Table of Contents


```
clear
close all
```

Intruder geometry

```
folder = 'robot';                                % cylinder, simple, robottip
object = 'T2';                                   % name of stl
triangle_size_calculation = 'rough';             % 'Fine', 'Normal', 'Rough',
 'VeryRough'
triangle_size_visualization = 'rough';           % 'Fine', 'Normal', 'Rough',
 'VeryRough'
rotation_angle = 0;                              % rotate intruder around x-
axis
```

Physical Properties

```
rho_c = 1617;                                    % bulk density of the sand in
 kg/m³
mu_int = 1.07;                                   % internal friction
 coefficient of the sand
mu_surf = 0.1806;                                % intruder-surface interaction
 coefficient
gravity = 9.81;                                  % gravity in m/s²
xi_n = rho_c * gravity * (894*mu_int^3 - 386*mu_int^2 + 89*mu_int); %
 initially in N/m³
```

Movement parameters

```
rotation = true;                                 % true or false
linear_velocity = 0.012;                         % linear velocity in m/s
direction_angle_xz = -90 * pi / 180;             % angle between direction and
 x-z-axis
direction_angle_y = -90 * pi / 180;              % angle between direction and
 y-axis
```

```
angular_velocity = [0, 0, -2*pi];               % angular velocity in rad/s
direction_vector = [round(cos(direction_angle_xz), 15) ...
    round(cos(direction_angle_y), 15) round(sin(direction_angle_xz), 15)];
```

Depth parameters

```
start_depth = 0;
end_depth = 0.180;
step_size = 0.01;
```

Plot selection

```
show_geometry = 1;
show_movement = 1;

show_f_quiver = 1;
show_alpha = 1;

show_f_scatter = 1;
show_f_scatterxyz = 0;

show_results = 1;

saveFigures = 0;
```

Miscellaneous

```
unit_test = false;          % Unit testing option
threshold = 1.0e-12;        % Eliminate floating points errors
```

Read STL file

```
TRG = stlread(strcat('./', folder, '/Models/', object, ...
    triangle_size_calculation, '.stl'));                              % mesh for
 calculation
TRG_visual = stlread(strcat('./', folder, '/Models/', object, ...
    triangle_size_visualization, '.stl'));                            % mesh for
 plots

TRG = rotateTriangulationX(TRG, rotation_angle);                      % rotate
 TRG object
TRG_visual = rotateTriangulationX(TRG_visual, rotation_angle);       % rotate
 Visual
```

Loop RFT functions over depths

```
num_steps = (end_depth - start_depth)./step_size;
depths = start_depth:step_size:end_depth;
results = zeros(7, num_steps);
```

```
step = 1;

for depth = start_depth:step_size:end_depth

    TRG = moveTriangulationZ(TRG, depth);                          % align
 align bottom with depth
    TRG_visual = moveTriangulationZ(TRG_visual, depth);            % align
 bottom with depth (visual)

    % step 1: process STL data
    [points, normals, areas, depth_list] = getStlData(TRG, TRG.Points',
 TRG.ConnectivityList');

    % step 2: define movement vector
    [movement, movement_normalized] = calcVelocity(points, direction_vector,
 linear_velocity, rotation, angular_velocity, threshold);

    % step 3: RFT conditions
    [include, normals_include, movement_normalized_include, movement_include,
 areas_include, points_include, depth_list_include] ...
        = checkConditions(points, normals, areas, movement,
 movement_normalized, depth_list, unit_test, threshold);

    % step 4: find local coordinate frame
    [z_local, r_local, theta_local] = findLocalFrame(normals_include,
 movement_normalized_include, movement_include, include, unit_test);

    % step 5: RFT characteristic angles
    [beta, gamma, psi] = findAngles(normals_include,
 movement_normalized_include, r_local, z_local, theta_local);

    % step 6: empirically determined force components
    [f1, f2, f3] = findFit(gamma, beta, psi, z_local,
 movement_normalized_include, normals_include, threshold, depth_list_include,
 include, unit_test);

    % step 7: find generic form of alpha, split into normal and tangential
    % components and add soil + interface properties for to get alpha
    [alpha_gen, alpha_gen_n, alpha_gen_t, alpha] = findAlpha(normals_include,
 movement_normalized_include, beta, gamma, psi, r_local, theta_local, z_local,
 f1, f2, f3, mu_surf, xi_n);

    % step 8: get forces
    [forces, pressures, force_x, force_y, force_z, resultant] =
 getForces(depth_list_include, areas_include, alpha);

    % step 9: get torques
    [T, torque_x, torque_y, torque_z] = getTorques(points_include,
 depth_list_include, forces, unit_test, include);


    results(:,step) = [depth; force_x; force_y; force_z; torque_x; torque_y;
 torque_z];
```

```
    disp(['Depth processed: ', num2str(depth), 'm']);
    step = step + 1;

end
```

```
Depth processed: 0m
Depth processed: 0.01m
Depth processed: 0.02m
Depth processed: 0.03m
Depth processed: 0.04m
Depth processed: 0.05m
Depth processed: 0.06m
Depth processed: 0.07m
Depth processed: 0.08m
Depth processed: 0.09m
Depth processed: 0.1m
Depth processed: 0.11m
Depth processed: 0.12m
Depth processed: 0.13m
Depth processed: 0.14m
Depth processed: 0.15m
Depth processed: 0.16m
Depth processed: 0.17m
Depth processed: 0.18m
```

