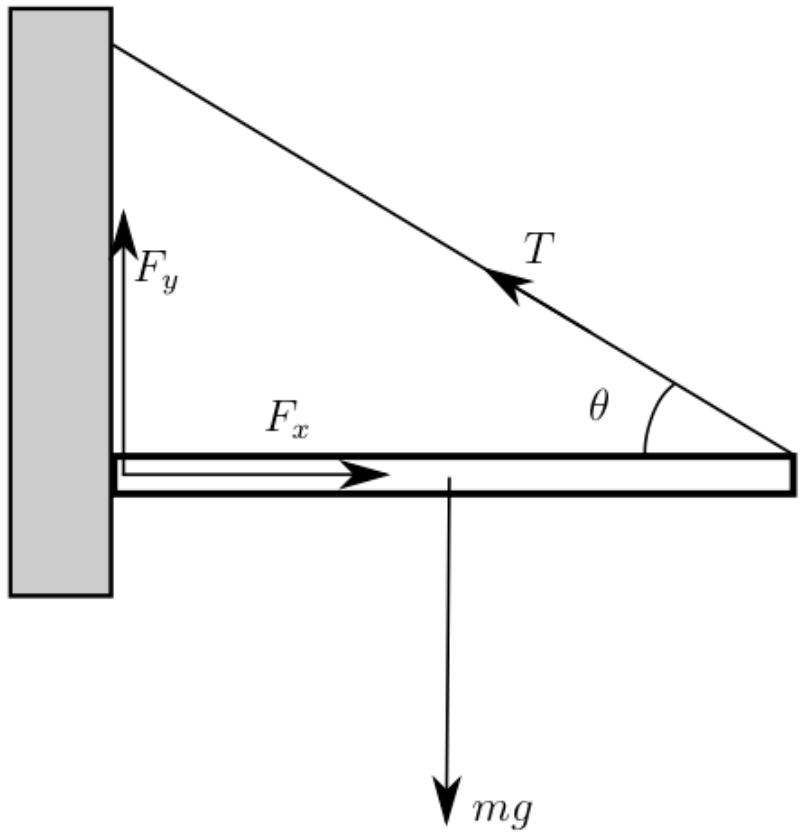
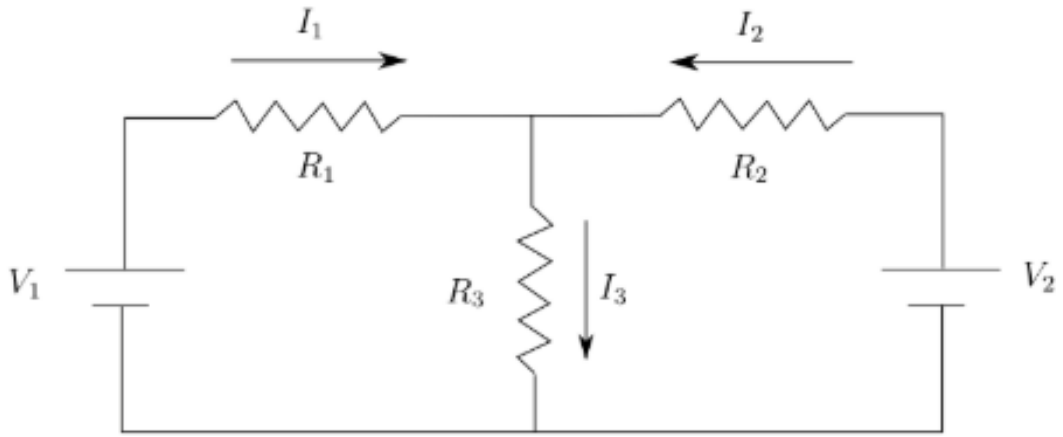
Cálculo Matricial





Algebra Matricial

Norma de um vector

V : espaço vectorial

norma: $\|\cdot\| : V \to \mathbb{R}$

1. $\|x\| \geq 0 \quad \forall x \in V \quad$ e $\quad \|x\| = 0 \Rightarrow x = 0$
2. $\|\alpha x\| = |\alpha| \cdot \|x\| \quad \forall \alpha \in \mathbb{R}, \forall x \in V$
3. $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in V$

