L'interpolazione "lineare"con motori passo passo

In una CNC si ha l'esigenza della movimentazione contemporanea e coordinata di più assi.

Ad esempio, per ottenere un movimento lineare nel piano, dalla posizione iniziale x_i, y_i, l'utensile deve giungere alla posizione finale x_f, y_f seguendo una linea retta. Le posizioni intermedie, se $x_i \neq x_f$, appartengono all'insieme di punti $P = \left\{(x, y) : \forall x,\ x_i < x < x_f,\ y = (x - x_i) \cdot \frac{y_f - y_i}{x_f - x_i} + y_i\right\}$ Se $x_i = x_f$ o se $y_i = y_f$ si tratta di un movimento che coinvolge un solo asse.

Per la movimentazione, l'uso dei motori passo passo unitamente ad una vite, permette di ottenere una successione di posizioni discrete ed equispaziate. Ad esempio, facendo uso di un motore passo passo pilotato con 400 passi/giro ed associato ad una vite di passo 4 mm, risulta che tra un passo e l'altro del motore l'avanzamento è di 0,01 mm. L'esigenza di ottenere un movimento lineare coinvolgendo più assi contemporaneamente si traduce quindi in una approssimazione della traiettoria ideale.

La figura 1 è stata realizzata ipotizzando un avanzamento unitario, una posizione iniziale $(x_i, y_i) = (0, 0)$ e una posizione finale $(x_f, y_f) = (10, 4)$. Sono visibili la traiettoria ideale (linea rossa) e quella meglio approssimata (linea blu) che si ottiene muovendosi lungo x e lungo y attraverso la griglia di un quadretto alla volta.

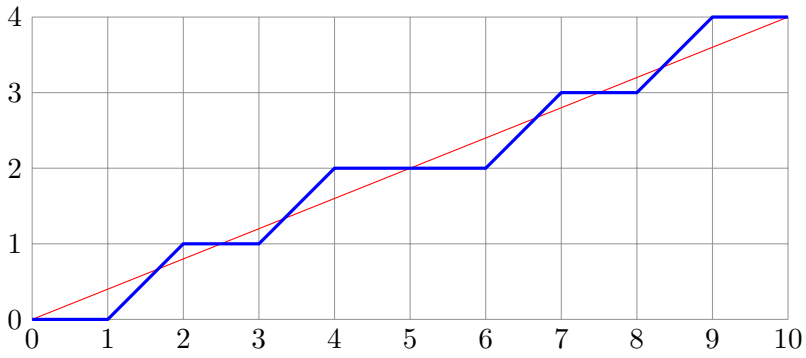

Figura 1: confronto tra traiettoria ideale e traiettoria discreta

La sequenza di posizioni ottimale è:

x	0	1	2	3	4	5	6	7	8	9	10
y	0	0	1	1	2	2	2	3	3	4	4

Tabella 1: successione delle posizioni discrete

Algoritmo per il calcolo della traiettoria ottimale

Le coordinate ottenibili sono necessariamente approssimate alle posizioni discrete determinate dal passo della vite e dal numero di passi del motore. Dalle coordinate di partenza, le coordinate da raggiungere determinano il numero intero di passi da effettuare.

Da esempi grafici bidimensionali (figure 2 e 3) si può osservare che, nell'asse lungo il quale viene effettuato il percorso maggiore, l'avanzamento di quadretto in quadretto non viene mai interrotto. Nell'altro asse l'avanzamento è alternato a delle pause. Si può allora pensare di assegnare all'asse che percorre la maggiore distanza la funzione di *master* e agli altri assi la funzione di *slave*. Lungo

