大学数学实验

Mathematical Experiments

第6讲

数值计算V：科学计算中的基本概念

科学计算研究的核心问题

算法的构造

算法的分析

构造算法的基本手段:近似

研究算法的核心问题:近似对计算结果的影响

浮点数

尾数　　阶码

2进制数(d_1=1)　　$\pm .d_1 d_2 \cdots d_t \times 2^s \quad (L \le s \le U)$

(非零)浮点数 f　　$m \le |f| \le M$

$$m = 2^{L-1}, \quad M = 2^U \sum_{i=1}^{t} 2^{-i} = 2^U (1 - 2^{-t})$$

IEEE标准(754): 2进制(Float)

	符号	尾数	阶码	位移(Bias)
single	1	23	8	127
double	1	52	11	1023

(阶码全0或全1保留)

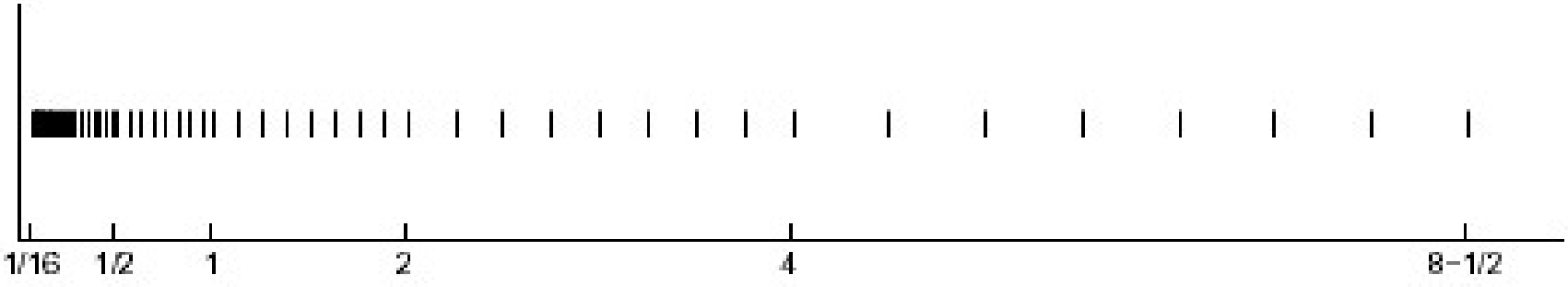
$-127 \le s \le 127$

$-1023 \le s \le 1023$

浮点数

$$\pm .d_1 d_2 \cdots d_t \times 2^s \quad (L \le s \le U)$$

取 t=4，L=-3，U=3，(正)浮点数的集合为

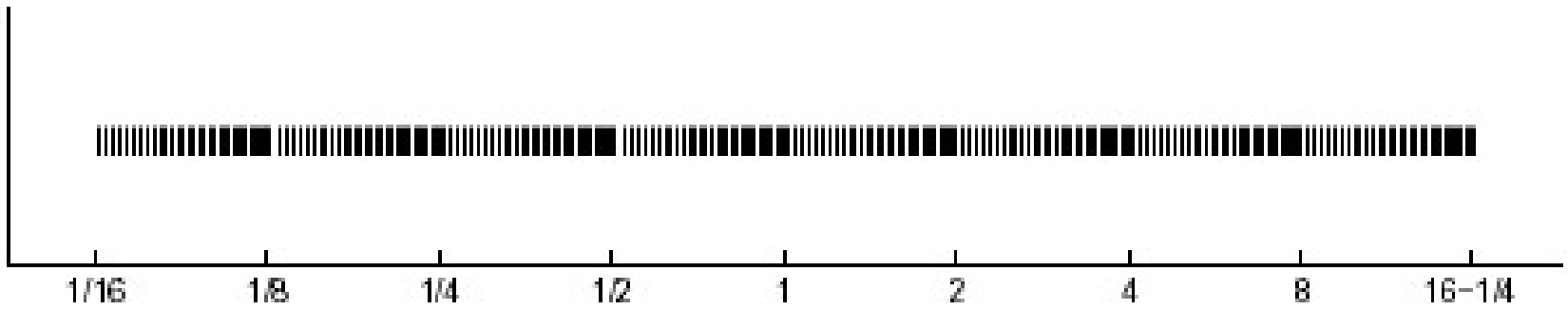


$$m = 2^{L-1} = 1/16, \quad M = 2^U (1 - 2^{-t}) = 8(1 - 1/16) = 7.5$$

特点：分布不均匀　　但各处相对误差一样

浮点数

- **如果取t=6，L=-3，U=4，这时采用对数坐标，则集合 F （正数部分）为**



能够精确表达的数总是有限的！

MATLAB　　eps = 2^{-52} = 2.2204e-016

realmin=2^{-1022}=2.2251e-308

realmax=(2-eps)*2^{1023}=1.7977e+308

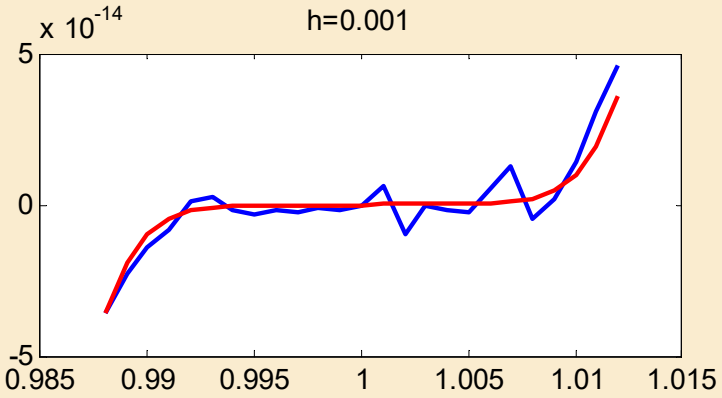
- format long
- a=4/3-1;
- b=a*3-1

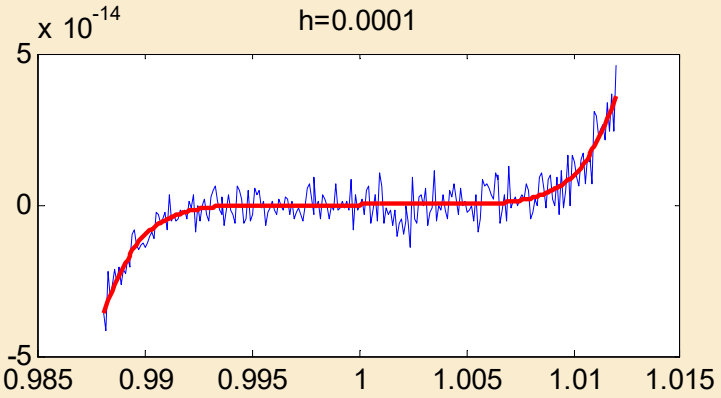
- b =
- -2.220446049250313e-16

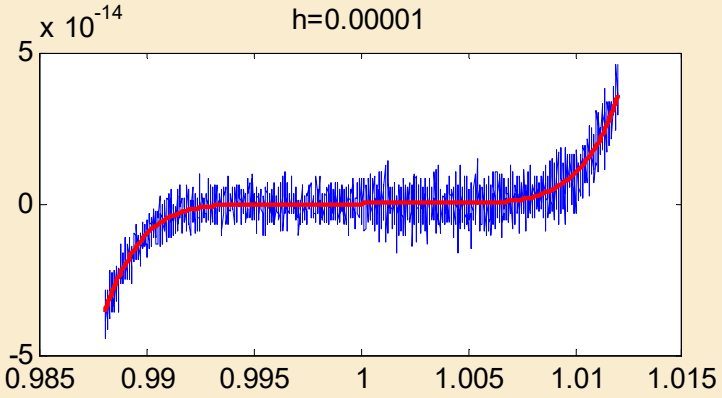
3*(4/3-1)-1

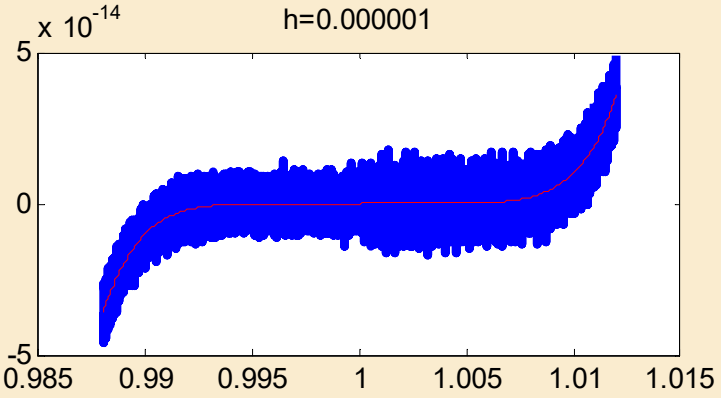
浮点数运算--- example (MATLAB)

$$(x-1)^7 \quad \text{VS} \quad x^7-7x^6+21x^5-35x^4+35x^3-21x^2+7x-1$$









浮点数运算不满足交换律

```
% 基本算数运算律验证I，加法交换律，向量化表示
clear
n=10000;  A=rand(1,n);  m=1e10;

tic; x1=0;  for i=1:n    x1=x1+A(i);  end
x1=n*x1; x1=x1+m; t(1)=toc;

tic; x2=m;
for j=1:n    for i=1:n      x2=x2+A(i);    end;  end
t(2)=toc;

tic; for k=1:100   x3=n*sum(A)+m; end; t(3)=toc/100;

[x1, x2, x3],x3-[x1,x2],t
```

浮点数运算不满足交换律

英国著名数值分析学家 Higham (1998):

Can you count on computers?

