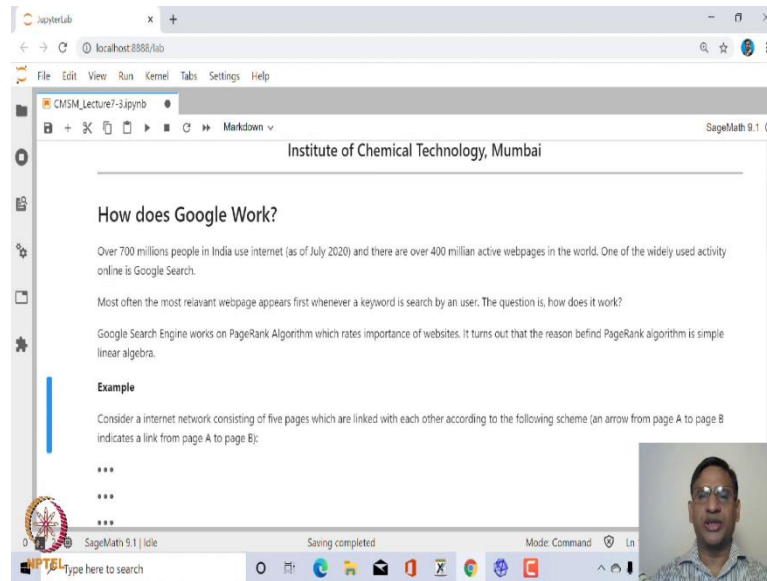


Computational Mathematics with SageMath
Prof. Ajit Kumar
Department of Mathematics
Institute of Chemical Technology, Mumbai

Lecture - 44
Google PageRank Algorithm Using SageMath

(Refer Slide Time: 00:15)

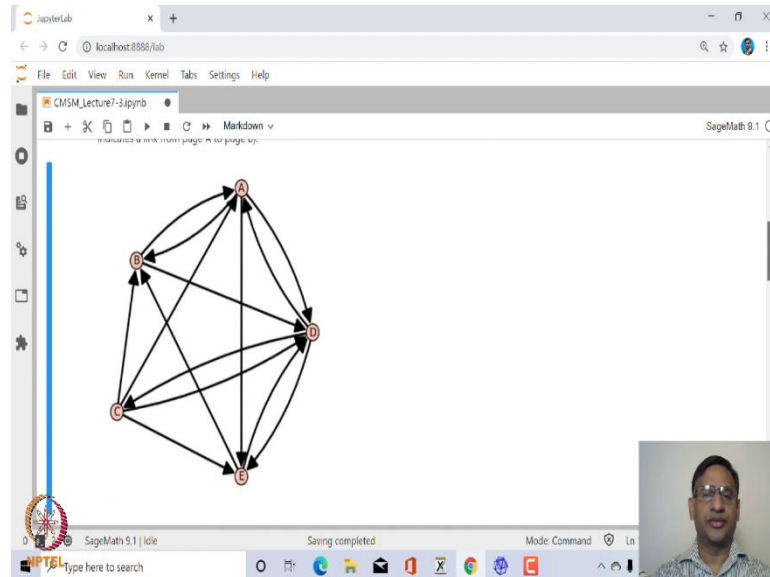


Welcome to the 44th lecture on Computational Mathematics with SageMath. In this lecture, we will look at Google PageRank Algorithm, and how search engine works. So, if you look at the currently, the internet users. For example, in India, over 700 million people use internet as of July 2020.

And there are about 400 million active web pages in the world. So, one of the widely used activity over the internet is search. Most often, when you search any keyword, you get the website which is the most relevant, that appears as the first website. So, you should be wondering, how does it work, how does it happen? So, actually it uses a very

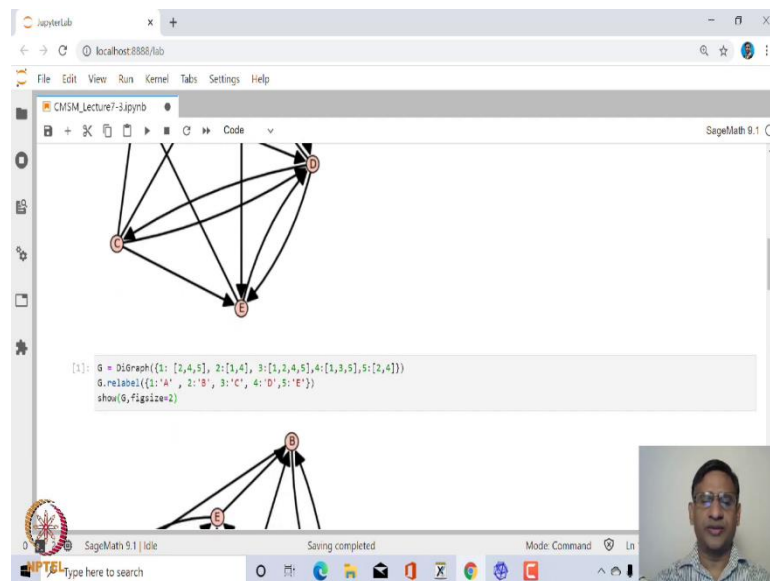
simple concept of linear algebra. So, we shall look at how it works, and the concepts involved in Google PageRank Algorithm.

(Refer Slide Time: 01:44)



So, let us start with a simple internet network of 5 web pages A, B, C, D, E, and the arrow means there is a link from one website to another. So, for example, there is an incoming arrow at A from C. So, there is a link or hyperlink from the website C to A, right?

(Refer Slide Time: 02:13)

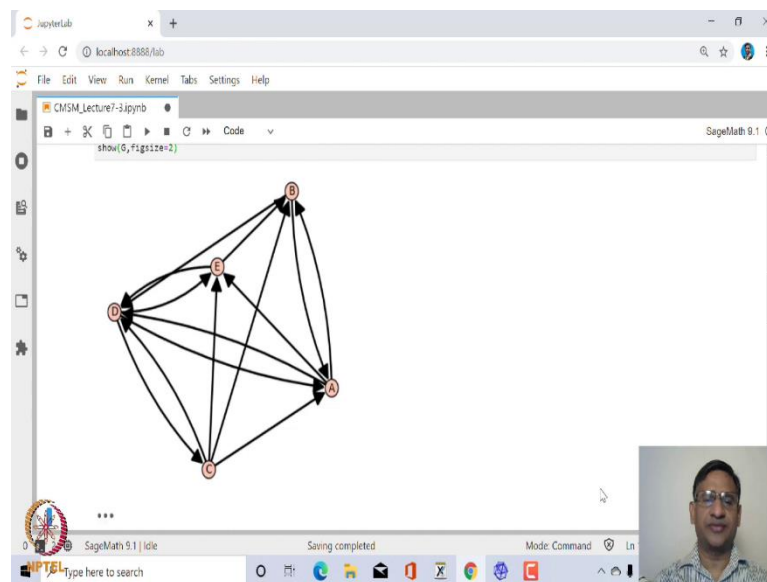


Now, this network you can generate in SageMath using DiGraph function, which is a package. DiGraph is a package for graph theory, Sage is very good at graph theory.

And DiGraph stands for Directed Graphs. So, you can plot this network using this DiGraph. So, what you are doing? You are saying that make a DiGraph, directed graph from a nodes 1, 2, 3, 4, 5 and from node 1, you, it, there is a link to 2, 4 and 5. From link 2, from webpage 2 you have a link to webpage 1 and 4, and so on.

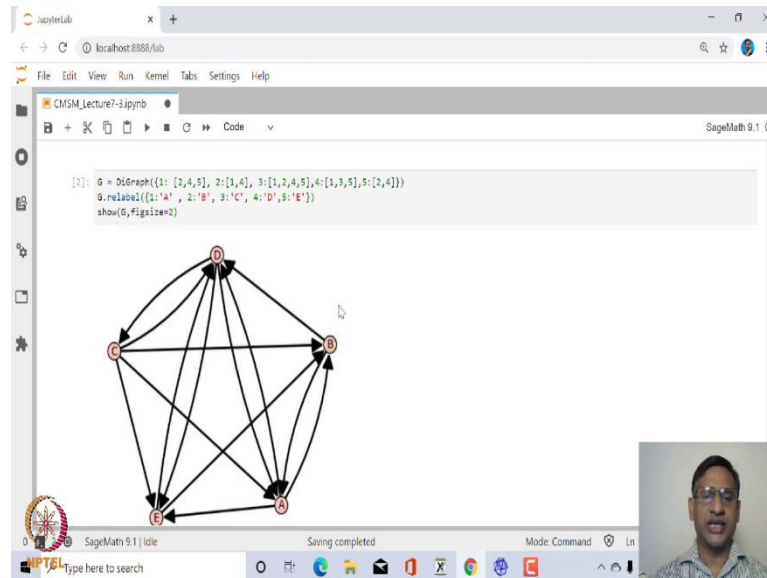
And you can relabel these nodes as A, B, C, D, E, that is what we have. You can, if you do not relabel, it will give you by default label as 1, 2, 3, 4, 5. So, let us run this.

