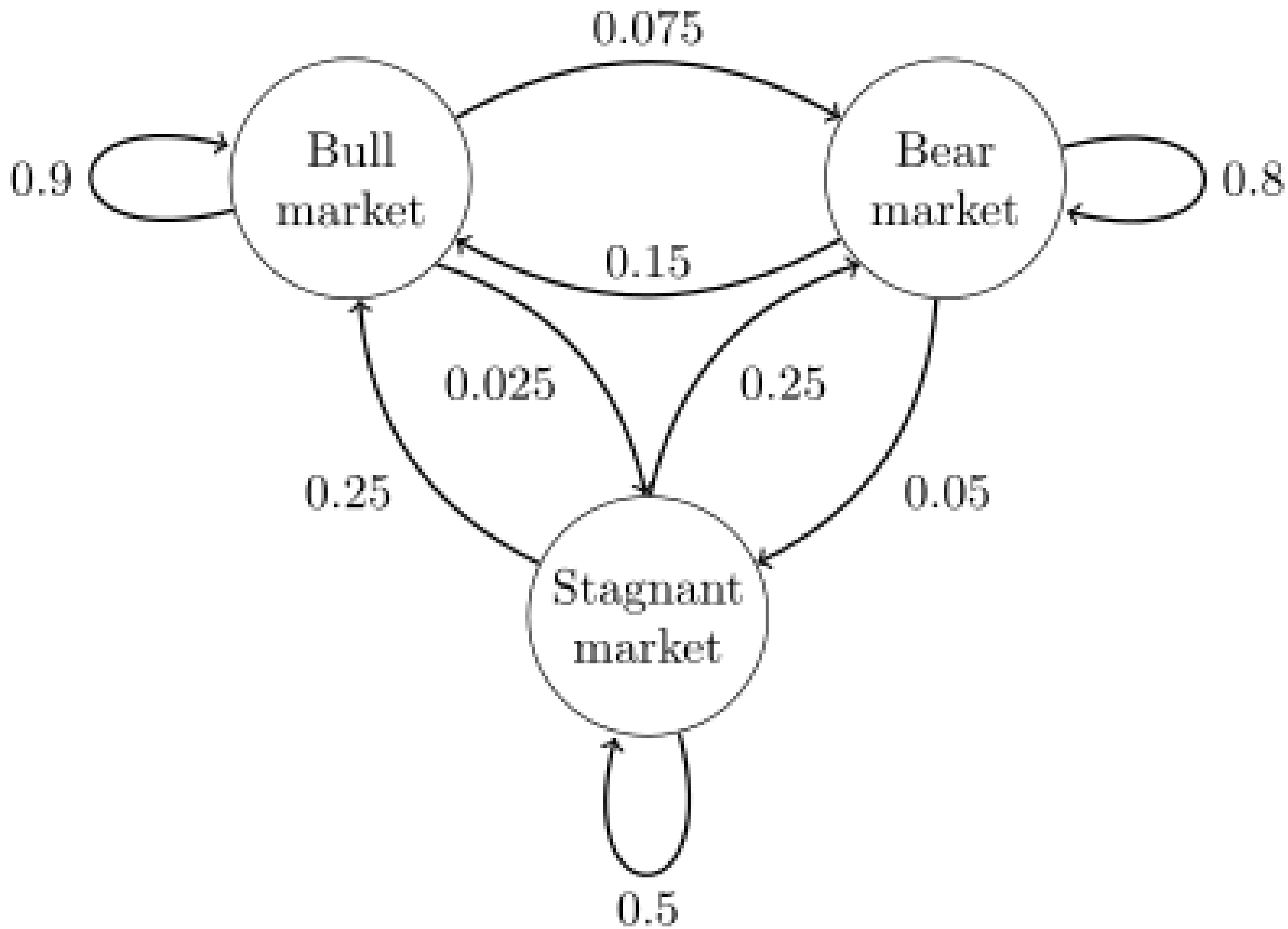


Observations

At any one time, most people aren't sick; all other things being equal, the value of gamma (recovery rate) controls how long people are sick, and hence how many are sick at any one time.

The transmissibility (beta) controls how fast the disease spreads. If beta is high enough, everyone is going to get the disease.

A Transition Matrix



System / States / Transitions

Some systems can be thought of as having a set of possible states. The system is always in some particular state.

