```
% Modify this code to calculate the joint torques
% Input Parameters
%   t - time
%   s - state of the robot
%   model - struct containing robot properties
%   ctrl - any user defined control parameters in student_setup.m
% Output
%   tau - 10x1 vector of joint torques
function tau = studentController(t, s, model, params)
```

Extract generalized coordinates and velocities

```
    q = s(1 : model.n);
    dq = s(model.n+1 : 2*model.n);
```

Not enough input arguments.

Error in studentController (line 12)
q = s(1 : model.n);

Contact Force Control (WITH OPTIMIZATION)

Obtain initial joint angles from initial state

```
    x0 = getInitialState(model);
    q0 = x0(1:model.n);
    dq0 = x0(model.n+1:2*model.n);

    [r_com, v_com] = computeComPosVel(q, dq, model);

    % Gains optimized for constant force

 kpx = 1000; % best so far: 1000 w/ 300 on kdx (can reject x forces up
 to 15)
    kdx = 300;

    kpy = 1000; % best so far: 1000 w/ 300 on kdy (can reject y forces
 up to 15)
    kdy = 300;

    kpz = 1000; % best so far: 1000 w/ 300 on kdz
    kdz = 300;

    kp = diag([kpx;kpy;kpz],0);
    kd = diag([kdx;kdy;kdz],0);

    Kr = 200*eye(3);
    Dr = 50*eye(3);

    % Mass of Cassie etc...
    m = model.M; g = [0;0;9.81];
```

```
% Coordinate transformation

R_des = rot_z(q0(6))*rot_y(q0(5))*rot_x(q0(4));
R_b = rot_z(q(6))*rot_y(q(5))*rot_x(q(4));

R_db = R_des.'*R_b;

Q = rotm2quat(R_db);
delta = Q(1);
epsilon = Q(2:4)';
epsilonhat = mapRtoRhat(epsilon);

% desired COM position, COM velocity, pelvis orientation
% (roll/pich/yaw) and rate of change of pelvis orientation
[r0_com, v0_com] = computeComPosVel(q0, dq0, model);
rCOMdes = r0_com; drCOMdes = zeros(3,1);
wd = zeros(3,1);

% Desired Force
fdes = m*g + -kp*( r_com - rCOMdes ) - kd*( v_com - drCOMdes );

% Desired Moment
Tr = -2*(delta*eye(3) + epsilonhat)*Kr*epsilon;
Tdes = R_db*(Tr - Dr*(dq(4:6) - wd));

% Construct Desired Wrench
Wdes = [fdes; flip(Tdes)];

[p1, p2, p3, p4] = computeFootPositions(q, model);

% Compute distance from COM to each contact point
r1 = p1 - r_com;
r2 = p2 - r_com;
r3 = p3 - r_com;
r4 = p4 - r_com;

r1hat = mapRtoRhat(r1);
r2hat = mapRtoRhat(r2);
r3hat = mapRtoRhat(r3);
r4hat = mapRtoRhat(r4);

G = [eye(3), eye(3), eye(3), eye(3);
     r1hat, r2hat, r3hat, r4hat];

% Optimization

alpha1 = 1;
alpha2 = 10e-3;
alpha3 = 10e-6;
I = [eye(3) zeros(3)];
O = [zeros(3) eye(3)];
%
H1 = 2*(alpha1*G'*I'*I*G + alpha2*G'*O'*O*G + alpha3);
```

```
    H1=(H1+H1')/2;
    f1 = (-2*alpha1*Wdes'*I'*I*G - 2*alpha2*Wdes'*O'*O*G);

   % Constraints
   % Friction Cone Approximation link
   % https://scaron.info/teaching/friction-cones.html
   mu = 0.8/(sqrt(2));
   A = [1 0 -mu zeros(1, 9);
       -1 0 -mu zeros(1, 9);
        0 1 -mu zeros(1, 9);
        0 -1 -mu zeros(1, 9);
        0 0 -1 zeros(1,9);
        zeros(1,3) 1 0 -mu zeros(1, 6);
        zeros(1,3) -1 0 -mu zeros(1, 6);
        zeros(1,3) 0 1 -mu zeros(1, 6);
        zeros(1,3) 0 -1 -mu zeros(1, 6);
        zeros(1,3) 0 0 -1 zeros(1,6);
        zeros(1,6) 1 0 -mu zeros(1, 3);
        zeros(1,6) -1 0 -mu zeros(1, 3);
        zeros(1,6) 0 1 -mu zeros(1, 3);
        zeros(1,6) 0 -1 -mu zeros(1, 3);
        zeros(1,6) 0 0 -1 zeros(1,3);
        zeros(1,9) 1 0 -mu ;
        zeros(1,9) -1 0 -mu;
        zeros(1,9) 0 1 -mu;
        zeros(1,9) 0 -1 -mu;
        zeros(1,9) 0 0 -1];

   b = [zeros(20,1)];

    options = optimset('Display','off', 'TolFun',1e-4);

    sol = quadprog(H1,f1,A,b, [], [], [], [], [],options);

    fc = sol;

   % Map Force to joint torques
    fc1 = [zeros(3,1);fc(1:3)];
    fc2 = [zeros(3,1);fc(4:6)];
    fc3 = [zeros(3,1);fc(7:9)];
    fc4 = [zeros(3,1);fc(10:12)];

    % tau_joints = -sum(J'*Fc)
    [J1f, J1b, J2f, J2b] = computeFootJacobians(s,model);

    % Sum of J'*fc (each J' should be 16x6, each fc should be 6x1, tau
 is
    % 16x1)
    tau = -(J1f'*fc1 + J1b'*fc2 + J2f'*fc3 + J2b'*fc4);

    % take out out xyz, roll/pitch/yaw from tau (only actuated joints)
    tau = tau(7:end);

end
```

```
function rhat = mapRtoRhat(r)

    rx = r(1);
    ry = r(2);
    rz = r(3);

    rhat = [0, -rz, ry;
            rz, 0, -rx;
            -ry, rx, 0];
end
```

Published with MATLAB® R2019b