

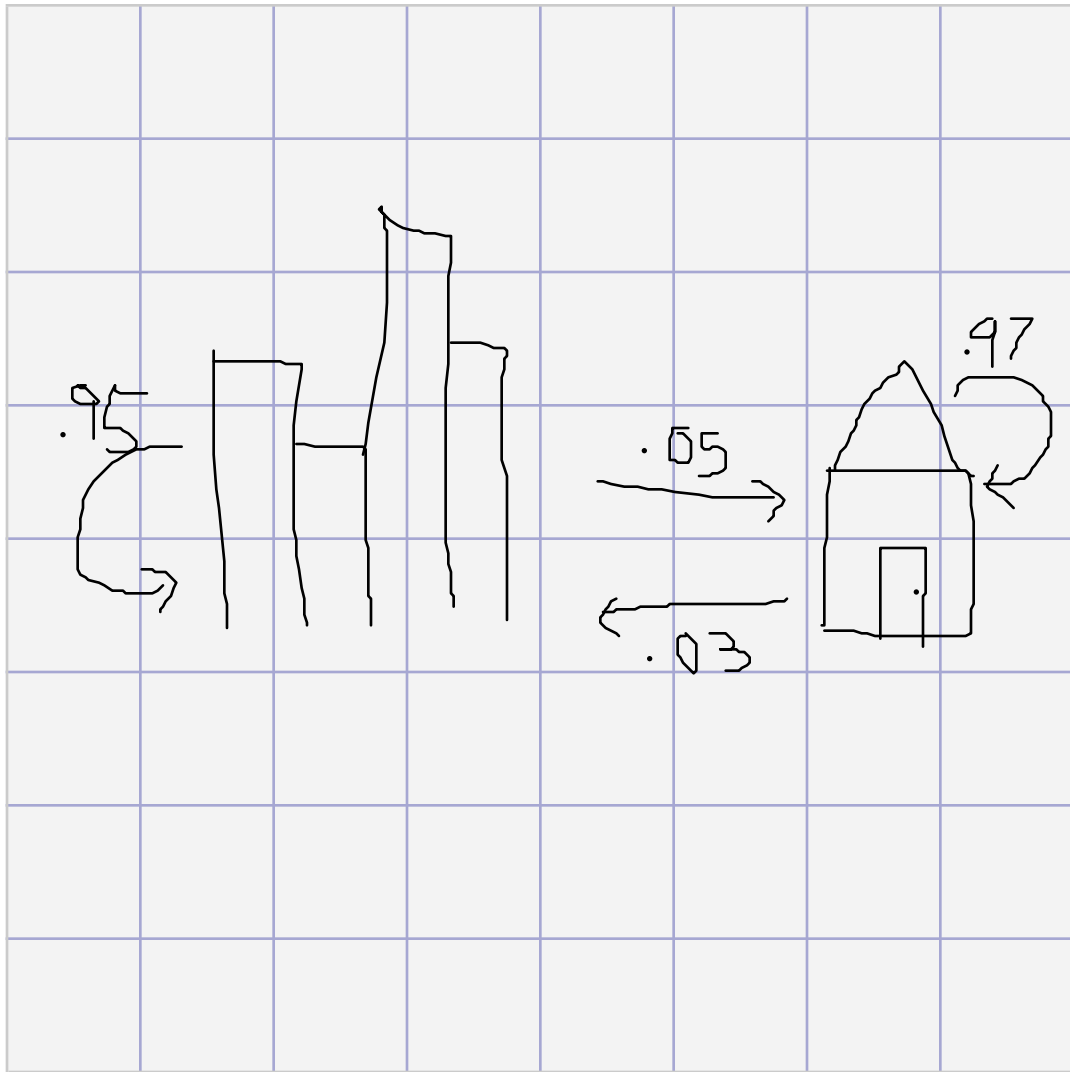
Markov Chains

by Leah W Berman, Christian Hellings, Lynne L Doty

```
> restart: with(LinearAlgebra): with(plots): with(plottools):  
Warning, the name changecoords has been redefined  
Warning, the assigned name arrow now has a global binding
```

▼ A small population example

[>] Suppose the movement of a population between a city and its suburbs may be modelled by the following: each year, 5% of the population moves from the city to the suburbs (and 95% of the city population stays in the city) and 3% of the population moves from the suburbs to the city (with the remaining 97% of the suburban population staying in the suburbs). The situation can be modelled with a diagram:



or a table:

[>

	From City	From Suburbs
To City	.95	.03
To Suburbs	.05	.97

[>

We can construct a matrix with the above data; this is called a **stochastic matrix**.
(Formally, a stochastic matrix is one in which the entries in each column sum to 1.)

[> $A := \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix};$

$$A := \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}$$

(1.1)

```
[>
```

Suppose in the year 2005, 60% of people live in the city and 40% of people live in the suburbs. We can make a vector representing this population distribution:

```
> b := <.6, .4 >;
```

$$b := \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} \quad (1.2)$$

```
[>
```

We can use this to estimate the population distribution in 2006:

```
> b1 := A.b;
```

$$b1 := \begin{bmatrix} 0.58199999999999962 \\ 0.41799999999999982 \end{bmatrix} \quad (1.3)$$

```
[>
```

So in the year 2006, we expect approximately 58% of people to live in the city and 42% to live in the suburbs.