Norma de um vector

→ norma 1 $\sum_{i=1}^{n} |x_i|$

→ norma ∞ $\max_{1 \leq i \leq n} |x_i|$

→ norma p $\left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, \quad (\text{com } p \geq 1)$

Norma Euclideana

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

- Example:

$$x = \begin{bmatrix} 1 \\ 100 \\ 9 \end{bmatrix} \text{ and } \hat{x} = \begin{bmatrix} 1.1 \\ 99 \\ 11 \end{bmatrix}$$

$\|\hat{x} - x\|_\infty = 2$ $\quad \frac{\|\hat{x}-x\|_\infty}{\|x\|_\infty} = 0.02$ $\quad \frac{\|\hat{x}-x\|_\infty}{\|\hat{x}\|_\infty} = 0.0202$

$\|\hat{x} - x\|_2 = 2.238$ $\quad \frac{\|\hat{x}-x\|_2}{\|x\|_2} = 0.0223$ $\quad \frac{\|\hat{x}-x\|_2}{\|\hat{x}\|_2} = 0.0225$

- Equivalência de normas de um vector

- For l_1 and l_2 norms, there exist c_1 and c_2 such that

$$c_1\|x\|_{l_1} \leq \|x\|_{l_2} \leq c_2\|x\|_{l_1}$$

  for all $x \in R^n$

- Example:

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2 \quad (1)$$

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty \quad (2)$$

$$\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty \quad (3)$$

- Therefore, you can just choose a norm for your convenience

- Aplicações das normas de um vector

1 Are two vectors (nearly) equal?

Floating point comparison of two scalars with absolute value:

$$\frac{|\alpha - \beta|}{|\alpha|} < \delta$$

where δ is a small tolerance.

Comparison of two vectors with norms:

$$\frac{\|y - z\|}{\|z\|} < \delta$$

Notice that

$$\frac{\|y - z\|}{\|z\|} < \delta$$

is **not equivalent to**

$$\frac{\|y\| - \|z\|}{\|z\|} < \delta.$$

This comparison is important in convergence tests for sequences of vectors.

2 Creating a Unit Vector

Given $u = [u_1, u_2, \ldots, u_m]^T$, the unit vector in the direction of u is

$$\hat{u} = \frac{u}{\|u\|_2}$$

The following are *not* unit vectors

$$\frac{u}{\|u\|_1} \qquad \frac{u}{\|u\|_\infty}$$

Norma induzida de uma matriz

$$\|A\|_l \equiv \max_{x \neq 0} \frac{\|Ax\|_l}{\|x\|_l} = \max_{\|x\|=1} \|Ax\|_l$$

1. $\forall x \in \mathbb{R}^n \quad \|Ax\| \leq \|A\| \cdot \|x\|$
2. $\|AB\| \leq \|A\| \cdot \|B\| \quad \forall A, B \in \mathbb{R}^{n \times n}$
3. $\|I\| = 1$, onde I é a matriz identidade

Normas 1 e ∞

Seja $A \in \mathbb{R}^{n \times n}$ de elemento genérico a_{ij}. Então verifica-se

$$\|A\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}|,$$

ou seja, $\|A\|_1$ é o máximo das somas por colunas dos valores absolutos dos elementos de A.

Seja $A \in \mathbb{R}^{n \times n}$ de elemento genérico a_{ij}. Então verifica-se

$$\|A\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|,$$

ou seja, $\|A\|_\infty$ é o máximo das somas por linhas dos valores absolutos dos elementos de A.

Norma 2 de uma matriz

Seja $A \in \mathbb{R}^{n \times n}$. *Então verifica-se* $\|A\|_2 = \sqrt{\rho(A^{\mathrm{T}} A)}$.

$\rho(C)$ é o **raio espectral** de $C \in \mathbb{R}^{n \times n}$ definido por

$$\rho(C) = \max_{1 \leq i \leq n} |\lambda_i|$$

onde $\lambda_1, \ldots, \lambda_n$ são os valores próprios de C.