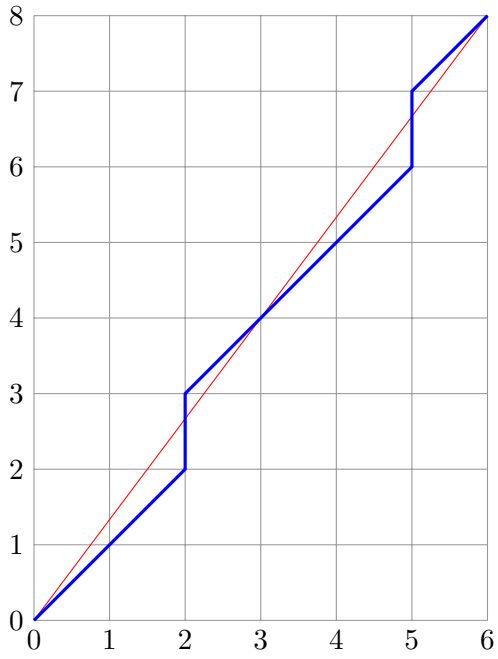

Figura 2: esempio di traiettoria

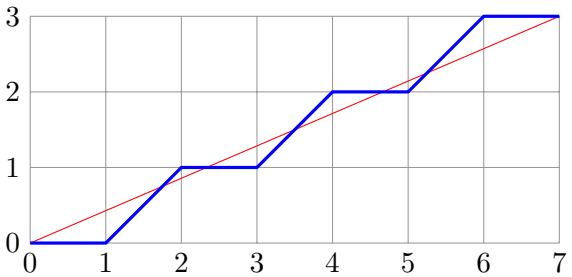

Figura 3: esempio di traiettoria

l'asse master l'avanzamento sarà regolare mentre lungo gli assi slave (indicati genericamente con j) l'avanzamento sarà condizionato dalla posizione del master e dalla distanza che i vari slave devono percorrere in rapporto con la distanza che il master deve percorrere:

$$posizione\,slave_j = posizione\,master \cdot \frac{distanza\,slave_j}{distanza\,master}$$

Utilizzando i passi:

$$passo\,slave_j = arrotonda\left(passo\,master \cdot \frac{passi\,slave_j}{passi\,master}\right)$$

Graficamente, lo step lungo l'asse slave avviene ogni qualvolta la traiettoria ideale (linea rossa) supera o uguaglia la mezzeria del lato del quadretto che taglia uscendo dallo stesso.

L'algoritmo che viene presentato nel seguito permette di ottenere a terminale video la sequenza di posizioni ottimali per un numero di passi a scelta per ciascun asse. Una volta acquisiti i dati da terminale, individua il numero massimo di passi ($maxpassi$) ovvero il percorso più lungo da eseguire. Tale valore viene utilizzato come condizione terminale per un ciclo for nella variabile i all'interno del quale, per ciascun asse, viene applicata l'equazione:

$$passo\,slave_j = arrotonda\left(passo_i \cdot \frac{passi\,slave_j}{maxpassi}\right)$$

```
#include <iostream>
#include <cmath>

//il numero motori puó essere variato a piacimento
#define numero_motori 2

//passi_mot -> array con i passi da fare per i motori
unsigned int passi_mot[numero_motori];

//passo_mot -> array con il passo attuale del motori
unsigned int passo_mot[numero_motori];

//max_passi -> numero massimo di passi
//è dichiarato double nonostante sia un intero senza segno
//affinché nell'operazione di divisione eseguita nel seguito
//venga effettuato il casting delle altre variabili a double
double max_passi;

using namespace std;

int main()
{
    //inserimento dati
    for(int i=0; i<numero_motori; i++)
    {
        cout<<"Inserisci il numero di passi per il motore "<<i<<" ";
        cin>>passi_mot[i];
    }

    //ricerca del massimo numero di passi
    max_passi=passi_mot[0];
    for(int i=1; i<numero_motori; i++)
        if (passi_mot[i]>max_passi) max_passi=passi_mot[i];
```

```
34
35          //calcolo e pubblicazione risultati
36          for(int i=1; i<=max_passi; i++)
37          {
38              for(int j=0; j<numero_motori; j++)
39              {
40                  cout<<round((passi_mot[j]/max_passi)*i)<<"\t";
41              }
42              cout<<"\n";
43          }
44
45          return 0;
46      }
```

Algoritmo "snello" per il calcolo della traiettoria ottimale

L'algoritmo precedentemente proposto fa uso delle operazioni di divisione, moltiplicazione ed arrotondamento. Tali operazioni sono onerose sotto il profilo del tempo richiesto al sistema hardware per produrre il risultato. Di seguito viene proposto un algoritmo che fa uso delle sole operazioni di somma e sottrazione.

Si osservi la figura 4 che ripropone la sequenza di passi della figura 3. In questo esempio l'asse master coincide con l'asse x ma questo non pregiudica la generalità di quanto viene esposto. Come detto, lo step lungo l'asse slave avviene ogni qualvolta la traiettoria ideale (linea rossa) supera o uguaglia la mezzeria del lato del quadretto che taglia uscendo dallo stesso.

