機械振動論
1質点の振動

アニメーション・数値シミュレーション・ステップ応答の操作

▼要約

振動論はあらゆる場面で重要な要素であり、この基礎理論を理解することはエンジニアにとって基本的なものです。

ここでは、1 質点の振動問題について、Maple を使って直感的なアニメーションにより、バネ定数や減衰定数や路面入力の変化で、振動がどのように変わるかわかりやすく解説している事例となります。

また、Maple を使えば、電子テキストとして容易に共有することができます。

▼事前準備

• ワークシートの初期化（リセット）

`restart` は Maple カーネルを初期化するコマンドです。これによりメモリー上に残っていたデータなどが削除されます。

```
> restart:
```

• パッケージの呼び出し

プロットやアニメーションなどに必要なのでロードしておきます。

```
> with( plots ):
> with( plottools ):
```

パッケージの詳細については以下ヘルプを参照ください。

・plots
・plottools

▼アニメーション

まずは基本的な振動をアニメーション表示にすることで視覚的に理解してみます。

地面の上を一輪車が走っている状態を表現します。

▼パラメーターの定義

• アニメーションのフレーム（*frame*）

frame はこれからアニメーションのフレームを追加していくための変数で、最初は空にしておきます。

コロン (:=) は代入です。

```
> frame := [ ]:
```

• 車輪の半径（*wheelr*）

wheelr は車輪の半径です。1 にセットしておきます。

```
> wheelr := 1 :
```

▼基本ブロック作成

• タイヤに繋がる直方体（*cuboidtire* ）

cuboidtire はタイヤにつながっている部分のオブジェクトを `cuboid` コマンド で作成しておきます。
`cuboid` コマンドの詳細についてはヘルプを参照ください。

対角の 2 点の座標で定義します。色は緑 (*green*) にしておきます。

これを後で `translate` コマンドで平行移動することで動かします。

```
> cuboidtire := cuboid( [-0.2,-0.2,-0.2], [0.2, 0.2, 1.2], color=green ) :
```

• 載っている物体を現す立方体（*cuboidcarry*）

cuboidcarry は上に載っている物体を現す立方体です。色は赤にします。

```
> cuboidcarry := cuboid ( [-0.5,-0.5, 0], [0.5, 0.5, 1], color=red) :
```

• 地面の表現 (*ground*(*t*))

ground(*t*) は地面の形を現す関数です。

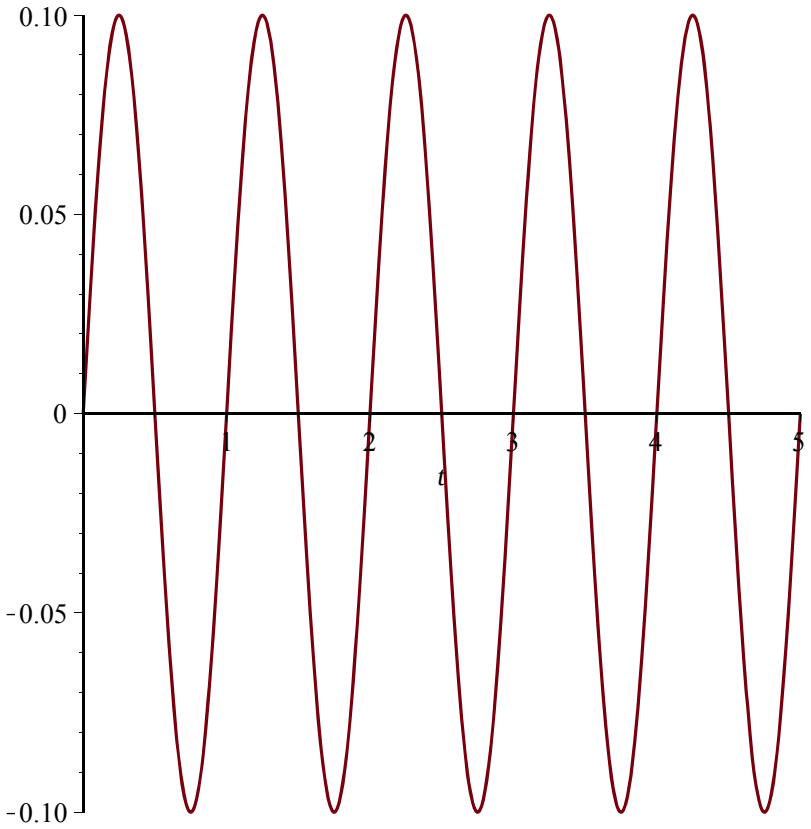
上記のように『 → 』を使って関数を定義します。

```
> ground := t→ sin(2 π t)/10
```

$$ground := t \rightarrow \frac{\sin(2\,\pi\,t)}{10} \qquad \textbf{(2.2.1)}$$

この関数をプロットしてみます。

```
> plot( ground(t), t=0..5);
```



▼アニメーション作成

以下の `for` ループで時間 (t が 0 から 1 秒まで 0.02 秒刻み) ごとにアニメーションのフレームを作成します。

スプリングは `tubeplot` コマンドを使って表現します。
スプリングの長さを `springl`で表します。

`wheelh` は車輪の中心の高さで `carryh` はその上の質点 (赤) の中心の高さの変数です。

地面には上記関数により `plot3d` コマンドで表現します。
車輪はシリンダーで表し、これを回転 (`rotate`コマンド)と平行移動 (`translate` コマンド)を使って地面の上に設置します。

