Markov Chains

```
<IPython.core.display.HTML object>
```

Andrei Markov, 1856 - 1922, St Petersburg.

Markov was part of the great tradition of mathematics in Russia, which carried into and beyond the Soviet era. Among other pursuits, Markov pioneered the study of systems in which the future state of the system depends only on the present state in a random fashion. Classic examples include the movement of stock prices and the dynamics of animal populations.

These have since been termed "Markov Chains."

Many applications in computing are concerned with how a system behaves over time.

Think of a Web server that is processing requests for Web pages, or network that is moving packets from place to place.

We would like to describe how systems like these operate, and analyze them to understand their performance limits.

The way we model this is:

- we define some vector that describes the state of the system, and
- we formulate a rule that tells us how to compute the next state of the system based on the current state of the system.

So we would say that the state of the system at time k is a vector $\mathbf{x_k} \in \mathbb{R}^n$, and

$$\mathbf{x_{k+1}} = T(\mathbf{x_k}), \quad \text{for time } k = 0, 1, 2...$$

where $T : \mathbb{R}^n \to \mathbb{R}^n$.

This situation is so common that it goes by many names:

- In physics, this is called a *dynamical system*. (Here, $\mathbf{x_k}$ might represent the position and velocity of a set of particles.)
- When studying algorithms, this is called a *recurrence relation*. (Here, $\mathbf{x_k}$ might represent the number of steps needed to solve a problem of size k.)
- Most commonly, this is called a *difference equation*. The reason for this terminology is that it is a discrete analog of a differential equation in k.

The vector $\mathbf{x_k}$ is called the *state vector.*

Of course, we are going to be particularly interested in the case where T is a linear transformation. Then we know that we can write the difference equation as:

$$\mathbf{x_{k+1}} = A\mathbf{x_k},$$

where $A \in \mathbb{R}^{n \times n}$. This is a *linear difference equation.*

Example. In the homework, we studied the following system:

We are interested in the population of two regions, say the city and the suburbs. Fix an initial year (say 2000) and let

$$\mathbf{x_0} = \begin{bmatrix} \text{population of the city in 2000} \\ \text{population of the suburbs in 2000} \end{bmatrix}.$$

Then

$$\mathbf{x_1} = \begin{bmatrix} \text{population of the city in 2001} \\ \text{population of the suburbs in 2001} \end{bmatrix},$$

$$\mathbf{x_2} = \begin{bmatrix} \text{population of the city in 2002} \\ \text{population of the suburbs in 2002} \end{bmatrix}, \ldots$$

We only concern ourselves with movements of people between the two regions (no immigration, emigration, birth, death, etc.).

We assume that measurements have shown the following pattern: in any given year, 5% of the people in the city move to the suburbs, and 3% of the people in the suburbs move to the city.

You can think of this as:

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

Then we can capture this update rule as a matrix:

$$A = \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix}.$$

We can see that this is correct by verifying that:

$$\begin{bmatrix} \text{city pop. in 2001} \\ \text{suburb pop. in 2001} \end{bmatrix} = \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix} \begin{bmatrix} \text{city pop. in 2000} \\ \text{suburb pop. in 2000} \end{bmatrix}.$$

Markov Chains

Let's look at A again:

$$A = \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix}.$$

We note that A has a special property: each of its columns adds up to 1.

Also, it would not make sense to have negative entries in A.

This reflects the fact that the total number of people in the system is not changing over time. This leads to three definitions:

Definition. A *probability vector* is a vector of nonnegative entries that sums to 1.

Definition. A *stochastic* matrix is a square matrix of nonnegative values whose columns each sum to 1.

Definition. A *Markov chain* is a dynamical system whose state is a probability vector and which evolves according to a stochastic matrix.

That is, it is a probability vector $\mathbf{x_0}$ and a stochastic matrix $A \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{x_{k+1}} = A\mathbf{x_k} \quad \text{for } k = 0, 1, 2, ...$$

So we think of a probability vector $\mathbf{x_0}$ as describing how things are "distributed" across various categories – the fraction of items that are in each category.

