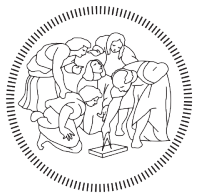
Calcolo Numerico
Algoritmi

Facoltà di Ingegneria Aerospaziale

Anno Accademico 2024-2025



POLITECNICO
MILANO 1863

Contents

List of Algorithms

Chapter 1

Risoluzione di Sistemi Lineari

1.1 Metodi diretti

1.1.1 Algoritmi di Sostituzione per matrici Triangolari

Algorithm 1 Algoritmo di Sostituzione in avanti (per matrici L)

$L\vec{y} = \vec{b}$
$y_1 = \dfrac{b_1}{l_{11}}$
for i = 2, ...,n **do**
$y_i = \dfrac{1}{l_{ii}}(b_i - \sum_{j=1}^{i-1} l_{ij}y_j)$
end for

Algorithm 2 Algoritmo di Sostituzione all'indietro (per matrici a U)

$U\vec{x} = \vec{y}$
$x_n = \dfrac{y_n}{u_{nn}}$
for i = n-1, ..., 1 **do**
$x_i = \dfrac{1}{u_{ii}}(y_i - \sum_{j=i+1}^{n} u_{ij}x_j)$
end for

1.1.2 Metodo di Eliminazione di Gauss

Algorithm 3 MEG

assegnare $A^{(1)} = A$
for k = 1, ..., n-1 **do**
  for i = k+1, ..., n **do**
    $l_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
    for j = k+1, ..., n **do**
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$
    end for
  end for
end for

Algorithm 4 MEG con pivoting sulle righe

assegnare $A^{(1)} = A$
for k = 1, ..., n-1 **do**
  for i = k+1, ..., n **do**
    cerco $r : |a_{rk}^{(k)}| > |a_{ik}^{(k)}|$
    scambio la riga k+1 con r
  end for
  for i = k+1, ..., n **do**
    $l_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
    for j = k+1, ..., n **do**
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$
    end for
  end for
end for

1.1.3 Metodo di Thomas

Algorithm 5 Calcolo di LU per matrici tridiagonali

$\alpha_1 = a_1$
for i = 2, ..., n **do**
    $\beta_i = \dfrac{l_i}{\alpha_1 - i}$
    $\alpha_i = a_i - \beta_i c_i - 1$
end for

Algorithm 6 Sostituzione in avanti e indietro per bidiagonali

▷ Sostituzione in avanti per bidiagonali

$y_1 = b_1$
for i = 2, ..., n **do**
    $y_i = b_i - \beta_i y_i - 1$
end for

▷ Sostituzione all'indietro per bidiagonali

$x_n = \dfrac{y_n}{\alpha_n}$
for i = n, ..., 2 **do**
    $x_i = \dfrac{y_i - c_i x_i + 1}{\alpha_i}$
end for

1.1.4 Metodo della fattorizzazione di Cholesky

Algorithm 7 Fattorizzazione di Cholesky

$r_{11} = \sqrt{a_{11}}$
for i = 2, ..., n **do**
    for j = 1, ..., i-1 **do**
        $R_{ji} = \dfrac{1}{R_{jj}}(a_{ij} - \sum_{k=1}^{j-1} r_{ki} r_{kj})$
    end for
    $r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2}$
end for

1.1.5 Fattorizzazione QR (per sistemi rettangolari)

Algorithm 8 Ortonormalizzazione di Gram-Schmidt

Data $A \in \mathbb{R}^{m \times n}, A = [\vec{a_1}, \vec{a_2}, ..., \vec{a_n}]$

$\tilde{q_1} = \dfrac{\vec{a_1}}{||\vec{a_1}||}$

for k = 1, ..., n-1 **do**

$\vec{q}_{k+1} = \vec{a}_{k+1} - \sum_{j=1}^{n} (\tilde{q_j} \cdot \vec{a}_{k+1}) \tilde{q_j}$

$\tilde{q}_{k+1} = \dfrac{\vec{q}_{k+1}}{||\vec{q}_{k+1}||}$

end for

1.2 Metodi iterativi

1.2.1 Metodi iterativi per Decomposizione Additiva

Algorithm 9 Metodo di Jacobi

Dato $\vec{x}^{(0)}$

for $k = 0, 1, ...$ **do**

for $i = 1, ..., n$ **do**

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

end for

end for

Algorithm 10 Metodo di Gauss-Seidel

Dato $\vec{x}^{(0)}$

for k = 0, 1, ... **do**

for i = 1, ..., n **do**

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

end for

end for

Algorithm 11 Metodi Richardson Precondizionati

Dato $\vec{x}^{(0)} \in \mathbb{R}^n$
for $k = 0, 1, ...$ **do**
$P\vec{z}^{(k)} = r^{(k)}$
$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \alpha_k z^{(k)}$
end for