上記4つのコマンド詳細については以下ヘルプを参照ください。

・tubeplot
・plot3d

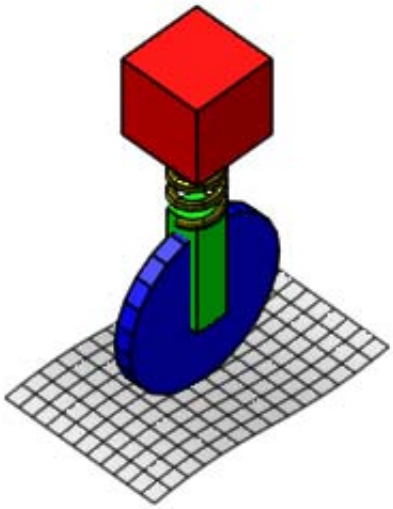
・rotate
・translate

```
> for t from 0 by 0.02 to 1 do
  # t が 0 から 1 秒まで 0.02 秒刻みで下記を実行する。

   #車輪の中心の高さを定義する：
   wheelh := ground(t):

   #地面プロットを作成する：
   pp := plot3d( [x, y, ground(t + x/4)], x = -Pi/2..Pi/2,
         y = -1..1, grid = [15, 12], color = gray ):

   #車輪プロットを今回のフレームに追加する：
   pp := pp, translate( rotate( cylinder( [0, 0, 0], wheelr,
         0.2, color = blue ), Pi/2, t*2*Pi, 0 ), 0, -0.1,
         wheelh + wheelr ):

   #車輪に繋がる直方体の位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( cuboidtire, 0, 0, wheelh + wheelr ):

   #質点の中心の高さを定義する：
   carryh := sin(2*Pi*t) / 4:

   #スプリングの長さを定義する：
   springl := carryh - wheelh + 1:

   #スプリングプロットを作成する：
   tt := tubeplot( [cos(u*8*Pi) / 4, sin(u*8*Pi) / 4,
         u*springl ], u = 0..1, radius = 0.05,
         color = yellow ):

   #スプリングの位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( tt, 0, 0, wheelh + 2*wheelr + 0.2 ):

   #載っている物体の位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( cuboidcarry, 0, 0, wheelh +
         2*wheelr + 0.2 + springl ):

   #アニメーションのフレームを追加していく
   frame := frame, display( pp ):

  end:
```

できたフレームをアニメーションで表示します。

そのためには、plots パッケージの display コマンドを用います。display コマンドの詳細についてはヘルプを参照ください。

```
> display( frame, insequence = true, scaling = constrained);
```



このモデルは単純化して 1 質点のモデルとしてシミュレーションすることができます。

地面の変化が入力になりそれに対して上の質点の振動がその応答となります。

次にこのモデルの運動方程式を考えてみます。

▼関数および変数の初期化

これはさきほど定義した関数 *ground* および代入した変数 *t* を初期化しています。

一旦代入されたものは式の中で変数として扱われませんので初期化する必要があります。

• 関数地面の表現（ *ground*(*t*) ）の初期化

```
> ground := 'ground';
```

• 時間の表現（ *t* ）の初期化

```
> t := 't';
```

• タイヤに繋がる直方体（*cuboidtire*）の初期化

```
> cuboidtire := 'cuboidtire';
```

• 載っている物体を現す立方体（*cuboidcarry*）の初期化

```
> cuboidcarry := 'cuboidcarry';
```

$$\begin{aligned} ground &:= ground \\ t &:= t \\ cuboidtire &:= cuboidtire \\ cuboidcarry &:= cuboidcarry \end{aligned} \qquad \textbf{(2.4.1)}$$

▼数値シミュレーション　ケース１

上記のモデルの運動方程式を計算してみます。

▼パラメーターの定義

運動方程式の各パラメーターに具体的な値を与えて数値シミュレーションを行います。

• 質量（ *Mass* ）

まず質量 *Mass* は 1 (Kg) とします。

> $Mass := 1;$

$$Mass := 1 \tag{3.1.1}$$

• 重力加速度（ *G* ）

重力加速度は MKS では 9.8 です。

> $G := 9.8;$

$$G := 9.8 \tag{3.1.2}$$

• 減衰定数（ *c* ）

まずは簡単に、減衰定数 *c* は 0 にします。

> $c := 0;$

$$c := 0 \tag{3.1.3}$$

• バネ定数（ *k* ）

バネ定数 *k* は 100 とします。

> $k := 100;$

$$k := 100 \tag{3.1.4}$$

• 地面の表現（ *ground*(*t*)）

ground(*t*) は地面の形を現す関数です。

地面は平面で変化なし (平面) とします。

> $ground := t \rightarrow 0$

$$\tag{3.1.5}$$

$$ground := t \to 0 \qquad \textbf{(3.1.5)}$$

運動方程式の計算

これで運動方程式は次のようになります。

```
> Mass·( diff(x(t), t$2) + G ) + c·diff(x(t), t) + k·(x(t) - ground(t) ) = 0;
```

$$\frac{d^2}{dt^2} x(t) + 9.8 + 100\, x(t) = 0 \qquad \textbf{(3.2.1)}$$

これを `dsolve` コマンドで解いてみます。 `dsolve`コマンドの詳細については ヘルプを参照ください。

```
> dsolve( {(3.2.1), x(0) = 0, D(x)(0) = 0}, x(t) );
```

