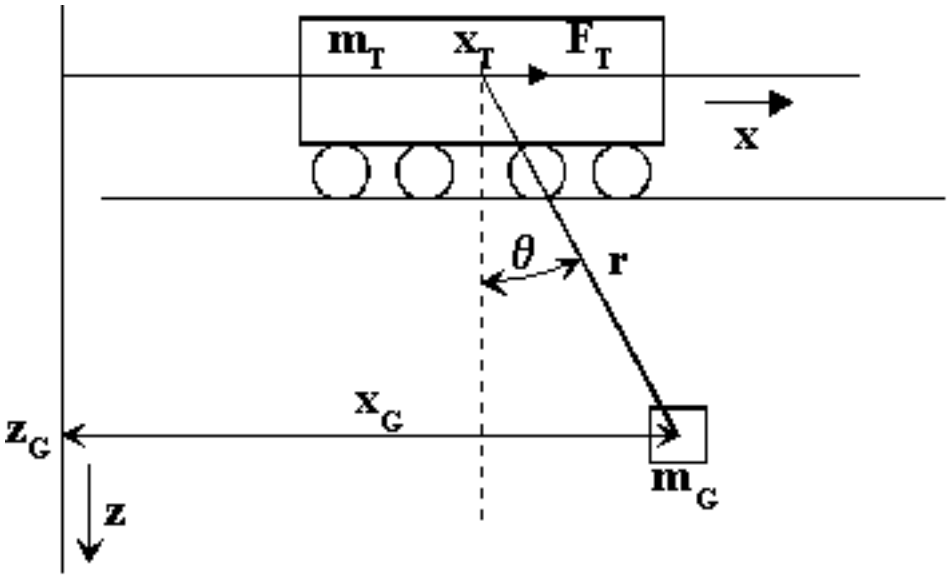
7. 制御設計の例

7.1 ローディングブリッジの制御装置

はじめに

```
> restart:
```

ローディング・ブリッジは、負荷をある地点から別の地点に運びます。
台車の加速と減速は好ましくない振動を発生してしまいます。
そのため負荷はさらに安定し難くなり、時間もかかってしまいます。
負荷がある地点から他の地点へ素早く移動し、すみやかに安定するような制御装置を設計します。



問題の定義

ローディング・ブリッジのパラメータは以下の通りです。

x_K	台車位置 [m] ,
m_K	台車の質量 [kg] ,
F_K	台車の駆動力 [N] ,
x_G, z_G	負荷の位置 [m] ,
m_G	負荷の質量 [m] ,
r	ロープの長さ [m] ,
theta	ロープの角度,
g	重力による加速度 $\left[\frac{m}{s^2}\right]$.

台車の駆動は一次遅れの装置のように振る舞い、その特性は増幅 $K_M\left[\frac{N}{V}\right]$ と時定数 $T_M\ [s]$ により決まります。
パラメータの値は、以下のとおりです。

```
> val:= m[K]=1000, m[G]=4000, r=10, g=9.81, K[M]=100, T[M]=1:
```

運動の方程式の決定

台車の加速度は、力を考慮することで得られます。
Sはロープを通して台車に伝播する力です。

```
> eq1:=m[K]*diff(x[K](t),t$2)=F[K](t)+S*sin(theta(t)):
```