What happens to the population over time: do most people stay in the city? Do most people move to the suburbs? One way is to calculate a sequence of vectors:

```
> b2 := map(x → evalf(x, 4), A.b1);
```

$$b2 := \begin{bmatrix} 0.5654 \\ 0.4346 \end{bmatrix} \quad (1.4)$$

```
> b3 := A.b2;
```

$$b3 := \begin{bmatrix} 0.55016799999999990 \\ 0.44983199999999952 \end{bmatrix} \quad (1.5)$$

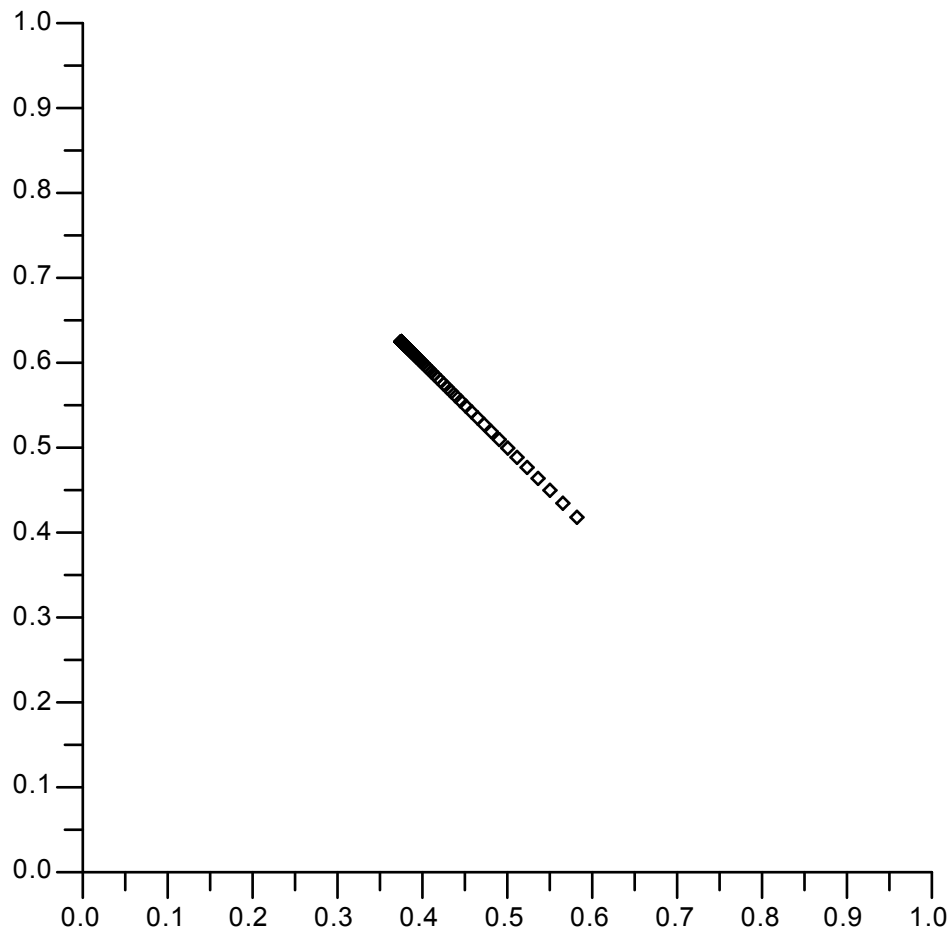
```
[>
```

Notice that $b2 = (A.b1) = A.(A.b) = A^2.b$ and $b3 = A.(b2) = A.(A^2.b) = A^3.b$. In general, $b_k = A^k.b$. So if we want to look at where people live a long time in the future, we could simply multiply by some appropriate power of A .

The sequence of vectors $b, b1 = A.b, b2 = A^2.b, b3 = A^3.b, \dots$ is called a **Markov chain**.

Let's look at what happens over the next hundred years: the collection **li** is the list of population distributions from 2006 to 2106.

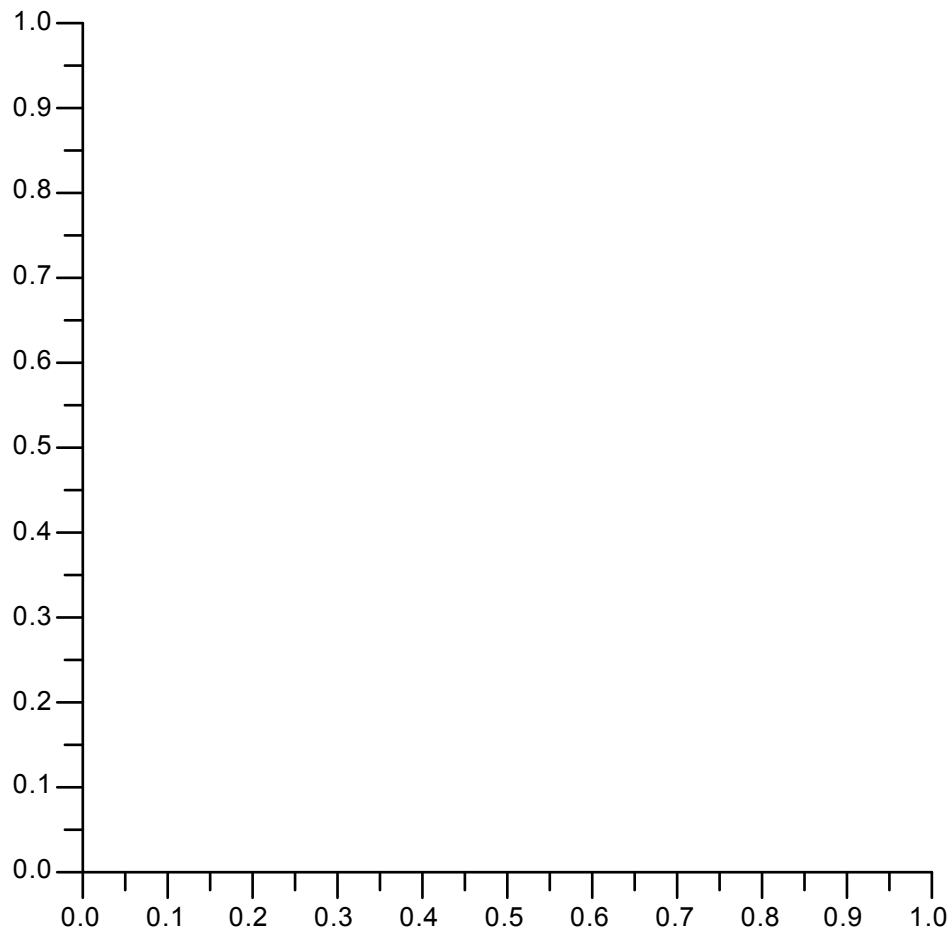
```
> li := [seq(A^k.b, k=1..100)];  
pointplot(li, color=blue, view=[0..1, 0..1]);
```