$$x(t) = -\frac{49}{500} + \frac{49}{500}\cos(10\, t) \qquad \textbf{(3.2.2)}$$

$x(0) = 0$ は高さの初期値 ($t = 0$での値) を与えています。

また D(x)は $x(t)$ の一階微分で $D(x)(0) = 0$ は初速 ($t = 0$での速度の初期値) を 0 にしています。

このように初期値を与えその後の変化を求める微分方程式の問題を初期値問題と呼びます。

右辺を取り出し `unapply` コマンドで関数化します。`rhs` コマンドは右辺 (right hand side) を取り出すコマンドです。

上記2つのコマンド詳細については以下ヘルプを参照ください。

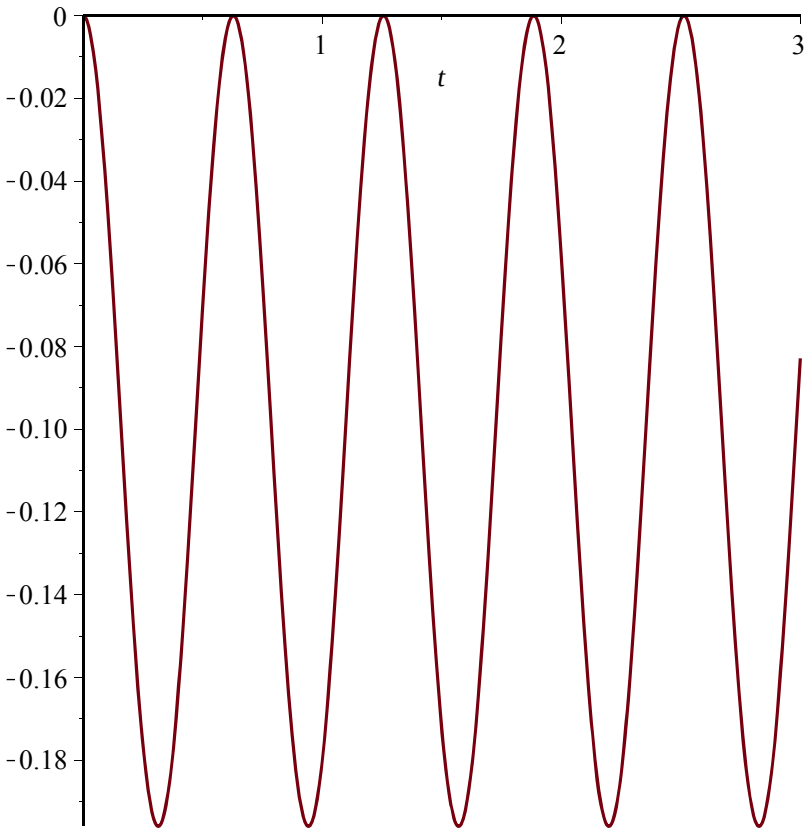
- unapply
- rhs

```
> ans := unapply(rhs((3.2.2)), t);
```

$$ans := t \to -\frac{49}{500} + \frac{49}{500}\cos(10\, t) \qquad \textbf{(3.2.3)}$$

これをプロットします。

```
> plot(ans(t), t = 0..3);
```



▼バネ定数 k の変更＆運動方程式の再計算

• バネ定数（ k ）

バネ定数 k の値を変えてみます。

> $k := 200;$

$$k := 200 \qquad \textbf{(3.3.1)}$$

• 運動方程式の再計算

> $Mass \cdot (diff(x(t), t\$2) + G) + c \cdot diff(x(t), t) + k \cdot (x(t) - ground(t)) = 0;$

$$\frac{d^2}{dt^2} x(t) + 9.8 + 200\, x(t) = 0 \qquad \textbf{(3.3.2)}$$

これを再度 `dsolve` コマンドで解いてみます。

> $dsolve(\{\textbf{(3.3.2)}, x(0) = 0, D(x)(0) = 0\}, x(t));$

$$x(t) = -\frac{49}{1000} + \frac{49}{1000}\cos\left(10\sqrt{2}\,t\right) \qquad \textbf{(3.3.3)}$$

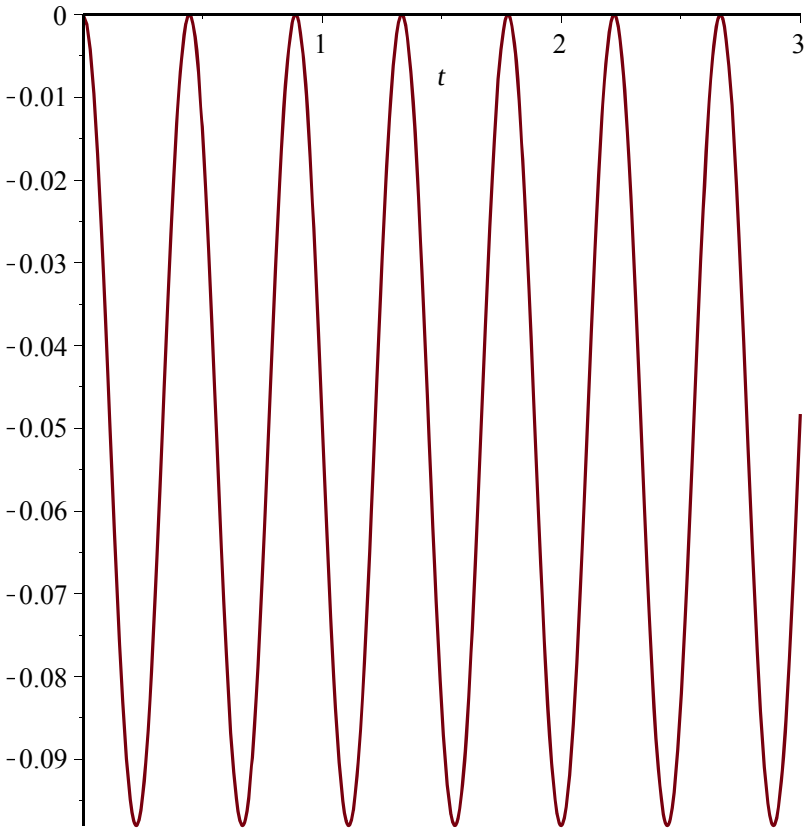
右辺を取り出し unapply コマンドで関数化します。

```
> ans := unapply( rhs((3.3.3)), t);
```

