# Esercizio

*Massimo punteggio: 10. Limite di pagine relazione (file .pdf): max. 12*

Consideriamo il seguente problema differenziale con valori al contorno

$$u''(x) + e^{u(x)} = 0, \qquad u(0) = u(1) = 0,$$

con $x \in [0, 1]$.

Obiettivo è risolvere numericamente il problema assegnato.

A tale scopo, si può utilizzare il *metodo alle differenze finite* come segue. Si discretizzi il dominio $[0, 1]$ mediante $n + 1$ sottointervalli con la stessa ampiezza, cioé poniamo $x_i = ih, i = 0, \ldots, n + 1$ con $h = 1/(n + 1)$. Si cerca una soluzione approssimata $u_i \approx u(x_i)$ per $i = 1, \ldots, n$ imponendo le condizioni al contorno $u_0 = u_{n+1} = 0$.

Approssimando la derivata seconda $u''$ con le formule alle differenze centrali

$$u''(x_i) = \frac{1}{h^2}\left(u(x_{i-1}) - 2u(x_i) + u(x_{i+1})\right) + O(h^2),$$

il valore della funzione incognita $u$ nei punti della discretizzazione può essere calcolato risolvendo il sistema non lineare $F(u) = 0$, dove $F : \mathbb{R}^n \to \mathbb{R}^n$ ha componenti

$$F_i(u) = \frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) + e^{u_i}, \quad i = 1, \ldots, n. \tag{1}$$

1. Si utilizzi il metodo di Newton per la risoluzione del problema assegnato considerando come approssimazione iniziale il vettore
$$w^0 = \alpha(x_1(1 - x_1), \ldots, x_n(1 - x_n))$$
con i seguenti valori di $\alpha$ e parametri opportuni (massimo numero di iterazioni e tolleranza per il criterio di arresto):
   - $\alpha = 0, 10, 20, 50$.

   In particolare, si risolva il problema per $n = 24$, e si eseguano i tre seguenti grafici:
   (a) un grafico che mostri successive approssimazioni della soluzione per uno scelto valore di $\alpha$;
   (b) un grafico che mostri le soluzioni finali ottenute per i vari valori di $\alpha$;
   (c) un grafico che mostri la storia di convergenza, i.e. $\|F(w^k)\|, k = 0, \ldots,$ del metodo per i vari valori di $\alpha$;

   Si discuta la convergenza del metodo di Newton alla luce dei risultati teorici e dei risultati ottenuti.

2. Si ricorda che ad ogni iterazione il metodo di Newton richiede la soluzione di un sistema lineare: dato $w^k$, $w^{k+1} = w^k + p^k$ dove il passo di Newton $p^k$ è la soluzione del sistema
$$J(w^k)p^k = -F(w^k), \text{ per } k = 0, 1, \ldots$$
   - Analizzare le proprietà della matrice Jacobiana di $F$ in (1) e stabilire quali tra i metodi numerici visti per la risoluzione di sistemi lineari (diretti e iterativi) possono essere applicati per calcolare il passo di Newton facendo riferimento ad opportuni risultati teorici.
   - Per valori grandi di $n$ (es. $n = 5000$) scegliere il metodo più opportuno per il calcolo del passo di Newton e giustificare sperimentalmente tale scelta anche confrontandola con l'implementazione standard. A tal fine, richiamare il risolutore lineare scelto per il calcolo di $p^k$.

# RELAZIONE

## *Descrizione function e script*

Il lavoro svolto per la redazione del progetto finalizzato al superamento dell'esame del modulo 2, del corso di Metodi numerici per l'Ingegneria civile, si compone, oltre che della presente relazione, di numero 5 script e 12 function.

Di seguito vi è una breve descrizione di quanto eseguito in MatLab senza entrare nel merito di come i codici siano stati implementati, ma con il mero intento di suggerire i risultati che essi restituiscono, salvo casi particolari in cui ci si rende conto sia di fondamentale importanza esporre come si è giunti alla soluzione finale.

- *Script:* **Progetto_punto1**: tramite questo script è possibile visionare l'implementazione necessaria alla risoluzione del punto 1 della traccia d'esame. Attraverso un menu suddiviso nei punti *a*, *b* e *c* l'utente può scegliere quale grafico e quali risultati visualizzare. Le function richiamate nel presente script sono:
  - *Function* **discretizzazione**: nella quale viene discretizzato il dominio, tramite la creazione di un vettore colonna *x* di *n+2* componenti, e definita l'approssimazione iniziale *w_0*, attraverso un vettore colonna di *n* componenti.
  - *Function* **sistema**: contenente la definizione del sistema non lineare *F(u)*.
  - *Function* **jacobiana**: contenente la definizione della matrice jacobiana *Jac*.
  - *Function* **newton**: coincidente con la *function newton_nd*, all'interno della quale si è proceduto al rinomino delle variabili (di input e output) in modo che restituisca una visualizzazione dei risultati direttamente in linea con la nomenclatura usata nella traccia.
  - *Function* **newton_plot**: coincidente con la *function* **newton**, sopra descritta, nella quale sono stati aggiunti i comandi necessari per poter generare, ad ogni iterazione, un grafico contenente l'approssimazione di una radice del sistema non lineare.

