Report :

Come primo step per il nostro tema d'anno, con il fine di dimensionare e verificare la fattibilità del progetto con l'hardware a nostra disposizione, abbiamo deciso di modellare e simulare la dinamica del manipolatore con Matlab e Simulink.

Il metodo da noi scelto per il dimensionamento prevede l'utilizzo di una traiettoria critica, realizzata tramite un polinomio del 7°grado. Come parametri del polinomio abbiamo imposto:

- Valori iniziali e finali di posizione, velocita e accelerazione.
- Accelerazione massima a $\frac{Tf}{4}$.
- Imposto il passaggio per il punto ($\frac{Tf}{2}$, 0)

Per poter rendere parametriche le dimensioni abbiamo espresso la massa e le posizioni dei link approssimandoli a dei parallelepipedi di dimensione generiche lz,ly,lx e considerando la densità costante.
Per rendere indipendente il modello dinamico dalle variabili q e $\dot{q}$ abbiamo deciso di realizzare una libreria tramite il symbolic toolbox, la quale è in grado di costruire le matrici B, C ed il vettore g in funzione delle variabili q e $\dot{q}$.
Nella prima implementazione avevamo deciso di modellare la dinamica senza l'utilizzo di alcun framework utilizzando l'equazione Lagrangiana. Questo però è stato piuttosto infruttuoso poiché anche dopo aver applicato diversi metodi di Controllo e dopo diverse iterazioni, la matrice B, e di conseguenza anche la matrice di Coriolis, non rappresentavano correttamente la dinamica del sistema causando la divergenza del sistema.
Dopo diversi tentativi e re-implementazioni infruttuosi, abbiamo deciso di comparare le matrici ottenute con il metodo dell'equazione Lagrangiana con quelle che ottenevamo utilizzando il Robotics toolbox di Peter Corke. Effettivamente è stata rilevata una differenza tra i due modelli dinamici, sebbene le matrici Jacobiane e le matrici di rotazione e posizione fossero corrette, motivo per il quale abbiamo deciso di utilizzare il modello ottenuto tramite l'utilizzo del Robotics toolbox per condurre le nuove simulazioni.

Per quanto riguarda il controllo abbiamo scelto d'implementare il controllo a dinamica inversa.

Seguono i risultati delle simulazioni.

```
for i=1:2
    Jl = obj.linkjacobian(i,lx);
    Jp = Jl(1:3,:);
    Jo = Jl(4:end,:);
    R = eye(3);
    if(i == 2)
        R = Rot(q(1)+q(2));
    else
        R = Rot(q(1));
    end
    ml=lx(i)*ly(i)*lz(i)*rho;
    Il = obj.linkinertia(i,lx,ly,lz,rho);
    Jm = obj.motorjacobian(i, lx,kr);
    Jpm = Jm(1:3,:);
    Jom = Jm(4:end,:);
    I_m = diag([0 0 Im(i)]);

    B=B+ml*(Jp.')*Jp+(Jo.')*R*Il*(R.')*Jo+mm(i)*(Jpm.')*Jpm+(Jom.')*R*I_m*(R.')*Jom;
end

% gravity component
g=obj.gravity(lx,ly,lz,rho, mm);

% coriolis matrix

for i=1:2
    for j=1:2
        C(i,j)=0;
        for k=1:2
            C(i,j)=simplify(C(i,j)+0.5*(diff(B(i,j), q(k))+diff(B(i,k),q(j))-diff(B(j,k),q(i)))*dq(k)); %mod
        end
    end
end
```

Figura 1: Implementazione modello dinamico con la Lagrangiana.

```
function [B, C, g, q, dq] = dyn(obj,lx,ly,lz,rho,mm,Im,kr)
    [manip, ~, q, dq] = obj.init(lx,ly,lz,rho,mm,Im,kr);
    B = manip.inertia(q.');
    C = manip.coriolis([q.' dq.']);
    g = manip.gravload(q.',[0;0;9.81].').';
    % TODO: dato che g doveva essere trasposto, non è che anche C
    % deve essere trasposta? B è simmetrico quindi è indifferente
end
```

