Analisi dinamica inversa di un quadrilatero articolato

Dati:

Si consideri un quadrilatero articolato caratterizzato dai seguenti parametri:

1. raggio della manovella A_0A: r = 100 mm;
2. lunghezza della biella AB: b = 315 mm;
3. lunghezza della bilanciere B_0B: c = 500 mm;
4. lunghezza del telaio A_0B_0: d = 425 mm;
5. angolo di inclinazione del telaio: $\delta = (7/4)\pi$;
6. momento d'inerzia della manovella $J_{A0} = 0.0245$ kg m^2;
7. massa della biella $m_2 = 2$ kg;
8. momento d'inerzia baricentrico della biella $J_2 = 0.016$ kg m^2;
9. massa del bilanciere $m_3 = 3.5$ kg;
10. momento d'inerzia baricentrico del bilanciere $J_3 = 0.066$ kg m^2;
11. la manovella ruoti con velocità angolare costante pari a n = 150 rpm.

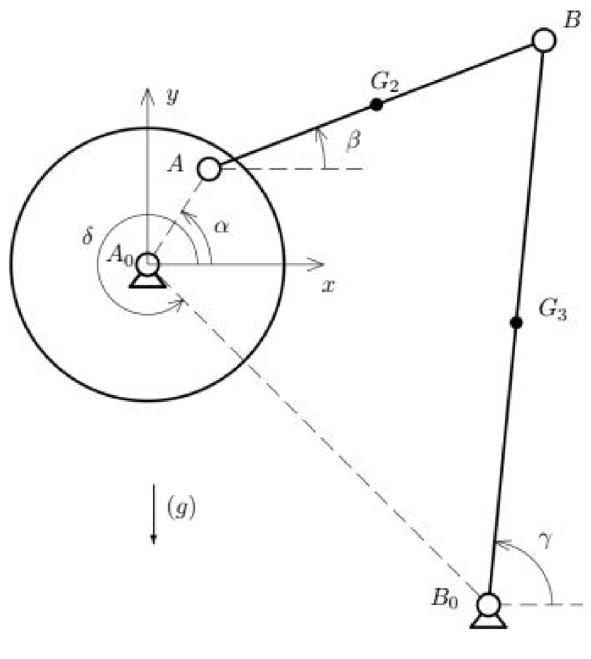

Figura 1: Rappresentazione schematica del sistema

Richiesta:

si chiede di determinare l'andamento della coppia motrice, rispetto all'angolo di manovella, che consente di mantenere la condizione di moto.

Soluzione:

Per la risoluzione del problema si scriva l'equazione di bilancio delle potenze:

$$M_m\dot{\alpha}+\mathbf{P}_2^T\mathbf{v_{G2}}+\mathbf{P}_3^T\mathbf{v_{G3}}-(m_2\mathbf{a_{G2}^T})\mathbf{v_{G2}}-(m_3\mathbf{a_{G3}^T})\mathbf{v_{G3}}-J_2\dot{\beta}\ddot{\beta}-J_3\dot{\gamma}\ddot{\gamma}=0 \qquad (1)$$

l'equazione può essere riscritta in forma matriciale dove:

$$\mathbf{P}_2=\begin{bmatrix}0\\-m_2g\end{bmatrix}\quad \mathbf{P}_3=\begin{bmatrix}0\\-m_3g\end{bmatrix} \qquad (2)$$

$$\mathbf{v}_{G2}=\begin{bmatrix}v_{G2x}\\v_{G2y}\end{bmatrix}\quad \mathbf{v}_{G3}=\begin{bmatrix}v_{G3x}\\v_{G3y}\end{bmatrix}\quad \mathbf{a}_{G2}=\begin{bmatrix}v_{G2x}\\v_{G2y}\end{bmatrix}\quad \mathbf{a}_{G3}=\begin{bmatrix}v_{G3x}\\v_{G3y}\end{bmatrix} \qquad (3)$$

Per il calcolo delle componenti lungo x e lungo y delle velocità e delle accelerazioni dei punti G2 e G3, si possono seguire diversi approcci, tra i quali l'approccio geometrico, quello con i numeri complessi e l'approccio vettoriale.