$$ans := t \rightarrow -\frac{49}{1000} + \frac{49}{1000}\cos\left(10\sqrt{2}\,t\right) \qquad \textbf{(3.3.4)}$$

これをプロットします。

```
> plot(ans(t), t = 0 ..3 );
```



上記２つのグラフを比較すると、振動数が変わったことがわかります。

このようにバネ定数 k は振動数に影響を与えます。同時に数値を見れば振幅も小さくなっていることが解ります。

k の値が大きくなることはバネが硬くなりあまり振動しなくなると同時に振動数 (周波数) が高くなります。

▼数値シミュレーション　ケース2

減衰定数の効果を入れてみます。

▼減衰定数 c の変更&運動方程式の再計算

- 減衰定数（ c ）

減衰定数 c を 0.1 にしてみます。

> $c := 0.1;$

$$c := 0.1 \qquad \textbf{(4.1.1)}$$

- 運動方程式の再計算

> $Mass \cdot (diff(x(t), t\$2) + G) + c \cdot diff(x(t), t) + k \cdot (x(t) - ground(t)) = 0;$

$$\frac{d^2}{dt^2} x(t) + 9.8 + 0.1 \left(\frac{d}{dt} x(t) \right) + 200\, x(t) = 0 \qquad \textbf{(4.1.2)}$$

これを再度解いてみます。

> $dsolve(\{\textbf{(4.1.2)}, x(0) = 0, D(x)(0) = 0\}, x(t));$

$$x(t) = \frac{49}{79999000} e^{-\frac{1}{20} t} \sin\left(\frac{1}{20} \sqrt{79999}\, t \right) \sqrt{79999} + \frac{49}{1000} e^{-\frac{1}{20} t} \cos\left(\frac{1}{20} \sqrt{79999}\, t \right) - \frac{49}{1000} \qquad \textbf{(4.1.3)}$$

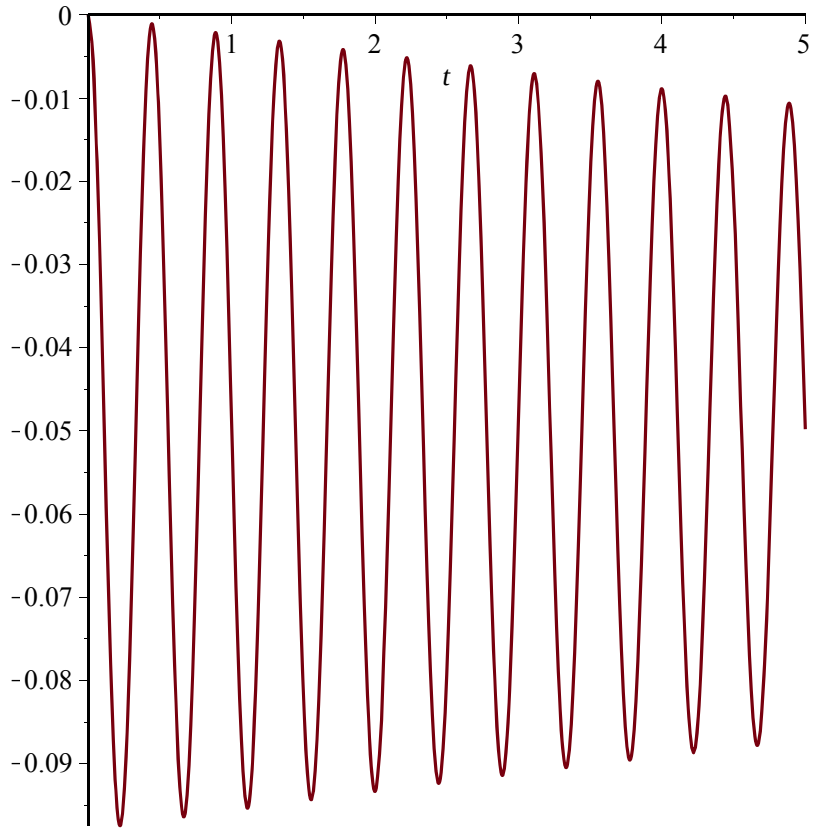
上式の右辺を関数化します。

> $ans := unapply(rhs(\textbf{(4.1.3)}), t);$

$$ans := t \rightarrow \frac{49}{79999000} e^{-\frac{1}{20} t} \sin\left(\frac{1}{20} \sqrt{79999}\, t \right) \sqrt{79999} + \frac{49}{1000} e^{-\frac{1}{20} t} \cos\left(\frac{1}{20} \sqrt{79999}\, t \right) - \frac{49}{1000} \qquad \textbf{(4.1.4)}$$

