```matlab
%{
mwave - A water wave and wave energy converter computation package
Copyright (C) 2014  Cameron McNatt

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

Contributors:
    C. McNatt
%}
classdef ConstraintMatCompUT < matlab.unittest.TestCase

    methods (Test)

        function test1(testCase)
            % one body, no hinge, global cg at cg 1
            cgs = [0, 0, 0];
            hin = [];

            P = ConstraintMatComp.HingedBodies(cgs, hin);

            Pex = diag([1 1 1 1 1 1]);

            for m = 1:6
                for n = 1:6
                    testCase.verifyEqual(P(m,n), Pex(m,n), 'AbsTol',
 1e-12);
                end
            end
        end

        function test2(testCase)
            % one body, no hinge, global cg somewhere else
            cgs = [0, 0, 0];
            hin = [];
            org = [1, 0, 0];

            P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
 org);

            % composite body moves by x = 1, y = 2, z = 3
            % and rolls 1, pitches 2, and yaws 0.
```

```
            s = [1 2 3 1 2 0]';

            % new origin doesn't change wrt roll, yaw is 0, so only
effect
            % is in pitch. Positive pitch of 2, inceases z position by
2*-(-1),
            % so z = 3 + 2*1, x and y are the same
            % anges of body 1 should all be the same
            qex = [1 2 5 1 2 0].';
            q = P.'*s;

            for m = 1:6
                testCase.verifyEqual(q(m), qex(m), 'AbsTol', 1e-12);
            end

            % composite body moves by x = 1, y = 2, z = 3
            % and rolls 1, pitches 0, and yaws 2.
            s = [1 2 3 1 0 2]';

            % new origin doesn't change wrt roll, pitch is 0, so only
effect
            % is in yaw. Positive yaw of 2, decreases y position by
2*1,
            % so y = 2 - 2*1, x and z are the same
            % anges of body 1 should all be the same
            qex = [1 0 3 1 0 2].';
            q = P.'*s;

            for m = 1:6
                testCase.verifyEqual(q(m), qex(m), 'AbsTol', 1e-12);
            end

            % try another location
            hin = [];
            org = [1, -3, -2];

            P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
org);

            % composite body moves by x = 1, y = 2, z = 3
            % and rolls 1, pitches 2, and yaws 3.
            s = [1 2 3 1 2 3]';

            % translation:  x = 1,      y = 2,      z = 3
            % roll:         x = +0,     y = -1*2,   z = +1*3
            % pitch:        x = +2*2,   y = +0,     z = +2*1
            % yaw:          x = -3*3,   y = -3*1,   z = +0
            % total:        x = -4,     y = -3,      z = 8
            % anges of body 1 should all be the same
            qex = [-4 -3 8 1 2 3].';
            q = P.'*s;

            for m = 1:6
                testCase.verifyEqual(q(m), qex(m), 'AbsTol', 1e-12);
```

```
        end
    end

    function test3(testCase)
        % two bodies, one hinge,
        % global cg at hinge
        cgs = [-1, 0, 2; 3, 0, -4];
        hin = [0, 0, 0];
        org = hin;

        P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