We can see the evolution over time in the following animation.

```
> p:= [seq( pointplot( [seq(li[i], i=1..5*t)], color=blue, view=[0.  
.1,0..1] ), t=0..20 )]:  
display( p, insequence=true );
```

Click on the plot and use the VCR controls in the toolbar. Setting the frames per second to 1 may be helpful.



>

It looks like the population is converging to some value! We can see what the population is in 2106:

> **A^(100).b;**

$$\begin{bmatrix} 0.375053822671795989 \\ 0.624946177328199015 \end{bmatrix}$$

(1.6)

>

So, approximately 37.5% of the people live in the city and 62.5% live in the suburbs in 2106.

We can look farther into the future: this time we'll look in the future by hundred year jumps.

> **li2:={seq(A^(100*k).b, k=1..5)};**

$$li2 := \left\{ \begin{bmatrix} 0.375000012875018551 \\ 0.624999987124971623 \end{bmatrix}, \begin{bmatrix} 0.375000000003074263 \\ 0.624999999996910915 \end{bmatrix} \right\}$$

(1.7)

$$\begin{bmatrix} 0.375053822671795989 \\ 0.624946177328199015 \end{bmatrix}, \begin{bmatrix} 0.37499999999993338 \\ 0.624999999999986899 \end{bmatrix}, \begin{bmatrix} 0.37499999999990729 \\ 0.624999999999984457 \end{bmatrix}$$

>

And by thousands:

$$li3 := \left\{ \begin{bmatrix} 0.37499999999925726 \\ 0.624999999999876210 \end{bmatrix}, \begin{bmatrix} 0.37499999999907130 \\ 0.624999999999845235 \end{bmatrix}, \begin{bmatrix} 0.37499999999962863 \\ 0.62499999999938049 \end{bmatrix}, \begin{bmatrix} 0.37499999999981459 \\ 0.62499999999969025 \end{bmatrix}, \begin{bmatrix} 0.37499999999944267 \\ 0.62499999999907185 \end{bmatrix} \right\} \quad (1.8)$$

>

It looks like the lists of vectors are converging---in the far, far future, about 37.5% will live in the city and 62.5% will live in the suburbs.

How does the initial distribution of the population (60% city, 40% suburbs) affect the long-range distribution of people in this area? We can start with 30% of the people living in the city and 70% in the suburbs.

$$\begin{aligned} > \mathbf{b} := \langle .3, .7 \rangle; \\ & \mathbf{b} := \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} \end{aligned} \quad (1.9)$$

$$\begin{aligned} > \mathbf{b1} := \mathbf{A}.\mathbf{b}; \\ & \mathbf{b1} := \begin{bmatrix} 0.30599999999999994 \\ 0.69399999999999950 \end{bmatrix} \end{aligned} \quad (1.10)$$

$$\begin{aligned} > \mathbf{b2} := \mathbf{A}.\mathbf{b1}; \\ & \mathbf{b2} := \begin{bmatrix} 0.31151999999999964 \\ 0.68847999999999980 \end{bmatrix} \end{aligned} \quad (1.11)$$

Let's skip ahead to 100 years from now:

$$\begin{aligned} > \mathbf{A}^{(100)}.\mathbf{b}; \\ & \begin{bmatrix} 0.374982059109398857 \\ 0.625017940890596035 \end{bmatrix} \end{aligned} \quad (1.12)$$

It looks like the same long-term distribution as before: 37.5% in the city, 62.5% in suburbs. So at least for this example, the initial distribution of folks doesn't seem to affect the long-term one. Is this always true? More later.

```
> bb:=<1,0>;  
A^(1000).bb;
```

$$bb := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$\begin{bmatrix} 0.3749999999999981459 \\ 0.62499999999999969025 \end{bmatrix} \quad (1.13)$$

Exercises

Suppose that in a different city, it was found that each year, 12% of the population moves from the city to the suburbs (and 88% of the city population stays in the city) and 4% of the population moves from the suburbs to the city (with the remaining 96% of the suburban population staying in the suburbs).

```
[>
```

1. What is the stochastic matrix representing the movements to and from the city and the suburbs?

```
[>
```

```
[>
```

2. If in 2005, 82% of the population lived in the city and 18% live in the suburbs, what will the population distribution be in 2006? What about 2008?

```
[>
```

```
[>
```

```
[>
```

3. In the far future, what do you think the population distribution will be?

```
[>
```

```
[>
```

Do you think the population distribution in the future depends on the initial distribution? Choose a random distribution (but make sure the total population percentages add up to 100%) and guess what the population distribution will be in the far future.

```
[>
```

```
[>
```

```
[>
```

A more complicated example

For this example we assume the population moves among three locations: City, Suburbs, Rural. Here's the table that indicates the movement of people from location to location.