(Refer Slide Time: 03:21)



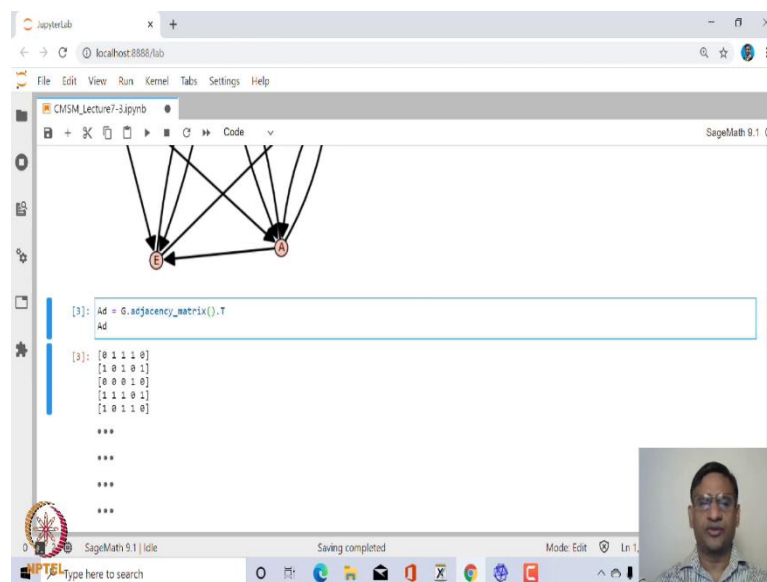
And you will see a network, this graph is plotted randomly. So, the position of each of this node and the arrow may differ when you do this.

(Refer Slide Time: 03:35)



For example, if I run this again, this may be slightly different from what you got earlier.

(Refer Slide Time: 03:44)

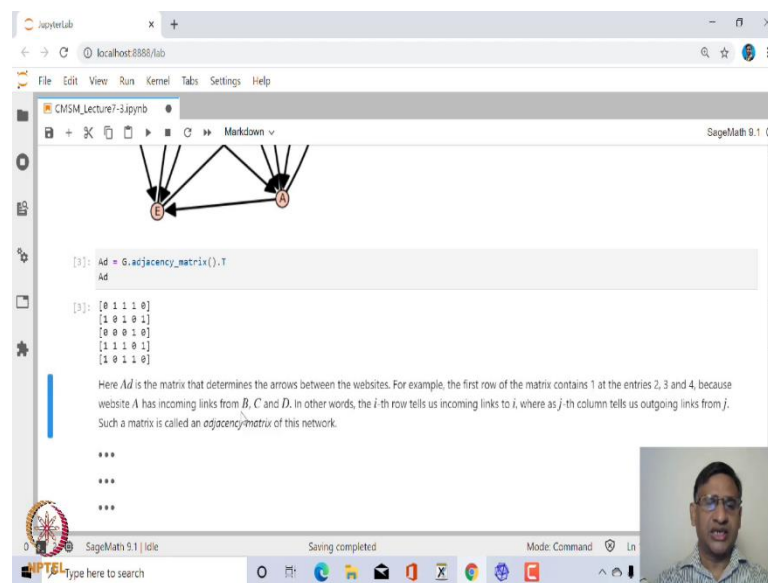


Now, if you look at this website, and to each of this arrow, you can assign to each, to this network you can assign a matrix of 0s and 1s. So, if there is a link from j to i, we can say

that the 'aij'th entry of that matrix is 1, otherwise 0. That matrix is known as adjacency matrix and Sage has inbuilt function to find adjacency matrix of any graph.

So, here G was defined as a graph of vertices and edges, this is directed graph. So, you can find adjacency matrix of this using `G.adjacency_matrix()` and then transpose. So, if, if you are taking i jth entry as 1, when there is a link between from i to j, then it will be an adjacency matrix. But as I said, we will look at, aij entry is equal to 1, when there is a link from j to i. So, that is why we are taking transpose of this, right?

(Refer Slide Time: 05:00)

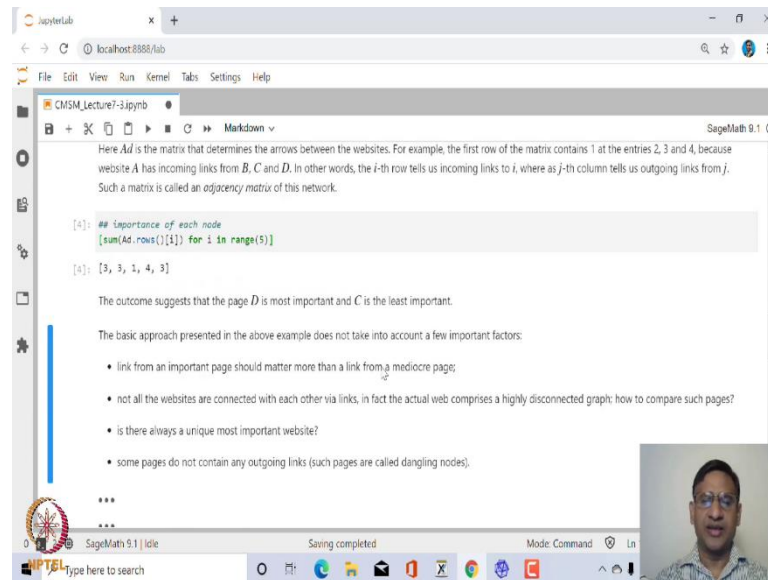


Now, if you look at this matrix, this adjacency matrix, and look at the rows and columns, so if you look at, for example, the rows, let us say look at 2nd row. 2nd row corresponds to the, the node B. So, if you look at the node B, you can see here, there is a link from C to B. There is a link from E to B, and then there is a link from A to B.

So, if you look at this 2nd row, then you can see here 1 C and D 1 C and E. So, those are the, the links, there is no link from C to B. So, the row represents the incoming links, whereas column represents outgoing links. So, for example, if I look at B, from B, there is only one outgoing link to D. So, the 2nd there is one outgoing link to A, as well and there is one outgoing link to A, and 1 to D.

Therefore, 2nd column will have entries $A[1,2]$ as 1, and, and, and $[4,2]$ as 1, right? So, this is how, this you can interpret this adjacency matrix, right? Now, this you can find out, importance of each of this webpage, by the number of incoming links.

(Refer Slide Time: 06:27)



Here A_d is the matrix that determines the arrows between the websites. For example, the first row of the matrix contains 1 at the entries 2, 3 and 4, because website A has incoming links from B, C and D. In other words, the i -th row tells us incoming links to i , where as j -th column tells us outgoing links from j . Such a matrix is called an adjacency matrix of this network.

```
[4]: ## Importance of each node
      [sum(A_d.rows()[i]) for i in range(5)]
```

```
[5]: [3, 3, 1, 4, 3]
```

The outcome suggests that the page D is most important and C is the least important.

The basic approach presented in the above example does not take into account a few important factors:

- link from an important page should matter more than a link from a mediocre page;
- not all the websites are connected with each other via links, in fact the actual web comprises a highly disconnected graph: how to compare such pages?
- is there always a unique most important website?
- some pages do not contain any outgoing links (such pages are called dangling nodes).

So, if you, if you find out the, some of the, the links on each website. So, the first website importance is 3 because there are 3 incoming links. And the, in this case, 4th has maximum, which is 4, and the 3rd one has only 1 incoming link which is 1. So, you can see here, 4th website has maximum importance, whereas the 3rd one has the least importance, right? Now, in, in this case, in this case, if you just simply look at the number of incoming links, etcetera, then many things are not taken into account.

For example, the, the link from an important webpage should matter more than link from mediocre webpage or, and also not all websites are connected. You can have website to websites which are not connected, in that case how do you look at that, right? And is there always a unique, most important website? So, there could be more than one website having the same importance, then which one should appear first? And some pages do not contain any outgoing links, such nodes are known as dangling nodes. So, in that case, what happens? So, these things are not taken into consideration, if you simply look at the

number of incoming links for importance of a website. So, how does one take into account these things?

(Refer Slide Time: 08:06)

The screenshot shows a JupyterLab window with a browser at localhost:8888/lab. The notebook, titled 'CMSM_Lecture7-3.ipynb', contains the following text:

- is there always a unique most important website?
- some pages do not contain any outgoing links (such pages are called dangling nodes).

Importance of Webpage

x_i , the importance score of the web page i

n_i , the number outgoing links from the page i .

In the example, the score of page A would be determined by the relation

$$x_1 = x_2 + x_3 + x_4.$$

However, in this way the page D may vote three times. To avoid such a situation, we normalise the vote by the number of links that the page contains, i.e. the score of page A should be rather given by a formula

$$x_1 = \frac{x_2}{2} + \frac{x_3}{4} + \frac{x_4}{3}.$$

Thus we get

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}x_2 + \frac{1}{4}x_3 + \frac{1}{3}x_4 \\ \frac{1}{3}x_1 + \frac{1}{4}x_3 + \frac{1}{2}x_5 \\ \frac{1}{3}x_4 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{4} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{3} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

The interface also shows a video feed of a person in the bottom right corner.

