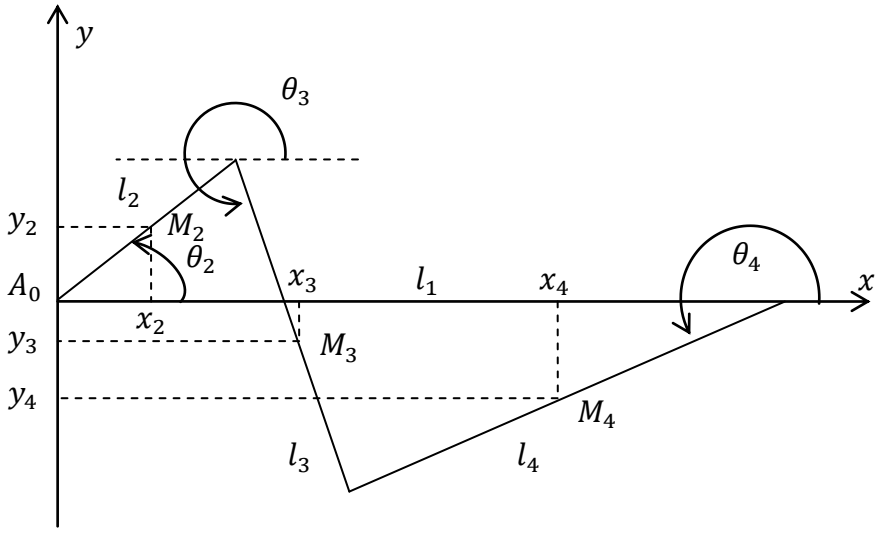
Indice

Quadrilatero articolato, configurazioni

1. Il quesito. E' assegnato il quadrilatero articolato le cui aste sono definite dalle seguenti misure

1.1) $$\begin{cases} l_2 = 0.2m \\ l_3 = 0.7m \\ l_4 = 0.5m \\ l_1 = 0.8m \end{cases}$$



Si assuma come sistema di coordinate quello indicato in figura, ovvero gli angoli assoluti delle aste, presi con segno (anomalia), e le coordinate dei baricentri di ciascuna asta, rospetto al sistema di riferimento A_0, x, y.

Allora si chiede di determinare le configurazioni del quadrilatero per una rotazione della manovella assegnata data da

1.2) $\theta_2 = 20°$

Si chiede inoltre di rappresentare la traiettoria del punto M_3 di mezzeria della biella (asta l_3) al variare in $[0,2\pi]$ dell'angolo θ_2 di rotazione della manovella (asta l_2).

2. Equazioni di vincolo. Il vettore delle coordinate lagrangiane sovrabbondanti è dato da

2.1) $$\{q\}^T = \{x_2 \quad y_2 \quad \theta_3 \quad x_3 \quad y_3 \quad \theta_4 \quad x_4 \quad y_4 \quad \theta_2\}^T$$

il quale viene sottoposto alla partizione

2.2) $$\{q\}^T = \{u|v\}^T = \{x_2 \quad y_2 \quad \theta_3 \quad x_3 \quad y_3 \quad \theta_4 \quad x_4 \quad y_4|\theta_2\}^T$$

Le equazioni di vincolo, immediatamente deducibili dalla figura sono

2.3) $$\{\Psi(q)\} = \{\Psi(u|v)\} = \begin{Bmatrix} x_2 - \frac{l_2}{2}\cos\theta_2 \\ y_2 - \frac{l_2}{2}\sin\theta_2 \\ x_3 - \frac{l_3}{2}\cos\theta_3 - x_2 - \frac{l_2}{2}\cos\theta_2 \\ y_3 - \frac{l_3}{2}\sin\theta_3 - y_2 - \frac{l_2}{2}\sin\theta_2 \\ x_4 - x_3 - \frac{l_3}{2}\cos\theta_3 - \frac{l_4}{2}\cos\theta_4 \\ y_4 - y_3 - \frac{l_3}{2}\sin\theta_3 - \frac{l_4}{2}\sin\theta_4 \\ x_4 - l_1 + \frac{l_4}{2}\cos\theta_4 \\ y_4 + \frac{l_4}{2}\sin\theta_4 \end{Bmatrix} = 0$$

Lo jacobiano relativo alle coordinate dipendenti si scrive dunque

$$2.4) \quad [\Psi(u)] = \begin{bmatrix} \frac{\partial \Psi_1}{\partial x_2} & \frac{\partial \Psi_1}{\partial y_2} & \frac{\partial \Psi_1}{\partial \theta_3} & \frac{\partial \Psi_1}{\partial x_3} & \frac{\partial \Psi_1}{\partial y_4} & \frac{\partial \Psi_1}{\partial \theta_4} & \frac{\partial \Psi_1}{\partial x_4} & \frac{\partial \Psi_1}{\partial y_4} \\ \frac{\partial \Psi_2}{\partial x_2} & \frac{\partial \Psi_2}{\partial y_2} & \frac{\partial \Psi_2}{\partial \theta_3} & \frac{\partial \Psi_2}{\partial x_3} & \frac{\partial \Psi_2}{\partial y_4} & \frac{\partial \Psi_2}{\partial \theta_4} & \frac{\partial \Psi_2}{\partial x_4} & \frac{\partial \Psi_2}{\partial y_4} \\ & & & & \cdots & & & \\ \frac{\partial \Psi_8}{\partial x_2} & \frac{\partial \Psi_8}{\partial y_2} & \frac{\partial \Psi_8}{\partial \theta_3} & \frac{\partial \Psi_8}{\partial x_3} & \frac{\partial \Psi_8}{\partial y_4} & \frac{\partial \Psi_8}{\partial \theta_4} & \frac{\partial \Psi_8}{\partial x_4} & \frac{\partial \Psi_8}{\partial y_4} \end{bmatrix}$$

e a conti fatti si ha

$$2.5) \quad [\Psi(u)] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & \frac{l_3}{2}\sin\theta_3 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -\frac{l_3}{2}\cos\theta_3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{l_3}{2}\sin\theta_3 & -1 & 0 & \frac{l_4}{2}\sin\theta_4 & 1 & 0 \\ 0 & 0 & -\frac{l_3}{2}\cos\theta_3 & 0 & -1 & -\frac{l_4}{2}\cos\theta_4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -\frac{l_4}{2}\sin\theta_4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{l_4}{2}\cos\theta_4 & 0 & 1 \end{bmatrix}$$

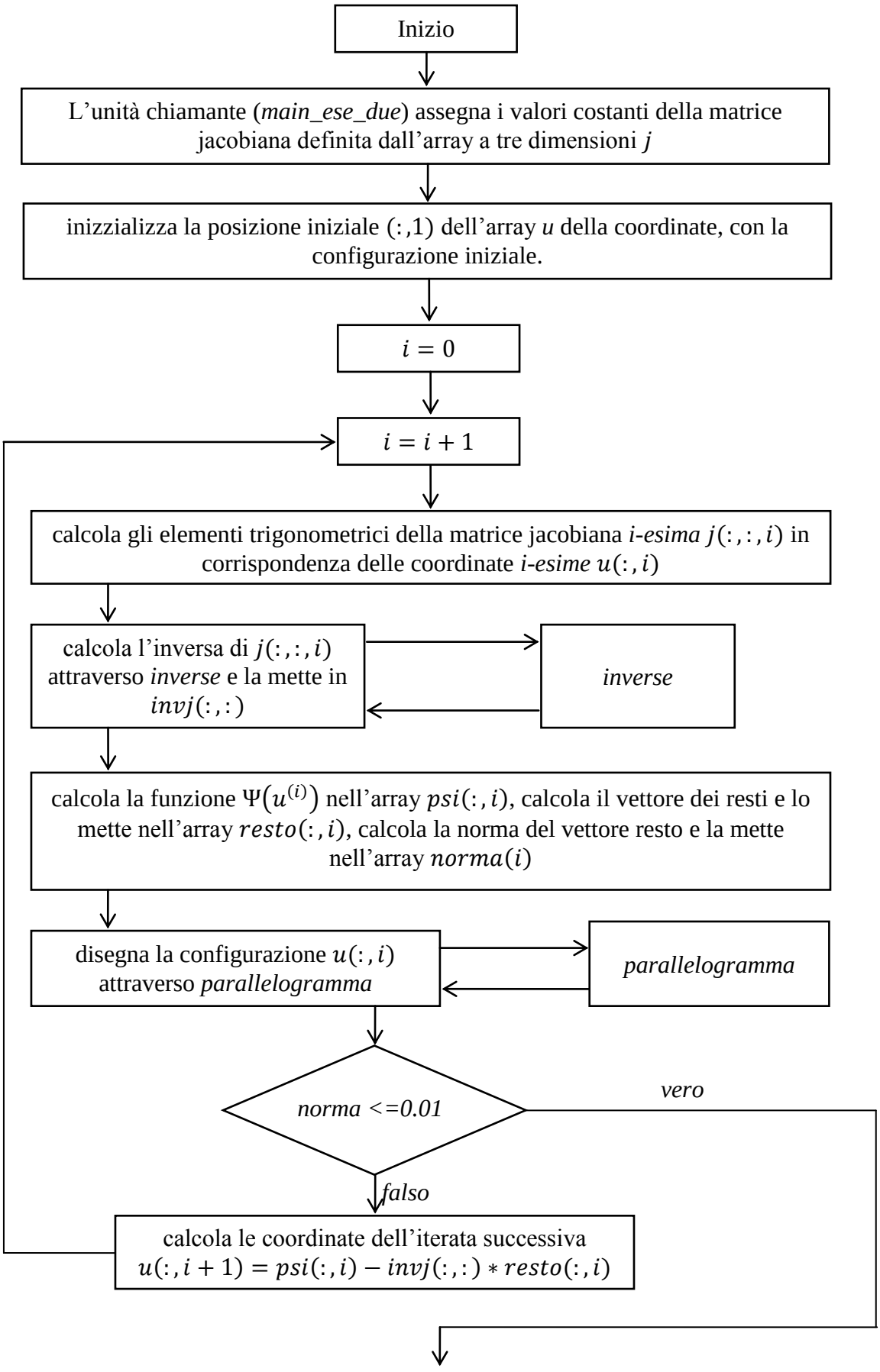
3. Metodo di Newton-Raphson. Il metodo numerico in parola applicato alla funzione vettoriale **2.3** di 8 elementi in 9 variabili, di cui la nona considerata costante, porge

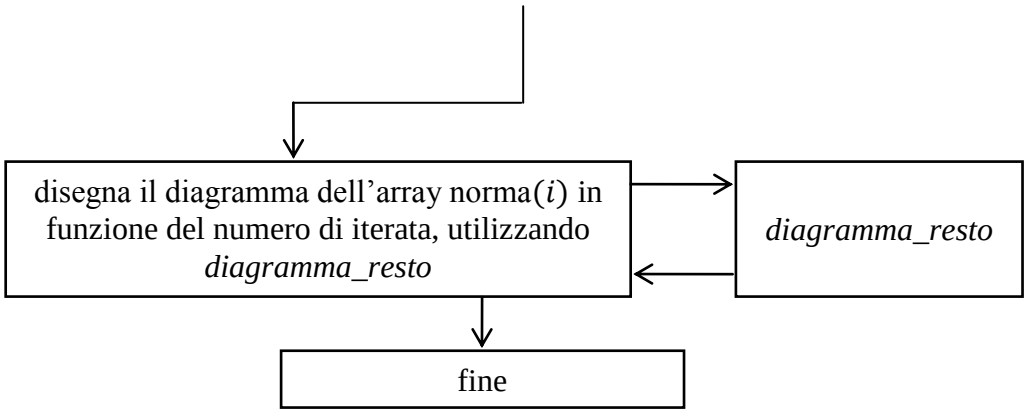
$$3.1) \quad \{u^{(i+1)}\} = \{u^{(i)}\} - [\Psi(u)]^{-1}\{\Psi(u^{(i)}|\theta_2)\}$$

