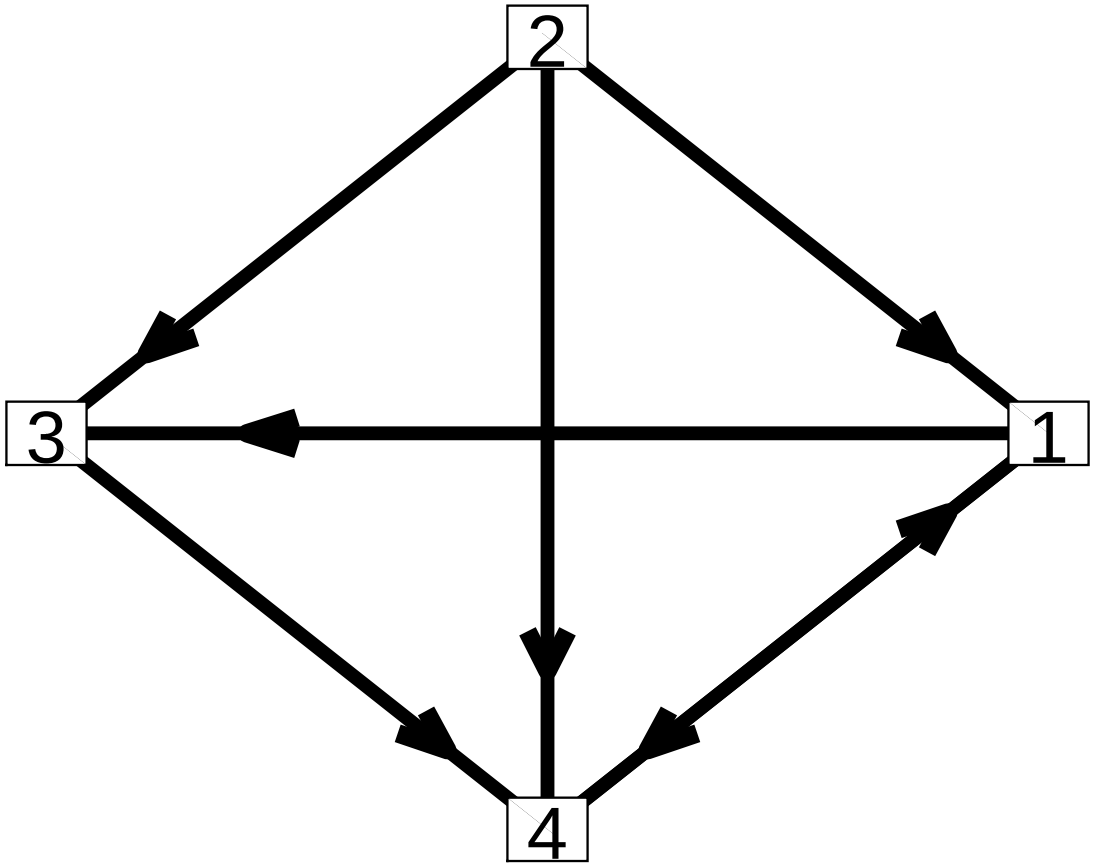
The google PageRank algorithm, session 4



Adjacency matrix.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

Let d_i denote the **out-degree** for node i. In this example $d_1 = 2$, $d_2 = 3$, $d_3 = 1$ and $d_4 = 1$.

The out-degrees are used when constructing the conditional probability matrix C.

The conditional probability matrix C in the basic model

Adjacency matrix.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

The out-degrees are the column sums and these are $d_1 = 2$, $d_2 = 3$, $d_3 = 1$ and $d_4 = 1$. The probability matrix C is

$$C = \begin{pmatrix} 0 & 1/3 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1/2 & 1/3 & 0 & 0 \\ 1/2 & 1/3 & 1 & 0 \end{pmatrix}$$

The entries in each column have been divided by the out-degree associated with the column. With this matrix this is okay as all the out-degrees are greater than 0.