例：把4开n次方，再平方n次，结果是4？存在误差？

$$\left(\left(\cdots\sqrt{\sqrt{4}}\right)^2\cdots\right)^2 = ?$$

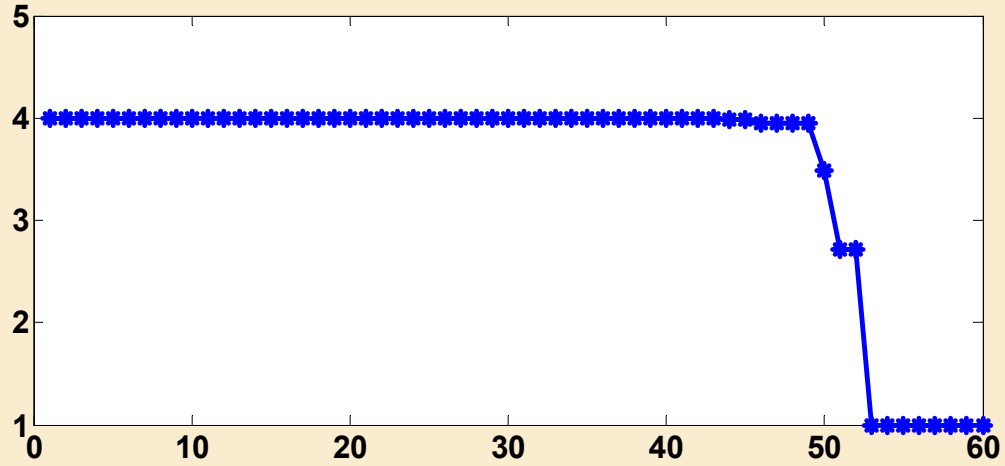
n=55左右：结果变成1

浮点运算：舍入误差

精确计算	近似计算
解析结果	数值结果
(Analytical)	**(Numerical)**

浮点数运算实例

$$\left(\left(\cdots\sqrt{\sqrt{4}}\right)^2\cdots\right)^2 = ?$$



$m = 40, \quad a = 3.999744390658126$

$m = 50, \quad a = 3.490342924893666$

$m = 52, \quad a = 2.718281808182473$

$m = 53, \quad a = 1$

科学计算中的基本概念

针对算法

- 复杂度(收敛性、计算量)
  -----误差估计和分析
  -----收敛速度
- 稳定性

针对问题

- 病态性

研究的出发点: 误差 !!

复杂度（收敛性，计算量）

- 数值积分（步长选择）

  p阶收敛

- 常微分方程初值问题（步长选择）

  p阶精度，刚性问题

- 线性方程组和非线性方程组收敛性与收敛速度

  线性、超线性、平方收敛

- 线性方程组直接法，迭代法

  Gauss消去$O(n^3)$，迭代矩阵的谱半径

稳定性

-----刻划算法的关键概念

- **考虑如下的序列**

$$E_n = \int_0^1 x^n e^x \, dx, \quad n = 1,2,\cdots$$

- **可以证明**

$$E_n = e - nE_{n-1}$$

$$0 < E_n < e/(n+1)$$

两个算法

----有什么差别，哪个可以用??