Figura 2: Implementazione tramite il Robotic Toolbox

Risultati Simulazioni:

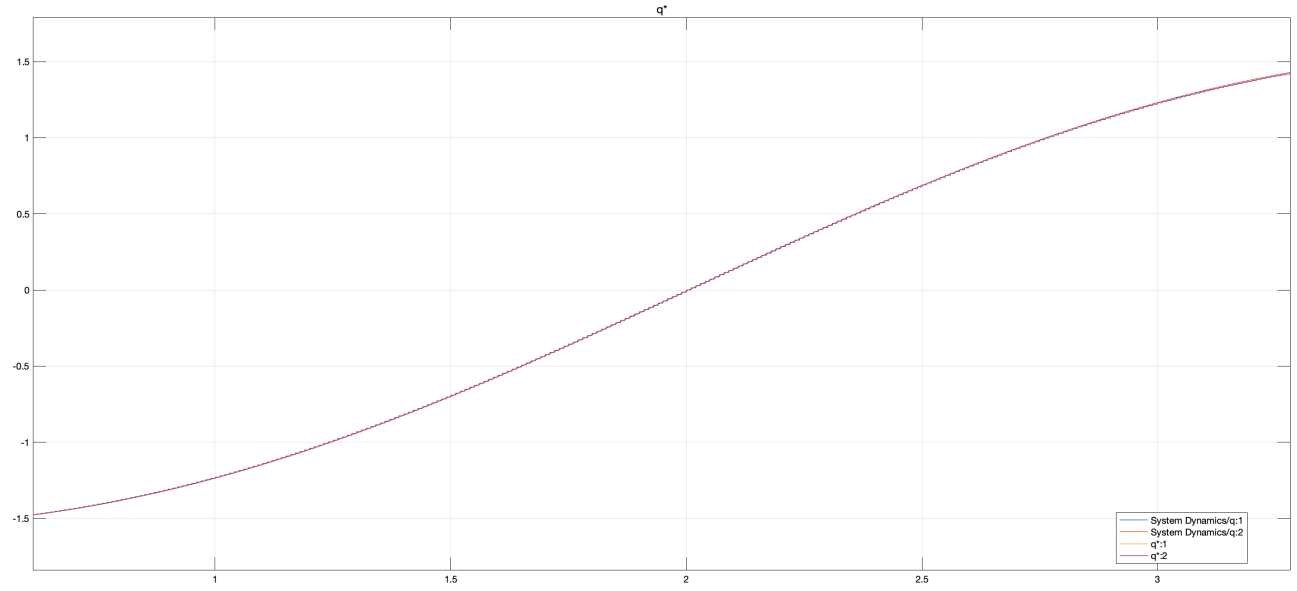

Figura 3: Posizione nello spazio dei giunti.

