インタラクティブシステム論 第9回

梶本裕之
Twitter ID kajimoto
ハッシュタグ #ninshiki

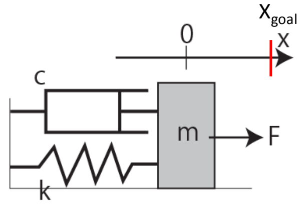
日程

4/10 インタラクティブシステム入門
4/17 Scilab入門
4/24 フーリエ変換
5/1 出張
5/8 フーリエ変換と線形システム
5/15 出張
5/22 信号処理の基礎
5/29 出張
6/5 信号処理応用1（相関）
6/12 信号処理応用2（画像処理）
6/19 ラプラス変換
6/26 中間確認レポート
7/3 古典制御の基礎
7/10 行列
7/17 行列と最小二乗法
7/24 ロボティクス
8/2〜8 期末テスト

7/17のみ講義室が A201教室に変更

制御の基礎の基礎

制御をしたい

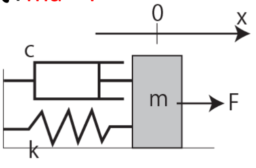


時刻0に，x=0の位置にあったおもりmを，
$x=X_{goal}$に移動したい．

どのような力F(t)を加えたらよいだろうか？

モデルを作る

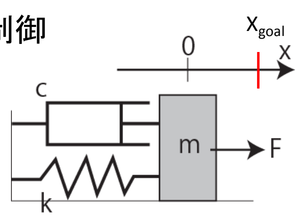
ニュートンの運動方程式：ma = F
おもりに加わる力
- 外力（制御入力）：*f*
- ダンパ：*-cv*
- バネ： *-kx*



運動方程式：

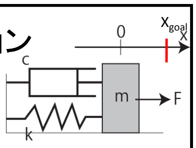
2階微分まで考えるシステム＝2次系
多くのシステムの近似モデルとして適用可能．

ON・OFF制御



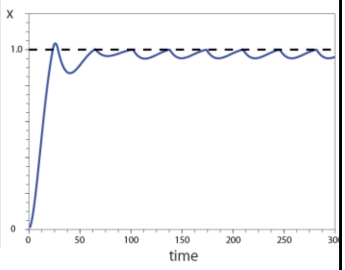
- 一番初めに考える制御
- 目的の位置より手前だったらF=1を加える．
- 目的の位置を超えたらF=0とする．

ON/OFF制御のシミュレーション



Scilabコード

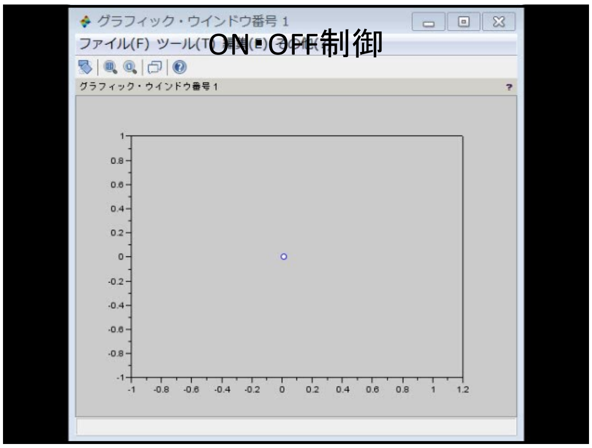
```
m=1.0;          //質量
c=1.0;          //ダンパ
k=1.0;          //バネ
xgoal=1.0;      //目標値
x=0;            //現在地
v=0;            //現在速度
dt = 0.1;       //時間間隔
xrecord =[];    //データ記録用
for t=1:300,

        a = F - k*x - c*v;
        v = v+a*dt;
        x = x+v*dt;
        xrecord = [xrecord,x];
end
plot(xrecord);
```



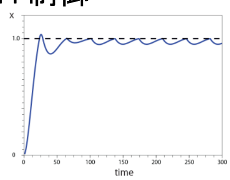
Scilabでアニメーション

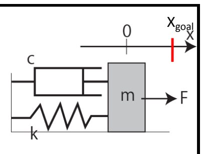
コードを最後に追加

```
// Draw initial figure
figure(1);
plot(xrecord(1),0,'o');
h_compound = gce();
h_compound.children.mark_size = 20;
h_compound.children.mark_background = 2;
h_axes = gca();
h_axes.data_bounds = [-1.5,-1.5;1.5,1.5];

// Animation Loop
i = 1;
while i<=length(xrecord)
    drawlater();
    h_compound.children.data = [xrecord(i),0]; drawnow();
    i = i+1;
    sleep(10);
end
```



