Progetto Robotica Industriale

Flavio Maiorana 182611

## Cinematica del Robot

Definita la tabella di Denavit-Hartenberg in base ai parametri a,alpha,d,theta, si puó facilmente ottenere la matrice di rototraslazione, e per mettere in relazione e , avremo che , che è effettivamente la matrice di rototraslazione utile per calcolare la posizione della terna utensile in . A tal fine ho utilizzato lo script *dirkin\_ant*, che definisce la tabella DH e crea le variabili simboliche, per poi invocare lo script *dirkin*, che riceve una tabella DH e produce concatenando le matrici omogenee dei giunti. Chiaramente, siccome i punti P1, P2, P3 sono espressi nel sistema , bisogna adattarli a , e per fare questo ho sfruttato il fatto che , e quindi, avremo che .

Lo step successivo è quello di stabilire una traiettoria.

## Problema 1: Inseguimento di una traiettoria punto-punto

Per quanto riguarda la traiettoria punto-punto, una volta definiti i punti da visitare p1…pn, il problema maggiore sta nello stabilire la legge oraria da seguire durante il tragitto. Un profilo possibile è quello polinomiale: , dove appunto viene modellato come un polinomio quintico che varia tra 0 e 1. Con 6 coefficienti da calcolare ho posto 6 condizioni su posizione, derivata e accelerazione e risolto il sistema con il toolbox simbolico di Matlab. Una volta trovato , ho normalizzato la variabile temporale fittizia s (che pure varia tra 0 e 1) per il tempo di durata effettiva del tragitto da p1 a p2, attraverso la relazione , in maniera tale da ottenere la durata desiderata di 10 secondi del tragitto totale. Di conseguenza in questo modo trovo posizioni e velocitá sostituendo nell’espressione simbolica lambda i valori normalizzati di s (ho suddiviso ogni tragitto in 50 samples temporali).

Un altro profilo di velocitá possibile è quello trapezoidale: consiste nel definire la legge oraria in base alla sua velocitá lungo il tragitto. Ho, a tal proposito, nello script mindist\_trapezoidal, imposto che il deve viaggiare da 0 a 1 a una “velocitá di crociera” pari a , dove è l’istante temporale in cui raggiungiamo il secondo punto e l’istante da cui partiamo. La definizione di questa velocitá di crociera puó essere considerata arbitraria: è sufficiente che rispetti il vincolo , dove e sono punto iniziale e finale. Questo vincolo è dovuto al fatto che, se scelgo come “punto di flesso” un punto superiore a metà del tragitto che devo fare, poi non ho abbastanza tempo per riabbassare la velocitá e raggiungere senza abbassare, appunto, repentinamente la velocitá. Nel caso in cui quel minore uguale diventasse un’uguaglianza, avrei un profilo di velocitá triangolare, piuttosto che trapezoidale.

## Problema 2: Inseguimento di una traiettoria con curvatura continua

La traiettoria con curvatura continua è utile nel caso in cui vogliamo avere un andamento piú docile del robot. Per realizzarlo basta risolvere un sistema lineare. Dati n punti, avrò n polinomi cubici (spline) per ogni coordinata nello spazio, e ciascuno viene valutato nel rispettivo intervallo di tempo in cui bisogna passare da un punto all’altro. Avrò quindi 4\*4 = 16 equazioni in 16 incognite per 4 punti da visitare, di cui 6 sulle posizioni, 6 sulle derivate prime e 4 sulle derivate seconde (ho 2 condizioni aggiuntive su derivata e posizione per le condizioni iniziali e finali) semplicemente basandomi sul principio della continuitá, ovvero

dove impongo a l’istante temporale in cui la terna utensile deve passare dal punto kappesimo. Risolvo poi il sistema con il risolutore numerico di sistemi simbolici di Matlab vpasolve, per poi calcolare numericamente con polyval le spline negli istanti discreti di tempo prestabiliti.

Un altro modo modo per ottenere una traiettoria a curvatura continua è quella di utilizzare una matrice quadrata nxn (n è il numero di istanti in cui visitare n punti) di Vandermonde, con cui ottenere i coefficienti dell’unico polinomio per ogni coordinata dello spazio. Naturalmente posso utilizzare questo metodo solo per ottenere le posizioni e per un numero di punti non troppo elevato non troppo distanti tra di loro, altrimenti diventa difficile invertire la matrice di Vandermonde. Il sistema che risolvo è in pratica questo:

## Andamento di posizione e velocitá

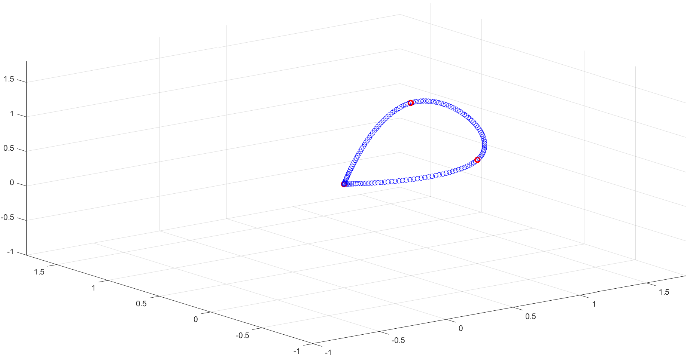
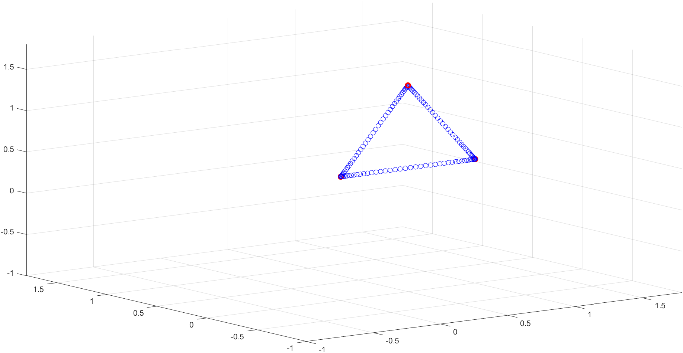
Una volta stabilito come è strutturato il robot, come caratterizzare la sua cinematica diretta (), e quale traiettoria seguire, bisogna capire come risolvere il problema inverso (), ovvero come e se, data una posizione nello spazio , sia possibile ottenere una rappresentazione della posizione del robot nello spazio dei giunti: in questo caso, in termini di, e. Per via analitica, affrontare la cinematica inversa vuol dire estrapolare le informazioni sulla posizione dalla cinematica diretta e combinare le 3 equazioni in modo tale da trovare i tre angoli desiderati risolvendo equazioni e sistemi lineari. e sono abbastanza immediate da trovare, e hanno entrambe due soluzioni ammissibili, una negativa e l’altra positiva. Infine, per trovare , so che la posizione sul piano dell’EE è pari a

per cui, siccome e , posso trovare risolvendo questo sistema:

A tal fine ho creato una funzione invkin\_ant, che restituisce una matrice di dimensione (3,1,4). Ció avviene perché per il robot antropomorfo ci sono quattro diverse possibilitá in cui potremmo raggiungere una posa che trova nel suo spazio destro. Ho scelto la soluzione a gomito alto perché altrimenti il secondo braccio sarebbe stato allineato al busto. Tuttavia, è solo grazie alla struttura particolarmente semplice del robot e alla particolare traiettoria considerata, che siamo in grado di trovare una soluzione alla cinematica inversa che sia: in forma chiusa e con un numero finito di soluzioni. Nel caso in cui il primo fatto non valga, bisognerebbe approcciarsi con un algoritmo numerico. Ho provato (numinvkin\_ant) a implementarne una soluzione basandomi sul metodo numerico di Newton, che è una soluzione iterativa basata sul concetto di espansione in serie di taylor della cinematica diretta troncata al primo ordine. Naturalmente anche l’algoritmo ha problemi nel caso in cui il secondo punto non sia rispettato, ovvero in presenza di singolaritá. Questo perché per invertire lo Jacobiano, è necessario che esso sia non-singolare. In caso contrario siamo in una situazione in cui il rango di J non è massimo, e quindi il nullo di J non è vuoto. Questo ovviamente mi dá problemi anche per l’inversione della cinematica differenziale. Nello script jacobian\_ant, che calcola sia lo jacobiano analitico che quello geometrico (se si ignora l’orientamento dell’EE, i due jacobiani coincidono), c’è anche un controllo sulle singolarità. Ad esempio, lo jacobiano ha determinante nullo se q3=0° oppure q3=180°, ovvero a gomito tutto steso. Una situazione singolare non avviene nella traiettoria presa in esame. Lo Jacobiano è quadrato, e quindi per invertire la cinematica differenziale analiticamente, mi basta che la matrice sia non-singolare, il che è vero per ogni t della nostra traiettoria. In conclusione, ottengo che per ogni campione di tempo considerato e la soluzione a questo sistema lineare è unica. Ho ottenuto lo Jacobiano analitico semplicemente calcolando le derivate parziali rispetto a ogni variabile di giunto. Lo Jacobiano geometrico invece è stato calcolato giustapponendo in ogni colonna (sono tante quanti sono i giunti):

dove è l’asse attorno al quale giunto ruota (o trasla se è prismatico) e è il vettore applicato dall’origine della terna all’origine della , quest’ultima mossa dal giunto e posizionata sul giunto . Naturalmente, siccome non è un presente un polso sul robot, non vi è presenza di orientamento nello Jacobiano, per cui solo le prime tre righe sono linearmente indipendenti.

