OpenModelica で多自由度系の固有振動数を計算する

○清水莉沙（国立天文台 先端技術センター）
(National Astronomical Observatory of Japan, Advanced Technology Center)

概要(Abstract)

1D-CAE は上流設計から適用可能な，多様な物理挙動を統合的かつ迅速に評価する手法であり，物理モデリング言語 Modelica はその有効な手段の一つとして活用される．Modelica は様々な物理モデルの時刻歴記述が簡単にできる一方, 非時刻歴解析には工夫が求められる．本報告では，オープンソース開発環境 OpenModelica を用い，多自由度系モデルの固有振動数を導出する手法とその実装を提案する．

1.1 MBD

Model Based Development（MBD，モデルベース開発）は，もともと制御ソフトウェアの開発手法として登場した．MBD では，システムをモデルとして表現し，コンピュータ上でシミュレーションを行うことで，動作検証や検討を進めていく．実機を製作する前の段階で検証を実施できるため，コスト削減につながるほか，モデル化によって設計意図の明確化や仕様の曖昧さが排除され，設計ミスの早期発見にも寄与するなどの利点がある．

近年では，制御分野にとどまらず，さまざまな物理モデルへと MBD の考え方を拡張する動きが広がっている．さらに，有限要素法に代表される 3D-CAE を含む，シミュレーション全般を活用した設計手法を MBD として捉える場合も増えている．

1.2 因果モデリングと非因果モデリング

MBD を実現する考え方の例として，1D-CAE が挙げられる．1D-CAE では設計対象を数理モデルを使いシステム全体を表現する．1D-CAE のモデルの記述においては，入力と出力を明示する因果モデリングと，入力と出力を明示しない非因果モデリングがある．因果モデリングは，関係が明確に定義されるような制御モデルに，非因果モデリングは入出力が定まっていない物理モデルに向いている．非因果モデリングのツールの代表例としては，Mathworks 社の Simscape や Modelica がある．Modelica はオブジェクト指向のモデリング言語であり，Modelica の生みの親が作ったツールである Dymola と，オープンソース GUI 環境である OpenModelica が，開発環境として広く使われている．

1.3 OpenModelica

Modelica は，パラメータや定数を除いた未知数の数と等式の数が一致するようにモデルを構成することを基本とする．シミュレーションにより，状態量や代数変数などが時系列で数値的に求められる．OpenModelica には，Modelica コードを記述して実行するための UI に加えて，多様な分野のライブラリが用意されている．オブジェクト指向・コンポーネント指向の設計により，GUI 上でコンポーネントのコネクタを接続することでモデルを構築でき，その接続に応じた接続方程式が自動生成される．

2.1 固有振動数の計算

振動する物体が外部の振動と同期してさらに大きく振動する現象を共振と呼ぶ. すべての物体・系は, 振動しやすい特有の周波数である固有振動数をもつ.

観測装置を設計する上で固有振動数を把握することは非常に重要である．運用や運搬，打上げ時のエンジン振動・音響振動などの環境振動が固有振動数と一致すると共振により構造が破損する可能性がある．また，冷凍機の振動，地面振動などの外乱が構造の固有振動数と一致すれば，共振で光学素子が振動してしまい，光学性能の劣化につながる．さらに，共振周波数の把握はモデルの妥当性評価にも利用される．振動試験の結果と解析結果を比較することで，モデルの妥当性やワークマンシップエラーの有無を確認できる．

Figure 1 に示すような質量m_1, m_2, m_3の物体がばね定数k_1, k_2, k_3, k_4のばねに繋がれた 3 自由度の減衰なしばねマス系の固有振動数を求める．運動方程式は以下のとおりである．

$$m_1\ddot{x_1} + k_1x_1 - k_2(x_2 - x_1) = 0$$
$$m_2\ddot{x_2} + k_2(x_2 - x_1) - k_3(x_3 - x_2) = 0$$
$$m_3\ddot{x_3} + k_3(x_3 - x_2) - k_4x_3 = 0$$