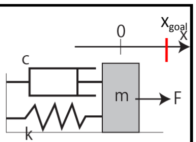
ON・OFF制御





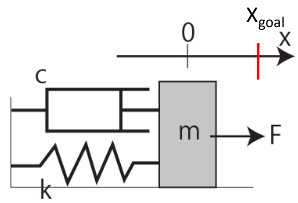
- 制御の最低限は含まれている：
  （1）対象の状態を見て，
  （2）目的との差を見て
  （3）出力を変化させる
- 目標値付近で永久に発振してしまう
- ごく簡単なハードウエアで実現できる
- 実装例：こたつ（ただしヒステリシスを入れている）

フィードバックとは



- 制御の最低限：
  （1）対象の状態を見て，
  （2）目的との差を見て
  （3）出力を変化させる
- これを，「フィードバック制御」という

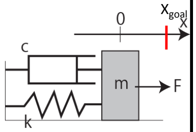
P制御



- ON/OFFだと発振してしまった．
- Fをもっと「なだらか」に変化させれば…

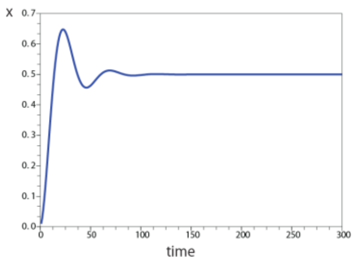
- 目標値X_{goal}と現在値xとの「差」に比例した力を加えればよいのでは？
  (比例＝Proportional)

P制御のシミュレーション

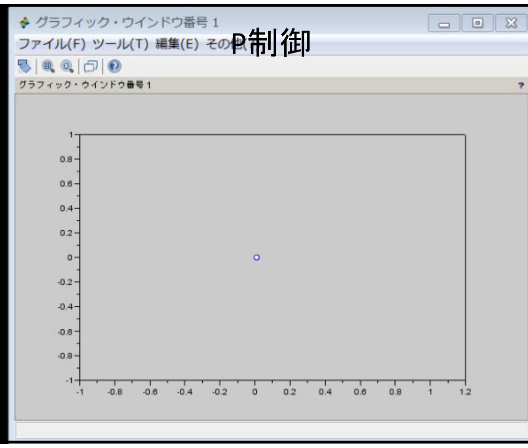


Scilabコード

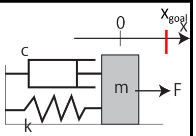
```
m=1.0;          //質量
c=1.0;          //ダンパ
k=1.0;          //バネ
xgoal=1.0;      //目標値
x=0;            //現在地
v=0;            //現在速度
dt = 0.1;       //時間間隔
xrecord =[];    //記録用

for t=1:300,
  F=
  a = F - k*x - c*v;
  v = v+a*dt;
  x = x+v*dt;
  xrecord = [xrecord,x];
end

plot(xrecord);
```

