Attitude Stabilization with Magnetic Actuation

12.07.2012 by jph

- *units are SI*
- *vectors are in satellite body coordinates unless otherwise stated*
- *center of mass of satellite is at (0,0,0)*

Torque generated by square coil

Edges of the coils

We derive the torque for a wire loop of square shape in the y-z plane.
4 edges with the following length and direction vectors

```
edg = {{0, L, 0}, {0, 0, -L}, {0, -L, 0}, {0, 0, L}};
```

Midpoints of the edges of the coils

The distance from the center of the coil to the midpoint of each edge is r

```
pnt = {ofs1, ofs2, ofs3} + # & /@ {{0, 0, r}, {0, r, 0}, {0, 0, -r}, {0, -r, 0}};
```

Torque excerted on satellite

The force F of a current I going through a straight wire in direction L in a magnet field B is $F = I\,(B \times L)$.

We assume that the coil has n windings and that a current I_X flows through the wires. Then, the torque accumulates as
$\tau = n \sum_{i=1}^{4} \text{pnt}_i \times (I\,(B \times L))$
$\tau = n \sum_{i=1}^{4} \text{pnt}_i \times (I\,(B \times \text{egd}_i))$

For a general B, we obtain

```
B = {B1, B2, B3}
```

```
Tor = n Ix Sum[Cross[pnt[[i]], Cross[B, edg[[i]]]], {i, 1, 4}] // Chop
```

```
{B1, B2, B3}
```

```
{0, -2 L n r B3 Ix, 2 L n r B2 Ix}
```

Relevant constants of satellite (in SI units)

Inertia tensor of satellite (at center of mass)

SI unit of inertia entry is kg * (m^2)

```
INE = {{i11, i12, i13}, {i12, i22, i23}, {i13, i23, i33}} /.
    {i11 → 0.005, i22 → 0.004, i33 → 0.003, i12 → -0.000002, i13 → -0.000002, i23 → -0.000003};
INE // MatrixForm
```

$$\begin{pmatrix} 0.005 & -2.\times 10^{-6} & -2.\times 10^{-6} \\ -2.\times 10^{-6} & 0.004 & -3.\times 10^{-6} \\ -2.\times 10^{-6} & -3.\times 10^{-6} & 0.003 \end{pmatrix}$$

The Eigenvectors and Eigenvalues of the inertia tensor determine the long term behaviour of the satellite. More specifically, around which axis the satellite will rotate, if the satellite is subject only to small random distrubances.

```
{val, vec} = Eigensystem[INE];
Print[#1] & /@ vec;
```

```
{0.999998, -0.00199699, -0.000996999}
{-0.00199399, -0.999994, 0.00300395}
{0.00100299, 0.00300196, 0.999995}
```

- **Maximum current in the coils X, Y, Z**

the currents can flow in either directions, but are each capped at a maximum amplitude:

```
IMAXX = 0.05;
IMAXY = 0.09;
IMAXZ = 0.1;
```

- **Force of current carrying straight wire in a magnetic field**

The amplitude of the Earth magnetic field ranges between approximately 25000 and 65000 nT, where 1 nT = 10^-9 T. The SI unit for the entries of B is T = N / (A m)

```
AMP = 65000. × 10^-9
```

```
0.000065
```

- **Torque excerted on satellite by coils X, Y, Z**

using the formulas in the first section
we simply add up:

```
TorX = {0, -1.08 B3 IX, 1.08 B2 IX};
TorY = {1.08 B3 IY, 0, -1.08 B1 IY};
TorZ = {-1.08 B2 IZ, 1.08 B1 IZ, 0};
TOR = TorX + TorY + TorZ
```

```
{1.08 B3 IY - 1.08 B2 IZ, -1.08 B3 IX + 1.08 B1 IZ, 1.08 B2 IX - 1.08 B1 IY}
```

