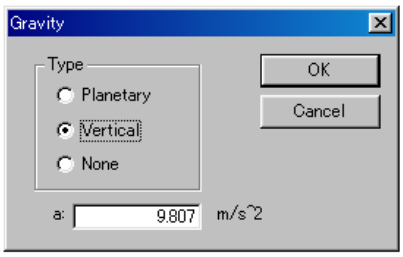
Working Model 2D 概要

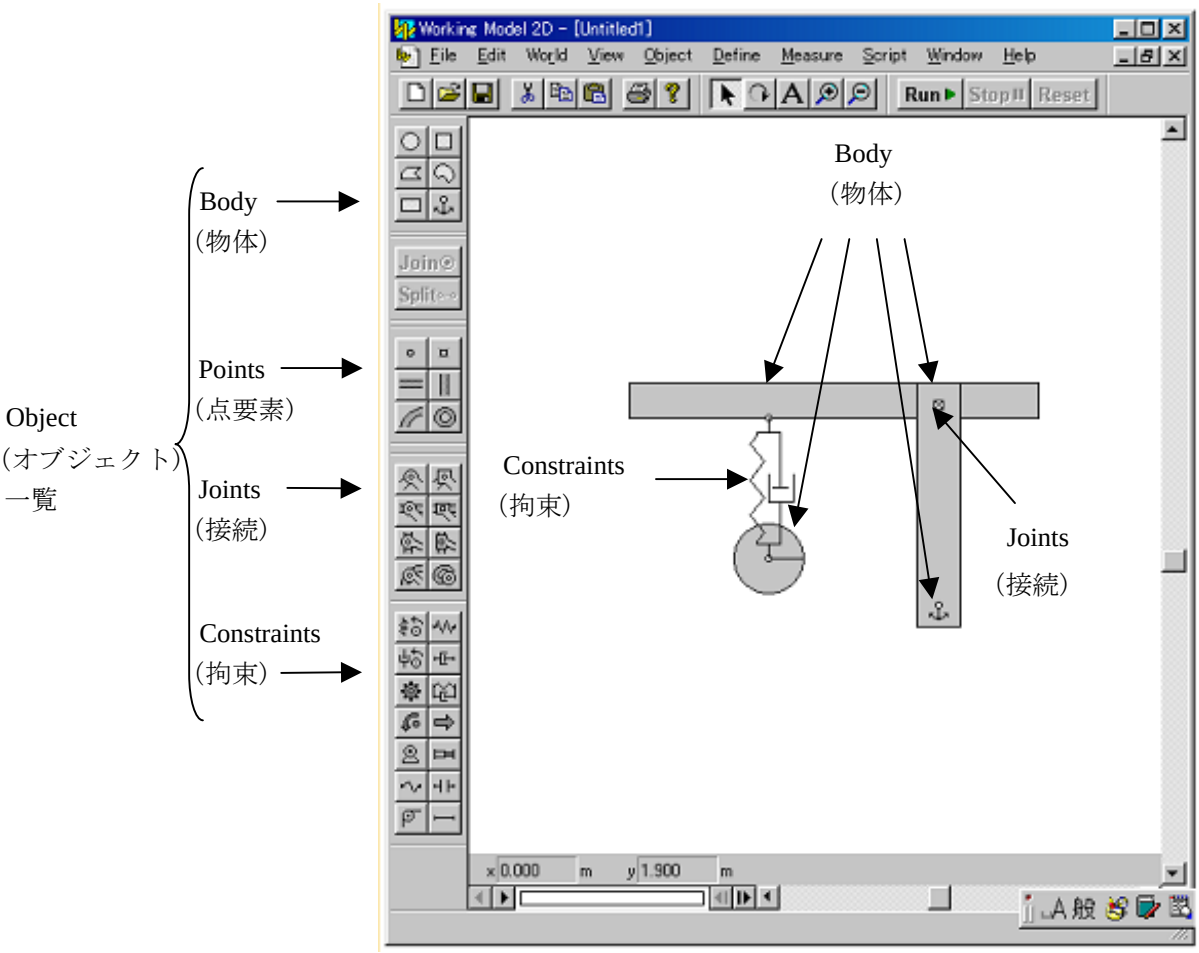
Working Model 2D： ２次元力学シミュレーションソフトウェア

World：力学モデルが動作するスペースと，その環境
Object：World の中に存在しうる，あらゆる『物と規則』

・World メニューから Gravity, Air Resistance などを設定する．
・World > Pause Control により計算停止条件を設定する．
・必要ならば，View > Workspace で軸やグリッドを表示させる．