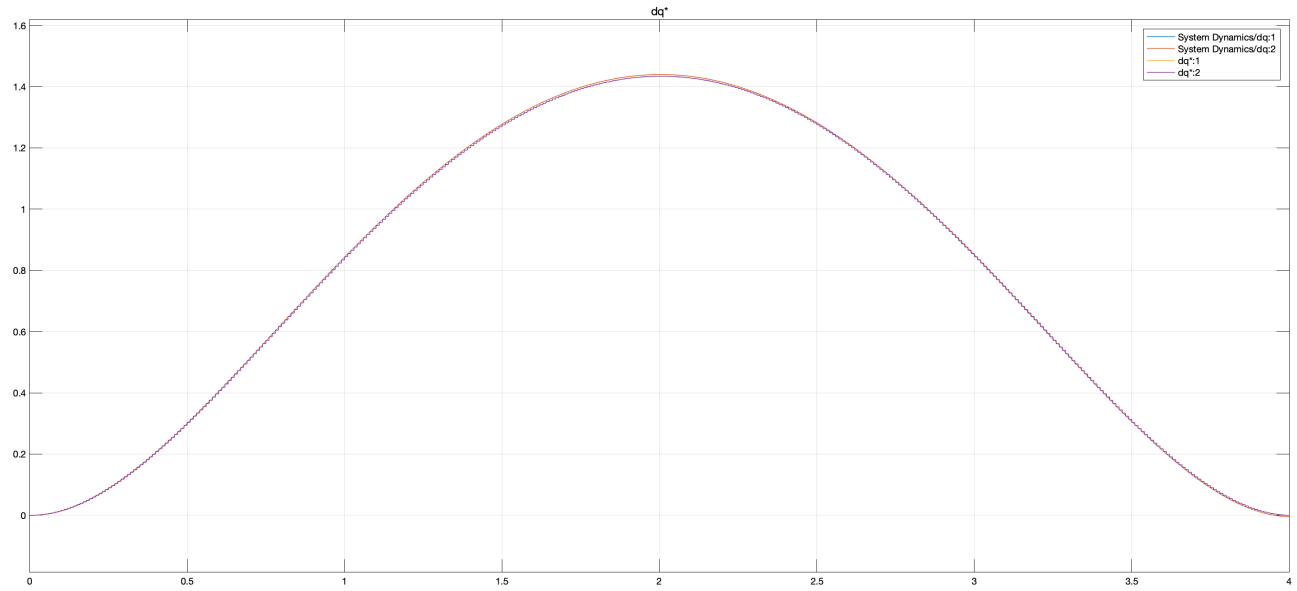

Figura 4: Velocità nello spazio di giunti.

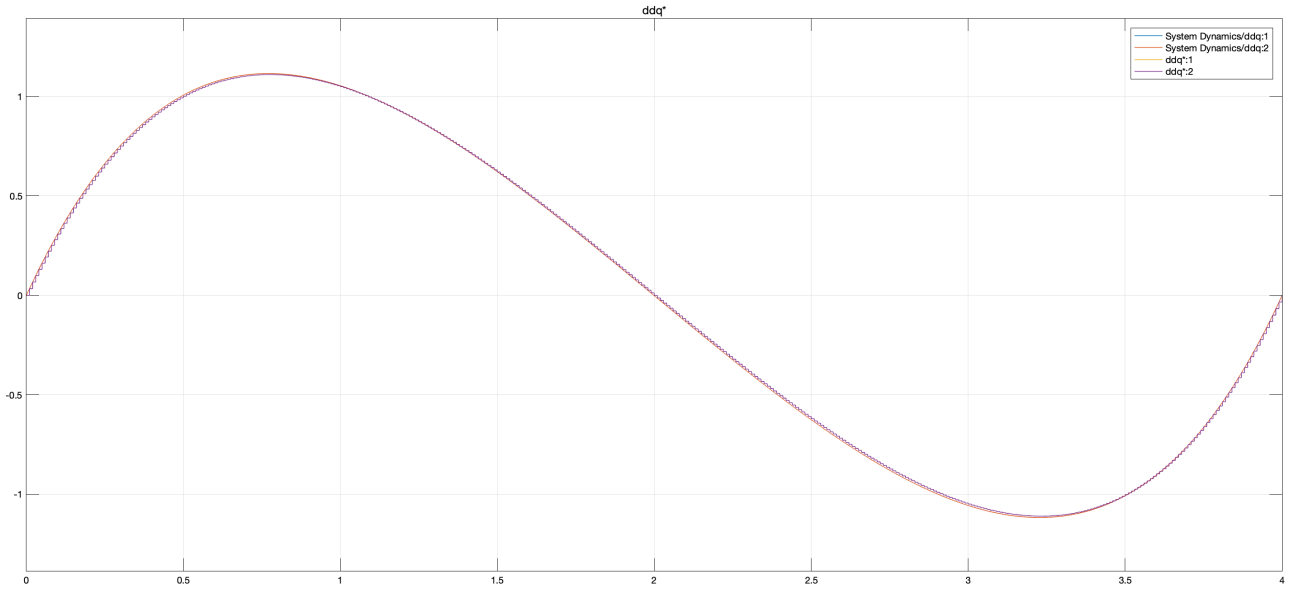

Figura 5: Accelerazioni nello spazio dei giunti.