org);

        % composite body moves by x = 1, y = 2, z = 3
        % and rolls 1, pitches 2, yaws 3, and flexes 4
        s = [1 2 3 1 2 3 4]';

        qex = zeros(12, 1);
        % Body 1
        % translation:  x = 1,      y = 2,      z = 3
        % roll:         x = +0,     y = -1*2,   z = -1*0
        % pitch:        x = +2*2,   y = +0,     z = +2*1
        % yaw:          x = -3*0,   y = -3*1,   z = +0
        % total:        x = 5,      y = -3,      z = 5
        % anges of body 1 should all be the same
        qex(1) = 5;
        qex(2) = -3;
        qex(3) = 5;
        qex(4) = 1;
        qex(5) = 2;
        qex(6) = 3;

        % Body 2
        % pitch angle is: 4 + 2
        % translation:  x = 1,          y = 2,      z = 3
        % roll:         x = +0,         y = 1*4,    z = -1*0
        % pitch:        x = -(4+2)*4,   y = +0,     z = -(4+2)*3
        % yaw:          x = -3*0,       y = 3*3,   z = +0
        % total:        x = -23,        y = 15,     z = -15
        % anges of body 2 should all be the same
        qex(7) = -23;
        qex(8) = 15;
        qex(9) = -15;
        qex(10) = 1;
        qex(11) = 6;
        qex(12) = 3;

        q = P.'*s;

        for m = 1:12
            testCase.verifyEqual(q(m), qex(m), 'AbsTol', 1e-12);
        end
    end
```

```
function test4(testCase)
    % two bodies, one hinge,
    % global cg at body 1
    cgs = [-1, 0, 2; 3, 0, -4];
    hin = [0, 0, 0];

    P = ConstraintMatComp.HingedBodies(cgs, hin);

    % composite body moves by x = 1, y = 2, z = 3
    % and rolls 1, pitches 2, yaws 3, and flexes 4
    s = [1 2 3 1 2 3 4]';

    qex = zeros(12, 1);
    % Body 1
    % origin is at cg 1
    % position and angles of body 1 should be the same
    qex(1) = 1;
    qex(2) = 2;
    qex(3) = 3;
    qex(4) = 1;
    qex(5) = 2;
    qex(6) = 3;

    % Body 2
    % pitch angle of 2 is: 4 + 2
    % translation:  x = 1,          y = 2,      z = 3
    % Body 1
    % roll:         x = 0,          y = 1*2,    z = 0
    % pitch:        x = -2*2,       y = 0,      z = -2*1
    % yaw:          x = 0,          y = 3*1,    z = 0
    % body 2
    % roll:         x = 0,          y = 1*4,    z = 0
    % pitch:        x = -(4+2)*4,   y = 0,      z = -(4+2)*3
    % yaw:          x = 0,          y = 3*3,    z = 0
    % total:        x = -27,        y = 20,     z = -17
    % anges of body 2 should all be the same
    qex(7) = -27;
    qex(8) = 20;
    qex(9) = -17;
    qex(10) = 1;
    qex(11) = 6;
    qex(12) = 3;

    q = P.'*s;

    for m = 1:12
        testCase.verifyEqual(q(m), qex(m), 'AbsTol', 1e-12);
    end
end

 function test5(testCase)
    % three bodies, two hinges,
    % global cg at cg 1
    cgs = [-1, 0, 2; 3, 0, -4; 5, 0, 6];
```

```
hins = [0, 0, 0; 4, 0, 0];

P = ConstraintMatComp.HingedBodies(cgs, hins);

% composite body moves by x = 1, y = 2, z = 3
% and rolls 1, pitches 2, yaws 3, and flexes1 4, flexes2
-4
s = [1 2 3 1 2 3 4 -4]';

qex = zeros(18, 1);
% Body 1
% origin is at cg 1
% position and angles of body 1 should be the same
qex(1) = 1;
qex(2) = 2;
qex(3) = 3;
qex(4) = 1;
qex(5) = 2;
qex(6) = 3;

% Body 2
% pitch angle of 2 is: 4 + 2