Example: if the magnetic field B is known for instance as
$B = 0.000065\ (0.2, 0.6, 0.8)$
and the currents in the coils at their max
then the torque would be of magnitude about

```
0.000065 TOR /. {B1 → -0.2, B2 → 0.8, B3 → 0.5, IX → IMAXX, IY → IMAXY, IZ → IMAXZ} // Simplify
% // Norm
```

$$\left\{-2.457\times 10^{-6}, -3.159\times 10^{-6}, 4.0716\times 10^{-6}\right\}$$

$$5.70912\times 10^{-6}$$

Equations of rotational rate

Rotation of satellite under conservation of momentum

With I=INE is the inertia tensor, and ω is the rotational rate vector, and τ the torque, the differential equation is

$I.d_t\,\omega = -\omega\times(I.\omega) + \tau$

```
vω = {ω1[t], ω2[t], ω3[t]};
dω = ∂t {ω1[t], ω2[t], ω3[t]};
eqs = 0 == #1 & /@ (INE.dω + Cross[vω, INE.vω] - {τ1, τ2, τ3})
```

$$\begin{aligned}\{0 == &-\tau_1 - 2.\times10^{-6}\,\omega_1[t]\,\omega_2[t] - 3.\times10^{-6}\,\omega_2[t]^2 + 2.\times10^{-6}\,\omega_1[t]\,\omega_3[t] - \\ &0.001\,\omega_2[t]\,\omega_3[t] + 3.\times10^{-6}\,\omega_3[t]^2 + 0.005\,\omega_1'[t] - 2.\times10^{-6}\,\omega_2'[t] - 2.\times10^{-6}\,\omega_3'[t], \\ 0 == &-\tau_2 + 2.\times10^{-6}\,\omega_1[t]^2 + 3.\times10^{-6}\,\omega_1[t]\,\omega_2[t] + 0.002\,\omega_1[t]\,\omega_3[t] - 2.\times10^{-6}\,\omega_2[t]\,\omega_3[t] - \\ &2.\times10^{-6}\,\omega_3[t]^2 - 2.\times10^{-6}\,\omega_1'[t] + 0.004\,\omega_2'[t] - 3.\times10^{-6}\,\omega_3'[t], \\ 0 == &-\tau_3 - 2.\times10^{-6}\,\omega_1[t]^2 - 0.001\,\omega_1[t]\,\omega_2[t] + 2.\times10^{-6}\,\omega_2[t]^2 - 3.\times10^{-6}\,\omega_1[t]\,\omega_3[t] + \\ &2.\times10^{-6}\,\omega_2[t]\,\omega_3[t] - 2.\times10^{-6}\,\omega_1'[t] - 3.\times10^{-6}\,\omega_2'[t] + 0.003\,\omega_3'[t]\}\end{aligned}$$

Simulation of angular rate and satellite attitude without control torque

Demonstration that mathematica solves the differential equations accurately

Mathematica solves the non-linear (!) differential equations of angular rate ω accurately:

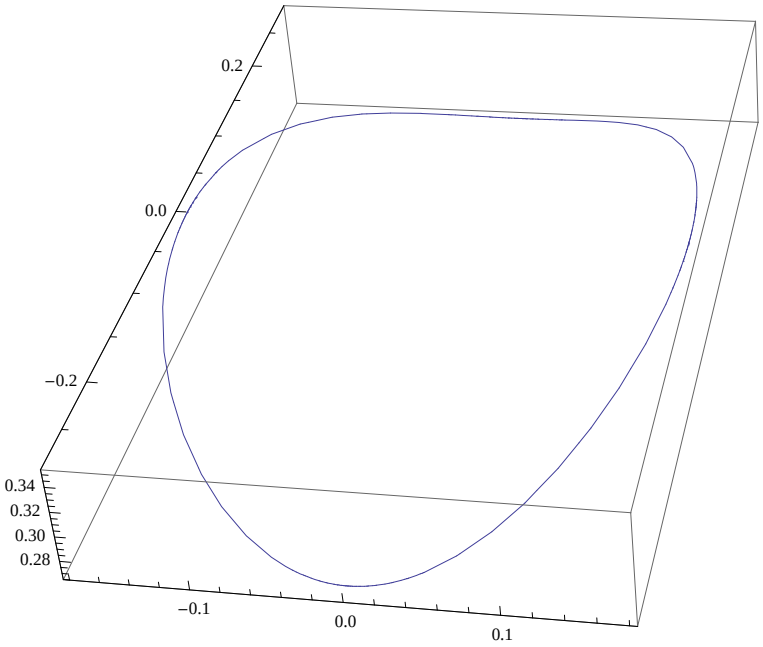
• in the absence of torque, the angular momentum is preserved.