World > Gravity で重力設定



・Body を先に作って，それらを Joint / Constraints で接続する．
・Object を左ダブルクリックして，Properties を設定する．
・Object を左クリックして選択して，Measure メニューで計測項目を設定する．
・[Run]で実行．World > Pause Control が設定されていないと[Stop]を押すまで停止しない．
・File>Export により Tab 区切りテキスト形式で計算データを出力することができる．

Working Model 2D と MATLAB のリンク

・Working Model 2D でモデルを作成し，各 Properties を設定.
・MATLAB で制御系を設計.
　注意：MATLAB と DDE 接続するためには，インターフェース用プログラム WM2M.EXE を実行させなければならない
・View -> Numbers…で精度を設定.
・World -> AccuracyのAnimation StepとIntegration Stepを設定.
・Meter や Control を作成し，MATLAB に渡したい Properties を設定.
　・Object を選択して，Measure > Position などで Output Object を作成して，Properties を設定.
　・Define > New Control で，Input Object を作成し，Force や Actuator Object の Properties で関連付け.
・Define > New Application Interface で Interface Object を作成し，その Properties を設定.
　・Application ボタンで WM2M を設定.
　・Document を”engine”とする.
　・Outputs, Inputs の Connect, Variable（MATLAB 変数名）を設定.
　・Initialize, Execute の設定.
　　（例）Initialize：u = 0;　Execute：u = -K*x; … MATLAB で設計した制御系
・[Run]で実行.
・File > Export で，Option で設定した Meter のデータをファイルに書き出す.

（補足）WM2M について
　Working Model 2D で Interface Object を使用して MATLAB (2007)以降を直接呼び出すモデルを[Run]させると，応答が無く，しばらくすると，タイムアウトエラーになり，シミュレーションできない.
　Working Model 2D は DDE により MATLAB と通信する．しかし，DDE は古い方式であり，新しい MATLAB ではサポートされなくたったため，通信することができなくなったためである.
　Working Model 2D と MATLAB のインターフェース用プログラム WM2M.EXE をダウンロードして，使用する.
・使用方法
　１．WM2M.EXE を実行する．MATLAB が Automation モードで起動するので，MATLAB で必要なコマンドや初期設定用 M ファイルなどを実行させる.
　　WM2M.EXE を実行する前に，MATLAB を Automation モードで起動させておいてもよい.
　２．Working Model 2D にファイルをロードし，以下のように Interface Object の設定を変更する.
　　・Application を“WM2M”に設定
　　・Document は“engine”とする
　３．Working Model 2D の［Run］でシミュレーションさせる.
　４．Working Model 2D のファイルを閉じる，または終了する.
　５．WM2M.EXE を終了させると，自動起動した MATLAB も終了する.
　６．MATLAB のみ先に終了させてしまった後で Working Model 2D でシミュレーションしたい場合は，一旦 WM2M.EXE を終了してから再実行するか，あるいは，WM2M.EXE の［ファイル］メニューの［MATLAB の起動］コマンドを実行してから，シミュレーションする.