% translation:  x = 1,          y = 2,      z = 3
% Body 1
% roll:         x = 0,          y = 1*2,    z = 0
% pitch:        x = -2*2,       y = 0,      z = -2*1
% yaw:          x = 0,          y = 3*1,    z = 0
% body 2
% roll:         x = 0,          y = 1*4,    z = 0
% pitch:        x = -(4+2)*4,   y = 0,      z = -(4+2)*3
% yaw:          x = 0,          y = 3*3,    z = 0
% total:        x = -27,        y = 20,     z = -17
qex(7) = -27;
qex(8) = 20;
qex(9) = -17;
qex(10) = 1;
qex(11) = 6;
qex(12) = 3;

% Body 3
% pitch angle of 3 is: (2 + 4) - 4 = 2
% start at position of body 2
% translation:  x = -27,        y = 20,     z = -17
% Body 2
% roll:         x = 0,          y = -1*4,   z = 0
% pitch:        x = (4+2)*4,    y = 0,      z = -(4+2)*1
% yaw:          x = 0,          y = 3*1,    z = 0
% Body 3
% roll:         x = 0,          y = -1*6,   z = 0
% pitch:        x = 2*6,        y = 0,      z = -2*1
% yaw:          x = 0,          y = 3*1,    z = 0
% total:        x = 9           y = 16,     z = -25
qex(13) = 9;
qex(14) = 16;
```

```
            qex(15) = -25;
            qex(16) = 1;
            qex(17) = 2;
            qex(18) = 3;

            q = P.'*s;

            for m = 1:18
                testCase.verifyEqual(q(m), qex(m), 'AbsTol', 1e-12);
            end
         end

        function test6(testCase)
            % check the computation of the mass matrix of composite
hinge
            % body

            % symmetric blocks, origin at hinge

            %
            %     _______  _______
            %    |   1   |||   2   |
            %    |_______|||_______|
            %
            %

            len1 = 6;
            wid1 = 1;
            hei1 = 2;

            len2 = len1;
            wid2 = wid1;
            hei2 = hei1;

            M1 = ConstraintMatCompUT.massBlock(len1, wid1, hei1);
            M2 = ConstraintMatCompUT.massBlock(len2, wid2, hei2);

            Mq = zeros(12,12);
            Mq(1:6, 1:6) = M1;
            Mq(7:12, 7:12) = M2;

            cg1 = [-len1/2, 0, 0];
            cg2 = [len2/2, 0, 0];

            cgs = [cg1; cg2];
            hin = [0 0 0];
            org = hin;

            P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
org);

            Ms = P*Mq*P.';

            % single large block
```

```
            Mex66 = ConstraintMatCompUT.massBlock(len1 + len2, wid1,
hei1);

            Mex = zeros(7, 7);
            Mex(1:6, 1:6) = Mex66;
            % parallel axis theorem to get flex inertia (i.e. flex
intertia
            % is pitch inertia of second body about hinge
            m2 = M2(1,1);
            r2 = len2/2;
            Ipp2 = M2(5,5);
            Mex(7,7) = Ipp2 + m2*r2^2;
            % heave-flex is the moment created in flex due to a
positve
            % motion in heave
            Mex(3,7) = -m2*r2;
            Mex(7,3) = Mex(3,7);

            % pitch-flex is the moment created in flex due to a
postive
            % pitch
            Mex(5,7) = Mex(7,7);
            Mex(7,5) = Mex(7,7);

            for m = 1:7
                for n = 1:7
                    testCase.verifyEqual(Ms(m,n),
Mex(m,n), 'AbsTol', ...