Ovvero, per esteso

$$3.2) \quad \begin{Bmatrix} {x_2}^{(i+1)} \\ {y_2}^{(i+1)} \\ {x_3}^{(i+1)} \\ {y_3}^{(i+1)} \\ {\theta_3}^{(i+1)} \\ {x_4}^{(i+1)} \\ {y_4}^{(i+1)} \\ {\theta_4}^{(i+1)} \end{Bmatrix} = \begin{Bmatrix} {x_2}^{(i)} \\ {y_2}^{(i)} \\ {x_3}^{(i)} \\ {y_3}^{(i)} \\ {\theta_3}^{(i)} \\ {x_4}^{(i)} \\ {y_4}^{(i)} \\ {\theta_4}^{(i)} \end{Bmatrix} - \left[\Psi(u^{(i)})\right]^{-1} \begin{Bmatrix} {x_2}^{(i)} - \frac{l_2}{2}\cos\theta_2 \\ {y_2}^{(i)} - \frac{l_2}{2}\sin\theta_2 \\ {x_3}^{(i)} - \frac{l_3}{2}\cos{\theta_3}^{(i)} - {x_2}^{(i)} - \frac{l_2}{2}\cos\theta_2 \\ {y_3}^{(i)} - \frac{l_3}{2}\sin{\theta_3}^{(i)} - {y_2}^{(i)} - \frac{l_2}{2}\sin\theta_2 \\ {x_4}^{(i)} - {x_3}^{(i)} - \frac{l_3}{2}\cos{\theta_3}^{(i)} - \frac{l_4}{2}\cos{\theta_4}^{(i)} \\ {y_4}^{(i)} - {y_3}^{(i)} - \frac{l_3}{2}\sin{\theta_3}^{(i)} - \frac{l_4}{2}\sin{\theta_4}^{(i)} \\ {x_4}^{(i)} - l_1 + \frac{l_4}{2}\cos{\theta_4}^{(i)} \\ {y_4}^{(i)} + \frac{l_4}{2}\sin{\theta_4}^{(i)} \end{Bmatrix}$$

dove -lo si ricorda- l'anomalia θ_2 è assegnata dalla **1.2**. Assegnata dunque una configurazione iniziale $\{u^{(1)}\}$ la **3.2** consente di calcolare la configurazione successiva $\{u^{(2)}\}$, che sostituita a sua volta nella **3.2** permette di calcolare la configurazione $\{u^{(3)}\}$, e via di seguito. Si dimostra che questo procedimento fornisce configurazioni via via più vicine a quella che annulla il vettore **2.3** delle equazioni di vincolo.


Inizio
L'unità chiamante (*main_ese_due*) assegna i valori costanti della matrice jacobiana definita dall'array a tre dimensioni *j*
inizzializza la posizione iniziale (:,1) dell'array *u* della coordinate, con la configurazione iniziale.
$i = 0$
$i = i + 1$
calcola gli elementi trigonometrici della matrice jacobiana *i-esima* $j(:,:,i)$ in corrispondenza delle coordinate *i-esime* $u(:,i)$
calcola l'inversa di $j(:,:,i)$ attraverso *inverse* e la mette in $invj(:,:)$
inverse
calcola la funzione $\Psi(u^{(i)})$ nell'array $psi(:,i)$, calcola il vettore dei resti e lo mette nell'array $resto(:,i)$, calcola la norma del vettore resto e la mette nell'array $norma(i)$
disegna la configurazione $u(:,i)$ attraverso *parallelogramma*
parallelogramma
norma <=0.01
vero
falso
calcola le coordinate dell'iterata successiva $u(:,i+1) = psi(:,i) - invj(:,:) * resto(:,i)$



Il criterio per interrompere le iterazioni è quello di valutare la norma del secondo vettore a secondo membro della **3.2**, detto **vettore dei resti**. Evidentemente quando essa si stabilizza ad un valore prossimo a zero il metodo ha fornito la sua soluzione.

4. Algoritmo risolutivo. Per applicare il metodo numerico descritto, al problema della configurazione del parallelogramma sono state scritte in Fortran le seguenti unità:

- l'unità chiamante *main_ese_due*, la quale si occupa di calcolare la **3.2**;
- il modulo *mod_ese_due* contenente le costanti geometriche **1.1**,**1.2** del problema nonché le procedure di modulo seguenti
    - subroutine *inverse* (scaricata dal sito della ODU University) la quale si occupa di invertire la matrice jacobiana **2.5**;
    - subroutine *parallelogramma* la quale permette di rappresentare nel piano A_0, x, y la configurazione del parallelogramma alla generica iterazione del metodo;
    - subroutine *diagramma_resto* la quale esegue il grafico della norma del vettore resto in funzione del numero di iterazione.

Il diagramma di flusso complessivo sviluppato dal codice è indicato in figura. Si aggiunge che il programma realizza le seguenti ulteriori operazioni di output non indicate nel diagramma:

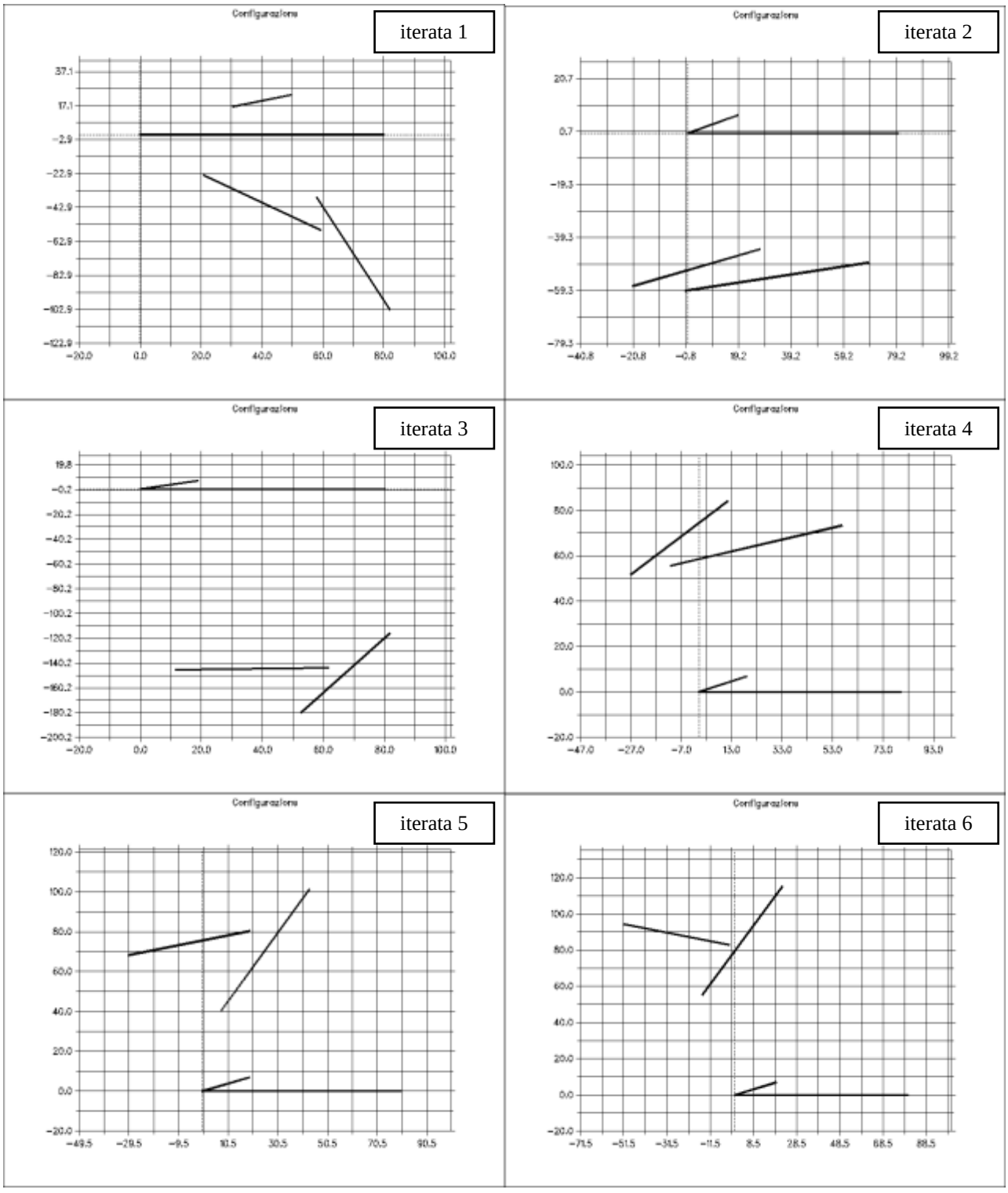
- ad ogni iterazione stampa sul monitor la configurazione, la matrice jacobiana, la sua inversa, il vettore dei resti e la sua norma;
- alla fine del flusso stampa sullo schermo il valore massimo e il minimo della norma del vettore dei resti.