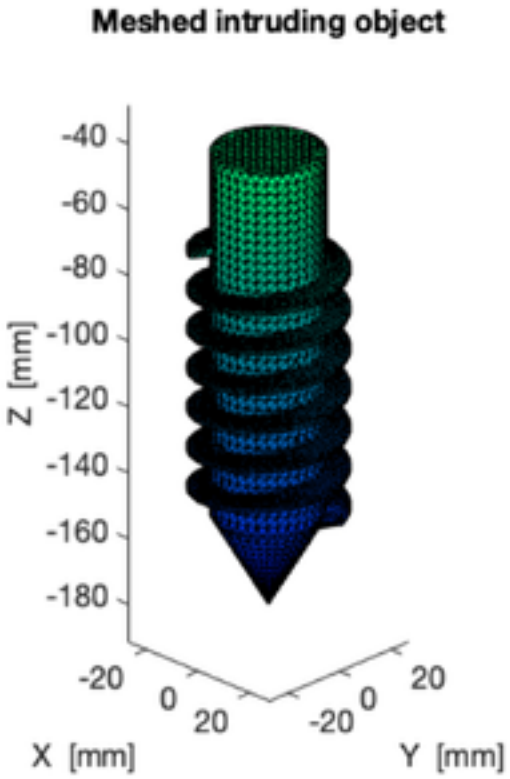
Intruder size readings

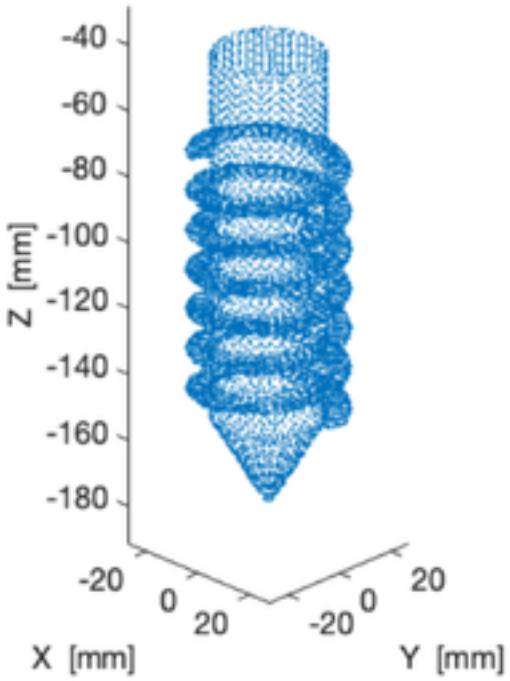
```
intruder_width_x = abs( max(points(:,1)) - min(points(:,1)) );
intruder_width_y = abs( max(points(:,2)) - min(points(:,2)) );
intruder_height = abs( max(points(:,3)) - min(points(:,3)) );
```