Metodo del Gradiente

Algorithm 12 Metodo del Gradiente

Dato $\vec{x}^{\,(0)} \in \mathbb{R}^n, \vec{r}^{\,(0)} = \vec{b} - A\vec{x}$
for k = 0, 1, ... **do**
$$\alpha_k = \frac{(\vec{r}^{(k)})^T \, \vec{r}^{(k)}}{(\vec{r}^{(k)})^T \, A \, \vec{r}^{(k)}}$$
$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \alpha_k \vec{r}^{(k)}$
$\vec{r}^{(k+1)} = \vec{b} - A\vec{x}^{\,(k)} = \vec{r}^{(k)} - \alpha_k A\vec{r}^{(k)}$
end for

Algorithm 13 Metodo del Gradiente Precondizionato

Dato $\vec{x}^{\,(0)} \in \mathbb{R}^n, \vec{r}^{(0)} = \vec{b} - A\vec{x}$
for k = 0, 1, ... **do**
$P\vec{z}^{(k)} = \vec{r}^{(k)}$
$$\alpha_k = \frac{(\vec{z}^{(k)})^T \, \vec{z}^{(k)}}{(\vec{z}^{(k)})^T \, A \, \vec{z}^{(k)}}$$
$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \alpha_k \vec{z}^{(k)}$
$\vec{r}^{(k+1)} = \vec{r}^{(k)} - \alpha_k A\vec{z}^{(k)}$
end for

Algorithm 14 Metodo del Gradiente Coniugato

▷ Le direzioni sono A-coniugate

Dato $\vec{x}^{\,(0)} \in \mathbb{R},\ \vec{r}^{\,(0)} = \vec{b} - A\vec{x}^{\,(0)},\ \vec{p}^{\,(0)}$

for k = 0, 1,... **do**

$$\alpha_k = \frac{(\vec{p}^{\,(k)})^T \vec{r}^{\,(k)}}{(\vec{p}^{\,(k)})^T A \vec{r}^{\,(k)}}$$

$$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \alpha_k \vec{p}^{\,(k)}$$

$$\vec{r}^{\,(k+1)} = \vec{r}^{\,(k)} - \alpha_k A \vec{p}^{\,(k)}$$

$$\beta_k = \frac{(\vec{p}^{\,(k)})^T A \vec{r}^{\,(k+1)}}{\vec{p}^{\,(k)} A \vec{p}^{\,(k)}}$$

$$\vec{p}^{\,(k+1)} = \vec{r}^{\,(k+1)} - \beta_k \vec{p}^{\,(k)}$$

end for

Chapter 2

Autovalori e Autovettori

Algorithm 15 Metodo delle Potenze (λ_{max})

Dato $\vec{x}^{\,(0)} \in \mathbb{C}^n$, con $||\,\vec{x}^{\,(0)}|| \neq 0$

$\vec{y}^{\,(0)} = \dfrac{\vec{x}^{\,(0)}}{||\,\vec{x}^{\,(0)}||}$ ▷ Normalizzazione

$\lambda^{(0)} = \vec{y}^{\,(0)H} A\, \vec{y}^{\,(0)}$

for k = 1, 2, ... **do**

$\vec{x}^{\,(k)} = A\, \vec{y}^{\,(k-1)}$

$\vec{y}^{\,(k)} = \dfrac{\vec{x}^{\,(k)}}{||\,\vec{x}^{\,(k)}||}$ ▷ Normalizzazione

$\lambda^{(k)} = \vec{y}^{\,(k)H} A\, \vec{y}^{\,(k)}$

end for

Algorithm 16 Metodo delle potenze inverse (λ_{min})

Dato $\vec{x}^{\,(0)} \in \mathbb{C}^n, \vec{x}^{\,(0)} \neq \vec{0}$

$\vec{y}^{\,(0)} = \dfrac{\vec{x}^{\,(0)}}{||\,\vec{x}^{\,(0)}||}$ ▷ Normalizzazione

$\mu^{(0)} = \vec{y}^{\,(0)H} A^{-1}\,\vec{y}^{\,(0)}$

for k = 1, 2, ... **do**

risolvo $A\vec{x}^{\,(k)} = \vec{y}^{\,(k-1)}$ ▷ Non assembliamo l'inversa!