--	--	--	--

	From City	From Suburbs	From Rural
To City	.96	.01	.015
To Suburbs	.03	.98	.005
To Rural	.01	.01	.980

Here's the stochastic matrix for this changing population.

```
> P := Matrix(<< 0.96 | 0.01 | 0.015 >,
              < 0.03 | 0.98 | 0.005 >,
              < 0.01 | 0.01 | 0.98 >>);
```

$$P := \begin{bmatrix} 0.96 & 0.01 & 0.015 \\ 0.03 & 0.98 & 0.005 \\ 0.01 & 0.01 & 0.98 \end{bmatrix} \quad (2.1)$$

Notice here that a large proportion of each group stays in the same place--there's very little shifting to a new group.

For this example, we look at how the population evolves using the idea that for any initial distribution vector, b , the proportion of the population in each group after k years is the corresponding entry in the vector $b_k = P^k \cdot b$. We start with the entire population in the City. Then we look at the distribution after 100 years and after 200 years.

```
>
> b := Vector[column](< 1 , 0 , 0 >);
```

$$b := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.2)$$

After 100 years, the population distribution is

```
> evolutionb := [seq(P^i.b, i=1..200)]:
  evolutionb[100];
```

$$\begin{bmatrix} 0.234403074176832676 \\ 0.448114428464967406 \\ 0.317482497358197723 \end{bmatrix} \quad (2.3)$$

After 200 years, the population distribution is

```
> evolutionb [ 200 ];
```

$$(2.4)$$

$$\begin{bmatrix} 0.233174691348810886 \\ 0.434245722321185245 \\ 0.332579586330000732 \end{bmatrix} \quad (2.4)$$

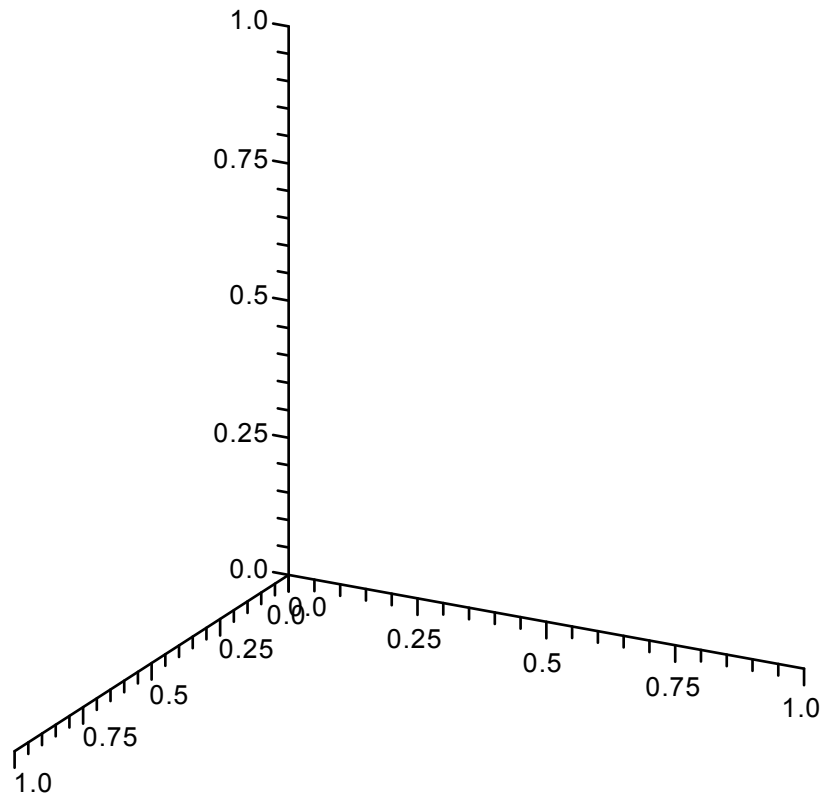
Now we'll look at this geometrically. We'll look at how the stochastic matrix moves two basis vectors.

```
> X := Vector[column](< 1 , 0 , 0 >);
```

$$X := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.5)$$

```
> evolutionX := [seq((VectorCalculus[DotProduct])(P^i, b), i = 1 .
. 100)]:
p1:= [seq( pointplot3d( [seq(evolutionb[i], i=1..5*t)], color=
blue, axes=normal, view=[0..1,0..1,0..1] ), t=0..20 )]:
display( p1, insequence=true );
```

Click on the plot and use the VCR controls in the toolbar.