Raio espectral e norma 2 são de cálculo trabalhoso!

Exemplo de normas

Sendo

$$A = \begin{bmatrix} -2 & 0 & 1 & 6 \\ -3 & -1 & 2 & 4 \\ 2 & 1 & -1 & 1 \\ 3 & -2 & 2 & 5 \end{bmatrix}$$

calcular $\|A\|_1$ e $\|A\|_\infty$.

```
>>> from numpy import linalg as LA
>>> A= np.array([[-2,0,1,6],[-3,-1,2,4],[2,1,-1,1],[3,-2,2,5]])
>>> LA.norm(A,1)
>>> LA.norm(A, np.inf)
```

Rank de uma matriz

The **rank** of a matrix, A, is the number of linearly independent columns in A.

```
>>> from numpy.linalg import matrix_rank

>>> matrix_rank(np.eye(4)) # Full rank matrix 4
>>> I=np.eye(4); I[-1,-1] = 0. # rank deficient matrix
>>> matrix_rank(I)
```

Sistemas de equações lineares

Sistemas na forma triangular

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1,n-1}x_{n-1} + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \cdots + a_{2,n-1}x_{n-1} + a_{2n}x_n &= b_2 \\ \vdots & \vdots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n &= b_{n-1} \\ a_{nn}x_n &= b_n \end{cases}$$

Substituição inversa

$$x_n = \frac{b_n}{a_{nn}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \qquad i = n-1, \ldots, 1$$

Sistemas de equações lineares

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = \omega_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = \omega_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = \omega_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = \omega_4 \end{cases}$$

$$\Leftrightarrow \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix}$$

Em python podemos resolver um sistemas de equações usando o comando solve() da biblioteca numpy.

```
>> from numpy.linalg import la
>> a = np.array([[1, 2], [3, 5]])
>> b = np.array([1, 2])
>> x = np.linalg.solve(a, b)
```

Eliminação Gaussiana

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} - \frac{a_{21}a_{12}}{a_{11}} & a_{23} - \frac{a_{21}a_{13}}{a_{11}} & a_{24} - \frac{a_{21}a_{14}}{a_{11}} \\ 0 & a_{32} - \frac{a_{31}a_{12}}{a_{11}} & a_{33} - \frac{a_{31}a_{13}}{a_{11}} & a_{34} - \frac{a_{31}a_{14}}{a_{11}} \\ 0 & a_{42} - \frac{a_{41}a_{12}}{a_{11}} & a_{43} - \frac{a_{41}a_{13}}{a_{11}} & a_{44} - \frac{a_{41}a_{14}}{a_{11}} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \omega_1 \\ \omega_2 - \frac{a_{21}\omega_1}{a_{11}} \\ \omega_3 - \frac{a_{31}\omega_1}{a_{11}} \\ \omega_4 - \frac{a_{41}\omega_1}{a_{11}} \end{pmatrix}$$

...

Matriz triangular superior

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & a_{34}^{(3)} \\ 0 & 0 & 0 & a_{44}^{(4)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \omega_1^{(1)} \\ \omega_2^{(2)} \\ \omega_3^{(3)} \\ \omega_4^{(4)} \end{pmatrix}$$

A solução obtém-se por substituição inversa:

$$x_m = \frac{1}{a_{mm}^{(m)}} \left(\omega_m^{(m)} - \sum_{k=m+1}^{n} a_{mk}^{(m)} x_k \right) \qquad m = n, n-1, n-2, \ldots, 1$$

Este método requer n^3 **operações** algébricas!

Eliminação Gaussiana com aritmética finita

Na substituição inversa $$x_m = \frac{1}{a_{mm}^{(m)}} \left(\omega_m^{(m)} - \sum_{k=m+1}^{n} a_{mk}^{(m)} x_k \right)$$

a propagação de erros é $$\varepsilon_{x_i} \leq \sum_{j=i+1}^{n} \frac{|a_{ij}|}{|a_{ii}|} \varepsilon_{x_j}$$

Então interessa que os quocientes $\frac{|a_{ij}|}{|a_{ii}|}$ sejam pequenos!