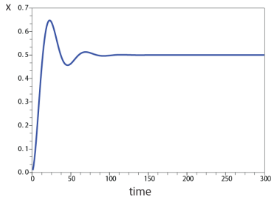


P制御



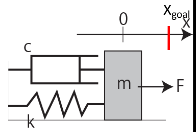
P制御のシミュレーション





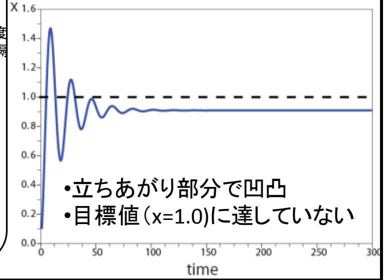
- 立ちあがり部分で凹凸
- 目標値(x=1.0)に達していない

比例定数を大きくしてみる



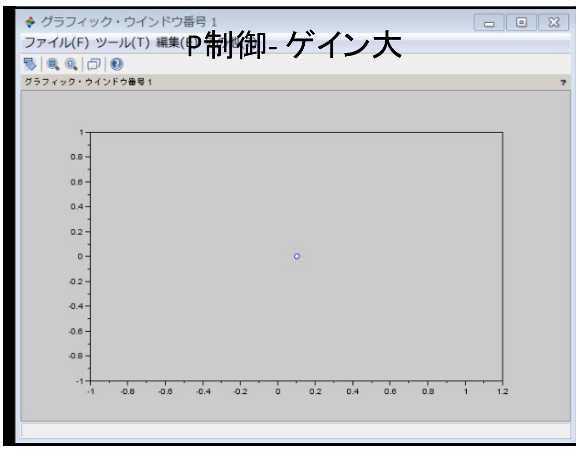
Scilabコード

```
m=1.0;          //質量
c=1.0;          //ダンパ
k=1.0;          //バネ
xgoal=1.0;      //目標値
x=0;            //現在地
v=0;            //現在速度
dt = 0.1;       //時間間隔
xrecord =[];    //記録用

for t=1:300,
  F=10.0 * (xgoal-x);
  a = F - k*x - c*v;
  v = v+a*dt;
  x = x+v*dt;
  xrecord = [xrecord,x];
end

plot(xrecord);
```

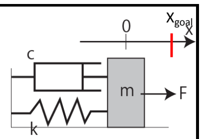


- 立ちあがり部分で凹凸
- 目標値(x=1.0)に達していない

P制御- ゲイン大

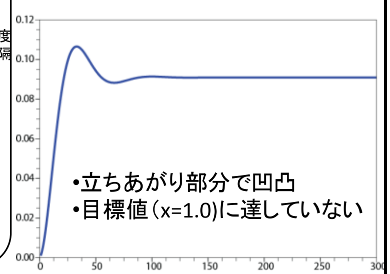


比例定数を小さくしてみる



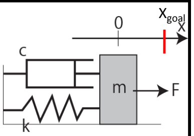
Scilabコード

```
m=1.0;          //質量
c=1.0;          //ダンパ
k=1.0;          //バネ
xgoal=1.0;      //目標値
x=0;            //現在地
v=0;            //現在速度
dt = 0.1;       //時間間隔
xrecord =[];    //記録用

for t=1:300,
  F=0.1 * (xgoal-x);
  a = F - k*x - c*v;
  v = v+a*dt;
  x = x+v*dt;
  xrecord = [xrecord,x];
end

plot(xrecord);
```



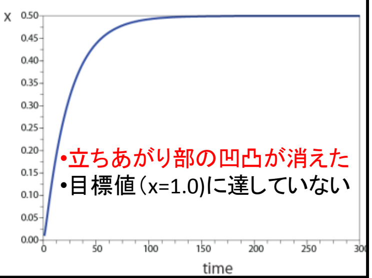
- 立ちあがり部分で凹凸
- 目標値(x=1.0)に達していない

ダンパが大きかったら？



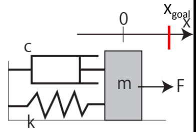
Scilabコード

```
m=1.0;          //質量
c=5.0;    //ダンパ
k=1.0;          //バネ
xgoal=1.0;      //目標値
x=0;            //現在地
v=0;            //現在速度
dt = 0.1;       //時間間隔
xrecord =[];    //記録用

for t=1:300,
  F=1.0 * (xgoal-x);
  a = F - k*x - c*v;
  v = v+a*dt;
  x = x+v*dt;
  xrecord = [xrecord,x];
end

plot(xrecord);
```



