Spacecraft Modeling, Attitude Determination, and Control: Quaternion-Based Approach

Modellazione ADCS Veicolo Spaziale Quaternioni

Capitolo 2: Dinamica e Proprietà delle Orbite

2.1 Dinamica Orbitale

Il moto di un veicolo spaziale è governato dalle leggi della meccanica classica. Partiamo dalla seconda legge di Newton, dove la forza $\mathbf{f}$ è la derivata del momento lineare $\mathbf{p} = m\mathbf{v}$ rispetto al tempo.

Il Problema dei Due Corpi

L'interazione tra due masse m_1 e m_2 a distanza r è descritta dalla **Legge di Gravitazione Universale**:

$$\mathbf{f} = G\frac{m_1 m_2}{r^3}\mathbf{r}$$

dove $G = 6.669 \cdot 10^{-11}, \mathrm{m}^3/(\mathrm{kg} \cdot \mathrm{s}^2)$ è la costante gravitazionale.

Intuizione Ingegneristica: In un sistema a due corpi (es. Terra e satellite), se la massa del corpo centrale M è molto maggiore di quella del satellite, il momento angolare specifico $\mathbf{h} = \mathbf{r} \times \mathbf{v}$ rimane costante. Questo implica che **l'orbita giace su un piano fisso nello spazio**, un concetto fondamentale per la stabilità della missione.

Definiamo il **parametro gravitazionale standard** $\mu = GM$ (per la Terra, noto come costante gravitazionale geocentrica). L'equazione differenziale del moto può essere trasformata in una forma lineare definendo $u = 1/r$, portando alla soluzione generale della traiettoria:

$$u = \frac{\mu}{h^2} + c\cos(\theta - \theta_0)$$

.

Riepilogo Schematico 2.1

- **Forza Gravitazionale:** Attrattiva e centrale.
- **Conservazione:** Il momento angolare $\mathbf{h}$ e l'energia totale E sono costanti del moto.

- **Piano Orbitale:** Risultato della conservazione di $\mathbf{h}$.

Formule Fondamentali 2.1

1. **Momento Angolare Specifico:** $h = r^2 \frac{d\theta}{dt}$ [unità: m^2/s].
2. **Energia Totale per Unità di Massa:** $E = \frac{v^2}{2} - \frac{\mu}{r}$ [unità: J/kg].
3. **Parametro Gravitazionale:** $\mu = GM$ [unità: $\mathrm{m}^3/\mathrm{s}^2$].

Applicazione Python 2.1

```
import numpy as np

def orbital_velocity(mu, r, a=None):
    # Calcola la velocità per orbita circolare o ellittica
    if a is None: # Orbita circolare
        return np.sqrt(mu / r)
    else: # Orbita ellittica (Equazione vis-viva)
        return np.sqrt(mu * (2/r - 1/a))

# Esempio: Velocità satellite a 400km (LEO)
mu_earth = 3.986e14 # m^3/s^2
r_leo = 6371000 + 400000 # r_terra + altitudine
print(f"Velocità LEO: {orbital_velocity(mu_earth, r_leo):.2f}
m/s")
```

2.2 Sezioni Coniche e Tipi di Orbite

La soluzione dell'equazione del moto descrive una **sezione conica**. La distanza r in funzione dell'anomalia vera θ è data da:

$$r = \frac{p}{1 + e\cos(\theta - \theta_0)}$$

.

Classificazione delle Orbite in base all'Eccentricità (e):

1. **Circolare ($e = 0$):** Energia $E < 0$. La velocità è costante: $v^2 = \mu/r$.
2. **Ellittica ($0 < e < 1$):** Energia $E < 0$. L'orbita è chiusa.
   - **Perigeo (r_p):** Punto più vicino, $\theta = 0^\circ$.
   - **Apogeo (r_a):** Punto più lontano, $\theta = 180^\circ$.

3. **Parabolica ($e = 1$):** Velocità di fuga, $E = 0$.
4. **Iperbolica ($e > 1$):** Energia $E > 0$. Il veicolo lascia il campo gravitazionale.

Interpretazione Fisica: Il **semi-asse maggiore (a)** determina la dimensione dell'orbita e la sua energia. Nelle orbite ellittiche, la velocità è massima al perigeo e minima all'apogeo.

Riepilogo Schematico 2.2

- **Geometria:** L'orbita è una sezione conica con il corpo centrale in uno dei fuochi.
- **Stato Energetico:** Orbite chiuse (cerchio/ellisse) hanno energia negativa; orbite aperte (iperbole) hanno energia positiva.

Formule Fondamentali 2.2

1. **Semi-latus rectum:** $p = h^2/\mu$.
2. **Eccentricità:** $e = \frac{r_a - r_p}{r_a + r_p}$.
3. **Relazione Energia/Semi-asse maggiore:** $E = -\frac{\mu}{2a}$.

2.3 Proprietà delle Orbite Kepleriane

In questa sezione approfondiamo la cinematica orbitale.

Le Leggi di Keplero

- **Seconda Legge (Legge delle Aree):** Il raggio vettore spazza aree uguali in tempi uguali ($dA/dt = h/2 = \text{costante}$).
- **Terza Legge (Legge dei Periodi):** Il quadrato del periodo orbitale è proporzionale al cubo del semi-asse maggiore:

$$T = 2\pi\sqrt{\frac{a^3}{\mu}}$$

.

Equazione del Tempo di Keplero

Per trovare la posizione al tempo t, usiamo l'**anomalia media (M)** e l'**anomalia eccentrica (ψ)**:

$$M = \psi - e \sin \psi = \omega_0 (t - t_p)$$

. Dove $\omega_0 = \sqrt{\mu / a^3}$ è il moto medio.

Formule Fondamentali 2.3

1. **Periodo Orbitale:** $T = 2\pi \sqrt{a^3 / \mu}$.
2. **Anomalia Media:** $M = \frac{2\pi}{T}(t - t_p)$.
3. **Relazione Anomalie:** $\tan(\theta/2) = \sqrt{\frac{1+e}{1-e}} \tan(\psi/2)$.

2.4 Trasferimento di Hohmann

È la manovra orbitale a **minimo consumo di propellente** per spostarsi tra due orbite circolari complanari.

Procedura Ingegneristica:

1. **Primo Impulso (Δv_1):** Al punto A, si accelera per immettersi in un'orbita ellittica di trasferimento.
2. **Secondo Impulso (Δv_2):** Al punto B (apogeo dell'ellisse), si accelera nuovamente per circolarizzare l'orbita alla nuova quota.

Il calcolo si basa sulla variazione di energia necessaria per cambiare il semi-asse maggiore dell'orbita.

Applicazione Python 2.4 (Hohmann Transfer)

```
def hohmann_transfer(mu, r1, r2):
    # Velocità iniziali e finali nelle orbite circolari
    v1 = np.sqrt(mu / r1)
    v2 = np.sqrt(mu / r2)

    # Semi-asse maggiore dell'ellisse di trasferimento
```

```
    a_trans = (r1 + r2) / 2

    # Velocità al perigeo e apogeo dell'ellisse di trasferimento
    v_perigee = np.sqrt(mu * (2/r1 - 1/a_trans))
    v_apogee = np.sqrt(mu * (2/r2 - 1/a_trans))

    dv1 = v_perigee - v1
    dv2 = v2 - v_apogee

    return dv1, dv2, dv1 + dv2

dv1, dv2, total_dv = hohmann_transfer(mu_earth, r_leo, 42164000) # Da LEO a GEO
print(f"Delta-V Totale per trasferimento GEO: {total_dv:.2f} m/s")
```

2.5 Orbite nello Spazio Tridimensionale

Per descrivere un'orbita nello spazio 3D, utilizziamo il sistema di coordinate inerziali geocentrico (basato sull'equatore e sull'equinozio di primavera).

I 6 Parametri Orbitali Classici (α):

1. a **(Semi-asse maggiore):** Dimensione dell'orbita.
2. e **(Eccentricità):** Forma dell'orbita.
3. i **(Inclinazione):** Angolo tra il piano orbitale e l'equatore.
4. Ω **(Ascensione retta del nodo ascendente - RAAN):** Orientamento del piano orbitale rispetto al punto vernale.
5. ω **(Argomento del perigeo):** Orientamento dell'orbita nel suo piano.
6. M **(Anomalia media):** Posizione del satellite lungo l'orbita al tempo t.

Materiale di Sintesi Finale

Mappa Concettuale Globale

1. **Dinamica (Leggi di Newton/Gravità)** $\rightarrow$ Portano alla conservazione di h ed E.
2. **Geometria (Sezioni Coniche)** $\rightarrow$ Definiscono traiettorie chiuse (missioni orbitanti) o aperte (missioni interplanetarie).

3. **Cinematica (Leggi di Keplero/Anomalie)** $\rightarrow$ Determinano la posizione nel tempo.
4. **Manovre (Hohmann)** $\rightarrow$ Applicazione pratica per il cambio di orbita.
5. **Parametrizzazione (6 Elementi Orbitali)** $\rightarrow$ Definizione univoca nello spazio 3D.

Formulario Completo

- **Velocità Orbitale (Generale):** $v = \sqrt{\mu(\frac{2}{r} - \frac{1}{a})}$
- **Equazione Polare:** $r = \frac{a(1-e^2)}{1+e\cos\theta}$
- **Periodo:** $T = 2\pi\sqrt{a^3/\mu}$
- **Energia Specifica:** $E = -\mu/2a$
- **Equazione di Keplero:** $M = \psi - e\sin\psi$

Guida di Ripasso Rapido

1. **Perché le orbite sono piane?** Perché la forza di gravità è centrale, quindi il momento angolare si conserva in direzione e modulo.
2. **Qual è il significato fisico dell'energia negativa?** Indica che il satellite è "legato" gravitazionalmente al corpo centrale; non ha abbastanza velocità per fuggire.
3. **Come si calcola la posizione futura?** Si calcola M dal tempo trascorso, si risolve l'equazione di Keplero per ψ, e si ricava l'anomalia vera θ.
4. **Quali parametri definiscono il piano orbitale in 3D?** L'inclinazione (i) e la RAAN (Ω).

Capitolo 3: Sequenze di Rotazione e Quaternioni

3.1 Sistemi di Riferimento (Frames) Fondamentali

Nello spazio non esiste un "alto" o un "basso" assoluto. Dobbiamo definire dei sistemi di riferimento (frame) per localizzare il satellite e i suoi strumenti.

- **Body-fixed Frame (Sistema Solidale):** Ha l'origine nel centro di massa del veicolo. Gli assi sono solitamente allineati con gli **assi principali di inerzia** del satellite (derivati dalla matrice d'inerzia J). L'asse X_b punta verso la direzione della velocità, Z_b verso il basso (nel piano orbitale) e Y_b completa la terna destrorsa.

- **Earth-Centered Inertial (ECI):** Fondamentale per applicare le leggi di Newton. L'asse X_I punta verso il **punto d'ariete** (equinozio vernale), Z_I lungo l'asse di rotazione terrestre e Y_I segue la regola della mano destra.
- **Local Vertical Local Horizontal (LVLH):** Molto usato per l'osservazione della Terra. L'asse Z_{lvlh} punta sempre verso il centro della Terra (**Nadir**).
- **Perifocal (PQW) e RSW Frame:** Il frame PQW ha il piano fondamentale coincidente con il piano dell'orbita (P verso il perigeo), mentre il frame RSW (Radial, Along-track, Cross-track) è centrato sul satellite e ruota con esso lungo l'orbita.

3.2 Rappresentazione Matematica delle Rotazioni

La trasformazione di un punto tra due sistemi di riferimento ruotati di un angolo θ avviene tramite **Matrici di Rotazione (DCM - Direction Cosine Matrix)**.

Ad esempio, una rotazione attorno all'asse Z (Rot3) trasforma le coordinate di un punto fisso P in un frame ruotato come segue:

$$[x_2\ y_2\ z_2] = [\cos\theta \quad \sin\theta \quad 0\ -\sin\theta \quad \cos\theta \quad 0\ 0 \quad 0 \quad 1]\,[x_1\ y_1\ z_1]$$

Intuizione Ingegneristica: Le matrici di rotazione sono **unitarie**. Questo significa che la loro trasformazione non cambia la lunghezza del vettore (non c'è deformazione, solo riorientamento) e la loro inversa coincide con la trasposta ($C^{-1} = C^T$).

Variazione Temporale e Derivate

In dinamica, non ci interessa solo l'assetto statico, ma quanto velocemente cambia. La derivata temporale della DCM $A(t)$ è legata alla velocità angolare ω dalla relazione:

$$\frac{dA}{dt} = -S(\omega)A(t)$$

Dove $S(\omega)$ è la matrice antisimmetrica (skew-symmetric) del vettore velocità angolare.

3.3 Trasformazioni e Parametri Orbitali

Esiste una relazione biunivoca tra la posizione/velocità del satellite (r, v) e i **sei parametri orbitali classici** $[a, e, i, \Omega, \omega, M]$.

- La trasformazione da ECI a PQW richiede tre rotazioni successive: Ω (longitudine del nodo ascendente), i (inclinazione) e ω (argomento del perigeo).

3.4 I Quaternioni: Lo Standard Industriale

Mentre gli angoli di Eulero soffrono del problema del "Gimbal Lock" (singolarità matematica), i **quaternioni** offrono una rappresentazione globale e priva di singolarità. Un quaternione $\bar{q}$ è composto da una parte scalare q_0 e una parte vettoriale q:

$$\bar{q} = q_0 + iq_1 + jq_2 + kq_3$$

Nel contesto aerospaziale, usiamo quaternioni unitari dove $q_0 = \cos(\alpha/2)$ e $q = \hat{e}\sin(\alpha/2)$, con $\hat{e}$ asse di rotazione e α angolo di rotazione.

Riepilogo Schematico del Capitolo

- **Frames:** ECI (inerziale), Body (satellite), LVLH (nadir-pointing), PQW (orbitale).
- **Rotazioni:** DCM (matrici 3x3), Angoli di Eulero (sequenze), Quaternioni (4 parametri).
- **Cinematica:** La derivata della DCM dipende dalla velocità angolare ω.
- **Vantaggi Quaternioni:** Nessuna singolarità, efficienza computazionale.

Elenco Formule Fondamentali

1. **DCM Rotazione Asse 3:**
   $Rot3(\theta) = [\cos\theta \quad \sin\theta \quad 0 \; -\sin\theta \quad \cos\theta \quad 0 \, 0 \quad 0 \quad 1]$.
2. **Cinematica DCM:** $\dot{A} = -S(\omega)A$.
3. **Prodotto di Quaternioni (Composizione):**
   $\bar{p} \otimes \bar{q} = p_0 q_0 - p \cdot q + p_0 q + q_0 p + p \times q$.
4. **Derivata del Quaternione:** $\dot{\bar{q}} = \frac{1}{2}\bar{q} \otimes (0, \omega)$.
5. **Matrice Antisimmetrica $S(\omega)$:** $[0 \quad -\omega_3 \quad \omega_2 \, \omega_3 \quad 0 \quad -\omega_1 \; -\omega_2 \quad \omega_1 \quad 0]$.