• after one cycle of ω, the orientation of the satellite matches the initial orientation

The differential equation that governs the orientation R is

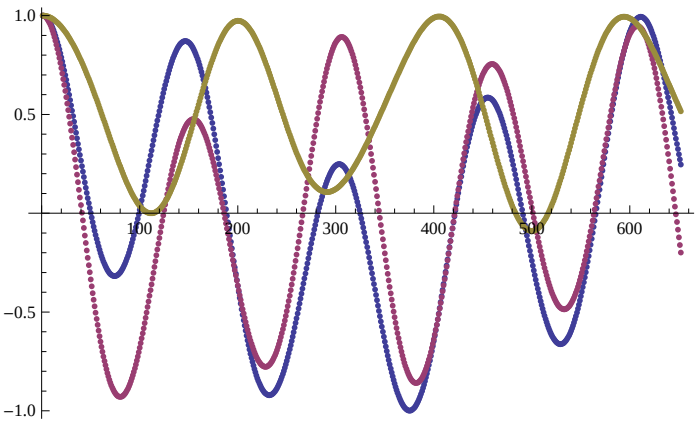
$d_t\,R = R.\exp(\Omega\ \mathrm{dt})$

```
f[R_, {dω1_, dω2_, dω3_}] := R.MatrixExp[{{0, -dω3, dω2}, {dω3, 0, -dω1}, {-dω2, dω1, 0}}]
T = 65;
degPsec = 0.0174532925;
sol = {ω1, ω2, ω3} /. NDSolve[Join[eqs /. {τ1 → 0, τ2 → 0, τ3 → 0},
       {ω1[0] == 10 degPsec, ω2[0] == 3 degPsec, ω3[0] == 20 degPsec}],
      {ω1, ω2, ω3},
      {t, 0, T}][[1]];
crv[t_] = #[t] & /@ sol;
ParametricPlot3D[crv[t], {t, 0, T}]
dt = 0.1;
ORI = Fold[f, IdentityMatrix[3], Table[dt crv[k], {k, 0, T, dt}]];
ORI // MatrixForm
ORL = FoldList[f, IdentityMatrix[3], Table[dt crv[k], {k, 0, T, dt}]];
Dimensions[ORL]
ListPlot[{ORL[[All, 1, 1]], ORL[[All, 2, 2]], ORL[[All, 3, 3]]}]
```



$$\begin{pmatrix} 0.24658 & -0.663203 & 0.706654 \\ 0.85186 & -0.199356 & -0.484346 \\ 0.462096 & 0.721401 & 0.515799 \end{pmatrix}$$