So, if you look at, for example, suppose we, let us say x_i denotes the importance, importance of the scores of the webpage i , when I say scores means, how many times you have the incoming links. And n_i the number of outgoing links from the webpage i . Then if you look at, for example, first node, first node A , then it has incoming links from B , C , and D , B , C , and D . And number of incoming links from B , number of outgoing links from B is 2 from C . It is 3, and from D , it is 1 here 1, 1, 2 and 3, right?

(Refer Slide Time: 08:57)

the score of page x should be rather given by a formula

$$x_1 = \frac{x_2}{2} + \frac{x_3}{4} + \frac{x_4}{3}$$

Thus we get

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}x_2 + \frac{1}{4}x_3 + \frac{1}{3}x_4 \\ \frac{1}{3}x_1 + \frac{1}{4}x_3 + \frac{1}{2}x_5 \\ \frac{1}{3}x_1 \\ \frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{4}x_3 + \frac{1}{2}x_5 \\ \frac{1}{3}x_1 + \frac{1}{4}x_3 + \frac{1}{3}x_4 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{4} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{4} & \frac{1}{3} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

In general, if a web consists of n pages and $L_k \subset \{1, 2, \dots, n\}$ denotes the set of pages which link to the page k , then we define

$$x_k = \sum_{j \in L_k} \frac{x_j}{|L_k|}$$

Thus finding x_1, x_2, \dots, x_n amounts to solving $Ax = x$. That is x is eigenvector of A corresponding to eigenvalue 1.

...

...

...

So, therefore, in this case, the x_1 , which is the importance of the scores at A should be equal to x_2 plus x_3 plus x_4 . However, if I look at x_2 , x_2 has 2 outgoing links. So, there is equal probability that it gives vote to A or to D.

For example, from B you have outgoing links to A, and to D. So, there is equal probability that B votes to A or D. So, therefore, the probability is half, therefore, x_1 will be, now x_2 divided by 2. Similarly, x_3 divided by 4 from x_3 , there are 4 outgoing links, and from x_4 there are only 3 outgoing links. So, this is x_1 is equal to x_2 by 2 plus x_3 by 4 plus x_4 by 3.

Similarly, you can do, find this relation for the 2nd node, that is for B, for B, you will get x_2 is equal to $\frac{1}{3}x_1$ plus $\frac{1}{4}x_3$ plus $\frac{1}{2}x_5$, from x_3 will be x_3 is equal to $\frac{1}{3}x_1$ by 3 x_4 , and x_4 will be this and x_5 will be this. Therefore, this set of equations reduces to x_1, x_2, x_3, x_4, x_5 is equal to this, and that can be written as matrix times x_1, x_2, x_3, x_4, x_5 .

So, you have these, these equations is written as, as linear equation $Ax = x$. So, if x is a solution of this system of linear equations, it amounts to saying that x is eigenvector of A with eigenvalue 1. So, in particular, you, this, this problem reduces to

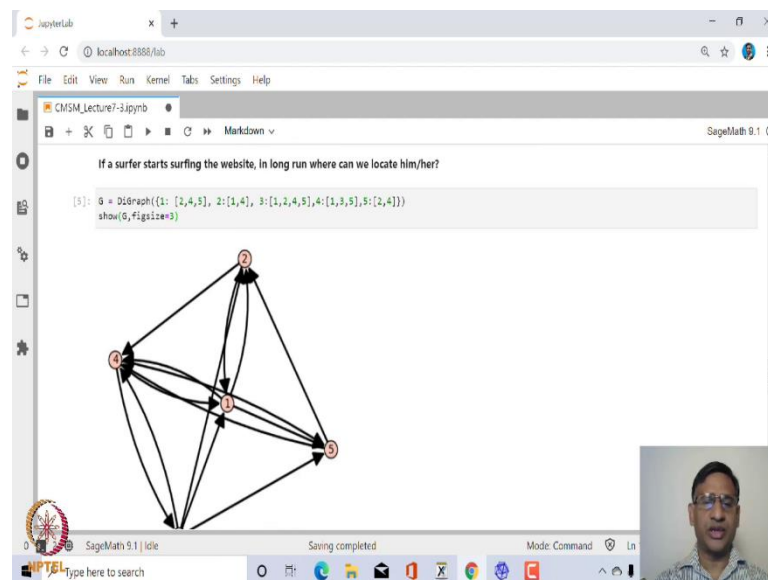
finding eigenvector of this particular matrix, which generally we call as transition matrix or probability matrix with eigenvalue 1.

Now, if you look at finding eigenvalue, eigenvector is not a big task. And also if you notice this, each, each in this matrix, each column's sum is 1, therefore, this is what is called column stochastic matrix. Similarly, you can define row stochastic matrix. So, and each entry in this is nonnegative because it is a probability, right?

So, therefore, therefore you can define the importance of any k th node as summation x_j upon summation n_j where, what is n_j ? n_j is the number of outgoing links from j , and what is j varying over L_k ? L_k is the, the subset of 1 to n , that denotes the set of incoming links, right?

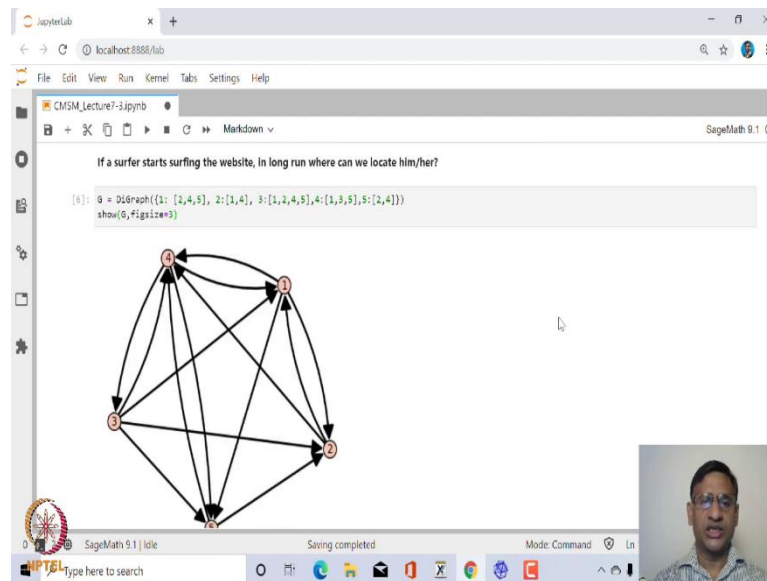
So, actually this problem of finding the importance of any webpage is, amounts to, or it is equivalent to finding eigenvalue and eigenvector with respect to eigenvalue 1. And this stochastic matrix, you can show that always has one as eigenvalue, right?

(Refer Slide Time: 12:16)

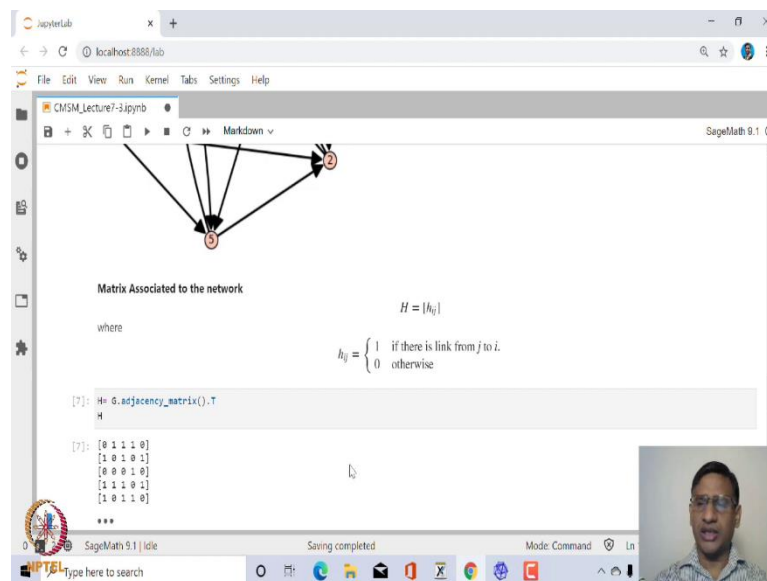


So, let us look at this again. So, this problem you can also think of it in the following way, if you, if you, somebody starts surfing the internet, then in long run where can you find him right after 1 click, after 2 click, after 100 clicks, after 1000 clicks? So, it will give you what is his probability of being on a particular website. So, how do we find

that? So, again let us look at the same, same network of 5 nodes, this is slightly better, same network of 5 nodes. (Refer Slide Time: 12:46)

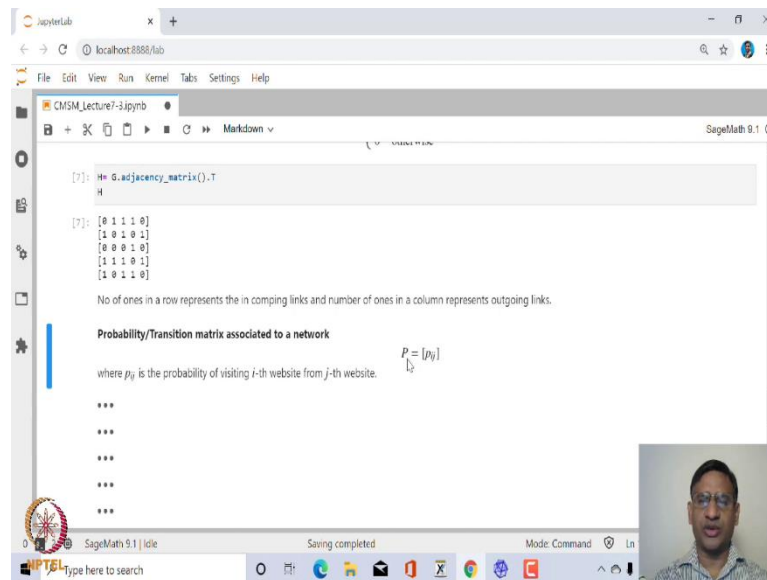


(Refer Slide Time: 12:52)



And to this network, we will associate a matrix which is called transition matrix, which I am calling as h, h_{ij} is 1, if there is a link from j to i and 0 otherwise. So, then this is the adjacency matrix, adjacency matrix.