And we think of the stochastic matrix A as describing how things "redistribute" themselves at each time step.

Question Time! Q11.1

Example. Suppose that in 2000 the population of the city is 600,000 and the population of the suburbs is 400,000. What is the distribution of the population in 2001? In 2002? In 2020?

Solution. First, we convert the population distribution to a probability vector. This is done by simply normalizing by the sum of the vector elements.

$$600,000 + 400,000 = 1,000,000.$$

$$\frac{1}{1,000,000}\begin{bmatrix} 600,000 \\ 400,000 \end{bmatrix} = \begin{bmatrix} 0.60 \\ 0.40 \end{bmatrix}.$$

Then the distribution of population in 2001 is:

$$\mathbf{x_1} = A\mathbf{x_0} = \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix}\begin{bmatrix} 0.60 \\ 0.40 \end{bmatrix} = \begin{bmatrix} 0.582 \\ 0.418 \end{bmatrix}.$$

And the distribution of the population in 2002 is:

$$\mathbf{x_2} = A\mathbf{x_1} = \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix}\begin{bmatrix} 0.582 \\ 0.418 \end{bmatrix} = \begin{bmatrix} 0.565 \\ 0.435 \end{bmatrix}.$$

Note that another way we could have written this is:

$$\mathbf{x_2} = A\mathbf{x_1} = A(A\mathbf{x_0}) = A^2\mathbf{x_0}.$$

To answer the question for 2020, i.e., $k = 20$, we note that

$$\mathbf{x_{20}} = \overbrace{A \cdots A}^{20}\mathbf{x_0} = A^{20}\mathbf{x_0}.$$

```
[12]: A = np.array(
          [[0.95,0.03],
           [0.05,0.97]])
      x0 = np.array([0.60,0.40])
      A20 = np.linalg.matrix_power(A,50)
      x20 = A20 @ x0
      print(x20)
```

```
[ 0.37847996  0.62152004]
```

So we find that after 20 years, only 42% of the population will remain in the city.

Question Time! Q11.2

Predicting the Distant Future

A important question about a Markov Chain is: what will happen in the distant future?

For example, what happens to the population distribution in our example "in the long run?" We noticed that the population of the city is going down. Will everyone eventually live in the suburbs?

Rather than answering that question right now, we'll take a more interesting example.

Suppose we have a system whose state transition is described by the stochastic matrix

$$P = \begin{bmatrix} .5 & .2 & .3 \\ .3 & .8 & .3 \\ .2 & 0 & .4 \end{bmatrix}$$

and which starts int the state

$$\mathbf{x_0} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Consider the Markov Chain defined by P and $\mathbf{x_0}$, that is the chain defined as