Approccio geometrico:

Considerando come origine del sistema di riferimento assoluto il punto A_0 (figura 1), le coordinate dei punti G2 e G3 possono essere scritte come:

$$\begin{cases} x_{G2}=rcos\alpha+AG_2cos\beta \\ y_{G2}=rsin\alpha+AG_2cos\beta \\ x_{G3}=dcos2\pi-\delta+B_0G_3cos\gamma=dcos\delta+B_0G_3cos\gamma \\ y_{G3}=-dsin2\pi-\delta+B_0G_3sin\gamma=dsin\delta+B_0G_3sin\gamma \end{cases} \qquad (4)$$

Eseguendo la derivata prima e seconda si ricavano velocità e accelerazione:

$$\begin{cases} v_{G2x}=-\dot{\alpha}rsin\alpha-\dot{\beta}AG_2sin\beta \\ v_{G2y}=\dot{\alpha}rcos\alpha+\dot{\beta}AG_2cos\beta \\ v_{G3x}=-\dot{\gamma}B_0G_3sin\gamma \\ v_{G3y}=\dot{\gamma}B_0G_3cos\gamma \end{cases} \qquad (5)$$

$$\begin{cases} a_{G2x}=-\dot{\alpha}^2rcos\alpha-\ddot{\alpha}rsin\alpha-\dot{\beta}^2AG_2cos\beta-\ddot{\beta}AG_2sin\beta \\ a_{G2y}=-\dot{\alpha}^2rsin\alpha+\ddot{\alpha}rcos\alpha-\dot{\beta}^2AG_2sin\beta+\ddot{\beta}AG_2cos\beta \\ a_{G3x}=-\dot{\gamma}^2B_0G_3cos\gamma-\ddot{\gamma}B_0G_3sin\gamma \\ a_{G3y}=-\dot{\gamma}^2B_0G_3sin\gamma+\ddot{\gamma}B_0G_3cos\gamma \end{cases} \qquad (6)$$

termini che contengono $\ddot{\alpha}$ saranno nulli poiché la velocità della manovella è considerata costante.

Per la determinazione delle velocità $\dot{\beta}$, $\dot{\gamma}$ e delle accelerazioni $\ddot{\beta}$, $\ddot{\gamma}$ si pu`o ricorrere all'analisi cinematica condotta mediante il "metodo dei loop vettoriali".

Analisi di posizione:

$$re^{i\alpha} + be^{i\beta} = ce^{i\gamma} + de^{i\delta} \qquad (7)$$

Per la risoluzione dei angoli β e γ dato l'angolo di manovella α, si usa il metodo metodo di Newton-Raphson. Quindi si sviluppa in serie di Taylor nell'intorno dei valori di primo tentativo β_0 e γ_0, arrestandosi al I ordine, la funzione che descrive la posizione

$$f(\alpha,\beta,\gamma) = re^{i\alpha} + be^{i\beta} - ce^{i\gamma} - de^{i\delta} = 0 \qquad (8)$$

$$f(\alpha,\beta,\gamma) \approx f(\alpha,\beta_0,\gamma_0) + \left.\frac{\partial f}{\partial \beta}\right|_{(\beta_0,\gamma_0)} (\beta-\beta_0) + \left.\frac{\partial f}{\partial \gamma}\right|_{(\beta_0,\gamma_0)} (\gamma-\gamma_0) \qquad (9)$$

$$\frac{\partial f}{\partial \beta} = ibe^{i\beta} \quad , \quad \frac{\partial f}{\partial \gamma} = -ice^{i\gamma} \qquad (10)$$

Il nuovo valore della soluzione sarà quindi valutato come:

$$\mathbf{x} = \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \mathbf{x}_0 - \begin{pmatrix} Real(ibe^{i\beta_0}) & Real(-ice^{i\gamma_0}) \\ Imag(ibe^{i\beta_0}) & Imag(-ice^{i\gamma_0}) \end{pmatrix}^{-1} \begin{bmatrix} Real(f(\alpha_0,\beta_0,\gamma_0)) \\ Imag(f(\alpha_0,\beta_0,\gamma_0)) \end{bmatrix} \qquad (11)$$

da cui è possibile scrivere la formula ricorsiva:

$$x_{i+1} = x_i - \mathbf{J}_i^{-1} R_i \qquad (12)$$

```
Dx = np.dot(-inv(J),R)
x = x + Dx
```

Analisi di velocità:

Integrando l'equazione complessa che descrive la posizione, si ottiene:

$$ir\dot{\alpha}e^{i\alpha} + ib\dot{\beta}e^{i\beta} - ic\dot{\gamma}e^{i\gamma} = 0 \qquad (13)$$

$$ir\dot{\alpha}e^{i\alpha} = -ib\dot{\beta}e^{i\beta} + ic\dot{\gamma}e^{i\gamma} \qquad (14)$$

$$\begin{bmatrix} \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \begin{pmatrix} Real(-ibe^{i\beta}) & Real(ice^{i\gamma}) \\ Imag(-ibe^{i\beta}) & Imag(ice^{i\gamma}) \end{pmatrix}^{-1} = \begin{bmatrix} Real(ir\dot{\alpha}e^{i\alpha}) \\ Imag(ir\dot{\alpha}e^{i\alpha}) \end{bmatrix} \qquad (15)$$

```
      xp = np.dot(inv(np.array([[cbetap.real, cgammap.real], [cbetap.imag,
cgammap.imag]])),np.array([Tn.real, Tn.imag]).transpose())
    betap = xp[0]
    gammap = xp[1]
```