(Refer Slide Time: 13:09)



The JupyterLab interface shows a SageMath 9.1 kernel. The code cell contains the following:

```
[7]: H = G.adjacency_matrix().T
H
```

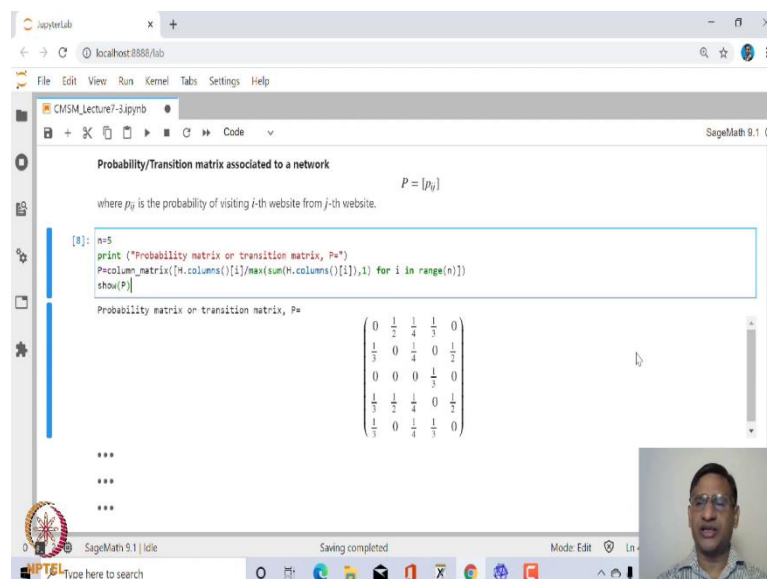
The output is a 5x5 matrix:

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Below the matrix, a text explanation states: "No of ones in a row represents the in coming links and number of ones in a column represents outgoing links." This is followed by a section titled "Probability/Transition matrix associated to a network" with the definition $P = [p_{ij}]$ where p_{ij} is the probability of visiting i -th website from j -th website.

And then we will find the probability matrix or transition matrix attached to this network. That is actually, each entry, you divide by the sum of the entries in the column. So, that is why it will be column, column stochastic matrix.

(Refer Slide Time: 13:22)



The JupyterLab interface shows the same SageMath 9.1 kernel. The code cell contains the following:

```
[8]: n=5
print ("Probability matrix or transition matrix, P=")
P=column_matrix([H.columns()[i]/max(sum(H.columns()[i]),1) for i in range(n)])
show(P)
```

The output is the transition matrix P, displayed as a column matrix:

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{4} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{4} & \frac{1}{3} & 0 \end{bmatrix}$$

Below the matrix, a text explanation states: "Probability matrix or transition matrix, P=".

So, you can see here it is quite easy, you take any of the columns of H, and divide it by this sum of the, the columns, entry in the columns for, and do it for each entry, right? So, this is the adjacency, this is the transition matrix associated to this network of 5 websites.

(Refer Slide Time: 13:51)

```

P is columns stochastic matrix.

[9]: ## Starting website of a surfer
v=vector([0,1,0,0,0])
P*v

[9]: (1/2, 0, 0, 1/2, 0)

[10]: P*(P*v)

[10]: (1/6, 1/6, 1/6, 1/6, 1/3)

[11]: ## Probability vector after k clicks
k=100
P^k*v.n(digits=6)

[11]: (0.219512, 0.195122, 0.0975618, 0.292683, 0.195122)

[12]: k=1000
P^k*v.n(digits=6)

[12]: (0.219512, 0.195122, 0.0975618, 0.292683, 0.195122)

```

Now, suppose, suppose somebody starts from, let us say 2nd website, he starts surfing this particular network from 2nd website. So, after 1 click for example, from the 2nd website, after 1 click from 2nd website, you have 2 outgoing links 1 to 1 and 1 to 4.

So, there is a half probability to be at 1 is half probability, at 4 also half, right? So, after, after this, after that is, 1 click, after 2nd click, this half again, you, from 1, from website 1 you have 1 outgoing link, 2 outgoing, 3 outgoing links. So, the probability will be half times 1 by 3 to here, half times 1 by 3 to here, half times 1 by 3 to here.

So, how do we obtain after 1 click? That you can obtain simply, by multiplying this, this vector 0, 1, 0, 0, 0 by this matrix, that is the probability matrix or transition matrix. So, if you say P into v the probability of being on website 1 is half, and on website 4 is half. If I, if I look at what happens after 2 clicks, so then I will have to again multiply this by P .

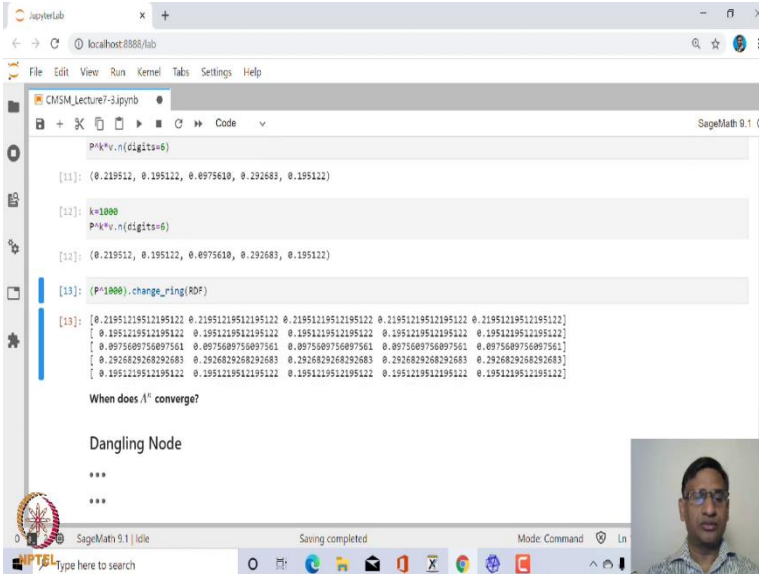
So, P into this, which is $P^2 v$, that will give you probability on website 1 is $1/6$ at 2 is $1/6$ at 3 is $1/6$ at 4 is $1/6$ and at 5th is one third. So, in general, you can find what happens to this probability after, let us say 100 clicks? After 100 clicks, the probability in decimal is like this 0.2195 and so on, right?

So, you can see here, in this case, the maximum probability is here at the website 4th. So, that is about 30 percent chance of him being at website 4th, and about 9 percent chance at website 3, and so on. So, you can, similarly you can look at, let us say what happens

after 1000 clicks. So, after 1000 clicks, again you can see here, this vector what you have got, that is again a stochastic vector.

Because, some of the entries in this, it will be 1 and each entry is nonnegative. So, this is actually almost same as this, it may differ in after so many decimal places, but they are almost identical. So, what it means is that, in long run, this will be the probability of him being on particular websites, and that you can arrange in decreasing order.

(Refer Slide Time: 16:37)



```

P = P^k, n(digits=6)
[11]: (0.219512, 0.195122, 0.0975610, 0.292683, 0.195122)

[12]: k=1000
P = P^k, n(digits=6)
[12]: (0.219512, 0.195122, 0.0975610, 0.292683, 0.195122)

[13]: (P^1000).change_ring(RDF)
[13]: [0.21951219512195122 0.21951219512195122 0.21951219512195122 0.21951219512195122 0.21951219512195122]
[0.1951219512195122 0.1951219512195122 0.1951219512195122 0.1951219512195122 0.1951219512195122]
[0.0975609756097561 0.0975609756097561 0.0975609756097561 0.0975609756097561 0.0975609756097561]
[0.2926829268292683 0.2926829268292683 0.2926829268292683 0.2926829268292683 0.2926829268292683]
[0.1951219512195122 0.1951219512195122 0.1951219512195122 0.1951219512195122 0.1951219512195122]

When does A^k converge?

Dangling Node

***

```

So, for example, the maximum is this. So, that is the, the probability of it being at website 4 is 30 and so on. The 2nd probability is this. So, that is this, and then these 2 have almost equal probability, and then this is the last one, right? So, that is how you can obtain the importance of the website.

So, now this you can translate into the, the internet, the keyword search. So, if somebody gives a keyword for searching on, on Google then, what it will do? It will search all over the places on the internet. And wherever that keyword is obtained, it will look at all these, these links and rank accordingly to this, this, right?