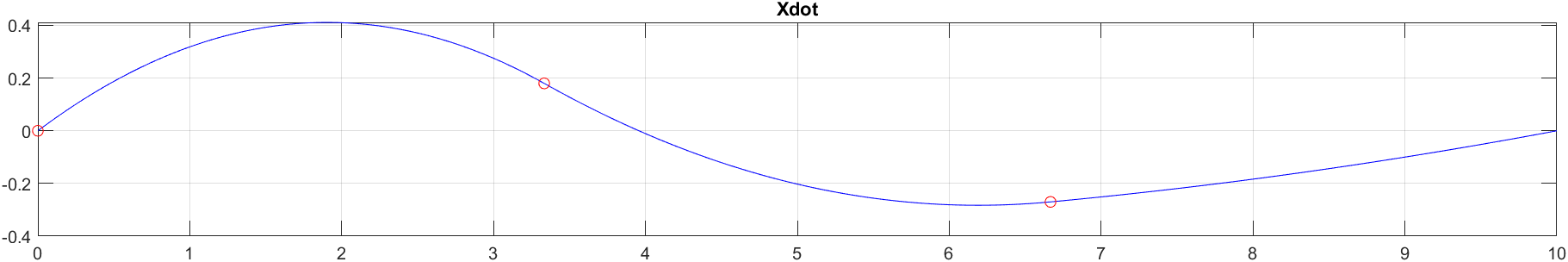
### Risultato finale



Sulla sinistra si vede il percorso a minima distanza tra un punto e l’altro, a destra invece il percorso in cui ogni coppia di punti è interpolato da una spline.

La seconda coordinata è sempre nulla, quindi non verrá riportata nelle figure.

La traiettoria continua ferma la mano del robot solo alla fine del percorso, infatti si vede che in prossimità dei punti di passaggio (in figura in rosso) non si annullano mai contemporaneamente tutte le tre le velocitá relative alle tre coordinate (figura 1), e neanche la curva delle posizioni (figura 2) si appiattisce nell’intorno del punto di passaggio, ma prosegue con lente variazioni di pendenza, cosa che invece non succede con la prima tipologia di traiettoria.



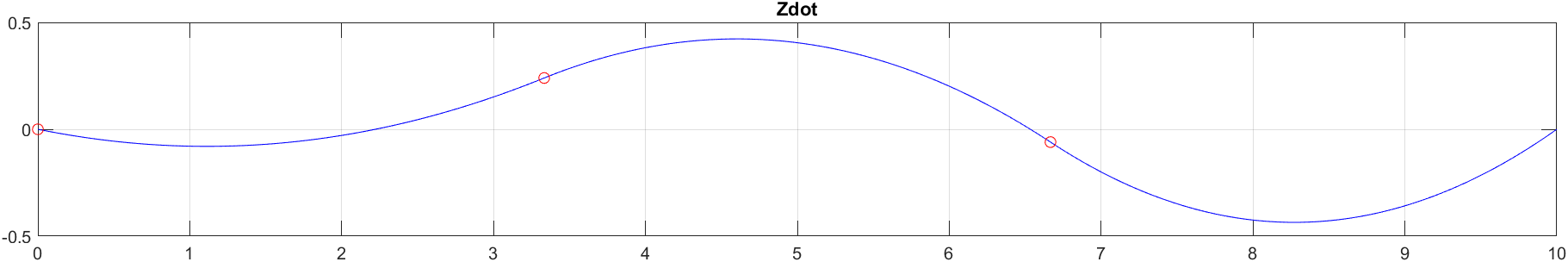
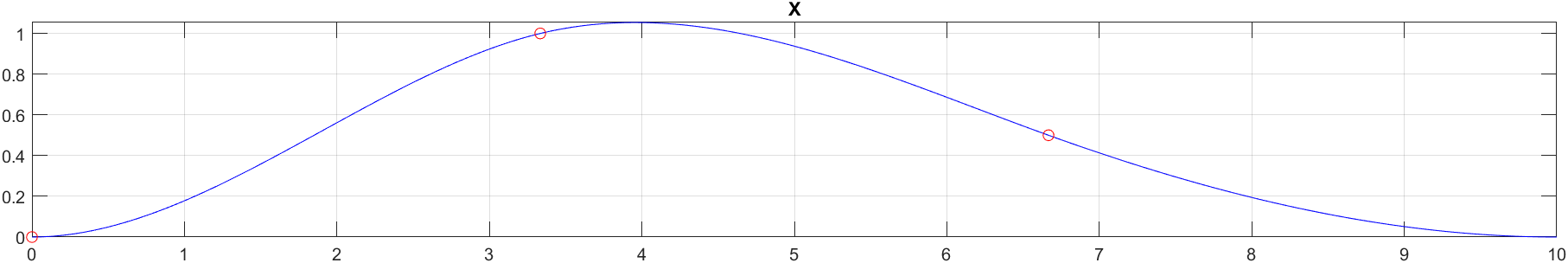