Para tal usam-se **estratégias de escolha de pivot**

de modo a evitar que a_{ii} sejam nº muito pequenos.

Pivotagem total (por coluna e linha): permutam-se as linhas e colunas de A e dos vectores x e ω de modo a colocar na diagonal o maior valor existente na coluna e linha do elemento "problemático"

$$\begin{pmatrix} 1 & 3 & 4 & 6 \\ 0 & 10^{-8} & 198 & 19 \\ 0 & -91 & 51 & 9 \\ 0 & 7 & 76 & 541 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 4 & 3 & 6 \\ 0 & 198 & 10^{-8} & 19 \\ 0 & 51 & -91 & 9 \\ 0 & 76 & 7 & 541 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} \omega_1 \\ \omega_3 \\ \omega_2 \\ \omega_4 \end{pmatrix}$$

Cálculo da matriz inversa usando Eliminação Gaussiana

- Podemos obter a inversa da matriz A reparando que o processo pode ser visto como a solução de n sistemas lineares...

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} a_{11}^{-1} & a_{12}^{-1} & a_{13}^{-1} & a_{14}^{-1} \\ a_{21}^{-1} & a_{22}^{-1} & a_{23}^{-1} & a_{24}^{-1} \\ a_{31}^{-1} & a_{32}^{-1} & a_{33}^{-1} & a_{34}^{-1} \\ a_{41}^{-1} & a_{42}^{-1} & a_{43}^{-1} & a_{44}^{-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Depois para qualquer ω resolve-se o sistema com um cálculo simples:

$$Ax = \omega \quad \Leftrightarrow \quad x = A^{-1}\omega$$

Factorização LU

A=L.U

Onde L é uma triangular inferior e U matriz triangular superior.

Para obter a solução do Sistema Ax=b

1) Ly=b
2) Ux=y

Conhecida a factorização de A basta resolver 2 sistemas de equações com matrizes triangulares.

Em python temos o comando de factorização LU:

```
>> from scipy.linalg import lu
>> A = np.array([[2, 5, 8, 7], [5, 2, 2, 8], [7, 5, 6, 6], [5, 4, 4, 8]])
>> p, l, u = lu(A)
```

Existem outras factorizações: Cholesky para matrizes Hermiteanas ($A=L^T.L$) -> chol(A)
Decomposição QR para rectangulares -> qr(A)

Erro e residuo de uma solução

Sistema de equações: $\boxed{Ax = b}$ (A não singular)

$\bar{x}$: solução exacta

$\tilde{x}$: solução aproximada

Erro da solução aproximada: $e = \bar{x} - \tilde{x}$

Resíduo da solução aproximada: $r = b - A\tilde{x}$

Relação entre erro e resíduo: $\boxed{r = Ae}$

$\tilde{x} = \bar{x} \quad \Rightarrow \quad e = 0 \wedge r = 0$

$\tilde{x} \neq \bar{x} \quad \Rightarrow \quad ?$

erro pequeno $\stackrel{?}{\Rightarrow}$ resíduo pequeno

resíduo pequeno $\stackrel{?}{\Rightarrow}$ erro pequeno

$\begin{bmatrix} 1.01 & 0.99 \\ 0.99 & 1.01 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ tem solução exacta $\bar{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$\longrightarrow$ se $\tilde{x} = \begin{bmatrix} 1.01 \\ 1.01 \end{bmatrix}$ tem-se $e = \begin{bmatrix} -0.01 \\ -0.01 \end{bmatrix}$ e $r = \begin{bmatrix} -0.02 \\ -0.02 \end{bmatrix}$

erro relativo: 1% em cada componente
resíduo relativo: 1% em cada componente

$\longrightarrow$ se $\hat{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ tem-se $e = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ e $r = \begin{bmatrix} -0.02 \\ 0.02 \end{bmatrix}$

erro relativo: 100% em cada componente
resíduo relativo: 1% em cada componente

Problemas mal condicionados

O **número de condição** da matriz A é definido por

$$\mathrm{cond}(A) = \|A\| \cdot \|A^{-1}\|$$

A relação entre erro e resíduo fica agora

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

Se $\mathrm{cond}(A) \approx 1$ a matriz diz-se **bem condicionada**

Se $\mathrm{cond}(A) \gg 1$ a matriz diz-se **mal condicionada**

Seja $\bar{x}$ a solução do sistema de equações $Ax = b$, onde A é não singular e b é não nulo.