At regular time intervals, the system can move (or transition) from one state to another, or stay where it is.

Each possible change has a known likelihood, known as the transition probability.

The Transition Idea

An old proverb suggests that the best guess for tomorrow's weather is that it will be the same as today's.

Let us simplify each day's weather into the three states sunny, cloudy or rainy.

Even if we don't understand weather, we can make a simple transition model by recording the weather every day, and noticing the likelihood of each possible transition.

Transition Records

Here is a calendar of 25 days of weather, recorded as sunny, cloudy, or rainy:

C-S-S-C-R-R-R-C-R-C-R-C-R-S-S-S-C-R-R-S-S-C-C-S-R

This gives us 24 daily transitions, and we can tabulate their frequencies:

S->S = 4	S->C = 3	S->R = 1
C->S = 2	C->C = 1	C->R = 5
R->S = 2	R->C = 3	R->R = 3

Transition Probabilities

From these daily transition frequencies:

S->S = 4 S->C = 3 S->R = 1 8 total transitions of S to something

C->S = 2 C->C = 1 C->R = 5 8 total transitions of C to something

R->S = 2 R->C = 3 R->R = 3 8 total transitions of R to something

we can give us a 3x3 matrix of transition probabilities:

S->S = 4/8 S->C = 3/8 S->R = 1/8

C->S = 2/8 C->C = 1/8 C->R = 5/8

R->S = 2/8 R->C = 3/8 R->R = 3/8

Simulation with rand()

If yesterday was sunny, can we simulate what will happen today?

S->S = 4/8   S->C = 3/8   S->R = 1/8

Let P be the value of rand():

If yesterday was sunny then:

if P <= 4/8:   sunny today

elseif 4/8 < P <= 4/8 + 3/8:   cloudy today

elseif 4/8 + 3/8 < P: rainy today

Similarly, we can simulate the followup to a cloudy or rainy day.

Weather Model

```
function today = weather_today ( yesterday )
r = rand ( )
if ( yesterday == 'Sunny' )
   if ( r < 0.5 )
     today = 'Sunny';
   elseif ( r < 0.875 )
     today = 'Cloudy'
   else
     today = 'Rainy'
   end
elseif ( yesterday == 'Cloudy' )
  if ( r < 0.25 )
     today = 'Sunny';
   elseif ( r < 0.375 )
     today = 'Cloudy'
   else
     today = 'Rainy'
   end
elseif ( yesterday == 'Rainy' )
  if ( r < 0.25 )
    today = 'Sunny';
  elseif ( r < 0.625 )
    today = 'Cloudy'
  else
    today = 'Rainy'
  end
end
```

Simulating More Weather

If we believe our transition probabilities are reasonable, we can simulate more weather, by noting yesterday's weather and choosing today's weather based on the probabilities;

```
for i = 0 : 50
  if ( i == 0 )
    today = 'S';
  else
    today = weather_today ( today );
  end
  fprintf ( '%c', today );
end
fprintf ( '\n' );
```

SSCCRSCCRRRCSSRSCRRSSSRSCRRSSSCRRSSSSCRCSCRCRCSSSSS

Transition Matrix

The table of probabilities is known as the transition matrix. Entry (i,j) of the matrix records the probability that, if we were in state i previously, we are going to move to state j.

Replacing fractions with decimals, our weather transition matrix is:

		Today			
		S	C	R	Sum
	S:	0.500	0.375	0.125	1.000
Yesterday	C:	0.250	0.125	0.625	1.000
	R:	0.250	0.375	0.375	1.000

Every row sums to 1, because whatever happened yesterday, **something** must happen today!

Modeling Population Changes

For a while in the 1960's, the following statement was approximately true:

Every year, 30% of the population of California leaves the state, and
every year, 10% of the population of the other states moves to California.

1) The statement sounds nonsensical. Can we write down some equations that give us numbers we can think about?

2) If 30% move out, and 10% move it, does this mean California is gradually going to have no population at all?

3) If this behavior lasts long enough, does the population curve of California look chaotic, go towards infinity, become negative, or oscillatory, or does it settle down?

One Person's Behavior

Suppose we model one person's behavior during this time, and assume that in 1960 they are living in California. Then there's a 30% chance they move out in 1961.

In 1962, if they are still in California, there's a 30% chance they move out then; but if they are outside of California, there's a 10% chance they move in.