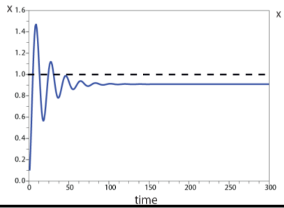
- 立ちあがり部の凹凸が消えた
- 目標値(x=1.0)に達していない

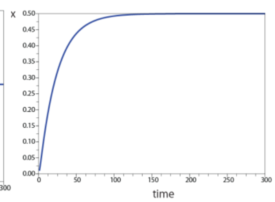
P制御のまとめ



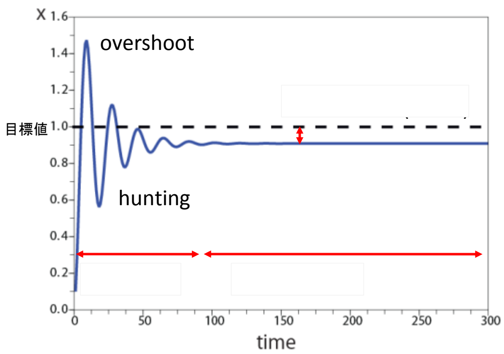
どうやら．．．

- 最終的に目標値(x=1.0)に達しない？
- はじめに振動するかどうかは，ダンパの大きさに依存？

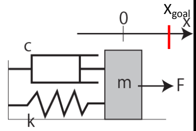




用語



P制御の数学



- システム

$$m\ddot{x}+c\dot{x}+kx=f$$

- 力の制御の仕方

$$f=$$

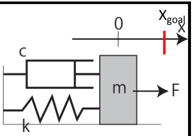
- つまり

$$m\ddot{x}+c\dot{x}+kx=$$

- 一般化して次の式を考えよう

$$\ddot{x}+a\dot{x}+bx=c$$

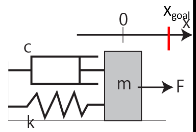
P制御の数学



$$\ddot{x}+a\dot{x}+bx=c$$

- ラプラス変換すると
- つまり，軌道x(t)のラプラス変換X(s)は，
- 2次方程式の根をλ1，λ2とすると，

P制御の数学

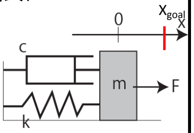


- 2次方程式の根をλ1，λ2とすると，
- 部分分数分解をすると，
- 結局，逆ラプラス変換をすると

つまり，

- <u>二次方程式の根λ1，λ2が過渡的なふるまいを決定し，</u>
- <u>定数項a1が，収束値を決定する．</u>

P制御の数学（２次方程式の根）



•2次方程式の根λ1，λ2について

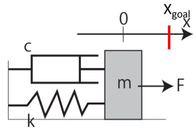
$$m\ddot{x}+c\dot{x}+kx=a(x_{goal}-x)$$

•根λ1，λ2は,$ms^2+cs+(k+a)=0$の根．

$$\lambda=\frac{-c\pm\sqrt{c^2-4m(k+a)}}{2m}$$

•高校生でもわかることが2つ！
•（1）λの実部は負である
•（2）c^2が4m(k+a)よりも小さいと，λは虚部を持つ

P制御の数学（２次方程式の根）



$$\lambda=\frac{-c\pm\sqrt{c^2-4m(k+a)}}{2m}$$

（1）λの実部は負である

だから，

$$x(t)=a_1+a_2\exp(\lambda_1 t)+a_3\exp(\lambda_2 t)$$

のexpの項はすぐに減衰する．

つまり，無限大に発散することはない（ひと安心！）

P制御の数学（２次方程式の根）

$$\lambda=\frac{-c\pm\sqrt{c^2-4m(k+a)}}{2m}$$

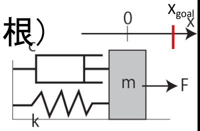
（2）c^2が4m(k+a)よりも小さいと，λは虚部を持つ

このとき，

$$x(t)=a_1+a_2\exp(\lambda_1 t)+a_3\exp(\lambda_2 t)$$

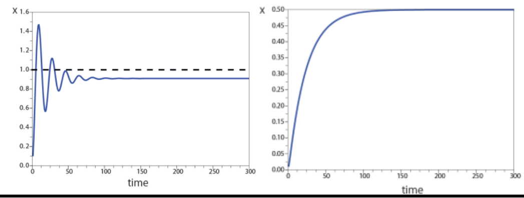
の，expの項は，exp($-c_1t+ic_2t$)＝exp($-c_1t$)・exp(ic_2t)
つまり，
•減衰する成分exp($-c_1t$)と，
•振動する成分exp(ic_2t)=cos(c_2t)+isin(c_2t)
に分けられる．
これこそが，はじめの振動の原因

P制御の数学（２次方程式の根）

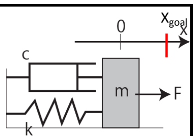


$$\lambda=\frac{-c\pm\sqrt{c^2-4m(k+a)}}{2m}$$

（2）c^2が4m(k+a)よりも小さいと，λは虚部を持つ
これこそが，はじめの振動の原因
つまり，「振動を抑えるためにはダンパ（ブレーキ）を大きくすればよい」ということ．



P制御の数学（再掲）



•2次方程式の根をλ1，λ2とすると，

$$X=\frac{c}{s(s-\lambda_1)(s-\lambda_2)}$$

•部分分数分解をすると，

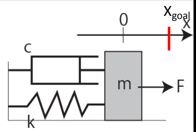
$$X=\frac{a_1}{s}+\frac{a_2}{(s-\lambda_1)}+\frac{a_3}{(s-\lambda_2)}$$