- *Script* **proprieta_Jac**: in tale script sono state valutate, con apporto numerico, le principali caratteristiche della matrice jacobiana. La simmetria è stata valutata verificando che la norma di (Jac - Jac') risultasse nulla; la diagonale dominanza è stata verificata controllando che il valore assoluto delle componenti poste sulla diagonale di Jac fosse maggiore o uguale alla somma dei valori assoluti delle restanti componenti sulla stessa riga e/o colonna; la definizione è stata valutata tramite la funzione 'chol' di MatLab.

- *Script* **convergenza_iterativi**: è stata verificata la convergenza dei metodi iterativi (Jacobi, Guass – Seidel e SOR) calcolando il raggio spettrale e l'$\omega$ ottimale (solo per il metodo SOR). Le function necessarie sono state:
  - *Function* **discretizzazione**: precedentemente esposta.

- *Function* **jacobiana**: precedentemente esposta.
- *Function* **omega**: la quale valuta $\omega$ ottimale, per il metodo SOR, e il raggio spettrale minimo ad esso connesso. In essa vengono definite inoltre le matrici D (diagonale), L (triangolare inferiore) e U (triangolare superiore) necessarie per il calcolo della matrice di iterazione dei restanti metodi iterativi.

❖ *Script* **confronto_diretti_iterativi**: nel presente script è stato risolto, per $\alpha = 10$, il sistema lineare
$J(w^0) \cdot u^1 = F(w^0)$ tramite i metodi iterativi (Jacobi, Guass – Seidel e SOR), i metodi diretti (eliminazione di Gauss e pivoting parziale) ed il backslash, valutando i tempi di risoluzione, il residuo relativo restituito e le iterazioni necessarie. Le function rievocate sono:

- *Function* **discretizzazione**: precedentemente esposta.
- *Function* **jacobiana**: precedentemente esposta.
- *Function* **sistema**: precedentemente esposta.
- *Function* **jacobi**: fornita durante il corso la quale implementa il metodo di Jacobi.
- *Function* **gseidel**: fornita durante il corso la quale implementa il metodo di Guass – Seidel.
- *Function* **sor**: fornita durante il corso la quale implementa il metodo SOR.
- *Function* **gauss_nopiv**: fornita durante il corso la quale implementa il metodo di eliminazione di Gauss.
- *Function* **secs2hms**: fornita durante il corso la quale converte il tempo in secondi in una stringa restituendo il tempo in ore, minuti e secondi.

❖ *Script:* **Progetto_punto2**: sono stati applicati, all'interno del metodo di Newton, e confrontati: il risolutore scelto, pivoting parziale, e mldivide. Si fa restituire una stringa riportante il miglior risolutore per il calcolo del passo $p^k$. Le function richiamate sono le seguenti:

- *Function* **discretizzazione**: precedentemente esposta.
- *Function* **jacobiana**: precedentemente esposta.
- *Function* **sistema**: precedentemente esposta.
- *Function* **newton_LUP**: corrispondente alla *function* ***newton***, nella quale è stato sostituito il risolutore lineare standard ( \ ) con il comando [L,U,P] = lu (Jac) per la fattorizzazione, e la conseguente risoluzione dei sistemi lineari da essa derivanti.
- *Function* **newton**: precedentemente esposta.

Ove necessario, per una migliore comprensione, si rimanda ai codici allegati.

## *Analisi critica dei risultati*

Considerando l'esercizio oggetto della traccia di esame, in cui era richiesta la ricerca delle soluzioni di un problema differenziale, si è proceduto alla risoluzione numerica dello stesso, attraverso codici MatLab, già descritti precedentemente, e le dovute considerazioni teoriche, alla base di quanto sviluppato ed ottenuto, che ora andremo ad esporre.

Come dati di input ci venivano forniti l'equazione differenziale, il dominio di appartenenza e due condizioni al contorno, qui riportate.

$$u''(x) + e^{u(x)} = 0 \qquad u(0) = u(1) = 0 \qquad x \in [0,1]$$

Veniva quindi restituito un sistema non lineare F(u) il quale è stato risolto attraverso l'applicazione del metodo di Newton, come consigliato dalla traccia, ottenendo soluzioni approssimate del problema, per diversi vettori di approssimazione iniziale. Per applicare tale metodo è necessaria la discretizzazione del dominio che, come indicatoci, è stato suddiviso in *n+1* sottointervalli attraverso la creazione di un vettore *x* di *n+2* componenti. Inoltre la traccia chiedeva di approssimare la derivata seconda di *u* tramite le formule alle differenze centrali.

I dati appena esposti sono stati inseriti come input in un codice MatLab. È stato inoltre necessario definire le tolleranze per il criterio di arresto e il numero di iterazioni massime. A tal proposito si sono assunti dei valori iniziali successivamente ritoccati in modo da garantire una giusta approssimazione, nonché la buona norma che suggerisce di verificare se sia possibile raggiungere un giusto compromesso tra la precisione del risultato e il costo computazionale.