Algorithm 1

$$E_1 = 1$$

$$E_n = e - nE_{n-1}$$

$$n = 2,3,\cdots$$

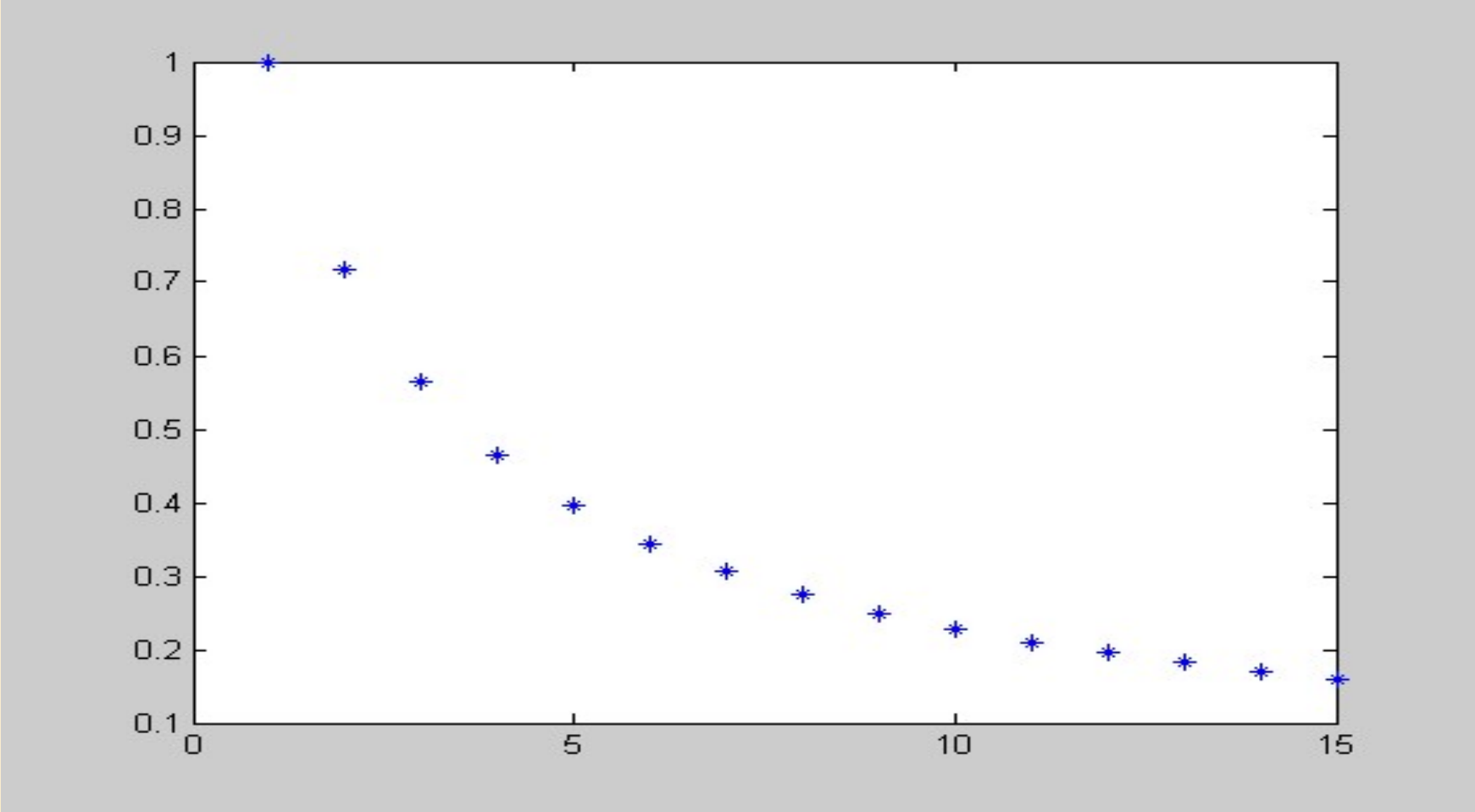
Algorithm 2

$$E_N = 0,$$

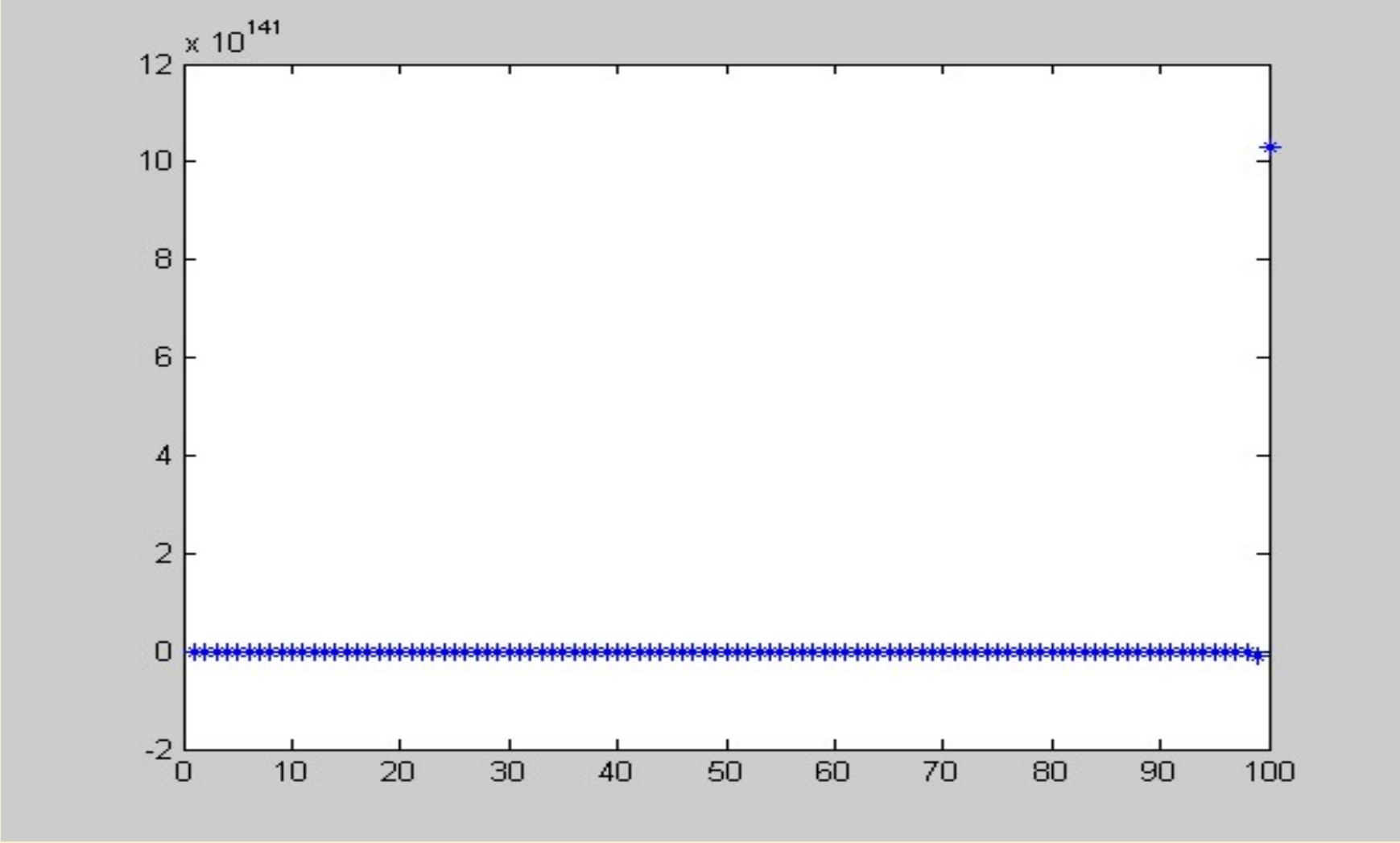
$$E_{n-1} = (e - E_n)/n$$

$$n = N, N-1, \cdots 2,1$$

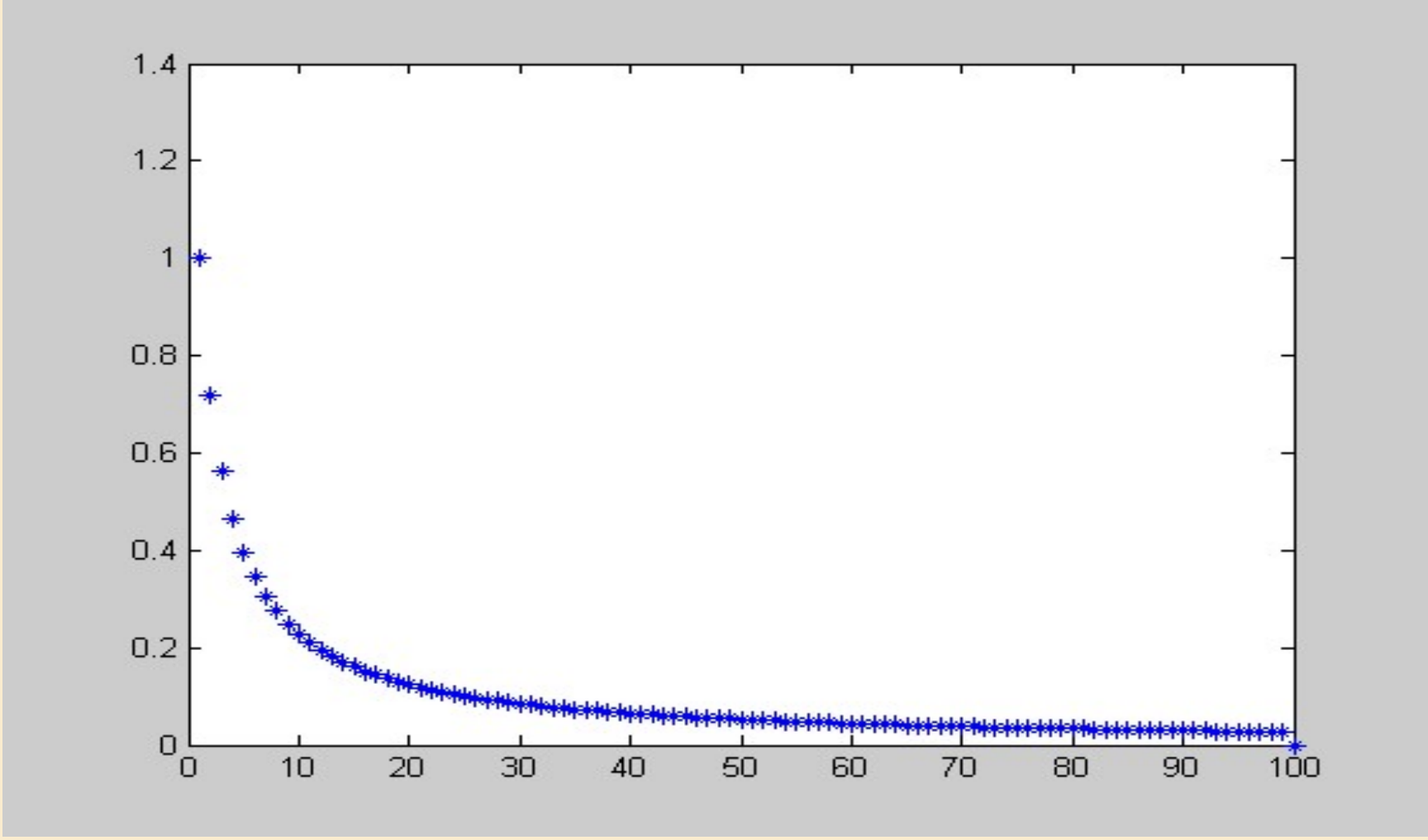
Algorithm 1 with n=15



Algorithm 1 with n=100



Algorithm 2 with n=100



科学结论的取得，不能依靠感觉

- 简单的计算发现，可以使用的算法是--

  Algorithm 2 !

- 计算中误差并不可怕，重要的是误差在算法中的传播。
- 稳定----算法中产生的任何误差，对后续计算的影响是衰减或可以控制的。
- 不稳定的算法=不能用的垃圾！

常微分方程初值问题单步法的稳定性

对初值问题 $\begin{cases} y' = f(x,y), & x \in (x_0, b], \\ y(x_0) = a, \end{cases}$ 的显式单步法为

$$y_0 = a, \quad y_{n+1} = y_n + h\,\varphi(x_n, y_n; h)$$

算法的稳定性即舍入误差的传播问题

如果在求解过程中,误差不增长,则称该算法是 稳定的

考虑试验方程 $y' = -\lambda y \quad (\lambda > 0)$

例如: 用 Euler 法求解方程 $y' = -2y$

$$y_{n+1} = y_n + h(-2y_n) = (1-2h)\,y_n$$

如果 y_n 的误差是 ε_n, 则由上述公式产生的 y_{n+1} 的误差

$$\varepsilon_{n+1} = (1-2h)\,\varepsilon_n$$

当 $|1-2h| < 1$ 时, Euler 法解此方程的稳定的

常微分方程初值问题单步法的稳定性

向前欧拉公式

$$y_{n+1}=y_n+hf(x_n,y_n)=(1-h\lambda)y_n \Rightarrow \varepsilon_{n+1}=(1-\lambda h)\varepsilon_n$$

$$|\varepsilon_{n+k}|\le|\varepsilon_n| \Rightarrow |1-h\lambda|\le 1 \Rightarrow h\le 2/\lambda$$

向后欧拉公式

$$y_{n+1}=y_n-h\lambda y_{n+1} \Rightarrow \varepsilon_{n+1}=\frac{1}{1+h\lambda}\varepsilon_n \Rightarrow h\text{任意}$$

经典龙格-库塔公式 $h\le 2.785/\lambda$

刚性现象与刚性方程

现象 振动系统或电路系统的数学模型

$$\ddot{x}+k\dot{x}+rx=f(t)\ (k,r>0), x(0)=a, \dot{x}(0)=b$$

k=2000.5, r=1000, a=1, b=-1999.5, $f(t)$=1

$$x(t)=e^{-2000t}-e^{-t/2}+1$$

瞬态解与稳态解

e^{-2000t} ~快瞬态解　　计算到t=0.005时已衰减到4.5×10^{-5}

$e^{-t/2}$ ~慢瞬态解　　计算到t=20时才衰减到4.5×10^{-5}

求稳态解

精度达到10^{-4}需算到t=20(由慢瞬态解λ=1/2决定)