                        1e-12);
                end
            end
        end

        function test7(testCase)
            % check the computation of the mass matrix of composite
hinge
            % body

            % origin at hinge

            %              _________
            %     _______ |         |
            %    |   1   ||    2    |
            %    |_______||         |
            %             |_________|
            %

            len1 = 6;
            wid1 = 1;
            hei1 = 2;

            len2 = 9;
            wid2 = wid1;
            hei2 = 4;
```

```
M1 = ConstraintMatCompUT.massBlock(len1, wid1, hei1);
M2 = ConstraintMatCompUT.massBlock(len2, wid2, hei2);

Mq = zeros(12,12);
Mq(1:6, 1:6) = M1;
Mq(7:12, 7:12) = M2;

cg1 = [-len1/2, 0, 0];
cg2 = [len2/2, 0, 0];

cgs = [cg1; cg2];
hin = [0 0 0];
org = hin;

P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
org);

Ms = P*Mq*P.';

Mex = zeros(7, 7);
m1 = M1(1,1);
m2 = M2(1,1);
r1 = len1/2;
r2 = len2/2;

% mass
m = m1 + m2;
Mex(1,1) = m;
Mex(2,2) = m;
Mex(3,3) = m;
% Roll - both on same roll axis
Mex(4,4) = M1(4,4) + M2(4,4);
% Pitch
Mex(5,5) = M1(5,5) + m1*r1^2 + M2(5,5) + m2*r2^2;
% Yaw
Mex(6,6) = M1(6,6) + m1*r1^2 + M2(6,6) + m2*r2^2;

% parallel axis theorem to get flex inertia (i.e. flex
intertia
% is pitch inertia of second body about hinge
Mex(7,7) = M2(5,5) + m2*r2^2;

% heave-pitch is the moment created in pitch due to a
positive
% heave about origin (i.e. hinge)
cg0 = (m1*cgs(1,:) + m2*cgs(2,:))./m;
r0 = org - cg0;
Mex(3,5) = m*r0(1);
Mex(5,3) = Mex(3,5);

% surge-yaw is just like heave-pitch, but negative
(rotations
% of pitch and yaw are opposite)
```

```
            Mex(2,6) = -m*r0(1);
            Mex(6,2) = Mex(2,6);

            % heave-flex is the moment created in flex due to a
positve
            % motion in heave
            Mex(3,7) = -m2*r2;
            Mex(7,3) = Mex(3,7);

            % pitch-flex is the moment created in flex due to a
postive
            % pitch
            Mex(5,7) = Mex(7,7);
            Mex(7,5) = Mex(7,7);

            for m = 1:7
                for n = 1:7
                    testCase.verifyEqual(Ms(m,n),
Mex(m,n), 'AbsTol', ...
                        1e-12);
                end
            end
        end

        function test8(testCase)
            % check the computation of the mass matrix of composite
hinge
            % body

            % origin at global cg

            %              _________
            %     _______ |         |
            %    |   1   ||    2    |
            %    |_______||         |
            %             |_________|
            %

            len1 = 6;
            wid1 = 1;
            hei1 = 2;

            len2 = 9;
            wid2 = wid1;
            hei2 = 4;

            M1 = ConstraintMatCompUT.massBlock(len1, wid1, hei1);
            M2 = ConstraintMatCompUT.massBlock(len2, wid2, hei2);

            Mq = zeros(12,12);
            Mq(1:6, 1:6) = M1;
            Mq(7:12, 7:12) = M2;

            cg1 = [-len1/2, 0, 0];
```

```
            cg2 = [len2/2, 0, 0];

            cgs = [cg1; cg2];
            hin = [0 0 0];

            m1 = M1(1,1);
            m2 = M2(1,1);
            m = m1 + m2;

            cg0 = (m1*cgs(1,:) + m2*cgs(2,:))./m;

            org = cg0;

            P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
org);

            Ms = P*Mq*P.';

            Mex = zeros(7, 7);

            % mass
            Mex(1,1) = m;
            Mex(2,2) = m;
            Mex(3,3) = m;
            % Roll - both on same roll axis
            Mex(4,4) = M1(4,4) + M2(4,4);

            r1 = cg1 - cg0;
            r1 = r1(1);
            r2 = cg2 - cg0;
            r2 = r2(1);

            % Pitch
            Mex(5,5) = M1(5,5) + m1*r1^2 + M2(5,5) + m2*r2^2;
            % Yaw
            Mex(6,6) = M1(6,6) + m1*r1^2 + M2(6,6) + m2*r2^2;

            % parallel axis theorem to get flex inertia (i.e. flex
intertia
            % is pitch inertia of second body about hinge
            rh2 = len2/2;
            Mex(7,7) = M2(5,5) + m2*rh2^2;

            % heave-flex is the moment created in flex due to a
positve
            % motion in heave
            Mex(3,7) = -m2*rh2;
            Mex(7,3) = Mex(3,7);

            % pitch-flex is the moment created in flex due to a
postive
            % pitch
            dr = cg2 - hin;
            dr = dr(1);
```

```
            Mex(5,7) = m2*dr*r2 + M2(5,5);
            Mex(7,5) = Mex(5,7);

            for m = 1:7
                for n = 1:7
                    testCase.verifyEqual(Ms(m,n),
Mex(m,n), 'AbsTol', ...