これを質量行列**M**と剛性行列**K**を使って行列で書き直すと，

$$\mathbf{M}\ddot{\mathrm{x}} + \mathbf{K}\mathrm{x} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}\begin{bmatrix} \ddot{x_1} \\ \ddot{x_2} \\ \ddot{x_3} \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 + k_4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

となる．これが非自明解$(x \neq 0)$を持つ条件は

$$\det(\mathbf{K} - \omega^2\mathbf{M}) = \begin{vmatrix} (k_1 + k_2) - m_1\omega^2 & -k_2 & 0 \\ -k_2 & (k_2 + k_3) - m_2\omega^2 & k_3 \\ 0 & -k_3 & (k_3 + k_4) - m_3\omega^2 \end{vmatrix} = 0$$

である．上記行列式を解いて得られる固有値ω_1，ω_2，ω_3 より，固有振動数f_1，f_2，f_3は以下の通り．

$$f_1 = \frac{\omega_1}{2\pi},\ f_2 = \frac{\omega_2}{2\pi},\ f_3 = \frac{\omega_3}{2\pi}$$

この手法は固有値解析と呼ばれ，系の質量行列**M**，剛性行列**K**が分かれば固有振動数が求められる．

2.1 Modelica で幅広い周波数成分を持つ入力に対する応答を調べ固有振動数の計算

実際の設計では構造が複雑な形状となることが多く，有限要素に分割して解析するのが一般的である. しかし本稿では, 有限要素モデルを使わない別のツールとして, Modelica を用いて作成したばねマス系の固有振動数を求めることを目的とする．Modelica は時刻歴シミュレーションを容易に実行できる一方で，システム特性を直接評価するような非時刻歴計算には必ずしも向いていない．そこで今回は，振動試験やハンマリング試験で用いられるような，幅広い周波数成分を含む入力を時刻歴で与える．その応答波形を周波数解析することで，共振が発生する周波数を同定する方法をとる．ただし，周波数解析など，OpenModelica で実装しにくいものは適宜 Python を利用することとした.

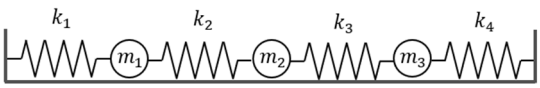


Figure 1 3 自由度のばねマスモデル

幅広い周波数成分を含む入力としては，サインスイープ，ホワイトノイズ，インパルスなどが挙げられる．サインスイープは正弦波の周波数を一定範囲内で時間とともに連続的に変化させた信号，ホワイトノイズはすべての周波数成分の振幅が等しい雑音で，いずれも振動試験で広く用いられる信号である．一方，インパルスは時間幅が無限小，高さが無限大で，積分すると 1 となる理想的な信号である．ハンマリング試験でハンマーを用いて入力する衝撃がこれに相当する．

これらの信号はいずれも理想的には広い周波数帯域を含むことができるが，実際には有限の時間幅で計算する必要があるため，測定したい周波数領域が十分に含まれているか注意しなければならない．今回は，S/N 比が比較的良いサインスイープとホワイトノイズを用いてモデルを構築した．

以下に，計算の流れを示す．

(1) OpenModelica の GUI を用いてばねマスモデルを構築する．ばねマスの固定点には強制変位を与え，応答を取得するためのセンサをばねマスの先に配置する．

(2) OMPython を利用して (1)で作成したばねマス Modelica モデルを解析し，結果を CSV 形式で出力する．OMPython は OpenModelica の Scripting API を呼び出すための Python ライブラリであり，(1) で作成したモデルの実行や結果の取得を自動化できる．

(3) 得られた変位データを Python に取り込み，NumPy を用いて FFT を実行する．今回は簡単のため，伝達関数の算出は行わず，出力応答そのものの周波数成分のみを評価する．