Considerando che generalmente la scelta di tali criteri viene eseguita attraverso l'esperienza dell'operatore si è proceduto con un metodo semi-iterativo al fine di giungere alla convenzione di cui sopra. Si è quindi dapprima scelto un numero massimo di iterazioni pari a 100 e di tolleranza (per il criterio di arresto sulle iterate e sulla funzione) pari ad 1e-16, ovvero dell'ordine di grandezza della precisione di macchina, in modo tale da ottenere una soluzione finale il meno possibile affetta da errore. La scelta iniziale è risultata ragionevole per $\alpha = 0$, in quanto il numero di iterazioni effettuate non era eccessivo, ma si è notato che per gli altri valori di $\alpha$, per cui era necessario sviluppare l'analisi, tale precisione comportava un costo computazionale troppo elevato rispetto ai risultati ottenuti. In particolare per $\alpha = 20$ era necessario un aumento di numero di iterazioni, in quanto, si verificava il criterio di arresto sulle iterazioni massime e non sulle tolleranze nonostante il metodo convergesse (il residuo diminuiva ad ogni iterata). Al fine di diminuire il costo computazionale si è quindi deciso di aumentare il valore delle tolleranze ad 1e-13, lasciando invariato il numero di iterazioni massime.

## *Punto 1*

Come già esposto consultando il codice [script: *Progetto_Punto1*] si dà la possibilità all'utente di scegliere cosa visualizzare, attraverso l'apposito menu, e di plottare singolarmente i grafici richiesti, i quali sono riportati di seguito al fine di valutare gli esiti ottenuti.

a) Successive approssimazioni della soluzione per uno scelto valore di α:

Partendo dalla function 'newton_nd', fornitaci durante il corso, è stato aggiunto il comando plot all'interno del ciclo for per ogni iterazione [function: *newton_plot*] in modo da ottenere l'andamento delle varie approssimazioni per α = 10 :

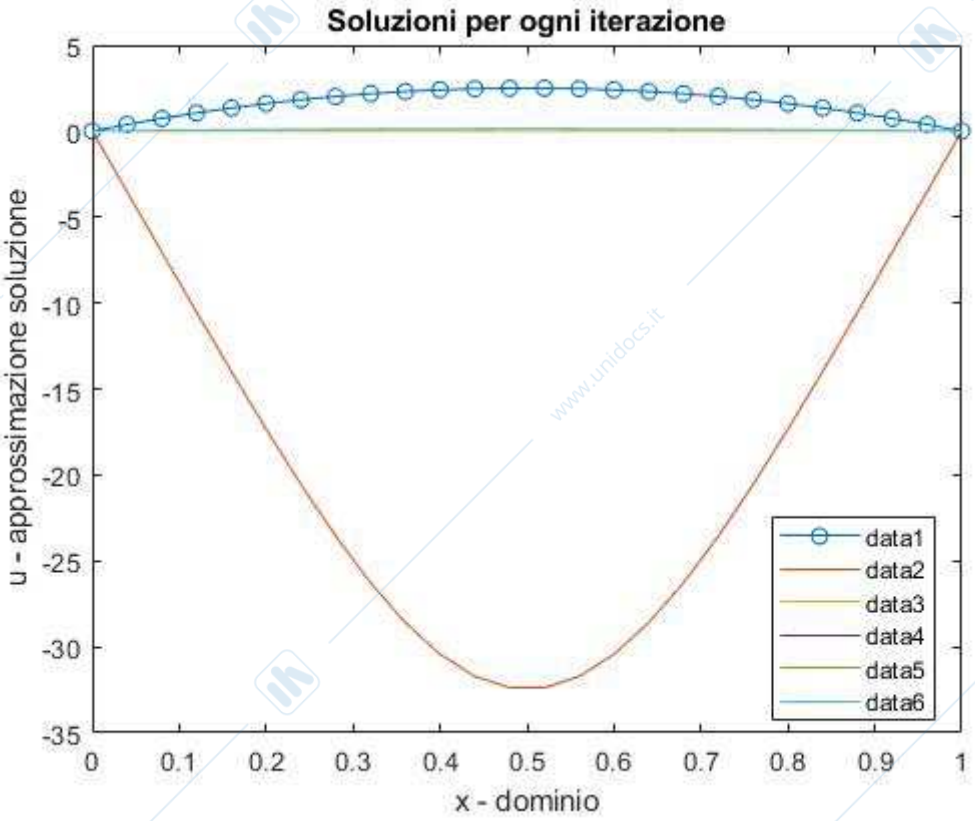


*Figura a. Approssimazioni successive della soluzione per α= 10*

Dal grafico ottenuto si può intuire la velocità di convergenza del metodo utilizzato, infatti tra l'iterazione n°1 e la n°2 si ha una forte differenza dell'approssimazione che diminuisce sempre di più nelle iterate successive, tanto da non riuscire a distinguerle graficamente in maniera significativa, se non attraverso un ingrandimento del grafico stesso. Alle medesime considerazioni si può giungere analizzando i dati riportati in tabella, in particolare la ||du|| diminuisce sempre di più ad ogni iterazione.

| N° ITERAZIONI | ||Fu|| - RESIDUO | ||du|| | ||du||/||duold||^2 |
|---|---|---|---|
| 0 | 6.67424578e+01 | | |
| 1 | 1.14543960e+03 | 1.19888606e+02 | 1.199e+02 |
| 2 | 4.84605652e+00 | 1.10783542e+02 | 7.708e-03 |
| 3 | 3.21341025e-02 | 5.06841477e-01 | 4.130e-05 |
| 4 | 1.86116710e-06 | 3.63418540e-03 | 1.415e-02 |
| 5 | 5.56038884e-14 | 2.07564950e-07 | 1.572e-02 |

CRITERIO ARRESTO = 2

Non avendo la soluzione esatta non è stato possibile valutare l'errore, ma usando come sua stima la differenza tra due iterate si denota che l'ordine di grandezza di quest'ultimo è circa il doppio ad ogni iterata. Considerando inoltre che il rapporto nell'ultima colonna è pressoché costante si può

definire la convergenza più che quadratica. Dalla tabella si riscontra anche che il metodo iterativo si è interrotto con il criterio di arresto 2, ovvero è stata raggiunta la tolleranza sulla funzione.