$$\mathbf{x_{k+1}} = P\mathbf{x_k} \quad \text{for } k = 0, 1, 2...$$

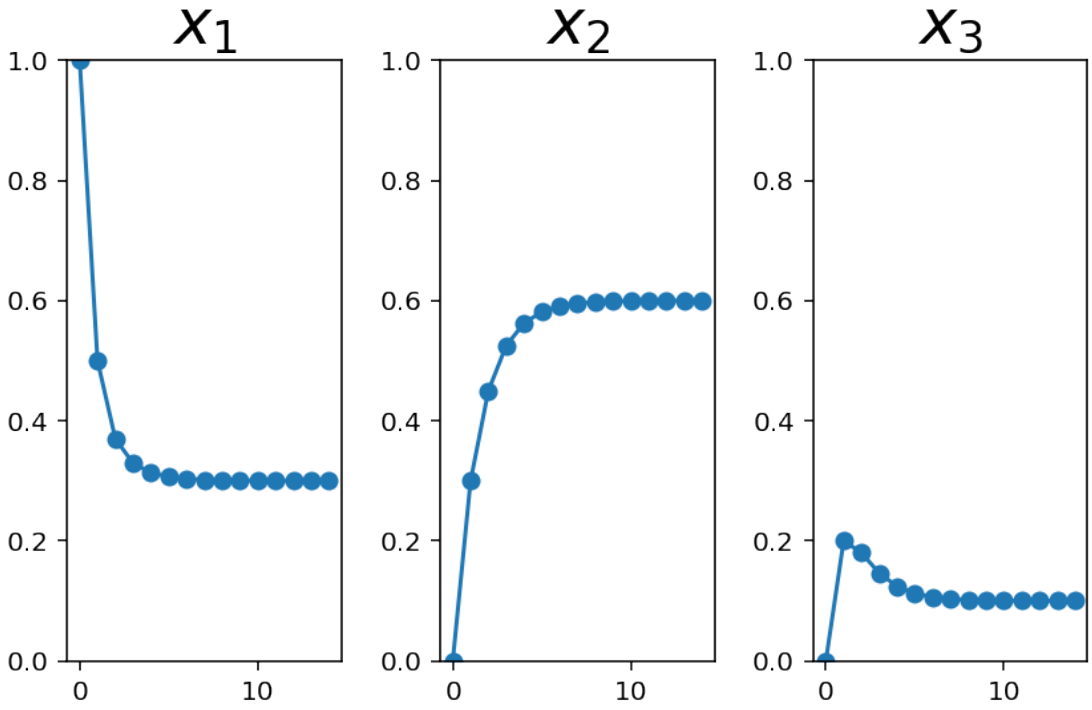
What happens to the system as time passes?

Let's compute the state vectors $\mathbf{x_1}, \ldots, \mathbf{x_{15}}$ to find out.

[13]:
```
A = np.array(
        [[.5,.2,.3],
         [.3,.8,.3],
         [.2, 0,.4]])
    x = np.array([1,0,0])
    xs = np.zeros((15,3))
    for i in range(15):
        xs[i] = x
        print('x({}) = {}'.format(i,x))
        x = A.dot(x)
```

```
x(0) = [1 0 0]
x(1) = [ 0.5  0.3  0.2]
x(2) = [ 0.37  0.45  0.18]
x(3) = [ 0.329  0.525  0.146]
x(4) = [ 0.3133  0.5625  0.1242]
x(5) = [ 0.30641  0.58125  0.11234]
x(6) = [ 0.303157  0.590625  0.106218]
x(7) = [ 0.3015689  0.5953125  0.1031186]
x(8) = [ 0.30078253  0.59765625  0.10156122]
x(9) = [ 0.30039088  0.59882813  0.10078099]
x(10) = [ 0.30019536  0.59941406  0.10039057]
x(11) = [ 0.30009767  0.59970703  0.1001953 ]
x(12) = [ 0.30004883  0.59985352  0.10009765]
x(13) = [ 0.30002441  0.59992676  0.10004883]
x(14) = [ 0.30001221  0.59996338  0.10002441]
```

What is going on here? Let's look at these values graphically.



Based on visual inspection, these vectors seem to be approaching

$$\mathbf{q} = \begin{bmatrix} .3 \\ .6 \\ .1 \end{bmatrix}.$$

The components of $\mathbf{x_k}$ don't seem to be changing much past about $k = 10$.

In fact, we can confirm that the this system would be stable at $\begin{bmatrix} .3 \\ .6 \\ .1 \end{bmatrix}$ by noting that:

$$\begin{bmatrix} .5 & .2 & .3 \\ .3 & .8 & .3 \\ .2 & 0 & .4 \end{bmatrix} \begin{bmatrix} .3 \\ .6 \\ .1 \end{bmatrix} = \begin{bmatrix} .15 + .12 + .03 \\ .09 + .48 + .03 \\ .06 + 0 + .04 \end{bmatrix} = \begin{bmatrix} .3 \\ .6 \\ .1 \end{bmatrix}.$$

This calculation is exact. So it seems that:

- the sequence of vectors is approaching $\begin{bmatrix} .3 \\ .6 \\ .1 \end{bmatrix}$ as a limit, and
- when and if they get to that point, they will stabilize there.