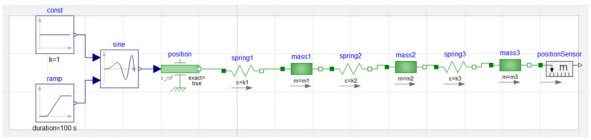
(4) SciPy の関数を用いて, FFT 結果から応答が増幅している周波数を抽出する. この周波数成分が，共振によって振幅が大きくなる固有振動数に相当する．

2.2 解析の結果

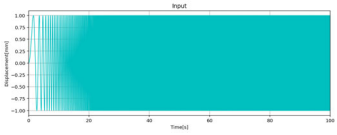
今回，多自由度系のばねマスとして，$m_1 = 0.4\,\mathrm{kg}$，$m_2 = 0.1\,\mathrm{kg}$，$m_3 = 0.6\,\mathrm{kg}$，$k_1 = 30\,\mathrm{N/m}$，$k_2 = 50\,\mathrm{N/m}$，$k_3 = 70\,\mathrm{N/m}$ の 3 自由度のばねマスを想定する．この系の固有振動数を，行列式を解くことによって求めると，0.69 Hz，2.10 Hz，5.79 Hz の 3 つが算出される．

サインスイープのモデルと(a) 入力波形，(b) 応答波形，(c) 入力波形の FFT の結果，(d) 応答波形の FFT の結果を Figure 2 に示す. 加振周波数は$f = 0$~20 Hz, 加振秒数は$t = 100$ s（スイープ速度0.2 Hz/s), 振幅 1, 結果は 1/1000 s ごとに取得した．同様に，ホワイトノイズのモデルの結果を Figure 3 に示す．ホワイトノイズはノイズパワー1，サンプルピリオド 1/1000，加振秒数は 100 s とした．エイリアシング防止のためバターワース 4 次のローパスフィルタを 50 Hz で設定した．結果は 0.1 s ごとに取得した．FFT のサンプリング周波数は 1000 Hz，窓関数（ハニングウィンドウ）を設定した．

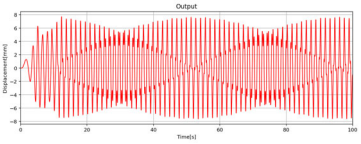
Figure 2，Figure 3 (e) より，どちらの入力波形においても 0.7 Hz，2.1 Hz，5.8 Hz に応答のピークを検出することができた．ホワイトノイズは S/N 比がサインスイープに比べて低く，十分な加振時間がなければピークが目立たなかった．どちらも，入力信号自体の設定や FFT のパラメータの設定項目が複数あるが，ホワイトノイズの設定では高周波成分によるエイリアシングの対策などさらに考慮しなくてはならない点が多く，サインスイープのほうが簡易にモデルを組むことができた．



