Today:

1) 4-bar position analysis

2) Matlab

3) Velocity analysis → power flow

---

midterm posted after class

→ due Tuesday before class

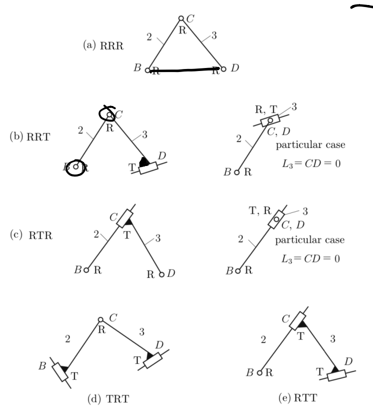
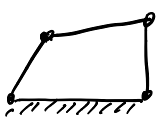


Fig. 1.9 Types of dyads: (a) RRR, (b) RRT, (c) RTR, (d) TRT, and (e) RTT

Dyads

4-bar linkage

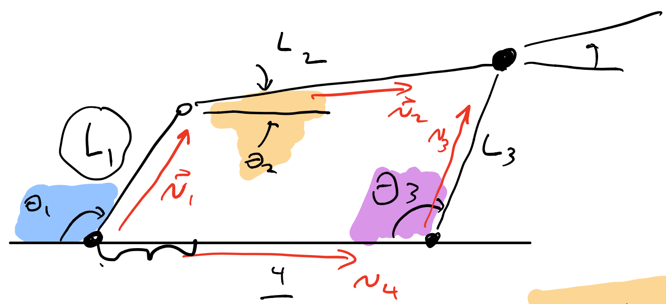


$= 3(l-1) - 2j$

$3 \cdot n - 2j$

l = All links

n = mobile links



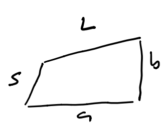
Closed chain → $f(\vec{x}) = 0$

Any link that can fully rotate → crank

Any that oscillates → rocker

$$\vec{v}_1 + \vec{v}_2 = \vec{v}_3 + \vec{v}_4$$

$$\rightarrow \vec{v}_1 + \vec{v}_2 - \vec{v}_3 - \vec{v}_4 = 0$$

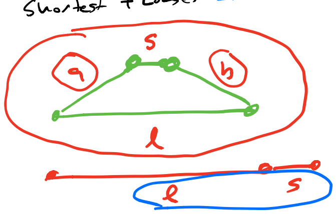


→

$$\boxed{\begin{aligned} x&: L_1 \cos\theta_1 + L_2 \cos\theta_2 - L_4 - L_3 \cos\theta_3 = 0 \\ y&: L_1 \sin\theta_1 + L_2 \sin\theta_2 - 0 - L_3 \sin\theta_3 = 0 \end{aligned}}$$

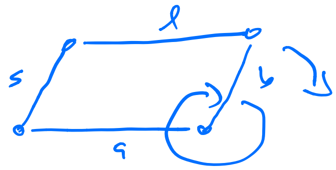
Rocker: Shortest + Longest $>$ a + b

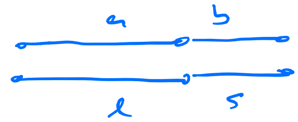
crank: Shortest + Longest $\leq$ a + b



$$\theta_1 + \theta_2 + \theta_3 + \theta_4 = 360$$

$l+s \geq a+b$

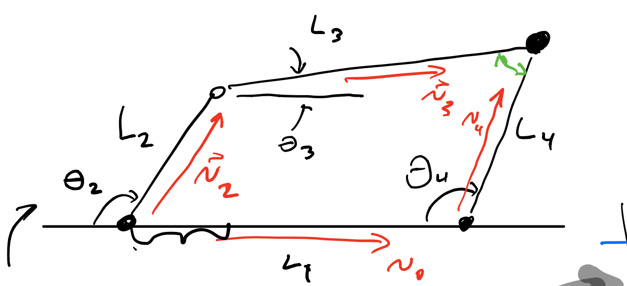




$l+s = a+b$

$\theta_1+\theta_2+\theta_3+\cancel{\theta_4} = 2\pi$

$f(\vec{x}) = 0$



$\vec{v}_1+\vec{v}_2 = \vec{v}_3+\vec{v}_4$

$\Rightarrow \vec{v}_1+\vec{v}_2-\vec{v}_3-\vec{v}_4 = 0$

$$\boxed{\begin{aligned} &x:\ L_2\cos\theta_2 + L_3\cos\theta_3 - L_1 - L_4\cos\theta_4 = 0 \\ &y:\ L_2\sin\theta + L_3\sin\theta_3 - 0 - L_4\sin\theta_4 = 0 \end{aligned}}$$

