転ばない

V850マイコン基板を使ってロボットを動かす

二足歩行ロボットの製作（後編）

岡田 紀雄

本誌2007年5月号付属V850マイコン基板を使用して，二足歩行ロボットを製作した．前編ではサーボ･モータの制御方法を，中編ではレート・ジャイロ・センサでバランスをとるA-D変換の処理方法や静歩行による歩行方法などを紹介した．今回は，ロボットの動きを滑らかに動かすために行う近似補間計算などを紹介する．なお，本記事で紹介するアプリケーションのソース・コードなどは本誌のWebサイト（http://www.cqpub.co.jp/interface/）からダウンロードできる．
（編集部）

1. 登録データの連続再生アルゴリズム

すべてを数学的な式で表現し，計算で動作を求めようとすると，大胆な動きの表現がとても難しくなります．そこで歩行以外は，登録した動きを滑らかに動作できる仕組みを取り入れることが望ましいと言えます．サーボごとに近似補間計算を実施すると，マイコンの負荷が大きくなりますが，サーボの制御周期からすると十分に短時間で処理できる計算といえます．ここでは，最も単純な直線近似補間方法と，多項式による近似方法の2種類を紹介します．これらは用途に合わせて使い分けます．

● 直線近似補間による表現の課題

直線近似補間は，慣性力やデータの連続性をあらかじめ理解している場合に計算処理負荷を軽減できる計算方法です．単調な動きで指定した座標（角度）に到達しなくても問題がないような動き，例えば歩行時の腕動作は直線近似で計算するとよいでしょう．

ある時刻と位置を(t, x)と表記した場合，(t_1, x_1)，(t_2, x_2)への動きを直線近似で表現すると，下記の式で計算できます．

$$x = x_1 + (x_2 - x_1) \times \frac{t - t_1}{t_2 - t_1} \quad \cdots\cdots (1)$$

実際に，ロボットの動作において腕の先の位置座標(x, y, z)と時刻tを(x, y, z, t)として表現した場合，(x_1, y_1, z_1, t_1)から(x_2, y_2, z_2, t_2)に動作を移す場合は，

$$\begin{aligned} x &= x_1 + (x_2 - x_1) \times \frac{t - t_1}{t_2 - t_1} \\ y &= y_1 + (y_2 - y_1) \times \frac{t - t_1}{t_2 - t_1} \quad \cdots\cdots (2) \\ z &= z_1 + (z_2 - z_1) \times \frac{t - t_1}{t_2 - t_1} \end{aligned}$$

となります．計算負荷を小さくするために，上記三つの式の共通項$(t - t_1) / (t_2 - t_1)$をttなどに置き換えて計算する場合，ttに丸めが発生します．丸めが発生すると計算誤差が大きくなるので，注意が必要です．誤差を小さくし，かつ計算速度を上げたい場合は，何度も繰り返し計算し，また定数となる$x_2 - x_1$，$y_2 - y_1$，$z_2 - z_1$，$t_2 - t_2$をxx，yy，zz，ttなどの変数に置き換えることにより，丸めの小さい効率的な計算コードを生成できます．

```
{
    xx = x2-x1 ;
    yy = y2-y1 ;
    zz = z2-z1 ;
    while( t < t2 ) {
        ct = t - t1 ;
        x = x1 + (xx * ct) / tt ;
        y = y1 + (yy * ct) / tt ;
        z = z1 + (zz * ct) / tt ;
        /* ここに実際の処理を記述 */
    }
}
```

さらに，上記のように変数ctを定義することで，簡略化できます．

このように，足先や手先の座標の動作を直線近似する方法を使用する場合やサーボの角度を求める場合は，逆運動学による計算が必要になります．一方で，手先だけの運動で記述できないような動作を行いたい場合，例えば手先の位置は動かさないで，ヒジ部だけを動かしたような動作（脇を締める動作）の場合は，サーボごとに動作座標（角度）を定義し，所望の時間帯で近似計算を行って求めることができます．

サーボの角度 r と時刻 t を $(r,\ t)$ として表現した場合，$(r_1,\ t_1)$ から $(r_2,\ t_2)$ に動作を移す場合は，

$$r = r_1 + (r_2 - r_1) \times \frac{t - t_1}{t_2 - t_1} \quad \cdots\cdots (3)$$

と表現します．これにより，任意の時刻 t におけるサーボ角度 r が求められます．

● 3 次多項式による近似補間

時刻と位置が**図 1** の A，B，C，D，E のように座標点が移り変わる場合，移り変わる時間が十分に大きいときは，目標位置に到達してから次の目標に移ることになります．しかし，サーボの回転速度よりも速いときは，目標位置に到達する前に次の目標位置に移る動作に入ってしまいます（変曲点 C，D 部の実際の動作）．つまり，少なくとも 2 点以上先の情報（時刻と座標）が重要になります．現在の時刻における位置が A の場合，B のみでなく C，D（E）の情報が重要ということです．