Figure 1



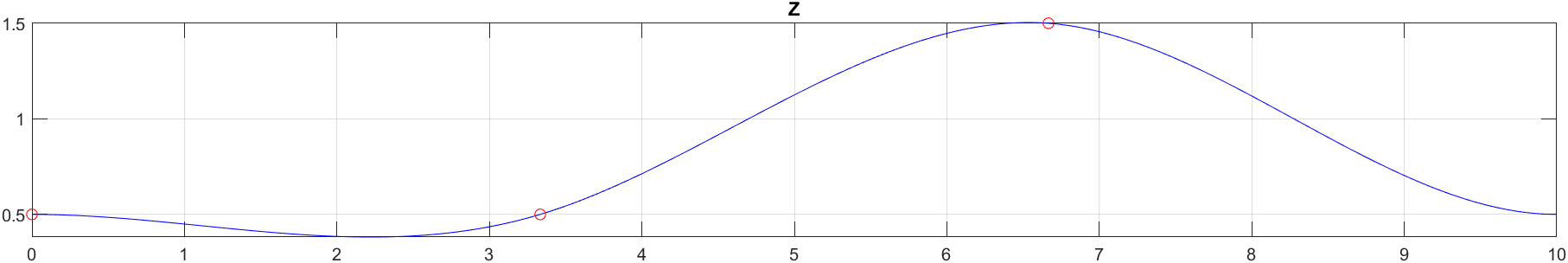
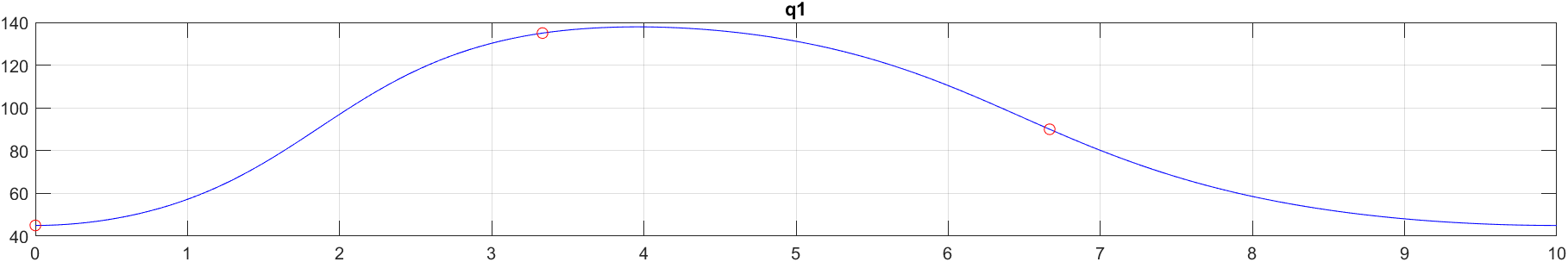
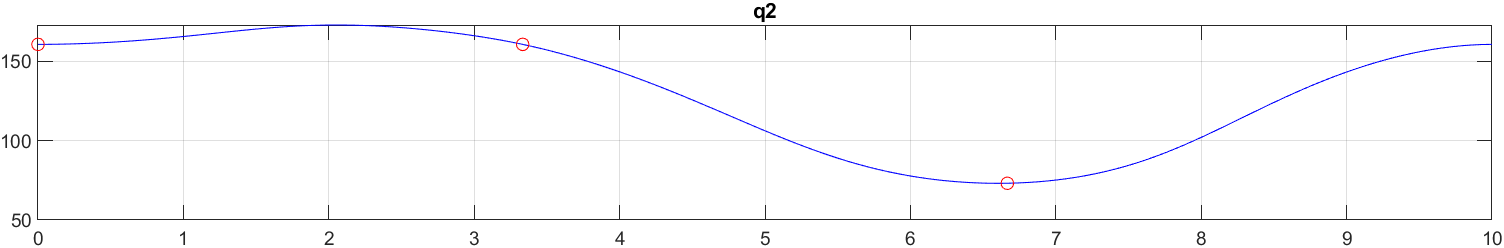


Figure 2

Della docilità del cambio di velocitá e quindi della continuitá ne risentono i giunti, che di conseguenza faranno scatti meno improvvisi. In figura 3 si vede che la curva che descrive l’andamento delle variabili di giunti nel tempo si appiattisce solo all’inizio alla fine; nei punti di passaggio la curva magari rallenta, ma non si ferma. Tornando al tema delle singolaritá, si puó inoltre vedere per ispezione che la terza variabile di giunto non si annulla mai, né arriva mai a 180° di valore, cosí in figura 1 e 2 Px e Py non sono mai contemporaneamente nulli (naturalmente sempre considerando che i punti riportati in figura si intende che siano visti dal sistema e non dal sistema che è ciò effettivamente usiamo come sistema base per la cinematica inversa e diretta), il che conferma che la matrice Jacobiana sará non singolare e quindi invertibile durante tutto il tragitto percorso dal robot.