Call plotting functions

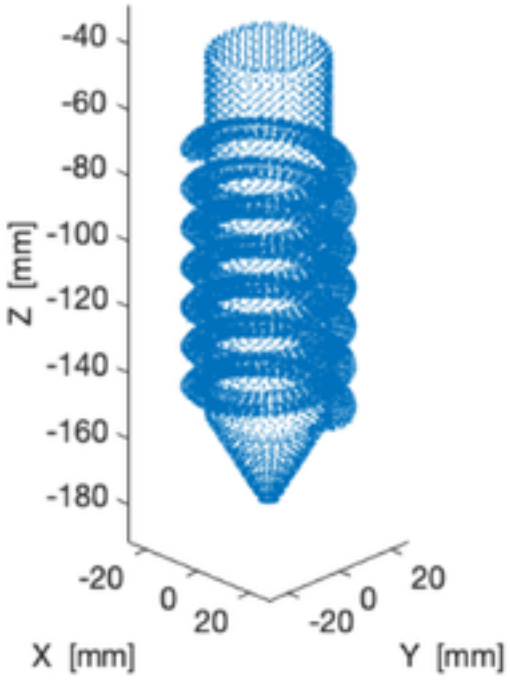
```
x_range = [min(points(:,1))-intruder_width_x/10
 max(points(:,1))+intruder_width_x/10];
y_range = [min(points(:,2))-intruder_width_y/10
 max(points(:,2))+intruder_width_y/10];
z_range = [min(points(:,3))-intruder_height/10
 max(points(:,3))+intruder_height/10];

functionHandles = {
    @plotGeometry,    show_geometry;
    @plotMovement,    show_movement;
    @plotFQuiver,     show_f_quiver;
    @plotAlpha,       show_alpha;
    @plotFScatter,    show_f_scatter;
    @plotFScatterxyz, show_f_scatterxyz;
    @plotResults,     show_results
};

for i = 1:size(functionHandles, 1)
    if functionHandles{i, 2}
        functionHandles{i, 1}();
```

```
    end
end
```

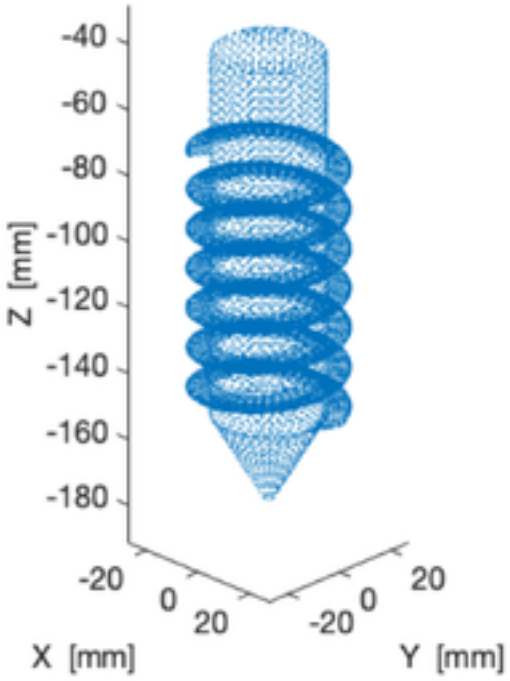

Meshed intruding object
-40
-60
-80
-100
-120
-140
-160
-180
Z [mm]
-20
0
20
X [mm]
-20
0
20
Y [mm]

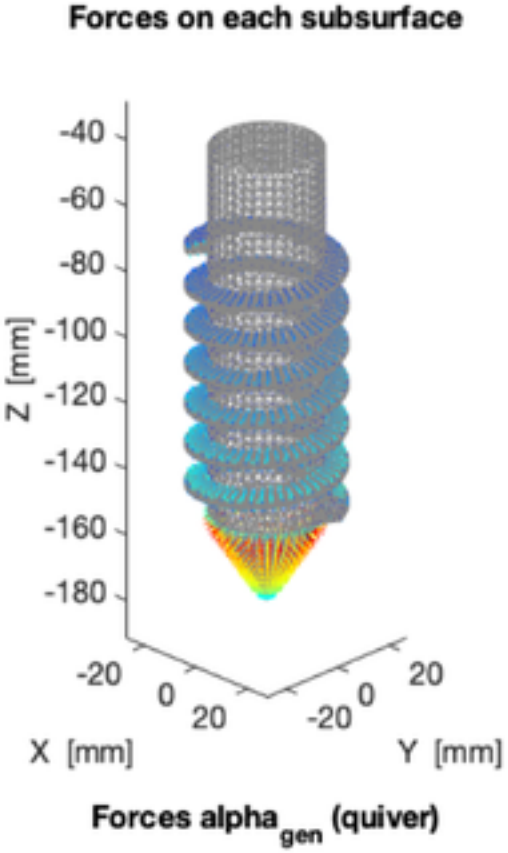

Centerpoints of the subsurfaces
-40
-60
-80
-100
-120
-140
-160
-180
Z [mm]
-20
0
20
X [mm]
-20
0
20
Y [mm]