Steady-State Vectors

This convergence to a "steady state" is quite remarkable. Is this a general phenomenon?

Definition. If P is a stochastic matrix, then a **steady-state vector** (or **equilibrium vector**) for P is a probability vector $\mathbf{q}$ such that:

$$P\mathbf{q} = \mathbf{q}.$$

It can be shown that **every stochastic matrix has a steady-state vector.** (We'll study this more closely in a later lecture.)

Example.

$\begin{bmatrix} .3 \\ .6 \\ .1 \end{bmatrix}$ is the steady-state vector for $\begin{bmatrix} .5 & .2 & .3 \\ .3 & .8 & .3 \\ .2 & 0 & .4 \end{bmatrix}$.

Example.

Recalling the population-movement example above.

The probability vector $\mathbf{q} = \begin{bmatrix} .375 \\ .625 \end{bmatrix}$ is a steady-state vector for the population migration matrix A, because

$$A\mathbf{q} = \begin{bmatrix} .95 & .03 \\ .05 & .97 \end{bmatrix} \begin{bmatrix} .375 \\ .625 \end{bmatrix} = \begin{bmatrix} .35625 + .01875 \\ .01875 + .60625 \end{bmatrix} = \begin{bmatrix} .375 \\ .625 \end{bmatrix} = \mathbf{q}.$$

To interpret this: if the total population of the region is 1 million, then if there are 375,000 persons in the city and 625,000 persons in the suburbs, the populations of both the city and the suburbs would stabilize – they would stay the same in all future years.

Finding the Steady State

OK, so it seems that the two Markov Chains we have studied so far each have a steady state. This leads to two questions:

- Can we compute the steady state?
  - So far we have guessed what the steady state is, and then checked. Can we compute the steady state directly?
- How do we know if:
  - a Markov Chain has a unique steady state, and
  - whether it will always converge to that steady state?

Let's start by thinking about how to compute the steady-state directly.

Example. Let $P = \begin{bmatrix} .6 & .3 \\ .4 & .7 \end{bmatrix}$. Find a steady-state vector for P.

Solution. Let's simply solve the equation $P\mathbf{x} = \mathbf{x}$.

$$P\mathbf{x} = \mathbf{x}$$

$$P\mathbf{x} - \mathbf{x} = \mathbf{0}$$

$$P\mathbf{x} - I\mathbf{x} = \mathbf{0}$$

$$(P - I)\mathbf{x} = \mathbf{0}$$

Now, $P - I$ is a matrix, so this is a linear system that we can solve.

$$P - I = \begin{bmatrix} .6 & .3 \\ .4 & .7 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -.4 & .3 \\ .4 & -.3 \end{bmatrix}.$$

To find all solutions of $(P - I)\mathbf{x} = \mathbf{0}$, we row reduce the augmented matrix:

$$\begin{bmatrix} -.4 & .3 & 0 \\ .4 & -.3 & 0 \end{bmatrix} \sim \begin{bmatrix} -.4 & .3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & -3/4 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

So $x_1 = \frac{3}{4}x_2$ and x_2 is free. The general solution is $\begin{bmatrix} \frac{3}{4}x_2 \\ x_2 \end{bmatrix}$.

This means that there are an infinite set of solutions. Which one are we interested in?

Remember that our vectors $\mathbf{x}$ are *probability vectors.* So we are interested in the solution in which the vector elements are nonnegative and sum to 1.

The simple way to find this is to take any solution, and divide it by the sum of the entries (so that the sum then adds to 1.)

Let's choose $x_2 = 1$, so the specific solution is:

$$\begin{bmatrix} \frac{3}{4} \\ 1 \end{bmatrix}.$$

Normalizing this by the sum of the entries ($\frac{7}{4}$) we get:

$$\mathbf{q} = \begin{bmatrix} \frac{3}{7} \\ \frac{4}{7} \end{bmatrix}.$$

So, we have found how to solve a Markov Chain for its steady state:

- Solve the linear system $(P - I)\mathbf{x} = \mathbf{0}$.