•結局，逆ラプラス変換をすると

$$x(t)=a_1+a_2\exp(\lambda_1 t)+a_3\exp(\lambda_2 t)$$

•つまり，
•二次方程式の根λ1，λ2が過渡的なふるまいを決定し，
•定数項a1が，収束値を決定する．

P制御の数学（定数項）



$$X=\frac{c}{s(s-\lambda_1)(s-\lambda_2)}$$

$$X=\frac{a_1}{s}+\frac{a_2}{(s-\lambda_1)}+\frac{a_3}{(s-\lambda_2)}$$

•結局，逆ラプラス変換をすると

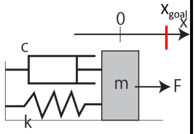
$$x(t)=a_1+a_2\exp(\lambda_1 t)+a_3\exp(\lambda_2 t)$$

•a1以外は時間がたてば消えるので，a1が収束値となる．

a1は部分分数分解を頑張らなければ求められない？

NO！

P制御の数学（定数項）



•システムの応答の式：

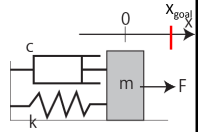
$$m\ddot{x}+c\dot{x}+kx=a(x_{goal}-x)$$

•いま考えたいのは「定常的」になった時だから，

よって

すなわち

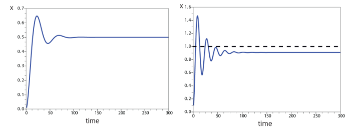
P制御の数学（定数項）



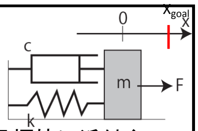
収束値： $x=\dfrac{a}{k+a}x_{goal}$

各定数の意味は，k:ばね定数，a:P制御の比例定数

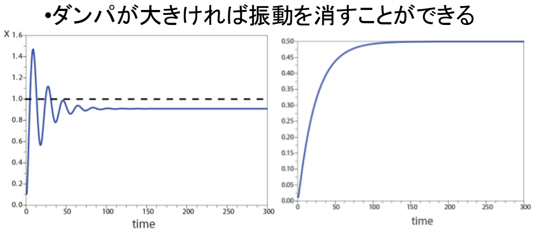
よって次のことがわかる
（1）aを大きくすれば，収束値は目標値に近付く
（2）しかし，収束値は目標値より常に小さい
（3）ただし「ばね成分」が0ならちゃんと収束する．



P制御のまとめ（再掲）

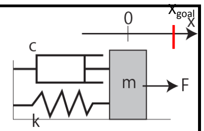


•最終的に目標値（x=1.0)には達しない
　•ただし，比例ゲインが大きければ目標値に近付く
•はじめに振動するかどうかは，ダンパの大きさに依存．
　•ダンパが大きければ振動を消すことができる



ラプラス変換によって，数学的に理解できた

レポート課題(1)

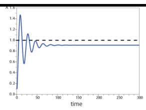


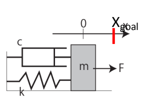
•P制御で振動しないためのダンパcの大きさの数値的な条件を求めよ．
ただし，k=1，m=1とする．

•ダンパをその値周辺(例えば±0.5)に設定したシミュレーションを行い，実際に振動が抑えられることを確認せよ．

•すべてコード中に記載すること．

PI制御





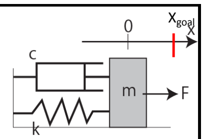
•P（比例）制御は，実は絶対に目標値に収束しない

•これを改善するため，「積分（Integral)」を用意する．

•PI制御：
　•目標値との誤差（P）成分と，
　•その誤差成分の時間的な累積(I)とを，

　•適当な係数で足し合わせて制御信号とする．

PI制御のシミュレーション



P成分：目標位置と現在位置の誤差
I成分：誤差の積分（累積）

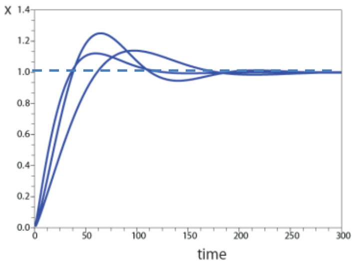
Scilabコード

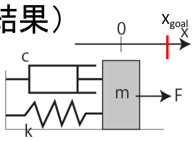
```
m=1.0;     //質量
c=5.0;     //ダンパ
k=1.0;     //バネ
xgoal=1.0;  //目標値
x=0;       //現在地
v=0;       //現在速度
dt = 0.1;   //時間間隔
xrecord =[];//データ記録用
P=0.8; //P成分
I=0.05; //I成分
i_seibun= 0;
```

```
for t=1:300,
  p_seibun =
  i_seibun =
  F=
  a = F - k*x - c*v;
  v = v+a*dt;
  x = x+v*dt;
  xrecord = [xrecord,x];
end

plot(xrecord);
```