一般に n 次関数で近似する場合，$n+1$ 個の情報が必要になります．言い換えれば，何個先までの情報を扱って近似するかということになります．四つ（三つ先）の情報を使って近似する場合は 3 次関数となります．

では，何次関数にするのがよいのでしょうか．これはさまざまな議論があります．ある 2 点の位置における座標と速度が分かっている場合に行う Ferguson/Coons 曲線近似と，連続 4 点の位置情報から近似を行う Catmull-Rom 曲線近似がよく使われます．いずれの場合も，四つの情報から補間を行うので，3 次関数ということになります．

Ferguson/Coons 曲線は，**図 1** の B，C の位置における座標と速さを $(x_n,\ v_n)$，$(x_{n+1},\ v_{n+1})$ と表した場合，

$$\begin{aligned} x(t) &= \alpha t^3 + \beta t^2 + \gamma t + \delta \\ \alpha &= 2x_n + v_n - 2x_{n+1} + v_{n+1} \\ \beta &= -3x_n - 2v_n + 3x_{n+1} - v_{n+1} \\ \gamma &= v_n \\ \delta &= x_n \end{aligned} \quad \cdots\cdots (4)$$

となります．動歩行における x 軸方向の計算はこの近似が効果的です．

Catmull-Rom 曲線は，位置 B，C の速さを位置 A，D との差分として表現する曲線です．位置 A，B，C，D を x_{n-1}，x_n，x_{n+1}，x_{n+2} として表した場合，ある時刻 t における B-C 間の位置 $x(t)$ は，

$$\begin{aligned} x(t) &= \frac{1}{2}(\alpha t^3 + \beta t^2 + \gamma t + \delta) \\ \alpha &= -x_{n-1} + 3x_n - 3x_{n+1} + x_{n+2} \\ \beta &= 2x_{n-1} - 5x_n + 4x_{n+1} - x_{n+2} \\ \gamma &= -x_{n-1} + x_{n+1} \\ \delta &= 2x_n \end{aligned} \quad \cdots\cdots (5)$$

となります．動作の位置情報を持たせたテーブルを用意して，ロボットを動作させるような場合は，こちらが効果的です．n=0，つまり最初の区間で近似曲線を得たい場合は，$x_{n-1} = x_n = x_0$，$x_{n+1} = x_1$，$x_{n+2} = x_2$ とします．また，$n+1$ が最終点の場合は，$x_{n+2} = x_{n+1}$ として計算します．**図 1** における A-B 間と D-E 間の近似に相当しますが，きれいに近似できています．

2. 高速化の検討と動作確認

ここでは，マイコンの処理負荷を全体的に下げ，高速化を図る検討や逐次動歩行計算した際の歩行動作，登録モーションを補間しながら動作している様子を紹介します．

● V850ES/JG2 のレジスタの活用

V850ES/JG2 を含め V850 シリーズ用に開発された CA850 は，いろいろなレジスタを外部レジスタ変数としてユーザ定義できる仕組みを提供しています．この外部レジスタ変数を使う場合は，1）C コンパイラで使うレジスタ数を減らす（**図 2**），2）外部レジスタ変数定義をする（**図 3**），の二つが必要です．1）は，「コンパイラ・オプションの設定」ダ

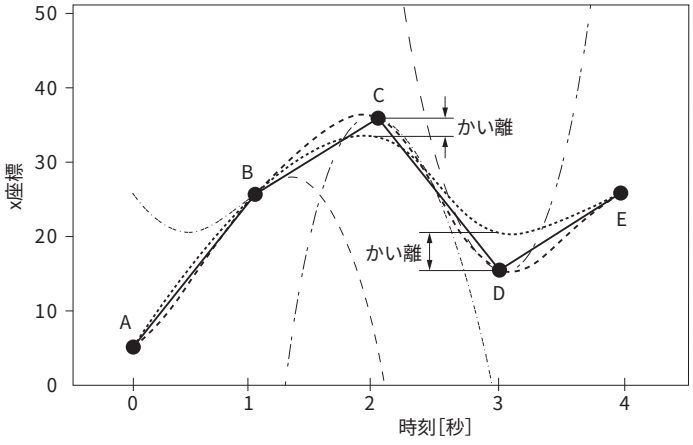

図 1　直線近似と 3 次関数近似による動作補間計算

点 A，B，C，D，E の座標に対して直線近似補間を行った場合，点 C，D について目標座標とかい離している．A-B, B-C, C-D, D-E 間の近似を合わせた形を逐次 3 次関数近似として示す．点 A の前の点は A と同じとし，点 E の次の点は E と同じとして計算する．