$\vec{y}^{\,(k)} = \dfrac{\vec{x}^{\,(k)}}{||\,\vec{x}^{\,(k)}||}$

$\lambda^{(k)} = \vec{y}^{\,(k)\,H} A\,\vec{y}^{\,(k)}$

end for

Algorithm 17 Metodo delle potenze inverse con shift ($\lambda \approx \bar{\lambda}$ (assegnato))

Dato $\vec{x}^{\,(0)} \in \mathbb{C}^n, \vec{x}^{\,(0)} \neq \vec{0}$

$\vec{y}^{\,(0)} = \dfrac{\vec{x}^{\,(0)}}{||\,\vec{x}^{\,(0)}||}$ ▷ Normalizzazione

$\mu^{(0)} = \vec{y}^{\,(0)\,H}(A - sI)^{\,-1}\,\vec{y}^{\,(0)}$

for k = 1, 2, ... **do**

risolvo $(A - sI)\,\vec{x}^{\,(k)} = \vec{y}^{\,(k-1)}$

$\vec{y}^{\,(k)} = \dfrac{\vec{x}^{\,(k)}}{||\,\vec{x}^{\,(k)}||}$

$\mu^{(k)} = \vec{y}^{\,(k)\,H}(A - sI)^{\,-1}\,\vec{y}^{\,(k)}$

end for

$\lambda^{(k)} = \dfrac{1}{\mu^{(k)}} + s$

Algorithm 18 Metodo delle iterazioni QR (per calcolo dello spettro)

Data $A \in \mathbb{R}^{n\times n}$, $A^{(0)} = A$

while not criterio d'arresto **do**

Data $A^{(k)}$, calcolo la fattorizzazione $A^{(k)} = Q^{(k+1)}R^{(k+1)}$

$A^{(k+1)} = R^{(k+1)}Q^{(k+1)}$

for i = 1, ..., n **do**

$\lambda_i^{(k+1)} = (A^{(k+1)})_{ii}$

end for

end while

Chapter 3

Equazioni e Sistemi non Lineari

Algorithm 19 Metodo di Bisezione

Pongo $k = 0,\ a^{(0)} = a,\ b^{(0)} = b,\ x^{(0)} = \dfrac{a^{(0)} + b^{(0)}}{2}$
for k = 1, 2, ... **do**
▷ Se x è la radice
  if $f(x^{(k-1)}) = 0$ **then**
    $\alpha = x^{(k-1)}$
    break
▷ Se la funzione cambia segno tra x e a
  else if $f(x^{(k-1)})f(a^{(k-1)}) < 0$ **then**
    $a^{(k)} = a^{(k-1)},\ b^{(k)} = x^{(k-1)}$
▷ Se la funzione cambia segno tra x e b
  else if $f(x^{(k-1)})f(b^{(k-1)}) < 0$ **then**
    $a^{(k)} = x^{(k-1)},\ b^{(k)} = b^{(k-1)}$
  end if
  $x^{(k)} = \dfrac{a^{(k)} + b^{(k)}}{2}$
end for

3.1 Metodo di Newton

Algorithm 20 Metodo di Newton

Dato $x^{(0)}$
for k = 1, 2, ... **do**
$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$
end for

Algorithm 21 Metodo di Newton - variante con incremento

Dato $x^{(0)}$
for k = 1, 2, ... **do**
$$\delta = -\frac{f(x^{(k)})}{f'(x^{(k)})}$$
$$x^{(k+1)} = x^{(k)} + \delta$$
end for

Algorithm 22 Metodo di Newton Modificato

Dato $x^{(0)}$ **for** $k = 0, 1, ...$ **do**
$$x^{(k+1)} = x^{(k)} - m^{(k)}\frac{f(x^{(k)})}{f'(x^{(k)})}$$ ▷ Dove m è la molteplicità della radice

Algorithm 23 Metodo di Newton Additivo

Dato $x^{(0)}$
for k = 0, 1, ... **do**
▷ Valutiamo m (molteplicità della radice) come:
$$m^{(k)} = \frac{x^{(k-1)} - x^{(k-2)}}{2x^{(k-1)} - x^{(k)} - x^{(k-2)}}$$
$$x^{(k+1)} = x^{(k)} - m^{(k)}\frac{f(x^{(k)})}{f'(x^{(k)})}$$
end for

Algorithm 24 Metodo di Newton per sistemi non lineari

Dato $\vec{x}^{\,(0)} \in \mathbb{R}^n$,
for k = 0, 1, ... **do** ▷ Dove J_f è la matrice Jacobiana
  $J_f(\vec{x}^{\,(k)})\ \vec{\delta} = -\vec{f}(\vec{x}^{\,(k)})$ ▷ Risoluzione di sist. lin. $n \times n$
  $\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \vec{\delta}$
end for