The conditional probability matrix C in the extended model

In the extended model a parameter $\alpha \in [0, 1]$ is introduced such that there is a probability of α that the next node is randomly chosen and $(1 - \alpha)$ is the probability that a link is followed. In this version the entries of $C = (c_{ij})$ are as follows.

$$c_{ij} = \begin{cases} 0, & i = j, \\ \dfrac{\alpha}{N-1} + (1-\alpha)\dfrac{a_{ij}}{d_j}, & d_j > 0 \text{ and } i \neq j. \\ \dfrac{1}{N-1} & d_j = 0 \text{ and } i \neq j. \end{cases}$$

A function file to get C given A and α – vectorised

```
function C = cmat2_mkw(A, alpha)

N = size(A, 1);

outdeg=sum(A);

C = zeros(N, N);

for from=1:N
    if outdeg(from)==0
      C(:, from)=1/(N-1);
    else
      C(:, from)=(1-alpha)*A(:, from)/outdeg(from)+...
                 alpha/(N-1);
    end
    C(from, from)=0;
end
```

A function file to get C given A and α – componentwise

```
function C = cmat2(A, alpha)

N = size(A, 1);
outdeg=sum(A);

C = zeros(N, N);
for from=1:N
    for to=1:N
        if from==to
            C(to, from) = 0;
        elseif outdeg(from)>0
            C(to, from) = alpha/(N-1) ...
                +(1-alpha)*A(to, from)/outdeg(from);
        else
            C(to, from) = 1/(N-1);
        end
    end
end
```

Saving the points and then plotting curves

```
% ..earlier statements not shown
N=size(A, 1);
pp=zeros(N, iter);
pp(:, 1)=p;

% create the points and show something
for i=2:iter
    pp(:, i) = C*pp(:, i-1);
end

figure(12);
x=1:iter;
plot(x, pp(1, :), x, pp(2, :), x, pp(3, :), x, pp(4, :));
legend('P1', 'P2', 'P3', 'P4');
xlabel('iteration');
```

For networks with more than N nodes this is likely to be one of the preferred ways of displaying how things change from one iteration to another.

Saving the points and using hold

```
% ..statements as before
figure(15);
clf
hold on
x=1:iter;
plot(x, pp(1, :), '^r', 'MarkerSize', 6);
plot(x, pp(2, :), 'ok', 'MarkerSize', 6);
plot(x, pp(3, :), '+b', 'MarkerSize', 6);
plot(x, pp(4, :), '*k', 'MarkerSize', 6);
legend('P1', 'P2', 'P3', 'P4');
xlabel('iteration');
hold off
```

We can only use this when N is small as we do not have enough marker types.

The steady state – an eigenvector

Mathematically we get each new probability by multiplying by C and in the Matlab program we have

```
p=C*p;
```

We overwrite the old `p` with the next `p`.

The examples suggest that there is a limit as the number of iterations tends to ∞. Mathematically this vector is such that

$$C\underline{p} = \underline{p}$$

with $\underline{p} = (p_i)$, $p_i \geq 0$ and

$$p_1 + p_2 + \cdots + p_N = 1.$$

$\underline{p}$ is an **eigenvector** of C with **eigenvalue** 1 normalised in the above way.

Other ways of writing things

$$C\underline{p} = \underline{p} \quad \text{is} \quad (C - I)\underline{p} = \underline{0}.$$

The last case is a homogeneous system with a non-trivial solution.

Once way to getting the specific solution that we want is to construct the augmented system.

$$\begin{pmatrix} -1 & c_{12} & \cdots & c_{1N} \\ c_{21} & -1 & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & -1 \\ 1 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

The augmented matrix has size $(N+1) \times N$. Although there are more equations than unknowns it can be shown that there is a unique solution.

Implementing in Matlab ...direct1.m

The script file direct1.m is approximately as follows.

```
fprintf('Adjacency matrix for network 7 follows\n')
A = [ 0 1 0 1 ;
      0 0 0 0 ;
      1 1 0 0 ;
      1 1 1 0 ]

alpha = 0.15;
C = cmat2(A, alpha);

N = size(A, 2);
M = [ C - eye(N, N);
      ones(1, N) ];
b = [ zeros(N, 1);
           1 ];

% use \ to get p and show p
fprintf('alpha=%f, probability vector p follows\n', alpha)
p = M\b
```

Sorting the entries in p

The entries in the vector p indicate the rankings of the N nodes. It would be good if Matlab can sort these for us rather than do this by hand.

```
% ..statements as before
fprintf('alpha=%f, probability vector p follows\n', alpha)
p = M\b
[a, o]=sort(p, 'descend');
for k=1:N
  j=o(k);
  fprintf('Node %2d, position=%2d, prob=%6.3f\n', ...
          j, k, p(j));
end
```

In the assignment you will have a network of between 10 and 15 nodes.