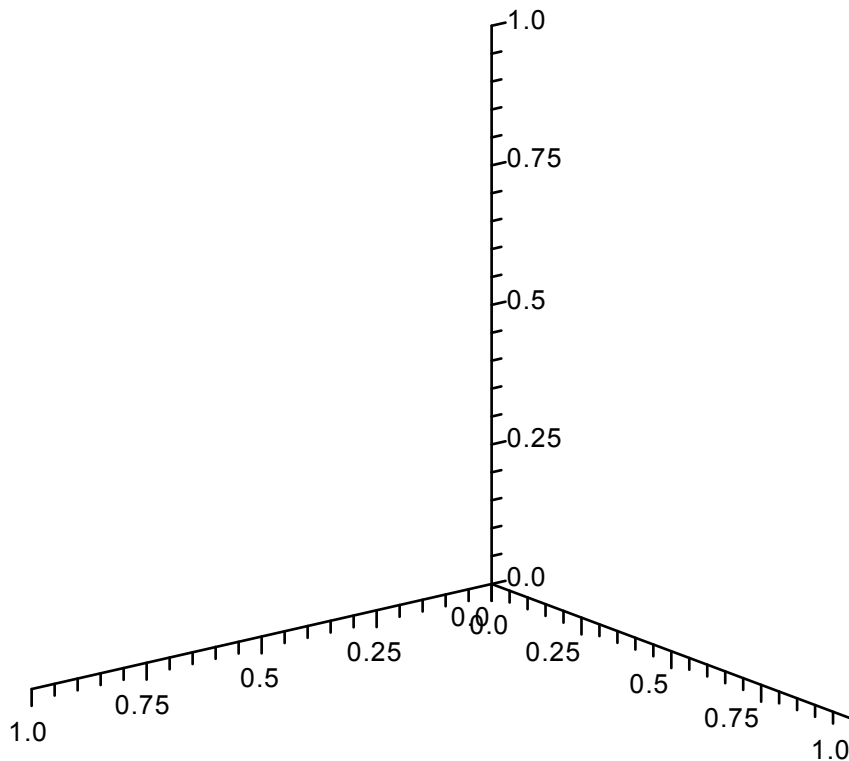
```
> Y := Vector[column](< 0, 1, 0 >);
```

$$Y := \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

(2.6)

```
> evolutionY := [seq(P^i.Y, i=1..100)]:
p2:= [seq( pointplot3d( [seq(evolutionY[i], i=1..5*t)], color=
green, axes=normal, view=[0..1,0..1,0..1] ), t=0..20)]:
p := [seq(display(p1[t],p2[t]), t=1..21)]:
display( p, insequence=true );
```

Click on the plot and use the VCR controls in the toolbar.



So it looks like the matrix moves both initial points to the same point--eventually. Now we'll look at the eventual location of all three basis vectors, as well as a random vector.

```
> Z := Vector[column](< 0, 0, 1 >);
   evolutionZ := [seq(P^i.Z, i=1..200)]:
```

$$Z := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (2.7)$$

```
> V := RandomVector(3, generator=rand(0..99));
   V := evalf(V/add(V[i], i=1..3));
   evolutionV := [seq(P^i.V, i=1..100)]:
```

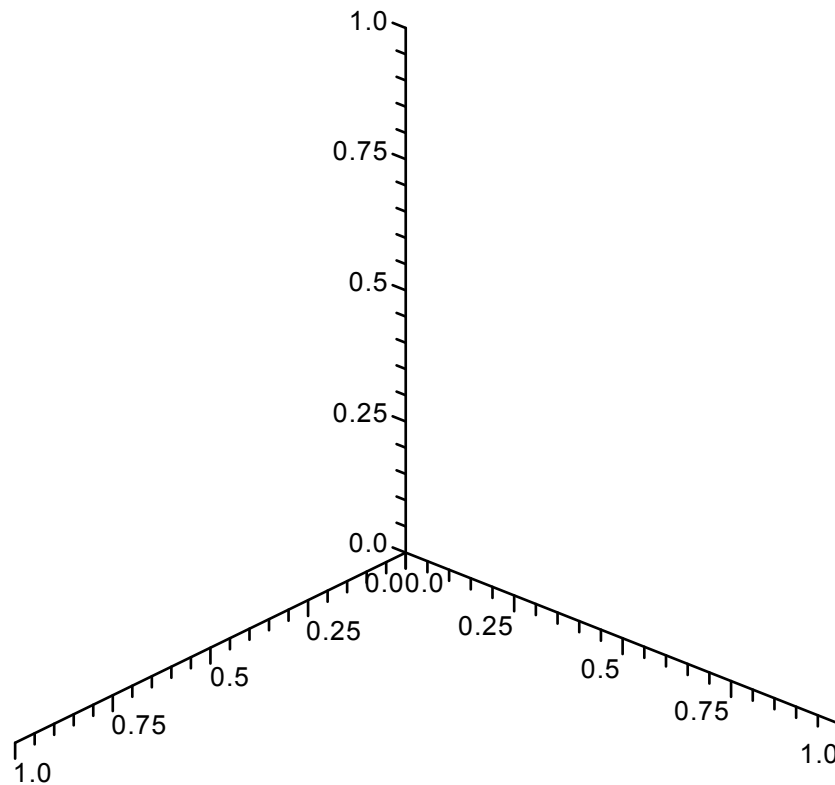
$$V := \begin{bmatrix} 0.3982683983 \\ 0.1904761905 \\ 0.4112554113 \end{bmatrix} \quad (2.8)$$

```
> p3 := [seq( pointplot3d( [seq(evolutionZ[i], i=1..5*t)], color=
   red, axes=normal, view=[0..1, 0..1, 0..1] ), t=0..20)]:
```

```

p4:= [seq( pointplot3d( [seq(evolutionV[i], i=1..5*t)], color=
violet, axes=normal, view=[0..1,0..1,0..1] ), t=0..20)]:
p := [seq(display(p[t],p3[t],p4[t]), t=1..21)]:
display( p, insequence=true );

```



Once again it appears that the initial distribution of population among the three locations has no effect on the eventual distribution. To get an idea of what the actual percentages are we need to do some algebra, and that's in the next section.