                        1e-12);
                end
            end
        end

        function test9(testCase)
            % check the computation of the mass matrix of composite
hinge
            % body

            % origin at global cg, including dz in the hinge

            %     _______   _________
            %    |       || |         |
            %    |   1   || |    2    |
            %    |_______|| |         |
            %             | |_________|
            %

            len1 = 6;
            wid1 = 1;
            hei1 = 2;

            len2 = 9;
            wid2 = wid1;
            hei2 = 4;

            M1 = ConstraintMatCompUT.massBlock(len1, wid1, hei1);
            M2 = ConstraintMatCompUT.massBlock(len2, wid2, hei2);

            Mq = zeros(12,12);
            Mq(1:6, 1:6) = M1;
            Mq(7:12, 7:12) = M2;

            cg1 = [-len1/2, 0, 2];
            cg2 = [len2/2, 0, 0];

            cgs = [cg1; cg2];
            hin = [0 0 1];

            m1 = M1(1,1);
            m2 = M2(1,1);
            m = m1 + m2;

            cg0 = (m1*cgs(1,:) + m2*cgs(2,:))./m;

            org = cg0;
```

```
            P = ConstraintMatComp.HingedBodies(cgs, hin, 'Origin',
org);

            Ms = P*Mq*P.';

            Mex = zeros(7, 7);

            % mass
            Mex(1,1) = m;
            Mex(2,2) = m;
            Mex(3,3) = m;

            r1 = cg1 - cg0;
            r2 = cg2 - cg0;

            % Roll
            Mex(4,4) = M1(4,4) + M2(4,4) + m1*r1(3)^2 + m2*r2(3)^2;

            % Pitch
            Mex(5,5) = M1(5,5) + m1*sum(r1.^2) + M2(5,5) +
m2*sum(r2.^2);
            % Yaw
            Mex(6,6) = M1(6,6) + m1*r1(1)^2 + M2(6,6) + m2*r2(1)^2;

            % There is also a roll-yaw coupling, because of the
difference
            % in the z-pos of the CGs of each body, which creates new
            % roll and yaw pricipal axes.
            Mex(4,6) = Ms(4,6);
            Mex(6,4) = Mex(4,6);

            % parallel axis theorem to get flex inertia (i.e. flex
intertia
            % is pitch inertia of second body about hinge
            rh2 = cg2 - hin;
            Mex(7,7) = M2(5,5) + m2*(rh2*rh2');

            % heave-flex is the moment created in flex due to a
positve
            % motion in heave
            Mex(1,7) = m2*rh2(3);
            Mex(7,1) = Mex(1,7);
            Mex(3,7) = -m2*rh2(1);
            Mex(7,3) = Mex(3,7);

            % pitch-flex is the moment created in flex due to a
postive
            % pitch
            Mex(5,7) = m2*r2*rh2' + M2(5,5);
            Mex(7,5) = Mex(5,7);

            for m = 1:7
                for n = 1:7
```

```
                    testCase.verifyEqual(Ms(m,n),
 Mex(m,n), 'AbsTol', ...
                        1e-12);
                end
            end
        end
    end

    methods (Static, Access = private)

        function [M] = massBlock(len, wid, hei)

            m = len*wid*hei;
            Ixx = m/12*(wid^2 + hei^2);
            Iyy = m/12*(len^2 + hei^2);
            Izz = m/12*(len^2 + wid^2);

            M = zeros(6, 6);
            M(1,1) = m;
            M(2,2) = m;
            M(3,3) = m;
            M(4,4) = Ixx;
            M(5,5) = Iyy;
            M(6,6) = Izz;
        end
    end
end
```

```
ans =

  ConstraintMatCompUT with no properties.
```

Published with MATLAB® R2015b