选取步长h由快瞬态解λ=2000决定

龙格-库塔公式　h<2.785/2000=0.0014　t=20需14286步

快、慢瞬态解的特征根相差悬殊 ⇒ 刚性现象(Stiff)

刚性现象与刚性方程

刚性方程

振动、电路及化学反应中的线性常系数方程组

$$\dot{x}(t) = Ax + f(t),\ \ x \in R^n$$

A的特征根λ_k (k=1,2,…,n) 的实部$Re(\lambda_k)$<0

$|\mathrm{Re}(\lambda_k)|$大 ~快瞬态解

$|\mathrm{Re}(\lambda_k)|$小 ~慢瞬态解

$$s = \frac{\max\limits_k |\mathrm{Re}(\lambda_k)|}{\min\limits_k |\mathrm{Re}(\lambda_k)|}$$

~刚性比

s>10 ⇒ 刚性方程

快瞬态解→步长充分小

慢瞬态解→积分区间长

⇒ 传统的数值方法无能为力

刚性非线性方程组 ⟹ 线性常系数方程组

线性化方法

刚性现象与刚性方程

刚性方程的MATLAB求解

ode23, ode45 解刚性方程的困难

步长自动变小　　计算时间很长

求解刚性方程的命令: ode23s, ode15s 等 （**用法相同**）

例
$$\begin{cases} \dot{x}_1 = x_1 + 2x_2 \\ \dot{x}_2 = -(10^6+1)x_1 - (10^6+2)x_2 \\ x_1(0) = 10^6/4, \ x_2(0) = 10^6/4 - 1/2 \end{cases}$$

特征根λ_1=-1,λ_2=-10^6
刚性比 s=10^6

解析解
$$\begin{cases} x_1(t) = (\frac{10^6}{4}+1)e^{-t} - e^{-10^6 t} \\ x_2(t) = -(\frac{10^6}{4}+1)e^{-t} + \frac{(10^6+1)}{2}e^{-10^6 t} \end{cases}$$

Stiff.m,stiff1.m

列主元消去法示例

例：3阶方程组 $\begin{bmatrix} 0.001 & 2.000 & 3.000 \\ -1.000 & 3.712 & 4.623 \\ -2.000 & 1.072 & 5.643 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.000 \\ 2.000 \\ 3.000 \end{bmatrix}$

四位有效数字精确解为 $x^* = (-0.4904, -0.05104, 0.3675)^T$

解：（1）高斯消去法

$$(A|b) = \begin{bmatrix} 0.001 & 2.000 & 3.000 & 1.000 \\ -1.000 & 3.712 & 4.623 & 2.000 \\ -2.000 & 1.072 & 5.643 & 3.000 \end{bmatrix} \xrightarrow[m_{22}=-2000]{m_{21}=-1000}$$

$$\begin{bmatrix} 0.001 & 2.000 & 3.000 & 1.000 \\ 0 & 2004 & 3005 & 1002 \\ 0 & 4001 & 6006 & 2003 \end{bmatrix} \xrightarrow{m_{32}=1.997} \begin{bmatrix} 0.001 & 2.000 & 3.000 & 1.000 \\ 0 & 2004 & 3005 & 1002 \\ 0 & 0 & 5.000 & 2.000 \end{bmatrix}$$

$\bar{x} = (-0.400, -0.09989, 0.4000)^T$

列主元消去法示例

（2）交换行，避免绝对值小的主元作除数。（列主元素法）

$$(A\,|\,b)=\begin{bmatrix}-2.000 & 1.072 & 5.643 & 3.000\\ -1.000 & 3.712 & 4.623 & 2.000\\ 0.001 & 2.000 & 3.000 & 1.000\end{bmatrix}\xrightarrow[m_{22}=-0.0005]{m_{21}=0.5000}$$

$$\begin{bmatrix}-2.000 & 1.072 & 5.643 & 3.000\\ 0 & 3.712 & 1.801 & 0.500\\ 0 & 2.001 & 3.003 & 1.002\end{bmatrix}\xrightarrow{m_{32}=0.6300}$$

$$\begin{bmatrix}-2.000 & 1.072 & 5.643 & 3.000\\ 0 & 3.712 & 1.801 & 0.500\\ 0 & 0 & 1.868 & 0.687\end{bmatrix}$$

$$x=(-0.4900,-0.05113,0.3678)^T$$

列主元消去法

$$[A^{(k)} \mid b^{(k)}] = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ & & \ddots & \vdots & & \vdots & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & b_k^{(k)} \\ & & & \vdots & & \vdots & \vdots \\ & & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{bmatrix}$$

- 选 $\left|a_{i_k k}^{(k)}\right| = \max\limits_{k \le i \le n} \left|a_{ik}^{(k)}\right|$
- 交换第k行和第i_k行
- 等价于 PA=LU， 其中P是一个排列阵

病态性

-----刻划模型的概念

- 考虑如下的问题

f(x)=(x-1)(x-2).....(x-20)

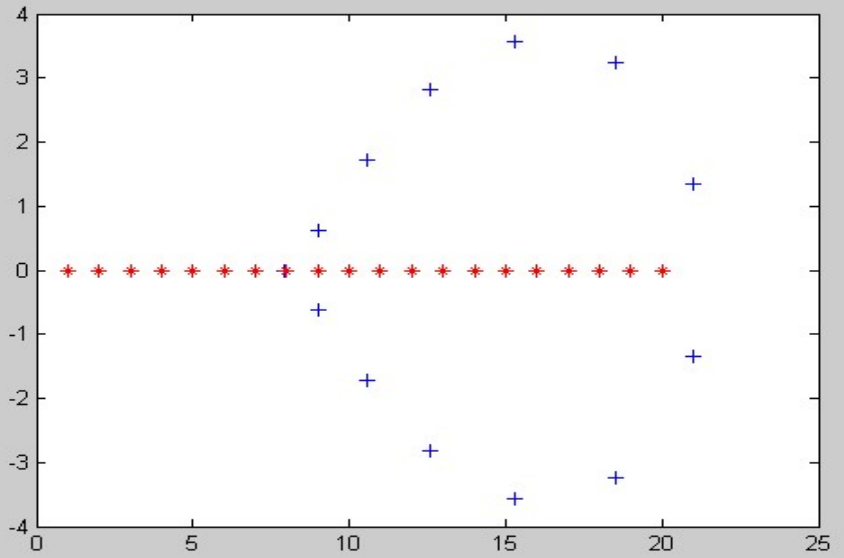
显然方程 **f(x)=0** 的解是

1 2 3 4 ..... 19 20

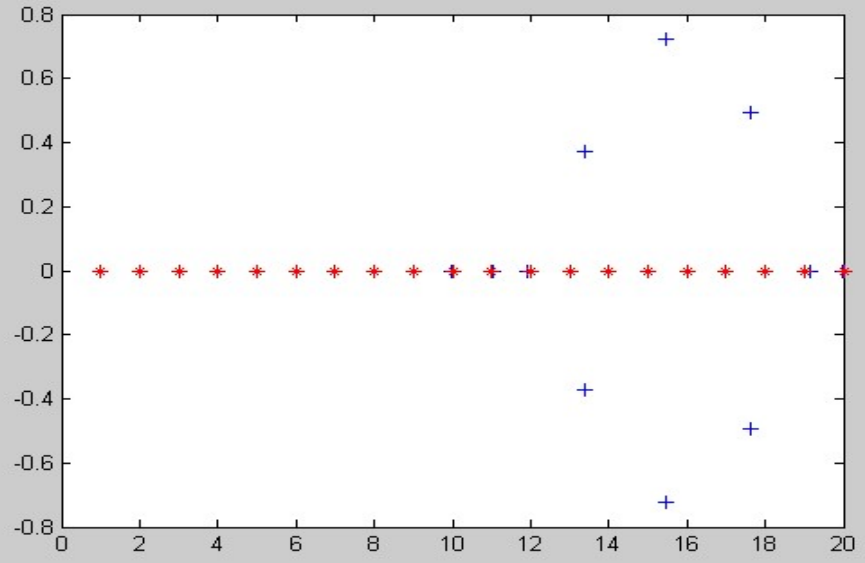
请问: 如下方程的解是什么?

$$f_\varepsilon(x) = f(x) + \varepsilon x^{18} = 0$$

ε=10e-5



ε=10e-8



较小的根不敏感

较大的根较敏感

病态！！

直 接 法 – 误 差 分 析 $Ax=b$

$$\begin{bmatrix}1 & 1\\ 1 & 1.01\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}2\\ 2\end{bmatrix} \Rightarrow \begin{bmatrix}1 & 1\\ 1 & 1.01\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}2\\ 2.01\end{bmatrix}$$

$$\Rightarrow \quad x=\begin{bmatrix}2\\ 0\end{bmatrix} \qquad \Rightarrow \quad x=\begin{bmatrix}1\\ 1\end{bmatrix}$$

x对b的扰动敏感

$Ax=b$，如果解 x 对 b 或 A 的扰动敏感，就称方程组是病态的，也称系数矩阵 A 是病态的。

定理： A 为 n 阶矩阵, $\det(A)\neq 0$, x 和 $x+\delta x$ 满足方程

$$Ax=b$$

$$(A+\delta A)(x+\delta x)=b+\delta b$$

其中 $b\neq 0$, 且 $\|\delta A\|$ 适当小, 使 $\dfrac{\|\delta A\|}{\|A\|}<\dfrac{1}{cond(A)}$