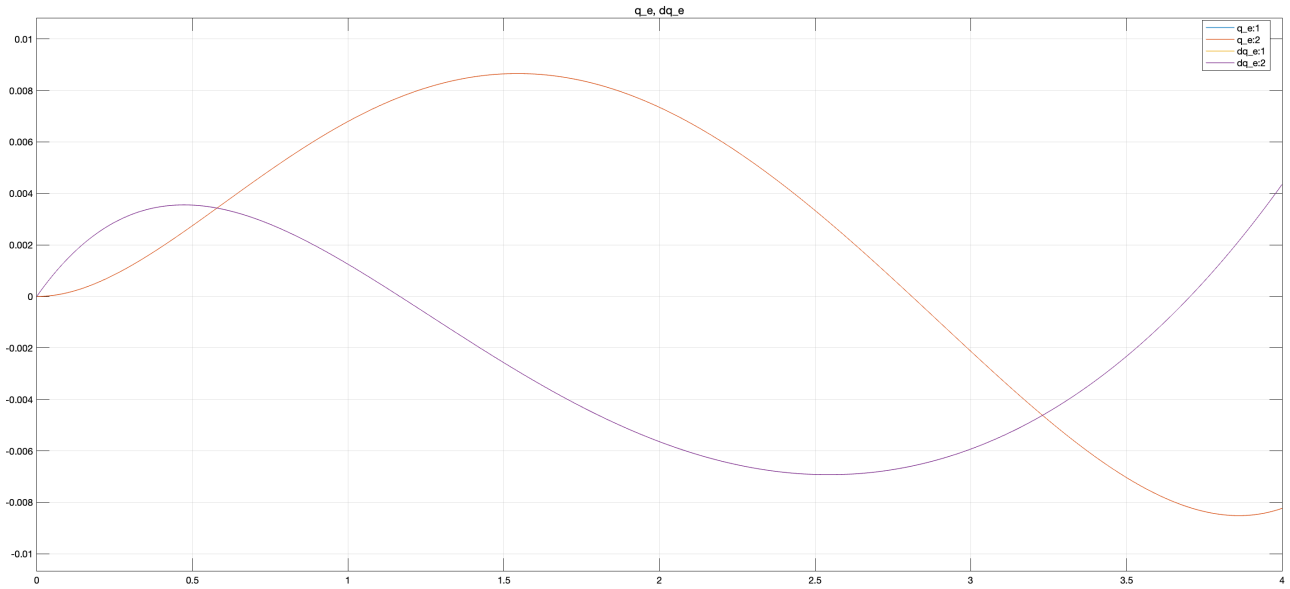

Figura 6: Errori di posizione e velocità.