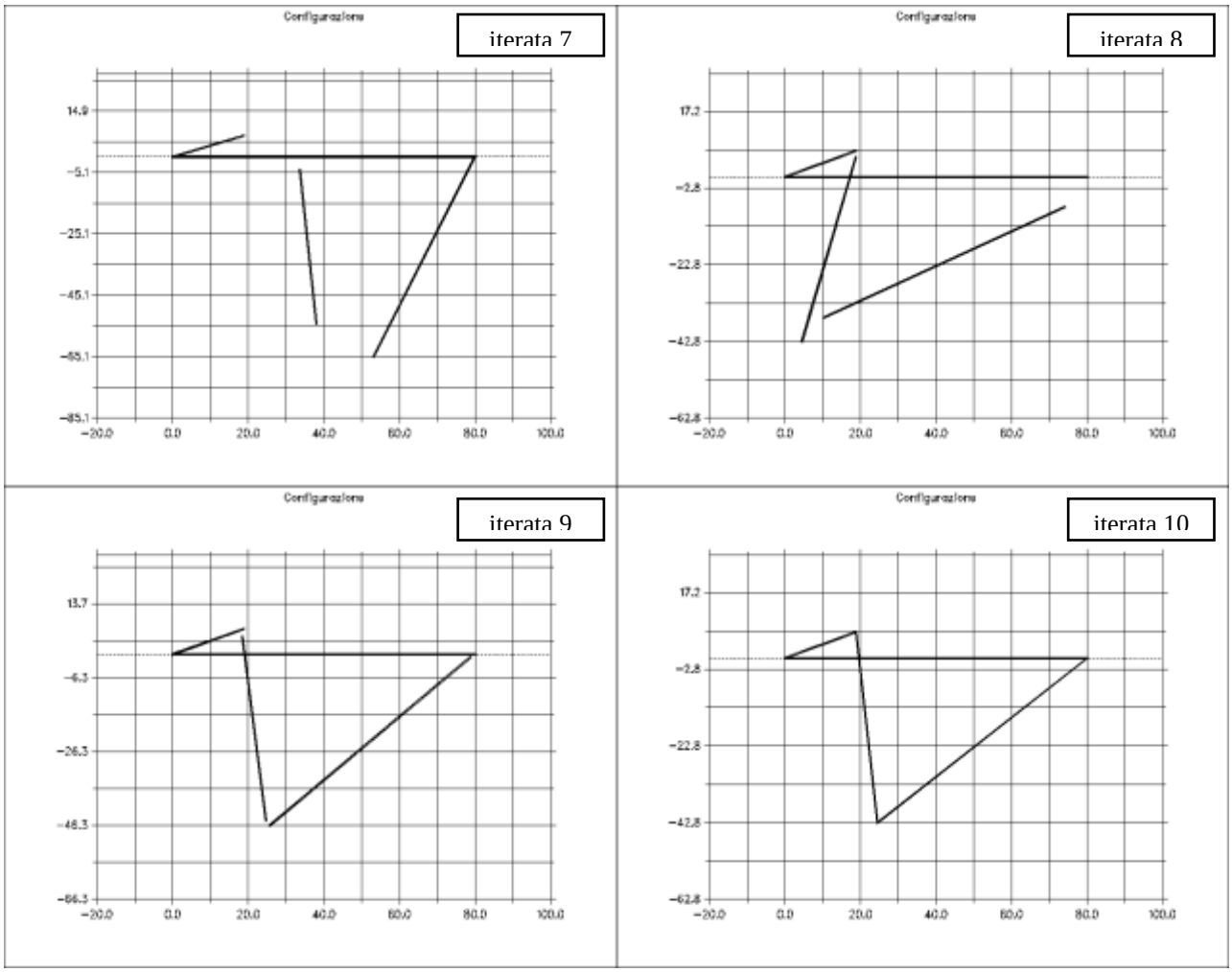
Si osserva altresì che ciascuno degli array utilizzati presenta un indice aggiuntivo che assume il valore del numero di iterata; se si considera ad esempio la matrice jacobiana, questo permette di registrare nel relativo array tutti i valori assunti da essa nel corso delle iterate. L'unico array per cui tale accorgimento abbia un impiego in questo programma è $norma$, che anziché avere un solo valore, è un array monodimensionale, e ciò consente di disegnarne il grafico. Per gli altri non c'è un impiego pratico, se non quello di prevedere una stampa delle configurazioni solo alla fine del ciclo indefinito (come effettivamente pensavo inizialmente di fare); oppure quello di avere in memoria i valori assunti nel corso delle iterate dalle varie funzioni.

5. Prima esecuzione. Sappiamo dal metodo grafico, illustrato in seguito, che le configurazioni possibili per il parallelogramma sono due. Quale delle due venga fornita dal programma dipenderà dalla configurazione assegnata iniziale. Si assegni la configurazione iniziale

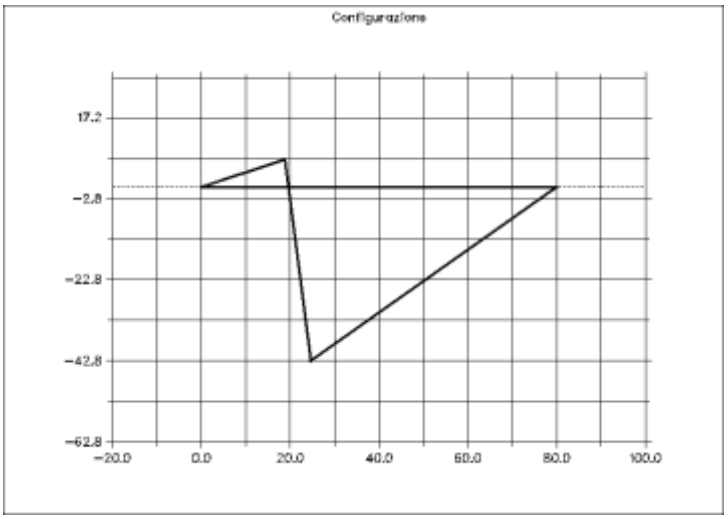
5.1) $x_2 = 40 \quad y_2 = 20 \quad \theta_3 = -40° \quad x_3 = 40 \quad y_3 = -40 \quad \theta_4 = 110° \quad x_4 = 70 \quad y_4 = -70$

con le lunghezze espresse in centimetri. Il programma effettua 11 iterazioni (compresa la prima configurazione) per soddisfare la condizione $norma \leq 0{,}01$.





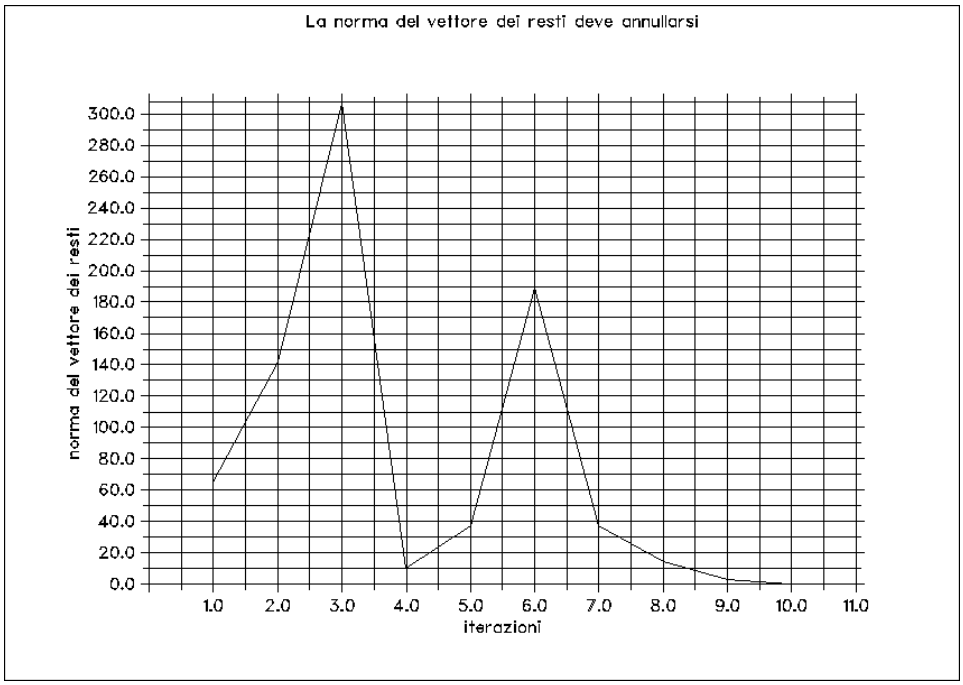
L'ultima iterata perviene alla configurazione seguente:



alla quale corrispondono le coordinate (in centimetri e gradi)

x_2	y_2	θ_3	x_3	y_3	θ_4	x_4	y_4
9.396926	3.4202013	-83.30455	21.708654	-17.989153	397.71222	52.31173	-21.409353

Per la norma del vettore dei resti si ottiene il seguente andamento.



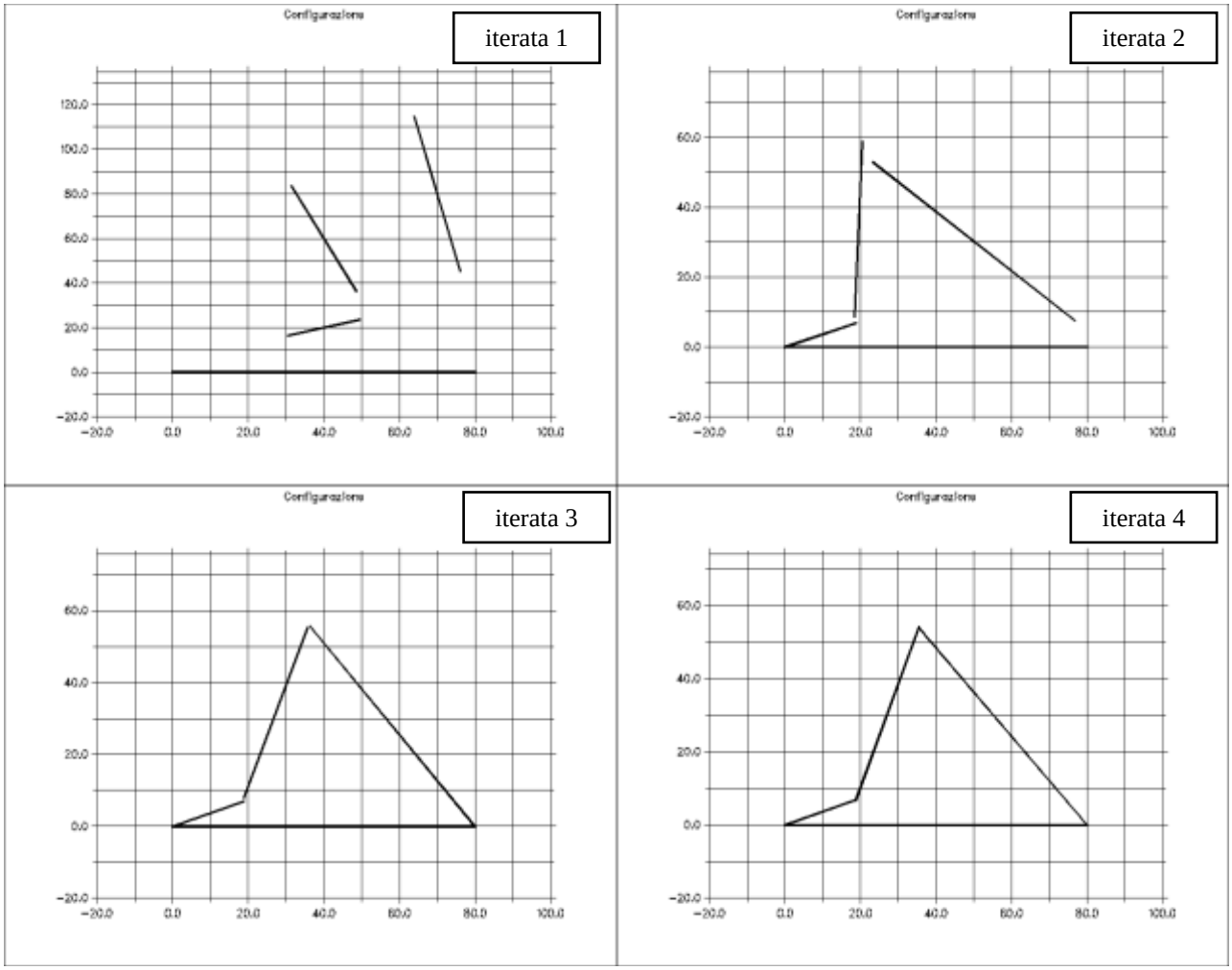
Il programma inoltre stampa su prompt le matrici e il vettore dei resti per alcune iterata. Riporto qualcuno di questi dati.