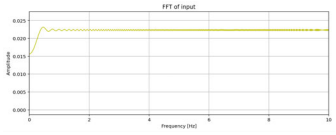
(a) Modelica モデル



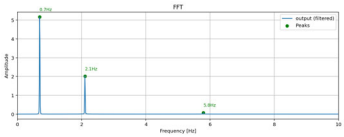
(b) 入力波形



(c) 応答波形

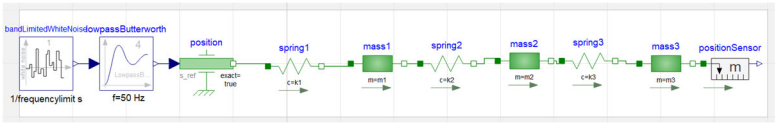


(d) 入力波形 FFT 結果

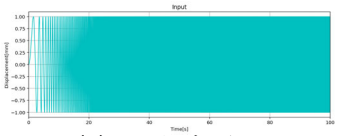


(e) 応答波形 FFT 結果

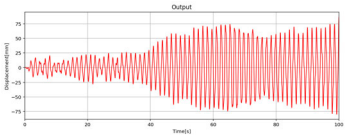
Figure 2 サインスイープ波形を入力したモデルの結果



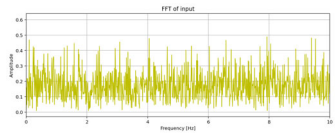
(a) Modelica モデル



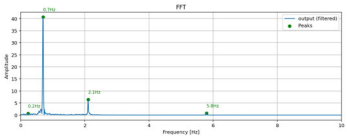
(b) 入力波形



(c) 応答波形



(d) 入力波形 FFT 結果



(e) 応答波形 FFT 結果

Figure 3 ホワイトノイズ波形を入力したモデルの結果

3.1 Modelica 線形化機能を使った固有振動数の計算

$\boldsymbol{x}(t)$はシステムの状態，$\boldsymbol{u}(t)$はシステムへの入力，$\boldsymbol{y}(t)$はシステムの出力としたとき，

$$\dot{\boldsymbol{x}} = f(x,u,t), \quad \boldsymbol{y} = g(x,u,t)$$

をある動作点(x_0,u_0)周りで

$$\Delta\dot{\boldsymbol{x}} = \mathbf{A}\Delta\boldsymbol{x}(t) + \mathbf{B}\Delta\boldsymbol{u}(t), \quad \Delta\boldsymbol{y} = \mathbf{C}\Delta\boldsymbol{x}(t) + \mathbf{D}\Delta\boldsymbol{u}(t)$$

と置き，線形状態空間モデルに変換することを線形化という．$\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ はそれぞれ動作点周りでの一次のテイラー展開

$$\mathbf{A} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{x_0,u_0}, \quad \mathbf{B} = \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{x_0,u_0}, \quad \mathbf{C} = \left.\frac{\partial g}{\partial \boldsymbol{x}}\right|_{x_0,u_0}, \quad \mathbf{D} = \left.\frac{\partial g}{\partial \boldsymbol{u}}\right|_{x_0,u_0}$$

で求められる．線形化は制御分野において，非線形なシステムの特性を調べる手法として活用されている．対象のシステムがもともと線形であれば，$\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ の係数行列は動作点に依存しない定数となり，特に，$\mathbf{A}$ はシステムの性質を表す行列となる．Figure 1 で示した質量m_1,m_2,m_3の物体とばね定数k_1,k_2,k_3,k_4で構築された 3 自由度ばねマスの場合，時刻tにおけるマスm_iの位置s_i，速度v_i は，$\boldsymbol{x}(t) = [s_1 \quad v_1 \quad s_2 \quad v_2 \quad s_3 \quad v_3]^{\mathrm{T}}$であれば

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\dfrac{k_1+k_2}{m_1} & 0 & \dfrac{k_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \dfrac{k_2}{m_2} & 0 & -\dfrac{k_2+k_3}{m_2} & 0 & \dfrac{k_3}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{k_3}{m_3} & 0 & -\dfrac{k_3+k_4}{m_3} & 0 \end{bmatrix}$$

となる．この$\mathbf{A}$行列の固有値を求めると，固有値解析で求めたものと同じ結果となる．

OpenModelica の GUI には実装されていないが，Modelica 用のコマンドラインである OMShell では線形化を利用することができる．このコマンドを実行すると，線形化の係数$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$の計算結果を記録した Modelica のモデルが出力される．今回は，この機能を使って 2.1 と同じパラメータの 3 自由度ばねマスの固有振動数を求める．

手順は以下の通りとする．

(1) OpenModelica の GUI を用いてばねマスモデルを構築する．

(2) OMPython を利用して(1)のモデルを線形化し，生成された Modelica ファイル，Linearized_model.mo の中にある行列$\mathbf{A}$を抽出する．

(3) NumPy を使い，行列$\mathbf{A}$の固有値を求め，固有振動数を計算する．

3.2 線形化モデルを用いた計算の結果

ばねマスモデルの図を Figure 4，線形化によって得られた Modelica ファイル Linearized_model.mo と，Python で記述し実行した固有振動数の計算結果を Figure 5 に示す．3 自由度系のモデルにおいて外力および出力の指定をしていないため，$\mathbf{B}, \mathbf{C}, \mathbf{D}$は 0 行列となっており，$\mathbf{A}$ にのみ値が入っている．$\mathbf{A}$ の行列の固有振動数を求めると，事前の計算と同じ 0.69 Hz, 2.10 Hz, 5.79 Hz を求めることができた．

