magnetforces

	Section	Page
Calculating forces between magnets	2	1
Functions for calculating forces and stiffnesses	8	4

1. About this file. This is a 'literate programming' approach to writing Matlab code using MATLABWEB. To be honest I don't know if it's any better than simply using the Matlab programming language directly. But ain't this documentation nice!

2. Calculating forces between magnets. This is the source to some code to calculate the forces (and perhaps torques) between two cuboid-shaped magnets with arbitary displacement and magnetisation direction.

If this code works then I'll look at calculating the forces for magnets with rotation as well.

⟨ `magnetforces.m` 2 ⟩ ≡

[*forces* *torques*] =
function *magnetforces*(*magnet_fixed*, *magnet_float*, *magnet_disp*);
⟨ Extract input variables 3 ⟩
⟨ Decompose orthogonal superpositions 4 ⟩
⟨ Transform appropriate coordinate systems 5 ⟩
⟨ Calculate forces 6 ⟩
⟨ Recombine results 7 ⟩
end
⟨ Functions for calculating forces and stiffnesses 8 ⟩

3. First of all, address the data structures required for the input and output. Because displacement of a single magnet has three components, plus sizes of the faces another three, plus magnetisation strength and direction (two) makes nine in total, we use one of Matlab's structures to pass the information into the function. Otherwise we'd have an overwhelming number of input arguments.

We use spherical coordinates to represent magnetisation angle, where *theta* is the angle from the vertical ($0 < \theta < \pi$) and *phi* is the angle around the horizontal plane ($0 < \phi < 2\pi$).

$\langle$ Extract input variables 3 $\rangle \equiv$

$a_1 = 0.5 * magnet\_fixed.dim(1);$
$b_1 = 0.5 * magnet\_fixed.dim(2);$
$c_1 = 0.5 * magnet\_fixed.dim(3);$
$a_2 = 0.5 * magnet\_float.dim(1);$
$b_2 = 0.5 * magnet\_float.dim(2);$
$c_2 = 0.5 * magnet\_float.dim(3);$

$J1r = magnet\_fixed.magn;$
$J2r = magnet\_float.magn;$
$J1t = magnet\_fixed.magdir(1);$
$J2t = magnet\_float.magdir(1);$
$J1p = magnet\_fixed.magdir(2);$
$J2p = magnet\_float.magdir(2);$

$dx = magnet\_disp(1);$
$dy = magnet\_disp(2);$
$dz = magnet\_disp(3);$

This code is used in section 2.

4. Superposition is used to turn an arbitrary magnetisation angle into a set of orthogonal magnetisations.

Each magnet can potentially have three components, which can result in up to nine force calculations for a single magnet.

$\langle$ Decompose orthogonal superpositions 4 $\rangle \equiv$

$J1x = J1r * \cos(J1p) * \sin(J1t);$
$J1y = J1r * \sin(J1p) * \sin(J1t);$
$J1z = J1r * \cos(J1t);$

$J2x = J2r * \cos(J2p) * \sin(J2t);$
$J2y = J2r * \sin(J2p) * \sin(J2t);$
$J2z = J2r * \cos(J2t);$

This code is used in section 2.

5. The expressions we have to calculate the forces assume a fixed magnet with positive z magnetisation only. Secondly, magnetisation direction of the floating magnet may only be z- or x-directions.

$\langle$ Transform appropriate coordinate systems 5 $\rangle \equiv$

This code is used in section 2.

6. Next

$\langle$ Calculate forces 6 $\rangle \equiv$

This code is used in section 2.

7. Next

$\langle$ Recombine results 7 $\rangle \equiv$

This code is used in section 2.

8. Functions for calculating forces and stiffnesses. The calculations for forces between differently-oriented cuboid magnets are all directly from the literature. The stiffnesses have been derived by differentiating the force expressions, but that's the easy part.

$\langle$ Functions for calculating forces and stiffnesses 8 $\rangle \equiv$

$\langle$ Parallel magnets force calculation 9 $\rangle$
$\langle$ Parallel magnets stiffness calculation 10 $\rangle$
$\langle$ Orthogonal magnets force calculation 11 $\rangle$
$\langle$ Orthogonal magnets stiffness calculation 12 $\rangle$

This code is used in section 2.