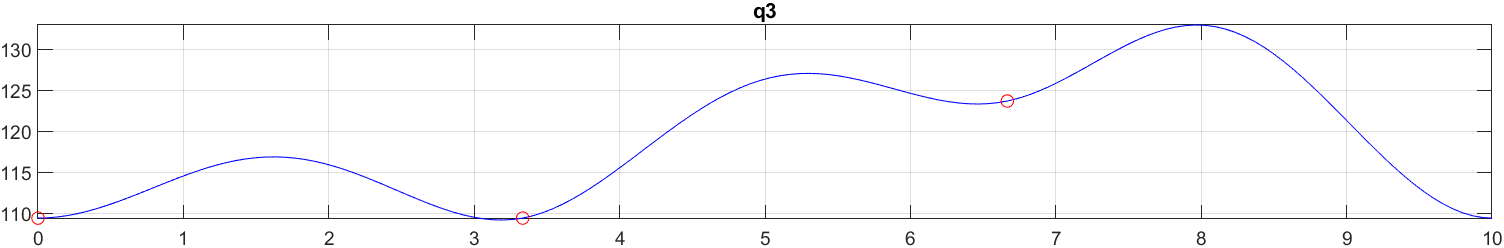
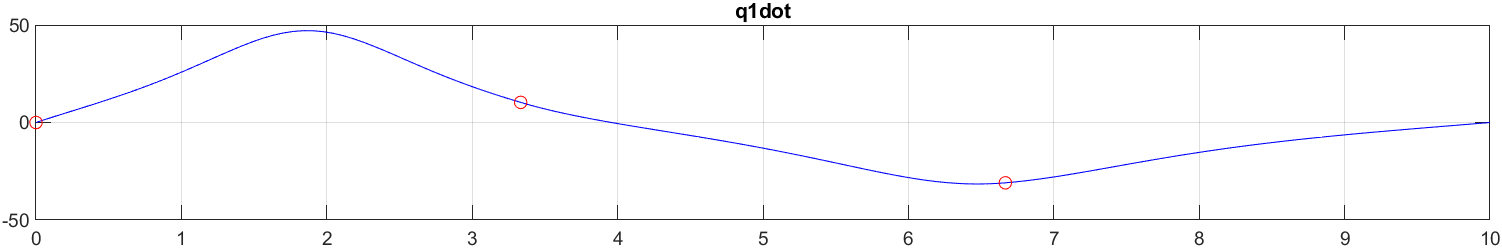
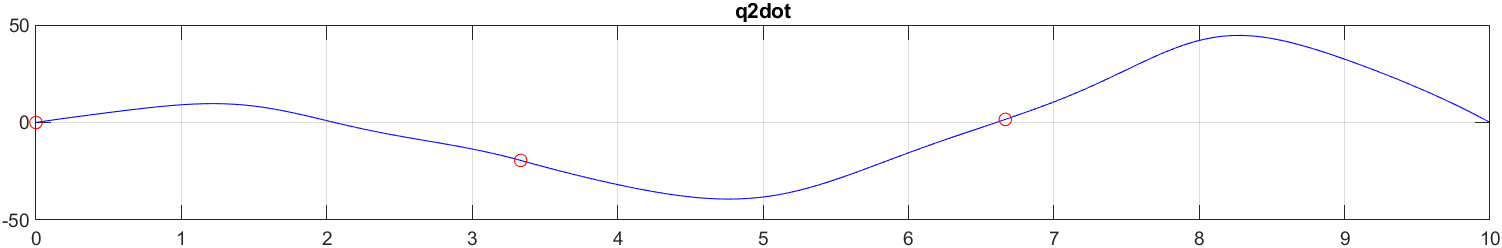


Figure 3

Si puó vedere, di seguito, molto bene la differenza anche tra figura 4 e figura 5. Nella prima vi è l’andamento in funzione del tempo della velocitá dei giunti con il percorso a curvatura continua, mentre nella seconda quello relativo al percorso a minima distanza. Si vede chiaramente che il range delle velocitá è piú elevato in figura 5 e le curve hanno delle variazioni con pendenza piú elevata, il che vuol dire variazione di velocitá piú repentina e piú frequente. Q1, ad esempio, arriva a una velocitá massima che sfora i 50 gradi/s con il percorso a minima distanza, cosa che non succede con le spline.