【例１】振子の制振制御

１．Working Model 2D で制御対象を作成

・World > Gravity で重力を Vertical に設定．必要ならば，View > Workspace や View > View Size を設定．

・Rectangle で Body を作成，左ダブルクリックして，mass などの Properties を設定．

・Rotational Damper を Rectangle に接続し，Properties の Torque を K s にし，K を適当に設定．

・Motor を Rectangle に接続し，Properties の Type を Torque に設定．

・Motor を選択したまま，Define > New Control > Torque で Control を Motor に接続．

・Rectangle を選択し，Measure > Position > Rotation Graph 等で Meter を作成し，Properties で y1 に角度 Body[1].p.r，y2 に角速度 Body[1].v.r，y3 に角加速度 Body[1].a.r を指定し，それらの表示範囲を設定．

・Measure > Time で時間表示用 Meter を作成．

・Rectangle をドラッグして初期角度を設定．または Properties で初期角度 O を数値で設定．

・World > Pause Control により計算停止条件を設定し，Motor Torque を 0 にして，[Run]ボタンで初期値応答（自由応答）のシミュレーションを実行．

・Rotation Measure を選択し，File > Export の Option で設定した Meter のデータをファイルに保存．（精度は View -> Numbers…で設定しておくか，適当に Meter をスケーリングしておく）

２．数式モデル

・運動方程式：質量 m，長さ L，慣性モーメント J，粘性係数 K，角 θ，入力トルク τ

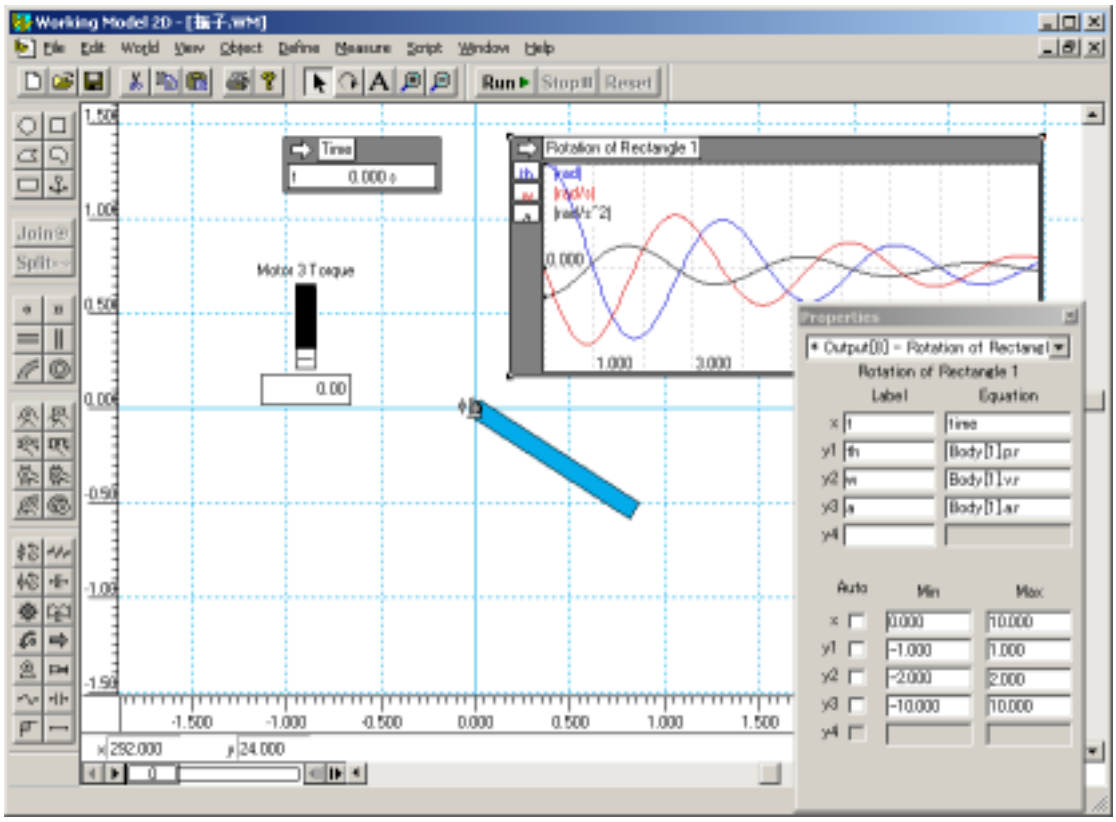
$$J\ddot{\theta} + K\dot{\theta} + \frac{mgL}{2}\sin\theta = \tau \tag{1}$$