Applicazione Pratica in Python: Moltiplicazione di Quaternioni

```
import numpy as np
```

```
def quaternion_multiply(p, q):
    """
    Calcola il prodotto di due quaternioni p e q.
    p, q: array-like (q0, q1, q2, q3)
    """
    p0, p1, p2, p3 = p
    q0, q1, q2, q3 = q

    r = np.array([
        p0*q0 - p1*q1 - p2*q2 - p3*q3,
        p0*q1 + p1*q0 + p2*q3 - p3*q2,
        p0*q2 - p1*q3 + p2*q0 + p3*q1,
        p0*q3 + p1*q2 - p2*q1 + p3*q0
    ])
    return r

# Esempio: due rotazioni consecutive
q1 = np.array([0.707, 0.707, 0, 0]) # Rotazione di 90 gradi
attorno a X
q2 = np.array([0.707, 0, 0.707, 0]) # Rotazione di 90 gradi
attorno a Y
q_total = quaternion_multiply(q1, q2)
print(f"Quaternione Risultante: {q_total}")
```

Sezione Conclusiva: Strumenti di Ripasso

Mappa Concettuale Globale

1. **Input Fisico:** Stato orbitale (r, v) e Momento di Inerzia (J).
2. **Sistemi di Riferimento:** Collegamento tra ECI (fisso), Orbitale (PQW/RSW) e Body (satellite).
3. **Rappresentazione Assetto:**
   - *Matrici (DCM):* Intuitive ma ridondanti (9 parametri).
   - *Eulero:* Visualizzabili ma con singolarità.
   - *Quaternioni:* Robusti e standard per il controllo.
4. **Dinamica/Cinematica:** Come le coppie esterne cambiano ω e come ω cambia l'assetto ($\dot{q}$).

Formulario Completo

- **Identità Quaternione:** $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$.
- **Operatore di Rotazione:** $v_{body} = \bar{q} \otimes v_{inertial} \otimes \bar{q}^*$.
- **DCM da Quaternione:** $C = (q_0^2 - q^T q)I + 2qq^T - 2q_0 S(q)$.
- **Momento Angolare Orbitale:** $h = r \times v$.
- **Inclinazione:** $\cos(i) = h_z / |h|$.

Guida di Ripasso Rapido

1. Perché usiamo tanti frame? Perché ogni operazione ha il suo riferimento naturale. Le leggi fisiche vivono nell'ECI, ma i sensori stellari leggono nel Body frame, e la missione solitamente richiede di puntare verso il Nadir (LVLH).

2. Come scelgo tra DCM, Eulero e Quaternioni?

- Usa **Eulero** per presentazioni o per piccoli angoli (es. analisi di stabilità lineare).
- Usa la **DCM** per trasformare vettori tra frame noti.
- Usa i **Quaternioni** per l'integrazione numerica a bordo del computer di volo e per manovre ad ampio angolo.

3. Il concetto di Cinematica vs Dinamica: La cinematica (trattata qui) descrive il *movimento* (come ruota il frame). La dinamica (Capitoli successivi) descrive le *cause* (momenti torcenti, inerzia).

4. Trasformazioni Chiave: Ricorda che per passare da parametri orbitali a posizione/velocità devi risolvere l'equazione di Keplero $M = \psi - e\sin(\psi)$ per trovare l'anomalia eccentrica, poi usare le rotazioni di Eulero (Ω, i, ω) per mappare il piano orbitale nello spazio 3D ECI.

4.1 Le Equazioni Generali del Sistema Veicolo Spaziale

Ogni studio sulla dinamica del volo spaziale inizia distinguendo due aspetti fondamentali: **dinamica** (le forze e i momenti che causano il moto) e **cinematica** (la descrizione geometrica del moto nel tempo).

4.1.1 Equazione della Dinamica

La dinamica di un corpo rigido nello spazio è governata dalla variazione del suo momento angolare. Se definiamo J come la **matrice d'inerzia** (espressa in

$kg \cdot m^2$):

$$J = [J_{11} \quad J_{12} \quad J_{13}\, J_{21} \quad J_{22} \quad J_{23}\, J_{31} \quad J_{32} \quad J_{33}]$$

e ω_I come la velocità angolare del corpo rispetto a un sistema inerziale, l'equazione del moto nel sistema solidale al corpo (body frame) è:

$$J\dot{\omega}_I = -\omega_I \times (J\omega_I) + \mathbf{t}_d + \mathbf{u}$$

Dove:

- $\mathbf{t}_d$: Coppie di disturbo (gravitazionali, aerodinamiche, etc.).
- $\mathbf{u}$: Coppie di controllo generate dagli attuatori.

Intuizione Ingegneristica: Il termine $-\omega_I \times (J\omega_I)$ rappresenta le coppie giroscopiche. In assenza di momenti esterni, il momento angolare si conserva, ma la velocità angolare può variare se il corpo non è simmetrico, portando a moti complessi come la precessione.

4.1.2 Equazione della Cinematica

Per descrivere l'orientamento, utilizziamo il **quaternione d'assetto** $\bar{q} = [q_0, \mathbf{q}^T]^T$:

- $q_0 = \cos(\alpha/2)$ (parte scalare)
- $\mathbf{q} = \hat{\mathbf{e}} \sin(\alpha/2)$ (parte vettoriale, dove $\hat{\mathbf{e}}$ è l'asse di rotazione)

Il vantaggio cruciale dei quaternioni rispetto agli angoli di Eulero è l'assenza di singolarità (il cosiddetto "gimbal lock"). Tuttavia, un modello che utilizzi tutte e quattro le componenti risulta **non controllabile** linearmente. Per ovviare a ciò, le fonti propongono l'uso della sola **parte vettoriale** $\mathbf{q} = [q_1, q_2, q_3]^T$.

Riepilogo Schematico - Cap. 4.1

- **Dinamica**: Basata sulla conservazione del momento angolare (Eulero).
- **Cinematica**: Utilizza quaternioni ridotti per evitare singolarità e garantire la controllabilità.
- **Variabili chiave**: J (Inerzia), ω (Velocità angolare), $\bar{q}$ (Assetto).

Formule Fondamentali

1. Dinamica: $J\dot{\omega}_I = -S(\omega_I)(J\omega_I) + \mathbf{t}_d + \mathbf{u}$
2. Cinematica ridotta: $\dot{\mathbf{q}} = \frac{1}{2}Q(q_1, q_2, q_3)\omega$

Applicazione Pratica (Python)

```
import numpy as np

def compute_dynamics(J, omega, u, td=np.zeros(3)):
    # Calcolo della derivata della velocità angolare
    # J: matrice d'inerzia (3x3)
    # omega: vel. angolare (3,)
    # u: controllo (3,)
    cross_term = np.cross(omega, J @ omega)
    omega_dot = np.linalg.inv(J) @ (-cross_term + td + u)
    return omega_dot

# Esempio: Cubesat 1U
J_sat = np.diag([0.001, 0.001, 0.001]) # kg*m^2
omega_init = np.array([0.1, 0, 0])      # rad/s
u_control = np.array()
print(f"Accelerazione angolare: {compute_dynamics(J_sat,
omega_init, u_control)}")
```

4.2 Il Modello di Puntamento Inerziale (Inertial Pointing)

Il modello "inertial pointing" è il più semplice: il satellite deve mantenere un orientamento fisso rispetto alle stelle. Si assume che non ci siano ruote d'inerzia ($h_w = 0$) e che i disturbi siano trascurabili.

Linearizzazione

Per progettare un controllore lineare, espandiamo il sistema attorno al punto di equilibrio $\mathbf{q} \approx 0$ e $\omega_I \approx 0$. Il sistema linearizzato assume la forma nello spazio di stato $\dot{x} = Ax + Bu$, dove:

$$A = \begin{bmatrix} 0_3 & 0_3 & \frac{1}{2} I_3 & 0_3 \end{bmatrix}, \quad B = \begin{bmatrix} J^{-1} & 0_3 \end{bmatrix}$$

Questo modello è **completamente controllabile**, permettendo l'uso di tecniche come il posizionamento dei poli o l'LQR.

4.3 Modello Nadir Pointing con Momentum Bias

In molte missioni LEO (Low Earth Orbit), il satellite deve puntare verso la Terra (Nadir). In questo caso, il riferimento non è più inerziale, ma è il sistema **LVLH** (Local Vertical Local Horizontal).

Caratteristiche del Modello:

1. **Momentum Wheel**: Viene installata una ruota d'inerzia lungo l'asse Y_b per fornire stabilità giroscopica (momentum bias).
2. **Velocità Orbitale (ω_0)**: Il riferimento LVLH ruota rispetto allo spazio inerziale con velocità $\omega_0 = \sqrt{\mu/r^3}$.
3. **Gradiente di Gravità (t_{gg})**: È il disturbo dominante in orbita bassa, causato dalla differenza di attrazione gravitazionale tra le diverse parti del satellite.

Equazione Linearizzata Completa

Il sistema finale per un satellite Nadir-pointing considera le interazioni tra velocità angolare, quaternioni ridotti, momento della ruota (h_2) e gradiente di gravità. Le equazioni accoppiano il rollio (q_1) e l'imbardata (q_3), mentre il beccheggio (q_2) rimane spesso disaccoppiato.

Riepilogo Schematico - Cap. 4.2 & 4.3

- **Inertial Pointing**: Semplice, linearizzato attorno allo zero.
- **Nadir Pointing**: Include ω_0, coppie di gradiente di gravità e ruote d'inerzia.
- **Controllabilità**: Garantita dall'uso della parte vettoriale del quaternione.

Formule Fondamentali

1. Velocità orbitale: $\omega_0 = v/r = 2\pi/p$
2. Coppia Gradiente Gravità: $t_{gg} = \begin{bmatrix} 6\omega_0^2(J_{33} - J_{22})q_1 & 6\omega_0^2(J_{33} - J_{11})q_2 & 0 \end{bmatrix}$ (per piccoli angoli)

Applicazione Pratica (Python)

```
def gravity_gradient_torque(J, omega0, q):
    # Calcolo semplificato della coppia di gradiente di gravità
    tx = 6 * (omega0**2) * (J - J) * q
    ty = 6 * (omega0**2) * (J - J) * q
```

```
    return np.array([tx, ty, 0])

J_diag = np.diag() # Esempio satellite asimmetrico
w0 = 0.0011 # Rad/s (tipico LEO)
q_err = np.array([0.01, 0.01, 0]) # Errore di assetto
print(f"Coppia di gradiente di gravità:
{gravity_gradient_torque(J_diag, w0, q_err)} Nm")
```

Mappa Concettuale Globale

1. **Modellazione d'Assetto**
   - **Dinamica (Eulero)**: Legame tra Momento Angolare e Coppie (J, ω, u, t_d).
   - **Cinematica (Quaternioni)**: Descrizione dell'orientamento senza singolarità.
     - *Quaternione Pieno*: Non controllabile linearmente.
     - *Quaternione Ridotto (Vettoriale)*: Controllabile, singolarità solo a $\pm\pi$.
2. **Configurazioni di Missione**
   - **Inertial Pointing**: Riferimento fisso, modello lineare semplice.
   - **Nadir Pointing**: Riferimento LVLH, include velocità orbitale ω_0 e stabilità tramite ruote d'inerzia (Momentum Bias).
3. **Disturbi Ambientali**
   - **Gradiente di Gravità**: Fondamentale per il puntamento Nadir.

Formulario Completo

- **Matrice Antisimmetrica**:
  $S(\omega) = [0 \quad -\omega_3 \quad \omega_2 \, \omega_3 \quad 0 \quad -\omega_1 \, -\omega_2 \quad \omega_1 \quad 0]$
- **Derivata Quaternione (Vettoriale)**: $\dot{\mathbf{q}} = \frac{1}{2}(\sqrt{1 - \mathbf{q}^T\mathbf{q}}\mathbf{I}_3 + S(\mathbf{q}))\omega$
- **Stato Inerziale Linearizzato**: $\dot{\omega} = J^{-1}u$, $\dot{\mathbf{q}} = \frac{1}{2}\omega$
- **Termini Nadir-Pointing (Esempi f41, f52)**:
  - $f_{41} = 8(J_{33} - J_{22})\omega_0^2 + 2h_2\omega_0$
  - $f_{52} = 6(J_{33} - J_{11})\omega_0^2$

Guida di Ripasso Rapido

1. **Perché i quaternioni ridotti?** Perché eliminano la ridondanza matematica del quarto elemento, rendendo il sistema linearizzato "completamente controllabile" per gli algoritmi moderni di controllo.
2. **Qual è il limite dei quaternioni ridotti?** Presentano una singolarità a $\pm 180^\circ$ ($\alpha = \pm\pi$), ma è il punto più lontano possibile dall'equilibrio operativo (0°), rendendoli ottimi per l'uso pratico.
3. **Differenza tra Inertial e Nadir pointing:** L'Inertial pointing punta alle stelle (sistema statico); il Nadir pointing segue la curvatura terrestre, introducendo una rotazione costante ω_0 e richiedendo la gestione del gradiente di gravità.
4. **Il ruolo della Momentum Wheel:** Fornisce un "bias" di momento angolare che stabilizza il satellite contro i disturbi esterni attraverso l'effetto giroscopico.

5.1 Coppie Gravitazionali (Gravity Gradient)

La **coppia di gradiente di gravità** nasce dal fatto che un satellite non è un punto materiale, ma un corpo esteso. Le parti del satellite più vicine alla Terra subiscono un'attrazione gravitazionale leggermente maggiore rispetto alle parti più lontane.

Derivazione e Formulazione

Partendo dalla legge di gravitazione universale di Newton, la forza che agisce su un elemento di massa dm del satellite è:

$$d\mathbf{f} = -\frac{\mu dm}{|\mathbf{r}|^3}\mathbf{r}$$

Dove $\mu = Gm_{Terra}$ è la costante gravitazionale geocentrica ($\approx 3.986 \times 10^{14}, m^3/s^2$).

