```
%% initialize
close all
clear all
format shorte
clc
pause( 1e-2 )

addpath('./tmtdyn');
addpath('./hll');
addpath('./eom');


%% parameters
l_xz2 = 73e-3 / 5 ;
l_y2 = 0.8e-3 ;
m_2 = 36e-3 / 15 ;

l_x1 = 76e-3 / 2 ;
l_z1 = 30e-3 ;
l_com1 = 135e-3 ;

g = 9.81 ; % gravity

% EBA
E = 1.5e3 ;
mu_pow = 1 ; % < 1: rate thining, > 1: rate thikening
mu_v = 5e1 ;
mu_u = 5e1;
mu_link = 7e-4 ;

phi = - 85 * pi/ 180 ;

% variables & values
vars = 'E_sym mu_pow_sym mu_v_sym mu_u_sym phi_sym mu_link_sym';
var_vals = [ E mu_pow mu_v mu_u phi mu_link ] ;


%% preprocess
G = E_sym / 3 ;
I_2 = 1 / 12 * m_2 * ...
    diag( [ l_y2^2+l_xz2^2 l_xz2^2+l_xz2^2 l_xz2^2+l_y2^2 ] ) ;
J_2 = 1 / 12 * [ l_xz2*l_y2^3 l_xz2^3*l_y2 l_xz2*l_y2^3+l_xz2^3*l_y2 ] ;
a_xy2 = l_xz2 * l_y2 ;
K_v = diag( [ G G E_sym ] ) * a_xy2 / l_xz2 ;
K_u = diag( [ E_sym E_sym G ] ) * diag( J_2 ) / l_xz2 ;


%% robot
tmtdyn()...
    .simulation()... % simulation
        .vars(vars)...
        .var_vals(var_vals)...
        .eom_derive()...
            .mex()...
            .optimize()...
        .end()...
        .analysis()...
            .dynamic_sim(2)...
        .end()...
        .post_process()...
            .animate()...
            .user_code()...
        .end()...
    .end()...
    .world()... % world
        .g([ 0 0 -g ])...
    .end() ...
    .robot('fabric_pendulum')... % robot
```

replace: .var('E_sym', E). var('mu_pow', mu_pow) ...

-> does this actually need to feed in the user code?
-> could feed in an object ...

is there even anything else? if not, why not just say g([0 0 -g])?

```
        .body('arm')...
            .mass(m_2)...
            .l_com([0 0 -l_com1])...
            .length(2*l_com)...
        .end()...
        .joint('arm_hinge')...
            .second_body(1)...
            .tr()...
                .rot_y()...
            .end()...
            .dof()...
                .init(phi_sym)...
                .damper()...
                    .visc(mu_link_sym)...
                .end()...
            .end()...
    .end()...
    .mesh('fabric')...
        .file_name('exp/exp2.iges')...
        .tol(1e-3)...
        .tr()...
            .rot_y(phi_sym)...
        .end()...
        .tr()...
            .trans_z(-l_z1)...
        .end()...
        .body('fabric')...
            .mass(m_2)...
            .inertia(I_2)...
        .end()...
        .joint('fabric_points')...
            .tr()...
                .trans([inf inf inf])...
                .rot_type('non_unit_quat')...
                .rot([inf inf inf inf])...
            .end()...
        .joint('fabric_links')...
            .spring()...
                .coeff([ diag( K_v ).' diag( K_u ).' ])...
                .init(nan)...
            .end()...
            .damper()...
                .visc([ mu_v_sym*ones(1,3) mu_u_sym*ones(1,3) ])...
                .power(mu_pow_sym)...
            .end()...
        .end()...
    .end()...
    .joint('clip_constraint_1')...
        .first_body(1)...
        .second_body(16)...
        .tr()...
            .trans([ l_x1 0 -l_z1 ])...
        .end()...
        .fixed([1 1 1])...
    .end()...
    .joint('clip_constraint_2')...
        .first_body(1)...
        .second_body(14)...
        .tr()...
            .trans([ -l_x1 0 -l_z1 ])...
        .end()...
        .fixed([1 1 1])...
    .end()...
    .run() ;
```

is l_com an easily understood keyword?

is this what's called 'tip' in THT Dyn?

this is really the fixity to the base ⇒ rename it to base_joint?

could make target body a param of base_joint (& call ...

-> what happens for mixed sequences of rot + trans? ⇒ can we use tr to indices?

can we make the link to tr more explicit? ⇒ eg. declare each dof directly with its tr bit?

file (perhaps use from_file(...)?)

would suggest spelling out: with_tolerance (...)

-> can't we combine these into a single tr() block and calculate the tr elements needed?

-> are we allowed l_com and length/tip here?

-> use specialised keyword to clarify the special function of this joint

-> what does this mean? can we capture in a keyword?

-> do we have to specify this or can we derive from other data?

-> what does this mean? can we capture in a keyword?

-> use specialised keyword to clarify the special function of this joint

-> bespoke keyword instead?

use constraint(...) instead

from

to

-> do we need the parameter? could we have different types of constraint() instead of the fixed(...) call?

use constraint(...) instead

see above ...