・線形化：角 θ を微小として，$\theta = 0, \dot{\theta} = 0$ の近傍で線形化

$$\ddot{\theta} + 2\varsigma\omega_n\dot{\theta} + \omega_n^2\theta = \tau/J, \quad 2\varsigma\omega_n = K/J \ , \omega_n^2 = mgL/(2J) \tag{2}$$

・状態変数を $\boldsymbol{x} = [\theta \quad \dot{\theta}]^T$，入力 $u = \tau$ とすると，状態方程式は

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u, \quad \boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\varsigma\omega_n \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 \\ 1/J \end{bmatrix} \tag{3}$$



３．パラメータ同定

Working Model 2D で作成した各 Properties を数式モデルに直接使用する場合は省略.

・Working Model 2D で Rectangle を角度 0 で静止させて，微小な一定トルクを与えてシミュレーションし，データを保存.（精度は View -> Numbers…で設定するか，適当に Meter をスケーリングしておく）

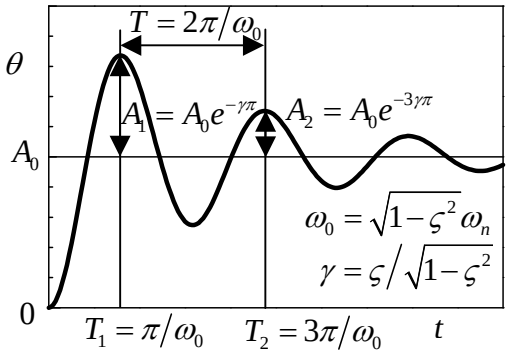
・保存されたステップ応答のデータ波形から，２次系で近似した数式モデルのパラメータを同定.

図から定常状態の角度 A_0，行き過ぎ量 A_1，時刻 T_1 を測定． $B=\ln(A_1/A_0)$ として，

$$\varsigma=\sqrt{\frac{B^2}{\pi^2+B^2}},\quad \omega_n=\frac{\pi}{\sqrt{1-\varsigma^2}T_1} \qquad (4)$$

$$J=\tau/(\omega_n^2\sin A_0) \qquad (5)$$

（自由応答の A_1 と A_2 から ω_n と ς を求めてもよい）



・あるいは，式(1)と角度，角速度，角加速度および入力トルクの測定値から，最小２乗法により $J,K,mgL/2$ を推定してもよい.

４．制御系の設計

線形化されたモデル(3)に基づき，極配置や最適レギュレータなどで状態フィードバック制御系を設計

$$u=-F\boldsymbol{x} \qquad (6)$$

５．制御系のシミュレーション：Working Model 2D から MATLAB 関数を実行

・Define > New Application Interface で Interface Object を作成し，その Properties を次のように設定.

・Application ボタンで“WM2M”を設定.