The output from the sorting

```
Adjacency matrix for network 7 follows
A =
     0     1     0     1
     0     0     0     0
     1     1     0     0
     1     1     1     0
alpha=0.150000, probability vector p follows
p =
    0.3661
    0.0476
    0.2087
    0.3776
Node  4, position= 1, prob= 0.378
Node  1, position= 2, prob= 0.366
Node  3, position= 3, prob= 0.209
Node  2, position= 4, prob= 0.048
```

Assessment of project 2 – dates etc

- Group poster to submit by 15:30 on Wed 22 March 2017.
- Group interview during the 10–11 meeting on Thu 23 March.
- Individual interview during the 10–11 meeting on Thu 23 March.
- Individual submission by Mon 3rd April 2017. The precise details to be made available in week 26.

Assessment of project 2 tasks

- For the group poster you need 10–15 mathematical topics which have some links between some of them.
- You need to go through the list and consider which topic is linked to any of the other topics and have a reason why.
- Once you have your N topics and the links between them you need to create the $N \times N$ adjacency matrix and appropriately include it in your version of "`direct1.m`". You need to run it to get the rankings when $\alpha = 0.15$.
- Your need to create a poster which has a picture of the network, the adjacency matrix, an explanation for some of the links and the rankings that the model generates.

Any aspect of the poster or your version of `direct1.m` may lead to a question in the interviews.

The challenge question

The rankings depend on α and at the moment you take $\alpha = 0.15$. Can you modify the programs and/or add statements to the Matlab program to show how each component of p varies as α varies? I will ask you about your progress on this at the interview.

The things that you need to consider are as follows.

- How do you run the program for many different values of α?
- How do you show the results for each value of α or how do you collect the results first and display later?

Hint: Consider again the graphics commands used earlier in this handout.

The poster and the challenge question

The interview part will be about the poster created and the attempts so far to the challenge question. The attempts on the challenge question also need to be sent to me on Wednesday 23rd March.

Some tips

- Take care that the adjacency matrix matches the network diagram. You may wish to use the function file `from_to.m` that I have used in the sessions. I can send you this file.

- If it easier to enter the transpose then you can have

  ```
  B=[ ..numbers for the transpose];
  A=B'
  ```

- I would consider more topics than the N that you have in the group poster. You may be asked later what you would add if the network is extended.

More advanced Matlab that may be used

If you want Matlab to handle an array of names and the associated probabilities then you can use a cell array to deal with mixed data. In the case of countries and populations the set-up can be as follows.

```
pop={...
'Bangladesh', 162221000,...
'Brazil',     192540000,...
'China',     1336070000,...
'India',     1177592000,...
'Indonesia',  231369500,...
'Japan',      127430000,...
'Mexico',     107550697,...
'Nigeria',    154729000,...
'Pakistan',   168840500,...
'Russia',     141927297,...
'USA',        308765000};

p=cell2mat( pop(2:2:end) );
N=length(p);
```

Sorting by population

Next we use `sort()` to determine the order vector o.

```
% .. as above
p=cell2mat( pop(2:2:end) );
N=length(p);
[v, o]=sort(p, 'descend');
for i=1:N
  j=o(i);
  fprintf('%10d  %s\n', p(j), pop{2*j-1});
end
```

The sorted output

Note that we refer to the cell array entry using the brackets `{}` in the `fprintf` statement

```
fprintf('%10d  %s\n', p(j), pop{2*j-1});
```

The output generated is as follows.

```
1336070000  China
1177592000  India
 308765000  USA
 231369500  Indonesia
 192540000  Brazil
 168840500  Pakistan
 162221000  Bangladesh
 154729000  Nigeria
 141927297  Russia
 127430000  Japan
 107550697  Mexico
```