I成分のイメージ：
誤差があると，どんどん累積して無視できなくなる

PI制御のシミュレーション（結果）

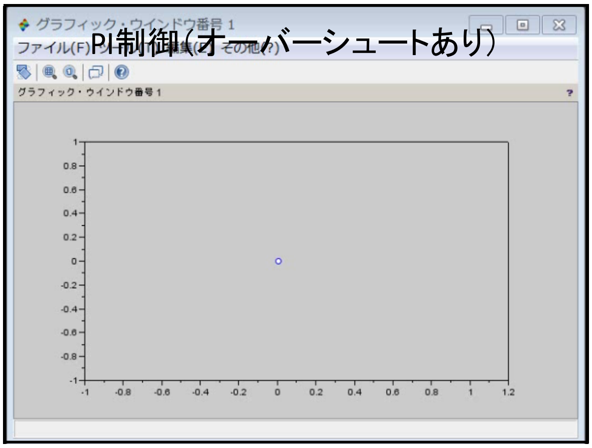




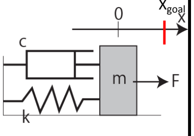
P成分とI成分を色々変えてみた.

●確かに，収束値は目標値と一致する.
●ただし目標値を行きすぎる（振動成分を持つ）こともあるようだ.

PI制御（オーバーシュートあり）



PI制御の数学（最終状態について）



•システム
$m\ddot{x}+c\dot{x}+kx=f$
•制御方式
•つまり
•両辺を微分して
•時間が無限に経過した定常状態では
•よって，最終的に目標値に一致する.

PI制御

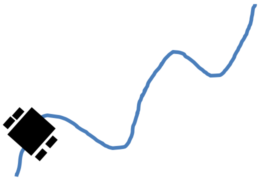
•最終的には目標値に一致する.
•比例ゲインが大きいと振動する.
•比例ゲインが小さいと収束は遅い.

•ゆっくりでもよいから完全に目標値に合わせたいときに使う.

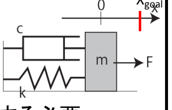
•（例）温度制御
•クーラー
•化学プラント

PI制御のデメリット

•Iは積分，すなわち時間遅れを含む.
•ある一定の目標に達するのが目標ならOK.
だが，目標値が時々刻々と変化する（軌道に沿って動かす等）場合，I成分による時間遅れが問題となる.

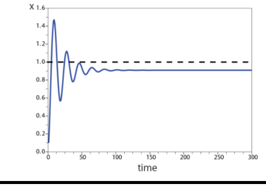


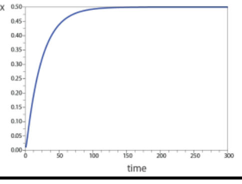
PD制御



•P（比例）制御では，
振動を抑えるために大きなダンパを用意する必要.
•言いかえれば，システムを物理的に変える必要.

•これを改善するため，バーチャルなダンパ（ブレーキ）を用意する.