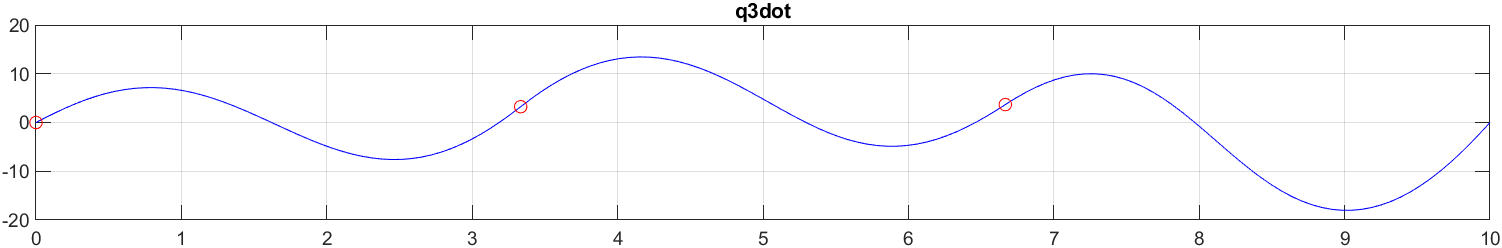
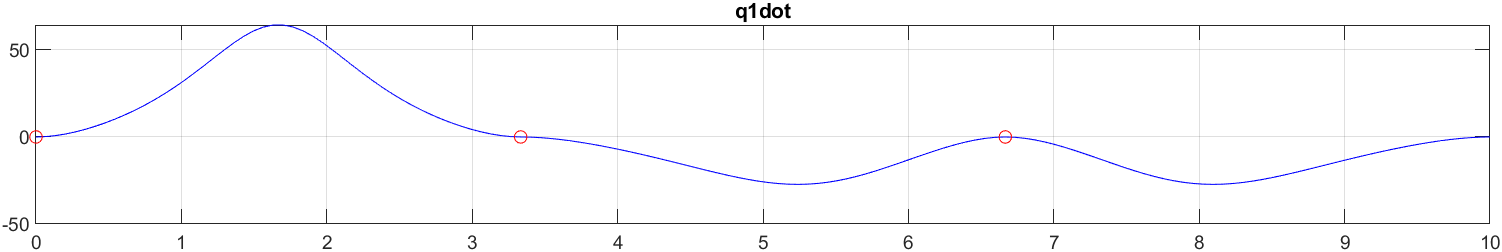
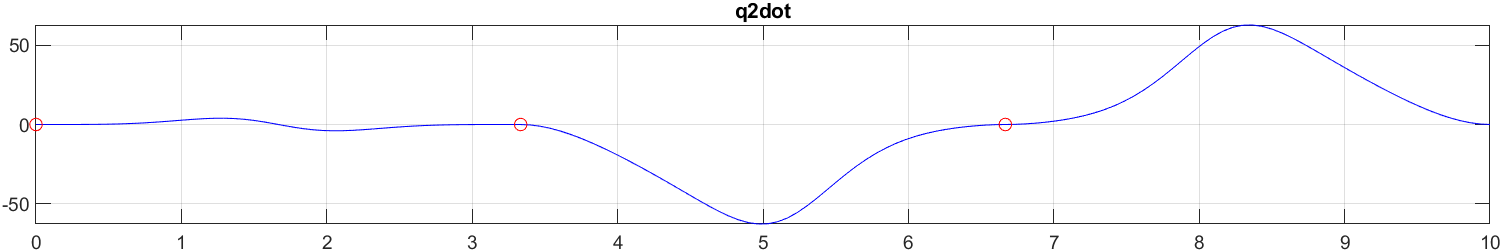


Figure 4





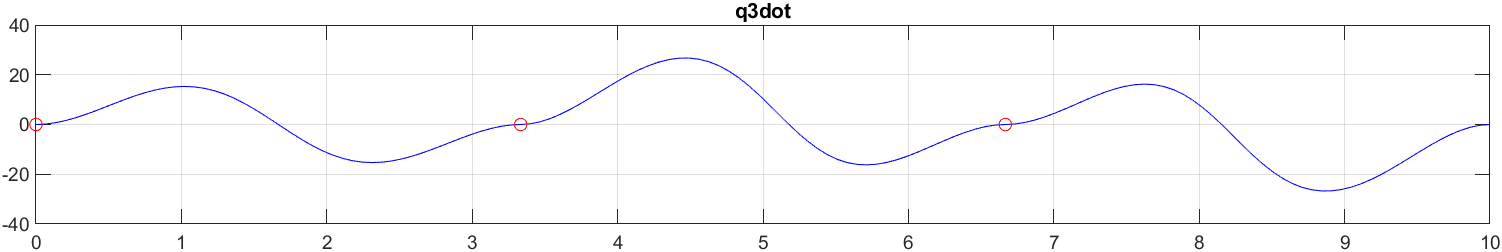
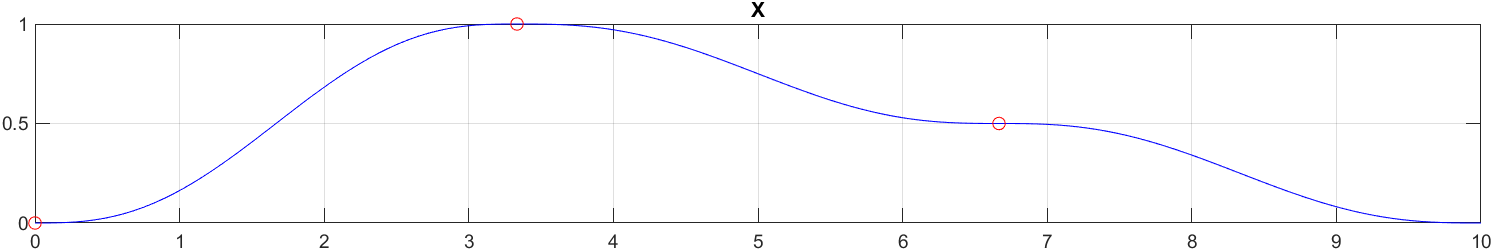


Figure 5

Di seguito in figura 6 e 7 l’andamento di posizione e velocitá nello spazio cartesiano durante il percorso a minima distanza.