負荷についての垂直方向と水平方向の力は、以下の通りです。

```
> eq2:=m[G]*diff(x[G](t),t$2)=-S*sin(theta(t)):
> eq3:=m[G]*diff(z[G](t),t$2)=m[G]*g-S*cos(theta(t)):
```
負荷の座標は、以下の通りです。
```
> eq4:=x[G](t)=x[K](t)+r*sin(theta(t)):
> eq5:=z[G](t)=r*cos(theta(t)):
```
これらの方程式を2回微分（ロープの長さは一定と仮定）します。
次に、 eq2 および eq3 に代入します。
これで以下が分かります。
```
> s1:=diff(eq4,t,t);
  s2:=diff(eq5,t,t);
  s3:=subs(s1,eq2);
  s4:=subs(s2,eq3);
```
最後の方程式を解き S を求めます。
次に、 s3 に代入します。
```
> s5:=S=solve(s4,S);
  tmp:=subs(s5,s3);
```
この方程式に $\frac{\cos(\mathrm{theta}(t))}{m_G}$ を掛け、方程式の右辺を左辺に移項します。
これで、位置 x[K] と角度 theta にのみ依存する微分方程式が得られます。
```
> tmp2:=expand(tmp/m[G]*cos(theta(t)));
  deq1:=simplify(lhs(tmp2)-rhs(tmp2))=0;
```
方程式 eq2 を方程式 eq1 に代入します。
その結果を方程式 s1 に代入します。
位置 x_K と角度 theta にのみ依存する２番目の微分方程式を得ます。
```
> tmp4:=algsubs(rhs(eq2)=lhs(eq2),eq1);
> deq2:=expand(subs(s1,tmp4));
```
$\sin(\mathrm{theta}(t))$ を $\mathrm{theta}(t)$ および $\cos(\mathrm{theta}(t))$ を１に設定することで微分方程式を線形化します。
角速度は小さいので $\frac{\mathrm{d}}{\mathrm{d}t}\,\mathrm{theta}(t)$ の２次の項は無視できます。
```
> lindeq1:=subs(sin(theta(t))=theta(t),cos(theta(t))=1,diff(theta
  (t),t)^2=0,deq1);
> lindeq2:=subs(sin(theta(t))=theta(t),cos(theta(t))=1,diff(theta
  (t),t)^2=0,deq2);
```
台車を駆動すると一次遅れの装置のように振る舞うので、次の微分方程式によって記述することができます。
(ここで K_M は増幅で T_M は時定数です。)
```
> deq3:=T[M]*diff(F[K](t),t)+F[K](t)=K[M]*u:
```
次の連立線形微分方程式を得ます。
```
> sys:={lindeq1, lindeq2, deq3};
```
システムの過渡応答のグラフィカルな表示
パラメータに値を代入し、以下の初期条件の下で連立微分方程式を解きます。
```
> sysval:=subs(val,u=10,sys):
  init:={x[K](0)=2,D(x[K])(0)=0,theta(0)=0,D(theta)(0)=0,F[K](0)=
  0}:
> sol:=dsolve(sysval union init,[x[K](t),theta(t),F[K](t)],type=
  numeric);
```
システムの位置、台車の速度とロープの角度の過渡応答を時間に関してプロットします。
```
> plot(['op(2,sol(t)[2])'], t=0..30, axes=boxed,title="Position as
  a function of Time",labels=["Time [s]", "x [K]"]);
> plot(['op(2,sol(t)[3])'], t=0..30, axes=boxed,title="Speed as a
  function of Time",labels=["Time [s]", "Speed [K]"]);
> plot(['op(2,sol(t)[4])'], t=0..30, axes=boxed,title="Angle as a
  function of Time",labels=["Time [s]", "Theta "]);
```

システム行列の設定

DEtools、LinearAlgebraおよびlinalgパッケージを使って、これまでの連立微分方程式を、一次の連立微分方程式に変更します。

```
> with(DEtools):with(LinearAlgebra):with(linalg):
```

convertsys コマンドを使って一次の連立方程式に変換します。

```
> syst:=convertsys(sys,init,[x[K](t), theta(t), F[K](t)],t,X,X_p);
```

genmatrix コマンドで連立方程式を行列の形式に変換します。

ここで、 $\frac{\mathrm{d}}{\mathrm{d}t} X(t) = AX + bu$ です。

```
> A:=genmatrix(map(rhs=0,syst[1]), [ X[1],X[2],X[3],X[4],X[5]],
  'inhom');
> b:=map(x->-1*x/u,inhom);
```

極配置による制御装置の設計

システムが可制御であるかの判定

最初に駆動装置の電圧により、システムが可制御であるか調べます。
制御行列を計算し、その行列式がゼロでないことを確かめます。

```
> Q:=concat(b, multiply(A,b), multiply(A^2,b),multiply(A^3,b),
  multiply(A^4,b));
> det(Q);
```

もし K_M がゼロでなく、 T_M, m_K そしてrが有限値であるならシステムは可制御です。

極を決定

制御回路を開発するために、制御システムの極を計算し表示します。

```
> pole:=eigenvalues(A);
> pole:=subs(val,[pole]);
> plot(map([Re,Im],pole),-2..0,-2.5..2.5,style=point,symbol=
  circle);
```

制御回路の目標は、制御エラーがゼロで、素早く、良く減衰する 安定な過渡効果を生成することです。
虚軸の極はできるだけ左にあったほうが良いので、結果として極は負の実部を持ちます。
干渉の感度をできるだけ低く押さえ、駆動がオーバーステアリング（過度に操作する）ことを防ぐためには、システムの速度を十分に速くするべきです。
すなわち、極を必要な限り左になるようにすれば良いのです。
制御回路の応答伝達関数が、素早く、良く減衰する2次遅れの装置となり、他の極の影響が無視できるように、 極を配置します。