i	vettore dei resti	norma
1	30.603 16.580 2.162 36.804 10.642 1.769 36.201 -15.938	65.20431
2	0.000 0.000 -6.030 -33.391 94.286 -4.135 -33.391 94.286	141.64392
3	-0.000 -0.000 2.470 44.500-212.804 -2.252 44.500-212.804	307.4781
4	-0.000 -0.000 -2.685 -2.636 -6.432 2.354 -2.636 -6.432	10.459149
5	-0.000 -0.000 3.619 21.944 -14.332 -3.118 21.944 -14.332	37.372982
6	-0.000 -0.000 1.250 -63.084 117.990 -0.162 -63.084 117.990	189.21909
7	-0.000 -0.000 0.376 24.251 -10.468 0.756 24.251 -10.468	37.36419
8	-0.000 -0.000 -0.414 -9.995 1.192 -0.285 -9.995 1.192	14.244634
9	-0.000 -0.000 0.011 -0.049 -2.077 0.052 -0.049 -2.077	2.9379098
10	-0.000 -0.000 -0.002 -0.053 -0.008 -0.001 -0.053 -0.008	0.07611848
11	-0.000 -0.000 0.000 0.000 -0.000 0.000 0.000 -0.000	0.00008695498

6. Seconda esecuzione. Cercando di ottenere la seconda configurazione congruente possibile, faccio girare il programma con le seguenti coordinate di innesco:

6.1) $x_2 = 40 \quad y_2 = 20 \quad \theta_3 = 110° \quad x_3 = 40 \quad y_3 = 60 \quad \theta_4 = -80° \quad x_4 = 70 \quad y_4 = 80$

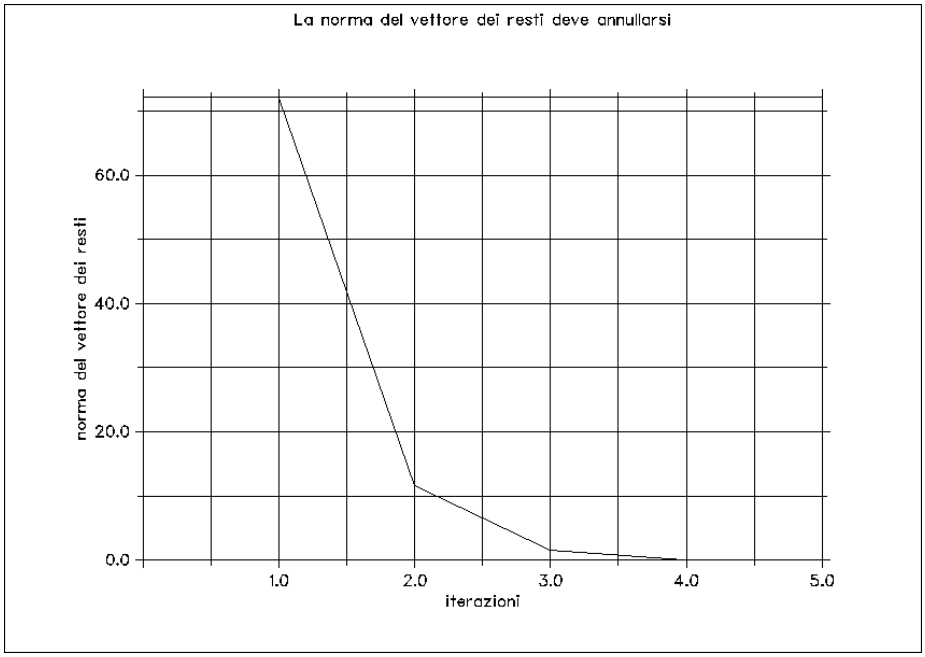
In questo caso si ha convergenza in 5 iterate: le configurazioni sono le seguenti.



La configurazione finale è

x_2	y_2	θ_3	x_3	y_3	θ_4	x_4	y_4
9.396926	3.4202013	70.55076	27.118143	30.413828	-50.46595	57.72122	26.993628

Il diagramma della norma del vettore dei resti, che segue, testimonia la rapida convergenza.



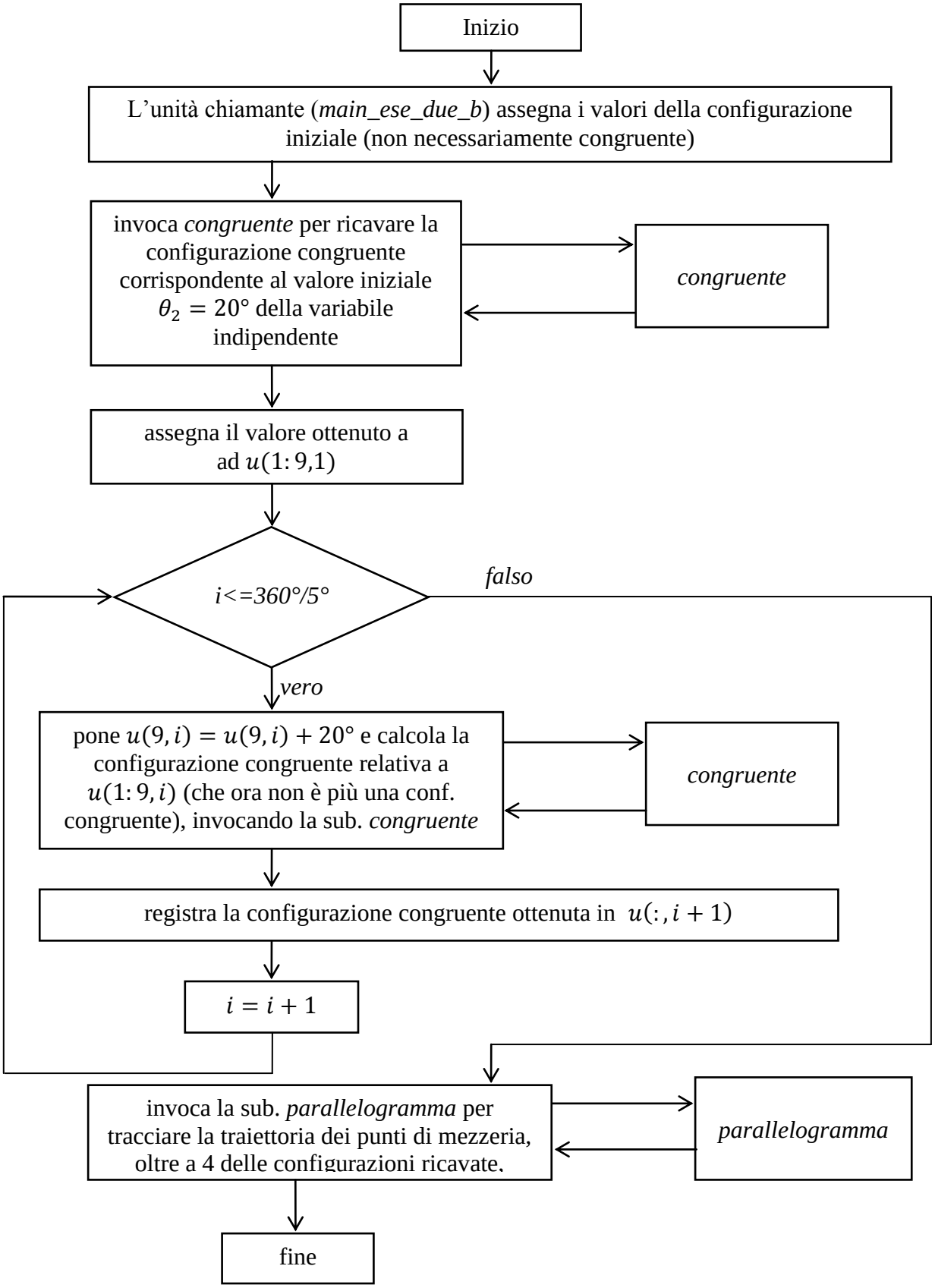
7. Secondo quesito. Il seguente algoritmo consente di ottenere la configurazione congruente del parallelogramma in funzione di θ_2, ovvero esplicita la funzione **2.3** rispetto alle variabili indipendenti, in modo numerico; quindi permette in particolare di esprimere le coordinate del punto $M3 \equiv (x_3, y_3)$ in funzione di θ_2, ciò che risponde al secondo questo proposto.
L'algoritmo utilizza i seguenti codici in Fortran:

- l'unità chiamante *main_ese_due_b* la quale si occupa principalmente di utilizzare le subroutine del modulo;
- il modulo *mod_ese_due_b* il quale contiene le seguenti subroutine:
  - *inverse* si occupa di invertire la matrice jacobiana;
  - *parallelogramma* traccia le traiettorie dei punti di mezzeria delle aste al variare di θ_2;
  - *congruente* calcola la configurazione congruente relativa ad ogni valore di θ_2, usando il metodo di Newton-Raphson e partendo ogni volta dalla configurazione congruente precedentemente calcolata, in cui la manovella viene sconnessa per aumentare l'angolo θ_2.

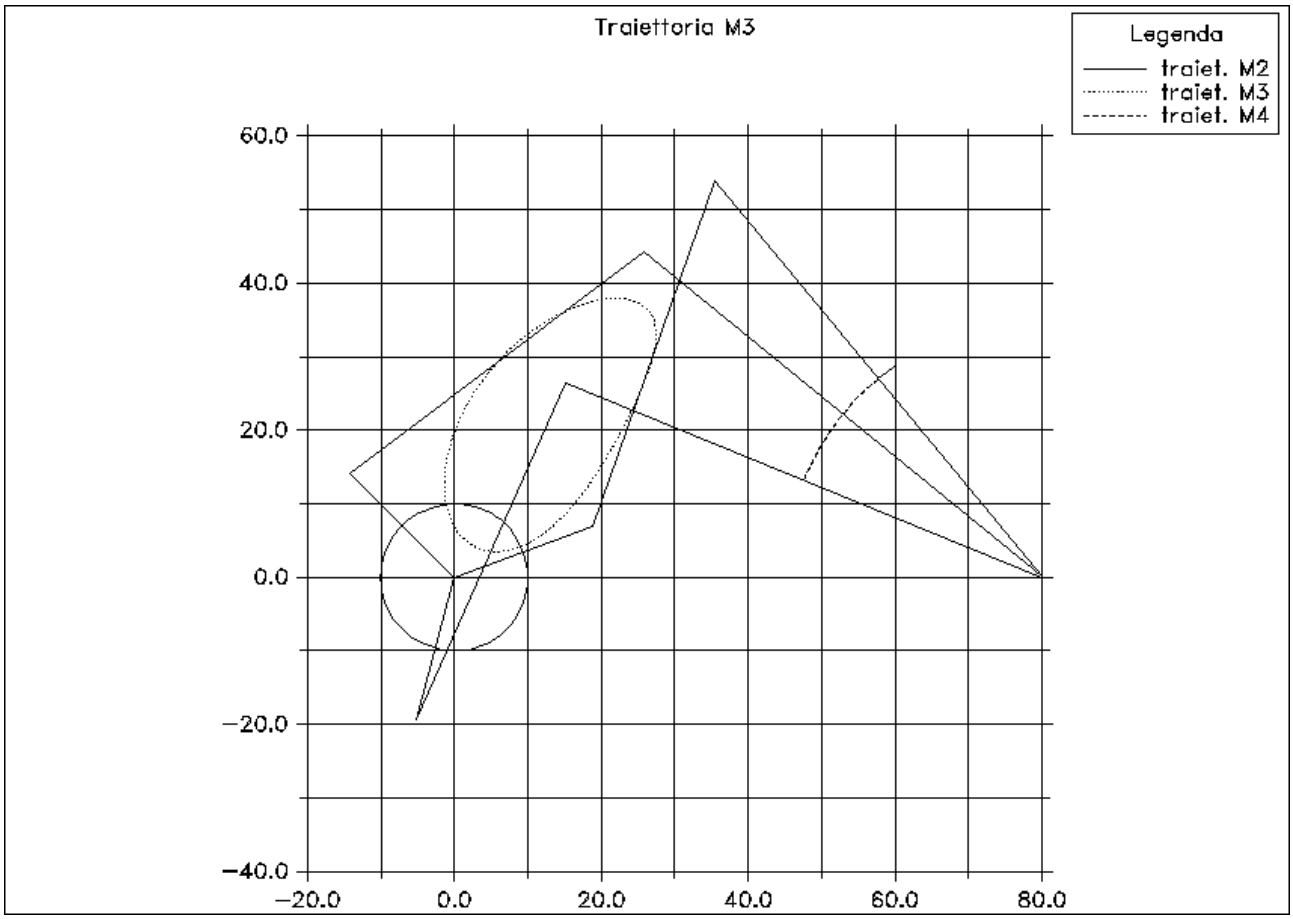
Evidentemente la subroutine *congruente* usa le stesse linee di codice del programma *main_ese_due*.
Si può notare tra l'altro che abbiamo una subroutine di modulo (*congruente*) che invoca un'altra subroutine (*inverse*), contenuta nello stesso modulo.
Per innescare il programma utilizzo una configurazione simile a quella congruente ottenuta alla fine della seconda esecuzione; uso cioè la configurazione