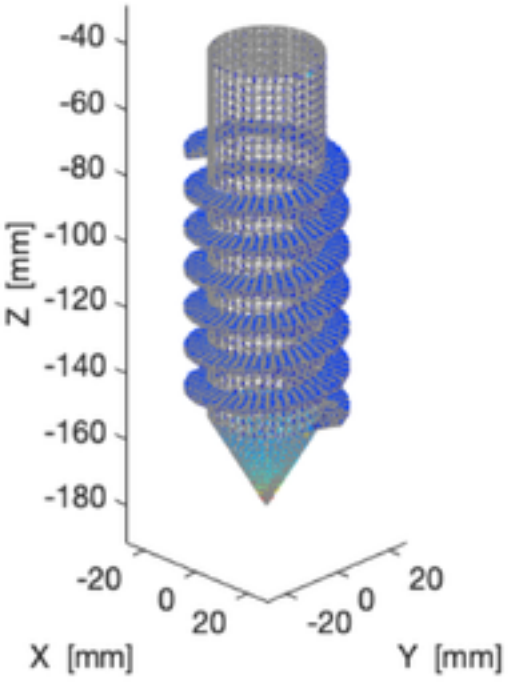
Normals of the subsurfaces

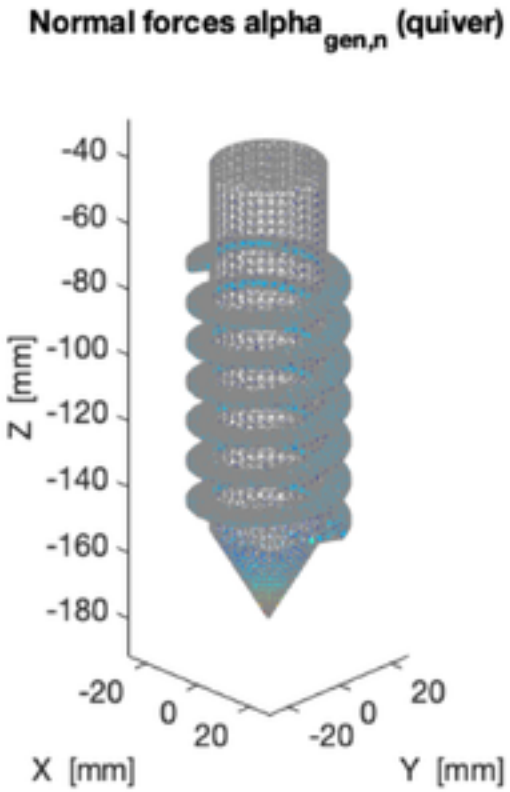


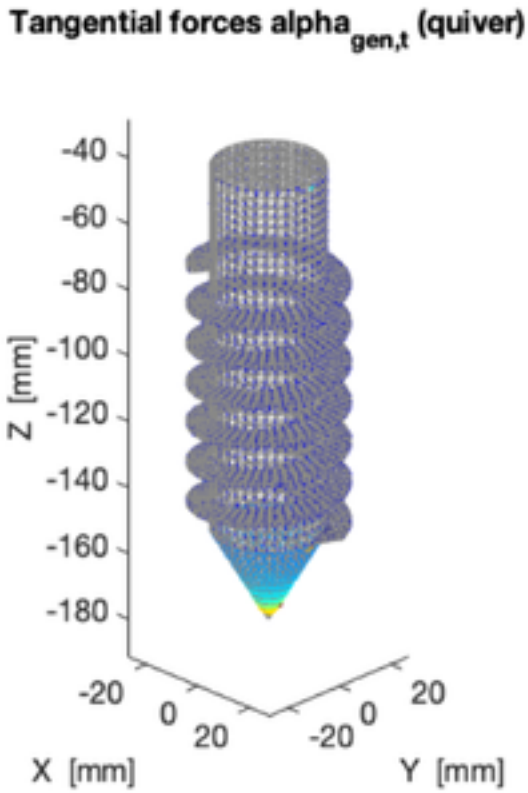
Direction vectors of rotation and translation