- The system will have an infinite number of solutions, with one free variable. Obtain a general solution.
- Pick any specific solution (choose any value for the free variable), and normalize it so the entries add up to 1.

Existence of, and Convergence to, Steady State

Finally: when does a system have a **unique** solution that is a probability vector, and how do we know it will converge to that vector?

Of course, a linear system in general might have no solutions, or it might have a unique solution that is not a probability vector. So what we are asking is, when does a system defined by a Markov Chain have an infinite set of solutions, so that we can find one of them that is a probability vector?

Definition. We say that a stochastic matrix P is *regular* if some matrix power P^k contains only strictly positive entries.

For

$$P = \begin{bmatrix} .5 & .2 & .3 \\ .3 & .8 & .3 \\ .2 & 0 & .4 \end{bmatrix},$$

We note that P does not have every entry strictly positive.

However:

$$P^2 = \begin{bmatrix} .37 & .26 & .33 \\ .45 & .70 & .45 \\ .18 & .04 & .22 \end{bmatrix}.$$

Since every entry in P^2 is positive, P is a regular stochastic matrix.

Theorem. If P is an $n \times n$ regular stochastic matrix, then P has a unique steady-state vector $\mathbf{q}$. Further, if $\mathbf{x_0}$ is any initial state and $\mathbf{x_{k+1}} = P\mathbf{x_k}$ for $k = 0, 1, 2, \ldots$, then the Markov Chain $\{\mathbf{x_k}\}$ converges to $\mathbf{q}$ as $k \to \infty$.

Note the phrase "any initial state." This is a remarkable property of a Markov Chain: it converges to its steady-state vector **no matter what state the chain starts in.**

We say that the long-term behavior of the chain has "no memory of the starting state."

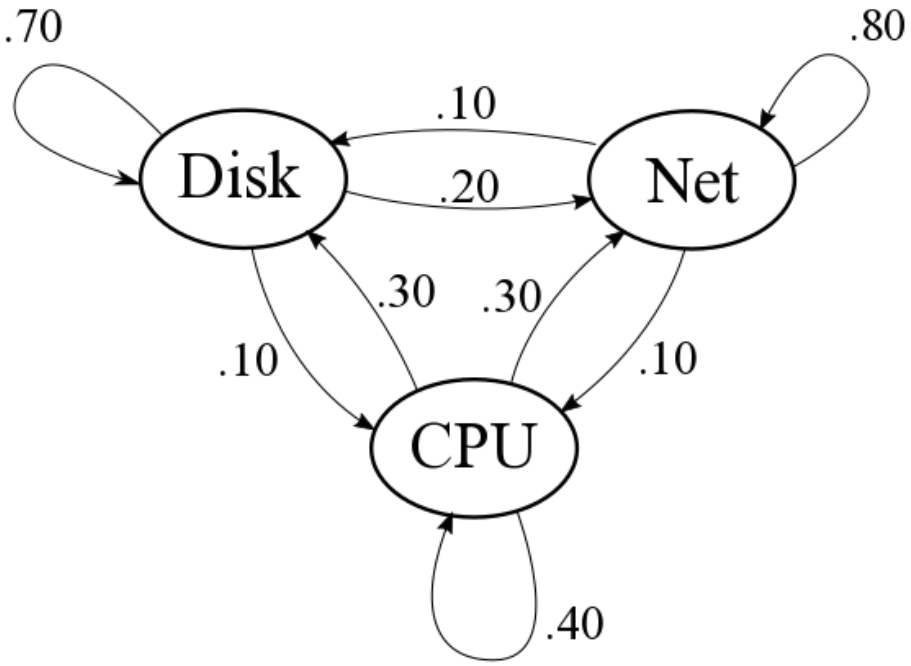
Question Time! Q11.3

Example. Consider a computer system that consists of a disk, a CPU, and a network interface.