>

If the animations are slowing Maple down, here are the three plots from above in static form. (Just change the colons to semicolons after each line.)

```

> pointplot3d(evolutionX,view=[0..1,0..1,0..1],
axes=normal,color=blue):
display({pointplot3d(evolutionX,color=blue),pointplot3d
(evolutionY,color=green)},view=[-1..1,-1..1,-1..1], axes=normal)
:
display({pointplot3d(evolutionX,color=blue),pointplot3d
(evolutionY,color=green),pointplot3d(evolutionZ,color=red),
pointplot3d(evolutionV,color=violet)},view=[-1..1,-1..1,-1..1],
axes=normal):

```

▼ Where do the vectors converge to?

In the previous sections, it looked like eventually, all the values converge to the same vector. That is,

$\lim_{n \rightarrow \infty} A^n \cdot b$ exists and equals some vector q . In fact, we would like to find a vector q (in the far future, or "at the limit") so that nothing changes — i.e., $A \cdot q = q$. Such a vector is called a **steady — state** vector. We would like to be able to determine the value of q , given A .

▼ The small example

Recall our original example, where each year, 5% of the population moves from the city to the suburbs (and 95% of the city population stays in the city) and 3% of the population moves from the suburbs to the city (with the remaining 97% of the suburban population staying in the suburbs). We can write the matrix using fractions instead of decimals to try to avoid round-off errors in our later calculations.

```
> A := <<95/100, 5/100>|<3/100, 97/100>>;
```

$$A := \begin{bmatrix} \frac{19}{20} & \frac{3}{100} \\ \frac{1}{20} & \frac{97}{100} \end{bmatrix} \quad (3.1.1)$$

We want a vector q so that $A \cdot q = q$. This should look familiar: this is just saying that we want an eigenvector associated with A , with eigenvalue 1. Only one of the eigenvectors makes sense as a population distribution, so we'll record it as q and its associated eigenvalue as λ . (Note $\lambda=1$.)

```
> eigs:=Eigenvectors(A, output=list);
```

$$eigs := \left[\left[1, 1, \left\{ \begin{bmatrix} 3 \\ 5 \end{bmatrix} \right\} \right], \left[\frac{23}{25}, 1, \left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\} \right] \right] \quad (3.1.2)$$

Sometimes the list of eigenvalues and eigenvectors comes out in a different order. I don't know why. If you're not getting the eigenvalue =1, replace the following code with

```
#q:=eigs[2,3,1];  
#lambda:=eigs[2,1];
```

after removing the # from in front.

```
> q:=eigs[1,3,1];  
lambda:=eigs[1,1];
```

$$q := \begin{bmatrix} \frac{3}{5} \\ 1 \end{bmatrix}$$

$$\lambda := 1 \tag{3.1.3}$$

>

However, this eigenvector that we found doesn't correspond to a population distribution. We can fix that by making the entries in the vector add up to 1, by dividing the vector by the sum of the entries:

> `normq:=q/add(q[i], i=1..(nops(q)-1));`

$$normq := \begin{bmatrix} \frac{3}{8} \\ \frac{5}{8} \end{bmatrix} \tag{3.1.4}$$

We can check that this vector is a steady-state vector:

> `A.normq;`

$$\begin{bmatrix} \frac{3}{8} \\ \frac{5}{8} \end{bmatrix} \tag{3.1.5}$$

>

There was another eigenvector for A, however, which didn't make sense as a population distribution. We record it and its associated eigenvalue.

> `q2:=eigs[2,3,1];`
`lambda2:=eigs[2,1];`

$$q2 := \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\lambda2 := \frac{23}{25} \tag{3.1.6}$$

>

It doesn't make sense when considered as a population distribution, but it still is useful to us.

We know that {q, q2} forms a basis for the eigenspace of A, and in fact, it forms a basis for \mathbb{R}^2 . We can check this, of course, by row-reducing:

> `ReducedRowEchelonForm(<q|q2>);`

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.1.7}$$

If we start with an arbitrary vector x0, we can write it as a linear combination of the vectors q and q2, say

$x_0 = s(q) + t(q_2)$. We'll check with the original population distribution:

> `bb:=<6/10, 4/10>;`

$$bb := \begin{bmatrix} \frac{3}{5} \\ \frac{2}{5} \end{bmatrix} \quad (3.1.8)$$

> `ans:=ReducedRowEchelonForm(<q|q2|bb>);`

$$ans := \begin{bmatrix} 1 & 0 & \frac{5}{8} \\ 0 & 1 & \frac{-9}{40} \end{bmatrix} \quad (3.1.9)$$

Recall that the entries in the last column are the coefficients of the linear combination of q and $q2$ that equal bb , which we can check.