6.1) $x_2 = 9 \quad y_2 = 3 \quad \theta_3 = 70° \quad x_3 = 27 \quad y_3 = 30 \quad \theta_4 = -50° \quad x_4 = 57 \quad y_4 = 26 \qquad \theta_2 = 20°$

Alcune delle configurazioni congruenti che vengono trovate a partire da questa aumentando ogni volta di 5° l'angolo di manovella θ_2 -fino a completare un giro- sono riportate in tabella.


Inizio
L'unità chiamante (*main_ese_due_b*) assegna i valori della configurazione iniziale (non necessariamente congruente)
invoca *congruente* per ricavare la configurazione congruente corrispondente al valore iniziale $\theta_2 = 20°$ della variabile indipendente
congruente
assegna il valore ottenuto a ad $u(1:9,1)$
i<=360°/5°
falso
vero
pone $u(9,i) = u(9,i) + 20°$ e calcola la configurazione congruente relativa a $u(1:9,i)$ (che ora non è più una conf. congruente), invocando la sub. *congruente*
congruente
registra la configurazione congruente ottenuta in $u(:,i+1)$
$i = i + 1$
invoca la sub. *parallelogramma* per tracciare la traiettoria dei punti di mezzeria, oltre a 4 delle configurazioni ricavate.
parallelogramma
fine

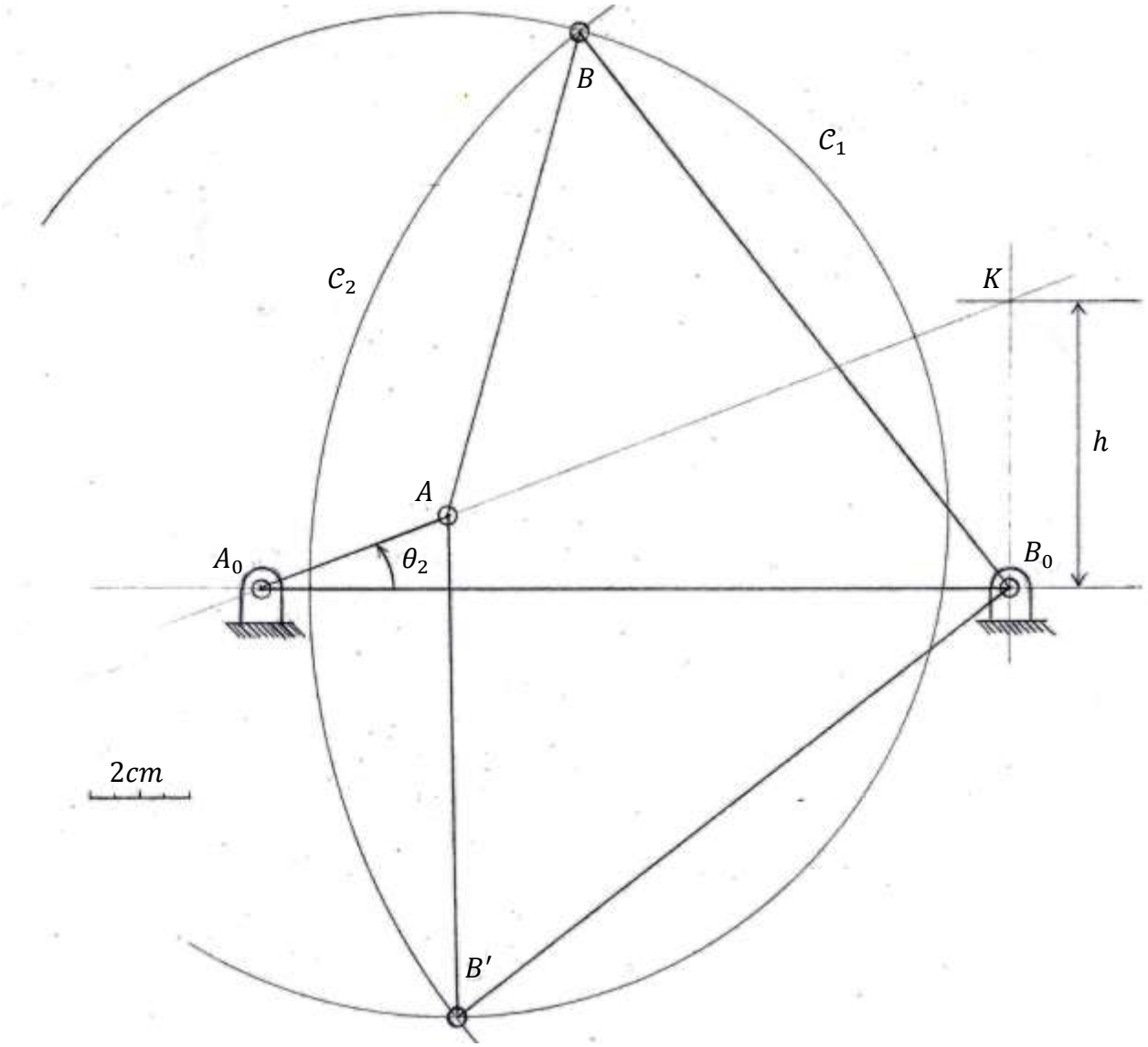
x_2	y_2	θ_3	x_3	y_3	θ_4	x_4	y_4	θ_2
9.396926	3.4202013	70.552284	27.117907	30.415138	-50.46702	57.72098	26.994938	20.
-7.660443	6.427878	36.89523	4.6724825	27.864597	-37.768875	52.332924	21.43672	140.
-1.736486	-9.848077	67.979324	5.900556	3.480062	-22.383488	47.637043	13.328138	260.
9.396929	3.4201934	70.55079	27.118134	30.413813	-50.465927	57.721207	26.99362	380.



Naturalmente la prima è l'ultima configurazione coincidono, e si può vedere quanto siano simili per esse i valori restituiti dal programma. Riporto allora il diagramma della traiettoria dei punti di mezzeria, calcolati in base a $360°:5° = 72$ iterazioni.; sono tracciate anche le quattro configurazioni indicate in tabella.

8. Soluzione grafica del primo quesito. Fisso una scala di rappresentazione $\sigma_L = 0{,}05 m/cm$ ottenendo per i quattro lati del parallelogramma le misure

8.1) $$\begin{cases} \overline{A_0A} = \frac{l_2}{\sigma_L} = \frac{0.2}{0.05} cm = 4cm \\ \overline{AB} = \frac{l_3}{\sigma_L} = \frac{0.7}{0.05} cm = 14cm \\ \overline{BB_0} = \frac{l_4}{\sigma_L} = \frac{0.5}{0.05} cm = 10cm \\ \overline{B_0A_0} = \frac{l_1}{\sigma_L} = \frac{0.8}{0.05} cm = 16cm \end{cases}$$



Si segue quindi questa costruzione:

- si traccia a partire da A_0 un segmento orizzontale di lunghezza $16cm$, individuando così il punto B_0;
- si traccia da B_0 un segmento verticale di altezza $h = \overline{A_0B_0}\tan\theta_2 = 5.82cm$ individuando il punto K;
- la manovella si troverà sulla retta per A_0, K e resta dunque individuato il punto A, posto su tale retta, a distanza $4cm$ da A_0;
- traccio la circonferenza $\mathcal{C}_1$ con centro in A e raggio $\overline{AB} = 14cm$;
- traccio la circonferenza $\mathcal{C}_2$ con centro in B_0 e raggio $\overline{BB_0} = 10cm$.

Restano così individuati i due punti B, B' e dunque le due configurazioni congruenti A_0, A, B, B_0 e A_0, A, B', B_0.