Questo dà indicazione del fatto che il metodo è giunto a convergenza, in quanto, il numero di iterazioni massime non è stato raggiunto, al contrario sono state bastevoli 5 iterazioni per giungere ad un errore sulla funzione pari alla tolleranza impostata.

b) Soluzioni finali ottenute per i valori di α:

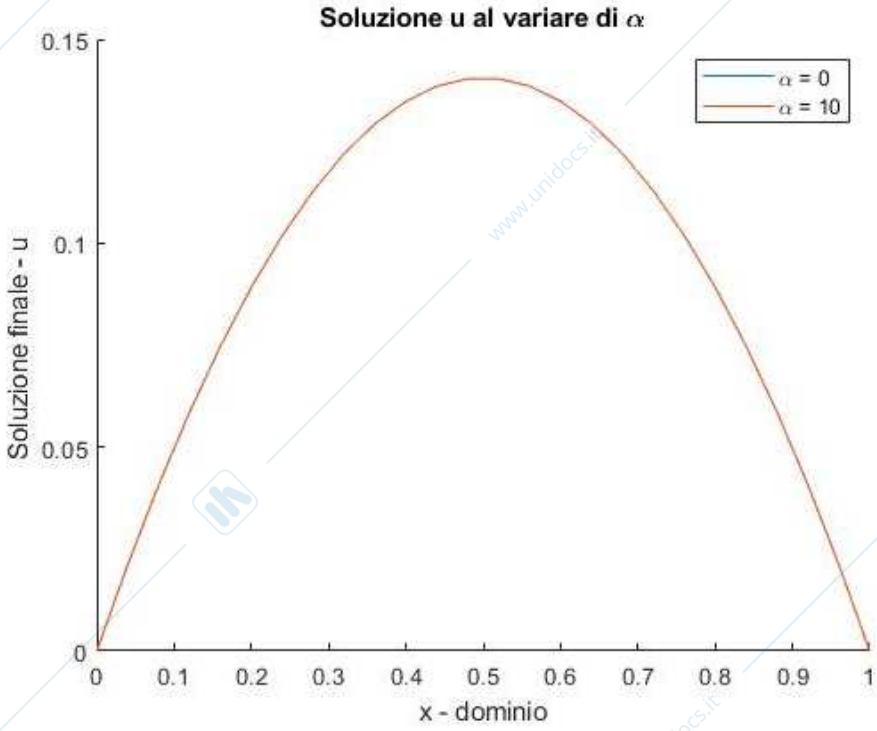


*Figura b.1 - Soluzione finale per α = 0 e α = 10*

Graficando le soluzioni finali per α = 0 e α = 10 è istantaneo notare che queste coincidano. Ciò denota come entrambe le approssimazioni iniziali siano vicine alla soluzione esatta. Inoltre, dal numero di iterazioni necessarie per giungere a convergenza, si può dedurre che α = 0 generi un'approssimazione iniziale più vicina alla soluzione rispetto a quella generata da α = 10.

| α | N° ITERAZIONI | ‖Fu‖ - RESIDUO | CRITERIO ARRESTO |
|---|---|---|---|
| 0 | 3 | 4.817703e-14 | 2 |
| 10 | 5 | 5.560389e-14 | 2 |
| 20 | 7 | 1.404428e-12 | 1 |
| 50 | 100 | 1.682682e+39 | 0 |