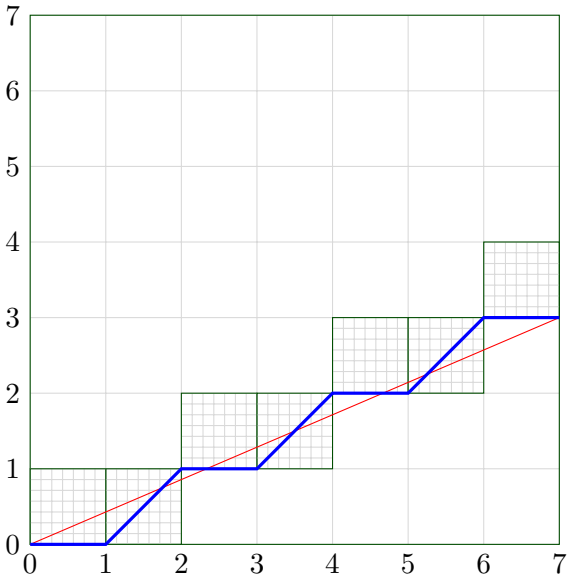

Figura 4: esempio di traiettoria

Ciascun quadretto verde è simile al quadrato di lato 7×7 che racchiude l'intero percorso.

La mezzeria di ciascun quadretto è per similitudine associabile a $\frac{\Delta x}{2}$.

A seguire la sequenza relativa ai primi step:

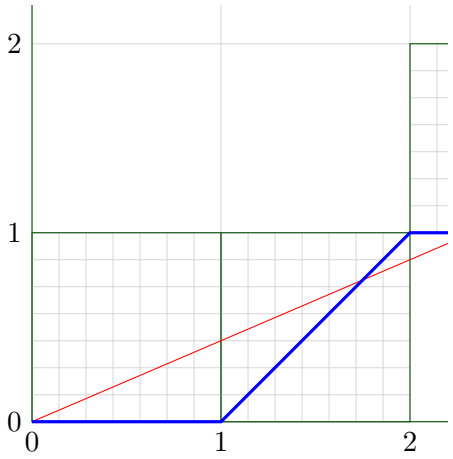

Figura 5: dettaglio del primo e secondo step in x

- primo step in x: osservando il primo quadretto verde ((0,0)(1,0)(1,1)(0,1)) (figura 5), si vede la linea rossa che taglia il lato verticale destro ad una altezza inferiore alla mezzeria ($\frac{3}{7}$ del lato). Lo step sull'asse y non è da eseguire;
- secondo step di x: si osservi il secondo quadretto verde ((1,0)(2,0)(2,1)(1,1). La linea rossa, uscendo dal quadretto intercetta il lato oltre la mezzeria ($\frac{6}{7}$ del lato). In contemporanea allo step di x si deve eseguire lo step in y;
- terzo step di x: si focalizzi l'attenzione sul terzo quadretto verde (figura 6). Intersezione a $\frac{2}{7}$ del lato. Nessuno step in y;
- quarto step di x: Intersezione a $\frac{5}{7}$ del lato. Step in y;

e via così fino al compimento dell'ultimo step in x.

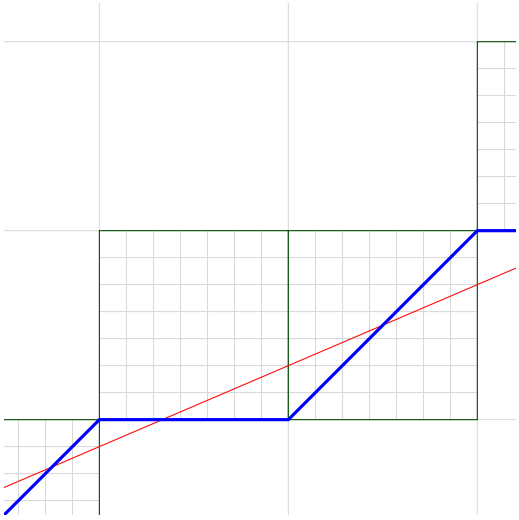

Figura 6: dettaglio del terzo e quarto step in x

Indicando con Δm e con Δs il numero di step che master e slave devono rispettivamente eseguire, il processo descritto è riassumibile nel flowchart di figura 7a.