Analisi di accelerazione:

Integrando l'equazione complessa che descrive le velocità, si ottiene:

$$ir\ddot{\alpha}e^{i\alpha} - r\dot{\alpha}^2 e^{i\alpha} + ib\ddot{\beta}e^{i\beta} - b\dot{\beta}^2 e^{i\beta} - ic\ddot{\gamma}e^{i\gamma} + c\dot{\gamma}^2 e^{i\gamma} = 0 \qquad (16)$$

$$-ib\ddot{\beta}e^{i\beta} + ic\ddot{\gamma}e^{i\gamma} = -ir\ddot{\alpha}e^{i\alpha} + r\dot{\alpha}^2 e^{i\alpha} + b\dot{\beta}^2 e^{i\beta} - c\dot{\gamma}^2 e^{i\gamma} \qquad (17)$$

$$\begin{bmatrix} \ddot{\beta} \\ \ddot{\gamma} \end{bmatrix} = \begin{pmatrix} Real(-ibe^{i\beta}) & Real(ice^{i\gamma}) \\ Imag(-ibe^{i\beta}) & Imag(ice^{i\gamma}) \end{pmatrix}^{-1} \begin{bmatrix} Real(-ir\ddot{\alpha}e^{i\alpha} + r\dot{\alpha}^2 e^{i\alpha} + b\dot{\beta}^2 e^{i\beta} - c\dot{\gamma}^2 e^{i\gamma}) \\ Imag(-ir\ddot{\alpha}e^{i\alpha} + r\dot{\alpha}^2 e^{i\alpha} + b\dot{\beta}^2 e^{i\beta} - c\dot{\gamma}^2 e^{i\gamma}) \end{bmatrix} \qquad (18)$$

```
xpp = np.dot(inv(np.array([[cbetap.real, cgammap.real], [cbetap.imag,
cgammap.imag]])),np.array([Tna.real, Tna.imag]).transpose())
    matx1.append(-xpp);
    betapp = -xpp[0]
    gammapp = -xpp[1]
```

Anche in questo caso, i termini contenenti $\ddot{\alpha}$ saranno nulli poiché la velocità della manovella è costante.