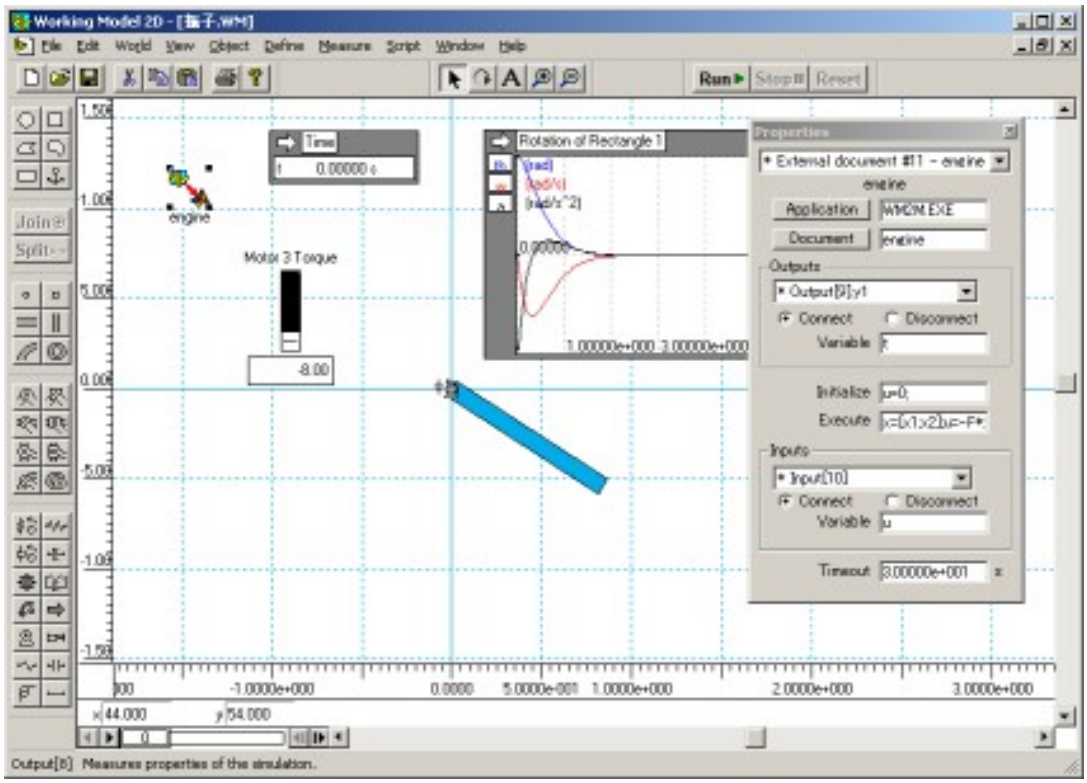
・Document を”engine”とする.

・Outputs, Inputs の Connect, Variable（MATLAB 変数名）を設定.

（例）Output[8].y1：x1， Output[8].y2：x2， Input[10]：u … [ ]内の数字は作成手順により異なる

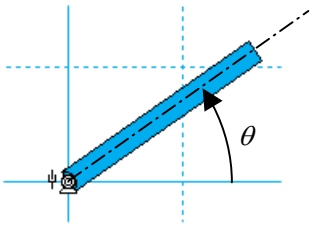
・Initialize, Execute の設定.

（例）Initialize：u=0； Execute：x=[x1;x2]; u=-F*x; … MATLAB で設計したコントローラ



【例２】１リンクアームの制御

１．Working Model 2D で制御対象を作成



・例１で作成した振子を，右図のように関節角度θをとるように修正する．
Rectangle 用 Meter をダブルクリックし，Properties で角度の式を
Body[1].p.r－pi/2　として，原点をずらせばよい．
（角度が非対称になるので注意．それがいやなら，横長の棒で作成し直す）

２．数式モデル

・上記の関節角度の取り方の変更に伴い，数式モデルを変更する．

$$J\ddot{\theta}+K\dot{\theta}+\frac{mgL}{2}\cos\theta=\tau \qquad (7)$$

３．パラメータ同定：省略

４．制御系の設計

・種々の方法で制御系を設計する．

（例）重力補償＋PD コントローラ

・次のページのような関数の M ファイルを作成する．

・MATLAB　で PD ゲインを設定（とりあえず例１で設計した状態フィードバックゲインでも可能）
KpKd = [8 6.5];　　など

５．制御系のシミュレーション

・Working Model 2D で Interface Object をダブルクリックして，設計したコントローラを利用するように変更する．

（例）Outputs：Output[9].y1：t　　（時刻）
Initialize：u=0;　　（Execute の if 文がないとうまく動かない？）
Execute：x=[x1;x2]; if t==0 x0=x1; end; u=c1linkpd(t,x,x0,xe,KpKd);

・MATLAB で最終目標値を設定する．

（例）xe = 0;　　や　　xe = pi/2;　　など
MATLAB ではなく，Working Model 2D の Initialize に書いてもよい．

・Working Model 2D でシミュレーションを行う．

（注）

・別の最終目標値でシミュレーションする場合は，一度 Interface Object の Input を Disconnect するなどして Meter のデータを消去しないと，前に計算された値をたどるだけで，再計算しない．

・制御に失敗して回転してしまって停止させた場合などは，続けてその位置から実行せず，Reset して，必ず初期位置から再スタートさせること．適当にドラッグして初期位置を変更すると，１回転以上していることがあるので注意．