プロットしてみます。

```
> plot(ans(t), t = 0 ..5);
```



減衰していることが確認できます。

▼数値シミュレーション　ケース３

地面の変化を入れてみます。

▼地面の表現 *ground*(*t*) の変更＆運動方程式の再計算

• 地面の表現（ *ground*(*t*) ）

> $ground := t \to \frac{\sin(t)}{5}$

$$ground := t \to \frac{1}{5}\sin(t) \qquad \textbf{(5.1.1)}$$

• 運動方程式の再計算

> $Mass\cdot(diff(x(t),t\$2)+G)+c\cdot diff(x(t),t)+k\cdot(x(t)-ground(t))=0;$

$$\frac{d^2}{dt^2}x(t)+9.8+0.1\left(\frac{d}{dt}x(t)\right)+200\,x(t)-40\sin(t)=0 \qquad \textbf{(5.1.2)}$$

これを解いてみます。

> $dsolve(\{\textbf{(5.1.2)}, x(0)=0, D(x)(0)=0\}, x(t));$

$$x(t) = -\frac{15725555051}{316804119899000}\,e^{-\frac{1}{20}t}\sin\left(\frac{1}{20}\sqrt{79999}\,t\right)\sqrt{79999} \qquad \textbf{(5.1.3)}$$
$$+\frac{194444949}{3960101000}\,e^{-\frac{1}{20}t}\cos\left(\frac{1}{20}\sqrt{79999}\,t\right)-\frac{49}{1000}+\frac{796000}{3960101}\sin(t)$$
$$-\frac{400}{3960101}\cos(t)$$

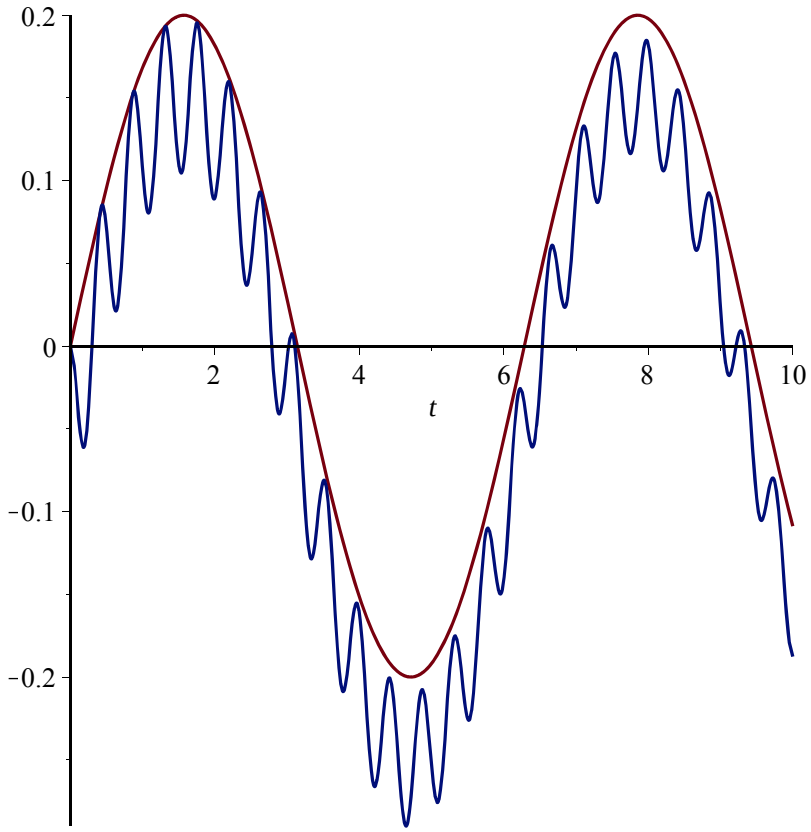
右辺を unapply コマンドにより関数化します。

> $ans := unapply(rhs(\textbf{(5.1.3)}), t);$

$$ans := t \to -\frac{15725555051}{316804119899000}\,e^{-\frac{1}{20}t}\sin\left(\frac{1}{20}\sqrt{79999}\,t\right)\sqrt{79999} \qquad \textbf{(5.1.4)}$$
$$+\frac{194444949}{3960101000}\,e^{-\frac{1}{20}t}\cos\left(\frac{1}{20}\sqrt{79999}\,t\right)-\frac{49}{1000}+\frac{796000}{3960101}\sin(t)$$
$$-\frac{400}{3960101}\cos(t)$$