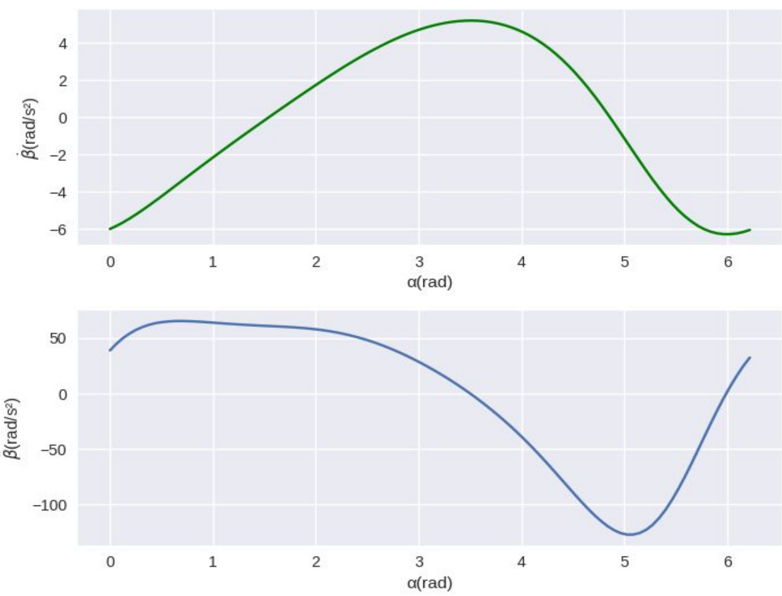

Figura 2: Velocità e accelerazione angolare della biella

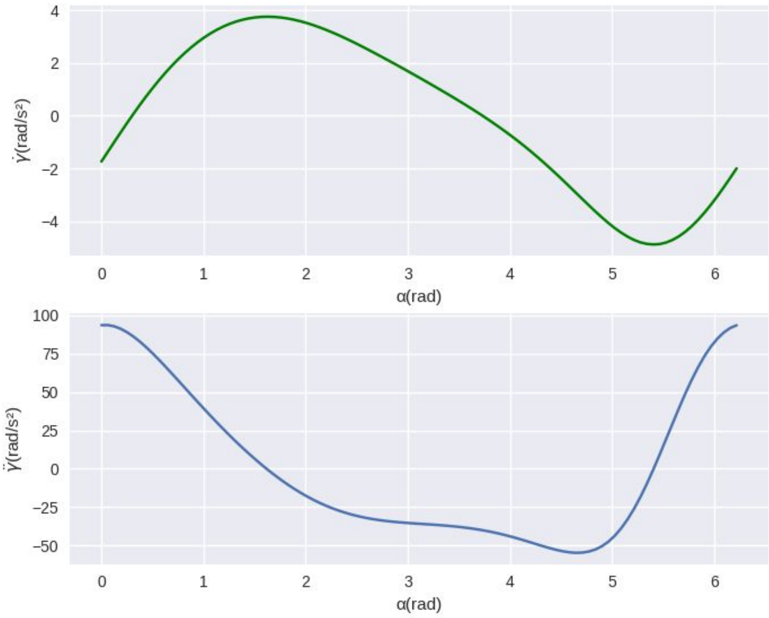

Figura 3: Velocità e accelerazione angolare del bilanciere

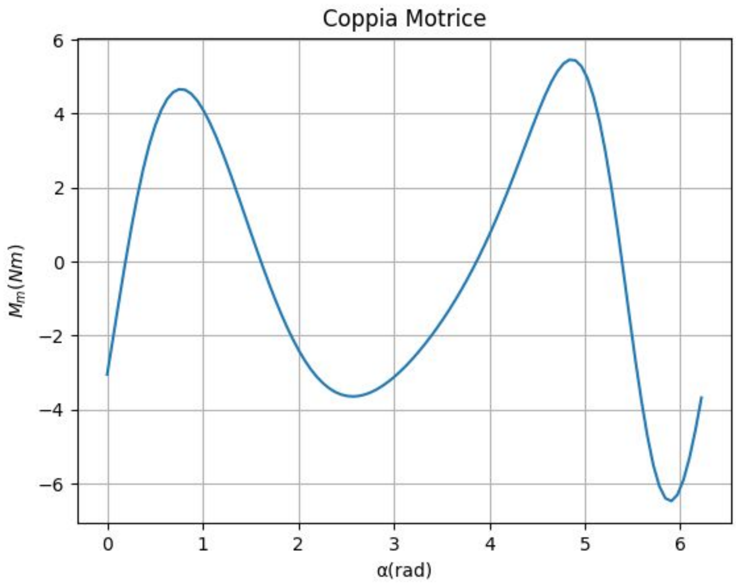

Figura 4: Coppia motrice

Codice:

```
# Author: Mohsen Ghalbi
# Date: 31/12/2019
# Description: Qudrilatero articolato

import numpy as np
import matplotlib.pyplot as plt
import cmath
from numpy.linalg import inv
import matplotlib as mpl

r=100e-3
b=315e-3
c=500e-3
d=425e-3
delta=7/4*np.pi
m2=2
J2=0.016
m3=3.5
J3=0.066
n=150; # [rpm]
w=n*2*np.pi/60  # [rad/s]

dalpha = 2*np.pi/100
alphaf = 2*np.pi
valpha = np.arange(0,alphaf, dalpha)
toll = 0.001
beta0 = 15*np.pi/180
gamma0 = 100*np.pi/180
Mm = []

def resto(alpha, x):
    beta = x[0]
    gamma = x[1]
    f = r*cmath.exp(complex(0,1)*alpha) +
b*cmath.exp(complex(0,1)*beta)-c*cmath.exp(complex(0,1)*gamma)-
d*cmath.exp(complex(0,1)*delta)
    return np.array([f.real, f.imag]).transpose()

def jacob(x):
    beta = x[0]
    gamma = x[1]
    dfbeta = complex(0,1)*b*cmath.exp(complex(0,1)*beta)
    dfgamma = -complex(0,1)*b*cmath.exp(complex(0,1)*gamma)
    return np.array([[dfbeta.real, dfgamma], [dfbeta.imag, dfgamma.imag]])

x = [beta0, gamma0]
matx = []
```

```
matx1 = []
for i in np.arange(0, len(valpha)):
    # anailisi di posizione
    Dx = [10*toll, 10*toll]
    R = [10*toll, 10*toll]

    while (abs(Dx[0]) > toll or abs(Dx[1]) > toll or abs(R[0]) > toll or
abs(R[1]) > toll):
        R = resto(valpha[i],x)
        J = jacob(x)
        Dx = np.dot(-inv(J),R)
        x = x + Dx
    beta = x[0]
    gamma = x[1]

    # analisi della velocità
    cbetap = -complex(0,1)*b*cmath.exp(complex(0,1)*beta)
    cgammap = complex(0,1)*c*cmath.exp(complex(0,1)*gamma)
    Tn = complex(0,1)*r*w*cmath.exp(complex(0,1)*valpha[i])
    xp = np.dot(inv(np.array([[cbetap.real, cgammap.real], [cbetap.imag,
cgammap.imag]])),np.array([Tn.real, Tn.imag]).transpose())
    betap = xp[0]
    gammap = xp[1]
    matx.append(xp);
    # analisi della accelerazione
    Tna = r*pow(w,
2)*cmath.exp(complex(0,1)*valpha[i])+b*pow(betap,2)*cmath.exp(complex(0,1)*beta)
-c*pow(gammap,2)*cmath.exp(complex(0,1)*gamma)
    xpp = np.dot(inv(np.array([[cbetap.real, cgammap.real], [cbetap.imag,