{652, 3, 3}



■ Simulation of control

• every 5 sec measure magnetic field

• in every 0...5 sec block apply magnetic torquers between t=2...5 sec

• during the simulation time, the external magnetic field vector B remains constant in world coordinates, but is therefore not constant in satellite body coordinates

```
Bom = {0, 0, 0};
STEP[{B_, R0_, ω0_, τ0_, t0_, dt_, CTr_, func_}] := Module[
   {Bo = Inverse[R0].B, R1, ω1 = ω0, smp = N[dt] Range[0, 5 - 1] / 5,
    mag, tor, cτr, t1 = t0 + dt, seg = Mod[t0, 5]},
   If[seg == 1, Bom = Bo];
   cτr = If[2 ≤ seg,
     func[Bom, ω0, t0],
     {Iₓ → 0, Iᵧ → 0, I_z → 0}
    ];
   mag = {B₁ → Bo⟦1⟧, B₂ → Bo⟦2⟧, B₃ → Bo⟦3⟧};
   tor = TOR /. cτr /. mag;
   sol = {ω₁, ω₂, ω₃} /. NDSolve[Join[eqs /. {τ₁ → tor⟦1⟧, τ₂ → tor⟦2⟧, τ₃ → tor⟦3⟧},
         {ω₁[0] == ω0⟦1⟧, ω₂[0] == ω0⟦2⟧, ω₃[0] == ω0⟦3⟧}],
        {ω₁, ω₂, ω₃}, {t, 0, dt}]⟦1⟧;
   crv[t_] = #[t] & /@ sol;
   R1 = Fold[f, R0, Table[smp⟦2⟧ crv[smp⟦k⟧], {k, Length[smp]}]];
   {B, Orthogonalize[R1], crv[dt], cτr, t1, dt, {Tor_x, Tor_y, Tor_z} /. mag /. cτr, func}
  ];
DISP[sim_] := Module[{},
   Print["Evolution of angular rate, in 3d, and norm of angular rate:"];
   Print[
    {ListPlot[Transpose[sim⟦All, 3⟧], PlotRange → All], ListPointPlot3D[sim⟦All, 3⟧],
     ListPlot[Norm /@ sim⟦All, 3⟧, PlotRange → All]}];
   Print["Control parameters: Iₓ Iᵧ I_z"];
   Print[{ListPlot[Iₓ /. sim⟦All, 4⟧],
     ListPlot[Iᵧ /. sim⟦All, 4⟧], ListPlot[I_z /. sim⟦All, 4⟧]}];
   Print["Orientation (3 vectors of state matrix):"];
   Print[{ListPlot[Transpose[sim⟦All, 2, 1⟧]], ListPlot[Transpose[sim⟦All, 2, 2⟧]],
     ListPlot[Transpose[sim⟦All, 2, 3⟧]]}];];
```

- **Initial conditions**

To compare different control strategies, we define a set of initial conditions

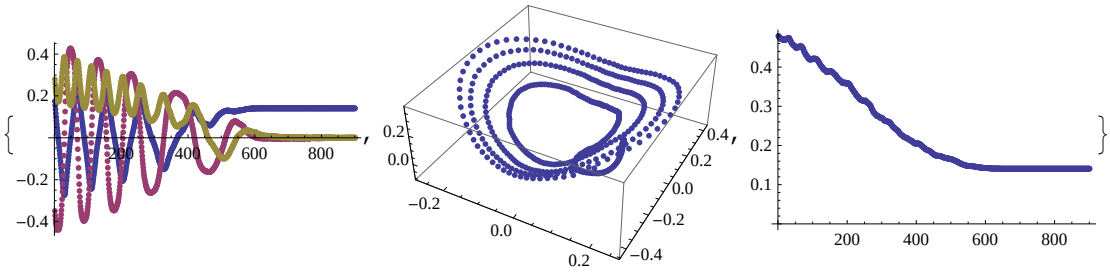
```
INIT = {
   {AMP {0, 0, 1}, IdentityMatrix[3],
    {10, -20, 16} degPsec, {Iₓ → 0, Iᵧ → 0, I_z → 0}, 0, 1, 0 IdentityMatrix[3]},
   {AMP {0, 0, 1}, IdentityMatrix[3], {16, 10, -20} degPsec,
    {Iₓ → 0, Iᵧ → 0, I_z → 0}, 0, 1, 0 IdentityMatrix[3]}
  };
```