A set of jobs are loaded into the system. Each job makes requests for service from each of the components of the system.

After receiving service, the job then next requests service from the same or a different component, and so on.

Jobs move between system components according to the following diagram. For example, after receiving service from the Disk, 70% of jobs return to the disk for another unit of service; 20% request service from the network interface; and 10% request service from the CPU.



Assume the system runs for a long time. Determine whether the system will stabilize, and if so, find the fraction of jobs that are using each device once stabilized. Which device is busiest? Least busy?

Solution. From the diagram, the movement of jobs among components is given by:

	From Disk	From Net	From CPU
To Disk	0.70	0.10	0.30
To Net	0.20	0.80	0.30
To CPU	0.10	0.10	0.40

This corresponds to the stochastic matrix:

$$P = \begin{bmatrix} 0.70 & 0.10 & 0.30 \\ 0.20 & 0.80 & 0.30 \\ 0.10 & 0.10 & 0.40 \end{bmatrix}.$$

First of all, this is a regular matrix because P has all strictly positive entries. So it has a steady-state vector.

Next, we find the steady state of the system by solving $(P - I)\mathbf{x} = \mathbf{0}$:

$$[P - I\ \mathbf{0}] = \begin{bmatrix} -0.30 & 0.10 & 0.30 & 0 \\ 0.20 & -0.20 & 0.30 & 0 \\ 0.10 & 0.10 & -0.60 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & -9/4 & 0 \\ 0 & 1 & -15/4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus the general solution of $(P - I)\mathbf{x} = \mathbf{0}$ is

$$x_1 = \frac{9}{4}x_3, \quad x_2 = \frac{15}{4}x_3, \quad x_3 \text{ free.}$$

Since x_3 is free (there are infinitely many solutions) we can find a solution whose entries sum to 1. This is:

$$\mathbf{q} \approx \begin{bmatrix} .32 \\ .54 \\ .14 \end{bmatrix}.$$

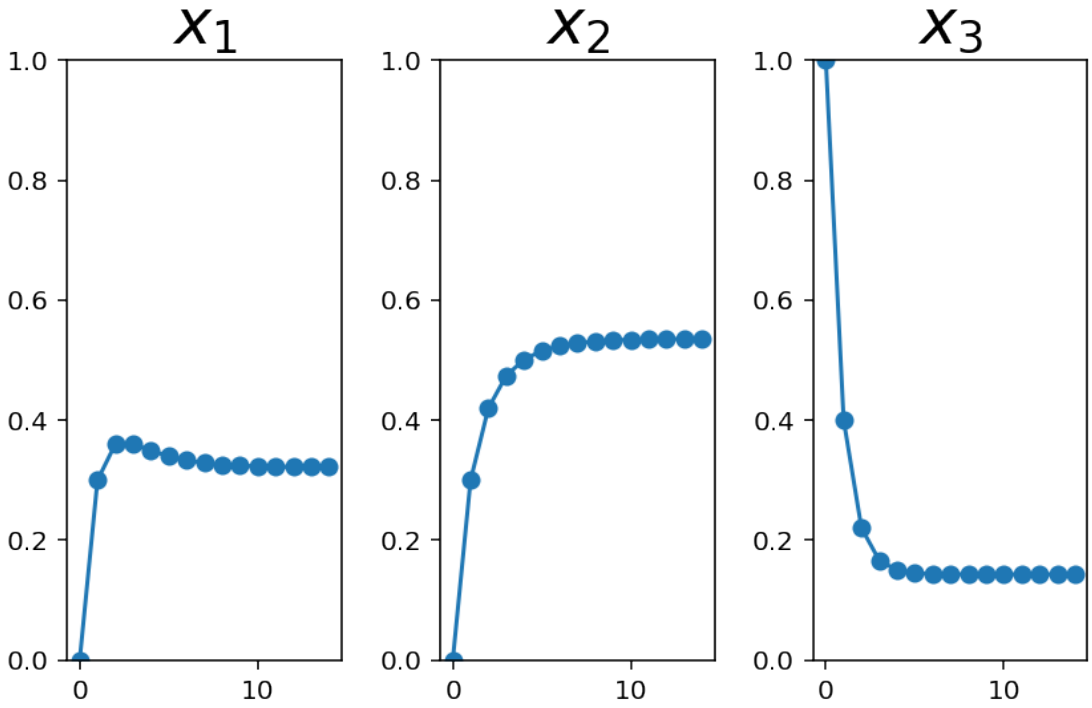
So we see that in the long run, about 32% of the jobs will be using the disk, about 54% will be using the network interface, and about 14% will be using the CPU.