Attraverso un'espansione in serie di Taylor e l'integrazione su tutto il corpo del satellite, arriviamo alla formula fondamentale della coppia gravitazionale:

$$\mathbf{t}_g = \frac{3\mu}{|\mathbf{R}|^5}\mathbf{R} \times \mathbf{J}\mathbf{R}$$

Variabili fisiche:

- $\mathbf{t}_g$: Vettore coppia gravitazionale $[Nm]$.
- $\mathbf{R}$: Vettore posizione dal centro della Terra al centro di massa del satellite $[m]$.
- $\mathbf{J}$: Diade (o matrice) d'inerzia del satellite $[kg \cdot m^2]$.

- μ: Costante gravitazionale geocentrica.

Intuizione Ingegneristica

Fisicamente, questa coppia tende ad allineare l'asse di minima inerzia del satellite con la verticale locale (nadir). È un effetto "stabilizzante" naturale se sfruttato correttamente (stabilizzazione a gradiente di gravità), ma un disturbo se il puntamento desiderato è differente.

Esempio Applicativo: Il satellite canadese *Alouette 1* subì una precessione rapida a causa delle lunghe antenne che creavano una grande differenza nei momenti d'inerzia, amplificando l'effetto del gradiente di gravità.

Riepilogo e Formule Fondamentali

- **Concetto:** Differenza di forza gravitazionale su un corpo esteso.
- **Formula compatta:** $\mathbf{t}_g = \frac{3\mu}{R^3}(\mathbf{u}_R \times \mathbf{J}\mathbf{u}_R)$ (dove $\mathbf{u}_R$ è il versore radiale).
- **Modello linearizzato (per controllo):**

$$t_{gx} = 3\omega_0^2(J_{33} - J_{22})q_1$$

$$t_{gy} = 3\omega_0^2(J_{33} - J_{11})q_2$$

```
# Simulazione rapida Coppia Gradiente di Gravità
import numpy as np

def gravity_gradient_torque(mu, R_vec, J_matrix):
    R_mag = np.linalg.norm(R_vec)
    # Formula: 3*mu/R^5 * (R x J*R)
    torque = (3 * mu / R_mag**5) * np.cross(R_vec,
np.dot(J_matrix, R_vec))
    return torque

# Dati esempio: Orbita LEO (7000 km dal centro), Satellite
asimmetrico
mu_earth = 3.986e14
R = np.array([0, 0, 7000e3])
J = np.diag() # kg*m^2
print(f"Torque Gravitazionale: {gravity_gradient_torque(mu_earth,
R, J)} Nm")
```

5.2 Coppie Indotte dall'Atmosfera (Aerodinamiche)

Nelle orbite basse (LEO), l'atmosfera residua, seppur tenue, genera una forza d'urto contro le superfici del satellite.

Il Modello Fisico

La forza aerodinamica $\mathbf{f}$ su un elemento d'area dA è modellata come:

$$\mathbf{f} = -\rho V^2 [(2 - \sigma_n - \sigma_t)(\mathbf{e}_v \cdot \mathbf{e}_n)\mathbf{e}_n + \sigma_t(\mathbf{e}_v \cdot \mathbf{e}_n)\mathbf{e}_v] dA$$

Variabili:

- ρ: Densità atmosferica $[kg/m^3]$. Fondamentale perché varia esponenzialmente con l'altitudine e l'attività solare.
- V: Velocità del satellite $[\approx 7.5, km/s$ in LEO$]$.
- σ_n, σ_t: Coefficienti di scambio di quantità di moto (solitamente ≈ 0.8).

La coppia aerodinamica risultante è $\mathbf{t}_a = \mathbf{r} \times \mathbf{f}$, dove $\mathbf{r}$ è il braccio tra il centro di massa e il centro di pressione aerodinamica.

Intuizione Ingegneristica

Immaginate il satellite come una banderuola. Se il "centro di pressione" (dove agisce il vento atmosferico) non coincide con il "centro di massa", il satellite inizierà a ruotare. Questo disturbo è estremamente difficile da prevedere con precisione perché la densità ρ fluttua con i cicli solari e l'attività geomagnetica.

5.3 Coppie Indotte dal Campo Magnetico

Il satellite interagisce con il campo magnetico terrestre (geomagnetico) come l'ago di una bussola.

Formulazione

La coppia magnetica $\mathbf{t}_m$ è data dal prodotto vettoriale tra il momento magnetico residuo del satellite $\mathbf{m}$ e l'induzione magnetica terrestre $\mathbf{r}_m$:

$$\mathbf{t}_m = \mathbf{m} \times \mathbf{r}_m$$

- $\mathbf{m}$ $[A \cdot m^2]$: Deriva da correnti interne, magnetismo residuo dei materiali o magnetotorque (attuatori).
- $\mathbf{r}_m$ $[Wb/m^2$ o Tesla$]$: Il campo geomagnetico, spesso modellato tramite l'**IGRF (International Geomagnetic Reference Field)** che utilizza polinomi di Legendre e armoniche sferiche.

Calcolo e Trasformazioni

Per calcolare questa coppia nel sistema di riferimento del satellite (Body Frame), dobbiamo eseguire una serie di trasformazioni di coordinate:

1. **Algoritmo IGRF:** Calcola il campo in coordinate sferiche geocentriche.
2. **Trasformazione ECI:** Da sferiche a inerziali.
3. **Trasformazione Orbitale (RSW/LVLH):** Attraverso i parametri orbitali (inclinazione, ascensione retta).
4. **Trasformazione Body:** Utilizzando gli angoli di assetto (rollio, beccheggio, imbardata).

5.4 Coppie da Radiazione Solare

I fotoni provenienti dal Sole trasportano quantità di moto. Quando colpiscono le superfici del satellite, esercitano una pressione infinitesima ma costante.

Modello della Pressione Solare

La pressione solare media a 1 AU è $P \approx 4.563 \times 10^{-6}, N/m^2$. La forza dipende dalle proprietà ottiche della superficie:

- ρ_s: Riflessione speculare.
- ρ_d: Riflessione diffusa.
- ρ_a: Assorbimento.

La coppia è $\mathbf{t}_s = \mathbf{r} \times \mathbf{f}$, dove $\mathbf{r}$ è il vettore dal centro di massa al centro di pressione ottica. Sebbene piccola, questa coppia è la principale fonte di disturbo per satelliti in orbita geostazionaria (GEO), dove l'atmosfera è assente.

5.5 Coppie Interne

Generate da movimenti *dentro* il satellite:

- Deployment di pannelli solari o bracci robotici.
- Movimenti di astronauti (in stazioni spaziali).
- Perdite di gas dai propulsori (leaks). Queste sono spesso le coppie più intense, sebbene di breve durata, e devono essere simulate accuratamente per testare la robustezza dei controllori.

Sistema di Studio Finale

Mappa Concettuale Globale

1. **Disturbi Esterni:**
   - *Gravitazionali:* Funzione della distribuzione di massa (J). Dominanti in LEO/MEO.
   - *Aerodinamici:* Funzione di ρ e geometria. Dominanti in LEO basso (< 500 km).
   - *Magnetici:* Funzione del momento residuo m. Variano con la posizione orbitale.
   - *Radiativi (Sole):* Funzione delle proprietà ottiche. Dominanti in GEO.
2. **Disturbi Interni:** Meccanismi mobili e flussi di fluidi.

Formulario Completo

- **Gravità:** $\mathbf{t}_g = \frac{3\mu}{R^5}\mathbf{R} \times \mathbf{J}\mathbf{R}$
- **Atmosfera:** $\mathbf{f}_{aero} \propto \rho V^2 A$, $\mathbf{t}_a = \mathbf{r}_{cp} \times \mathbf{f}_{aero}$
- **Magnetismo:** $\mathbf{t}_m = \mathbf{m} \times \mathbf{B}$
- **Radiazione Solare:** $\mathbf{P}_{sol} \approx 4.56 \mu N/m^2$, $\mathbf{t}_s = \mathbf{r}_{opt} \times \mathbf{f}_{sol}$

Guida di Ripasso Rapido

1. **Perché studiamo i disturbi?** Perché introducono incertezze non modellate nel controller nominale; servono per test di simulazione ad alta fedeltà.
2. **Qual è il disturbo maggiore in LEO?** Solitamente la gravità (gradiente), seguita dalla coppia magnetica e aerodinamica (a seconda dell'altezza).
3. **Come si calcola il campo magnetico?** Usando il modello IGRF aggiornato ogni 5 anni e trasformando i vettori attraverso i riferimenti ECI -> Orbitale -> Body.
4. **Cosa influenza la coppia solare?** L'orientamento del Sole rispetto al satellite e la frazione di assorbimento/riflessione delle superfici.

Buongiorno. Sono il vostro professore di Ingegneria Aerospaziale e oggi approfondiremo uno dei pilastri fondamentali della navigazione spaziale: la **determinazione d'assetto (Attitude Determination)**.

Senza una conoscenza precisa dell'orientamento, un satellite non può puntare le sue antenne verso terra, i suoi pannelli solari verso il Sole o i suoi strumenti scientifici verso un obiettivo planetario. In questa lezione analizzeremo i metodi

analitici e numerici per risolvere questo problema, basandoci rigorosamente sulla documentazione tecnica a nostra disposizione.

Capitolo 6: Il Problema di Wahba e la Matrice K di Davenport

6.1 Inquadramento Fisico e Intuizione Ingegneristica

Il problema fondamentale consiste nel determinare la rotazione (espressa come matrice A o quaternione q) che porta un set di vettori noti in un sistema di riferimento (es. inerziale, r_i) a coincidere con le misure effettuate dai sensori di bordo nel sistema solidale al veicolo (body frame, b_i). I sensori tipici includono sensori solari, magnetometri o star tracker. Poiché la lunghezza dei vettori non aggiunge informazioni sull'orientamento, lavoriamo esclusivamente con **vettori unitari** (adimensionali).

6.2 Definizione Matematica: Il Problema di Wahba

Nel 1965, Wahba definì il problema come la ricerca della matrice di rotazione ottimale A che minimizza la funzione di costo dei minimi quadrati:

$$J(A) = \frac{1}{2} \sum_{i=1}^{m} a_i |b_i - A r_i|^2$$

Dove:

- b_i: Vettore misurato nel body frame (senza unità).
- r_i: Vettore di riferimento (effemeridi) (senza unità).
- a_i: Pesi associati alla confidenza del sensore ($\sum a_i = 1$).

6.3 La Soluzione di Davenport (q-method)

Davenport dimostrò che minimizzare $J(A)$ equivale a massimizzare la forma quadratica $\bar{q}^T K \bar{q}$. Il problema si trasforma nella ricerca dell'**autovettore corrispondente al massimo autovalore** della matrice K (4x4):

$$K = \begin{bmatrix} \sigma & z^T \\ z & H - \sigma I \end{bmatrix}$$

Variabili fisiche:

- $B = \sum a_i b_i r_i^T$: Matrice di covarianza incrociata.
- $\sigma = Tr(B)$: Traccia di B.
- $S = B + B^T$ (indicata anche come H nelle fonti).

- z: Vettore costruito dalle componenti antisimmetriche di B ($z = [B_{23} - B_{32}, B_{31} - B_{13}, B_{12} - B_{21}]^T$).

6.4 Algoritmi QUEST e FOMA

Il calcolo degli autovettori era computazionalmente oneroso negli anni '80.

- **QUEST (QUaternion ESTimator):** Utilizza il polinomio caratteristico di grado 4 e il metodo di Newton-Raphson per trovare l'autovalore massimo λ_{max}, partendo da un'ipotesi iniziale $\lambda \approx 1$.
- **FOMA:** Simile a QUEST, ma trova direttamente la matrice di rotazione in modo esplicito.

Riepilogo Schematico

- **Input:** Coppie di vettori (b_i, r_i) e pesi a_i.
- **Obiettivo:** Minimizzare l'errore di allineamento.
- **Metodo:** Trasformazione in problema di autovalori/autovettori (Matrice K).
- **Efficienza:** Newton-Raphson per evitare il calcolo simbolico pesante.

Formule Fondamentali

1. **Problema di Wahba:** $\min \frac{1}{2} \sum a_i |b_i - A r_i|^2$.
2. **Equazione agli autovalori:** $K\bar{q} = \lambda \bar{q}$.
3. **Matrice K:** Costruita da σ, z, H derivati dalla matrice B.

Applicazione Pratica (Python)

```
import numpy as np

def compute_k_matrix(b_vectors, r_vectors, weights):
    # b_vectors, r_vectors: liste di vettori 3x1
    B = np.zeros((3, 3))
    for b, r, a in zip(b_vectors, r_vectors, weights):
        B += a * np.outer(b, r)

    sigma = np.trace(B)
    S = B + B.T
    z = np.array([B-B, B-B, B-B])
```

```
    K = np.zeros((4, 4))
    K = sigma
    K[0, 1:] = z
    K[1:, 0] = z
    K[1:, 1:] = S - sigma * np.eye(3)
    return K

# Esempio: due sensori con pesi uguali
b = [np.array(), np.array()]
r = [np.array(), np.array()]
w = [0.5, 0.5]
K = compute_k_matrix(b, r, w)
eigenvalues, eigenvectors = np.linalg.eig(K)
q_opt = eigenvectors[:, np.argmax(eigenvalues)]
print(f"Quaternione ottimale: {q_opt}")
```

Capitolo 2: Soluzioni Analitiche e Metodi Numerici Avanzati

6.1 Soluzione Analitica per due vettori

Se disponiamo di soli due vettori (es. Sole e campo magnetico), esiste una soluzione chiusa estremamente efficiente (solo 37 operazioni in virgola mobile) che non richiede il calcolo di autovalori:

$$\bar{q} \propto (b_2 \cdot r_1 - b_1 \cdot r_2, (b_1 - r_1) \times (b_2 - r_2))$$

. Questa formula è ideale per sistemi embedded con limitata potenza di calcolo.

6.2 Metodo Riemann-Newton

Per aumentare la robustezza, si può ottimizzare direttamente sul **manifold sferico** (poiché il quaternione deve avere norma unitaria). Invece di cercare lungo linee rette, si cerca lungo le **geodetiche** della sfera. L'aggiornamento avviene tramite:

$$\bar{q}_{k+1} = \frac{\bar{q}_k + y_k}{|\bar{q}_k + y_k|}$$

.

6.3 Metodo SVD (Singular Value Decomposition)

Markley propose di risolvere il problema di Wahba direttamente per la matrice A usando la decomposizione ai valori singolari della matrice $C = RB^T$. Se $C = UDV^T$, allora la matrice di rotazione ottimale è:

$$A_{opt} = U\text{diag}(1, 1, \det(U)\det(V))V^T$$

Questo metodo è rinomato per la sua stabilità numerica.

6.4 Determinazione della velocità angolare (ω)

Se non disponiamo di giroscopi, possiamo stimare ω differenziando il quaternione nel tempo:

$$\omega = 2E\frac{d\bar{q}}{dt}$$

Dove E è una matrice costruita con le componenti del quaternione. Per eliminare il rumore della differenziazione, è necessario un filtro passa-basso (es. Butterworth).