プロットしてみます。

```
> plot({ground(t), ans(t)}, t=0..10);
```



赤は地面の高さで青が質点の高さです。

この地面のように強制的に振動させようとすることを強制振動と呼びます。

▼パラメーターの定義＆基本ブロック作成

上記内容をアニメーション表示でよりわかりやすくなります。

• アニメーションのフレーム（*frame*）

frame はアニメーションのフレームを追加していくための変数で、最初は空にしておきます。

```
> frame := [ ]:
```

• 車輪の半径（*wheelr*）

wheelr は車輪の半径です。

```
> wheelr := 1 :
```

• タイヤに繋がる直方体（*cuboidtire*）

cuboidtire はタイヤにつながっている部分を `cuboid` コマンドで作成しておきます。

対角の 2 点の座標で定義します。色は緑 (*green*) にします。

これを後で `translate` コマンドで平行移動してアニメーションします。

```
> cuboidtire := cuboid( [ -0.2, -0.2, -0.2 ], [ 0.2, 0.2, 1.2 ], color = green) :
```

• 載っている物体を現す立方体（*cuboidcarry*）

cuboidcarry は上に載っている物体を現す立方体です。色は赤にします。

```
> cuboidcarry := cuboid( [ -0.5, -0.5, 0 ], [ 0.5, 0.5, 1 ], color = red) :
```

▼アニメーション作成

以下の for ループで時間 (t が 0 から 1 秒まで) ごとにアニメーションのフレームを作成します。

スプリングは tubeplot を使って表現します。
スプリングの長さを springl で表します。

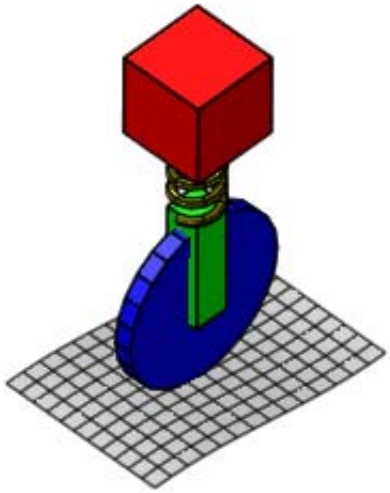
wheelh は車輪の中心の高さで carryh は上の質点 (赤) の中心の高さです。

質点の高さは dsolve による解の ans を使用します。

```
> for t from 0 by 0.05 to 5 do
  # t が 0 から 5 秒まで 0.05 秒刻みで下記を実行する。

   #車輪の中心の高さを定義する：
   wheelh := ground(t):

   #地面プロットを作成する
   pp := plot3d( [x, y, ground(x+t)], x = -Pi/2..Pi/2,
         y = -1..1, grid = [15, 12], color = gray ):

   #車輪プロットを今回のフレームに追加する：
   pp := pp, translate( rotate( cylinder( [0, 0, 0], wheelr,
         0.2, color = blue ), Pi/2, t*2*Pi, 0), 0, -0.1,
         wheelh + wheelr ):

   #車輪に繋がる直方体の位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( cuboidtire, 0, 0, wheelh + wheelr ):

   #質点の中心の高さを定義する：
   carryh := ans(t):

   #スプリングの長さを定義する：
   springl := carryh - wheelh + 1:

   #スプリングプロットを作成する：
   tt := tubeplot( [cos(u*8*Pi) / 4, sin(u*8*Pi) / 4,
         u*springl], u = 0..1, radius = 0.05,
         color = yellow ):

   #スプリングの位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( tt, 0, 0, wheelh + 2*wheelr + 0.2):

   #載っている物体の位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( cuboidcarry, 0, 0, wheelh +
         2*wheelr + 0.2 + springl):

   #アニメーションのフレームを追加していく
   frame := frame, display( pp ):
  end:
```

できたフレームをアニメーション表示します。

> *display*(*frame*, *insequence* = *true*, *scaling* = *constrained*);



▼数値シミュレーション　ケース4

地面が一段の階段状 (ステップ状) になっている場合のシミュレーションをしてみます。

▼パラメーターの定義

• 変数の初期化

時間の変数 *t* をリセットしておきます。

```
> t :='t';
```

$$t := t \tag{6.1.1}$$

• 質量（ *Mass* ）

質量は 1 (Kg) とします。

```
> Mass := 1;
```

• 減衰定数（ *c* ）

粘性減衰定数は 0.3 とします。

```
> c := 0.3;
```

$$\begin{aligned} Mass &:= 1 \\ c &:= 0.3 \end{aligned} \tag{6.1.2}$$

▼地面の表現 *ground*(*t*) の変更＆運動方程式の再計算

地面の変化を入れてみます。

階段関数には `Heaviside` 関数を使用します。これは超関数 (δ関数の積分) です。

超関数がわからなくても以下のプロットのような階段上の関数だと思えば問題ありません。

非常に小さい値 *eps* を使用します。
これは超関数 `Heaviside` の階段の点での値が定義されないので、わずかにずらすためです。
超関数 `Heaviside` の詳細についてはヘルプを参照ください。

```
> eps := 10^(-10);
```