(即 $\|\delta A\|\|A^{-1}\|<1$), 则有

$$\frac{\|\delta x\|}{\|x\|}\leq\frac{1}{1-\|A^{-1}\|\|\delta A\|}cond(A)\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right)$$

剩余向量与近似解的关系

定理: A 为 n 阶矩阵, $\det(A)\neq 0$, $b\neq 0$, x 是方程 $Ax=b$ 的精确解, $\bar{x}$ 是方程组的一个近似解, 对应近似解 $\bar{x}$ 的剩余向量定义为 $r=b-A\bar{x}$, 则

$$\frac{1}{cond(A)}\frac{\|r\|}{\|b\|}\leq\frac{\|\bar{x}-x\|}{\|x\|}\leq cond(A)\frac{\|r\|}{\|b\|}$$

大学数学实验

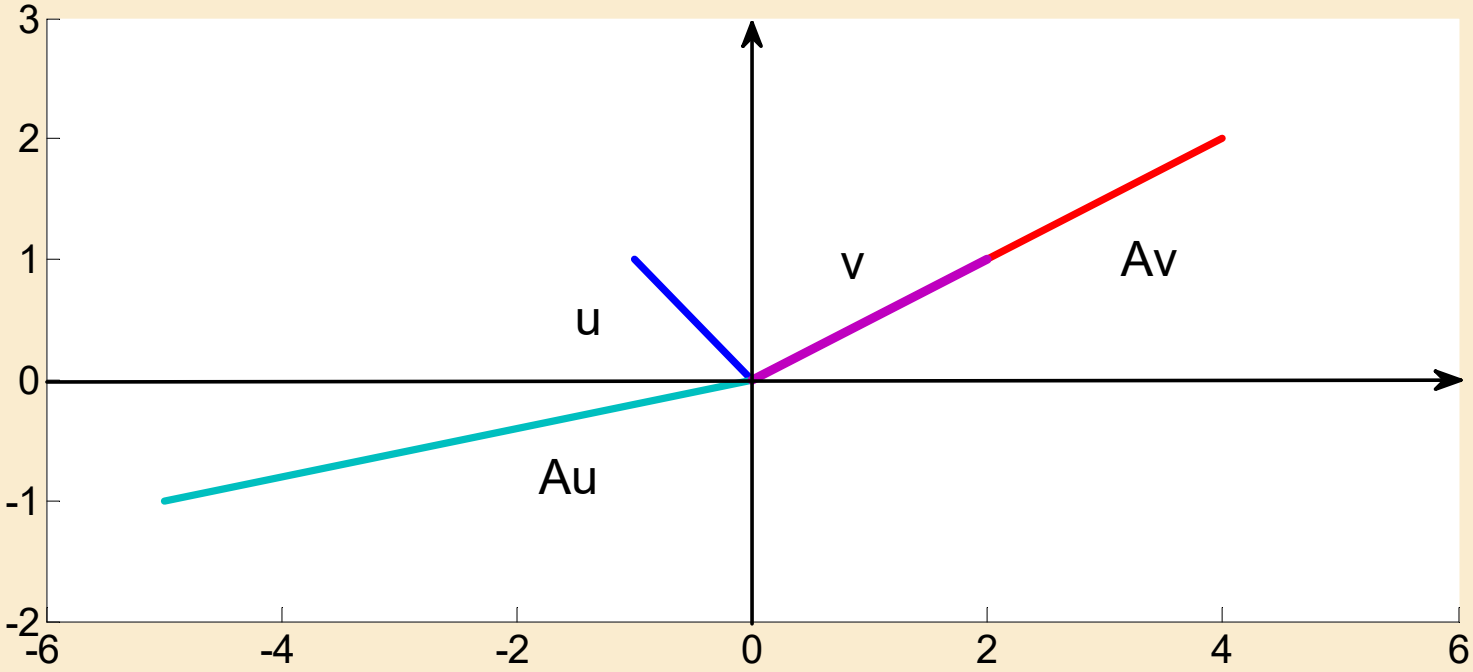
Experiments in Mathematics

第6讲*

数值计算V* 矩阵特征值问题

$$Ax = \lambda x \qquad P(\lambda) = \det(\lambda I - A) = 0$$

$$A = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}, \quad u = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad v = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



Computing the Google PageRank

The World's Largest Eigenvalue Problem

What is PageRank

A search with Google's search engine usually returns a very large number of pages. E.g., a search on 'weather forecast' returns 5.5 million pages.

Although the search returns several million pages, the most relevant pages are usually found within the top ten or twenty pages in the list of results.

How does the search engine know which pages are the most important?

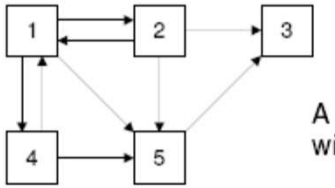
Google assigns a number to each individual page, expressing its importance. This number is known as the PageRank and is computed via the eigenvalue problem

$$Pw=\lambda w$$

where P is based on the link structure of the Internet.

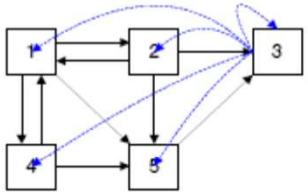
The key problem is to formulate the link structure, i.e., the matrix P, in a proper way.

The Link Structure Matrix *P*



A tiny internet with 5 pages.

The model forming the basis of the PageRank algorithm is a random walk through all the pages of the Internet. Let $p_t(x)$ denote the possibility of being on page x at time t. The PageRank of page x is expressed as $\lim(p_t(x))$ for $t \to \infty$. To make sure the random walk process does not get stuck, pages with no out-links (here: page 3) are assigned artificial links or "teleporters" to all other pages.



$$P = \begin{pmatrix} 0 & 1/3 & 1/5 & 1/2 & 0 \\ 1/3 & 0 & 1/5 & 0 & 0 \\ 0 & 1/3 & 1/5 & 0 & 1 \\ 1/3 & 0 & 1/5 & 0 & 0 \\ 1/3 & 1/3 & 1/5 & 1/2 & 0 \end{pmatrix}$$

The matrix P is irreducible and stochastic and therefore the random walk can be expressed as a Markov chain, and the PageRank of all pages can be computed as the principal eigenvector of P.

The PageRank Algorithm

The Google matrix P is currently of size 4.2×10^9 and therefore the eigenvalue computation is not trivial. To find an approximation of the principal eigenvector the *power method* is used:

```
w_0 = initial guess
For k = 1 to 50
      w_k = P*w_{k-1}
End
Return w_50
```

The special properties of the matrix P ensures that the largest eigenvalue is $\lambda = 1$, rendering normalisation in the power method unnecessary. Fast convergence of the power method makes 50 iterations adequate.

Because the computation involves an extremely large matrix, the matrix-vector multiplications must be implemented in parallel on multi-processor systems.