Riepilogo Schematico

- **Casi speciali:** Per $m = 2$, formula analitica diretta.
- **Manifold:** L'ottimizzazione su sfera (Riemann) è più elegante e veloce.
- **SVD:** Metodo robusto alternativo al quaternione per ottenere la matrice A.
- **Dinamica:** Possibilità di derivare ω dai soli sensori vettoriali.

Formule Fondamentali

1. **Analitica (m=2):** $q \propto (\Delta\cdot, \Delta\times)$.
2. **SVD:** $A_{opt} = U^+V_+^T$.
3. **Velocità angolare:** $\omega = 2E\dot{\bar{q}}$.

Applicazione Pratica (Python: SVD Method)

```
def solve_wahba_svd(b_matrix, r_matrix):
    # b_matrix, r_matrix: matrici 3xm dove ogni colonna è un
vettore
    C = r_matrix @ b_matrix.T
    U, S, Vt = np.linalg.svd(C)
    d = np.linalg.det(U) * np.linalg.det(Vt)
```

```
    A_opt = U @ np.diag([1, 1, d]) @ Vt
    return A_opt
```

Mappa Concettuale Globale

1. **Ingresso Dati:** Misure Sensori (b_i) + Modelli Astronomici (r_i).
2. **Modellazione Errore:** Problema di Wahba (Minimi Quadrati).
3. **Percorso Matrice K (Davenport):**
   - *Analitico:* Polinomio di 4° grado (ESOQ).
   - *Iterativo:* QUEST / FOMA (Newton-Raphson).
   - *Geometrico:* Riemann-Newton (Ottimizzazione su sfera).
4. **Percorso Diretto:**
   - *Due vettori:* Formula analitica veloce.
   - *SVD:* Decomposizione ai valori singolari (Stabilità massima).
5. **Output Secondario:** Stima di ω tramite derivata temporale e filtraggio.

Formulario Completo

- **Wahba Cost:** $J(A) = 1 - Tr(AB^T)$.
- **K-Matrix Components:**
  - $B = \sum a_i b_i r_i^T$.
  - $\sigma = Tr(B)$.
  - $z = [B_{23} - B_{32}, B_{31} - B_{13}, B_{12} - B_{21}]^T$.
- **Quartic Polynomial (QUEST):** $\lambda^4 - (a+b)\lambda^2 - c\lambda + (ab + c\sigma - d) = 0$.
- **Two-Vector Identity:** $\bar{q} = \frac{(b_2 \cdot r_1 - b_1 \cdot r_2, (b_1 - r_1) \times (b_2 - r_2))}{\text{norma}}$.
- **Angular Rate:** $\omega = 2E\dot{\bar{q}}$ con E matrice 3x4.

Guida di Ripasso Rapido

1. Perché la determinazione d'assetto è fondamentale? Serve per il puntamento di antenne/strumenti e per calcolare l'errore da fornire agli attuatori nel loop di controllo.

2. Cosa si intende per "Vettori di Riferimento"? Sono vettori calcolati tramite effemeridi (Sole, stelle) o modelli di campo (IGRF per il magnetismo) nel sistema inerziale.

3. Qual è il vantaggio del metodo QUEST? È estremamente veloce perché riduce un problema di ottimizzazione a una ricerca di radice di un polinomio di 4° grado, spesso risolvibile con poche iterazioni di Newton.

4. Quando dovrei usare il metodo SVD? Quando la stabilità numerica è prioritaria rispetto alla velocità di calcolo o quando si preferisce lavorare direttamente con matrici di rotazione invece che con quaternioni.

5. Cosa fare se non si hanno giroscopi? Si può stimare la velocità angolare calcolando la variazione temporale dell'assetto ($\Delta q / \Delta t$) e applicando un filtro per rimuovere il rumore ad alta frequenza.

Parte I: Capitolo 7 - Misurazioni dei Vettori Astronomici

La determinazione dell'assetto si basa sul confronto tra **vettori di riferimento** (noti tramite modelli matematici o cataloghi) e **vettori misurati** dai sensori di bordo. In questo capitolo, utilizzeremo la notazione r_i per i vettori di riferimento (subscript o per oggetti astronomici, m per campo magnetico, s per il Sole) e b_i per i corrispondenti vettori misurati nel frame del corpo del satellite.

7.1 Sensori Stellari (Star Trackers)

Le stelle sono punti di riferimento ideali perché, proiettate sulla **sfera celeste**, hanno posizioni essenzialmente fisse.

- **Concetto Fisico:** Uno Star Tracker utilizza una camera CCD per rilevare le stelle nel suo campo di vista (FOV). Se l'assetto fosse perfettamente allineato con il frame LVLH, la direzione $-Z$ punterebbe verso un oggetto noto r_o.
- **Intuizione Ingegneristica:** Il sistema richiede un **catalogo stellare** (mappa di stelle abbastanza luminose e uniformemente distribuite) e algoritmi di identificazione dei pattern per passare dall'immagine di punti luminosi a un vettore misurato b_o.
- **Modalità di Funzionamento:**
  1. **Acquisizione Iniziale:** Identificazione "al buio" tramite riconoscimento di pattern.

2. **Tracking:** Molto più semplice, segue le stelle già identificate calcolando i centroidi sul piano focale.

7.2 Campo Magnetico Terrestre e Magnetometri

Il vettore del campo geomagnetico è ampiamente utilizzato, specialmente per satelliti in orbita bassa (LEO).

- **Modello Ephemeris (r_m):** Si basa sul modello **IGRF** (International Geomagnetic Reference Field), che descrive il potenziale scalare magnetico V tramite armoniche sferiche.
  - Formula: $V(r, \theta, \phi) = a \sum_{n=1}^{\infty} \sum_{m=0}^{n} (\frac{a}{r})^{n+1} P_n^m \cos(\theta) (g_n^m \cos(m\phi) + h_n^m \sin(m\phi))$.
  - Il vettore r_m si ottiene come $-\text{grad}(V)$.
- **Misurazione (b_m):** Il sensore standard è il **magnetometro flux-gate** (TAM), che fornisce il vettore nel frame del corpo senza necessità di elaborazioni complesse, data la bassa rumorosità dello strumento.

7.3 Vettore Sole e Sensori Solari

Il Sole fornisce un riferimento vettoriale molto forte e facilmente identificabile.

- **Geometria Ephemeris (r_s):** Il vettore Sole nel frame ECI dipende dalla longitudine eclittica (λ) e dall'inclinazione dell'eclittica (ϵ).
  - $r_s = [\cos(\lambda), \cos(\epsilon)\sin(\lambda), \sin(\epsilon)\sin(\lambda)]^T$.
  - Il calcolo richiede la data Giuliana (JD) per determinare la posizione temporale del Sole.
- **Sensori Solari Grossolani (CSS):** Misurano una corrente I_i proporzionale al coseno dell'angolo tra il vettore Sole b_s e la normale al sensore n_i.
  - Formula di misurazione: $b_i = I_i / I_0 = n_i \cdot b_s$.
- **Soluzione Ingegneristica:** Se sono disponibili almeno 3 sensori con correnti valide, si usa la **pseudo-inversa** per risolvere il sistema e ottenere il vettore unitario $\hat{b}_s$.

Esempio Applicativo: Calcolo del Vettore Sole in Python

Questo script calcola la posizione approssimativa del Sole dato il tempo universale (semplificato).

```
import numpy as np

def get_sun_vector(T_UT1):
```

```
    # Longitudine media L e anomalia media g (valori semplificati
da sorgenti)
    L = 280.460 + 36000.770 * T_UT1
    g = 357.527 + 35999.050 * T_UT1

    # Conversione in radianti per funzioni trig
    g_rad = np.radians(g)

    # Longitudine eclittica lambda
    lambda_sun = L + 1.9146 * np.sin(g_rad) + 0.0199 * np.sin(2 *
g_rad)
    lambda_rad = np.radians(lambda_sun)

    # Inclinazione epsilon (circa 23.44 gradi)
    epsilon = np.radians(23.439 - 0.013 * T_UT1)

    rs = np.array([
        np.cos(lambda_rad),
        np.cos(epsilon) * np.sin(lambda_rad),
        np.sin(epsilon) * np.sin(lambda_rad)
    ])
    return rs / np.linalg.norm(rs)

# Esempio: T_UT1 calcolato dalla data Giuliana
print("Vettore Sole (ECI):", get_sun_vector(0.24))
```

Riepilogo Schematico (Cap. 7)

- **Sensori:** Star Tracker (precisione massima), Magnetometro (semplicità/LEO), CSS (affidabilità/Sole).
- **Processo:** 1. Calcolo r_i (riferimento); 2. Misura b_i (sensore); 3. Algoritmi di determinazione (es. TRIAD o QUEST, non dettagliati qui ma impliciti).

Formule Fondamentali

1. **Vettore Sole ECI:** $r_s = [\cos\lambda, \cos\epsilon\sin\lambda, \sin\epsilon\sin\lambda]^T$.
2. **Misura CSS:** $b_i = n_i \cdot b_s$.

Parte II: Capitolo 8 - Stima dell'Assetto del Veicolo Spaziale

Passiamo dalla semplice determinazione (statica) alla **stima** (dinamica), che permette di gestire il rumore dei sensori e le incertezze del modello.

8.1 Background Probabilistico

La stima si basa sui processi di **Gauss-Markov**, dove lo stato futuro dipende solo dal presente (proprietà di Markov) e gli errori seguono una distribuzione normale (Gaussiana).

- **Rumore Bianco:** Una sequenza X_k con media zero e covarianza $Cov(X_i, X_j) = R_i \delta_{ij}$ (incorrelata nel tempo).

8.2 Filtro di Kalman Lineare (LKF)

È lo strumento ottimale per sistemi lineari con rumore bianco. Si basa sulla **proiezione ortogonale** per minimizzare la covarianza dell'errore.

- **Logica Predict-Correct:**
  1. **Predizione:** Usa il modello fisico per prevedere lo stato $\hat{x}_{k+1|k}$.
  2. **Aggiornamento (Update):** Usa la misura y_{k+1} per correggere la predizione tramite il **Guadagno di Kalman** K.

8.3 Filtro di Kalman Esteso (EKF)

Poiché la cinematica dei quaternioni e la dinamica spaziale sono intrinsecamente **non lineari**, usiamo l'EKF.

- **Metodo:** Il sistema viene linearizzato tramite l'espansione di Taylor (Jacobiane) attorno alla stima corrente.
- **Jacobiane Fondamentali:** F_{k-1} (matrice di transizione di stato) e H_k (matrice di osservazione).

8.4 EKF per lo Stato del Veicolo Spaziale

Un approccio moderno include sia la **cinematica** (quaternioni) che la **dinamica** (equazioni di Eulero).

- **Vantaggi:** L'uso della dinamica permette di stimare l'assetto anche in caso di rumore eccessivo nei giroscopi o in loro assenza.

- **Modello a Quaternione Ridotto:** Invece del quaternione completo (4 componenti), si usano le 3 componenti vettoriali $q = [q_1, q_2, q_3]^T$. Questo evita il vincolo di normalizzazione unitaria e semplifica il calcolo.

Esempio Applicativo: Iterazione Filtro di Kalman in Python

Simulazione semplificata di un passo di aggiornamento dello stato.

```
import numpy as np

# Parametri iniziali
x_hat = np.array([0.1, 0.0, 0.0]) # Stato (es. velocità angolare)
P = np.eye(3) * 0.1               # Covarianza errore
Q = np.eye(3) * 0.01              # Rumore di processo
R = np.eye(3) * 0.05              # Rumore di misura
H = np.eye(3)                     # Matrice di osservazione

def kalman_step(x_prev, P_prev, measurement):
    # 1. Predizione (modello identità semplificato)
    x_pred = x_prev
    P_pred = P_prev + Q

    # 2. Guadagno di Kalman
    S = H @ P_pred @ H.T + R
    K = P_pred @ H.T @ np.linalg.inv(S)

    # 3. Aggiornamento
    x_new = x_pred + K @ (measurement - H @ x_pred)
    P_new = (np.eye(len(x_prev)) - K @ H) @ P_pred

    return x_new, P_new

# Una misura arriva
z = np.array([0.12, 0.01, -0.01])
x_hat, P = kalman_step(x_hat, P, z)
print("Stato aggiornato:", x_hat)
```

Riepilogo Schematico (Cap. 8)

- **Filosofia:** Combinare conoscenza del modello (fisica) e dati sensoriali (osservazione).

- **Componenti:** Vettore di stato (velocità angolare ω, quaternione q, drift giroscopico β).
- **Robustezza:** L'EKF è robusto agli errori di modellazione, come l'incertezza sulla matrice d'inerzia J.

Formule Fondamentali

1. **Equazione di Stato Discreta:** $x_{k+1} = Ax_k + Bu_k + w_k$.
2. **Guadagno di Kalman:** $K_{k+1} = P_{k+1|k}C^T(CP_{k+1|k}C^T + R)^{-1}$.
3. **Aggiornamento Covarianza (Forma di Joseph):** $P_{k|k} = (I - KH)P_{k|k-1}(I - KH)^T + KRK^T$.

Materiale Conclusivo del Corso

Mappa Concettuale Globale

1. **INPUT (Sorgenti Dati):**
   - *Modelli:* IGRF (Campo Magnetico), Ephemeris (Sole), Cataloghi (Stelle).
   - *Sensori:* Magnetometri, Sensori Solari, Star Trackers, Giroscopi.
2. **ELABORAZIONE (Algoritmi):**
   - *Linearizzazione:* Jacobiane per sistemi non lineari.
   - *Filtro di Kalman:* Predizione (Dinamica/Cinematica) $\rightarrow$ Guadagno $\rightarrow$ Correzione (Misura).
3. **OUTPUT (Stima):**
   - Assetto (Quaternione), Velocità Angolare, Bias dei sensori.

Formulario Completo

- **Data Giuliana (JD):** Calcolo basato su anno, mese, giorno, ore, minuti, secondi.
- **Dinamica Spaziale:** $\dot{\omega} = -J^{-1}\omega \times (J\omega) + J^{-1}u + \phi_1$.
- **Cinematica Quaternioni:** $\dot{q} = \frac{1}{2}\Omega(\omega)$.
- **Predizione Covarianza:** $P_{k+1|k} = A_k P_{k|k} A_k^T + Q_k$.
- **Aggiornamento Stato:** $\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - C\hat{x}_{k+1|k})$.

Guida di Ripasso Rapido

1. **Differenza tra Determinazione e Stima:** La prima è istantanea e sensibile al rumore; la seconda usa la storia temporale e il modello fisico per

"pulire" il dato.