We could simulate the location of such a person for 20 years if we wish.

One Person's History

```
m = 51;
for i = 1:m
  year = 1959 + i;
  if ( i == 1 )
    s = 'C';
  else
    r = rand ( );
    if ( s == 'C' )
      if ( r < 0.30 )
        s = 'U';
      end
    elseif ( s == 'U' )
      if ( r < 0.10 )
        s = 'C';
      end
    end
  end
end
```

CCCUUUUUUUUUUUUUUUUUCUUUUUUUCCCUUUUUUUUUCUUUUUUUUUU

Model EVERYBODY

Suppose that in 1960, California's population was 16 million, and the remaining US population was 164 million.

Then in 1961, our transition data suggests:

* 30% of 16 million people moved OUT of CA.

* 10% of 164 million people moved INTO CA.

In fact, we can track the CA and US populations from year to year, if we believe our model.

california.m

```
m = 21;
ca = zeros ( 1, m );
us = zeros ( 1, m );

for i = 1 : m
  year = 1959 + i;

  if ( year == 1960 )
    ca(i) =  16000000;
    us(i) = 164000000;
  else
    [ ca(i), us(i) ] = california_update ( ca(i-1), us(i-1) );
  end

end
```

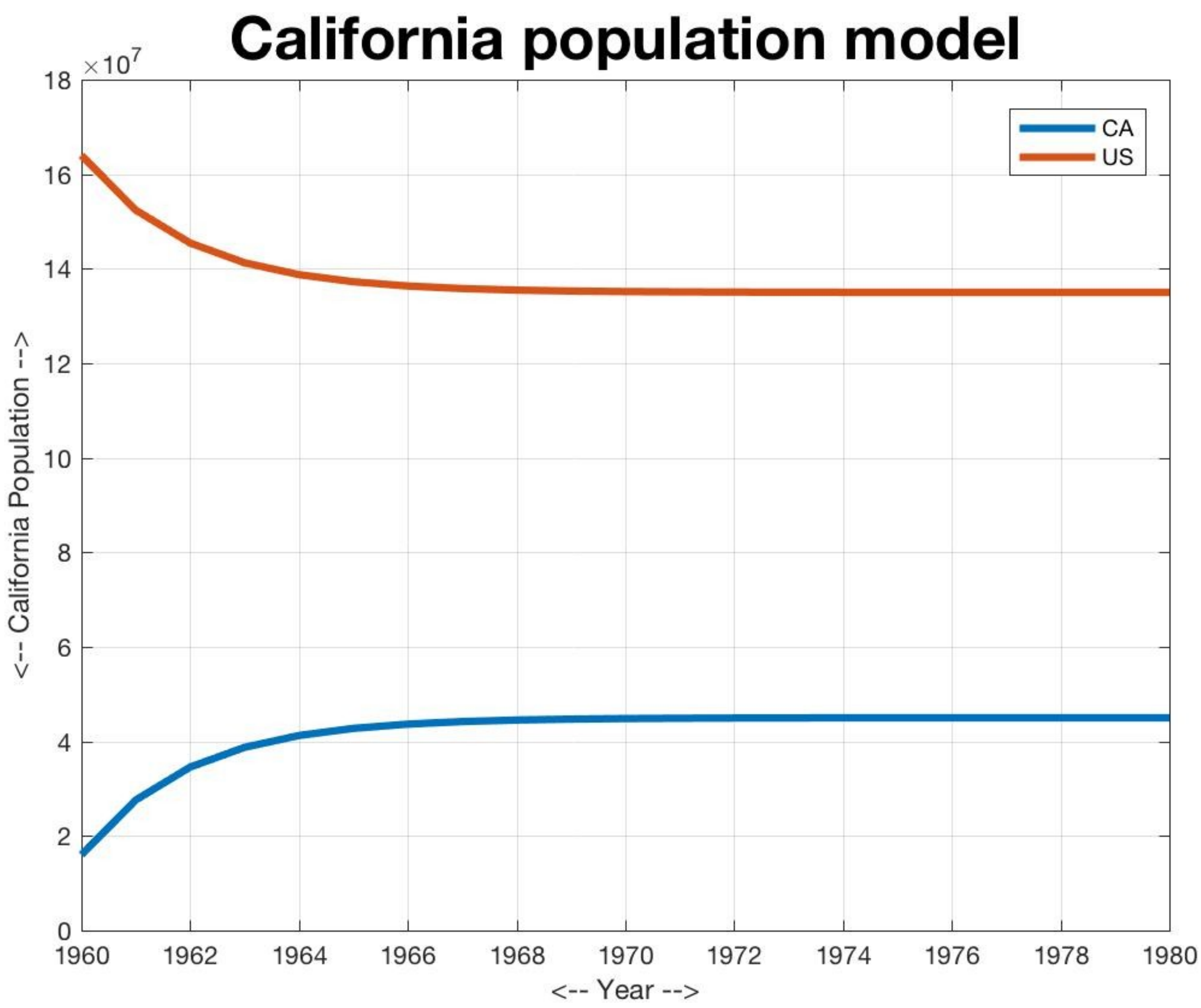
california_update.m

```
function [ ca, us ] = california_update ( ca, us )

  ca_old = ca;
  us_old = us;

  us = us_old - 0.10 * us_old + 0.30 * ca_old;
  ca = ca_old + 0.10 * us_old - 0.30 * ca_old;

  return
end
```