> `s:=Column(ans,3)[1];`
`t:=Column(ans,3)[2];`
`s*q + t*q2;`

$$s := \frac{5}{8}$$

$$t := \frac{-9}{40}$$

$$\begin{bmatrix} \frac{3}{5} \\ \frac{2}{5} \end{bmatrix} \quad (3.1.10)$$

So, to find the next vector in the sequence, $x1$, if we call the coefficients of the linear combination we would compute

$$x1 = A.x0$$

$$x1 = A.(s(q) + t(q2))$$

$$x1 = sA.(q) + t(A.q2)$$

But q and $q2$ are eigenvectors, so we know what $A.q$ and $A.q2$ equal! With a little reorganization, we get

$$x1 = \lambda(sq) + (\lambda2)(t(q2)).$$

We can check with our example:

> `A.bb;`
> `lambda*s*q + lambda2*t*q2;`

We can repeat the argument we made for $x1$ to say something about $x2$:

$$x2 = A.(\lambda(sq) + (\lambda2)(t(q2)))$$

$$x2 = \lambda s(Aq) + (\lambda2)t(A.(q2))$$

and using the fact, again, that q and $q2$ are eigenvectors, we see that

$$x2 = (\lambda^2)sq + (\lambda2)^2(t)(q2)$$

In fact, this generalizes to any vector in the sequence:

$$xk = (\lambda^k)sq + (\lambda2)^k(t)(q2).$$

We can check this (converting to decimals so it's easier to compare the vectors):

```
> evalf((A^10).bb);
evalf(lambda^(10)*s*q + lambda2^(10)*t*q2);
```

$$\begin{bmatrix} 0.4727374022 \\ 0.5272625978 \\ 0.4727374022 \\ 0.5272625978 \end{bmatrix} \quad (3.1.11)$$

Since λ_2 is less than 1,

```
> lambda2;
```

$$\frac{23}{25} \quad (3.1.12)$$

as $k \rightarrow \infty$, $(\lambda_2)^k \rightarrow 0$. Therefore, no matter what x_0 we start with, as $k \rightarrow \infty$, $x_k \rightarrow (\lambda_1)^k s \mathbf{q}$. We can check this by choosing a random vector \mathbf{randb} and looking at the difference between $A^{200} \cdot \mathbf{randb}$ and $(\lambda_1)^k s \mathbf{q}$. Of course, since we're starting with a different x_0 , there will be different values for the linear combination of the basis eigenvectors that we will need to use (i.e., s will be different).

Here's the random vector.

```
> r:= RandomVector(2,generator=rand(0..10));
randb := r/add(r[i],i=1..2);
```

$$\mathbf{randb} := \begin{bmatrix} \frac{5}{6} \\ \frac{1}{6} \end{bmatrix} \quad (3.1.13)$$

...and we compute \mathbf{randb} as a linear combination of \mathbf{q} and \mathbf{q}_2 .

```
> ans:=ReducedRowEchelonForm(<q|q2|randb>);
```

$$\mathbf{ans} := \begin{bmatrix} 1 & 0 & \frac{5}{8} \\ 0 & 1 & \frac{-11}{24} \end{bmatrix} \quad (3.1.14)$$

```
> srand:=Column(ans,3)[1];
trand:=Column(ans,3)[2];
srand*q +trand*q2;
```

$$\begin{aligned} \mathbf{srand} &:= \frac{5}{8} \\ \mathbf{trand} &:= \frac{-11}{24} \end{aligned} \quad (3.1.15)$$

$$\begin{bmatrix} \frac{5}{6} \\ \frac{1}{6} \end{bmatrix} \quad (3.1.15)$$

Check that things work again

```
> evalf(A^(200).randb);
evalf(lambda^(200)*srand*q + lambda2^(200)*trand*q2);
```

$$\begin{bmatrix} 0.3750000262 \\ 0.6249999738 \\ 0.3750000262 \\ 0.6249999738 \end{bmatrix} \quad (3.1.16)$$

```
> srand·q;
```

$$\begin{bmatrix} \frac{3}{8} \\ \frac{5}{8} \end{bmatrix} \quad (3.1.17)$$

And compute the difference between $A^{(200)}.\mathbf{randb}$ and $\lambda\mathbf{sq}$ for our \mathbf{srand} .

```
> evalf(A^(200).randb)-lambda*srand*q;
```

$$\begin{bmatrix} 2.62000000028628222 \cdot 10^{-8} \\ -2.61999999473516709 \cdot 10^{-8} \end{bmatrix} \quad (3.1.18)$$

The difference is very small! If we try it for a bigger exponent, we go beyond Maple's level of precision:

```
> evalf(A^(1000).randb)-lambda*srand*q;
```

$$\begin{bmatrix} 0. \\ 0. \end{bmatrix} \quad (3.1.19)$$

(HERE'S SOME STUFF THAT I'VE SEEN REFERENCES TO BUT I CAN'T FIND A PRECISE REFERENCE AT THE MOMENT) Notice that this result depended on two things: (1) $\lambda = 1$, and (2) $\lambda_2 < 1$. A theorem which is beyond the scope of this class (the *Stochastic Matrix Theorem?* *Perron-Fröbenius theorem?*) says that if your stochastic matrix A is sufficiently nice (some power of A has all positive entries), then for every eigenvalue λ_i , $|\lambda_i| \leq 1$ and there exists an eigenvalue $\lambda = 1$. So in such a case, a modification of the above argument would show that if you start with a sufficiently nice stochastic matrix, with eigenvalue $\lambda = 1$ and corresponding eigenvector \mathbf{q} and the rest of the eigenvalues strictly less than one in absolute value, given any starting vector \mathbf{x}_0 , that as $k \rightarrow \infty$, $\mathbf{x}_k \rightarrow s\mathbf{q}$, where s is the coefficient of \mathbf{q} when you write \mathbf{x}_0 as a linear combination of the eigenvalues of A .