・Working Model で制御対象を作成の時，重心回りの慣性モーメントをあまりに適当に設定するとうまくいかない．細長い形状にして，moment は Planar のままを推奨．変更した場合は，その値を設計時に用いず，パラメータ同定を行ったほうがよい．なお，例題の数式モデルにおけるJは，重心回りの慣性モーメントでないことに注意．

（平板の定理）辺の長さa,bで質量mの均質な長方形平板の重心を通り平板に垂直な軸回りの慣性モーメント$J_G=m(a^2+b^2)/12$．

（平行軸の定理）任意の軸回りの慣性モーメントJは，重心を通りこの軸に平行な軸回りの慣性モーメントをJ_G，平行軸間の距離をdとすると，$J=J_G+md^2$．

```
function u = c1linkpd(t,x,x0,xe,KpKd)

% １リンクアームの重力補償＋PD 補償
%   t ：時刻
%   x ：状態変数 [θ,θ']T
%   x0：初期角度
%   xe：最終目標角度
%   KpKd：PD ゲイン（設計した状態 FB ゲインでも可）
% Working Model 2D ： 1LINK.wm

% リンクパラメータなど（各自のモデルに合わせる）
g=9.8;
L=1;
J=1;
m=0.82;
K=0.5;

% 目標軌道（2LINK 用の軌道生成関数を流用）
xi=[x0;0];
xd=[xe;0];
todr=4;                                 % ４次の軌道（0,1,2,4 から選択）
vamax=[1;1;10;10];                      % 角速度，角加速度の最大値
osize=2;                                % 目標角度のみ（2,4,6 から選択：下参照）
xr = trgen_fn(t,xi,xd,todr,vamax,osize);
                                        % osize=6 の時 xr = [q1;q2;q1';q2';q1'';q2'']
                                        % 目標角度=xr(1)
% 重力補償
ug = m*g*L/2*cos(x(1));

% PD 補償器
upd = (xr(1)-x(1))*KpKd(1) - x(2)*KpKd(2);

% 操作量
u = ug + upd;
```

```
function u = c1linkidb(t,x,x0,xe,KpKd)

% １リンクアームの動的補償(FB)＋PD 補償
%   t ：時刻
%   x ：状態変数 [θ,θ']T
%   x0：初期角度
%   xe：最終目標角度
%   KpKd：PD ゲイン
% Working Model 2D ： 1LINK.wm

% リンクパラメータなど
g=9.8;
L=1;
J=1;
m=0.82;
K=0.5;

% 目標軌道（2LINK 用の軌道生成関数を流用）
xi=[x0;0];
xd=[xe;0];
todr=4;
vamax=[1;1;10;10];
osize=6;
xr = trgen_fn(t,xi,xd,todr,vamax,osize);

% PD 補償器
upd = (xr(1)-x(1))*KpKd(1) +
      (xr(3)-x(2))*KpKd(2);

% 逆動力学による動的補償(FB)
uh = J*xr(5) + K*x(2) + m*g*L/2*cos(x(1));

% 操作量
u = uh + upd;
```

```
function u = c1linkct(t,x,x0,xe,KpKd)

% １リンクアームの計算トルク法
%   t ：時刻
%   x ：状態変数 [θ,θ']T
%   x0：初期角度
%   xe：最終目標角度
%   KpKd：PD ゲイン
% Working Model 2D ： 1LINK.wm

% リンクパラメータなど
g=9.8;
L=1;
J=1;
m=0.82;
K=0.5;

% 目標軌道（2LINK 用の軌道生成関数を流用）
xi=[x0;0];
xd=[xe;0];
todr=4;
vamax=[1;1;10;10];
osize=6;
xr = trgen_fn(t,xi,xd,todr,vamax,osize);

% 加速度算出器
qa2 = xr(5) + (xr(1)-x(1))*KpKd(1) +
              (xr(3)-x(2))*KpKd(2);

% 逆動力学
uh = J*qa2 + K*x(2) + m*g*L/2*cos(x(1));

% 操作量
u = uh;
```