Seja $\tilde{x}$ a solução do sistema de equações (perturbado) $Ax = \tilde{b}$.

Então verifica-se que

$$\frac{\|\bar{x} - \tilde{x}\|}{\|\bar{x}\|} \leq \mathrm{cond}(A) \frac{\|b - \tilde{b}\|}{\|b\|}.$$

O sistema de equações $Ax = b$, onde

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 4 & 3 & 1 \\ 2 & 2 & 3 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

tem solução $\bar{x} = \begin{bmatrix} -0.2 & 1 & -0.2 \end{bmatrix}^{\mathrm{T}}$.

Se $\tilde{b} = \begin{bmatrix} 1.1 & 2.2 & 0.9 \end{bmatrix}^{\mathrm{T}}$, a solução é $\tilde{x} = \begin{bmatrix} -0.62 & 1.7 & -0.42 \end{bmatrix}^{\mathrm{T}}$

Seja $\bar{x}$ a solução do sistema de equações $Ax = b$, onde A é não singular.

Seja $\tilde{x}$ a solução do sistema de equações (perturbado) $\tilde{A}x = b$, onde $\tilde{A}$ é não singular.

Então verifica-se que

$$\frac{\|\bar{x} - \tilde{x}\|}{\|\tilde{x}\|} \leq \mathrm{cond}(A) \frac{\|\tilde{A} - A\|}{\|A\|}.$$

O sistema de equações $Ax = b$, onde

$$A = \begin{bmatrix} 1 & 5 & 10 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 16 \\ -5 \\ 1 \end{bmatrix},$$

tem solução $\bar{x} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$.

Se a matriz dos coeficientes for $\tilde{A} = \begin{bmatrix} 1 & 5 & 10 \\ 0 & 1 & -6 \\ 0 & 0 & 1.1 \end{bmatrix}$ a solução é

$\tilde{x} = \begin{bmatrix} \frac{51}{11} & \frac{5}{11} & \frac{10}{11} \end{bmatrix}^{\mathrm{T}}$.

Refinamento iterativo de uma solução

Nos casos de problemas mal condicionados é possível melhorar a solução usando refinamento iterativo.

0) Factorizar A em decomposição LU

1) $A.u = b$

2) $r = b - A.\tilde{u}$

3) $A.z = r$

4) $u = \tilde{u} + z$

5) Enquanto $|| u - \tilde{u} || > \varepsilon$ voltar a repetir desde o passo 1.

Exemplo:

```
>> A=LA.hilbert(12)
>> b=range(11,0,-1)
>> np.cond(A)
```

Solução de problemas sobredeterminados e subdeterminados

Nos casos de problemas sobredeterminados ou subdeterminados:

- decomposição QR e SVD

  1) Resolver Q.y=b
  2) Resolver R.x=y

- pseudo inversa, permite obter a solução com a norma 2 menor.

  $A.x=b \Leftrightarrow A^T.A.x= A^T.b \Leftrightarrow x=(A^T.A)^{-1} A^T.b \Leftrightarrow x = A^+.b$

  onde A^+ é a pseudo-inversa de A. Na numpy obtém com o comando pinv().

Exemplos (comparar comando solve() e usando pinv) :

```
>> A=[[1 2 3 4]; [-5 3 2 7]]
>> b=[1, 2]
>> A=[[1 2]; [1 2]; [1 2]; [1 2];]
>>b= [1, 1.03, 0.97, 1.01]
```

Métodos iterativos para sistemas de equações lineares

- Métodos diretos são lentos $O(N^3)$, em especial para sistemas com matrizes grandes mas esparsas.

- Menos sensíveis a erros de arredondamento.