2. **Perché il Quaternione Ridotto?** Evita che il filtro debba forzare il quaternione ad avere norma 1 dopo ogni calcolo, riducendo la complessità computazionale.
3. **Il ruolo della Matrice d'Inerzia (J):** Essenziale nell'EKF che include la dinamica. Permette di prevedere come ruoterà il satellite in base ai momenti applicati.
4. **Stabilità Numerica:** In contesti reali, si preferisce la forma di Joseph o i filtri Square Root per garantire che la matrice di covarianza P rimanga definita positiva.

Parte 1: Controllo LQR per Veicoli Spaziali Nadir-Pointing e Inertial-Pointing

1.1 Introduzione e Modellizzazione

Il controllo d'assetto mira a mantenere il veicolo spaziale in un orientamento specifico (puntamento verso la Terra o Nadir, oppure verso un punto fisso nello spazio o Inerziale). Storicamente, i modelli basati sui quaternioni completi presentavano problemi di **controllabilità**. Tuttavia, le fonti dimostrano che un **modello a quaternioni ridotto** (usando solo le componenti vettoriali) è pienamente controllabile e presenta una struttura lineare semplice, ideale per il design moderno.

1.2 Il Regolatore Quadratico Lineare (LQR)

Il cuore della strategia proposta è l'**LQR**, che cerca di minimizzare una funzione di costo L definita come:

$$L = \frac{1}{2} \int_0^\infty (x^T Q x + u^T R u) dt$$

Definizione delle Variabili:

- x: Vettore di stato (include velocità angolari ω e componenti del quaternione q). Unità: $[\mathrm{rad/s}]$ e $[dimensionale]$.
- u: Vettore dei momenti di controllo. Unità: $[\mathrm{N \cdot m}]$.
- Q: Matrice di peso dello stato (costo dell'errore di assetto). Deve essere definita positiva.
- R: Matrice di peso del controllo (costo dell'energia consumata). Deve essere definita positiva.

Intuizione Ingegneristica: L'LQR permette di bilanciare la precisione del puntamento (tramite Q) con il risparmio di propellente o energia elettrica per gli attuatori (tramite R).

1.3 Soluzione Analitica per il Puntamento Inerziale

Per un satellite a puntamento inerziale con matrice d'inerzia J diagonale, le fonti forniscono una soluzione analitica elegante per l'equazione di Riccati, evitando calcoli numerici complessi. La legge di controllo risultante è $u = -[D, K]x$, dove D e K sono matrici di guadagno diagonali:

- **Guadagno di smorzamento (d_i):** $d_i = \sqrt{\frac{q_{1i}}{r_i} + J_{ii}\sqrt{\frac{q_{2i}}{r_i}}}$.
- **Guadagno di rigidezza (k_i):** $k_i = \sqrt{\frac{q_{2i}}{r_i}}$.

Interpretazione Fisica: Il controllo agisce come un sistema massa-molla-smorzatore virtuale. k_i definisce la "forza" di richiamo verso l'assetto desiderato, mentre d_i assicura che le oscillazioni si smorzino.

1.4 Stabilità Globale

Utilizzando una funzione di Lyapunov V, le fonti dimostrano che, sebbene il controller sia progettato su un modello linearizzato, esso garantisce la **stabilità asintotica globale** del sistema non lineare originale, a patto di rispettare vincoli specifici sulle matrici di peso (es. $R = cQ_{22}$ o $R = cQ_{22}J$).

Riepilogo Schematico (Cap. 9.1 - 9.2)

- **Modello:** Quaternioni ridotti (controllabile).
- **Obiettivo:** Minimizzare errore e consumo energetico.
- **Stabilità:** Garantita globalmente per il sistema non lineare.
- **Struttura Gain:** Diagonale (disaccoppiamento degli assi).

Elenco Formule Fondamentali

1. **Equazione di Riccati:** $-FA - A^TF + FBR^{-1}B^TF - Q = 0$.
2. **Legge di Controllo:** $u = -R^{-1}B^TFx = -Gx$.
3. **Autovalori ad anello chiuso (λ):** $\lambda_i = \frac{-s_i \pm \sqrt{s_i^2 - 2t_i}}{2}$.

Applicazione Pratica in Python

```
import numpy as np

def calculate_lqr_gains(J_diag, q1, q2, r):
    """
    Calcola i guadagni analitici D e K per un satellite a
puntamento inerziale.
    J_diag: lista delle inerzie principali [J11, J22, J33]
    q1, q2, r: pesi diagonali per Q e R
    """
    D = np.sqrt(q1/r + np.array(J_diag) * np.sqrt(q2/r))
    K = np.sqrt(q2/r) * np.ones_like(J_diag)
    return D, K

# Esempio basato su fonte
J =
q_val, r_val = 5, 8
D_gains, K_gains = calculate_lqr_gains(J, q_val, q_val, r_val)

print(f"Guadagni di smorzamento (D): {D_gains}")
print(f"Guadagni di rigidezza (K): {K_gains}")
```

Parte 2: Assegnazione dei Poli Robusta e Reiezione dei Disturbi

2.1 Robustezza del Sistema

Un problema critico nel controllo spaziale è l'incertezza del modello (es. variazioni nei momenti d'inerzia J). Le fonti introducono il concetto di **Assegnazione dei Poli Robusta**. Un sistema è robusto se piccoli errori di modellizzazione non spostano gli autovalori verso il semipiano destro (instabilità).

2.2 Il Criterio dei Vettori Propri

La robustezza si misura tramite il determinante della matrice dei vettori propri X.

- **Obiettivo:** Massimizzare $|\det(X)|$.
- **Significato Geometrico:** Più i vettori propri sono vicini all'ortogonalità, più il sistema è insensibile alle perturbazioni dei parametri.

Le fonti dimostrano un risultato teorico fondamentale: **il design LQR per il sistema a quaternioni ridotto è intrinsecamente un design di assegnazione dei poli robusta**.

2.3 Reiezione dei Disturbi

Massimizzare la robustezza (minimizzando il numero di condizionamento $\kappa_2 = |X||X^{-1}|$) riduce anche l'impatto delle coppie di disturbo t_d (es. pressione solare, gradiente di gravità) sull'output del sistema.

2.4 Confronto: Quaternioni vs. Angoli di Euler

In simulazioni Monte Carlo, il controller basato su quaternioni ha mostrato una superiorità netta:

- **Quaternioni:** 300 successi su 300 test con grandi errori iniziali e perturbazioni d'inerzia.
- **Angoli di Euler:** Solo 132 successi su 300 nelle stesse condizioni.
  Intuizione: Gli angoli di Euler soffrono di singolarità e non sono adatti per manovre ad ampio angolo.

Riepilogo Schematico (Cap. 9.3)

- **Robustezza:** Misurata tramite il condizionamento dei vettori propri.
- **LQR = Robust:** La struttura diagonale del controllo LQR garantisce la massima robustezza possibile per questo modello.
- **Performance:** Eccellente reiezione dei disturbi e gestione di manovre "Large-Angle".

Elenco Formule Fondamentali

1. **Sottospazio dei Vettori Propri (S_i):** $S_i = x : (A - \lambda I)x \in R_c(B)$.
2. **Relazione Poli/Guadagni:** $k_i = 2\omega_{in}^2 J_{ii}$ e $d_i = 2\zeta_i \omega_{in} J_{ii}$.
3. **Tempo di Assestamento (T_s):** $T_s \approx \frac{4}{\zeta \omega_n}$.

Applicazione Pratica in Python

```
def check_robustness(X):
    """
    Calcola l'indice di robustezza basato sul determinante dei
```

```
vettori propri.
    """
    det_X = np.linalg.det(X)
    cond_X = np.linalg.cond(X)
    return abs(det_X), cond_X

# Esempio concettuale di matrice di vettori propri
X_example = np.eye(6) # Caso ideale di massima robustezza
score, cond = check_robustness(X_example)
print(f"Punteggio Robustezza (Determinante): {score}, Numero
Condizionamento: {cond}")
```

Materiale di Chiusura dell'Intero Corso

Mappa Concettuale Globale

1. **Modellizzazione:** Satellite $\rightarrow$ Quaternioni Ridotti $\rightarrow$ Sistema Lineare Controllabile.
2. **Sintesi Controllo:**
   - **LQR:** Ottimizzazione Costo/Energia $\rightarrow$ Soluzione Analitica Diagonale.
   - **Pole Placement:** Scelta frequenza naturale (ω_n) e smorzamento (ζ).
3. **Verifica:**
   - **Stabilità di Lyapunov:** Passaggio dal lineare al non lineare globale.
   - **Robustezza:** Massimizzazione $|\det(X)| \rightarrow$ Insensibilità a errori J.
4. **Applicazione:** Reiezione disturbi e manovre Large-Angle.

Formulario Completo

- **Dinamica Lineare:** $\dot{x} = Ax + Bu$.
- **Guadagno Ottimo:** $G = [diag(d_i), diag(k_i)]$.
- **Poli desiderati:** $\lambda = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$.
- **Vincolo Global Stability:** $R = cQ_{22}$.

Guida di Ripasso Rapido

1. **Perché usare i quaternioni ridotti?** Perché eliminano la ridondanza del quaternione completo (che rende il sistema non controllabile) e le singolarità degli angoli di Euler.

2. **Qual è il vantaggio della soluzione analitica LQR?** Permette di progettare il controllo direttamente dai parametri fisici (J_{ii}) e dai requisiti di missione (ω_n, ζ) senza risolvere numericamente Riccati ad ogni iterazione.
3. **Come garantisco che il mio satellite sia stabile anche per grandi rotazioni?** Assicurandosi che la matrice di peso R sia proporzionale alla matrice di peso dello stato Q_{22} (e opzionalmente all'inerzia J), garantendo una funzione di Lyapunov sempre decrescente.
4. **Cosa significa che il design è "robusto"?** Significa che se il momento d'inerzia reale del satellite differisce da quello stimato del 10%, il sistema rimarrà comunque stabile e le prestazioni non degraderanno significativamente.
5. **Criterio di progetto pratico:** Se serve un tempo di assestamento $T_s < 10s$, imposta $\zeta\omega_n = 0.4$ e calcola i guadagni d_i, k_i di conseguenza.

Modulo 1: Attuatori per Veicoli Spaziali (Cap. 10)

Gli attuatori sono i muscoli del sistema di controllo d'assetto (ACS); essi generano le coppie necessarie per orientare il satellite secondo le specifiche di missione.

10.1 Reaction Wheels e Momentum Wheels

Entrambi utilizzano un volano (flywheel) azionato da motori elettrici.

- **Reaction Wheel (Ruota di Reazione):** Viene accelerata o decelerata per creare una coppia che compensi i disturbi o ruoti il satellite.
- **Momentum Wheel (Ruota d'Inerzia):** Ruota costantemente ad alta velocità per fornire un "bias di momento", rendendo il satellite resistente ai cambiamenti d'assetto (stabilizzazione giroscopica).

Equazione Fondamentale: La coppia u è generata dalla variazione del momento angolare del volano:

$$u = -\dot{h}_w = -J_w\dot{\omega}$$

- h_w [kg·m²/s]: Vettore momento angolare del volano.
- J_w [kg·m²]: Momento d'inerzia del volano lungo l'asse di rotazione.
- $\dot{\omega}$ [rad/s²]: Accelerazione angolare del volano.

Intuizione Ingegneristica: Una ruota non può accelerare all'infinito (limite di saturazione). Quando raggiunge la velocità massima, è necessaria una "gestione del momento" tramite altri attuatori (magnetici o thruster) per scaricare l'eccesso di velocità.

10.2 Control Moment Gyros (CMG)

A differenza delle ruote di reazione, i CMG mantengono la velocità del volano costante ma ne cambiano l'orientamento tramite un giunto cardanico (**gimbal**).

Equazione della Coppia:

$$t_c = (g \times h)\omega_g$$

- g: Vettore unitario dell'asse del gimbal.
- h: Momento angolare del volano.
- ω_g [rad/s]: Velocità di rotazione del gimbal.

Interpretazione Fisica: Il grande vantaggio è l'**amplificazione della coppia**: una piccola velocità del gimbal produce una coppia d'uscita ortogonale molto elevata. Tuttavia, soffrono di **singolarità del gimbal** dove la coppia desiderata diventa irrealizzabile.

10.3 Magnetic Torque Rods (Magnetotorquer)

Sono bobine che interagiscono con il campo magnetico terrestre creando un dipolo magnetico.

- **Vantaggi:** Leggeri, affidabili (nessuna parte mobile), alimentati da pannelli solari.
- **Svantaggi:** La coppia t_m è limitata: $t_m = m \times r_m$.
  - m [A·m²]: Momento magnetico creato dalla bobina.
  - r_m [Tesla]: Intensità del campo magnetico terrestre.

Vincolo Fisico: Non è possibile generare coppia parallelamente al campo magnetico terrestre, rendendo il controllo su 3 assi impossibile in ogni istante con soli magnetotorquer.

10.4 Thrusters (Propulsori)

Generano forza tramite l'espulsione di propellente. **Equazione della Spinta (F):**

$$F = V_e \frac{dm}{dt} + A_e(P_e - P_a)$$

- V_e [m/s]: Velocità di scarico relativa.
- dm/dt [kg/s]: Tasso di consumo del carburante.
- I_{sp} [s]: Impulso specifico, definisce l'efficienza: $I_{sp} = \frac{F}{\dot{m} g_0}$.

Riepilogo Schematico - Capitolo 10

1. **Scambio di Momento:** Ruote (variazione velocità) vs CMG (variazione direzione).
2. **Attuatori Magnetici:** Solo per orbite basse (LEO); ideali per il desaturamento.
3. **Thruster:** Coppie elevate, ma limitati dal budget di carburante (propellente).

Formule Fondamentali

1. Coppia Ruota: $u = -J_w \dot{\omega}$
2. Coppia CMG: $t_c = (g \times h)\omega_g$
3. Coppia Magnetica: $t_m = m \times r_m$
4. Efficienza Propulsiva: $I_{sp} = \frac{F}{\dot{m} g_0}$

Applicazione Pratica Python

```
import numpy as np

def reaction_wheel_torque(J_w, alpha):
    """Calcola la coppia prodotta da una ruota di reazione."""
    # alpha è l'accelerazione angolare in rad/s^2
    return -J_w * alpha

def magnetic_torque(m_dipole, b_field):
    """Calcola la coppia magnetica generata (prodotto
vettoriale)."""
    return np.cross(m_dipole, b_field)

# Esempio: Ruota con inerzia 0.05 kg*m^2 accelerata a 2 rad/s^2
u = reaction_wheel_torque(0.05, 2.0)
print(f"Coppia generata: {u} Nm")
```

Modulo 2: Controllo d'Assetto tramite Coppie Magnetiche (Cap. 11)

Il controllo magnetico è complesso perché il sistema è intrinsecamente **tempo-variante (LTV)** a causa del moto del satellite lungo l'orbita.