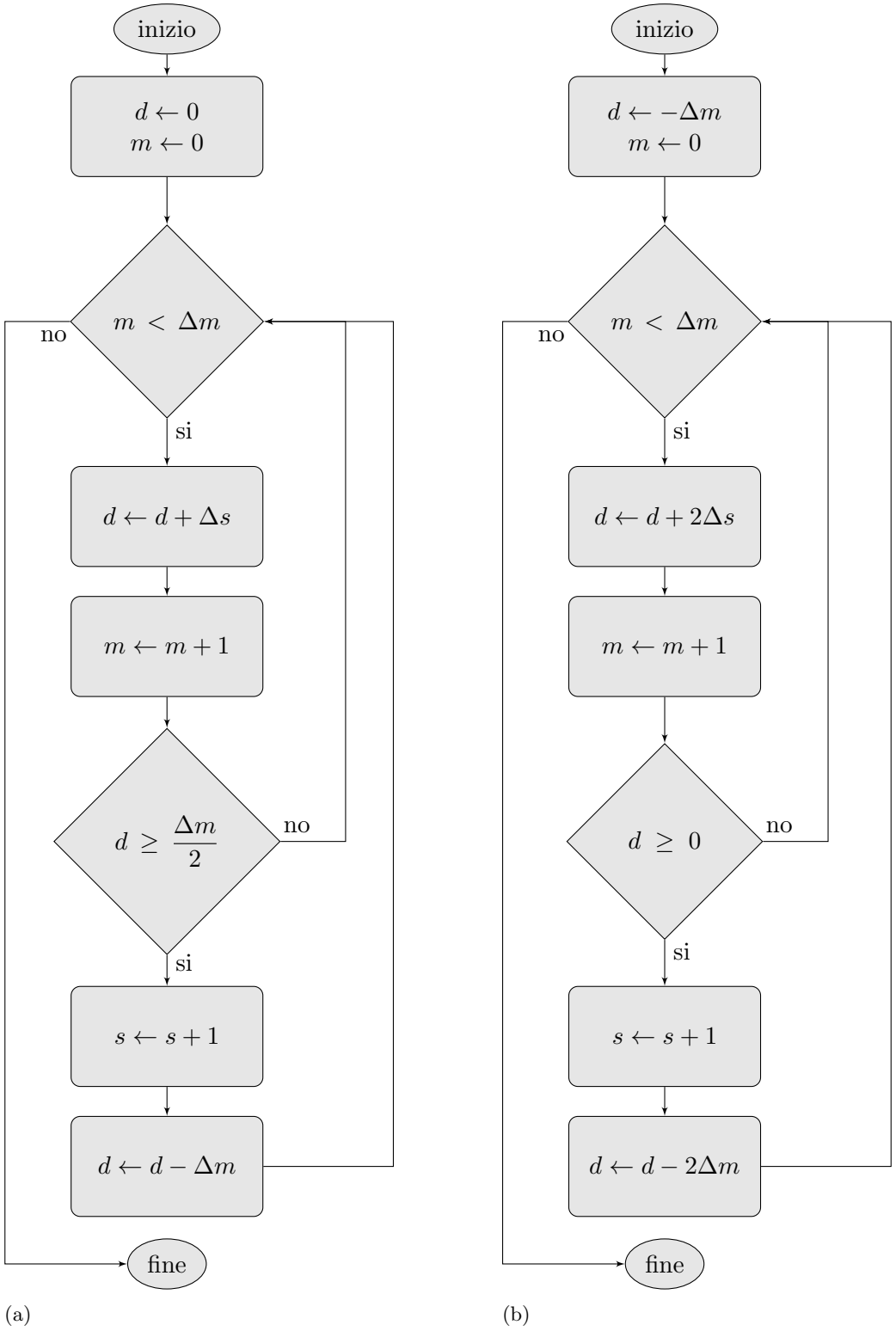

Figura 7: flowchart dell'algoritmo "snello"

Sostituendo il parametro decisionale d con $\frac{d+\Delta m}{2}$ ed eseguendo alcuni adattamenti si ottiene il flowchart di figura 7b.

A seguire il software in C scritto sulla base del flowchart di figura 7b.

```
#include <iostream>

//il numero motori puó essere variato a piacimento
#define numero_motori 2

//passi_mot -> array con i passi da fare per i motori
unsigned int passi_mot[numero_motori];

//passo_mot -> array con il passo attuale del motori
unsigned int passo_mot[numero_motori];

//max_passi -> numero massimo di passi
unsigned int max_passi;

using namespace std;

int main()
{
    //inserimento dati
    for(int i=0; i<numero_motori; i++)
    {
        cout<<"Inserisci il numero di passi per il motore "<<i<<" ";
        cin>>passi_mot[i];
    }

    //ricerca del massimo numero di passi
    max_passi=passi_mot[0];
    for(int i=1; i<numero_motori; i++)
        if (passi_mot[i]>max_passi) max_passi=passi_mot[i];

    //parametro decisionale
    int d=-max_passi;

    //calcolo e pubblicazione risultati
    for(int i=0; i<max_passi; i++)
    {
        for(int j=0; j<numero_motori; j++)
        {
            d+=2*passi_mot[j];
            if (d>=0)
            {
                passo_mot[j]++;
                d-=2*max_passi;
            }
            cout<<passo_mot[j]<<"\t";
        }
        cout<<"\n";
    }
    return 0;
}
```