Computing project Part III

Assignment 6 − Symmetric Top

A couple of things have to change compared to the previous assignments where we used the Hamiltonian to determine time−derivatives. Here we already have an explicit form for the equations of motion (i.e. an expression for the time−derivatives) and we merely have to numerically integrate it with respect to time.
The equations of motion are given by:

```
In[1]:= ω̇1[ω2_, ω3_] := ((I2 - I3) / I1) ω2 ω3;
        ω̇2[ω1_, ω3_] := ((I3 - I1) / I2) ω1 ω3;
        ω̇3[ω1_, ω2_] := ((I1 - I2) / I3) ω1 ω2;
```

However, it turns out to be far more convenient to define the time−derivative as a function that operates on a vector {ω1,ω2,ω3} and returns a vector:

```
In[4]:= ωdot[ω_] := {((I2 - I3) / I1) ω[[2]] ω[[3]],
           ((I3 - I1) / I2) ω[[1]] ω[[3]], ((I1 - I2) / I3) ω[[1]] ω[[2]]};
```

We still use i as a counter, with maximum value imax:

```
In[5]:= imax = 1000;
```

Time starts at t0 and timestep size is given by dt. Values are stored in an array T:

```
In[6]:= t0 = 0;
        dt = 0.01;
        T = Table[t0, {i, 1, imax}];
```

Moments of inertia I1, I2, I3 are fixed:

```
In[9]:= I1 = 2;
        I2 = 2;
        I3 = 4;
```

Define array to store energy in:

```
In[12]:= EE = Table[0, {i, 1, imax}];
```

■ 1

Set initial conditions for ω and define array to store its values in:

```
In[13]:= ω0 = {0, 0, 3};
         ω = Table[ω0, {i, 1, imax}];
```

Now to do the actual calculation, according to the predictor−corrector method. Note how simple this looks with ω and ωdot defined as/on vectors.

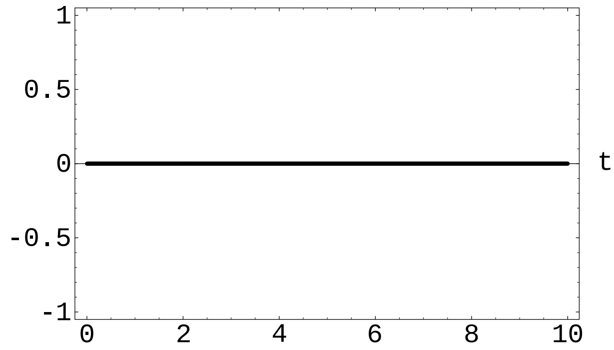
```
In[15]:= Do[ T[[i]] = T[[i - 1]] + dt;
          ωpred = ω[[i - 1]] + ωdot[ ω[[i - 1]] ] dt;
          ω[[i]] = ω[[i - 1]] + (ωdot[ ω[[i - 1]] ] + ωdot[ ωpred ]) dt / 2;
          , {i, 2, imax}]
```

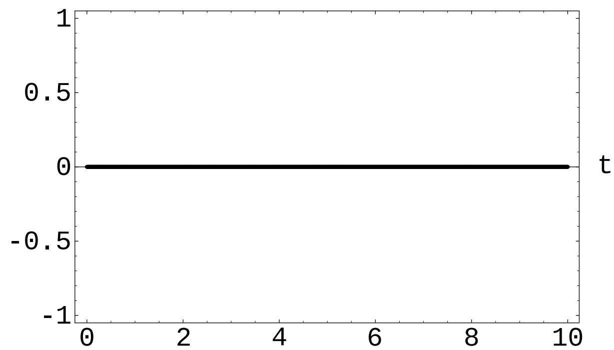
Calculate the energy:

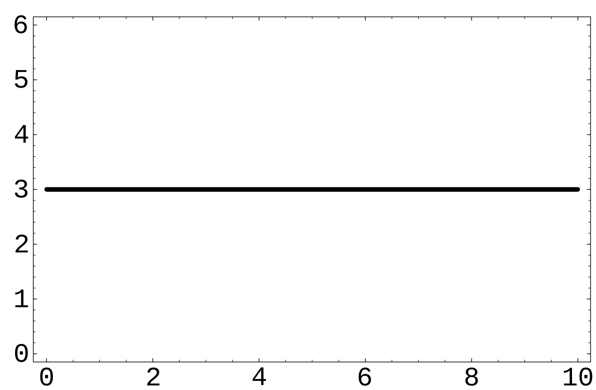
```
In[16]:= EE = I1 Transpose[ω][[1]]^2 + I2 Transpose[ω][[2]]^2 + I3 Transpose[ω][[3]]^2;
```

Plot ω1, ω2 and ω3 as function of time. Plot energy as function of time.