So, right, next you can look at, now in this case, what it amounts to is that finding very large power of this matrix P. Now, if you look at, this is only 5 websites with, network with 5 websites, but in general, as I mentioned, there are about 400 million websites,

which are active currently. So, therefore, you will be dealing with 400 million by 400 million matrix.

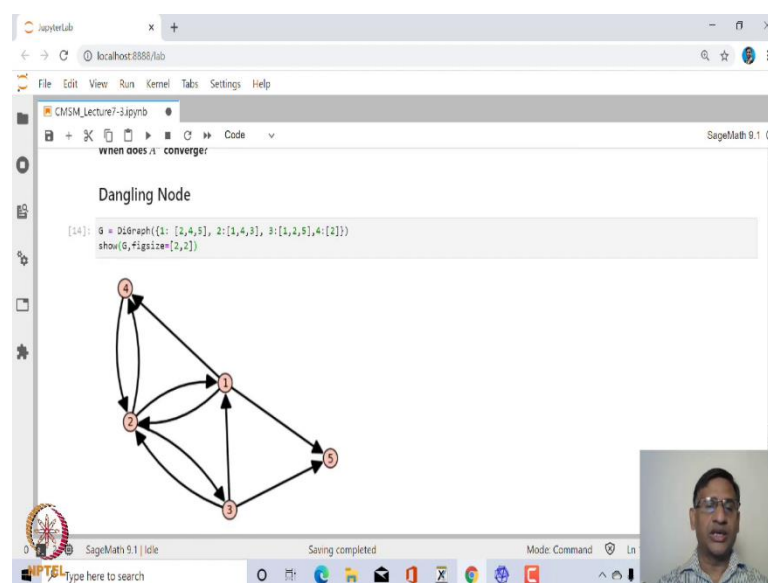
And finding power of such a large matrix is just impossible, even the very, very efficient computer will have problem. So, therefore, in that case, one has to find this notion using eigenvalue, eigenvector diagonalizations, which we already spoke about, right?

So, and also now, if you look at, for example, the power, let us say 1000 power of this matrix, transition matrix P , what you see is each column is the vector which you obtained here, the probability vector. This is what is each, this is what is called stationary vector. And so each, each column will be actually eigenvector with eigenvalue 1.

So, in particular, P to the power n converges. Now, again you can ask when does a matrix to the power n , you can think of this as sequence of matrix power, when does it converge. So, there is a whole theory about convergence of this power of matrices, I will give you reference little later. So, you can go through that reference and learn about these convergences.

For example, one of the criteria for any stochastic matrix to be to be convergent is that, it should be a regular matrix, it should be a regular stochastic matrix. That is same as saying some power of P should have all positive entries, right?

(Refer Slide Time: 19:35)

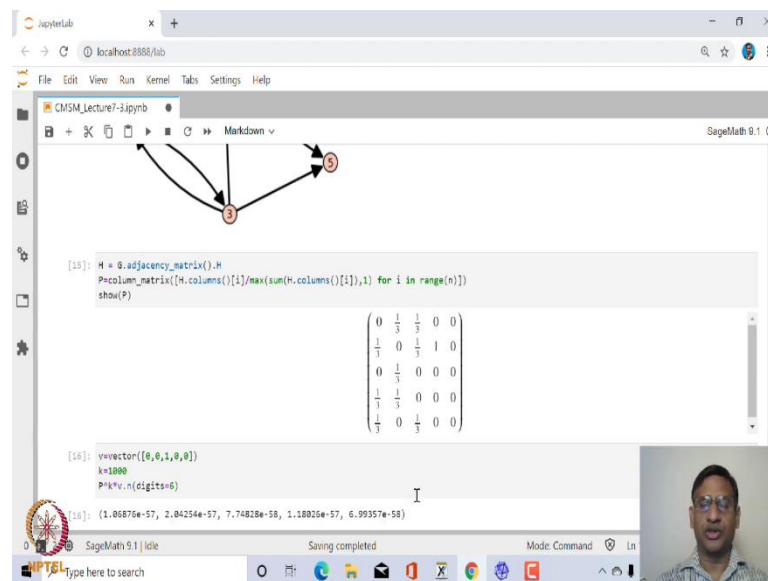


Now, you can also have a situation where you have a dangling node. What is a dangling node?

A dangling node is one where you can go to some particular website, but after that, there is no outgoing link from there. So, for example, many times you open click on an image of something on a link, and it opens in a separate website. Or you could have a link to a pdf file, when you open this, open this pdf file, it opens in a separate browser or separate tab.

In that case, you may not be able to go from that, we, there may not be any outgoing link; however, in actual practice, what we do rather, we go back or we, we close that particular tab and open a new tab. So, this we do, but in an algorithm, it has to be taken care, taken into account. So, this is a dangling node, where you can see here, the node 5 is a dangling node. From 5 there are incoming links, but there is no outgoing link from website 5.

(Refer Slide Time: 20:42)

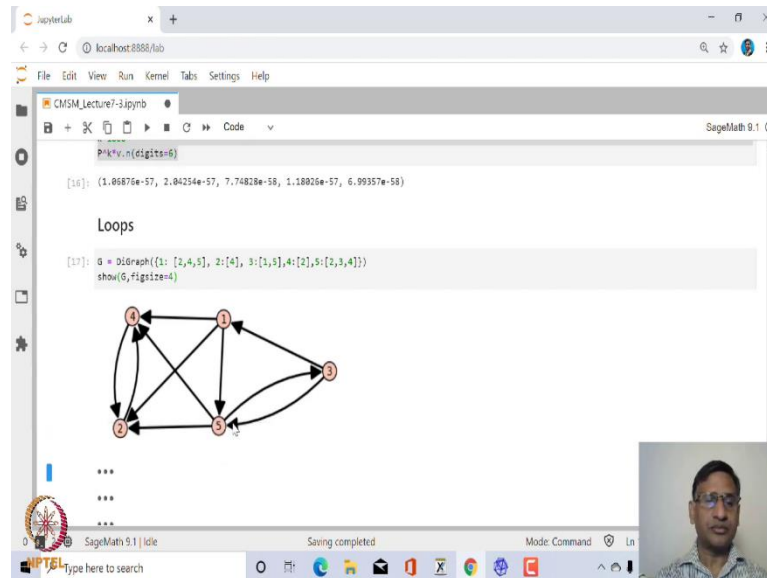


So, if in this case, suppose, let us look at, we, we find out adjacency matrix and the transition matrix of this, this particular network, and in case we start from the website 3 and then look at what happens in long run.

So, you can see here, all this, probability of going at each website is 0, because it has got stuck at webpage 5. So, this is not going to work, we cannot find rank or importance

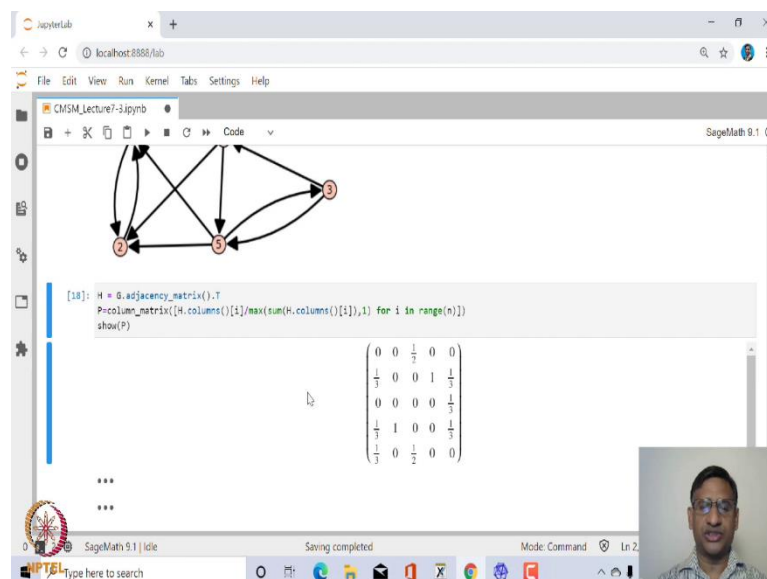
of website using, in this case, using this particular way. So, there must be other way of dealing with this. You can also have a situation of loop.

(Refer Slide Time: 21:21)



So, for example, if you look at this particular network, if you look at this particular network, then if you see here, the example 2 and 4. So, from 2 you can go to 4 and from 4 you can go to 2, and you cannot go anywhere else. So, this is a loop.

(Refer Slide Time: 21:43)



So, now again in this case, if I start with some, some website so this is an adjacency matrix, and this transition matrix.

(Refer Slide Time: 21:49)

```

[19]: v=vector([0,1,0,0,0])
      k=10000
      P^k*v.n(digits=6)

[19]: (0.000000, 0.000000, 0.000000, 1.000000, 0.000000)

***
***
***
***
***

```

And if I start with some, let us say 2nd website, then the probability of being on 2nd website is 1, and everywhere else it is 0, that again is not good.

(Refer Slide Time: 22:03)

```

[20]: v=vector([1,0,0,0,0])
      k=10000
      P^k*v.n(digits=6)

[20]: (5.97504e-2021, 0.500000, 5.39519e-2021, 0.500000, 0.51571e-2021)

***
***
***
***
***

```

If I start with, let us say website 1 and then see what happens after let us say some, some 10,000 clicks, some 10,000 clicks.

