A paper with drawings and text

AI-generated content may be incorrect.

A paper with drawings and text

AI-generated content may be incorrect.

MATLAB Code ::

%Cassie foot force application

clc

clear all

close all

syms c1 c2 r h

%x represents the travel distance of the linear actuator

function output=calc(x)

h=.032;

r=.055;

c1=.025;

c2=.025;

%x=.02; %m

% Define the solver function

function [theta\_sol, phi\_sol, exitflag] = solveNonlinearEqs(eq1, eq2, initial\_guess)

% SOLVENONLINEAREQS Solves two nonlinear equations dependent on theta and phi

% using optimization with bounds [0, pi/2].

%

% Inputs:

% eq1 - Anonymous function of (theta, phi) for the first equation

% eq2 - Anonymous function of (theta, phi) for the second equation

% initial\_guess - [theta0, phi0], initial guess within [0, pi/2]

%

% Outputs:

% theta\_sol - Solution for theta

% phi\_sol - Solution for phi

% exitflag - Optimization exit flag (1 = success, others indicate issues)

% Default initial guess if not provided

if nargin < 3 || isempty(initial\_guess)

initial\_guess = [pi/4, pi/4] % Midpoint of [0, pi/2]

end

% Validate initial guess

if any(initial\_guess < 0) || any(initial\_guess > pi/2)

error('Initial guess must be within [0, pi/2].');

end

% Define the objective function (sum of squared residuals)

objective = @(x) (eq1(x(1), x(2))^2 + eq2(x(1), x(2))^2);

% Bounds for theta and phi: [0, pi/2]

lb = [0, 0]; % Lower bounds

ub = [pi/2, pi/2]; % Upper bounds

% Optimization options (optional: display iterations)

options = optimoptions('fmincon', ...

'Display', 'iter', ... % Show iteration progress

'Algorithm', 'sqp', ... % Sequential quadratic programming

'TolFun', 1e-8, ... % Function tolerance

'TolX', 1e-8); % Step size tolerance

% Solve using fmincon

[x, ~, exitflag] = fmincon(objective, initial\_guess, [], [], [], [], ...

lb, ub, [], options);

% Extract solutions

theta\_sol = x(1);

phi\_sol = x(2);

% Verify solution (optional)

residual1 = eq1(theta\_sol, phi\_sol);

residual2 = eq2(theta\_sol, phi\_sol);

fprintf('Solution: theta = %.6f, phi = %.6f\n', rad2deg(theta\_sol), rad2deg(phi\_sol));

fprintf('Residuals: eq1 = %.2e, eq2 = %.2e\n', residual1, residual2);

if exitflag <= 0

warning('Optimization did not converge successfully (exitflag = %d).', exitflag);

end

end

%Equation 1

%angle of tarsal segment relation with theta and phi

psi=@(theta, phi) phi+asin(h/c2\*sin(phi));

%Equation 2

eq1=@(theta, phi) c2/r-sin(theta)\*sin(phi)/(sin(pi)-(phi+asin(h/c2\*sin(phi))));

%Equaiton 3

eq2=@(theta, phi) cos(phi)\*cos(theta)-(c1-x)/r;

[theta\_sol, phi\_sol, exitflag]=solveNonlinearEqs(eq1, eq2)

output=[rad2deg(theta\_sol) rad2deg(phi\_sol)];

end

x=linspace(0,.053);

thet\_vals=[];

phi\_vals=[];

vals=[];

for i = 1 : length(x)

aa=[];

a= calc(x(i));

aa=[x(i) (a)];

vals=[vals transpose(aa)];

end

thet\_vals=vals(2,:);

phi\_vals=vals(3,:);

F=1; %% assuming unit force applied by the linear actuators

Fx=[]

for i = 1 : length(x)

Fx=[Fx F\*cos(deg2rad(thet\_vals(i)))\*cos(deg2rad(phi\_vals(i)))\*cos(deg2rad(phi\_vals(i)))];

end

plot(x,Fx)

grid on

xlabel('x travel')

ylabel('Fx')