$$\tag{6.2.1}$$

$$eps := \frac{1}{10000000000} \tag{6.2.1}$$

ステップ関数を定義します。

• 地面の表現（ $ground(t)$ ）

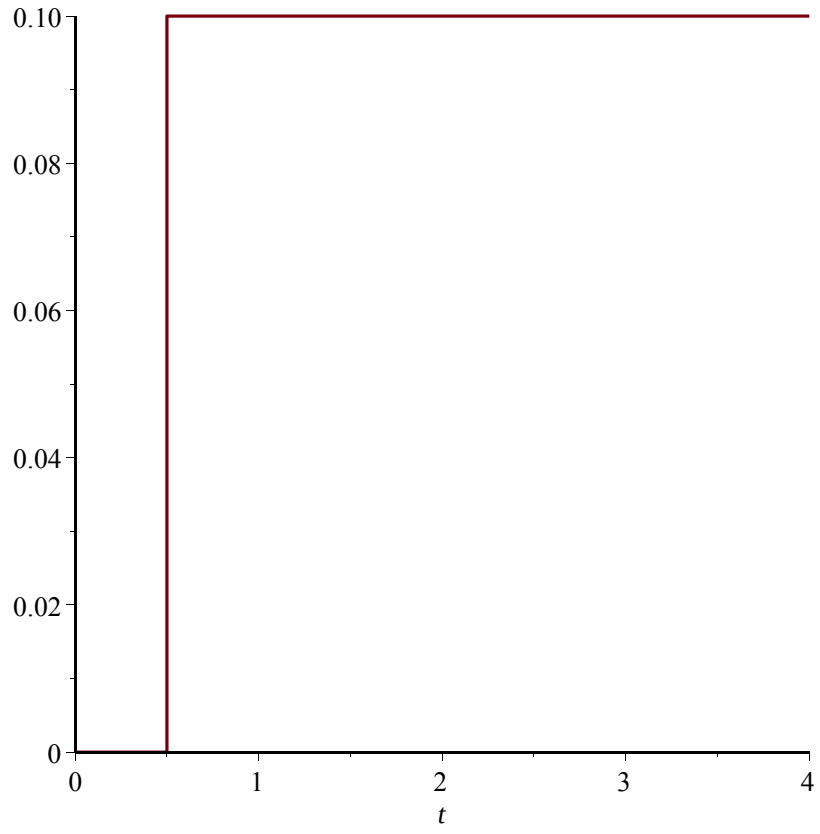
$t = 0.5(+ eps)$ の点でステップが発生します。

> $ground := t \rightarrow 0.1\ \mathrm{Heaviside}(t - 0.5 + eps)$

$$ground := t \rightarrow 0.1\ \mathrm{Heaviside}(t - 0.5 + eps) \tag{6.2.2}$$

プロットしてみます。

> $plot(ground(t), t = 0..4);$



• 運動方程式の再計算

> $Mass \cdot (diff(x(t), t\$2) + G) + c \cdot diff(x(t), t) + k \cdot (x(t) - ground(t)) = 0;$

$$\frac{d^2}{dt^2}x(t)+9.8+0.3\left(\frac{d}{dt}x(t)\right)+200\,x(t)-20.0\,\mathrm{Heaviside}(t-0.4999999999)=0 \quad \textbf{(6.2.3)}$$

これを解いてみます。

```
> dsolve({(6.2.3), x(0)=0, D(x)(0)=0}, x(t));
```

$$\begin{aligned}x(t) = {} & \frac{147}{79991000}\,e^{-\frac{3}{20}t}\sin\left(\frac{1}{20}\sqrt{79991}\,t\right)\sqrt{79991} \\ & +\frac{49}{1000}\,e^{-\frac{3}{20}t}\cos\left(\frac{1}{20}\sqrt{79991}\,t\right)-\frac{49}{1000}+\frac{1}{10}\,\mathrm{Heaviside}\left(t-\frac{4999999999}{10000000000}\right)-\frac{3}{799910}\sqrt{79991}\,\mathrm{Heaviside}\left(t-\frac{4999999999}{10000000000}\right) \\ & e^{-\frac{3}{20}t+\frac{14999999997}{200000000000}}\sin\left(\frac{1}{200000000000}\sqrt{79991}\,(10000000000\,t-4999999999)\right)-\frac{1}{10}\,\mathrm{Heaviside}\left(t-\frac{4999999999}{10000000000}\right) \\ & e^{-\frac{3}{20}t+\frac{14999999997}{200000000000}}\cos\left(\frac{1}{200000000000}\sqrt{79991}\,(10000000000\,t-4999999999)\right)\end{aligned} \quad \textbf{(6.2.4)}$$