(Refer Slide Time: 22:12)

```

[1]:

$$P = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 1 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 1 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix}$$


[2]:
v = vector([1, 0, 0, 0, 0])
k = 10000
P^k * v, n(digits=6)

[3]:
(9.71663e-2821, 0.500000, 1.81537e-2820, 0.500000, 1.59156e-2820)

```

Then you can see here, again the probability at 2nd website is 0.5 at 4th website is 0.5 and remaining all 0, right? So, in case you have dangling node or loop, this way of finding eigenvector with respect to eigenvalue 1 may not work, or this just finding the importance, by taking the large power times, the initial vector will not work, ok?

(Refer Slide Time: 22:42)

```

[1]:

$$P = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix}$$


[2]:
v = vector([1, 0, 0, 0, 0])
k = 10000
P^k * v, n(digits=6)

[3]:
(9.71663e-2821, 0.500000, 1.81537e-2820, 0.500000, 1.59156e-2820)

```

How to handle dangling nodes and loops?

Assume that a web network consists of n pages. Let S be an $n \times n$ matrix with all entries equal to $\frac{1}{n}$. We replace the probability matrix P attached to this network

$$G = \alpha P + (1 - \alpha)S.$$

Here α is called the *damping factor* which tells us how much times a surfer follow the hyperlink that teleporting.

So, so the question is, how does one deal with such things? Now, Google what it does? It, it deals with such situations by defining what is called Google matrix G , which is

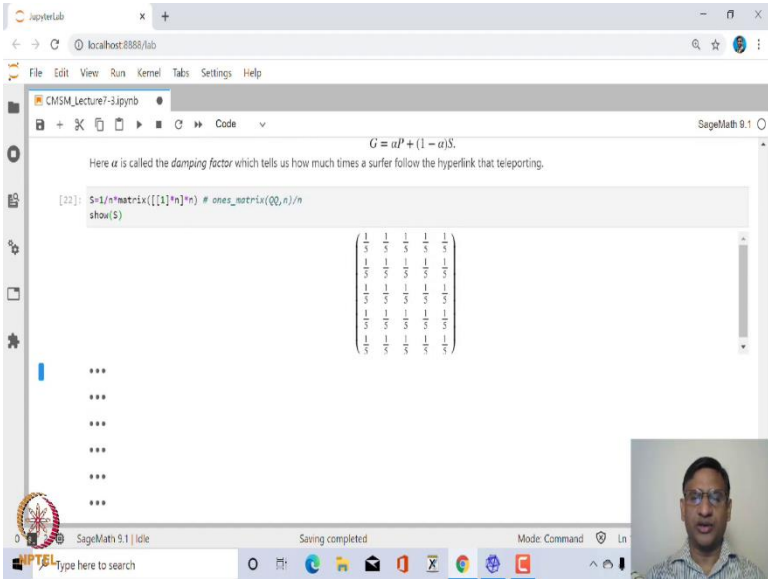
defined as a weighted average of this transition matrix P and the matrix S where S is a matrix with all entries are 1 by n .

So, 1 by n , all its entries are 1 by n , where n is the number of websites in, in a particular network. It is simply, it simply means that every website is connected to every other, right? So, that is why the probability of each will be 1 by n , right?

So, this is how we, we modify now, or we update this transition matrix to this Google matrix, which is weighted average of P and S and here α is known as damping factor, and α lies between 0 and 1, I should mention here α lies between 0 and 1.

So, generally Google takes this value a damping factor about 0.85. I mean, this has been obtained using, I mean the large, by working with several values and coming up with some, something, which is, which works. So, you can take α to be 0.85.

(Refer Slide Time: 24:10)



The screenshot shows a JupyterLab window with a SageMath 9.1 kernel. The code cell contains the following text:

```

G = alpha * P + (1 - alpha) * S
Here alpha is called the damping factor which tells us how much times a surfer follow the hyperlink that teleporting.
[[22]: S=1/n*matrix([[1]*n]*n) # ones_matrix(QQ,n)/n
show(S)

```

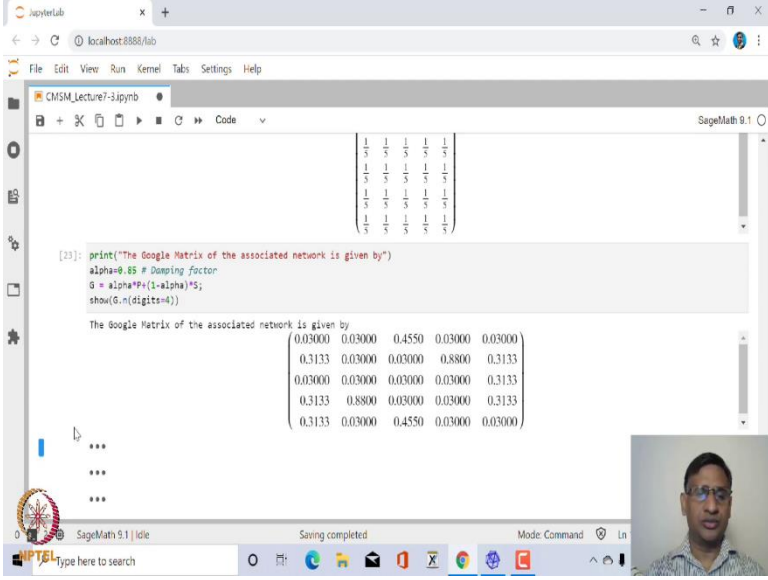
The output of the code is a 5x5 matrix with all entries equal to 1/5:

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix}$$

Below the matrix, there are five lines of three asterisks (***) indicating that the output has been truncated.

So, let us define this matrix S , which is 1 for the above network, which is a loop, this is S .

(Refer Slide Time: 24:22)



```

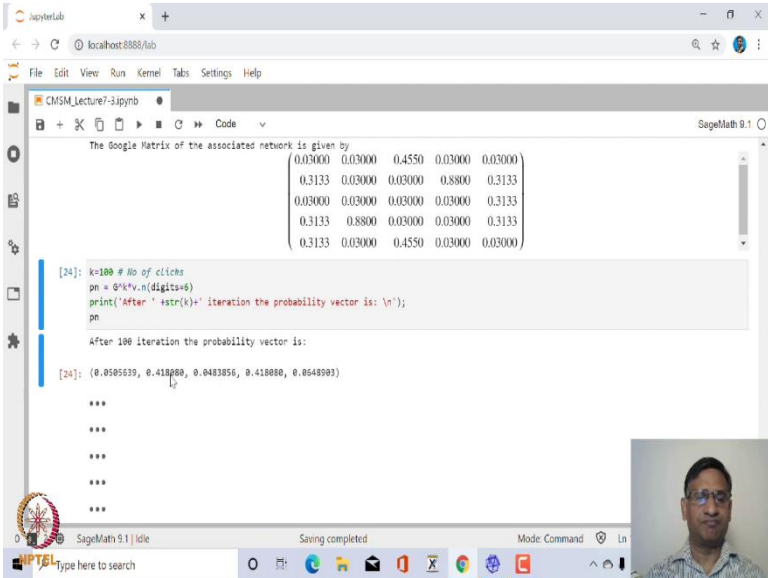
In [23]: print("The Google Matrix of the associated network is given by")
alpha=0.85 # Damping factor
G = alpha*S + (1-alpha)*ones(n,n)/n
show(G.n(digits=4))

The Google Matrix of the associated network is given by
(0.03000 0.03000 0.4550 0.03000 0.03000)
(0.3133 0.03000 0.03000 0.8800 0.3133)
(0.03000 0.03000 0.03000 0.03000 0.3133)
(0.3133 0.8800 0.03000 0.03000 0.3133)
(0.3133 0.03000 0.4550 0.03000 0.03000)

```

And then let us define the, the Google matrix G , which is α times P plus 1 minus α times S , and take α to be 0.85 , which is the damping factor. And then in this case, this matrix reduces to this, or updates to this. Now you can see here, there is no, there is no 0 column.

(Refer Slide Time: 24:42)



```

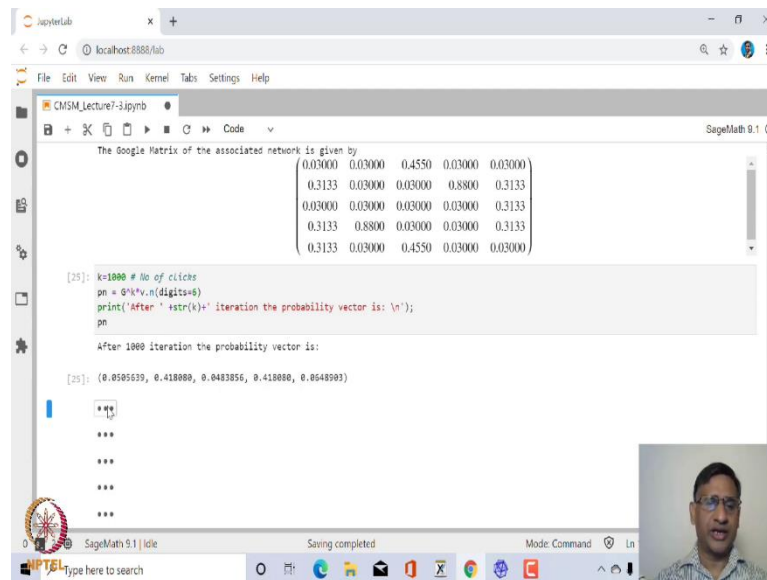
In [24]: k=100 # No of clicks
pn = G^k*v.n(digits=6)
print("After " + str(k) + " iteration the probability vector is: \n");
pn

After 100 iteration the probability vector is:
[24]: (0.0505639, 0.418088, 0.0483856, 0.418088, 0.0648903)

```

Now, in this case, if I, if I look at what happens after, after let us say 100 clicks. So, then you will not get this 0 . Of course, in this case you can see here, the, the probability of this being a webpage 2 is high, and also at 4 is high compared to others, right?