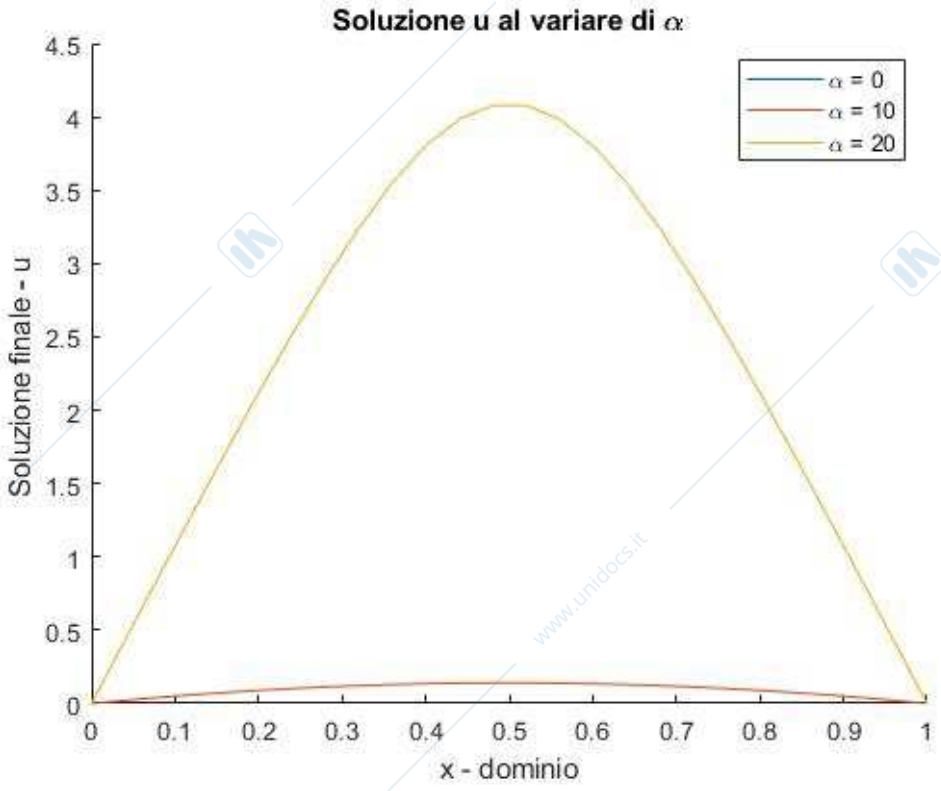


*Figura b.2 Soluzioni finali per α = 0, α = 10 e α = 20*

Per α = 20 il metodo di Newton converge ma con una soluzione finale diversa rispetto a quella ottenuta per α = 0 e α = 10. Da questo si evince la presenza di una seconda radice del problema differenziale vicina all'approssimazione iniziale generata da α = 20.

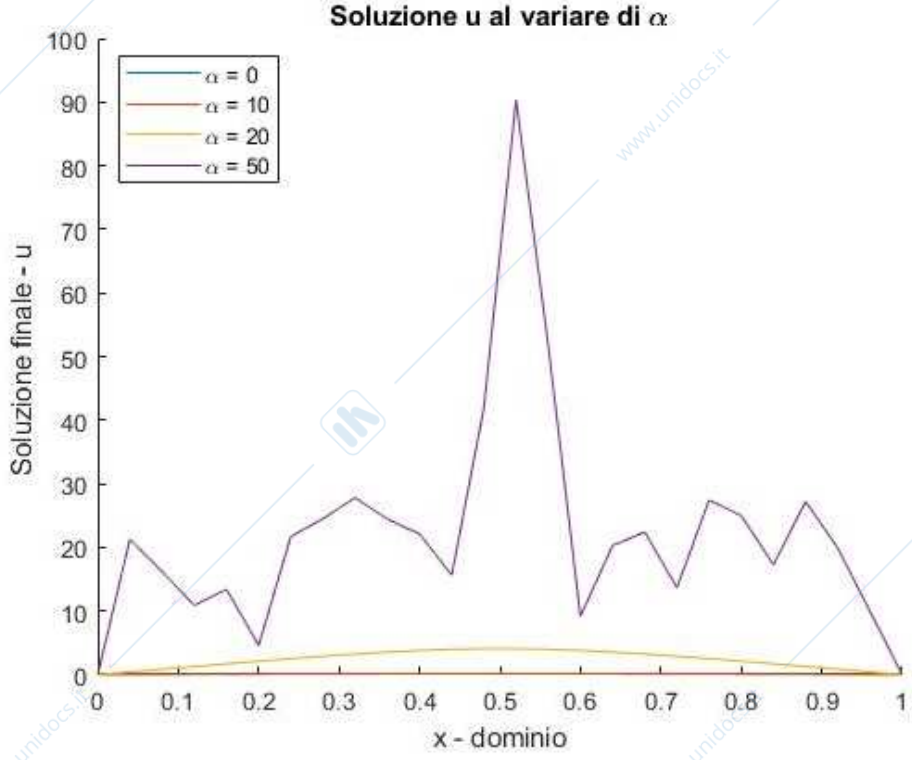


*Figura b.3 Soluzioni finali per ogni valore di α*

Per α = 50 il raggiungimento del numero di iterazioni massime e l'eccessivo ordine di grandezza del residuo (coincidente con la ||Fu|| relativa all'ultima iterazione) evidenzia come il metodo si allontani sempre di più dalla soluzione esatta.

MatLab inoltre, per quest'ultimo caso, restituisce un Warning sulla singolarità della matrice Jacobiana; il metodo con queste matrici non riesce a convergere.

Quanto ottenuto è perfettamente in linea con la teoria del metodo di Newton, la quale, prevede la convergenza solo per approssimazioni iniziali vicine alle soluzioni del problema.

c) Storia di convergenza del metodo per i vari valori di α

Infine, plottando il residuo all'aumentare del numero di iterazioni, si ha una definitiva conferma di quanto emerso precedentemente. Già a colpo d'occhio è evidente come per α = 0, 10, 20 il residuo diminuisca al crescere delle iterazioni fino ad arrivare a convergenza, mentre per α = 50 si ha un'iniziale diminuzione di quest'ultimo seguita da un incremento sempre maggiore, che mostra l'allontanarsi dal raggiungimento della convergenza. In assenza di limitazioni sulle iterazioni massime il metodo avrebbe tentato di giungere a convergenza, aumentando sempre più il residuo, fino a restituire come valore di quest'ultimo la sigla NaN.