Again, the important fact here is that we did not have to concern ourselves with the state in which that the system started. The influence of the starting state is eventually lost.

Let's demonstrate that computationally. Let's say that at the start, all the jobs happened to be using the CPU. Then:

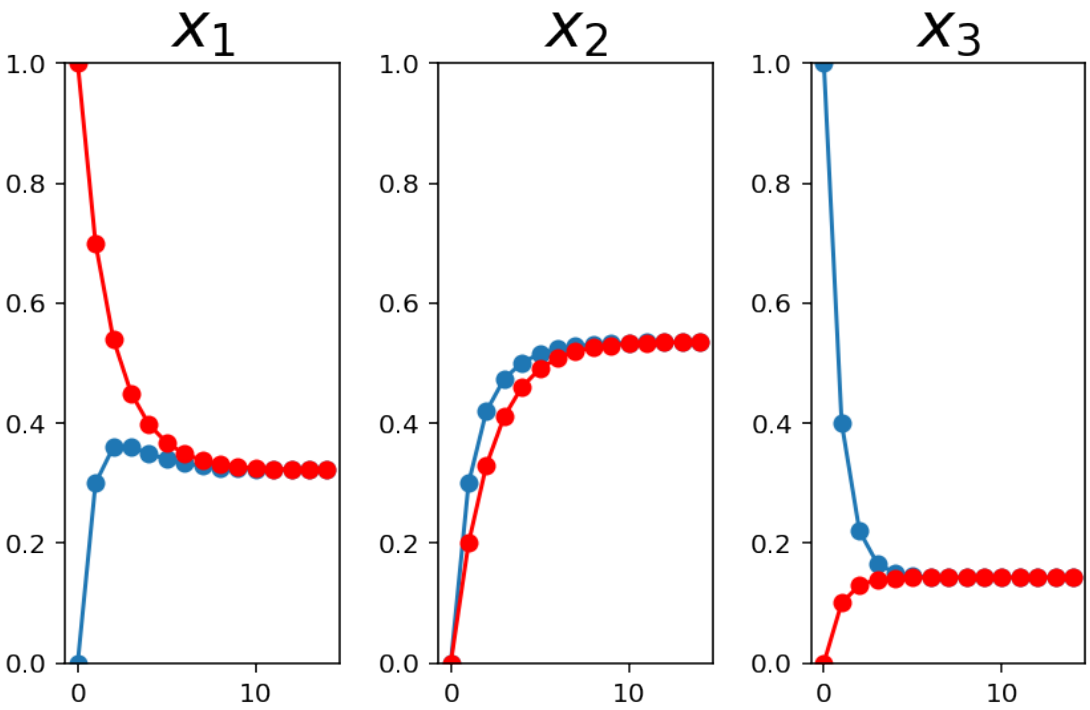
$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Then let's look at how the three elements of the $\mathbf{x}$ vector evolve with time, by computing $P\mathbf{x}$, $P^2\mathbf{x}$, $P^3\mathbf{x}$, etc.



Notice how the system starts in the state $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ but quickly (within about 10 steps) reaches the equilibrium state that we predicted: $\begin{bmatrix} .32 \\ .54 \\ .14 \end{bmatrix}$.

Now let's compare what happens if the system starts in a different state, say $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$:



This shows graphically that even though the system started in a very different state, it quickly converges to the steady state regardless of the starting state.