cgammap.imag]])),np.array([Tna.real, Tna.imag]).transpose())
    matx1.append(-xpp);
    betapp = -xpp[0]
    gammapp = -xpp[1]
    # velocità e accelerazioni baricentri
    vG2 =
complex(0,1)*w*r*cmath.exp(complex(0,1)*valpha[i])+complex(0,1)*betap*b/2*cmath.
exp(complex(0,1)*beta);
    vG3 = complex(0,1)*gammap*c/2*cmath.exp(complex(0,1)*gamma)
    aG2 =
-pow(w,2)*r*cmath.exp(complex(0,1)*valpha[i])+complex(0,1)*betapp*b/2*cmath.exp(
complex(0,1)*beta)-pow(betap,2)*b/2*cmath.exp(complex(0,1)*beta)
    aG3 =
complex(0,1)*gammapp*c/2*cmath.exp(complex(0,1)*gamma)-pow(gammap,2)*c/2*cmath.e
xp(complex(0,1)*gamma)
    g = 9.81
    P2 = np.array([0, -m2*g])
    P3 = np.array([0, -m3*g])
    vvG2 = np.array([vG2.real, vG2.imag]).transpose()
    vvG3 = np.array([vG3.real, vG3.imag]).transpose()
    vaG2 = np.array([aG2.real, aG2.imag])
```

```python
    vaG3 = np.array([aG3.real, aG3.imag])

Mm.append((-np.dot(P2,vvG2)-np.dot(P3,vvG3)-(-m2*np.dot(vaG2,vvG2))-(-m3*np.dot(
vaG3,vvG3))-(-J2*betapp*betap)-(-J3*gammapp*gammap))/w)

X = [x[0] for x in matx]
X1 = [x[0] for x in matx1]
Y = [x[1] for x in matx]
Y1 = [x[1] for x in matx1]

# Grafico della coppia motrice
plt.plot(valpha, Mm)
plt.xlabel('\u03B1(rad)')
plt.ylabel('$M_m(Nm)$')
plt.title('Coppia Motrice')
plt.grid()
fig = plt.gcf()
fig.canvas.set_window_title('Coppia Motrice')
plt.show()

#Grafico velocità e accelerazione della biella
mpl.style.use('seaborn')
fig, (ax1, ax2) = plt.subplots(2);
ax1.set_xlabel('\u03B1(rad)')
ax1.set_ylabel('$\dot\u03B2$(rad/s\u00b2)')
ax1.set_title('Velocità angolare della biella')
ax1.plot(valpha, X, color='green')
ax2.set_xlabel('\u03B1(rad)')
ax2.set_ylabel('$\ddot\u03B2$(rad/s\u00b2)')
ax2.set_title('Accelerazione angolare della biella')
ax2.plot(valpha, X1)
fig.canvas.set_window_title('Velocità & Accelerazione biella')
plt.show()

#Grafico velocità e accelerazione del bilanciere
fig, (ax1, ax2) = plt.subplots(2);
ax1.set_xlabel('\u03B1(rad)')
ax1.set_ylabel('$\dot\u03B3$(rad/s\u00b2)')
ax1.set_title('Velocità angolare del bilanciere')
ax1.plot(valpha, Y, color='green')
ax2.set_xlabel('\u03B1(rad)')
ax2.set_ylabel('$\ddot\u03B3$(rad/s\u00b2)')
ax2.set_title('Accelerazione angolare del bilanciere')
ax2.plot(valpha, Y1)
fig.canvas.set_window_title('Velocità & Accelerazione bilanciere')
plt.show()
```