- **Control using simple discrete optimization**

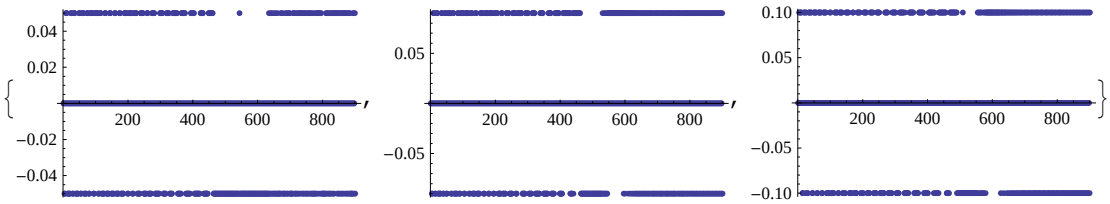
An alternative control strategy that minimizes the scalar product of torque τ and ω, manages to stabilize attitude slightly faster.

```
box[cx_, cy_, cz_] := {
    If[cx == 1, IMAX_X, -IMAX_X],
    If[cy == 1, IMAX_Y, -IMAX_Y],
    If[cz == 1, IMAX_Z, -IMAX_Z]
  };
STEEP[Bom_, ω0_, t0_] := Module[
  {cor = Flatten[Array[box, {2, 2, 2}], 2],
   subbf = {B_1 → Bom[[1]], B_2 → Bom[[2]], B_3 → Bom[[3]]},
   subst, argsbt = {I_X → 0, I_Y → 0, I_Z → 0}, min = 0, val},
  For[i = 1, i ≤ Length[cor], i++,
   subst = {I_X → cor[[i, 1]], I_Y → cor[[i, 2]], I_Z → cor[[i, 3]]};
   val = TOR.ω0 /. subst /. subbf;
   If[val < min, min = val; argsbt = subst]
  ]; argsbt
 ]
DISP[NestList[STEP, Append[#1, STEEP], 900]] & /@ INIT;
```

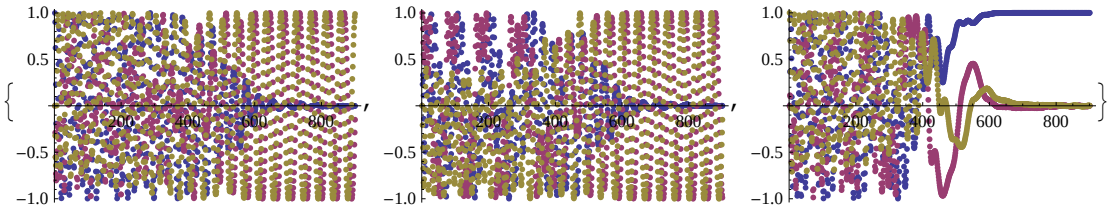
Evolution of angular rate, in 3d, and norm of angular rate:



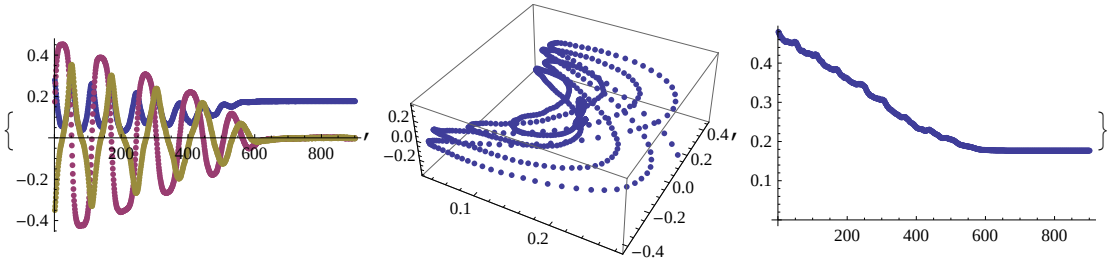
Control parameters: I_X I_Y I_Z



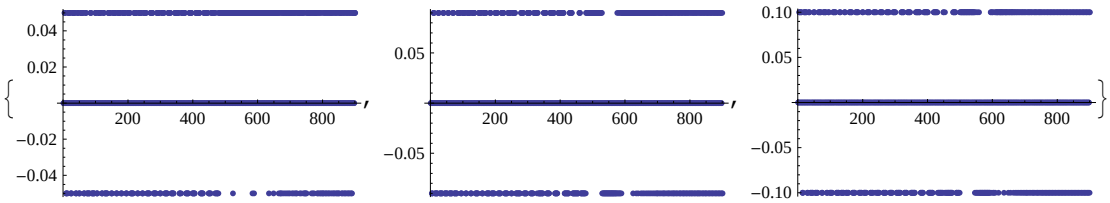
Orientation (3 vectors of state matrix):



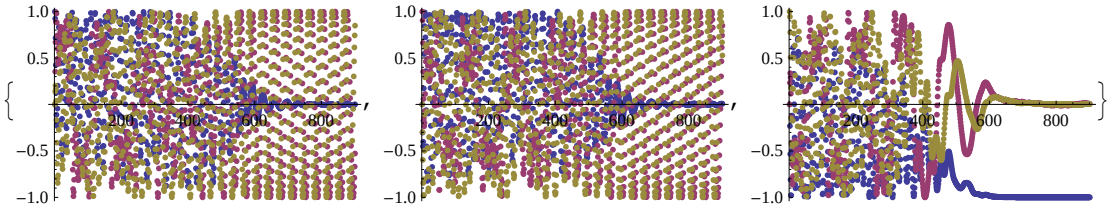
Evolution of angular rate, in 3d, and norm of angular rate:



Control parameters: I_X I_Y I_Z



Orientation (3 vectors of state matrix):



■ Binary export for visualization

```
CppTensorWrite[file_, tensor_, type_: "Real64"] := Module[
    {ndims = Length[Dimensions[tensor]]},
    BinaryWrite[file, ndims, "Integer32"];
    BinaryWrite[file, #1, "Integer32"] & /@ Dimensions[tensor];
    BinaryWrite[file, #1, type] & /@ Flatten[Transpose[tensor, Reverse[Range[ndims]]]];
    Close[file];
    Print[file <> " " <> ToString[Dimensions[tensor]]];
   ];

ECPP[res_] := Module[{dir = "d:\\cpp9\\attitude\\get\\nb\\"},
   CppTensorWrite[dir <> "bfl.dat", Transpose[Append[#1, 0] & /@ DAT[[All, 1]], {2, 1}]];
   CppTensorWrite[dir <> "pos.dat",
     Transpose[{{#1[[1, 1]], #1[[1, 2]], #1[[1, 3]], 0},
                {#1[[2, 1]], #1[[2, 2]], #1[[2, 3]], 0},
                {#1[[3, 1]], #1[[3, 2]], #1[[3, 3]], 0},
                {0, 0, 0, 1}} & /@ DAT[[All, 2]], {3, 1, 2}]];
   CppTensorWrite[dir <> "ome.dat", Transpose[Append[#1, 0] & /@ DAT[[All, 3]], {2, 1}]];
   CppTensorWrite[dir <> "ctr.dat", Transpose[{I_X, I_Y, I_Z} /. DAT[[All, 4]], {2, 1}]];
   CppTensorWrite[dir <> "trq.dat", Transpose[{Append[#1, 0] & /@ DAT[[All, 7, 1]],
       Append[#1, 0] & /@ DAT[[All, 7, 2]], Append[#1, 0] & /@ DAT[[All, 7, 3]]}, {2, 3, 1}]];
  ]

DAT = NestList[STEP, Append[INIT[[1]], STEEP], 600];
ECPP[DAT]
```