9. Codice dell'unità chiamante per il primo quesito.

```
!programma principale per la seconda esercitazione di meccanica applicata
!usa il modulo mod_ese_uno
!11/10/2012
```

```
PROGRAM main_ese_due

USE DISLIN      !libreria grafica
USE mod_ese_due

!sezione dichiarativa

IMPLICIT NONE

!dichiaro il vettore delle coordinate dipendenti

REAL,DIMENSION(8,100)::u

!dichiaro la funzione delle equazioni di vincolo

REAL,DIMENSION(8,100)::psi

!dichiaro la matrice jacobiana e la sua inversa

REAL,DIMENSION (8,8,100)::J
REAL(KIND=8),DIMENSION (8,8)::invj

!dichiaro il vettore dei resti

REAL,DIMENSION(8,100)::resto

!dichiaro la norma del vettore dei resti

REAL,DIMENSION(100):: norma

!dichiaro i valori di innesco delle coordinate

REAL::x2 =40
REAL::y2 =20
REAL::theta3 =-40*(3.1415927/180)
REAL::x3 =40
REAL::y3 =-40
REAL::theta4=110*(3.1415927/180)
REAL::x4=70
REAL::y4=-70

!altre variabili di lavoro

REAL(KIND=8),DIMENSION(8,8)::matrix
INTEGER::i !indici dei cicli
CHARACTER(len=10):: chiusura !serve per chiudere il programma

!sezione esecutiva

CALL BMPMOD (300,'inch','resolution') !fisso risoluzione dell'immagine in formato .bmp

!inizializzo i valori d'innesco di u

u(1,1) = x2
u(2,1) = y2
u(3,1) = theta3
u(4,1) = x3
```

```
u(5,1) = y3
u(6,1) = theta4
u(7,1) = x4
u(8,1) = y4

!inizializzo la matrice jacobiana con i valori costanti

j(:,:,:)=0

j(1,1,:) = 1
j(2,2,:) = 1
j(3,1,:) = -1
j(3,4,:) = 1
j(4,2,:) = -1
j(4,5,:) = 1
j(5,4,:) = -1
j(5,7,:) = 1
j(6,5,:) = -1
j(6,8,:) = 1
j(7,7,:) = 1
j(8,8,:) = 1

!inizializzo il vettore resto

resto=0

!ciclo del metodo Newton-Raphson

i=0

ciclo: DO

                                        i=i+1

                                        x2     = u(1,i)
                                        y2     = u(2,i)
                                        theta3 = u(3,i)
                                        x3     = u(4,i)
                                        y3     = u(5,i)
                                        theta4 = u(6,i)
                                        x4     = u(7,i)
                                        y4     = u(8,i)


                                        !calcolo la matrice jacobiana

                                        j(3,3,i) = (0.5)*l3*SIN(theta3)
                                        j(4,3,i) = -(0.5)*l3*COS(theta3)
                                        j(5,3,i) = (0.5)*l3*SIN(theta3)
                                        j(5,6,i) = (0.5)*l4*SIN(theta4)
                                        j(6,3,i) = -(0.5)*l3*cos(theta3)
                                        j(6,6,i) = -(0.5)*l4*cos(theta4)
                                        j(7,6,i) = -(0.5)*l4*sin(theta4)
                                        j(8,6,i) = (0.5)*l4*cos(theta4)

                                        !calcolo l'inversa della matrice jacobiana
```

```
matrix(:,:)=j(:,:,i)

CALL inverse(matrix,invj,8)

!calcolo la funzione psi

psi(1,i) = x2 - ((0.5)*l2*COS(theta2))
psi(2,i) = y2 - (0.5)*l2*SIN(theta2)
psi(3,i) = x3 -(l3*0.5)*COS(theta3) - x2 - (l2*0.5)*COS(theta2)
psi(4,i) = y3 -(l3/2.)*SIN(theta3) - y2 - (l2*0.5)*SIN(theta2)
psi(5,i) = x4 -x3 -(0.5)*l3*COS(theta3) -(0.5)*l4*COS(theta4)
psi(6,i) = y4 -y3 -(0.5)*l3*SIN(theta3) -(0.5)*l4*SIN(theta4)
psi(7,i) = x4 - l1 + (0.5)*l4*COS(theta4)
psi(8,i) = y4 + (0.5)*l4*sin(theta4)

!calcola il vettore resto

resto(:,i)=MATMUL(invj,psi(:,i))

!calcola la norma del resto

norma(i)=SQRT(DOT_PRODUCT(resto(:,i),resto(:,i)))

100 FORMAT(8F8.3)

WRITE(*,*) " "
WRITE(*,*) "ITERATA",i
WRITE(*,*) " "
WRITE(*,*) "Le coordinate sono"
WRITE(*,*) "theta2=", theta2*180/3.1415927
WRITE(*,*) "x2=",     u(1,i)
WRITE(*,*) "y2=",     u(2,i)
WRITE(*,*) "theta3=", u(3,i)*180/3.1415927
WRITE(*,*) "x3=",     u(4,i)
WRITE(*,*) "y3=",     u(5,i)
WRITE(*,*) "theta4=", u(6,i)*180/3.1415927
WRITE(*,*) "x4=",     u(7,i)
WRITE(*,*) "y4=",     u(8,i)
WRITE(*,*) " "
WRITE(*,*) "La matrice jacobiana vale"
WRITE(*,100) j(1,:,i)
WRITE(*,100) j(2,:,i)
WRITE(*,100) j(3,:,i)
WRITE(*,100) j(4,:,i)
WRITE(*,100) j(5,:,i)
WRITE(*,100) j(6,:,i)
WRITE(*,100) j(7,:,i)
WRITE(*,100) j(8,:,i)
WRITE(*,*) " "
WRITE(*,*) "La sua inversa vale"
WRITE(*,100) invj(1,:)
WRITE(*,100) invj(2,:)
WRITE(*,100) invj(3,:)
WRITE(*,100) invj(4,:)
WRITE(*,100) invj(5,:)
WRITE(*,100) invj(6,:)
WRITE(*,100) invj(7,:)
```

```
                                WRITE(*,100) invj(8,:)
    WRITE(*,*) " "
                                WRITE(*,*) "Il vettore dei resti vale"
                                WRITE(*,100) resto(:,i)
                                WRITE(*,*) " "
                                WRITE(*,*) "La sua norma vale", norma(i)

                                !disegno il parallelogramma alla iterata i

                                CALL parallelogramma(x2,y2,x3,y3,theta3,x4,y4,theta4,i)

                                !verifico il valore della norma e in caso esco

                                IF (norma(i)<=0.01) EXIT ciclo

                                !calcola le coordinate della iterata successiva

                                u(1:8,i+1)=u(1:8,i)-resto(1:8,i)

END DO ciclo

iter_i = i
iter_r = REAL(iter_i)

!Traccio il diagramma del resto

CALL diagramma_resto (norma)

WRITE (*,*)"Per chiudere il programma premi una lettera qualunque."
WRITE (*,*)"Tutti i dati andranno persi."
READ  (*,*) chiusura

STOP

END PROGRAM main_ese_due
```

10. Codice del modulo per il primo quesito.

```
!modulo per la seconda esercitazione di meccanica apl.
!invocato dal programma main_ese_uno
!12/10/2011

MODULE mod_ese_due

!sezione dichiarativa

IMPLICIT NONE

!dichiaro le costanti geometriche del quadrilatero

REAL(KIND=8), PARAMETER:: l2=20.
REAL(KIND=8), PARAMETER:: l3=50.
REAL(KIND=8), PARAMETER:: l4=70.
REAL(KIND=8), PARAMETER:: l1=80.

REAL(KIND=8), PARAMETER:: theta2=20*(3.1415927/180)
```

```
!dichiaro il numero di iterazioni

INTEGER:: iter_i !intero
REAL::    iter_r !relae

!fisso il formato di output per i diagrammi

CHARACTER(len=4):: formato ='bmp '

!scrivo le subroutine

CONTAINS

!---------------------------------------------------------------

 subroutine inverse(a,c,n)
!============================================================
! Inverse matrix
! Method: Based on Doolittle LU factorization for Ax=b
! Alex G. December 2009
!-----------------------------------------------------------
! input ...
! a(n,n) - array of coefficients for matrix A
! n      - dimension
! output ...
! c(n,n) - inverse matrix of A
! comments ...
! the original matrix a(n,n) will be destroyed
! during the calculation
!============================================================
implicit none
integer n
double precision a(n,n), c(n,n)
double precision L(n,n), U(n,n), b(n), d(n), x(n)
double precision coeff
integer i, j, k

! step 0: initialization for matrices L and U and b
! Fortran 90/95 aloows such operations on matrices
L=0.0
U=0.0
b=0.0

! step 1: forward elimination
do k=1, n-1
  do i=k+1,n
    coeff=a(i,k)/a(k,k)
    L(i,k) = coeff
    do j=k+1,n
      a(i,j) = a(i,j)-coeff*a(k,j)
    end do
  end do
end do

! Step 2: prepare L and U matrices
! L matrix is a matrix of the elimination coefficient
! + the diagonal elements are 1.0
```

```
do i=1,n
  L(i,i) = 1.0
end do
! U matrix is the upper triangular part of A
do j=1,n
  do i=1,j
    U(i,j) = a(i,j)
  end do
end do

! Step 3: compute columns of the inverse matrix C
do k=1,n
  b(k)=1.0
  d(1) = b(1)
! Step 3a: Solve Ld=b using the forward substitution
  do i=2,n
    d(i)=b(i)
    do j=1,i-1
      d(i) = d(i) - L(i,j)*d(j)
    end do
  end do
! Step 3b: Solve Ux=d using the back substitution
  x(n)=d(n)/U(n,n)
  do i = n-1,1,-1
    x(i) = d(i)
    do j=n,i+1,-1
      x(i)=x(i)-U(i,j)*x(j)
    end do
    x(i) = x(i)/u(i,i)
  end do
! Step 3c: fill the solutions x(n) into column k of C
  do i=1,n
    c(i,k) = x(i)
  end do
  b(k)=0.0
end do
end subroutine inverse

!---------------------------------------------------------------------

SUBROUTINE parallelogramma (x2,y2,x3,y3,theta3,x4,y4,theta4,iterata)

!sezione dichiarativa

!dichiaro gli argomenti fittizi

!dichiaro le coordinate

REAL,INTENT(IN)::x2,y2,x3,y3,theta3,x4,y4,theta4

!dichiaro il numero di iterata

INTEGER,INTENT(IN)::iterata

!dichiaro le variabili locali

!dichiaro e assegno gli estremi dei lati l2
```

```
REAL::l2A0x
REAL::l2A0y
REAL::l2Ax
REAL::l2Ay

!dichiaro e assegno gli estremi del lato l3

REAL::l3Ax
REAL::l3Ay
REAL::l3Bx
REAL::l3By

!dichiaro e assegno gli estremi del lato l4

REAL::l4Bx
REAL::l4By
REAL::l4B0x
REAL::l4B0y

!dichiaro e assegno gli estremi del lato l1

REAL::l1B0x
REAL::l1B0y
REAL::l1A0x
REAL::l1A0y

!dichiaro il minimo e ilmassimo delle ascisse e delle ordinate

REAL:: xmin
REAL:: xmax
REAL:: ymin
REAL:: ymax

!sezione esecutiva

!assegno gli estremi dei lati l2

l2A0x = x2 - (0.5*l2*COS(theta2))
l2A0y = y2 - (0.5*l2*SIN(theta2))
l2Ax  = x2 + (0.5*l2*COS(theta2))
l2Ay  = y2 + (0.5*l2*SIN(theta2))

!assegno gli estremi del lato l3

l3Ax = x3 - 0.5*l3*COS(theta3)
l3Ay = y3 - 0.5*l3*SIN(theta3)
l3Bx = x3 + 0.5*l3*COS(theta3)
l3By = y3 + 0.5*l3*SIN(theta3)

!assegno gli estremi del lato l4

l4Bx  = x4 - 0.5*l4*COS(theta4)
l4By  = y4 - 0.5*l4*SIN(theta4)
l4B0x = x4 + 0.5*l4*COS(theta4)
l4B0y = y4 + 0.5*l4*SIN(theta4)
```

```
!assegno gli estremi del lato l1

l1B0x  = 80.
l1B0y  = 0
l1A0x  = 0
l1A0y  = 0

!calcolo l'asscissa minima e quella massima

xmin = MIN(l2A0x,l2Ax, l3Ax, l3Bx, l4Bx, l4B0x, l1B0x, l1A0x)
xmax = MAX(l2A0x,l2Ax, l3Ax, l3Bx, l4Bx, l4B0x, l1B0x, l1A0x)

!calcolo l'ascissa minima e quella massima

ymin = MIN(l2A0y,l2Ay, l3Ay, l3By, l4By, l4B0y, l1B0y, l1A0y)
ymax = MAX(l2A0y,l2Ay, l3Ay, l3By, l4By, l4B0y, l1B0y, l1A0y)

!alcune istruzioni grafiche

CALL METAFL (formato)  !indico il formato dell'output
CALL SCRMOD ('revers') !scritta nera su fondo bianco

CALL DISINI                              !richaima alcune impostazioni di default

CALL PAGERA          !traccio un bordo per il piano xy
CALL DUPLX          !font a doppio spessore

CALL AXSPOS (450,1800) !coordinate angolo basso sinistra
CALL AXSLEN (2200,1500)!lunghezza dei due assi in pixel

CALL NAME (' ','x')    !nome delle ascisse
CALL NAME (' ','y')    !nome delle ordinate

CALL TITLIN ("Configurazione",1) !prima riga del titolo

CALL GRAF (xmin-20.,xmax+20.,xmin-20,20.,ymin-20.,ymax+20.,ymin-20.,20.)

CALL GRID (2,2)       !impone una griglia sul piano coordinato

CALL TITLE !stampa il titolo di cui sopra

CALL DASH   !tratteggio per gli assi coordinati
CALL XAXGIT !traccio la retta y=0
CALL YAxGIT !traccio la retta x=0

CALL MYLINE (1,1)  !impone una linea continua
CALL LINWID (8)                          !spessore della linea

!disegno ciascuno dei lati indicando gli estremi

CALL RLINE (l2A0x, l2A0y, l2Ax, l2Ay)
CALL RLINE (l3Ax, l3Ay, l3Bx, l3By)
CALL RLINE (l4Bx, l4By, l4B0x, l4B0y)
CALL RLINE (l1B0x, l1B0y, l1A0x, l1A0y)

CALL DISFIN
```

```
END SUBROUTINE parallelogramma

!-----------------------------------------------------------------

SUBROUTINE diagramma_resto (norma)

!sezione dichiarativa

!dichiaro gli argomenti fittizi

REAL,INTENT(IN),DIMENSION(iter_i):: norma

!dichiaro le variabili locali

INTEGER::i !indice del ciclo
REAL,DIMENSION(iter_i)::x                    !qui metto le ascisse
REAL:: max                  !il massimo della norma
REAL:: min                                          !il minimo della norma

!sezione esecutiva

!inizializzo l'array delle ascisse

x(1)=1.

ciclo_ascisse: DO i=2,iter_i,1

            x(i)=x(i-1)+1.

END DO ciclo_ascisse

WRITE(*,*) x

CALL METAFL (formato)  !indico il formato dell'output
CALL SCRMOD ('revers') !scritta nera su fondo bianco

CALL DISINI

CALL PAGERA          !traccio un bordo per il piano xy
CALL DUPLX           !font a doppio spessore

CALL AXSPOS (450,1800) !coordinate angolo basso sinistra
CALL AXSLEN (2200,1500)!lunghezza dei due assi in pixel

CALL NAME ('iterazioni','x')                  !nome delle ascisse
CALL NAME ('norma del vettore dei resti','y')    !nome delle ordinate

CALL TITLIN ("La norma del vettore dei resti deve annullarsi",1) !prima riga del titolo

max=MAXVAL(norma) !il massimo della funzione
min=MINVAL(norma) !il minimo valore della funzione

WRITE(*,*) "il minimo della norma del vettoredei resti vale", min
WRITE(*,*)"il massimo della norma del vettore dei resti vale",max

CALL GRAF (0.,x(iter_i),1.,1.0,min,max,0.,20.)
```

```
CALL GRID (2,2)       !impone una griglia sul piano coordinato

CALL TITLE !stampa il titolo di cui sopra

CALL CURVE (x,norma,iter_i) !plotto il diagramma del resto

CALL DASH   !tratteggio per gli assi coordinati
CALL XAXGIT !traccio la retta y=0
CALL YAxGIT !traccio la retta x=0

CALL DISFIN

END SUBROUTINE diagramma_resto

!---------------------------------------------------------------------

END MODULE mod_ese_due
```