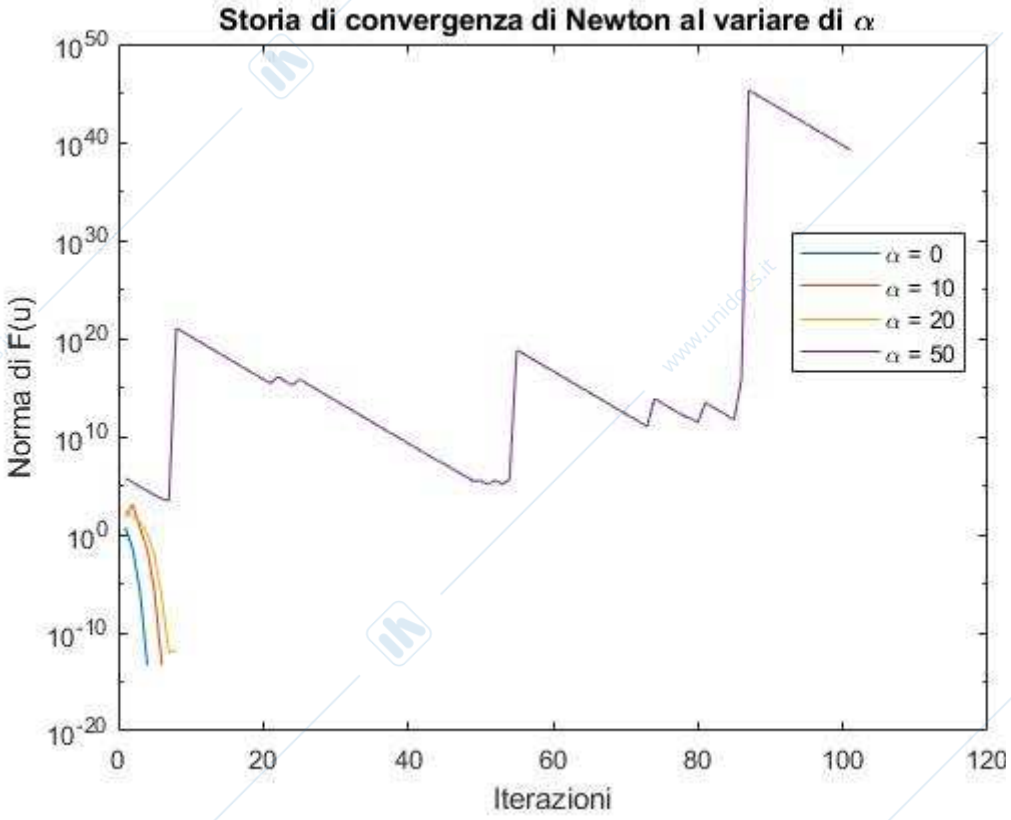


*Figura c - Storia di convergenza per n° iterazioni max = 100*

Per ottenere i risultati appena esposti è stato utilizzato, come risolutore del sistema lineare, il comando MatLab mldivide ' \ '.

## *Punto 2*

Per scegliere un metodo di risoluzione alternativo tra quelli studiati (diretti e iterativi) è stato necessario analizzare le caratteristiche della matrice Jacobiana [script : *proprietà_Jac*] .

$$Jac_{nxn} = \begin{bmatrix} -\frac{2}{h^2} + e^{u(1)} & \frac{1}{h^2} & & & & & & \\ \frac{1}{h^2} & -\frac{2}{h^2} + e^{u(2)} & \frac{1}{h^2} & & & & & \\ & \frac{1}{h^2} & -\frac{2}{h^2} + e^{u(3)} & \frac{1}{h^2} & & & & \\ & & \frac{1}{h^2} & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & -\frac{2}{h^2} + e^{u(n-2)} & \frac{1}{h^2} & & \\ & & & & \frac{1}{h^2} & -\frac{2}{h^2} + e^{u(n-1)} & \frac{1}{h^2} & \\ & & & & & \frac{1}{h^2} & -\frac{2}{h^2} + e^{u(n)} \end{bmatrix}$$

Scegliendo il vettore u avente tutte le componenti (24) nulle e n = 24 si ottiene la seguente matrice jacobiana:

$$Jac_{24x24} = \begin{bmatrix} -1249 & 625 & & & & & & & & & & \\ 625 & \ddots & \ddots & & & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & & & \\ & & \ddots & \ddots & \ddots & & & & & & & \\ & & & \ddots & \ddots & \ddots & & & & & & \\ & & & & \ddots & \ddots & \ddots & & & & & \\ & & & & & \ddots & \ddots & \ddots & & & & \\ & & & & & & \ddots & \ddots & 625 & & & \\ & & & & & & & 625 & -1249 \end{bmatrix}$$

- E' possibile constatare già visivamente che la stessa è una tridiagonale simmetrica (proprietà verificata anche matematicamente).