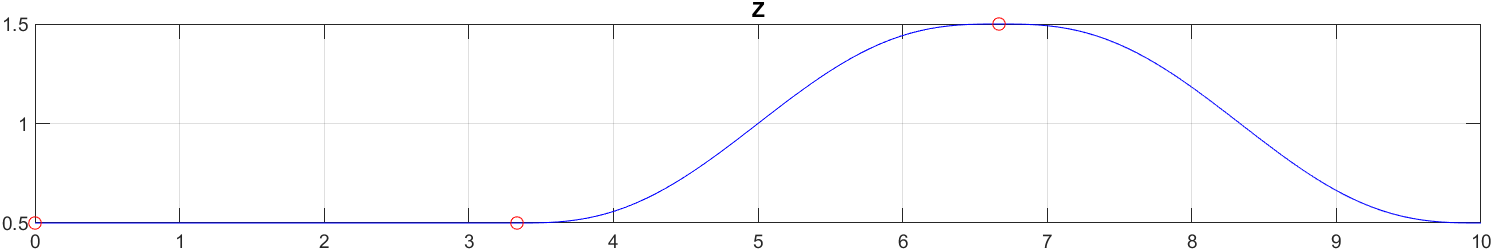
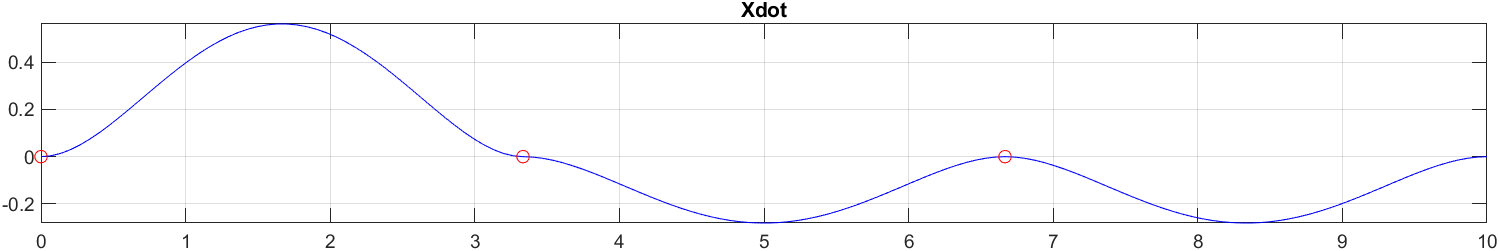


Figure 6



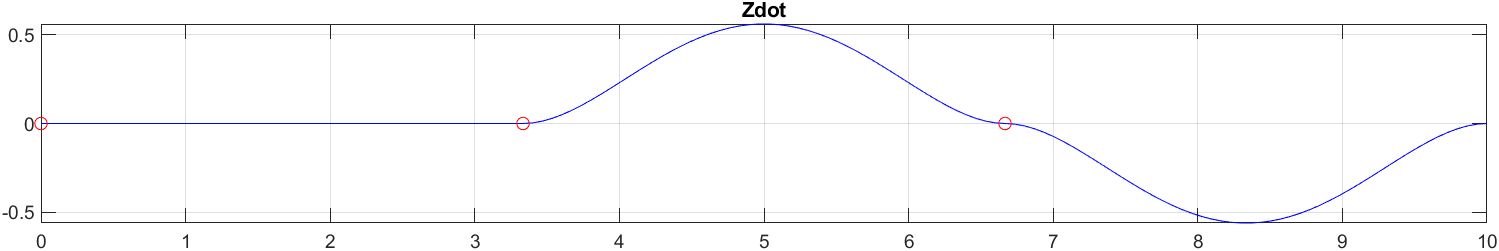
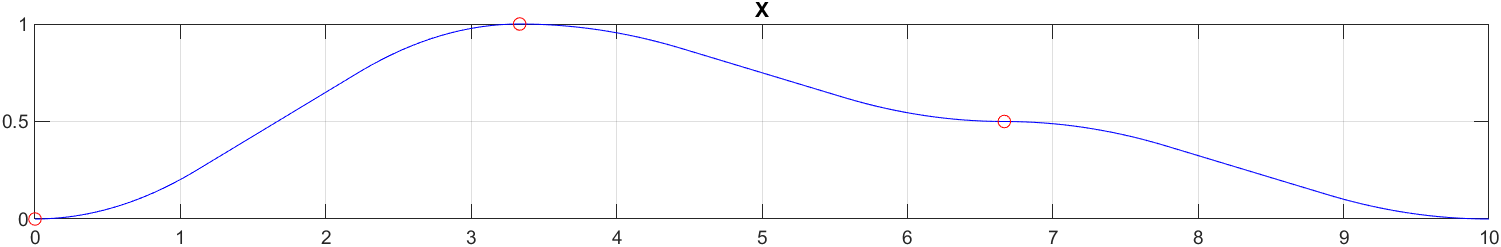


Figure 7

Si puó individuare il profilo di velocitá polinomiale dal fatto che la curva tra un punto rosso e l’altro nel caso delle posizioni è appunto un polinomio con velocitá e accelerazione iniziale e finale nulla, mentre nel caso delle velocitá assume una forma parabolica, comunque schiacciata all’inizio e alla fine per via delle accelerazioni.