DTU

数 学 建 模 实 例

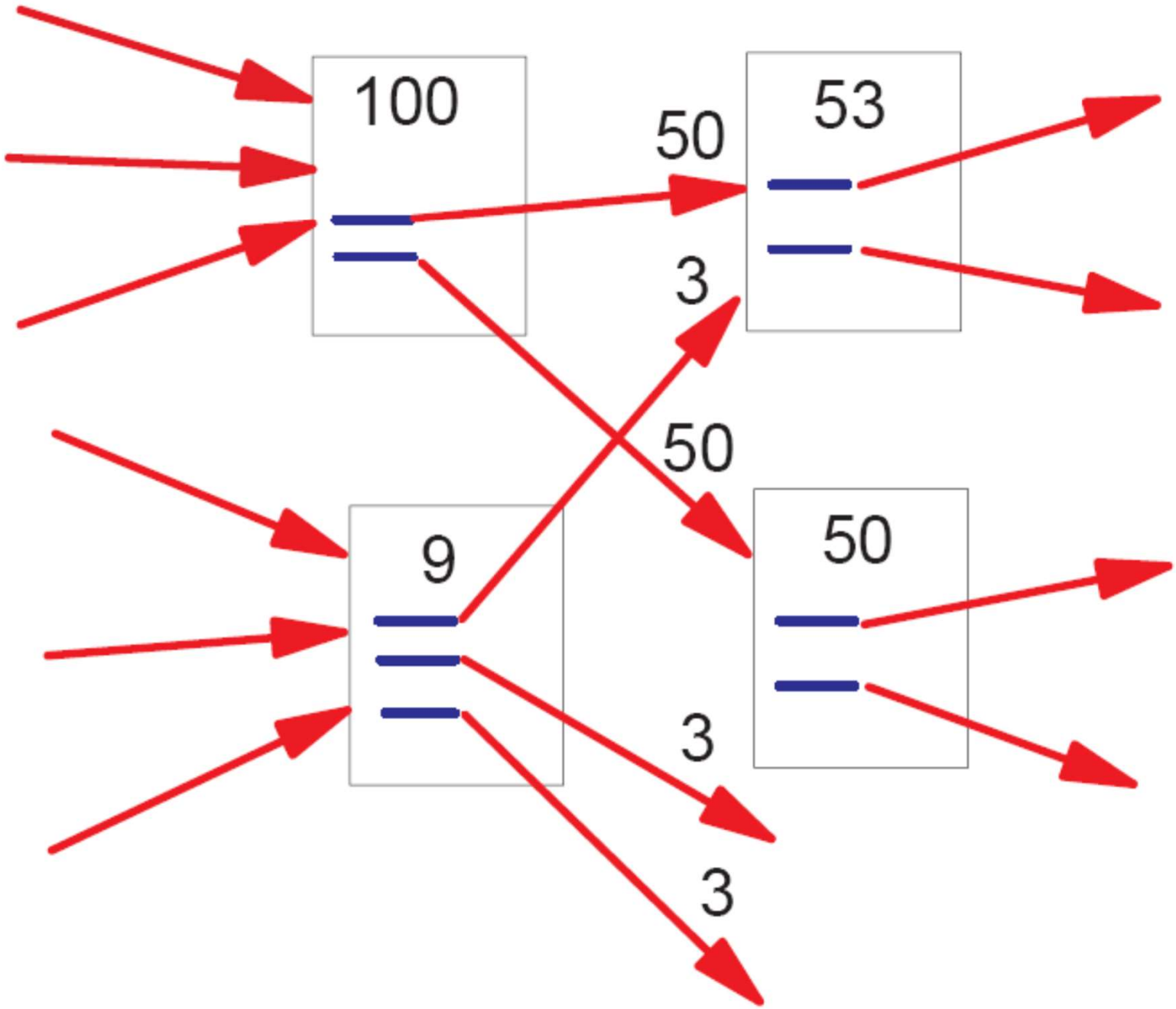
PageRank: Ranking Web Pages

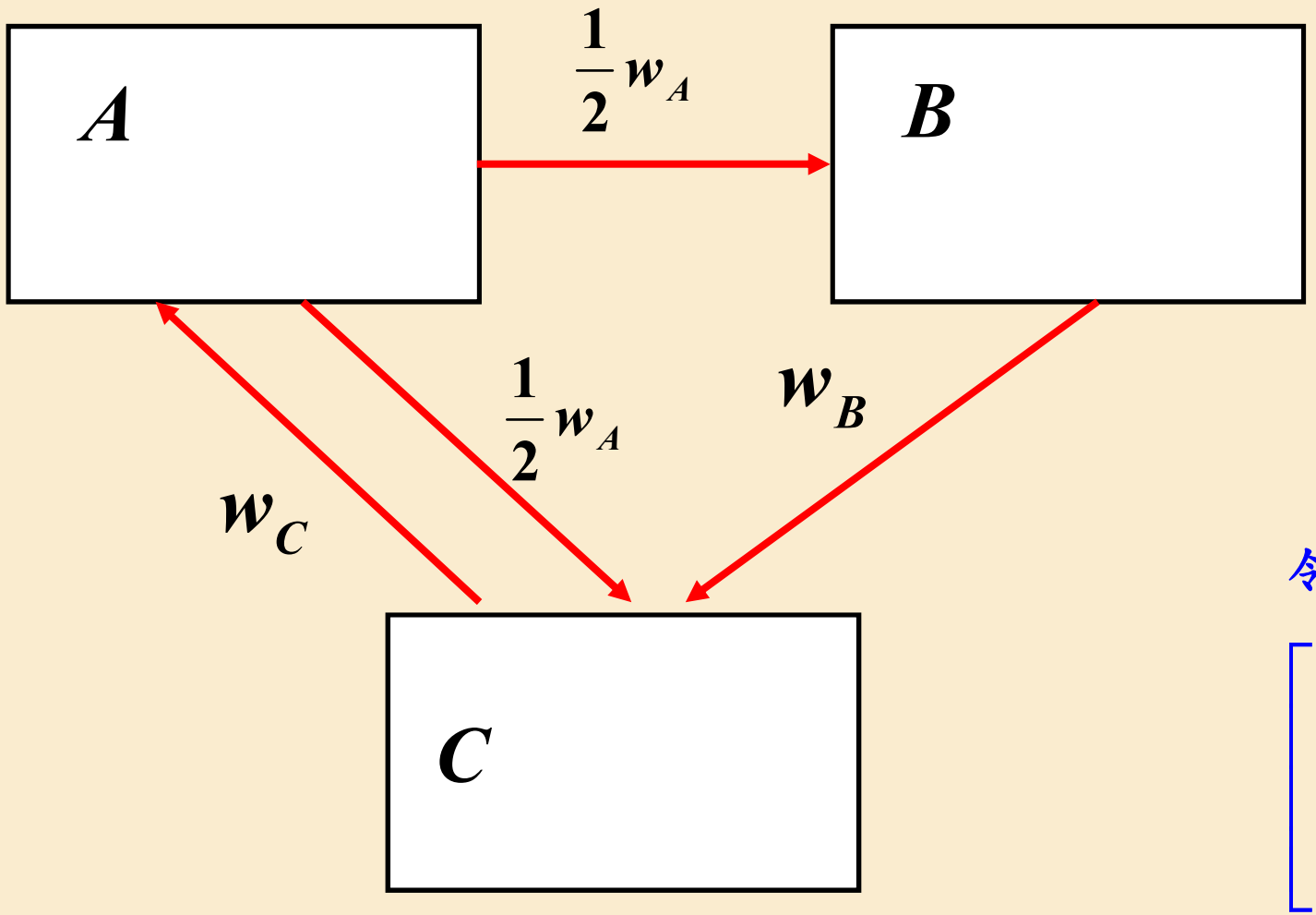
Google™

PageRank模型的原始思想：依赖于网页的拓扑结构——网页间的相互链接关系

数学工具：Markov链，矩阵特征值

Cartoon illustrating basic principle of PageRank. The size of each face is proportional to the total size of the other faces which are pointing to it.

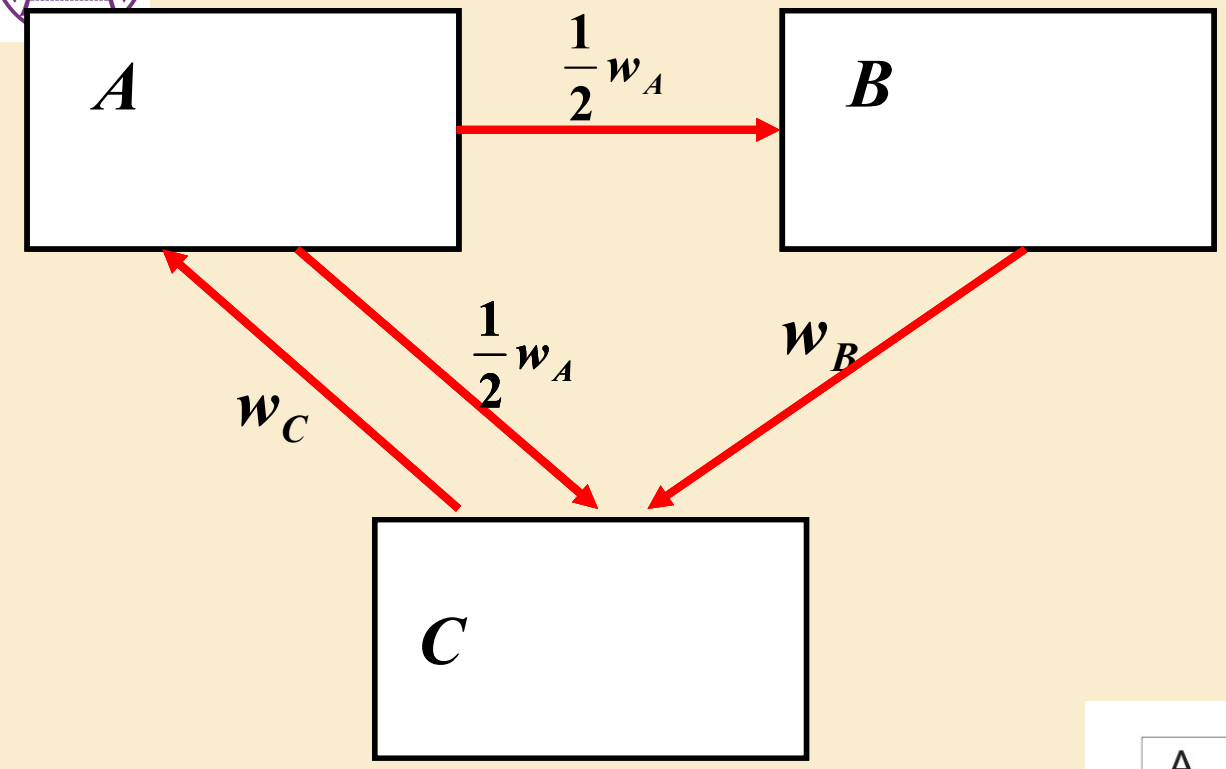

100
50
53
3
50
9
50
3
3



邻接矩阵

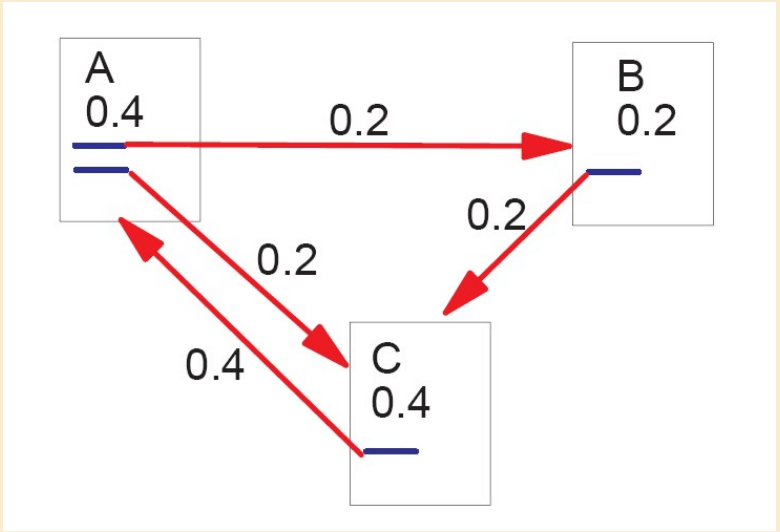
$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