11.1 Modello Lineare Tempo-Variante

Per un satellite puntato al Nadir, usiamo il **modello a quaternioni ridotti** per evitare singolarità e ridondanze. L'equazione di stato linearizzata è:

$$\dot{x} = Ax + B(t)m$$

Il campo magnetico $b(t)$ varia periodicamente lungo l'orbita secondo l'inclinazione i_m.

11.2 Controllabilità

Teorema di Bhat/Theorem 11.2: L'assetto è completamente controllabile tramite coppie magnetiche **se e solo se** l'orbita non è sul piano equatoriale magnetico ($i_m \neq 0$) e le matrici d'inerzia del satellite rispettano specifiche disuguaglianze (es. $2J_{11} + J_{33} \neq J_{22}$). **Intuizione:** Poiché il campo magnetico cambia direzione lungo l'orbita, lo spazio dei sottospazi controllabili cambia, permettendo nel tempo la stabilizzazione completa.

11.3 Design LQR Periodico

Poiché il sistema è periodico (periodo orbitale T), un controllo feedback costante è inefficiente. Si utilizza l'equazione di **Riccati Periodica**. L'algoritmo ottimizzato proposto nelle fonti utilizza la **decomposizione di Schur** su matrici simplettiche per calcolare la soluzione P_k in modo computazionalmente efficiente.

11.4 Controllo Combinato Assetto e Desaturazione

Invece di separare il controllo d'assetto dalla gestione del momento delle ruote, si può progettare un unico controllore LQR che gestisca entrambi simultaneamente.

- **Stati:** Quaternioni, velocità angolare del corpo e velocità delle ruote di reazione.
- **Obiettivo:** Minimizzare l'errore d'assetto e riportare la velocità delle ruote vicino allo zero (desaturazione) usando i magnetotorquer.

Riepilogo Schematico - Capitolo 11

1. **LTV:** Il sistema magnetico dipende dalla posizione orbitale.
2. **Controllabilità:** Possibile solo se l'orbita è inclinata rispetto all'equatore magnetico.

3. **LQR Periodico:** Risolve il problema dell'ottimizzazione per sistemi che si ripetono ogni orbita.

Formule Fondamentali

1. Modello LTV: $\dot{x} = Ax + B(t)m$
2. Condizione Inerziale: $2J_{11} + J_{33} \neq J_{22}$
3. Controllo Ottimo (LQR): $u_k = -K_k x_k$ (dove K_k è periodico)

Applicazione Pratica Python

```
# Simulazione concettuale di un sistema LTV Periodico
def get_periodic_B(t, orbit_period):
    """Esempio di variazione periodica della matrice B."""
    val = np.sin(2 * np.pi * t / orbit_period)
    return np.array([[0, val], [-val, 0]]) # Semplificato

# In una missione reale, useremmo l'algoritmo di Schur
# per calcolare il guadagno K[t] basato sulla matrice P[t] di
Riccati.
```

Materiale di Chiusura

Mappa Concettuale Globale

- **Obiettivo:** Controllo d'Assetto (Attitude Control).
  - **Mezzi (Attuatori):**
    - *Interni (Scambio Momento):* Ruote di Reazione (Coppie piccole/precise), CMG (Coppie elevate/complesse).
    - *Esterni (Coppie Ambientali/Massa):* Magnetotorquer (Soli LEO, per desaturazione), Thruster (Spostamenti rapidi, dispendio massa).
  - **Metodologie di Controllo:**
    - *Linearizzazione:* Modello a quaternioni ridotti.
    - *Ottimizzazione:* LQR Periodico (per gestire la natura tempo-variante del campo magnetico).

Formulario Completo

1. **Coppia Volano:** $u = -J_w \dot{\omega}$
2. **Coppia CMG:** $t_c = (\hat{g} \times h)\omega_g$

3. **Coppia Magnetica:** $t_m = m \times r_m$
4. **Spinta Propulsore:** $F = V_e \dot{m} + A_e(P_e - P_a)$
5. **Dinamica Linearizzata (Magnetico):** $\dot{x} = Ax + B(t)m$
6. **Guadagno LQR:** $K_k = (R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k$

Guida di Ripasso Rapido (Essentials)

- **Ruote vs CMG:** Ricorda che le ruote cambiano la *magnitudo* del momento angolare (accelerano), mentre i CMG cambiano la *direzione* (ruotano l'asse).
- **Il problema della Saturazione:** Le ruote accumulano momento a causa dei disturbi esterni. Senza un attuatore esterno (come i magnetotorquer), la ruota raggiungerebbe la velocità massima e non potrebbe più fornire coppia.
- **Perché i Quaternioni Ridotti?** Evitano la ridondanza del quaternione a 4 componenti mantenendo la linearità necessaria per il design LQR.
- **Il ruolo del Campo Magnetico:** Non è costante. Un satellite che "vede" un campo magnetico in una direzione a $t = 0$ ne vedrà una diversa a $t = T/4$. Questa variazione è ciò che rende il sistema completamente controllabile su 3 assi nel tempo.
- **Lifting Method:** È una tecnica matematica per trasformare un sistema periodico complicato in un sistema tempo-invariante aumentato, rendendo la soluzione di Riccati molto più veloce da calcolare al computer.

Questo sistema di studio vi fornisce le basi matematiche e fisiche per comprendere come vengono mantenuti in assetto i satelliti moderni. Studiate le derivazioni delle matrici simplettiche per l'esame, poiché sono fondamentali per gli algoritmi di controllo ottimo.

Capitolo 12: Manovra di Assetto e Innalzamento dell'Orbita

12.1 Manovra di Assetto (Attitude Maneuver)

Concetti e Intuizione Ingegneristica

Una manovra di assetto è il processo di riorientamento di un veicolo spaziale da una configurazione iniziale a una desiderata. L'intuizione fondamentale risiede nel fatto che, sebbene esistano diverse rappresentazioni (angoli di Euler, matrici di coseni direttori), il controllo basato sui **quaternioni** è superiore. Questo perché i quaternioni non presentano singolarità matematiche (come il *gimbal lock* degli angoli di Euler) e richiedono un carico computazionale inferiore.

Prendiamo come esempio il satellite **Orbview-2**. Prima dell'innalzamento dell'orbita, esso è stabilizzato con puntamento verso il basso (*nadir*). Per attivare i motori, deve ruotare di **90° attorno all'asse y** affinché i propulsori siano allineati al vettore velocità.

Modellazione Matematica

Sia $\bar{q} = (q_0, q_1, q_2, q_3)$ il quaternione di assetto attuale e $\bar{p} = (p_0, p_1, p_2, p_3)$ quello desiderato. L'errore di assetto $\bar{r}$ è definito come la rotazione relativa necessaria per passare dallo stato attuale a quello target:

$$\bar{r} = \bar{p}^{-1} \otimes \bar{q} = \bar{p}^{*} \otimes \bar{q}$$

Espresso in forma matriciale, l'errore $\bar{r}$ si calcola come:

$$[r_0\ r_1\ r_2\ r_3] = [p_0 \quad p_1 \quad p_2 \quad p_3 \quad -p_1 \quad p_0 \quad p_3 \quad -p_2 \quad -p_2 \quad -p_3 \quad p_0 \quad p_1 \quad -p_3 \quad p_2 \quad -p_1 \quad p_0]$$

Il **controllore PD** (Proporzionale-Derivativo) standard per questa manovra è:

$$\mathbf{u} = -K\mathbf{r} - D\boldsymbol{\omega}$$

Dove:

- $\mathbf{u}$: Coppia di controllo [N·m].
- $\mathbf{r}$: Parte vettoriale del quaternione d'errore (r_1, r_2, r_3) [adimensionale].
- $\boldsymbol{\omega}$: Velocità angolare del corpo [rad/s].
- K, D: Matrici dei guadagni positivi [costanti di progetto].

Esempio Applicativo e Simulazione Python

Se il satellite deve ruotare di 90° attorno a y, partendo da un assetto nullo $\bar{q} = (1, 0, 0, 0)$, il target sarà $\bar{p} = (\cos(\pi/4), 0, \sin(\pi/4), 0)$.

```
import numpy as np

def compute_quaternion_error(p, q):
    # p: target quaternion [p0, p1, p2, p3]
    # q: current quaternion [q0, q1, q2, q3]
    matrix = np.array([
        [p,  p,  p,  p],
        [-p, p,  p, -p],
        [-p, -p, p,  p],
        [-p, p, -p,  p]
    ])
    return np.dot(matrix, q)

# Esempio Orbview-2: Rotazione 90° asse Y
```

```
q_init = np.array()
p_target = np.array([np.cos(np.pi/4), 0, np.sin(np.pi/4), 0])
r_error = compute_quaternion_error(p_target, q_init)
print(f"Errore Quaternione r: {r_error}")
```

Riepilogo Schematico 12.1

- **Obiettivo:** Riorientamento spaziale privo di singolarità.
- **Vantaggio:** I quaternioni sono più efficienti e robusti degli angoli di Euler.
- **Logica:** Calcolo dell'errore tramite prodotto tensoriale $\bar{p}^* \otimes \bar{q}$ e applicazione di coppia correttiva proporzionale all'errore e allo smorzamento.

Formule Fondamentali 12.1

1. **Errore Quaternione:** $\bar{r} = \bar{p}^* \otimes \bar{q}$
2. **Legge di Controllo PD:** $\mathbf{u} = -K\mathbf{r} - D\boldsymbol{\omega}$

12.2 Innalzamento dell'Orbita (Orbit-raising)

Concetti e Intuizione Ingegneristica

L'innalzamento d'orbita (es. da 300 km a 705 km) richiede l'uso di propulsori (*thrusters*). La sfida ingegneristica è mantenere l'assetto corretto mentre i motori spingono, poiché eventuali disallineamenti creano coppie di disturbo che devono essere compensate dai propulsori stessi o da ruote di reazione.

Orbview-2 utilizza una **ruota a momento** (momentum wheel) allineata all'asse -y per la stabilità giroscopica e quattro propulsori montati sulla faccia anti-nadir, inclinati di 5° per generare coppie di controllo su tutti gli assi.

Dinamica del Veicolo Spaziale

L'equazione del momento angolare totale $\mathbf{h}$ è:

$$\dot{\mathbf{h}} = J\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \mathbf{h} + \mathbf{m}$$

Dove:

- J: Matrice d'inerzia diagonale $[J_x, J_y, J_z]$ [kg·m²].
- $\mathbf{h} = [J_x\omega_x, J_y\omega_y + h_w, J_z\omega_z]^T$ con h_w momento della ruota [N·m·s].
- $\mathbf{m}$: Coppie di controllo generate dai propulsori [N·m].

Le coppie $\mathbf{m}$ sono legate ai livelli di spinta dei singoli propulsori T_i [N] tramite la geometria dei bracci di leva R e le direzioni delle forze F:

$$\mathbf{m} = \sum (\mathbf{r}_i \times \mathbf{f}_i) T_i$$

Controllo LQR e Linearizzazione

Per progettare un controllo digitale, il sistema viene linearizzato e discretizzato. Si aggiungono **stati integrativi** $\mathbf{i}_q$ per eliminare l'errore a regime stazionario, tipico dei controllori PID. Il sistema assume la forma:

$$\mathbf{x}_9(n+1) = \Phi \mathbf{x}_9(n) + \Gamma \mathbf{u}(n)$$

Il controllo ottimale **LQR** minimizza una funzione di costo J bilanciando precisione dell'assetto e consumo di propellente.

```
# Definizione matrici A e B linearizzate (eq. 12.17)
def get_system_matrices(Jx, Jy, Jz, hw):
    A = np.zeros((6, 6))
    A = hw / Jx
    A = -hw / Jz
    A[3:6, 0:3] = 0.5 * np.eye(3)
    return A

Jx, Jy, Jz = 189, 159, 114 # kg*m^2
hw = -2.8 # N*m*s
A_sys = get_system_matrices(Jx, Jy, Jz, hw)
print("Matrice di stato A (linearizzata):\n", A_sys)
```

Riepilogo Schematico 12.2

- **Missione:** Trasferimento di Hohmann tramite accensioni motori.
- **Attuatori:** 4 Propulsori (canted 5°) + 1 Ruota a momento.
- **Metodologia:** Linearizzazione del modello a quaternioni e sintesi LQR discreta con azione integrale.

Formule Fondamentali 12.2

1. **Dinamica d'assetto:** $J\dot{\boldsymbol{\omega}} = \mathbf{h} \times \boldsymbol{\omega} + \mathbf{m}$
2. **Cinematica Quaternioni (linearizzata):** $\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega}$
3. **Integrazione Discreta:** $\mathbf{i}_q(n+1) = \mathbf{i}_q(n) + t_s \cdot \mathbf{q}(n)$

12.3 Confronto tra Quaternioni e Angoli di Euler

Analisi e Risultati

Le fonti mostrano un confronto diretto tra un design basato su angoli di Euler e uno su quaternioni ridotti. Entrambi utilizzano il metodo LQR con gli stessi parametri (intervallo campionamento 4s, matrici di peso Q e R identiche).

Risultati della simulazione:

- Il design basato sui quaternioni mostra risposte leggermente migliori in termini di **sovraelongazione (overshoot)** e **tempo di assestamento (settling time)**.
- Nelle figure 12.4-12.9 delle fonti si osserva che le linee continue (quaternioni) convergono a zero in modo più fluido rispetto alle tratteggiate (Euler).

Mappa Concettuale Globale

1. **Ingresso Missione:** Target di assetto (es. 90° per orbit-raising).
2. **Rappresentazione Stato:** Quaternioni (evitano singolarità).
3. **Modello Dinamico:** Equazioni di Eulero per il corpo rigido + Ruota a momento.
4. **Attuazione:** Matrice di mapping propulsori (Forza/Braccio di leva).
5. **Controllo:** LQR Discreto con Stati Integrali (per robustezza e precisione).
6. **Performance:** Superiorità dei quaternioni rispetto agli angoli di Euler.