Diverso invece, è il risultato se si utilizza un profilo trapezoidale della velocitá, caso in cui, come in figura 8, la curva tra un punto e l’altro è meno schiacciata agli estremi, ma è meno inclinata nel tratto intermedio, il che significa una velocitá nel complesso piú uniforme e in media meno elevata rispetto al profilo polinomiale, che predilige muoversi molto lentamente in prossimità dei punti.



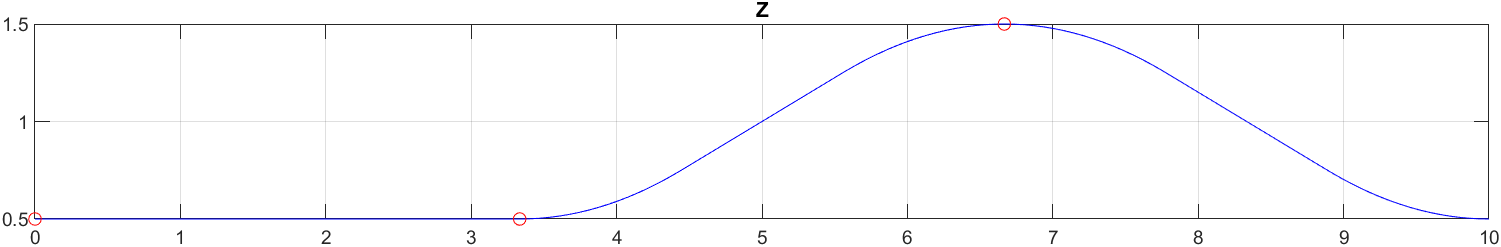
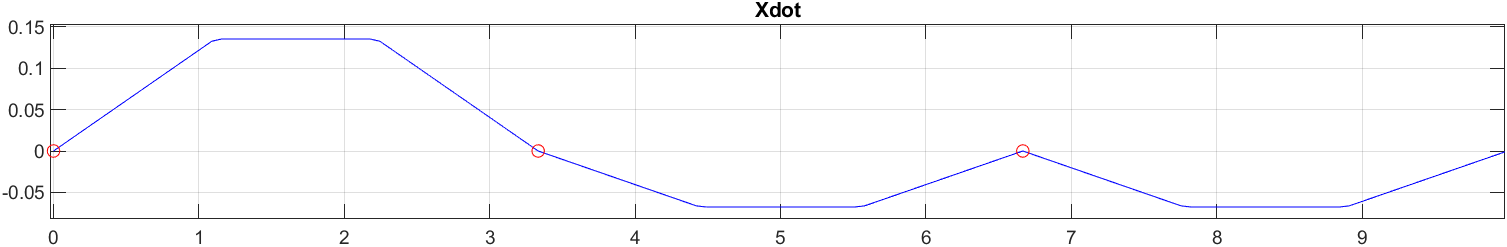


Figure 8



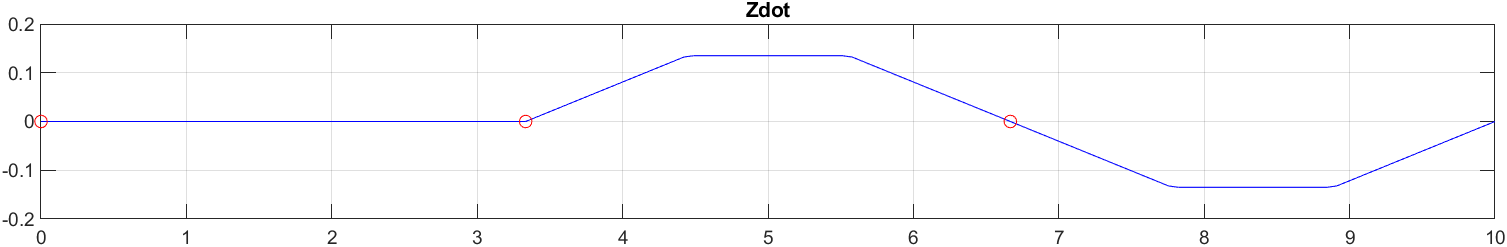


Figure 9

In quest’ultima figura, in particolare, si puó notare la velocitá trapezoidale, con un primo tratto con accelerazione costante, un secondo con accelerazione nulla, ovvero velocitá costante, e un terzo a decelerazione costante, ovvero a velocitá linearmente decrescente. Si puó anche notare che il massimo di velocitá raggiunto è poco sopra 0.1, contro una velocitá di circa 0.5 per il profilo di velocitá polinomiale.