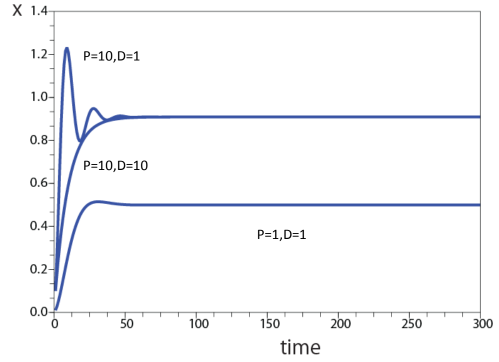
PD制御のシミュレーション

- •P成分：目標値と現在地の差に比例
- •D成分：速度に比例したブレーキ．つまりダンパ

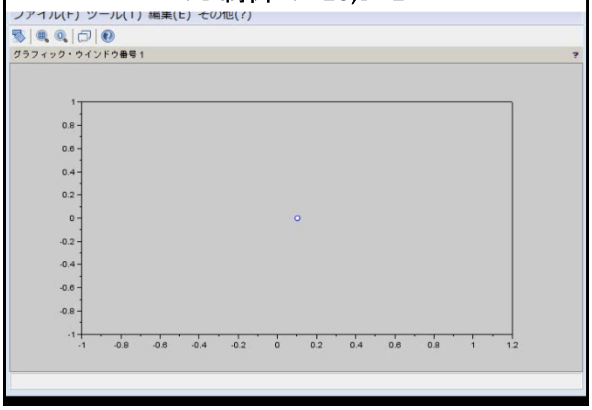
Scilabコード

```
m=1.0;        //質量
c=1.0;        //ダンパ
k=1.0;        //バネ
xgoal=1.0;    //目的地
x=0;          //現在地
v=0;          //現在速度
dt = 0.1;     //時間間隔
xrecord =[];  //データ記録用
P=1.0;        //P成分
D=1.0;        //I成分

for t=1:300,
  F=
  a = F - k*x - c*v;
  v = v+a*dt;
  x = x+v*dt;
  xrecord = [xrecord,x];
end

plot(xrecord);
```

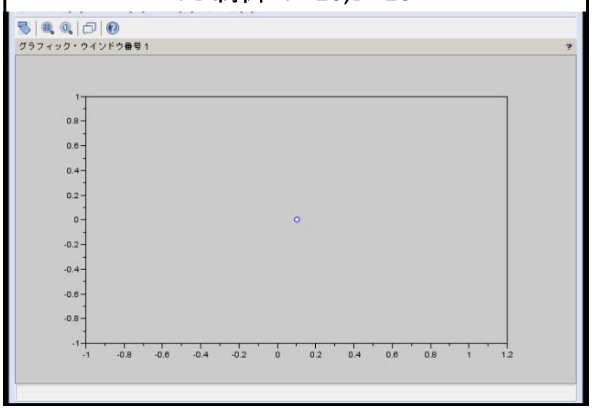
PD制御のシミュレーション(結果)



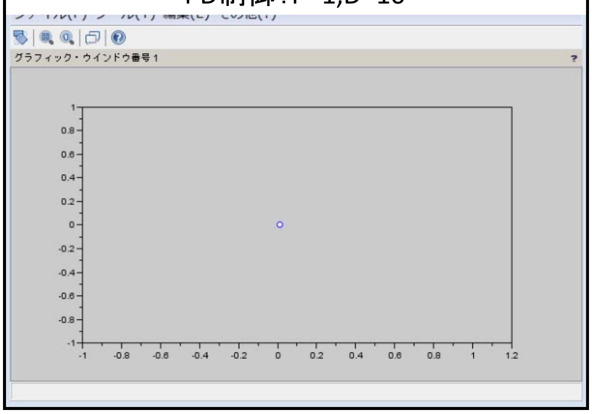
PD制御：P=10,D=1



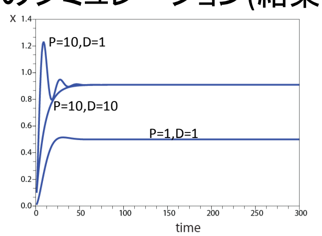
PD制御：P=10,D=10



PD制御：P=1,D=10

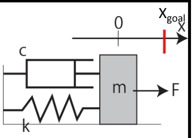


PD制御のシミュレーション(結果)



- •（振動について）
  - •Dを大きくすれば振動が抑えられる．
  - •でもPを大きくしたらやっぱり振動する．
  - •それでもDをもっと大きくすれば振動は抑えられる．
- •（収束について）
  - •P制御と同様に定常偏差がある（目標値に収束しない）

PD制御の数学



- システム

$$m\ddot{x}+c\dot{x}+kx=f$$

- 力の制御の仕方
- つまり
- これは，P制御において，ダンパ成分がcからb+cに増えたことを意味する．

PD制御の数学

- これは，P制御において，ダンパ成分がcからb+cに増えたことを意味する．

$$m\ddot{x}+(b+c)\dot{x}+kx=a(x_{goal}-x)$$

- P制御で振動しない条件は：$\lambda=\dfrac{-c\pm\sqrt{c^2-4m(k+a)}}{2m}$ が虚部を持たないことだった．つまり，

$$c^2-4m(k+a)\geq 0$$

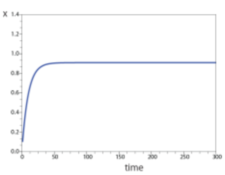
- これが，PD制御では，ダンパ成分が増えたことにより，

$$(b+c)^2-4m(k+a)\geq 0$$

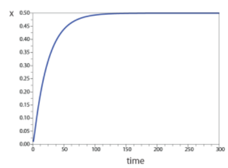
- つまり，より振動しにくくなった．
- P制御と同じだから定常偏差が残る．

PD制御の利点

- ダンパが等価的に増えることで，「振動しにくくなる」
- だからPゲインを思い切り上げられる
- だから結果としてP制御に比べて目標到達速度が速い
- 厳密に目標値に達することよりも，高速に移動することが目標の場合に多く使われる．(例)ロボットアーム