9. The expressions here follow directly from Akoun and Yonnet [1].

Inputs:	(a, b, c)	the half dimensions of the fixed magnet
	(A, B, C)	the half dimensions of the floating magnet
	(dx, dy, dz)	distance between magnet centres
	$(J, J2)$	magnetisations of the magnet(s) in the z-direction
Outputs:	(Fx, Fy, Fz)	Forces of the second magnet

⟨Parallel magnets force calculation 9⟩ ≡

```
function [Fx Fy Fz] = forces_parallel(a, b, c, A, B, C, dx, dy, dz, J, J2)
    % You probably want to call
    % warning off MATLAB:divideByZero
    % warning off MATLAB:log:logOfZero
  if nargin < 11
    J2 = J;
  elseif nargin < 10
    error('Wrong number of input arguments.')
  end
  [index_h, index_j, index_k, index_l, index_p, index_q] = ndgrid([0 1]);
  index_sum = (-1) .^ (index_h + index_j + index_k + index_l + index_p +
    index_q);
    % (Using this method is actually LESS efficient than using six for
    % loops for h..q over [0 1], but it looks a bit nicer, huh?)
  u = dx + A*(-1) .^ index_j - a*(-1) .^ index_h;
  v = dy + B*(-1) .^ index_l - b*(-1) .^ index_k;
  w = dz + C*(-1) .^ index_q - c*(-1) .^ index_p;
  r = sqrt(u .^ 2 + v .^ 2 + w .^ 2);
  f_x = ...
  +0.5*(v .^ 2 - w .^ 2) .* log(r - u) ...
  +u .* v .* log(r - v) ...
  +v .* w .* atan(u .* v ./ r ./ w) ...
  +0.5*r .* u;
  f_y = ...
  +0.5*(u .^ 2 - w .^ 2) .* log(r - v) ...
  +u .* v .* log(r - u) ...
  +u .* w .* atan(u .* v ./ r ./ w) ...
  +0.5*r .* v;
  f_z = ...
  -u .* w .* log(r - u) ...
  -v .* w .* log(r - v) ...
  +u .* v .* atan(u .* v ./ r ./ w) ...
  -r .* w;
```

$fx = index\_sum \;.\!* \; f\_x;$
$fy = index\_sum \;.\!* \; f\_y;$
$fz = index\_sum \;.\!* \; f\_z;$

$magconst = J{*}J2/(4{*}\pi{*}(4{*}\pi{*}1 \cdot 10^{-7}));$
$Fx = magconst{*}\mathrm{sum}(fx(:));$
$Fy = magconst{*}\mathrm{sum}(fy(:));$
$Fz = magconst{*}\mathrm{sum}(fz(:));$

end

This code is used in section 8.

10. And these are the stiffnesses.

Inputs:	(a, b, c)	the half dimensions of the fixed magnet
	(A, B, C)	the half dimensions of the floating magnet
	(dx, dy, dz)	distance between magnet centres
	$(J, J2)$	magnetisations of the magnet(s) in the z-direction
Outputs:	(Kx, Ky, Kz)	Stiffnesses of the 2nd magnet