２次遅れ装置の応答伝達関数 G およびステップ関数応答 h は、次のようになります。

```
> G:=s->K/(1+2*d*T*s+T^2*s^2):
> h:=t->K-K/sqrt(1-d^2)*exp(-d*t/T)*sin(sqrt(1-d^2)*t/T):
```

減衰振動の包絡

```
> he:=t->K-K/sqrt(1-d^2)*exp(-d*t/T):
```

限界 $he(infinity) = K$ との差

```
> abs(he-K)=K/sqrt(1-d^2)*exp(-d*t/T):
```

減衰は d =0.7 が適当です。
t =25s 後に、距離が 2%を過ぎないよう、その値を固定します。
これにより T の値が、次のように計算されます。

```
> r1:=K*2/100=K/sqrt(1-d^2)*exp(-d*t/T):
> r2:=solve(subs(d=0.7,t=25,r1),{T});
```

これは応答伝達関数を導き出します。

```
> G_:=subs(d=0.7,t=25,r2,G(s));
```

もう一つのの極を-1に設定します。
これで特性多項式を得ます。

```
> p:=(s+1)^3*simplify(denom(G_)/lcoeff(denom(G_)));
```

これで極の分布は、次の様になります。

```
> plot(map([Re, Im],[fsolve(p=0,s,complex)]),-2 .. 0,-1 .. 1,style
  = point,symbol = circle);
```

制御回路のパラメータの計算

制御回路のパラメータを計算するために、 可制御行列の逆行列の最後の行が必要です。

```
> Q_:=inverse(Q);
> Q_5:=row(Q_,5);
```

システム行列を特性多項式に入れ Q_5 をかけて、制御行列 R が得られます。

```
> R:=scalarmul(Q_5,coeff(p,s,0)):
> for i from 1 to 5 do
   R:=matadd(R,multiply(Q_5,A^i),1,coeff(p,s,i)):
  end do:
> i:='i':
> R:=subs(val,evalm(R));
```

フィルターの増幅

```
> TMP:=array(1..5,1..5,[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,
  0,0,0],[b[5]*R[i]$i=1..5]]);
> s[1]:=1/multiply(multiply([1,0,0,0,0],inverse(matadd(TMP,A,1,-1)
  )),b);
```

制御回路に対する方程式の計算

シミュレーションを行うために、制御回路に対する方程式を計算します。

```
> X:=vector([x[1](t),x[2](t),x[3](t),x[4](t),x[5](t)]):
> Xp:=vector([diff(x[1](t),t),diff(x[2](t),t),diff(x[3](t),t),diff
  (x[4](t),t),diff(x[5](t),t)]):
> R:=convert(evalm(R),vector);
> r2:=scalarmul(b,innerprod(R,X));
> SYS:=geneqns(A,X,matadd(matadd(Xp,scalarmul(b,s[1]*w),1,-1),r2,
  1,1));
```

パラメータの値を代入し、台車の目的地を x=10 に設定します。
そして以下の初期条件の下に微分方程式を解きます。

```
> SYS:=simplify(subs(val, w=10, SYS));
> INIT:={x[1](0)=2,x[2](0)=0,x[3](0)=0,x[4](0)=0,x[5](0)=0}:
> sol:=dsolve(SYS union INIT,[x[1](t),x[2](t),x[3](t),x[4](t),x[5]
  (t)],type=numeric);
```

制御回路の過渡応答のグラフィカルな表示

時間に対する、制御回路の位置、台車の位置、ロープの角度の過渡応答をグラフ表示します。

```
> plot(['op(2,sol(t)[2])'], t=0..30, axes=boxed,title="Position as
  a function of Time",labels=["Time [s]", "Position [K]"]);
> plot(['op(2,sol(t)[3])'], t=0..30, axes=boxed,title="Speed as a
  function of Time",labels=["Time [s]", "Speed [K]"]);
> plot(['op(2,sol(t)[4])'], t=0..30, axes=BOXED,title="Angle as a
  function of Time",labels=["Time [s]", "Theta"]);
```

結論

Maple を使ってローディング・ブリッジの微分方程式を設定し、それを線形化しました。
この微分方程式から、極配置によって制御装置を設計しました。
この制御装置により、負荷は、事前に指定した時間後に、指定した誤差内で安定するようになります。

参考文献

Otto Föllinger, Regelungstechnik, Hüthig.

```
>
```

Copyright (c) 2009, Cybernet Systems, Co., Ltd. All rights reserved.