- Per la valutazione della definitezza si è proceduto all'applicazione della fattorizzazione di Cholesky. Il comando 'chol (Jac)' in Matlab, se Jac è definita positiva, restituisce una matrice triangolare superiore, derivante dalla fattorizzazione, ed un parametro flag = 0; viceversa, se Jac non è definita positiva, restituisce un valore intero positivo del parametro flag. Nel caso in esame flag = 2, quindi la jacobiana non è una matrice definita positiva. In alternativa sarebbe stato necessario il calcolo di tutti gli autovalori della matrice, ma tale procedimento è gravato da un costo computazionale elevato.

- Per quanto concerne lo studio della dominanza diagonale della matrice, tale caratteristica non risulta verificata né per righe né per colonne.
  Questa condizione è stata appurata imponendo la diseguaglianza tra le componenti della diagonale principale e la somma delle restanti componenti delle righe o delle colonne e constatando la veridicità di quanto imposto.

*ANALISI METODI DI RISOLUZIONE DEI SISTEMI LINEARI:*

A causa della definitezza della matrice non è possibile applicare né l'algoritmo di ***Cholesky*** né il metodo del ***Gradiente coniugato***.

Per lo stesso motivo risulterebbe imprudente anche l'applicazione del metodo ***SOR***, in quanto, nel caso di matrici non definite positive, non si ha la certezza della convergenza.
Inoltre la matrice, non essendo a diagonale dominanza, non da certezza di convergenza neanche del metodo di ***Jacobi*** e ***Gauss – Seidel***.

Per verificare la convergenza degli ultimi tre metodi analizzati è stato necessari valutare che il raggio spettrale delle matrici di iterazione fosse minore di 1. Nel caso di jacobiana di dimensioni 500x500, sono stati ottenuti i seguenti risultati [script: *convergenza_iterativi*]:

| $\rho$ (J) – Jacobi | $\rho$ (G) – Gauss – Seidel | $\omega$ ottimale SOR | $\rho$ (H($\omega$)) – SOR |
|---|---|---|---|
| 9.999991e-01 | 9.999983e-01 | 1.9 | 9.999669e-01 |

È possibile notare che i metodi giungono a convergenza, ma molto lentamente, infatti $\rho$ è sempre prossimo a 1.

La particolare struttura della matrice in esame farebbe protendere la scelta verso il ***metodo di eliminazione di Gauss*** piuttosto che di uno basato sul pivoting in quanto scambiando righe e/o colonne la struttura della jacobiana verrebbe rovinata. Il metodo di Gauss, però, per matrici senza caratteristiche di diagonale dominanza, quali la jacobiana oggetto di studio, non risulta sempre adatto. Un esempio è il caso in cui si presentino elementi nulli sulla diagonale principale. Per tale motivo sarebbe preferibile non utilizzarlo. Quand'anche non si verificasse questa ipotesi, un algoritmo basato sul pivoting, eviterebbe moltiplicatori troppo grandi, aumentando la stabilità dell'algoritmo stesso, questi infatti porterebbero ad un'amplificazione degli errori di arrotondamento, con conseguente instabilità. Per prevenire quanto esposto è necessario evitare elementi pivot troppo piccoli e l'algoritmo del ***pivoting parziale*** ci si assicura che ad ogni passo si sta usando l'elemento pivot più grande possibile.
Il ***pivoting totale*** viene escluso in quanto è un metodo che viene molto rallentato dalla ricerca del più piccolo elemento pivot contenuto sia nelle righe che nelle colonne. Seppur questo rallentamento porti ad un miglioramento della stabilità questo risulta essere solo marginale.

A riprova di quanto detto è stato risolto il sistema lineare in esame considerando n = 500 (essendo troppo oneroso fare tale valutazione per n più grandi) e $\alpha = 10$ [script: *confronto_diretti_iterativi*].

| | METODO | TEMPO | RESIDUO RELATIVO | ITERAZIONI |
|---|---|---|---|---|
| METODI DIRETTI | Gauss NO pivoting | 9 mins, 4.71 secs | 6.953047e-11 | \ |
| | LUP | 0.00 secs | 1.196425e-10 | \ |
| | Mldivide | 0.00 secs | 1.230232e-10 | \ |
| METODI ITERATIVI | Jacobi | 8.91 secs | 8.744095e-01 | 1000 |
| | Gauss-Seidel | 8.74 secs | 8.298668e-01 | 1000 |
| | SOR | 9.37 secs | 7.144123e-01 | 1000 |

In generale, per matrici di grandi dimensioni, si tende a prediligere un metodo iterativo rispetto ad uno diretto, per evitare un'eccessiva occupazione di memoria dovuta al principio su cui si basano tali metodi. Nel caso oggetto di studio però, dovendo risolvere un sistema lineare contenuto in