⟨ Parallel magnets stiffness calculation 10 ⟩ ≡

```
function [Kx Ky Kz] = stiffness_parallel(a, b, c, A, B, C, dx, dy, dz, J, J2)
      % You probably want to call
      % warning off MATLAB:divideByZero
      % warning off MATLAB:log:logOfZero
  if nargin < 11
    J2 = J;
  elseif nargin < 10
    error('Wrong␣number␣of␣input␣arguments.')
  end
  [index_h, index_j, index_k, index_l, index_p, index_q] = ndgrid([0 1]);
  index_sum = (−1) .^ (index_h + index_j + index_k + index_l + index_p +
      index_q);
      % Using this method is actually less efficient than using six for
      % loops for h..q over [0 1]. To be addressed.
  u = dx + A∗(−1) .^ index_j − a∗(−1) .^ index_h;
  v = dy + B∗(−1) .^ index_l − b∗(−1) .^ index_k;
  w = dz + C∗(−1) .^ index_q − c∗(−1) .^ index_p;
  r = sqrt(u .^ 2 + v .^ 2 + w .^ 2);
  k_x = ...
  −r ...
  −(u .^ 2.∗v) ./ (u .^ 2 + w .^ 2) ...
  −v .∗ log(r − v);
  k_y = ...
  −r ...
  −(v .^ 2.∗u) ./ (v .^ 2 + w .^ 2) ...
  −u .∗ log(r − u);
  k_z = −k_x − k_y;
  kx = index_sum .∗ k_x;
  ky = index_sum .∗ k_y;
  kz = index_sum .∗ k_z;
  magconst = J∗J2/(4∗π∗(4∗π∗1 · 10^−7));
  Kx = magconst∗sum(kx(:));
  Ky = magconst∗sum(ky(:));
  Kz = magconst∗sum(kz(:));
```

end

This code is used in section 8.

11. Orthogonal magnets forces.

⟨ Orthogonal magnets force calculation 11 ⟩ ≡

This code is used in section 8.

12. Orthogonal magnets stiffnesses.

⟨ Orthogonal magnets stiffness calculation 12 ⟩ ≡

This code is used in section 8.

Index of `magnetforces`

atan : 9
a_1 : 3
a_2 : 3
b_1 : 3
b_2 : 3
cos : 4
c_1 : 3
c_2 : 3
dim : 3
dx : 3, 9, 10
dy : 3, 9, 10
dz : 3, 9, 10
error : 9, 10
f_x : 9
f_y : 9
f_z : 9
forces : 2
forces_parallel : 9
fx : 9
Fx : 9
fy : 9
Fy : 9
fz : 9
Fz : 9
index_h : 9, 10
index_j : 9, 10
index_k : 9, 10
index_l : 9, 10
index_p : 9, 10
index_q : 9, 10
index_sum : 9, 10
J1p : 3, 4
J1r : 3, 4
J1t : 3, 4
J1x : 4
J1y : 4
J1z : 4
J2 : 9, 10
J2p : 3, 4
J2r : 3, 4
J2t : 3, 4
J2x : 4
J2y : 4
J2z : 4
k_x : 10
k_y : 10
k_z : 10
kx : 10
Kx : 10
ky : 10
Ky : 10
kz : 10
Kz : 10
log : 9, 10
magconst : 9, 10
magdir : 3
magn : 3
magnet_disp : 2, 3
magnet_fixed : 2, 3
magnet_float : 2, 3

magnetforces : 2
nargin : 9, 10
ndgrid : 9, 10
phi : 3
sin : 4
sqrt : 9, 10
stiffness_parallel : 10
sum : 9, 10
theta : 3
torques : 2

List of Refinements in `magnetforces`

⟨ `magnetforces.m` 2 ⟩
⟨ Calculate forces 6 ⟩ Used in section 2.
⟨ Decompose orthogonal superpositions 4 ⟩ Used in section 2.
⟨ Extract input variables 3 ⟩ Used in section 2.
⟨ Functions for calculating forces and stiffnesses 8 ⟩ Used in section 2.
⟨ Orthogonal magnets force calculation 11 ⟩ Used in section 8.
⟨ Orthogonal magnets stiffness calculation 12 ⟩ Used in section 8.
⟨ Parallel magnets force calculation 9 ⟩ Used in section 8.
⟨ Parallel magnets stiffness calculation 10 ⟩ Used in section 8.
⟨ Recombine results 7 ⟩ Used in section 2.
⟨ Transform appropriate coordinate systems 5 ⟩ Used in section 2.

References

[1] Gilles Akoun and Jean-Paul Yonnet. “3D analytical calculation of the forces exerted between two cuboidal magnets”. In: *IEEE Transactions on Magnetics* MAG-20.5 (Sept. 1984), pp. 1962–1964. DOI: `10.1109/TMAG.1984.1063554`.