PD制御：P=10,D=10



P制御：P=0.5

モータとPD制御

（再考）収束値：$x=\dfrac{a}{k+a}x_{goal}$

- モータには通常「ばね成分」はない．
- よって収束値が目標値とずれるという問題が生じにくい
- このため，「目標値が時々刻々と変化する」（＝軌道に沿って動かす）場合，PD制御が多く使われる．

重力

（再考）収束値：$x=\dfrac{a}{k+a}x_{goal}$

- モータには通常「ばね成分」はない．
- しかし実際には，アームの姿勢によって，重力による復元力が働く．
- これはばね項とみなせる．
- よって，ただのPD制御では，収束値が目標値に達しないことが多い．
- ⇒「重力補償項」を考える．

現在の姿勢で必要な「重力に打ち勝つ力」を計算し，指令信号に加えることで，重力を無視した制御が可能．

PID制御

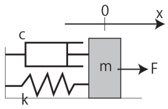
ここまでのすべてを合わせたもの．

- P:現在の値と目標値との誤差（比例成分＝Proportional）
- I:誤差の積分(Integral)
- D:速度(微分成分＝Derivative)

それぞれの役割は

- P:早く目標に達する．
- I:定常偏差を無くす
- D:振動を抑える．

（再考）伝達関数と制御



$$m\ddot{x}+c\dot{x}+kx=f$$

両辺をラプラス変換すると

$$(ms^2+cs+k)X=F$$

$$X(s)=\frac{F}{ms^2+cs+k}$$

つまり，制御とは，元のシステム

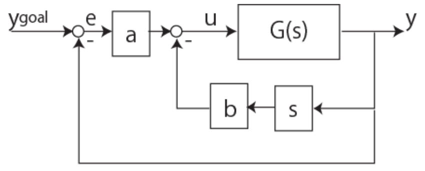
$$G(s)=\frac{1}{ms^2+cs+k}$$ （伝達関数）

に，適切な入力 $F(s)$ を与えて，

望ましい軌道 $X(s)$ を得る操作である．

ブロック線図

制御の流れを図にしたもの．微分をs，積分を1/sと書く．



ygoal：目標値，y：現在の値，e：誤差，u：システムへの入力
a:比例ゲイン，b：微分ゲイン，G:システムの伝達関数

流れに従って考えれば，自然にラプラス変換表現が得られる．
上図はPD制御の場合．

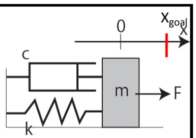
レポート課題（1）

これまでと同じバネマスダンパ系に対して
PID制御を実装したうえで，P,I,Dの係数を変化させ，
（1）なるべく早く目標値に達し，
（2）振動しない（オーバーシュートがない）
ようにせよ．

※実は大きければ大きいほどよくなる。実際の制御では、<u>出力の制限</u>が制御性能をほとんど決めてしまう。
※余裕があれば適当に出力の最大値の制限を設けたうえで試してみよ。

インピーダンス制御

一般化：インピーダンス制御



•システム（バネ成分は無いことも多い）

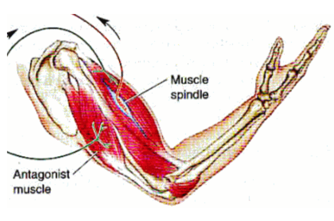
$$m\ddot{x}+c\dot{x}+kx=f$$

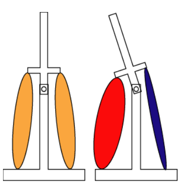
•力の制御の仕方

•つまり

•これは，元のインピーダンスm,c,kを，m+m′, c+c′, k+k′に変化させたことを意味する．

人間＝インピーダンス可変ロボット





•一つの自由度(関節)に対して，二つの筋肉で駆動（拮抗筋）

•力とインピーダンス(やわらかさ)の二つの情報を出力している．
•筋Aと筋Bの差＝外力
•筋Aと筋Bの和＝柔らかさ

力覚ディスプレイ＝
インピーダンス提示装置