▼ Exercises

4. Find the steady-state vector for the situation given in example 1. Check that it works.

>
>

5. Not all stochastic matrices will lead to a steady state. Investigate the long term behavior of the matrix W defined here: (This may belong in another location, but I'll put it here for now.)

$$W := \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & .5 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & .5 & 0 \end{bmatrix}$$

>

The large example

We can play the same game with the large example. I'll rename P and write it with rational numbers to avoid roundoff errors:

```
> PP := Matrix(<< 96/100 | 1/100 | 015/1000 >,
               < 3/100 | 98/100 | 005/1000 >,
               < 1/100 | 1/100 | 98/100 >>);
```

$$PP := \begin{bmatrix} \frac{24}{25} & \frac{1}{100} & \frac{3}{200} \\ \frac{3}{100} & \frac{49}{50} & \frac{1}{200} \\ \frac{1}{100} & \frac{1}{100} & \frac{49}{50} \end{bmatrix} \quad (3.2.1)$$

>

We compute the eigenvalues and eigenvectors, and extract the eigenvector associated with the eigenvalue 1.

```
> peigs := Eigenvectors(PP, output=list);
```

$$peigs := \left[\left[\frac{19}{20}, 1, \left\{ \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \right\} \right], \left[1, 1, \left\{ \begin{bmatrix} \frac{7}{10} \\ \frac{13}{10} \\ 1 \end{bmatrix} \right\} \right], \left[\frac{97}{100}, 1, \left\{ \begin{bmatrix} \frac{1}{4} \\ -\frac{5}{4} \\ 1 \end{bmatrix} \right\} \right] \right] \quad (3.2.2)$$

```
> qp := peigs[2,3,1];
```

$$qp := \begin{bmatrix} \frac{7}{10} \\ \frac{13}{10} \\ 1 \end{bmatrix} \quad (3.2.3)$$

```
> newqp:=qp/add(qp[i], i=1..3);
add(newqp[i], i=1..3);
```

$$newqp := \begin{bmatrix} \frac{7}{30} \\ \frac{13}{30} \\ \frac{1}{3} \end{bmatrix} \quad (3.2.4)$$

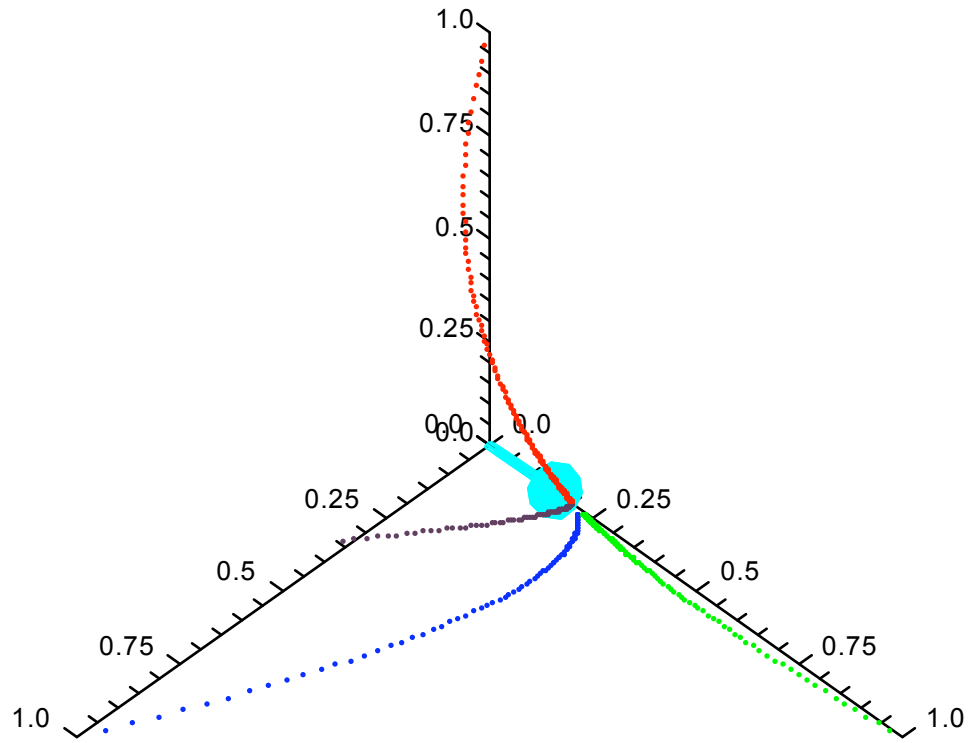
We can plot, as before, the chains, and this time we will also plot the eigenvector. (Be patient; this takes a little while.)

```
> N:=100:
X := Vector[column](< 1 , 0 , 0 >):
Y := Vector[column](< 0, 1 , 0 >):
Z := Vector[column](< 0, 0 , 1 >):
evolutionX := [seq(P^i.X, i=1..N)]:
evolutionY := [seq(P^i.Y, i=1..N)]:
evolutionZ := [seq(P^i.Z, i=1..2*N)]:
```

```
> rV := RandomVector(3, generator=rand(0..10)):
V := evalf(rV/add(rV[i], i=1..3));
evolutionV := [seq(P^i.V, i=1..N)]:
```

$$V := \begin{bmatrix} 0.5555555556 \\ 0.1666666667 \\ 0.2777777778 \end{bmatrix} \quad (3.2.5)$$

```
> display({
pointplot3d(evolutionX,color=blue),
pointplot3d(evolutionY,color=green),
pointplot3d(evolutionZ,color=red),
pointplot3d(evolutionV,color=violet),
arrow(<0,0,0>,newqp,.02, .1, .2,cylindrical_arrow,color=cyan,
thickness=5)},
view=[0..1,0..1,0..1], axes=normal);
```



Notice that the cyan arrow, the eigenvector associated with eigenvalue 1, adjusted so the sum of its entries is also 1, points directly at the point of convergence of the four markov chains!

>