(Refer Slide Time: 25:07)



```
CM/SM_Lecture7-3.ipynb
SageMath 9.1

The Google Matrix of the associated network is given by

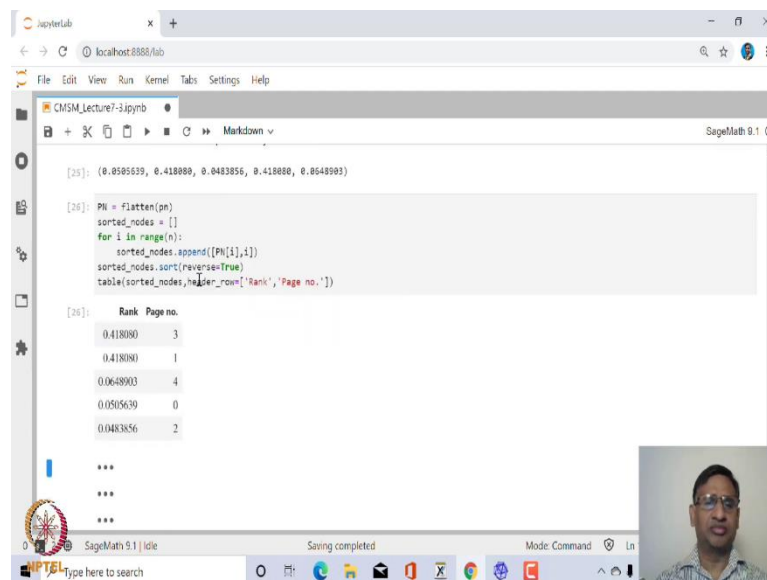
$$\begin{pmatrix} 0.03000 & 0.03000 & 0.4550 & 0.03000 & 0.03000 \\ 0.3133 & 0.03000 & 0.03000 & 0.8800 & 0.3133 \\ 0.03000 & 0.03000 & 0.03000 & 0.03000 & 0.3133 \\ 0.3133 & 0.8800 & 0.03000 & 0.03000 & 0.3133 \\ 0.3133 & 0.03000 & 0.4550 & 0.03000 & 0.03000 \end{pmatrix}$$


[25]: k=1000 # No of clicks
pn = 0.4*v.n(digits=6)
print('After '+str(k)+' iteration the probability vector is: \n');
pn

After 1000 iteration the probability vector is:
[25]: (0.0505639, 0.418080, 0.0483856, 0.418080, 0.0648903)
```

Instead of 100, if you make it 1,000 click, then again you will see that, this, this is what is you get as a stationary vector.

(Refer Slide Time: 25:14)



```
CM/SM_Lecture7-3.ipynb
SageMath 9.1

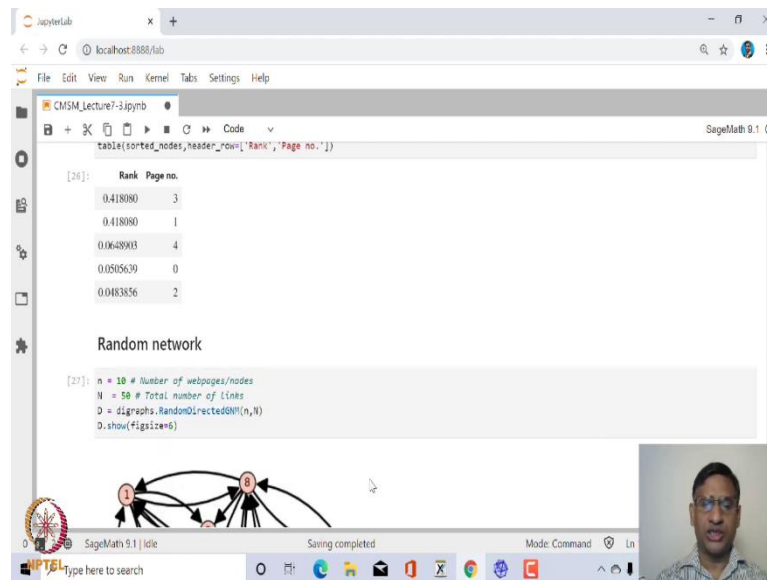
[25]: (0.0505639, 0.418080, 0.0483856, 0.418080, 0.0648903)

[26]: pn = Flatten(pn)
sorted_nodes = []
for i in range(n):
    sorted_nodes.append([pn[i],i])
sorted_nodes.sort(reverse=True)
table(sorted_nodes,header_row=['Rank', 'Page no.'])

[26]:
Rank  Page no.
0.418080  3
0.418080  1
0.0648903  4
0.0505639  0
0.0483856  2
```

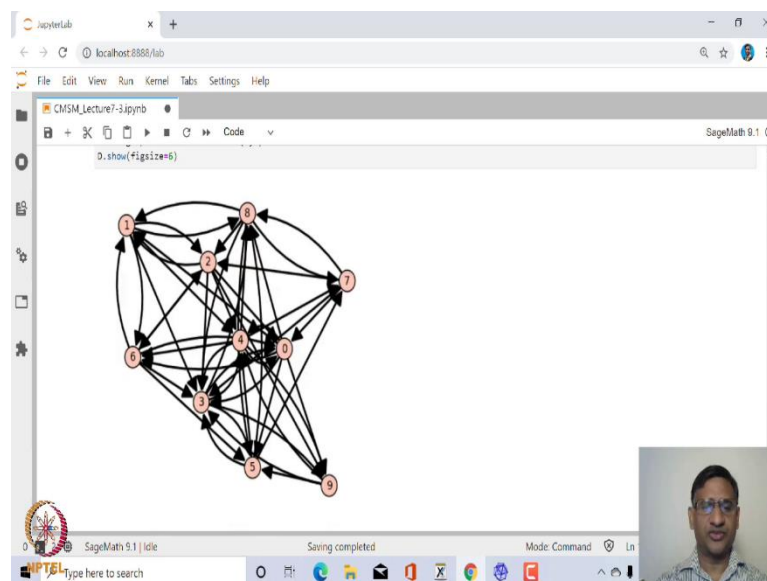
Similarly, and you can rank this, you can rank these things using, there is a, there is a option called sort. So, you can sort this in reverse order, that is decreasing order, and then you can print. This is the rank of all these websites, sorted as decreasing order of the importance and this is a webpage number. So, 3rd website has maximum rank, followed by the next one which is the 1st one and followed by the 4th one, and so on, right?

(Refer Slide Time: 25:54)



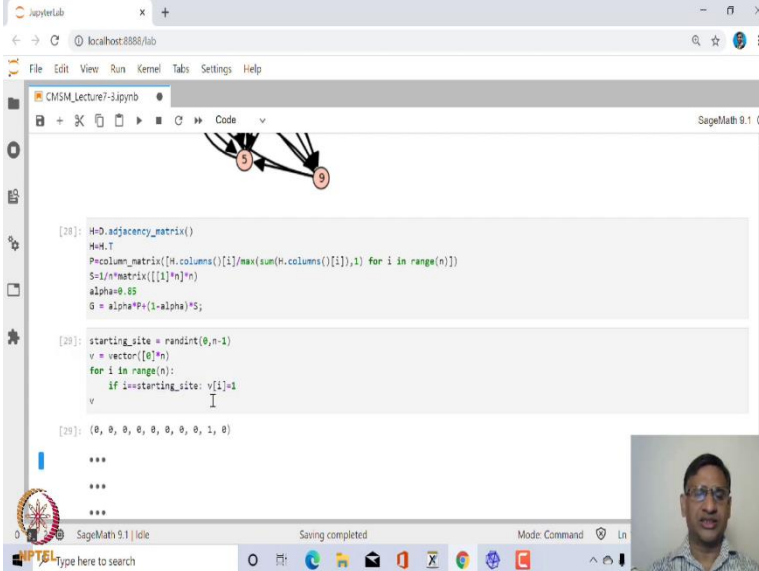
So, now you can, you can look at, for example, if I have a large network ok, if I have large network. So, Sage has inbuilt function to generate a random graph. So, let us say there is a, there is an option called random directed graph, using G, some algorithm which is called GNM. So, inside this digraph you have this option. So, let us take, you have a website of 10 webpages, and then let us say there are 50 links, that means 50 edges. So, you have a random network with 10 websites and 50 links, that is not very big. And let us say the define D to be the, the graph and then let us ask it to show.