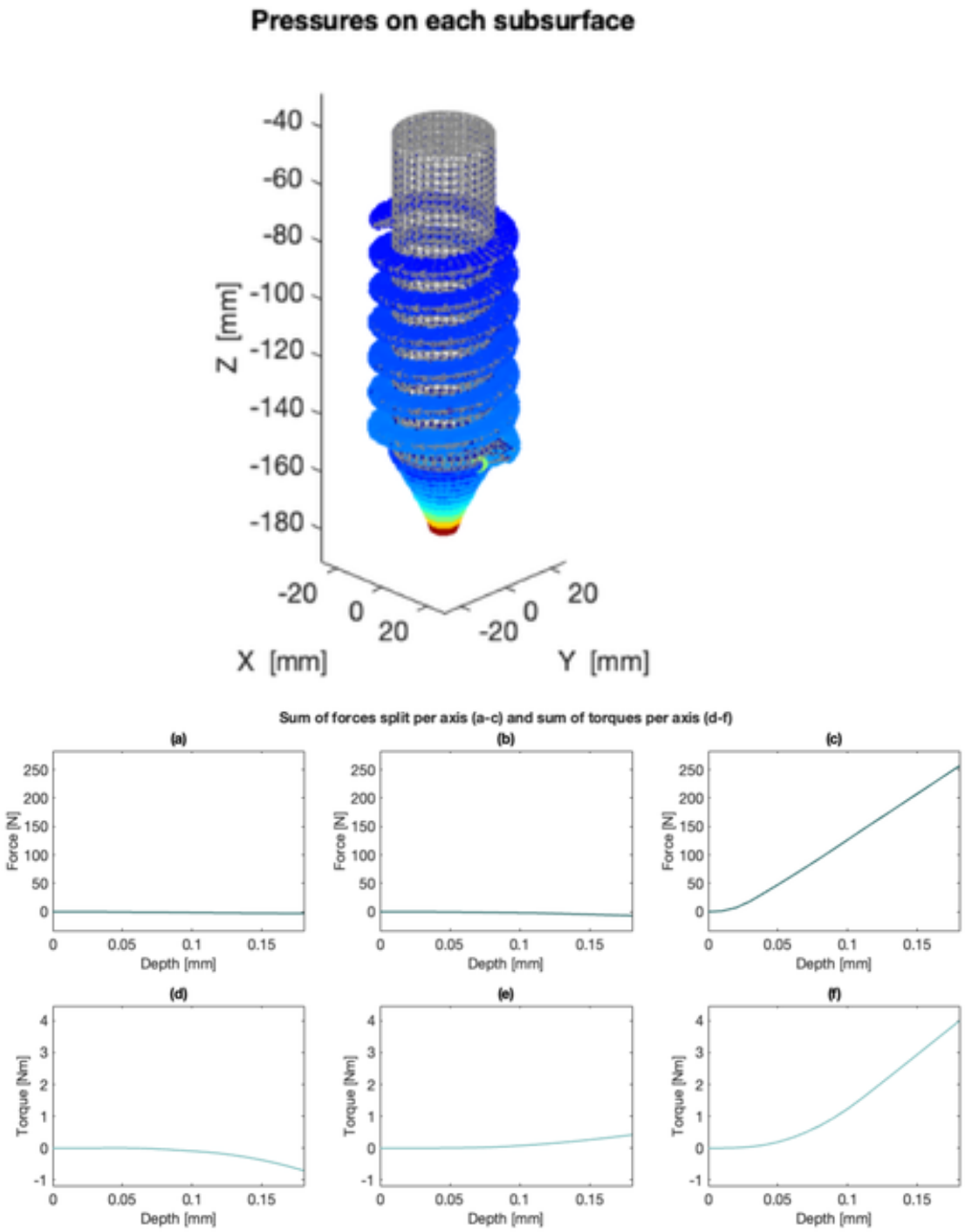



Forces on each subsurface
-40
-60
-80
-100
-120
-140
-160
-180
Z [mm]
-20
0
20
X [mm]
-20
0
20
Y [mm]


Forces alpha gen (quiver)
-40
-60
-80
-100
-120
-140
-160
-180
Z [mm]
-20
0
20
X [mm]
-20
0
20
Y [mm]


Normal forces alpha gen,n (quiver)
-40
-60
-80
-100
-120
-140
-160
-180
Z [mm]
-20
0
20
X [mm]
-20
0
20
Y [mm]


Tangential forces alpha gen,t (quiver)
-40
-60
-80
-100
-120
-140
-160
-180
Z [mm]
-20
0
20
X [mm]
-20
0
20
Y [mm]



Finish

```
varList = evalin('caller', 'who');
matches = regexp(varList, '^show_\w*', 'match');
matches = vertcat(matches{:});
clear(matches{:})
```

```
disp("Done!");
```

```
Done!
```

Published with MATLAB® R2023a