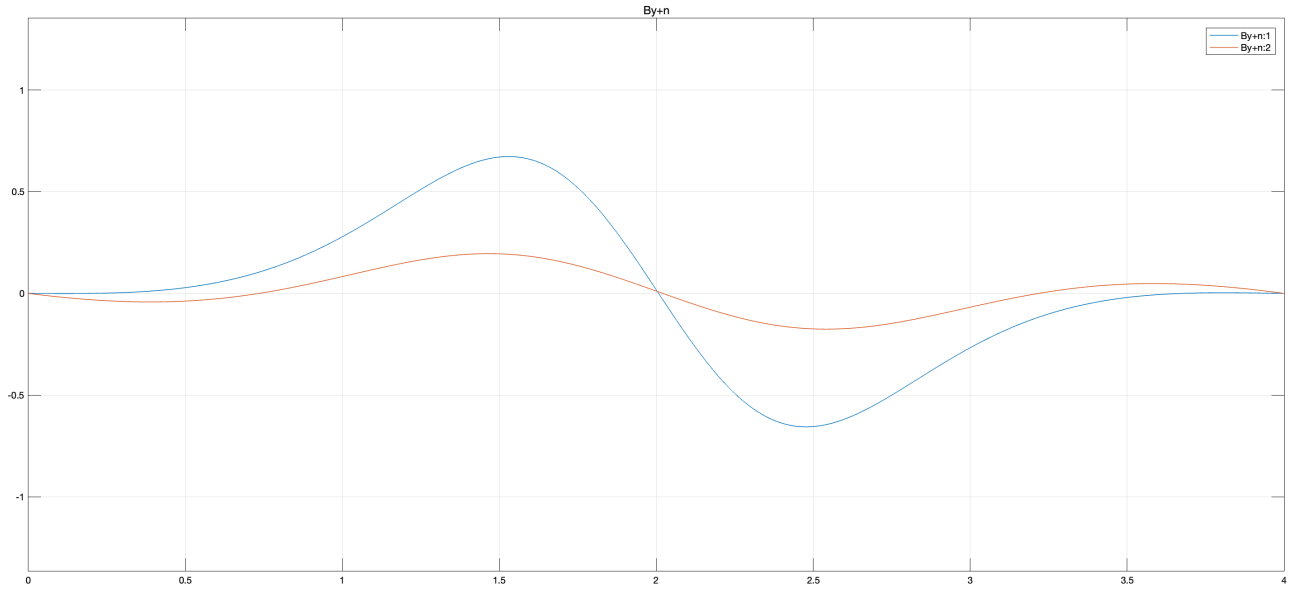

Figura 7: Segnale generato dal controllo.