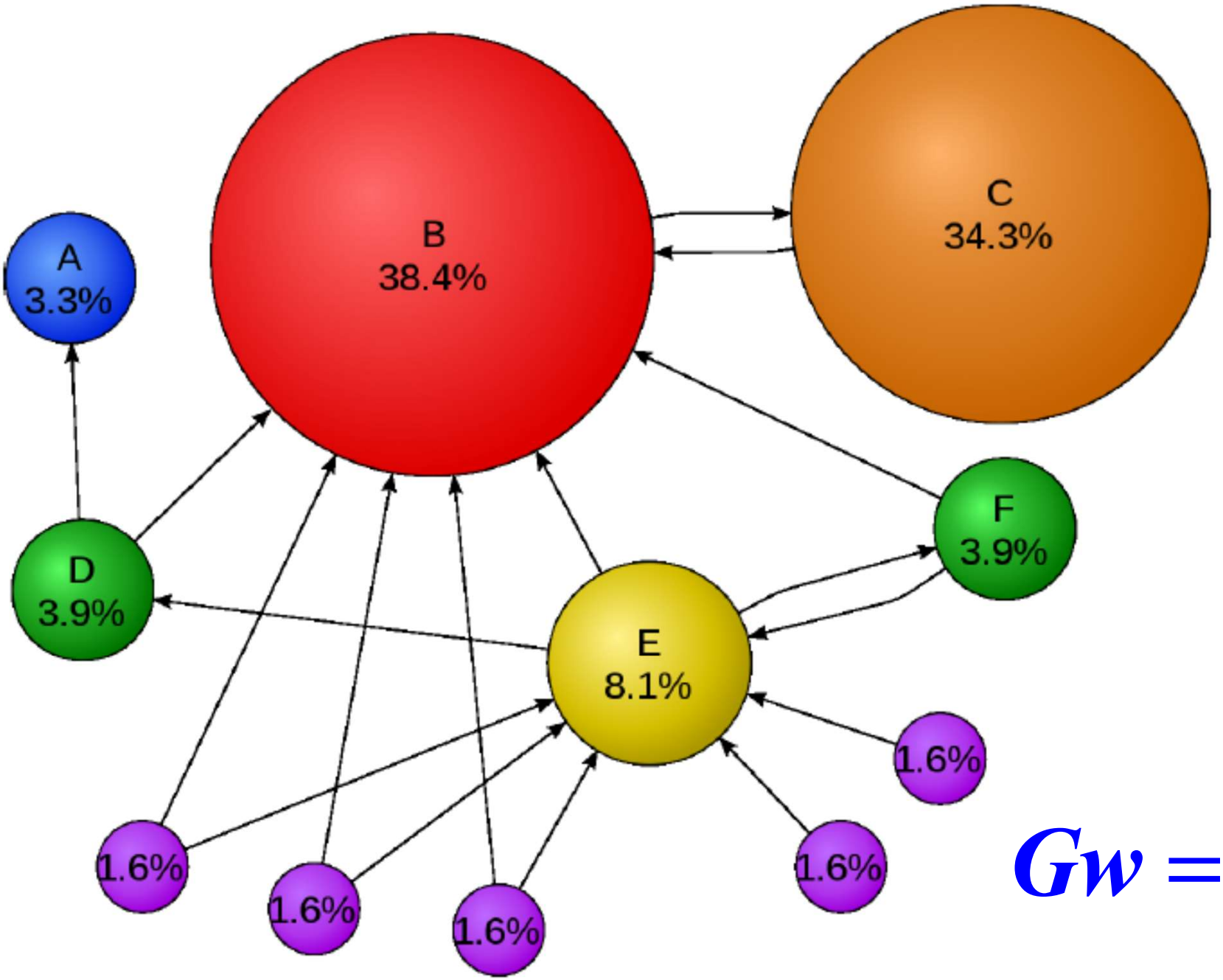
1	1/3	1/3	1/3
2	1/3	1/6	1/2
3	1/2	1/6	1/3
4	1/3	1/4	5/12
5	5/12	1/6	5/12
6	5/12	5/24	3/8
7	3/8	5/24	5/12
8	5/12	3/16	19/48
9	19/48	5/24	19/48
10	19/48	19/96	13/32



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

$$\begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{bmatrix} \times \begin{bmatrix} w_A \\ w_B \\ w_C \end{bmatrix} = \begin{bmatrix} w_A \\ w_B \\ w_C \end{bmatrix} \Rightarrow$$





$$Gw = w$$

```
%pagerank计算实例
A=zeros(11);
A(2,3)=1;A(3,2)=1;A(4,[1 2])=1;A(5,[2 4 6])=1;A(6,[2 5])=1;
A(7,[2 5])=1;A(8,[2 5])=1;A(9,[2 5])=1;A(10,[5])=1;A(11,[5])=1;

b=sum(A,2); b(2:11)=1./b(2:11); G=diag(b)*A;

b=ones(1,11)/11;
b=b*G; b=b/sum(b)   % 循环

G=0.85*G;  G=0.15*ones(11)/11+G;   G(1,:)=ones(1,11)/11;

b=ones(1,11)/11;
b=b*G    % 循环

[v,e]=eig(G');    v=v(:,1); v=v/sum(v)
```

Random Surfer Model

- 访问者可能会随意打开一个网页,即打开网页有一定的随机性
- PageRank $R(u) = d \bullet \sum_{v \in B_u} \frac{R(v)}{N_v} + (1-d) \bullet \frac{1}{N}$
- N为网页节点总数，d为衰减系数（0<d<1）
- 网页的PageRank值仅d部分在它所链接到的网页中分配，剩下的部分在整个网络的所有网页中分配。
- d的值通常为0.85左右。

矩阵特征值：幂法

设 $A \in R^{n\times n}$，有n个线性无关的特征向量（即A可对角化）

设 A的特征值$\lambda_1, \lambda_2, \cdots, \lambda_n$满足

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

且相应的特征向量为 $u_1, u_2, \cdots, u_n$

幂法(基本想法)

任取非零向量 $x_0 \in R^n$,

$$x_0 = c_1u_1 + c_2u_2 + \cdots + c_nu_n$$

$$x_1 = Ax_0 = c_1\lambda_1u_1 + c_2\lambda_2u_2 + \cdots + c_n\lambda_nu_n$$

$\cdots$

$$x_n = Ax_{n-1} = A^nx_0 = c_1\lambda_1{}^nu_1 + c_2\lambda_2{}^nu_2 + \cdots + c_n\lambda_n{}^nu_n$$

设$c_1 \neq 0$, 则

$$x_n = \lambda_1{}^n\left(c_1u_1 + c_2\frac{\lambda_2{}^n}{\lambda_1{}^n}u_2 + \cdots + c_n\frac{\lambda_n{}^n}{\lambda_1{}^n}u_n\right) \to \lambda_1{}^nc_1u_1$$

$$\lambda_1 \approx max\{x_n\} \Big/ max\{x_{n-1}\}$$

- 计算实例

$$\boldsymbol{A} = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix} \quad \text{and} \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1}$$

$$\boldsymbol{ratio} = \frac{\|\boldsymbol{x}_k\|_\infty}{\|\boldsymbol{x}_{k-1}\|_\infty}$$

k	$\boldsymbol{x}_k^T$		Ratio
0	0.0	1.0	
1	0.5	1.5	1.500
2	1.5	2.5	1.667
3	3.5	4.5	1.800
4	7.5	8.5	1.889
5	15.5	16.5	1.941
6	31.5	32.5	1.970
7	63.5	64.5	1.985
8	127.5	128.5	1.992