1960 to 1970

Year	CA Pop	US pop	Total
1960	16000000	164000000	180000000
1961	27600000	152400000	180000000
1962	34560000	145440000	180000000
1963	38736000	141264000	180000000
1964	41241600	138758400	180000000
1965	42744960	137255040	180000000
1966	43646976	136353024	180000000
1967	44188186	135811814	180000000
1968	44512911	135487089	180000000
1969	44707747	135292253	180000000
1970	44824648	135175352	180000000

Reaching an Equilibrium?



There is a Natural Balance Point

The population data seems to be driving towards values of 45 million for California and 135 million for the rest of the US.

If we plug these values into our formula:

```
us = us_old - 0.10 * us_old + 0.30 * ca_old;
ca = ca_old + 0.10 * us_old - 0.30 * ca_old;
```

the new values are the same as the old:

135m = 135m – 0.10 * 135m + 0.30 * 45m

45m = 45m + 0.10 * 135m – 0.30 * 45m

Case #2: Hawaii

Our textbook considers a more complicated example involving four Hawaiian islands, Oahu, Kauai, Maui and Lanai.

Suppose in the year 2000 each island had one million inhabitants, but we had the following transition data for moving from one island to another:

	OA	KA	MA	LA	(sum)
OA	0.32	0.18	0.27	0.23	1.00
KA	0.17	0.43	0.22	0.18	1.00
MA	0.11	0.32	0.39	0.18	1.00
LA	0.46	0.33	0.14	0.07	1.00

What the Matrix Means

Row I of that matrix describes all the places a person on island I can go to, with a probability. Thus, 32% of the Oahu residents stay on Oahu, 17% move to Kauai, and so on.

Column J of the matrix describes all the places a person moving to island J can come from, with a probability. Thus, next year's population of Lanai will be 23% of Oahu's population, 18% of Kauai's population, and so on.