Métodos iterativos para sistemas de equações lineares

1. Substituir a equação $Ax = b$ pela equação equivalente

$$x = Gx + d$$

2. Escolher um valor inicial $x_{(0)} \in \mathbb{R}^n$

3. Gerar a sucessão $\{x_{(k)}\}$, pela relação de recorrência

$$\boxed{x_{(k+1)} = Gx_{(k)} + d} \qquad k = 0, 1, \ldots.$$

A sucessão $\{x_{(k)}\}$ deverá ser convergente para $A^{-1}b$!

Convergência dos métodos iterativos para sistemas de equações lineares

Sejam $G \in \mathbb{R}^{n\times n}$ *e* $d \in \mathbb{R}^n$. *Se* $\|G\| < 1$, *então*

1. *existe uma e uma só solução* $\bar{x} \in \mathbb{R}^n$ *da equação*

$$x = Gx + d,$$

2. *a sucessão* $\{x_{(k)}\}$, *gerada por*

$$x_{(k+1)} = Gx_{(k)} + d, \qquad k = 0, 2, \ldots,$$

   converge para $\bar{x}$, *qualquer que seja o ponto inicial* $x_{(0)}$.

3. *o erro de aproximação de* $\bar{x}$ *por* $x_{(k+1)}$, $\bar{x} - x_{(k+1)}$, *satisfaz*

$$\|\bar{x} - x_{(k+1)}\| \leq \frac{\|G\|}{1 - \|G\|}\|x_{(k+1)} - x_{(k)}\|, \qquad k = 1, 2, \ldots.$$

Convergência dos métodos iterativos para sistemas de equações lineares

Uma matriz $A \in \mathbb{R}^{n\times n}$ diz-se **estritamente diagonalmente dominante por linhas** quando

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \qquad i = 1, \ldots, n.$$

Teorema

Sejam $A \in \mathbb{R}^{n\times n}$ e $b \in \mathbb{R}^n$. Se a matriz A for estritamente diagonalmente dominante por linhas então a sucessão gerada pelo método converge para a única solução do sistema de equações $Ax = b$, qualquer que seja o ponto inicial $x_{(0)}$.

Métodos iterativo de Jacobi

$$\boldsymbol{L}\boldsymbol{x}^{(p)} + \boldsymbol{D}\boldsymbol{x}^{(p+1)} + \boldsymbol{U}\boldsymbol{x}^{(p)} = \boldsymbol{b}$$

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \qquad L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \qquad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ & & & \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\boldsymbol{x}^{(p+1)} = \boldsymbol{D}^{-1}(\boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(p)})$$

Expressão de recorrência

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)} \right]$$

Métodos iterativo de Gauss Seidel

$$\boldsymbol{L}\boldsymbol{x}^{(p+1)} + \boldsymbol{D}\boldsymbol{x}^{(p+1)} + \boldsymbol{U}\boldsymbol{x}^{(p)} = \boldsymbol{b}$$

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \qquad L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \qquad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ & & & \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\boldsymbol{x}^{(p+1)} = (\boldsymbol{L} + \boldsymbol{D})^{-1}(\boldsymbol{b} - \boldsymbol{U}\boldsymbol{x}^{(p)})$$

Expressão de recorrência

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)} \right]$$

Métodos iterativos com Relaxação

Em vez de aceitar o valor calculado $\mathbf{x^{(p+1)}}$ como novo valor imediatamente, guarda-se temporariamente em $\mathbf{t^{(p+1)}}$, combina-se com o valor anterior $\mathbf{x^{(p)}}$:

$$\mathbf{x^{(p+1)} = (1- \omega).x^{(p)} + \omega. t^{(p+1)}}$$

onde ω é parâmetro de relaxação. Se ω<1 temos sub-relaxação, usa-se para problemas de convergência dificil. Se ω>1 temos sobre-relaxação, usa-se para acelerar a convergência.

JACOBI

Expressão de recorrência com relaxação $(\omega > 0)$

$$x_i^{(k+1)} = x_i^{(k)} + \omega \cdot \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)} \right]$$

GAUSS-SEIDEL

Expressão de recorrência com relaxação $(\omega > 0)$

$$x_i^{(k+1)} = x_i^{(k)} + \omega \cdot \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)} \right]$$

O método Gauss-Seidel com **sobre-relaxação** (1<ω<2) designa-se por método **SOR (Sucessive Over-Relaxation)**, existindo um ω ótimo para o qual a convergência é máxima.