11. Codice dell'unità chiamante per il secondo quesito.

```
!programma principale per la seconda esercitazione di meccanica applicata
!usa il modulo mod_ese_due_b
!11/10/2012

PROGRAM main_ese_due_b

USE DISLIN      !libreria grafica

USE mod_ese_due_b

!sezione dichiarativa

IMPLICIT NONE

!dichiaro il vettore delle coordinate dipendenti

REAL,DIMENSION(9,100)::u

!dichiaro l'array che contiene la conf congruente

REAL,DIMENSION(9)::conf

!dichiaro i valori di innesco delle coordinate

REAL::x2    =  9
REAL::y2    =  3
REAL::theta3 =  70*(3.1415927/180)
REAL::x3    =  27
REAL::y3    =  30
REAL::theta4 = -50*(3.1415927/180)
REAL::x4    =  57
REAL::y4    =  26
REAL::theta2 =  20*(3.1415927/180)

!dichiaro l'incremento angolare di theta2

REAL::delta_rad
```

```
INTEGER::delta_gra = 5

!dichiaro il numero di incrementi angolari di theta2

INTEGER::n

!dichiaro l'indice del ciclo

INTEGER::i

!dichiaro una stringa che serve per chiudere il programma

CHARACTER(len=10):: chiusura

!sezione esecutiva

!calcolo delta_rad

delta_rad = delta_gra*(3.1415927/180)

!calcolo la configurazione congruente relativa ai valori di innesco

CALL congruente (x2,y2,theta3,x3,y3,theta4,x4,y4,theta2,conf)

WRITE(*,*) " "
                                    WRITE(*,*) "La configurazione per theta2=", theta2*180/3.1415927,"e'"
                                    WRITE(*,*) " "
                                    WRITE(*,*) "x2=",    conf(1)
                                    WRITE(*,*) "y2=",    conf(2)
                                    WRITE(*,*) "theta3=", conf(3)*180/3.1415927
                                    WRITE(*,*) "x3=",    conf(4)
                                    WRITE(*,*) "y3=",    conf(5)
                                    WRITE(*,*) "theta4=", conf(6)*180/3.1415927
                                    WRITE(*,*) "x4=",    conf(7)
                                    WRITE(*,*) "y4=",    conf(8)
                                    WRITE(*,*) "theta2=", conf(9)*180/3.1415927
                                    WRITE(*,*) " "

!assegno la prima configurazione congruente

u(:,1) = conf

!calcolo n

n=1+INT(360/delta_gra)

!ciclo per la determinazione delle configurazioni

ciclo: DO i=2,n,1

                                    x2    = u(1,i-1)
                                    y2    = u(2,i-1)
                                    theta3 = u(3,i-1)
                                    x3    = u(4,i-1)
                                    y3    = u(5,i-1)
                                    theta4 = u(6,i-1)
                                    x4    = u(7,i-1)
```

```
                                    y4    = u(8,i-1)
                                    theta2 = u(9,i-1)+delta_rad

                                    !calcolo la configurazirazion congruente con theta2

                                    CALL congruente (x2,y2,theta3,x3,y3,theta4,x4,y4,theta2,conf)

                                    !assegno la configurazione congruente a u

                                    u(:,i)=conf

                                    WRITE(*,*) " "
                                    WRITE(*,*) "La configurazione per theta2=", theta2*180/3.1415927,"e'"
                                    WRITE(*,*) " "
                                    WRITE(*,*) "x2=",    conf(1)
                                    WRITE(*,*) "y2=",    conf(2)
                                    WRITE(*,*) "theta3=", conf(3)*180/3.1415927
                                    WRITE(*,*) "x3=",    conf(4)
                                    WRITE(*,*) "y3=",    conf(5)
                                    WRITE(*,*) "theta4=", conf(6)*180/3.1415927
                                    WRITE(*,*) "x4=",    conf(7)
                                    WRITE(*,*) "y4=",    conf(8)
                                    WRITE(*,*) "theta2=", conf(9)*180/3.1415927
                                    WRITE(*,*) " "

END DO ciclo

!traccio il diagramma con la traiettoria di M4 e con
!alcune delle 180/delta_gra configurazioni

CALL parallelogramma (u,n)

WRITE (*,*)"Per chiudere il programma premi una lettera qualunque."
WRITE (*,*)"Tutti i dati andranno persi."
READ  (*,*) chiusura

STOP

END PROGRAM main_ese_due_b
```