4 まとめ

今回は，Modelica を用いて多自由度ばねマス系の固有振動数を求める二つの手法を検討した．1 つ目は，広帯域入力を加えて応答を FFT しピーク周波数から固有振動数を同定する方法である．サインスイープ入力は設定項目が少なく扱いやすかった．2 つ目は，線形化機能により $\mathbf{A}$ 行列を取得し固有値を求める方法であり，線形系であれば簡便に固有振動数を算出できる．Modelica のモデルを利用することで減衰や外乱の追加，伝達関数の算出，多領域分野との統合など，さらなる拡張が期待できる．

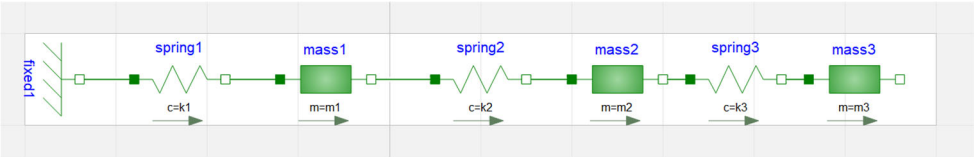


Figure 4 3 自由度のばねマスモデル

```
model linearized_model "Eigenvalue_mo_20251119"
  parameter Integer n = 6 "number of states";
  parameter Integer m = 0 "number of inputs";
  parameter Integer p = 0 "number of outputs";

  parameter Real x0[n] = {0, 0, 0, 0, 0, 0};
  parameter Real u0[m] = zeros(0);

  parameter Real A[n, n] =
    [0, 1, 0, 0, 0, 0;
    -200, 0, 125, 0, 0, 0;
    0, 0, 0, 1, 0, 0;
    499.9999999999999, 0, -1200, 0, 700, 0;
    0, 0, 0, 0, 0, 1;
    0, 0, 116.6666666666667, 0, -116.6666666666667, 0];

  parameter Real B[n, m] = zeros(n, m);

  parameter Real C[p, n] = zeros(p, n);

  parameter Real D[p, m] = zeros(p, m);


  Real x[n](start=x0);
  input Real u[m];
  output Real y[p];

  Real 'x_mass1.s' = x[1];
  Real 'x_mass1.v' = x[2];
  Real 'x_mass2.s' = x[3];
  Real 'x_mass2.v' = x[4];
  Real 'x_mass3.s' = x[5];
  Real 'x_mass3.v' = x[6];
equation
  der(x) = A * x + B * u;
  y = C * x + D * u;
end linearized_model;
```

(a) 線形化によって得られた Modelica モデル

```
Linearizing Eigenvalue_mo_20251119 and generating linarized_model.mo ...
->Loaded linearized_model.mo successfully.
[[ 0.00000000e+00  1.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-2.00000000e+02  0.00000000e+00  1.25000000e+02  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 5.00000000e+02  0.00000000e+00 -1.20000000e+03  0.00000000e+00
   7.00000000e+02  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  1.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  1.16666667e+02  0.00000000e+00
  -1.16666667e+02  0.00000000e+00]]

--- Results ---
Mode 1:
  Eigenvalue: 0.00+36.38j
  Damping (real part): 0.00
  Frequency: 5.79 Hz
Mode 3:
  Eigenvalue: 0.00+13.21j
  Damping (real part): 0.00
  Frequency: 2.10 Hz
Mode 5:
  Eigenvalue: -0.00+4.35j
  Damping (real part): -0.00
  Frequency: 0.69 Hz

--- Summary ---
1:0.69 Hz
2:2.10 Hz
3:5.79 Hz

Time: 4.992 sec
```

(b) 固有値計算結果

Figure 5 線形化モデルを用いた計算の結果