Newton, metodo già di per sé iterativo, potrebbe risultare deleteria l'applicazione combinata di un ulteriore metodo iterativo, in quanto, tale procedimento, può creare problemi di convergenza del metodo di Newton. Inoltre si renderebbe necessaria l'imposizione di una tolleranza avente ordine molto piccolo, in modo tale da ottenere una soluzione del sistema lineare caratterizzata da un residuo relativo il più piccolo possibile. Ricercare codesti risultati, mediante la jacobiana costruita, porterebbe ad un numero di iterazioni eccessivo, come è possibile notare dai risultati ottenuti sia nel confronto tra i metodi sia nel calcolo del raggio spettrale, e di conseguenza ad un costo computazionale insostenibile.
Considerando le risultanze per una matrice di dimensioni n = 500, si può ipotizzare che all'aumentare delle dimensioni si ottengano risultati che andranno sempre peggiorando. In particolare saranno necessarie un numero di iterazioni sempre maggiori per ridurre l'ordine di grandezza del residuo relativo.
Si fa notare che essendo la jacobiana una matrice sparsa, nel caso in esame, il dispendio di memoria richiesto per l'applicazione di metodi diretti risulta meno oneroso, per tali motivi la scelta è ricaduta tra questi. In particolare il metodo di Gauss senza pivoting risulta avere una precisione superiore agli altri, ma a causa dell'eccessivo tempo impiegato nella risoluzione di un singolo sistema lineare e considerando la numerosità di questi ultimi contenuta all'interno del metodo di Newton si è giunti a preferire, per il confronto con l'implementazione standard, l'applicazione del pivoting parziale in quanto risultato più veloce e con un residuo paragonabile al precedente.

Dunque, per scegliere il metodo più opportuno, per il calcolo del passo di Newton è stato risolto il problema differenziale sia con l'implementazione standard (risolutore lineare: mldivide) sia con l'applicazione del metodo ritenuto maggiormente adatto a valle delle considerazioni teoriche finora esposte [script: *Progetto_punto2*].

Seguendo le indicazioni riportate nella traccia si è scelto di porre n = 4900, lasciando invariati i valori delle tolleranze e del numero di iterazioni massime. Si è inoltre deciso di eseguire il calcolo del passo di Newton per un'approssimazione iniziale generata da $\alpha = 10$. I risultati ottenuti sono riportati di seguito.

***RISOLUTORE LUP:***

| N° ITERAZIONI | ‖Fu‖ - RESIDUO | ‖du‖ | ‖du‖/‖duold‖^2 |
|---|---|---|---|
| 0 | 9.71176749e+02 | | |
| 1 | 1.55342713e+04 | 1.62797217e+03 | 1.628e+03 |
| 2 | 6.77413223e+01 | 1.50049493e+03 | 5.662e-04 |
| 3 | 4.48588359e-01 | 7.08446950e+00 | 3.147e-06 |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| 23 | 2.24962860e-08 | 1.47858162e-13 | 4.110e+11 |
| 24 | 2.23998302e-08 | 7.05391952e-13 | 3.227e+13 |
| 25 | 2.16854916e-08 | 4.16519891e-13 | 8.371e+11 |
| 26 | 2.23521264e-08 | 8.61608825e-14 | 4.966e+11 |

***RISOLUTORE MLDIVIDE:***

| N° ITERAZIONI | ‖Fu‖ - RESIDUO | ‖du‖ | ‖du‖/‖duold‖^2 |
|---|---|---|---|
| 0 | 9.71176749e+02 | | |
| 1 | 1.55342713e+04 | 1.62797217e+03 | 1.628e+03 |
| 2 | 6.77413223e+01 | 1.50049493e+03 | 5.662e-04 |
| 3 | 4.48588359e-01 | 7.08446949e+00 | 3.147e-06 |
| 4 | 2.58201890e-05 | 5.06526058e-02 | 1.009e-03 |
| 5 | 2.17999503e-08 | 2.87535157e-06 | 1.121e-03 |
| 6 | 2.23820018e-08 | 1.24139527e-12 | 1.502e-01 |
| 7 | 2.13093778e-08 | 4.52367225e-13 | 2.935e+11 |
| 8 | 2.26523802e-08 | 4.91596552e-14 | 2.402e+11 |

| RISOLUTORE LINEARE | N° ITERAZIONI | ‖Fu‖ - RESIDUO | CRITERIO ARRESTO |
|---|---|---|---|
| LUP | 26 | 2.235213e-08 | 1 |
| MLDIVIDE | 8 | 2.265238e-08 | 1 |

Confrontando i due risolutori si è notato un comportamento molto simile in quanto, il residuo ottenuto è pressoché identico. Si verifica in entrambi i casi che il metodo si interrompe per il raggiungimento della tolleranza sulle iterate. Ciò che fa protendere verso la scelta dell'implementazione standard anziché quella contenente il metodo di Gauss con pivoting parziale è il numero di iterazioni. Infatti in quest'ultimo caso dalla sesta iterazione in poi la norma della differenza tra due successive soluzioni inizia ad oscillare attorno ad un certo ordine di grandezza, comportando un costo computazionale maggiore. Il risolutore 'MLDIVIDE' al contrario riesce a raggiungere la tolleranza richiesta in sole 8 iterazioni e questo denota una maggiore stabilità dell'algoritmo.

Questo è risultato perfettamente in linea con quanto si immaginava, infatti la funzione '\' di MatLab sceglie sempre, tra tutti gli algoritmi esistenti quello più appropriato per la risoluzione del sistema sia esaminando la matrice che la sua struttura.

In conclusione, dallo svolgimento di questo progetto, si evince come sia fondamentale, per il metodo di Newton, l'approssimazione iniziale e come, al variare di essa, il metodo possa o meno convergere. Inoltre, per quanto concerne i metodi di risoluzione dei sistemi lineari, la preferenza di uno anziché dell'altro è strettamente legata alle caratteristiche e alle dimensioni della matrice dei coefficienti.