3.2 Metodi delle Iterazioni di Punto Fisso

Algorithm 25 Metodi delle iterazioni di punto fisso

Dato $\vec{x}^{\,(0)}$,
for k = 0, 1, ... **do**
  $\vec{x}^{\,(k+1)} = \phi(\vec{x}^{\,(k)})$
end for

Chapter 4

Problemi di Ottimizzazione

4.1 Metodi Derivative-Free

Algorithm 26 Metodo della Sezione Aurea

Sia $k = 0,\ a^{(0)} = a,\ b^{(0)} = b,\ x^{(0)} = \dfrac{a^{(0)} + b^{(0)}}{2}$

for k = 0, 1, ... **do**

▷ Dove φ è la sezione aurea

$c^{(k)} = a^{(k)} + \dfrac{b^{(k)} - a^{(k)}}{\varphi + 1}$

$d^{(k)} = a^{(k)} + \dfrac{b^{(k)} - a^{(k)}}{\varphi}$

if $(\phi(c^{(k)}) > \phi\left(d^{(k)}\right))$ **then**

▷ Nuovo intervallo [c; b]

$a^{(k+1)} = c^{(k)};\ b^{(k+1)} = b^{(k)}$

else

▷ Nuovo intervallo [a; d]

$a^{(k+1)} = a^{(k)};\ b^{(k+1)} = d^{(k)}$

end if

$x^{(k+1)} = \dfrac{a^{(k+1)} + b^{(k+1)}}{2}$

end for

4.2 Metodi di Discesa (Line Search)

Algorithm 27 Metodo di discesa (generale)

Dato $\vec{x}^{\,(0)} \in \mathbb{R}^n$
for k = 0, 1, ... **do**
$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \alpha_k \, \vec{d}^{\,(k)}$
end for

Algorithm 28 Metodo di Backtracking (per il calcolo di α_k)

$\alpha_k = 1$, scelgo c_1 e $\rho \in \left[\frac{1}{10}; \frac{1}{2}\right]$
while $\phi\left(\vec{x}^{\,(k)} + \alpha_k \, \vec{d}^{\,(k)}\right) > \phi\left(\vec{x}^{\,(k)} + c_1 \alpha_k \nabla\left(\phi(\vec{x}^{\,(k)})\right) \, \vec{d}^{\,(k)}\right)$ **do**
▷ While **not** (prima condizione di Wolfe)
$\alpha_k = \rho \, \alpha_k$
end while

Algorithm 29 Metodo di Newton per problemi di ottimo

$\vec{x}^{\,(0)}$
for k = 0, 1, ... **do**
assemblo il vettore $\nabla\phi\left(\vec{x}^{\,(k)}\right) \in \mathbb{R}^n$ e $H_\varphi\left(\vec{x}^{\,(k)}\right) \in \mathbb{R}^{n \times n}$
risolvo il sistema $H_\varphi\left(\vec{x}^{\,(k)}\right) \, \vec{d} = -\nabla\phi\left(\vec{x}^{\,(k)}\right)$
$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \alpha_k \, \vec{d}$
▷ il peso α_k viene scelto con il metodo di backtracking
▷ per aumentare la robustezza dell'algoritmo
end for

Algorithm 30 Metodo BFGS (quasi-Newton)

Dato $\vec{x}^{\,(0)}$, $B^{(0)} = I$

for k= 0, 1, ... **do**

$d^{k+1} = -B^{(k)} \nabla\phi\left(\vec{x}^{\,(k)}\right)$

determino α_k con backtracking

$\vec{x}^{\,(k+1)} = \vec{x}^{\,(k)} + \alpha_k \vec{d}^{\,(k)}$

$\vec{\delta}^{\,(k+1)} = \vec{x}^{\,(k+1)} - \vec{x}^{(k)}$

$\vec{s}^{\,(k)} = \nabla\phi\left(\vec{x}^{\,(k+1)}\right) - \nabla\phi\left(\vec{x}^{\,(k)}\right)$

$$\rho_k = \frac{1}{\left(\vec{\delta}^{\,(k)}\right)^T \vec{s}^{\,(k)}}$$

$$B^{(k+1)} = \left(I - \rho_k \vec{\delta}^{(k)} \left(\vec{s}^{(k)}\right)^T\right) B^{(k)} \left(I - \rho_k \vec{s}^{(k)} \left(\vec{\delta}^{(k)}\right)^T\right) + \rho_k \vec{\delta}^{(k)} \left(\vec{\delta}^{(k)}\right)^T$$

end for