Formulario Completo

- **Cinematica dei Quaternioni:** $\dot{\bar{q}} = \frac{1}{2}\bar{q} \otimes \bar{\omega}$ (Modello non lineare).
- **Dinamica con Ruota:** $\{J_x\dot{\omega}_x = (J_y - J_z)\omega_y\omega_z + h_w\omega_z + m_x \; J_y\dot{\omega}_y = (J_z - J_x)\omega_z\omega_x + m_y \; J_z\dot{\omega}_z = (J_x - J_y)\omega_x\omega_y -$.
- **Matrici di Discretizzazione:** $\Phi = e^{At_s}$; $\Gamma = \int_0^{t_s} e^{A(t-\tau)}Bd\tau$.
- **Costo LQR:** $J = \frac{1}{2}\sum(x^TQx + u^TRu)$.

Guida di Ripasso Rapido

1. **Perché usare i quaternioni?** Perché descrivono qualsiasi rotazione senza le singolarità matematiche tipiche degli angoli di Euler (punti in cui le equazioni falliscono, es. pitch = 90°).
2. **Qual è il ruolo della ruota a momento (hw)?** Fornisce stabilità giroscopica passiva lungo l'asse normale all'orbita, riducendo lo sforzo dei propulsori.
3. **Come si gestiscono i propulsori?** Attraverso una matrice di distribuzione che trasforma la coppia richiesta dal controllore (u) nei comandi di accensione (T) dei singoli motori, considerando la loro inclinazione (canted 5°).
4. **Cos'è l'azione integrale nel controllo d'assetto?** È l'aggiunta dell'integrale dell'errore di posizione (quaternione) agli stati del sistema. Serve a compensare coppie di disturbo costanti (es. pressione solare o disallineamenti motori) che altrimenti causerebbero un errore permanente.
5. **Interpretazione dei grafici:** Se vedete oscillazioni che si smorzano, il sistema è stabile. I quaternioni riducono queste oscillazioni più velocemente degli angoli di Euler.

Spero che questo sistema di studio vi sia utile per la preparazione del vostro esame di Dinamica e Controllo dei Voli Spaziali.

Capitolo 13: Controllo MPC dell'Assetto

1.1 Introduzione e Motivazione Ingegneristica

Il **Model Predictive Control (MPC)** si basa sull'idea di risolvere ripetutamente un problema di controllo aggiornato basandosi sulle informazioni più recenti dello stato del sistema. Storicamente, l'MPC non era usato nei satelliti a causa della limitata potenza di calcolo di bordo, ma oggi, con computer più potenti, è diventato un'area di ricerca molto attiva per manovre di rendezvous, volo in formazione e controllo di assetto.

L'attrattiva principale è la capacità di gestire i **vincoli di saturazione degli attuatori** (ad esempio, la spinta massima di un propulsore) direttamente nel processo di design.

1.2 Formulazione del Problema: Da CLQR a Convex QP

Il cuore del sistema è il **Constrained Linear Quadratic Regulator (CLQR)**. Consideriamo un sistema lineare discreto tempo-invariante:

$$x_{s+1} = Ax_s + Bu_s$$

dove:

- $x \in \mathbb{R}^r$: stato del sistema (es. angoli di Eulero o quaternioni ridotti e velocità angolari).
- $u \in \mathbb{R}^m$: vettore di controllo (es. coppie dei motori a reazione o spinte dei propulsori).
- A, B: matrici del sistema che descrivono la dinamica.

Il Vincolo Ingegneristico (Box Constraint): Gli attuatori hanno dei limiti. Definiamo i vincoli come:

$$-e \leq u_s \leq e$$

dove e è un vettore di tutti uno. Questo significa che ogni componente del controllo è normalizzato tra -1 e 1 (saturazione).

Funzione di Costo (J): Vogliamo minimizzare l'errore e lo sforzo di controllo su un orizzonte N:

$$J = \min \frac{1}{2} x_{t+N}^T P x_{t+N} + \frac{1}{2} \sum_{k=0}^{N-1} [x_{t+k}^T Q x_{t+k} + u_{t+k}^T R u_{t+k}]$$

- $P, Q \geq 0$: matrici di peso sullo stato (importanza della precisione dell'assetto).
- $R > 0$: matrice di peso sul controllo (costo energetico).

1.3 Riduzione del Problema

Una delle innovazioni principali presentate nelle fonti è la riduzione delle variabili. Invece di risolvere per tutti gli stati e controlli ($Nr + Nm$ variabili), il problema viene ridotto a un **Problema di Programmazione Quadratica (QP) con soli vincoli "a scatola" (box constraints)**:

$$\min \frac{1}{2} x^T H x + c^T x \quad \text{soggetto a} \quad -e \leq x \leq e$$

Questa forma è molto più piccola (solo Nm variabili) e non ha vincoli di uguaglianza, rendendo la soluzione online molto più rapida.

1.4 L'Algoritmo: Feasible Interior-Point con Arc-Search

Per risolvere questo QP, si propone un algoritmo di **punto interno ammissibile**. **Intuizione Ingegneristica:** A differenza degli algoritmi "infeasible", dove le soluzioni intermedie potrebbero violare i vincoli, un algoritmo "feasible" garantisce che ogni iterazione sia fisicamente realizzabile. Se il computer di bordo deve interrompere il calcolo prematuramente, avremo comunque un comando sicuro da inviare agli attuatori.

L'algoritmo utilizza una **ricerca ad arco (arc-search)** invece di una ricerca lineare per seguire il "percorso centrale" (central path) verso l'ottimo, migliorando l'efficienza computazionale.

Snippet Python: Riduzione da MPC a QP

```
import numpy as np

def reduce_to_qp(A, B, Q, R, P, N, x_t):
    """
    Esempio concettuale di costruzione della matrice H e del
vettore c
    per la riduzione a box-constrained QP.
    """
    m = B.shape
    n_vars = N * m
    # Nota: Nelle fonti, H e c vengono costruiti usando potenze di
A e matrici Phi
    # Qui simuliamo una matrice H definita positiva
    H = np.eye(n_vars) + 0.1 * np.random.randn(n_vars, n_vars)
    H = H.T @ H

    # c dipende dallo stato attuale x_t
    c = np.random.randn(n_vars)

    return H, c

# Parametri ipotetici
A = np.eye(2) # Dinamica semplificata
B = np.array([[0.1], [0.5]])
Q = np.eye(2)
R = np.array([])
P = np.eye(2)
x_t = np.array([0.1, -0.05]) # Stato attuale

H, c = reduce_to_qp(A, B, Q, R, P, 10, x_t)
print(f"Dimensione matrice QP (H): {H.shape}")
```

Riepilogo Schematico - Capitolo 13

- **Problema:** Controllo di assetto con attuatori che saturano.

- **Metodo:** MPC trasformato in QP con vincoli a scatola (box constraints).
- **Vantaggio:** Riduzione drastica del carico computazionale (rimozione vincoli di uguaglianza).
- **Algoritmo:** Arc-search interior point (garantisce ammissibilità ad ogni passo).

Elenco Formule Fondamentali

1. **Dinamica Discreta:** $x_{s+1} = Ax_s + Bu_s$
2. **Vincoli Attuatori:** $-e \leq u_s \leq e$
3. **QP Standard:** $\min \frac{1}{2}x^T Hx + c^T x$
4. **Misura di Dualità (μ):** $\mu = \frac{p^T \omega}{2n}$ (indica quanto siamo vicini all'ottimo)

1.5 Esempio Applicativo: Satellite OrbView-2

Il testo cita il caso del satellite **OrbView-2** durante una manovra di innalzamento d'orbita (orbit-raising).

- **Attuatori:** 4 propulsori fissi (1 Newton) montati sugli angoli del satellite.
- **Stato:** $x = [\omega_x, \omega_y, \omega_z, q_1, q_2, q_3]^T$ (velocità angolari e quaternioni ridotti).
- **Risultato:** L'algoritmo converge in soli **0.88 secondi** dopo 20 iterazioni in ambiente Matlab, dimostrando di essere pronto per applicazioni in tempo reale.

Mappa Concettuale Globale

1. **Ingresso:** Stato attuale del satellite (x_t) e modello dinamico (A, B).
2. **Trasformazione:** L'MPC viene "compresso" in un problema QP con vincoli $-1 \leq u \leq 1$.
3. **Ottimizzazione:**
   - Si parte da un **punto iniziale ammissibile** (es. $u = 0$).
   - Si usa l'**Arc-Search** per muoversi lungo l'ellisse che approssima il percorso centrale.
   - Si monitora la **misura di dualità** μ; quando $\mu < \epsilon$, abbiamo la soluzione ottima.

4. **Uscita:** Il primo comando della sequenza ottima (u_t) viene inviato ai propulsori.
5. **Ciclo:** Si ripete tutto al passo temporale successivo (Orizzonte Recedente).

Formulario Completo

- **Matrice Hessiana Ridotta (H):** $H = \Phi_N^T P \Phi_N + \sum_{k=1}^{N-1} Q_k + R_N$
- **Vettore Lineare (c):** $c^T = x_t^T (A^T P \Phi_N + \sum_{k=1}^{N-1} S_k)$
- **Condizioni KKT (Ottimalità):**
  - $Hx + c + \lambda - \gamma = 0$
  - $\lambda_i(1 - x_i) = 0$
  - $\gamma_i(1 + x_i) = 0$
- **Inizializzazione Suggerita:** $x_0 = 0$, $\lambda_{0,i} = 4(1 + |c|_2) - c_i/2$, $\gamma_{0,i} = 4(1 + |c|_2) + c_i/2$.

Guida di Ripasso Rapido (2 Pagine)

Punti Chiave da Ricordare

1. **Perché MPC?** Gestisce nativamente i vincoli. Se un satellite deve ruotare velocemente ma i suoi motori sono deboli, l'MPC pianifica la manovra più veloce possibile senza "chiedere" ai motori più di quanto possano dare.
2. **La "Box Constraint":** È la forma più semplice di vincolo ($|u| \leq u_{max}$). Sfruttando questa semplicità, eliminiamo la necessità di calcoli pesanti richiesti per vincoli generici.
3. **Feasibility (Ammissibilità):** In ambito aerospaziale, l'affidabilità è tutto. Un algoritmo che fornisce sempre una soluzione "sicura" (anche se non ancora perfetta) è preferibile a uno che potrebbe fallire se il tempo scade.
4. **Complessità Polinomiale:** L'algoritmo proposto ha una complessità di $O(\sqrt{n}\log(1/\epsilon))$, il che significa che il tempo di calcolo cresce in modo prevedibile e lento all'aumentare della precisione richiesta o della dimensione del problema.

Domande Tipiche d'Esame

- **D:** Qual è il vantaggio principale della riduzione a box constraints?

- **R:** Riduce drasticamente il numero di variabili e rimuove i vincoli di uguaglianza, permettendo l'uso di algoritmi di punto interno molto veloci e che garantiscono l'ammissibilità dei controlli in ogni istante.
- **D:** Cosa succede se l'algoritmo viene interrotto prima della convergenza?
- **R:** Grazie all'approccio "feasible interior-point", la soluzione ottenuta all'ultima iterazione completa è comunque valida (rispetta i limiti dei motori) e "quasi-ottima", permettendo il funzionamento del satellite in sicurezza.

14.1 Modellistica del Veicolo Spaziale con VSCMG

Concetti e Intuizione Ingegneristica

Un **Control Moment Gyro (CMG)** è un attuatore che sfrutta l'effetto giroscopico per generare coppie elevate. Tradizionalmente, il volano ruota a velocità costante e la coppia viene prodotta cambiando la velocità del *gimbal* (il telaio mobile). Il **VSCMG** aggiunge un grado di libertà: la velocità del volano può variare.

Intuizione fisica: Mentre un CMG convenzionale può generare coppia solo in una direzione perpendicolare all'asse del gimbal, un VSCMG può generare coppie in un intero piano, rendendo il sistema intrinsecamente più flessibile e capace di evitare le temute "singolarità" (punti in cui l'attuatore non può fornire la coppia richiesta).

Definizioni e Formule Fondamentali

Immaginiamo di avere N VSCMG. Definiamo i tre assi unitari nel sistema di riferimento del corpo (Body Frame):

- s_j: Asse di rotazione (spin) del volano.
- g_j: Asse del gimbal (costante rispetto al corpo).
- t_j: Asse trasverso (direzione della coppia giroscopica), definito come $t_i = g_i \times s_i$.

Momento Angolare Totale (h):

$$h = J_b\omega + A_s J_s \omega_s + A_g J_g \omega_g$$

Dove:

- J_b: Matrice d'inerzia del veicolo ($kg \cdot m^2$).
- ω: Velocità angolare del veicolo (rad/s).
- A_s, A_g: Matrici le cui colonne sono gli assi s_j e g_j.
- J_s, J_g: Inerzie dei volani e dei gimbal ($kg \cdot m^2$).

- ω_s, ω_g: Velocità di rotazione dei volani e dei gimbal (rad/s).

Dinamica del Sistema: Derivando il momento angolare nel tempo e considerando le coppie esterne t_e, otteniamo l'equazione del moto:

$$J_b\dot{\omega} + A_t J_s \Omega_s \omega_g + \omega \times h = A_s t_s + A_g t_g + t_e$$

Dove t_s e t_g sono le coppie generate dall'accelerazione/decelerazione rispettivamente dei volani e dei gimbal.

Rappresentazione State-Space (LTV)

Il sistema può essere linearizzato attorno a un punto di equilibrio zero (tutto fermo), diventando un sistema **Lineare Tempo-Variante (LTV)**:

$$\dot{x} = A(t)x + B(t)u + Ct_e$$

Lo stato x include le velocità dei gimbal, dei volani, la velocità del corpo e i quaternioni dell'assetto.

Esempio Applicativo: Configurazione a Piramide

Nelle applicazioni reali, si usano spesso 4 VSCMG in configurazione a piramide (angolati di $\theta = 54.75°$) per garantire ridondanza e una distribuzione sferica delle capacità di coppia.

```
import numpy as np

def get_pyramid_geometry(theta_deg):
    theta = np.radians(theta_deg)
    # Matrice Ag per configurazione a piramide (costante)
    Ag = np.array([
        [np.sin(theta), 0, -np.sin(theta), 0],
        [0, np.sin(theta), 0, -np.sin(theta)],
        [np.cos(theta), np.cos(theta), np.cos(theta),
np.cos(theta)]
    ])
    return Ag

# Esempio parametri inerziali (dalle fonti)
Jb = np.diag() # kg*m^2
Js = 0.7 # Inerzia volano
Jg = 0.1 # Inerzia gimbal
```

14.2 Strategie di Controllo: Gain Scheduling vs MPC

Concetti e Analisi Critica

Il controllo di un sistema LTV è complesso perché la stabilità non dipende solo dagli autovalori istantanei, ma anche dalla velocità con cui la matrice $A(t)$ varia nel tempo (Teorema 14.1).