このように超関数が入っていても `dsolve` コマンドは解を求めることができます。

右辺を `unapply` コマンドを使って関数化します。

```
> ans := unapply(rhs((6.2.4)), t);
```

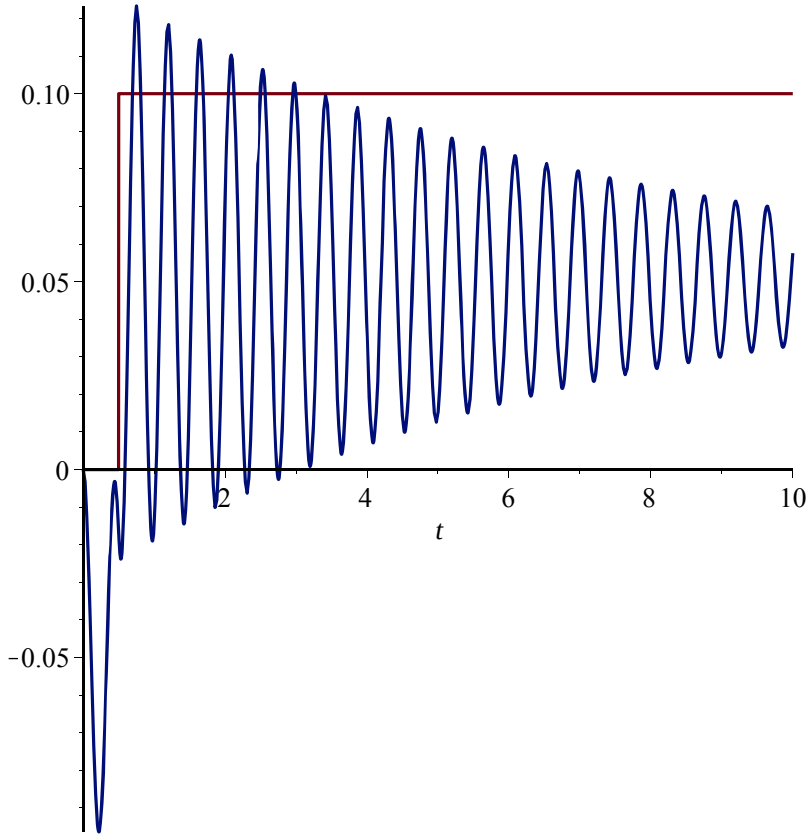
$$\begin{aligned}ans := t\rightarrow {} & \frac{147}{79991000}\,e^{-\frac{3}{20}t}\sin\left(\frac{1}{20}\sqrt{79991}\,t\right)\sqrt{79991} \\ & +\frac{49}{1000}\,e^{-\frac{3}{20}t}\cos\left(\frac{1}{20}\sqrt{79991}\,t\right)-\frac{49}{1000}+\frac{1}{10}\,\mathrm{Heaviside}\left(t-\frac{4999999999}{10000000000}\right)-\frac{3}{799910}\sqrt{79991}\,\mathrm{Heaviside}\left(t-\frac{4999999999}{10000000000}\right) \\ & e^{-\frac{3}{20}t+\frac{14999999997}{200000000000}}\sin\left(\frac{1}{200000000000}\sqrt{79991}\,(10000000000\,t-4999999999)\right)-\frac{1}{10}\,\mathrm{Heaviside}\left(t-\right.\end{aligned} \quad \textbf{(6.2.5)}$$

$$-\frac{4999999999}{10000000000}\Bigg)$$

$$e^{-\frac{3}{20}t+\frac{14999999997}{200000000000}}\cos\Bigg(\frac{1}{200000000000}\sqrt{79991}\,(10000000000\,t-4999999999)\Bigg)$$

プロットしてみます。

```
> plot({ans(t), ground(t)}, t=0..10);
```



ステップ応答の様子がよくわかります。

▼基本ブロック作成

これをまたアニメーション表示してみます。

• アニメーションのフレーム（*frame*）

frame はアニメーションのフレームを追加していくための変数で、最初は空に

しておきます。

```
> frame := [ ]:
```

• 車輪の半径（ *wheelr* ）

wheelr は車輪の半径です。

```
> wheelr := 1 :
```

• タイヤに繋がる直方体（ *cuboidtire* ）

cuboidtire はタイヤにつながっている部分を `cuboid` コマンドで作成しておきます。

対角の 2 点の座標で定義します。色は緑 (*green*) にします。

これを後で `translate` コマンドで平行移動してアニメーションします。

```
> cuboidtire := cuboid( [ -0.2, -0.2, -0.2 ], [ 0.2, 0.2, 1.2 ], color = green) :
```

• 載っている物体を現す立方体（ *cuboidcarry* ）

cuboidcarry は上に載っている物体を現す立方体です。色は赤にします。

```
> cuboidcarry := cuboid( [ -0.5, -0.5, 0 ], [ 0.5, 0.5, 1 ], color = red) :
```

▼アニメーション作成

以下の for ループで時間 (t が 0 から 1 秒まで) ごとにアニメーションのフレームを作成します。

スプリングは tubeplot を使って表現します。
スプリングの長さを springl で表します。

wheelh は車輪の中心の高さで carryh は上の質点 (赤) の中心の高さです。

```
> for t from 0 by 0.05 to 5 do
  # t が 0 から 5 秒まで 0.05 秒刻みで下記を実行する。

   #車輪の中心の高さを定義する：
   wheelh := ground(t):

   #車輪の中心の高さを定義する：
   pp := plot3d( [x, y, ground(x+t)], x = -Pi/2..Pi/2,
         y = -1..1, grid = [15, 12], color = gray ):

   #車輪プロットを今回のフレームに追加する：
   pp := pp, translate( rotate( cylinder( [0, 0, 0], wheelr,
         0.2, color = blue ), Pi/2, t*2*Pi, 0), 0, -0.1,
         wheelh + wheelr ):

   #車輪に繋がる直方体の位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( cuboidtire, 0, 0, wheelh + wheelr ):

   #質点の中心の高さを定義する：
   carryh := ans(t):

   #スプリングの長さを定義する：
   springl := carryh - wheelh + 1:

   #スプリングプロットを作成する：
   tt := tubeplot( [cos(u*8*Pi) / 4, sin(u*8*Pi) / 4,
         u*springl], u = 0..1, radius = 0.05,
         color = yellow ):

   #スプリングの位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( tt, 0, 0, wheelh + 2*wheelr + 0.2):

   #載っている物体の位置を時間によって変更させてフレームに追加する：
   pp := pp, translate( cuboidcarry, 0, 0, wheelh +
         2*wheelr + 0.2 + springl):

   #アニメーションのフレームを追加していく
   frame := frame, display( pp ):
  end:
```

できたフレームをアニメーション表示します。

```
> display( frame, insequence = true, scaling = constrained);
```