Métodos iterativos. Exemplo.

Comparar o método de Jacobi, Gauss-Seidel e o método SOR com $\omega = 1.25$ na resolução do sistema de equações

$$\begin{cases} 4x_1 + 3x_2 & = 24 \\ 3x_1 + 4x_2 - x_3 & = 30 \\ -x_2 + 4x_3 & = -24 \end{cases}$$

cuja solução é $x_1 = 3$, $x_2 = 4$, $x_3 = -5$. Em ambos os casos partir de $x_{1,(0)} = x_{2,(0)} = x_{3,(0)} = 1$.

Problema aos valores próprios

Uma matriz quadrada A, quando aplicada sobre certos vectores (vectores próprios), obtém-se o próprio vector multiplicado por uma constante λ (valor próprio) :

$$Ax = \lambda x$$

Estes valores próprios podem ser obtidos resolvendo a equação:

$$det(A - \lambda I) = 0.$$

Na numpy os valores próprios de A são determinados usando:

>> D,v=np.linalg.eig(A)

Exemplo:

$$A = \begin{bmatrix} 2 & 4 & 3 \\ 4 & 4 & 3 \\ 3 & 3 & 5 \end{bmatrix}$$

V =

0.8197	-0.2674	0.5066
-0.5587	-0.5685	0.6039
-0.1265	0.7780	0.6154

D =

-1.1894	0	0
0	1.7764	0
0	0	10.4130

Método geral para obter todos os valor próprios. Método QR.

- Suppose $A \in \mathbb{R}^{n \times n}$, then QR method is to apply iterations as follows

$$A_{m-1} = Q_m R_m$$

$$A_m = R_m Q_m$$

where Q_m is a orthogonal matrix, R_m is an upper triangular matrix.

- Finally R_m will tend to

$$\begin{pmatrix} \lambda_1 & * & \cdots & * \\ & \lambda_2 & \ddots & * \\ & & \ddots & * \\ & & & \lambda_n \end{pmatrix}.$$

We find all of the eigenvalues of A! The eigenvectors are the columns of Q_m .

If A_{m-1} is in upper Hessenberg form A_m will also be, and thus faster to factorize.

Método da potência para obter o maior valor próprio.

Se estivermos interessados em obter apenas o maior valor próprio de A, podemos usar a seguinte iteração:

$$\boldsymbol{y}^{(p+1)} = \boldsymbol{A}\boldsymbol{z}^{(p)}$$

$$\boldsymbol{y}^{(p+1)} = \boldsymbol{k}^{(p+1)}\boldsymbol{z}^{(p+1)} \qquad \boldsymbol{p} = \mathbf{0}, \mathbf{1}, \cdots$$

Com $k^{(p+1)} = || y^{(p+1)}||_2$. Fazendo o quociente de $y^{(k+1)}/z^{(k)}$ obtemos a estimativa do valor próprio.

O menor valor próprio pode ser obtido de:

$$\boldsymbol{\mu}_i\boldsymbol{x}_i = \boldsymbol{A}^{-1}\boldsymbol{x}_i \qquad \boldsymbol{\lambda}_i = \frac{\mathbf{1}}{\boldsymbol{\mu}_i}.$$

Fazendo a iteração: $\boldsymbol{y}^{(p+1)} = \boldsymbol{A}^{-1}\boldsymbol{z}^{(p)}$. Ou alternativamente, resolvendo o sistema de equações : $\boldsymbol{A}\boldsymbol{y}^{(p+1)} = \boldsymbol{z}^{(p)}$

Exemplo: Calcular o maior e o menor valor próprio de T

$$\boldsymbol{T} = \begin{bmatrix} \mathbf{2} & -\mathbf{1} & \mathbf{0} & \mathbf{0} \\ -\mathbf{2} & \mathbf{2} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{2} & \mathbf{2} & -\mathbf{1} \\ \mathbf{0} & \mathbf{0} & -\mathbf{2} & \mathbf{2} \end{bmatrix}$$