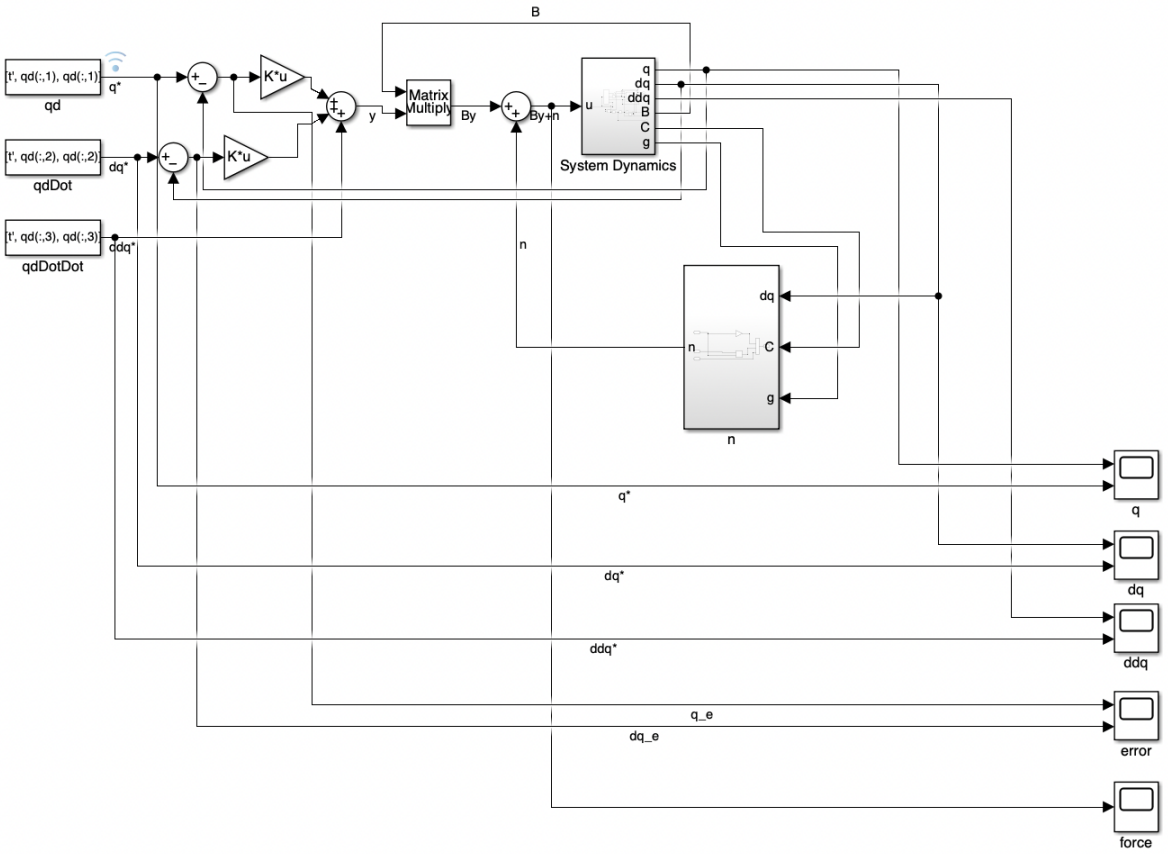

Figura 8: Modello Simulink del controllo.

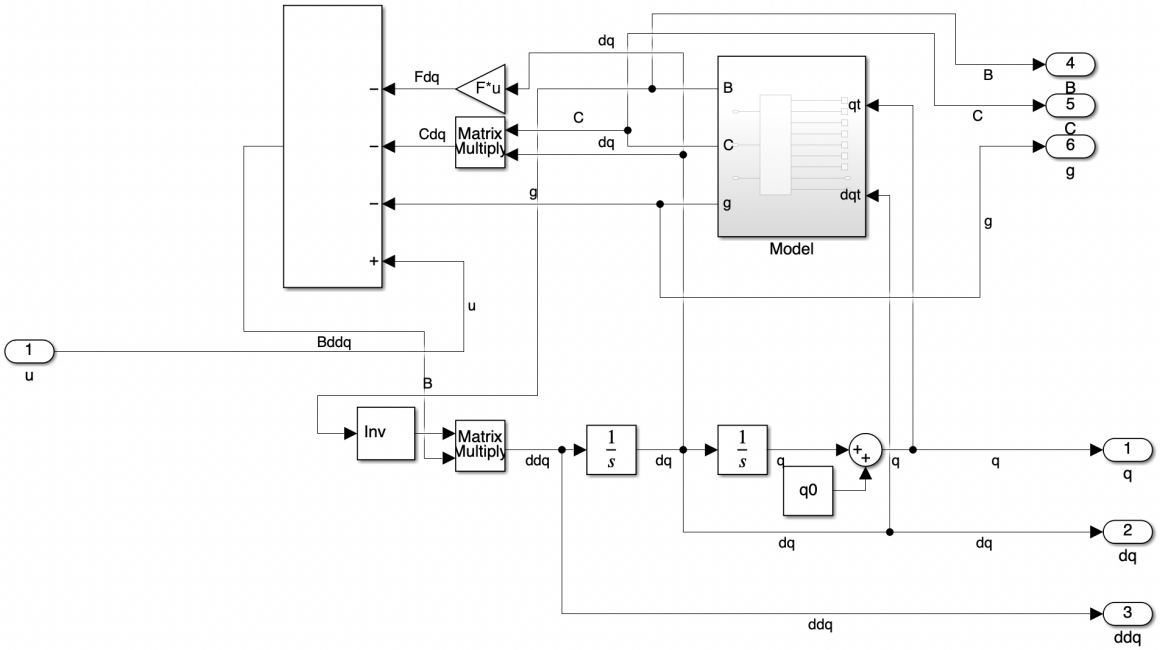

Figura 9: Modello della Dinamica.