1. **Gain Scheduling:** Si progettano controllori per vari "punti congelati". Se il sistema ha molti parametri (come $N = 4$ gimbals), il numero di modelli esplode (es. 8^{18} modelli), rendendo il calcolo impossibile.
2. **Model Predictive Control (MPC) con Robust Pole Assignment:** Questa è la soluzione proposta. Il guadagno K viene calcolato **online** ad ogni periodo di campionamento aggiornando le matrici A e B.

Riepilogo Schematico Capitolo 14

- **Vantaggio VSCMG:** Eliminazione delle singolarità e risparmio energetico.
- **Modellazione:** Uso di quaternioni e matrici d'inerzia tempo-varianti.
- **Controllo:** Preferenza per l'MPC online grazie all'efficienza computazionale degli algoritmi di assegnazione dei poli.

Mappa Concettuale Globale

- **Attuatore (VSCMG)** $\rightarrow$ Genera Coppie (t_s, t_g) $\rightarrow$ Basato su velocità volano (ω_s) e gimbal (ω_g).
- **Dinamica del Veicolo** $\rightarrow$ Equazioni di Eulero modificate $\rightarrow$ Stato: $[\omega_g, \omega_s, \omega, q]$.
- **Metodologia di Controllo** $\rightarrow$ Sistema LTV $\rightarrow$ Robust Pole Assignment (Algoritmo robpole).
- **Obiettivo** $\rightarrow$ Stabilizzazione dell'assetto ($x \rightarrow 0$) evitando singolarità giroscopiche.

Formulario Completo

1. **Momento Angolare:** $h = J_b\omega + A_s J_s \omega_s + A_g J_g \omega_g$

2. **Cinematica (Quaternioni ridotti):** $\dot{q} = g(q, \omega) \approx \frac{1}{2}\omega$ (per piccoli angoli).
3. **Coppie di Controllo:** $t_s = -J_s\dot{\omega}_s$ e $t_g = -J_g\dot{\omega}_g$.
4. **Sistema LTV:** $\dot{x} = A(t)x + B(t)u$.
5. **Stabilità LTV (Criterio):** $|\dot{\tilde{A}}(t)| \leq \beta$ (la variazione deve essere limitata).

Guida di Ripasso Rapido (2 Pagine Equivalent)

Punto 1: Perché i VSCMG? I CMG classici "si bloccano" in configurazioni singolari dove non possono generare coppia in certe direzioni. I VSCMG usano la variazione della velocità del volano come "salvagente" per garantire controllabilità costante.

Punto 2: La sfida matematica Il sistema non è Lineare Tempo-Invariante (LTI). Le matrici A e B cambiano perché gli assi di rotazione dei volani (A_s) ruotano con i gimbal. Dobbiamo trattarlo come un sistema LTV.

Punto 3: Implementazione del Controllo Dimenticate il Gain Scheduling offline per sistemi così complessi. La strada maestra è l'assegnazione dei poli robusta calcolata in tempo reale. L'algoritmo deve essere efficiente (es. decomposizione QR per gestire la matrice B tempo-variante).

Punto 4: Risultati attesi In una simulazione tipica, vedrete le velocità del corpo ω convergere a zero in circa 100-120 secondi, con i quaternioni che seguono lo stesso andamento, dimostrando la stabilità asintotica del sistema.

```
# Applicazione Pratica: Calcolo Online Semplificato (Pseudocode)
def control_step(current_state, t):
    # 1. Update As and At based on current gimbal angles
    As, At = update_geometry(current_state.gamma)

    # 2. Build LTV Matrices A(t) and B(t)
    A_t = build_A_matrix(current_state, As, At)
    B_t = build_B_matrix(current_state, As, At)

    # 3. Solve Pole Assignment (Robpole)
    desired_poles = [-3.0, -3.1, -2.9, ...] # Valori tipici dalla
fonte
    K = robpole(A_t, B_t, desired_poles)

    # 4. Compute Control Torque
```

```
    u = K @ current_state
    return u
```

Questa struttura vi fornisce tutto il necessario per comprendere e simulare il controllo d'assetto di nuova generazione. Studiate attentamente le interazioni tra i termini giroscopici (cross-products) nelle equazioni della dinamica, poiché rappresentano il cuore del fenomeno fisico.

Capitolo 15: Rendezvous e Docking di Veicoli Spaziali

1.1 Introduzione e Concetto di "Soft Docking"

Il rendezvous spaziale è il processo mediante il quale un veicolo "chaser" si avvicina a un "target". La fase critica è il **soft docking**: l'obiettivo ingegneristico non è solo il contatto, ma l'assenza di oscillazioni che attraversino la linea dello zero (relative position/attitude). In termini fisici, ciò significa evitare che il chaser "rimbalzi" o oscilli attorno al punto di contatto, prevenendo collisioni strutturali.

1.2 Dinamica Traslazionale Relativa

Per orbite circolari, si usano spesso le equazioni di Hill (o Clohessy-Wiltshire), ma per orbite ellittiche e docking ravvicinato è necessario un modello non lineare completo. Definiamo il vettore posizione relativa nel sistema di riferimento orbitale (RSW) come:

$$\mathbf{p} = \mathbf{r}_c - \mathbf{r}_t = x\mathbf{e}_r + y\mathbf{e}_s + z\mathbf{e}_w$$

Variabili e Unità di Misura:

- $\mathbf{r}_c, \mathbf{r}_t$: Vettori posizione chaser/target dal centro della Terra (m).
- x, y, z: Coordinate relative lungo i versori radiale, trasversale e normale (m).
- θ: Anomalia vera (rad).
- μ: Costante gravitazionale geocentrica ($\approx 3.986 \times 10^{14}, m^3/s^2$).

Il modello non lineare della traslazione è espresso come:

$$m_c\ddot{\mathbf{d}} + C_t(\dot{\theta})\dot{\mathbf{d}} + D_t(\dot{\theta}, \ddot{\theta}, r_c)\mathbf{d} + n_t(r_c, r_t) = \mathbf{f}_a + \mathbf{f}_d$$

- **Intuizione Ingegneristica:** C_t rappresenta l'effetto di Coriolis, mentre D_t e n_t racchiudono le forze gravitazionali e centrifughe. Notate che se $r_c = r_t$, il termine non lineare n_t scompare.

1.3 Dinamica dell'Attitudine Relativa

Utilizziamo i quaternioni ridotti per descrivere l'orientamento relativo. L'equazione dinamica nel riferimento del chaser è:

$$J_c\dot{\omega} + C_r(\omega, q)\omega + n_r(\omega, q) = \mathbf{t}_c + \mathbf{t}_d$$

- J_c: Matrice d'inerzia del chaser ($kg \cdot m^2$).
- ω: Velocità angolare relativa (rad/s).
- q: Quaternione (adimensionale).

1.4 Controllo Predittivo (MPC) e Robust Pole Assignment (RPA)

Il sistema è **Lineare Tempo-Variante (LTV)**. La sfida è che molti parametri variano nel tempo (es. la distanza e l'anomalia vera). **Strategia di controllo:**

1. **Cancellazione dei disturbi:** Si usa una parte della forza dei propulsori per annullare i termini non lineari n_t e n_r.
2. **Posizionamento dei poli:** Per garantire il *soft docking*, tutti i poli del sistema a ciclo chiuso devono essere assegnati sull'**asse reale negativo**. Questo elimina fisicamente ogni componente oscillatoria (niente seni o coseni nella risposta temporale).

Esempio Applicativo

In un'orbita ellittica con eccentricità $e = 0.73$, il sistema deve calcolare in tempo reale il guadagno $K(t)$ per compensare la variazione di velocità orbitale.

```
import numpy as np
from scipy import signal

# Esempio: Definizione semplificata della matrice A per dinamica
traslazionale
def get_A_matrix(mc, theta_dot):
    # mc: massa chaser, theta_dot: velocità angolare orbitale
    # Matrice semplificata basata su Ct(theta_dot)
    A = np.zeros((6, 6))
    A[0:3, 3:6] = np.eye(3)
    # Termini di Coriolis semplificati
    A = 2 * theta_dot
    A = -2 * theta_dot
    return A
```

```
mc = 10.0 # kg
theta_dot = 0.0011 # rad/s (valore tipico LEO)
A_t = get_A_matrix(mc, theta_dot)
print("Matrice di stato (Traslazione):\n", A_t)
```

Riepilogo Schematico (Cap. 15)

- **Obiettivo:** Avvicinamento senza collisioni (Soft Docking).
- **Modello:** 6 DOF accoppiati (traslazione + rotazione).
- **Metodo:** MPC basato su Robust Pole Assignment.
- **Risultato:** Zero oscillazioni sulla linea dello zero, forze minime (~0.17 N).

Formule Fondamentali

1. **Posizione Relativa:** $\mathbf{p} = \mathbf{r}_c - \mathbf{r}_t$
2. **Dinamica LTV:** $\dot{\mathbf{x}} = A(t)\mathbf{x} - n_d(t) + B\mathbf{f}_a$
3. **Condizione Soft Docking:** $Re[\lambda_i(t)] \leq -\mu$ (poli reali negativi).

Capitolo 16: Modellazione Multi-Body e Controllo LUVOIR

2.1 Sistemi Multi-Body e il Metodo di Kane

Le missioni moderne come il telescopio LUVOIR sono sistemi multi-body complessi. Il metodo di Kane è superiore per sistemi con giunti rotanti collegati in strutture ad albero. L'equazione di Kane in forma di Stoneking è:

$$(\Omega^T[J]\Omega + V^T[M]V)\dot{\mathbf{x}}_g = \mathbf{r}_1 + \mathbf{r}_2$$

Definizioni:

- Ω: Diade delle velocità angolari parziali.
- V: Diade delle velocità lineari parziali.
- $[J], [M]$: Matrici globali d'inerzia e massa.
- $\dot{\mathbf{x}}_g$: Accelerazioni generalizzate.

2.2 Il Modello a Tre Corpi (LUVOIR)

Il telescopio è modellato come: **Spacecraft Bus + Boom + Payload**.

- **Gimbal (Joints):** Collegano i corpi e permettono il puntamento di precisione.
- **Interpretazione Fisica:** Il movimento del payload (telescopio) reagisce sul bus (corpo principale). Ignorare queste interazioni porta a errori di puntamento inaccettabili.

2.3 Linearizzazione Simbolica

Poiché la derivazione manuale è impossibile, si utilizza la differenziazione simbolica per ottenere il modello lineare:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

Lo stato $\mathbf{x}$ include angoli di Eulero del bus, angoli dei gimbal e le rispettive velocità.

2.4 LQR vs Robust Pole Assignment (RPA)

1. **LQR (Linear Quadratic Regulator):** Minimizza il consumo di energia ma produce spesso oscillazioni (jitter).
2. **RPA:** Permette di forzare i poli ad essere reali. **Intuizione:** L'RPA viene usato dopo l'LQR per "affinare" il design, eliminando le vibrazioni residue che disturberebbero le immagini del telescopio.

```
# Simulazione di puntamento con controllo proporzionale-derivativo
(PD) semplificato
# In un sistema multi-body, i guadagni devono essere robusti
def multi_body_dynamics(x, u, J_system):
    # x = [angolo, velocità angolare]
    # J_system: momento d'inerzia equivalente
    inv_J = np.linalg.inv(J_system)
    x_dot = np.array([x, (inv_J @ u)])
    return x_dot

J_luvoir = np.diag() # kg*m^2 (esempio)
state = np.array([0.1, 0]) # 0.1 rad errore di puntamento
control_torque = -50 * state - 20 * state # PD Control
print("Derivata dello stato:", multi_body_dynamics(state,
control_torque, J_luvoir))
```

Riepilogo Schematico (Cap. 16)

- **Metodo:** Equazioni di Kane (Stoneking's form).
- **Sistema:** LUVOIR (3 corpi: Bus, Boom, Payload).
- **Confronto:** RPA consuma meno energia dell'LQR e riduce le oscillazioni a 0.45 Hz.

Elenco Formule Fondamentali

1. **Velocità Angolare Parziale:** $\omega = \Omega \dot{q}$
2. **Velocità Lineare Parziale:** $v = V \dot{q}$
3. **Costo LQR:** $J = \frac{1}{2} \int (x^T Q x + u^T R u) dt$

Mappa Concettuale Globale

1. **Modellazione Dinamica:**
   - *Traslazione (Rendezvous):* Modelli non lineari ellittici ($m_c \ddot{d} + \dots$).
   - *Rotazione (Multi-body):* Metodo di Kane per sistemi articolati.
2. **Variabili di Stato:**
   - Posizione relativa (x, y, z), Quaternioni (q), Angoli di gimbal (γ, λ).
3. **Strategia di Controllo:**
   - *Feedback Linearization:* Per annullare i termini non lineari spaziali (n_d).
   - *Robust Pole Assignment (RPA):* Cuore del sistema per garantire il "Soft Docking" e il "Precision Pointing" tramite poli reali negativi.
4. **Validazione:**
   - Simulazione su modelli rigidi e successiva verifica su modelli flessibili (High Fidelity).

Formulario Completo

- **Dinamica Orbitale (RSW):** $r_t = \frac{p}{1+e\cos\theta}$
- **Evoluzione Quaternione:** $\dot{q} = \frac{1}{2} T(q) \omega$
- **Modello di Kane (Matriciale):** $L \dot{x}_g = r_1 + r_2$
- **Energia Consumata:** $E = \int |u(t)| dt$
- **Trasformazione di Coordinate (RSW a ECI):** $r_{|I} = O_{I/s} r_{|s}$

Guida di Ripasso Rapido (Quick Review)

1. Perché il metodo di Hill non basta? Perché assume orbite circolari e piccole distanze lineari. Per il docking finale ellittico, le forze di accoppiamento non lineari sono significative.

2. Qual è il vantaggio principale di Kane rispetto a Lagrange? Kane evita il calcolo delle derivate parziali dell'energia cinetica, usando velocità parziali e forze generalizzate, rendendolo computazionalmente più efficiente per sistemi spaziali multi-body complessi.

3. Cosa si intende per "Robust Pole Assignment"? È una tecnica che posiziona i poli del sistema per minimizzare la sensibilità agli errori di modellazione (come i modi flessibili non modellati di un telescopio).

4. Come si ottiene il soft docking? Assegnando tutti i poli del sistema sul semiasse reale negativo. Fisicamente, questo trasforma la risposta del sistema da una "oscillazione smorzata" a un "decadimento esponenziale puro", evitando che il veicolo superi il punto di contatto.

5. LQR vs RPA per LUVOIR: L'LQR è ottimo per il design iniziale, ma l'RPA è necessario per garantire che le vibrazioni dei gimbal (che possono verificarsi a specifiche frequenze, es. 0.45 Hz) siano eliminate rapidamente per non degradare la qualità delle immagini.