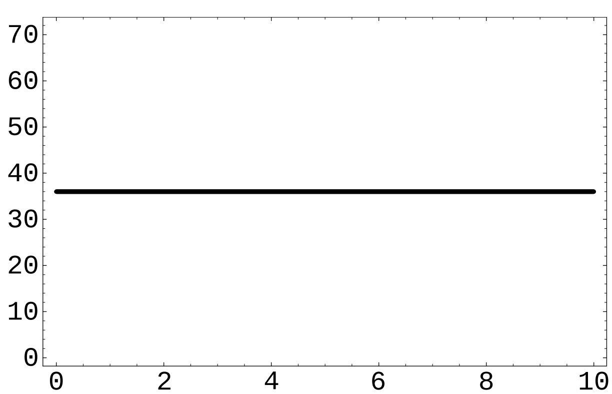
```
In[17]:= ListPlot[Transpose[{T, Transpose[ω][[1]]}], Frame → True, AxesLabel → {"t", "ω1"}];
         ListPlot[Transpose[{T, Transpose[ω][[2]]}], Frame → True, AxesLabel → {"t", "ω1"}];
         ListPlot[Transpose[{T, Transpose[ω][[3]]}], Frame → True, AxesLabel → {"t", "ω1"}];
```








```
In[18]:= ListPlot[Transpose[{T, EE}], Frame → True, AxesLabel → {"T", "Energy"}];
```



Mmmh, it seems as if nothing happens. But wait, that is exactly what we would expect for these initial conditions. The vector ω precesses around the z axis with zero amplitude, i.e., it stays constant.

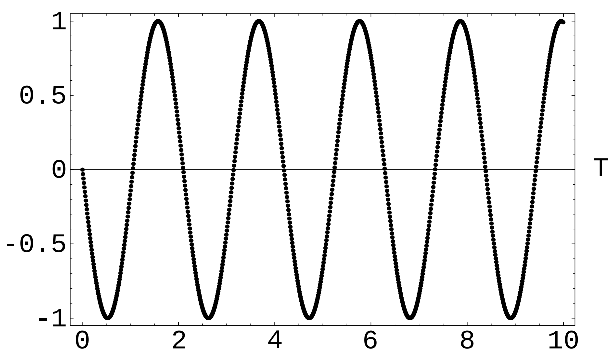
■ 2

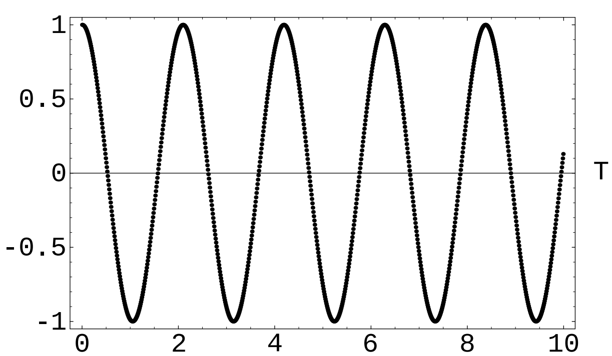
Now for some different initial conditions, where we would expect small oscillations around the z axis.

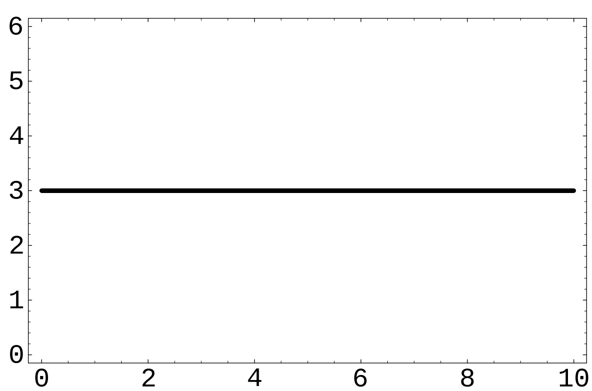
```
In[19]:=  ω0 = {0, 1, 3};
          ω = Table[ω0, {i, 1, imax}];
```

Calculate ω, energy and plot $\omega 1$, $\omega 2$, $\omega 3$ and energy.

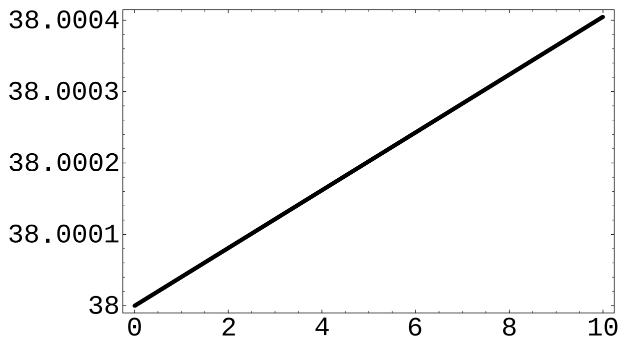
```
In[21]:=  Do[ T[[i]] = T[[i - 1]] + dt;
           ωpred = ω[[i - 1]] + ωdot[ ω[[i - 1]] ] dt;
           ω[[i]] = ω[[i - 1]] + (ωdot[ ω[[i - 1]] ] + ωdot[ ωpred ]) dt / 2;
           , {i, 2, imax}]

In[22]:=  EE = I1 Transpose[ω][[1]]^2 + I2 Transpose[ω][[2]]^2 + I3 Transpose[ω][[3]]^2;

In[23]:=  ListPlot[Transpose[{T, Transpose[ω][[1]]}], Frame → True, AxesLabel → {"T", "ω1"}];
          ListPlot[Transpose[{T, Transpose[ω][[2]]}], Frame → True, AxesLabel → {"T", "ω1"}];
          ListPlot[Transpose[{T, Transpose[ω][[3]]}], Frame → True, AxesLabel → {"T", "ω1"}];
```







```
In[24]:= ListPlot[Transpose[{T, EE}], Frame → True, AxesLabel → {"T", "Energy"}];
```



Yes, indeed oscillations around the z axis; the energy stays constant up to some very small error due to our numerical approximation.