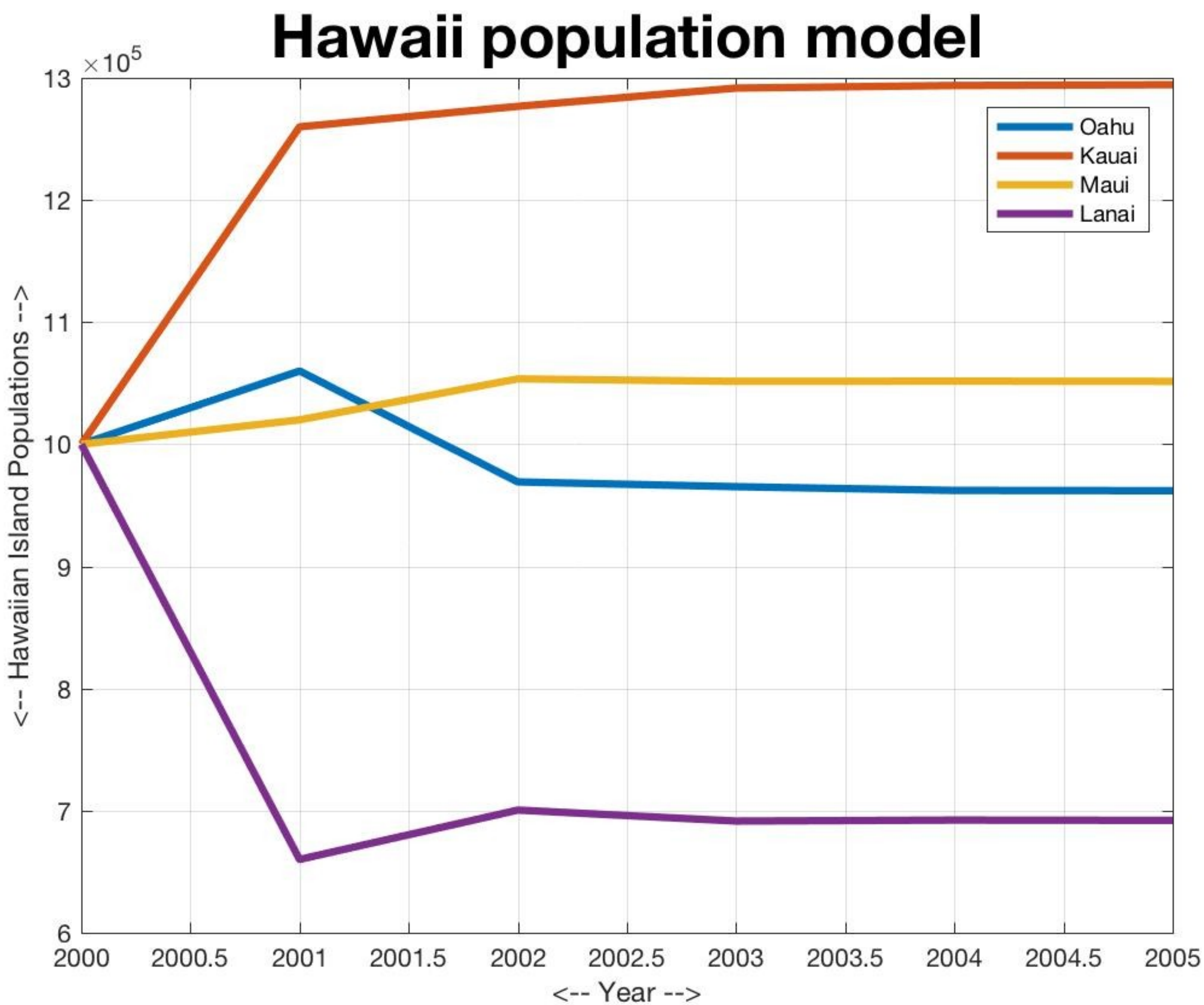
hawaii_update.m

```
function [ oa, ka, ma, la ] = hawaii_update ( oa, ka, ma, la )

  oa_old = oa;
  ka_old = ka;
  ma_old = ma;
  la_old = la;

  oa = 0.32 * oa_old + 0.17 * ka_old + 0.11 * ma_old + 0.46 * la_old;
  ka = 0.18 * oa_old + 0.43 * ka_old + 0.32 * ma_old + 0.33 * la_old;
  ma = 0.27 * oa_old + 0.22 * ka_old + 0.39 * ma_old + 0.14 * la_old;
  la = 0.23 * oa_old + 0.18 * ka_old + 0.18 * ma_old + 0.07 * la_old;

  return
end
```

Quick Changes, Then Settling

Year	Oahu	Kauai	Maui	Lanai	Total
2000	1000000	1000000	1000000	1000000	4000000
2001	1060000	1260000	1020000	660000	4000000
2002	969200	1276800	1053600	700400	4000000
2003	965280	1291764	1051540	691416	4000000
2004	962210	1293869	1051713	692208	4000000
2005	961969	1294538	1051525	691968	4000000

Population Settles Down



The Transition Matrix

The calculations we have done with the transition matrix are actually a tiny taste of linear algebra.

For the Hawaii population problem, if we call the old population xold and next year's population xnew, and we call the transition matrix A, then in linear algebra, it makes sense to say

xnew = A * xold

MATLAB will let us set up calculations this way; we will soon see some examples of how to do this.

Homework #10

hw050: Create a contour plot of a function that represents a "valley".

hw051: Create a surface plot of a function that exhibits four deep depressions.

hw052: Use the contour() function to draw a family of ellipses.

Homework #10 is due by midnight, Friday November 17th.

Homework #9 is due by tomorrow night.