(Refer Slide Time: 26:45)



When you show this, this is what you get, it looks somewhat complicated, not too much, right? Now what we can do? We will find out.

(Refer Slide Time: 26:54)



```

[28]: H=adjacency_matrix()
      H=H.T
      P=column_matrix([H.columns()[i]/max(sum(H.columns()[i]),1) for i in range(n)])
      S=1/n*matrix([[1]*n]*n)
      alpha=0.85
      G = alpha*P+(1-alpha)*S;

[29]: starting_site = randint(0,n-1)
      v = vector([0]*n)
      for i in range(n):
          if i==starting_site: v[i]=1
      v

[29]: (0, 0, 0, 0, 0, 0, 0, 1, 0)

***
***
***

```

Let us say adjacency matrix to this and then transition matrix of this website, of this network and then find this matrix S , which is 1 by n for every i and j and take α to be 0.85 and G to be that Google matrix to be α times P plus 1 minus α times S . Now, if you see here, this α is equal to 0.85 , what it means is that about 85 percent of the chance, 85 percent times, the, the user follow this link, and 15 percent of times, he uses either back click or close the website, and go to, to new, new, close the tab and go to new tab, that is what is called teleporting, right? So, let us run this, and then let us again take a random vector. So, what, how we are generating? Take a random integer between 0 and n minus 1 , because index starts from 0 and goes up to n minus 1 , in this case, and take a vector which is 0 vector, and then at i th generate a random, random website number, and replace that i th entry, which is 0 , by 1 . So, in this case, you see here this is the last website, last but one website, if I do once more this is now 4 th website.

So, this is the starting place for a user. The user starts surfing from webpage 4 . Now, let us find out what happens after 100 clicks. So, after a 100 clicks, this is what you get, this is the, the probability of each of this website.

(Refer Slide Time: 28:22)

```

P = column_matrix([H.columns()[i]/max(sum(H.columns()[i]),1) for i in range(n)])
S = 1/n*matrix([1]*n*n)
alpha = 0.85
G = alpha*P + (1-alpha)*S;

[30]: starting_site = randint(0,n-1)
v = vector([0]*n)
for i in range(n):
    if i != starting_site: v[i] = 1
v

[30]: (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)

[31]: k=100 # No of clicks
pn = G^k*v.n(digits=6)
print('After ' + str(k) + ' iteration the probability vector is: \n');
pn

After 100 iteration the probability vector is:

[31]: (0.101325, 0.0698839, 0.0727529, 0.162761, 0.126620, 0.109348, 0.0746444, 0.0736968, 0.123688, 0.0852791)

***

```

And then if you look at the one which has the maximum probability, that will, that is the website which will appear first. So, for example, in this case, this seems to be the, the maximum, no this, this one is the maximum, the 4th website.

(Refer Slide Time: 29:01)

```

pn = G^k*v.n(digits=6)
print('After ' + str(k) + ' iteration the probability vector is: \n');
pn

After 100 iteration the probability vector is:

[31]: (0.101325, 0.0698839, 0.0727529, 0.162761, 0.126620, 0.109348, 0.0746444, 0.0736968, 0.123688, 0.0852791)

[32]: k=1000 # No of clicks
pn = G^k*v.n(digits=6)
print('After ' + str(k) + ' iteration the probability vector is: \n');
pn

After 1000 iteration the probability vector is:

[32]: (0.101325, 0.0698839, 0.0727529, 0.162761, 0.126620, 0.109348, 0.0746444, 0.0736968, 0.123688, 0.0852791)

[33]: P = Flatten(pn)
sorted_nodes = []
for i in range(n):
    sorted_nodes.append([P[i],i])
sorted_nodes.sort(reverse=True)
table(sorted_nodes,header_row=['Rank','Page no.'])

[33]: Rank Page no.

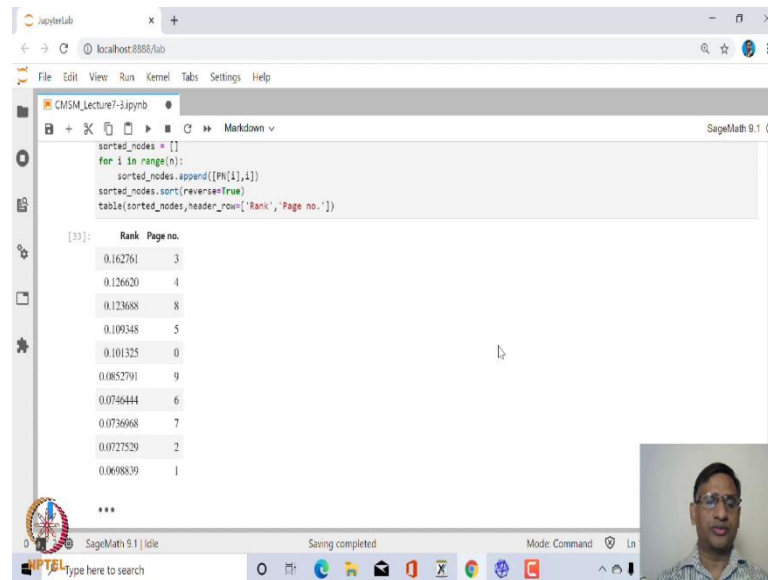
```

And instead of, instead of 100 clicks, suppose if I say, let me just copy and then paste it in the next cell, if I, if I say, let us say for example, 1000 clicks, then again, these, these two are almost similar.

So, this, this, this Google matrix, Google, you can check that Google matrix is again a stochastic matrix, it will be column stochastic matrix. In fact, it is a regular column

stochastic matrix, therefore, the n th power, as n becomes large and large, this will converge and this will give you the stationary vector, which is eigenvector corresponding to eigenvalue 1. So, this is what you get. So, this converges.

(Refer Slide Time: 29:48)



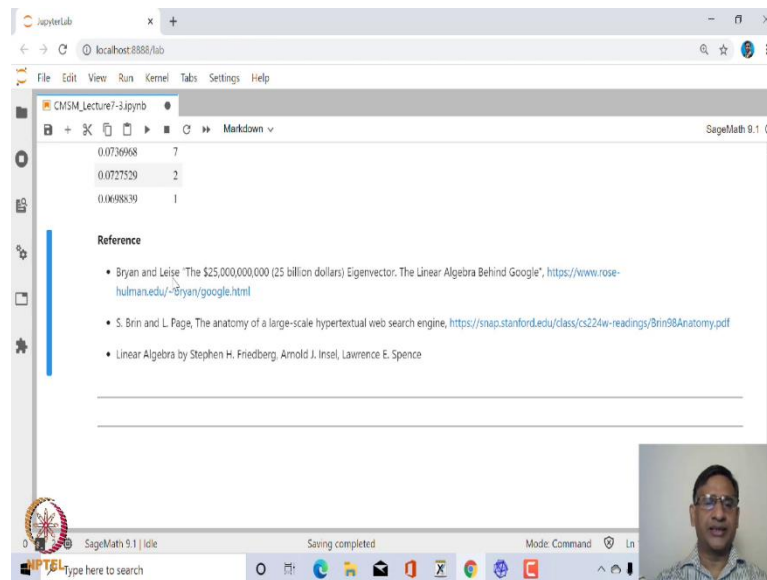
```
sorted_nodes = []
for i in range(n):
    sorted_nodes.append([Pi[1],1])
sorted_nodes.sort(reverse=True)
table(sorted_nodes,header_row=['Rank','Page no.'])
```

Rank	Page no.
0.162761	3
0.126620	4
0.123608	8
0.109348	5
0.101325	0
0.0852791	9
0.0746444	6
0.0736968	7
0.0727529	2
0.0698839	1

And you can again sort this and arrange this in decreasing order of the rank. So, 3rd website has maximum rank, followed by the 4th, that followed by the 8th, and so on. The 1st website has the least rank; the 2nd website has a least rank, this is the 4th website, actually. 3 index 3 is 4th website, 4th website has link. So, I think I should say here, ok. This does not matter anyway, because this link which we have generated, it says there is one 0th website. So, this is, this is how you can rank the webpages and this is how this Google search work, with a, with a simple notion of eigenvalues, eigenvector.

So, there are, there are lots of other applications of linear algebra. In fact, linear algebra is full of applications in science and engineering, but we covered some of it, not all of them. And you can refer to books, some standard books on linear algebra, it will give you some other applications. For example, one book which you can refer, is a book by Anton. This is linear algebra and its applications and this has a chapter of about 20 applications, right?

(Refer Slide Time: 31:10)



For this, more on Google search algorithm, you can look at this, this particular article by Bryan and Leise, which says the 25 dollars eigenvectors, linear algebra behind the Google, right? Or you can also look at the original paper by Brin and Paige who invented this Google search engine, and this is also available here.

You can also look at, for example, for the convergence of this, stochastic matrices etcetera. You can look at this book on linear algebra by Stephen Friedberg and Arnold Insel and then Lawrence, Spence, ok? So, these, these are some of the references for studying more concepts in linear algebra and some of its applications, ok? So, let me stop here. From next lecture onwards, we will start looking at some concepts in numerical methods.

Thank you very much.