12. Codice del modulo per il secondo quesito.

```
!modulo per la seconda esercitazione di meccanica apl.
!invocato dal programma main_ese_uno
!12/10/2011

MODULE mod_ese_due_b

!sezione dichiarativa

IMPLICIT NONE

!dichiaro le costanti geometriche del quadrilatero

REAL, PARAMETER:: l2=20.
REAL, PARAMETER:: l3=50.
REAL, PARAMETER:: l4=70.
```

```
REAL, PARAMETER:: l1=80.

!fisso il formato di output per i diagrammi

CHARACTER(len=4):: formato ='bmp '

!scrivo le subroutine

CONTAINS

!---------------------------------------------------------------

 subroutine inverse(a,c,n)
!============================================================
! Inverse matrix
! Method: Based on Doolittle LU factorization for Ax=b
! Alex G. December 2009
!-----------------------------------------------------------
! input ...
! a(n,n) - array of coefficients for matrix A
! n      - dimension
! output ...
! c(n,n) - inverse matrix of A
! comments ...
! the original matrix a(n,n) will be destroyed
! during the calculation
!============================================================
implicit none
integer n
double precision a(n,n), c(n,n)
double precision L(n,n), U(n,n), b(n), d(n), x(n)
double precision coeff
integer i, j, k

! step 0: initialization for matrices L and U and b
! Fortran 90/95 aloows such operations on matrices
L=0.0
U=0.0
b=0.0

! step 1: forward elimination
do k=1, n-1
  do i=k+1,n
    coeff=a(i,k)/a(k,k)
    L(i,k) = coeff
    do j=k+1,n
      a(i,j) = a(i,j)-coeff*a(k,j)
    end do
  end do
end do

! Step 2: prepare L and U matrices
! L matrix is a matrix of the elimination coefficient
! + the diagonal elements are 1.0
do i=1,n
  L(i,i) = 1.0
end do
```

```
! U matrix is the upper triangular part of A
do j=1,n
  do i=1,j
    U(i,j) = a(i,j)
  end do
end do

! Step 3: compute columns of the inverse matrix C
do k=1,n
  b(k)=1.0
  d(1) = b(1)
! Step 3a: Solve Ld=b using the forward substitution
  do i=2,n
    d(i)=b(i)
    do j=1,i-1
      d(i) = d(i) - L(i,j)*d(j)
    end do
  end do
! Step 3b: Solve Ux=d using the back substitution
  x(n)=d(n)/U(n,n)
  do i = n-1,1,-1
    x(i) = d(i)
    do j=n,i+1,-1
      x(i)=x(i)-U(i,j)*x(j)
    end do
    x(i) = x(i)/u(i,i)
  end do
! Step 3c: fill the solutions x(n) into column k of C
  do i=1,n
    c(i,k) = x(i)
  end do
  b(k)=0.0
end do
end subroutine inverse

!---------------------------------------------------------------------

SUBROUTINE parallelogramma (u,n)

!sezione dichiarativa

!dichiaro gli argomenti fittizi

!dichiaro l'array di tutte le configurazioni calcolate

REAL,INTENT(IN),DIMENSION(9,100)::u

!dichiaro il numero di configurazioni calcolate

INTEGER,INTENT(IN)::n

!dichiaro le variabili locali

!stringa usata per la legenda

CHARACTER(len=30)::stringa
```

```
!dichiaro l'indice del ciclo

INTEGER::i

!dichiaro le coordinate

REAL::x2,y2,theta3,x3,y3,theta4,x4,y4,theta2

!dichiaro gli estremi dei lati l2

REAL::l2A0x
REAL::l2A0y
REAL::l2Ax
REAL::l2Ay

!dichiaro gli estremi del lato l3

REAL::l3Ax
REAL::l3Ay
REAL::l3Bx
REAL::l3By

!dichiaro gli estremi del lato l4

REAL::l4Bx
REAL::l4By
REAL::l4B0x
REAL::l4B0y

!dichiaro gli estremi del lato l1

REAL::l1B0x
REAL::l1B0y
REAL::l1A0x
REAL::l1A0y

!dichiaro le coordinate di M2,M3,M4

REAL,DIMENSIOn(100):: xM2,xM3,xM4
REAL,DIMENSION(100):: yM2,yM3,yM4

!sezione esecutiva

CALL METAFL ('bmp')  !indico il formato dell'output
CALL SCRMOD ('revers') !scritta nera su fondo bianco

!fisso risoluzione dell'immagine in formato .bmp

CALL BMPMOD (25000,'meter','resolution')

CALL DISINI                              !richaima alcune impostazioni di default

CALL PAGERA          !traccio un bordo per il piano xy
CALL DUPLX          !font a doppio spessore

CALL AXSPOS (700,2000) !coordinate angolo basso sinistra
CALL AXSLEN (1700,1700)!lunghezza dei due assi in pixel
```

```
CALL NAME (' ','x')                              !nome delle ascisse
CALL NAME (' ','y')                              !nome delle ordinate
CALL TITLIN ("Traiettoria M3",1)   !prima riga del titolo

CALL GRAF (-20.,80.,-20.,20.,-40.,60.,-40.,20.)

CALL GRID (2,2)                    !impone una griglia sul piano coordinato

CALL TITLE                                 !stampa il titolo di cui sopra

CALL CHNCRV ('line')     !usa tratti diversi per le tre curve

CALL DASH                                   !tratteggio per gli assi coordinati

CALL XAXGIT                                !traccio la retta x=0
CALL YAxGIT                                !traccio la retta y=0

ciclo: DO i=1,n,1

                                   !assegno le coordinate

                                   x2     = u(1,i)
                                   y2     = u(2,i)
                                   theta3 = u(3,i)
                                   x3     = u(4,i)
                                   y3     = u(5,i)
                                   theta4 = u(6,i)
                                   x4     = u(7,i)
                                   y4     = u(8,i)
                                   theta2 = u(9,i)

                                   !assegno gli estremi dei lati l2

                                   l2A0x = x2 - (0.5*l2*COS(theta2))
                                   l2A0y = y2 - (0.5*l2*SIN(theta2))
                                   l2Ax  = x2 + (0.5*l2*COS(theta2))
                                   l2Ay  = y2 + (0.5*l2*SIN(theta2))

                                   !assegno gli estremi del lato l3

                                   l3Ax = x3 - 0.5*l3*COS(theta3)
                                   l3Ay = y3 - 0.5*l3*SIN(theta3)
                                   l3Bx = x3 + 0.5*l3*COS(theta3)
                                   l3By = y3 + 0.5*l3*SIN(theta3)

                                   !assegno gli estremi del lato l4

                                   l4Bx  = x4 - 0.5*l4*COS(theta4)
                                   l4By  = y4 - 0.5*l4*SIN(theta4)
                                   l4B0x = x4 + 0.5*l4*COS(theta4)
                                   l4B0y = y4 + 0.5*l4*SIN(theta4)

                                   !assegno gli estremi del lato l1

                                   l1B0x  = 80.
                                   l1B0y  = 0
```

```
l1A0x  = 0
l1A0y  = 0

!calcolo la posizione di M2, M3, M4

xM2(i) = x2
yM2(i) = y2
xM3(i) = x3
yM3(i) = y3
xM4(i) = x4
yM4(i) = y4

!alcune istruzioni grafiche

!traccio la traiettoria di M4

!traccio alcune configurazioni

CALL MYLINE (1,1)  !impone una linea continua
CALL LINWID (1)            !spessore della linea

condizione: IF (i==1) THEN

        !disegno ciascuno dei lati indicando gli estremi

        CALL RLINE (l2A0x, l2A0y, l2Ax, l2Ay)
        CALL RLINE (l3Ax, l3Ay, l3Bx, l3By)
        CALL RLINE (l4Bx, l4By, l4B0x, l4B0y)
        CALL RLINE (l1B0x, l1B0y, l1A0x, l1A0y)

ELSE IF (i==24) THEN condizione

        !disegno ciascuno dei lati indicando gli estremi

        CALL RLINE (l2A0x, l2A0y, l2Ax, l2Ay)
        CALL RLINE (l3Ax, l3Ay, l3Bx, l3By)
        CALL RLINE (l4Bx, l4By, l4B0x, l4B0y)
        CALL RLINE (l1B0x, l1B0y, l1A0x, l1A0y)

ELSE IF (i==48) THEN condizione

        !disegno ciascuno dei lati indicando gli estremi

        CALL RLINE (l2A0x, l2A0y, l2Ax, l2Ay)
        CALL RLINE (l3Ax, l3Ay, l3Bx, l3By)
        CALL RLINE (l4Bx, l4By, l4B0x, l4B0y)
        CALL RLINE (l1B0x, l1B0y, l1A0x, l1A0y)

END IF condizione

END DO ciclo

CALL CURVE (xM2,yM2,n)
CALL CURVE (xM3,yM3,n)
CALL CURVE (xM4,yM4,n)

!imposto la legenda
```

```
CALL LEGINI (stringa,3,10) !carattere, righe, lunghezza

CALL LEGLIN (stringa,'traiet. M2',1)
CALL LEGLIN (stringa,'traiet. M3',2)
CALL LEGLIN (stringa,'traiet. M4',3)

CALL LEGTIT ('Legenda')   !titolo legenda
CALL LEGEND (stringa,3)                !posizine in alto a destra

CALL DISFIN

END SUBROUTINE parallelogramma

!---------------------------------------------------------------------

SUBROUTINE congruente (x2b,y2b,theta3b,x3b,y3b,theta4b,x4b,y4b,theta2b,conf)

!sezione dichiarativa

IMPLICIT NONE

!dichiaro gli argomenti fittizi

REAL,INTENT(OUT),DIMENSION(9)::conf
REAL,INTENT(IN)::x2b,y2b,theta3b,x3b,y3b,theta4b,x4b,y4b,theta2b

!dichiaro le variabili locali

!dichiaro le coordinate

REAL::x2,y2,theta3,x3,y3,theta4,x4,y4,theta2

!dichiaro il vettore delle coordinate dipendenti

REAL,DIMENSION(8,100)::u

!dichiaro la funzione delle equazioni di vincolo

REAL,DIMENSION(8,100)::psi

!dichiaro la matrice jacobiana e la sua inversa

REAL,DIMENSION (8,8,100)::J
REAL(KIND=8),DIMENSION (8,8)::invj

!dichiaro il vettore dei resti

REAL,DIMENSION(8,100)::resto

!dichiaro la norma del vettore dei resti

REAL,DIMENSION(100):: norma

!altre variabili di lavoro

REAL(KIND=8),DIMENSION(8,8)::matrix
```

```
INTEGER::i !indici dei cicli

!sezione esecutiva

x2     = x2b
y2     = y2b
theta3 = theta3b
x3     = x3b
y3     = y3b
theta4 = theta4b
x4     = x4b
y4     = y4b
theta2 = theta2b

CALL BMPMOD (300,'inch','resolution') !fisso risoluzione dell'immagine in formato .bmp

!inizializzo i valori d'innesco di u

u(1,1) = x2
u(2,1) = y2
u(3,1) = theta3
u(4,1) = x3
u(5,1) = y3
u(6,1) = theta4
u(7,1) = x4
u(8,1) = y4

!inizializzo la matrice jacobiana con i valori costanti

j(:,:,:)=0

j(1,1,:) = 1
j(2,2,:) = 1
j(3,1,:) = -1
j(3,4,:) = 1
j(4,2,:) = -1
j(4,5,:) = 1
j(5,4,:) = -1
j(5,7,:) = 1
j(6,5,:) = -1
j(6,8,:) = 1
j(7,7,:) = 1
j(8,8,:) = 1

!inizializzo il vettore resto

resto=0

!ciclo del metodo Newton-Raphson

i=0

ciclo: DO

                                        i=i+1

                                        x2     = u(1,i)
```

```
y2    = u(2,i)
theta3 = u(3,i)
x3    = u(4,i)
y3    = u(5,i)
theta4 = u(6,i)
x4    = u(7,i)
y4    = u(8,i)


!calcolo la matrice jacobiana

j(3,3,i) = (0.5)*l3*SIN(theta3)
j(4,3,i) = -(0.5)*l3*COS(theta3)
j(5,3,i) = (0.5)*l3*SIN(theta3)
j(5,6,i) = (0.5)*l4*SIN(theta4)
j(6,3,i) = -(0.5)*l3*cos(theta3)
j(6,6,i) = -(0.5)*l4*cos(theta4)
j(7,6,i) = -(0.5)*l4*sin(theta4)
j(8,6,i) = (0.5)*l4*cos(theta4)

!calcolo l'inversa della matrice jacobiana

matrix(:,:)=j(:,:,i)

CALL inverse(matrix,invj,8)

!calcolo la funzione psi

psi(1,i) = x2 - ((0.5)*l2*COS(theta2))
psi(2,i) = y2 - (0.5)*l2*SIN(theta2)
psi(3,i) = x3 -(l3*0.5)*COS(theta3) - x2 - (l2*0.5)*COS(theta2)
psi(4,i) = y3 -(l3/2.)*SIN(theta3) - y2 - (l2*0.5)*SIN(theta2)
psi(5,i) = x4 -x3 -(0.5)*l3*COS(theta3) -(0.5)*l4*COS(theta4)
psi(6,i) = y4 -y3 -(0.5)*l3*SIN(theta3) -(0.5)*l4*SIN(theta4)
psi(7,i) = x4 - l1 + (0.5)*l4*COS(theta4)
psi(8,i) = y4 + (0.5)*l4*sin(theta4)

!calcola il vettore resto

resto(:,i)=MATMUL(invj,psi(:,i))

!calcola la norma del resto

norma(i)=SQRT(DOT_PRODUCT(resto(:,i),resto(:,i)))

!verifico il valore della norma e in caso esco

IF (norma(i)<=0.01) EXIT ciclo

!calcola le coordinate della iterata successiva

u(1:8,i+1)=u(1:8,i)-resto(1:8,i)

END DO ciclo

!assegna il valore a conf
```

```
conf(1) = u(1,i)
conf(2) = u(2,i)
conf(3) = u(3,i)
conf(4) = u(4,i)
conf(5) = u(5,i)
conf(6) = u(6,i)
conf(7) = u(7,i)
conf(8) = u(8,i)
conf(9) = theta2

END SUBROUTINE congruente

!------------------------------------------------------------------------------

END MODULE mod_ese_due_b
```