标准化的幂法

任取非零向量 $x_0 \in R^n$,

$$\begin{cases} y_1 = Ax_0, & x_1 = \dfrac{y_1}{\|y_1\|_\infty} = \dfrac{Ax_0}{\|Ax_0\|_\infty} \\ y_2 = Ax_1, & x_2 = \dfrac{y_2}{\|y_2\|_\infty} = \dfrac{A^2x_0}{\|A^2x_0\|_\infty} \\ \vdots & \vdots \\ y_k = Ax_{k-1}, & x_k = \dfrac{y_k}{\|y_k\|_\infty} = \dfrac{A^kx_0}{\|A^kx_0\|_\infty} \end{cases}$$

$$\|y_k\|_\infty \to \lambda_1$$

- 幂法实例

$$\boldsymbol{A} = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix} \quad \text{and} \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_{k-1},$$

$$\boldsymbol{x}_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_\infty$$

$$\|\boldsymbol{y}_k\|_\infty \to |\lambda_1|$$

$$\boldsymbol{x}_k \to \boldsymbol{u}_1 / \|\boldsymbol{u}_1\|_\infty$$

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	0.333	1.0	1.500
2	0.600	1.0	1.667
3	0.778	1.0	1.800
4	0.882	1.0	1.889
5	0.939	1.0	1.941
6	0.969	1.0	1.970
7	0.984	1.0	1.985
8	0.992	1.0	1.992

QR方法

$$A = A_1 \in R^{n\times n}$$

$$\begin{cases} A_k = Q_k R_k, & Q_k\text{为正交矩阵，}\ R_k\text{为上三角矩阵} \\ A_{k+1} = R_k Q_k, & k = 1,2,\cdots \end{cases}$$

(1) A_{k+1} 相似于 A_k, 即 $A_{k+1} = Q_k{}^T A_k Q_k$

(2) $A_{k+1} = (Q_1 Q_2 \cdots Q_k)^T A_1 (Q_1 Q_2 \cdots Q_k) = \tilde{Q}_k{}^T A_1 \tilde{Q}_k$

(3) $A^k = (Q_1 Q_2 \cdots Q_k)(R_k \cdots R_2 R_1) = \tilde{Q}_k \tilde{R}_k$

利用正交化过程可以得到可逆矩阵的QR分解.

设 $A=(\alpha_1, \alpha_2,\ldots, \alpha_n)$ 是一个 n 阶可逆矩阵, 则 A 的列向量组 $\alpha_1, \alpha_2,\ldots, \alpha_n$ 构成 R^n 的基, 利用施密特正交化方法可以得到标准正交基 $\gamma_1, \gamma_2,\ldots, \gamma_n$. 在施密特正交化过程, 有

$$\alpha_1=\beta_1, \qquad \alpha_2=\frac{(\alpha_2,\beta_1)}{(\beta_1,\beta_1)}\beta_1+\beta_2,$$

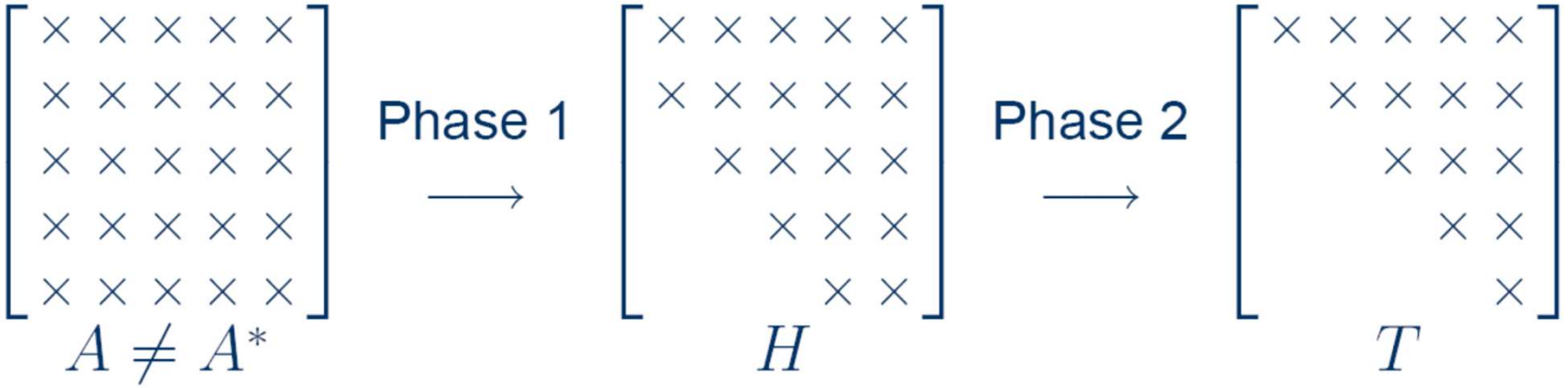
$$\alpha_3=\frac{(\alpha_3,\beta_1)}{(\beta_1,\beta_1)}\beta_1+\frac{(\alpha_3,\beta_2)}{(\beta_2,\beta_2)}\beta_2+\beta_3,\cdots,$$

$$\alpha_n=\frac{(\alpha_n,\beta_1)}{(\beta_1,\beta_1)}\beta_1+\frac{(\alpha_n,\beta_2)}{(\beta_2,\beta_2)}\beta_2+\cdots+\frac{(\alpha_n,\beta_{n-1})}{(\beta_{n-1},\beta_{n-1})}\beta_{n-1}+\beta_n,$$

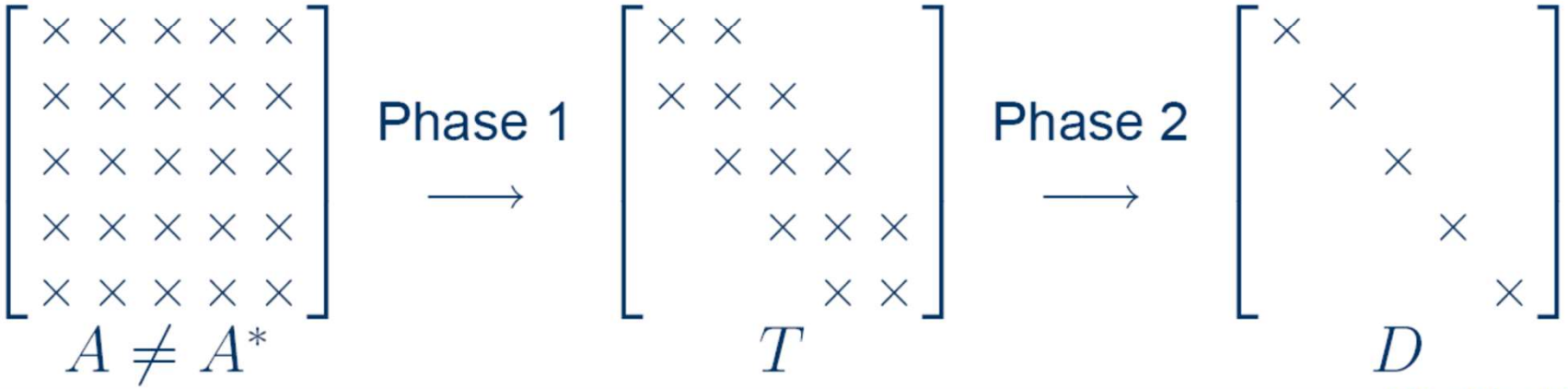
$$(\alpha_1,\alpha_2,\cdots,\alpha_n)=(\beta_1,\beta_2,\cdots,\beta_n)\begin{bmatrix}1 & & * \\ & 1 & \\ & & \ddots \\ & & & 1\end{bmatrix}=(\gamma_1,\gamma_2,\cdots,\gamma_n)\begin{bmatrix}\|\beta_1\| & & & * \\ & \|\beta_2\| & & \\ & & \ddots & \\ & & & \|\beta_n\|\end{bmatrix}.$$

QR方法实现的基本思想

General A: First to *upper-Hessenberg* form, then to upper-triangular



Hermitian A: First to *tridiagonal* form, then to diagonal



为什么要化为Hessenberg型

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1^*} \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}$$

A $\qquad$ Q_1^*A $\qquad$ $Q_1^*AQ_1$

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1^*} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

A $\qquad$ Q_1^*A $\qquad$ $Q_1^*AQ_1$

Hessenberg型

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1^*} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & \times & \times & \times \end{bmatrix}$$

$$Q_1^* A Q_1 \qquad Q_2^* Q_1^* A Q_1 \qquad Q_2^* Q_1^* A Q_1 Q_2$$

$$\underbrace{Q_{m-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{m-2}}_{Q} = H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

对上Hessenberg矩阵做QR分解

$$A_1 = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{bmatrix} \longrightarrow \Omega_{12}A_1 = A_2 = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & x & x & x \\ 0 & 0 & x & x \end{bmatrix}$$

$$\longrightarrow \Omega_{23}A_2 = A_3 = \begin{bmatrix} x & x & x & x \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix}$$

$$\longrightarrow \Omega_{34}A_3 = A_4 = \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} = R.$$

```
% QR算法
[P,H] = hess(G');   A=H;
for k=1:50
    [q,r]=qr(A);    A=r*q, pause
end

[P,H] = hess(G');   A=H;
for k=1:100
    [q,r]=qr(A);   A=r*q;
end
```