Loop closure equation

$\vec{v}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$

$f(\vec{x}) = 0$

$\underline{\vec{X}} = \begin{bmatrix} \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$

full configuration

$\theta_2 \to$ Driver joint

$\theta_4 \to$ output joint

want to eliminate

x: $L_2\cos\theta_2 + L_3\cos\theta_3 - L_1 - L_4\cos\theta_4 = 0$

y: $L_2\sin\theta_2 + L_3\sin\theta_3 - \cancel{0} - L_4\sin\theta_4 = 0$

$\theta_2 \to \theta_4$

1) $\underline{L_3\cos\theta_3} = L_1 + L_4\cos\theta_4 - L_2\cos\theta_2$

2) $\underline{L_3\sin\theta_3} = +\underline{L_4\sin\theta_4} - L_2\sin\theta_2$

square and add 1 & 2

$(L_1 + L_4\cos\theta_4 - L_2\cos\theta_2)(L_1 + L_4\cos\theta_4 - L_2\cos\theta_2)$

$$L_3(\underline{\underline{\cos^2\theta_3+\sin^2\theta_3}}) = L_4^2\sin^2\theta_4 + L_2^2\sin^2\theta_2 - 2L_4L_2\sin\theta_4\sin\theta_2 +$$

$$+L_1^2 + L_1L_4\cos\theta_4 - L_1L_2\cos\theta_2 + L_1L_4\cos\theta_4$$

$$+L_4^2\cos^2\theta_4 - L_2\cos\theta_2 L_4\cos\theta_4 - L_1L_2\cos\theta_2$$

$$- L_2L_4\cos\theta_2\cos\theta_4 + L_2^2\cos^2\theta_2$$

$$= L_2^2 + L_4^2 + L_1^2 - 2L_1L_2\cos\theta_2 + 2L_1L_4\cos\theta_4$$

$$-2L_2L_4\Big(\underbrace{\sin\theta_4\sin\theta_2+\cos\theta_2\cos\theta_4}_{\cos(\theta_2-\theta_4)}\Big)$$

$$L_3^2 = L_2^2+L_4^2+L_1^2 - 2L_1(L_4\cos\theta_4 - L_2\cos\theta_2) - 2L_2L_4\cos(\theta_2-\theta_4)$$

$$D_3 = \frac{L_1^2+L_2^2-L_3^2}{L_2L_4} \qquad D_2 = \frac{L_1}{L_2} \qquad D_1 = \frac{L_1}{L_4}$$

Freudenstein equation

$$\boxed{D_3 - \cos(\theta_2-\theta_4) + D_2\cos(\theta_4) - D_1\cos(\theta_2) = 0}$$

* Simple 1-equation constraint relationship that solves the $\theta_2 \rightarrow \theta_4$

Two methods to solve equation:

1) Symbolic tools to explicitly solve

```
%% Start anew

clear
clc
close all

%% Define geometry

theta_2 = pi/2;
l1 = 2;
l2 = 1;
l3 = 2;
l4 = 1;

%% Set up the problem symbolically

% define variables
syms theta_3_sym theta_4_sym

assume(theta_3_sym, 'real');
assume(theta_4_sym, 'real');

assume(theta_3_sym >= 0 & theta_3_sym < 2*pi)
assume(theta_4_sym >= 0 & theta_4_sym < 2*pi)

% define constraint equations
A = l2*cos(theta_2) + l3*cos(theta_3_sym) - l4*cos(theta_4_sym) - l1;
B = l2*sin(theta_2) + l3*sin(theta_3_sym) - l4*sin(theta_4_sym);

%% Solve symbolically

sol1 = solve([A==0, B==0], [theta_3_sym, theta_4_sym]);

theta_3 = eval(sol1.theta_3_sym)
theta_4 = eval(sol1.theta_4_sym)

% two solutions
```

```
%% Let's look at an animation!!

% Original geometry
% l1 = 2;
% l2 = 1;
% l3 = 2;
% l4 = 1;

% test something else
l1 = 2;
l2 = 1;
l3 = 2;
l4 = 1.5;


for theta_2 = 0:0.1:100000

    a = mod(theta_2, 2*pi);

    % define constraint equations
    A = l2*cos(theta_2) + l3*cos(theta_3_sym) - l4*cos(theta_4_sym) - l1;
    B = l2*sin(theta_2) + l3*sin(theta_3_sym) - l4*sin(theta_4_sym);

    %% Solve symbolically

    sol1 = solve([A==0, B==0], [theta_3_sym, theta_4_sym]);

    theta_3 = eval(sol1.theta_3_sym);
    theta_4 = eval(sol1.theta_4_sym);

    %% plot
    ii=1;
    l1_vec = [l1, 0];
    l2_vec = [l2*cos(theta_2), l2*sin(theta_2)];
    l3_vec = l2_vec + [l3*cos(theta_3(ii)), l3*sin(theta_3(ii))];
    l4_vec = l1_vec + [l4*cos(theta_4(ii)), l4*sin(theta_4(ii))]; % -x because epsilon wrt interior angle


    clf;

    plot([0, l1_vec(1)], [0, l1_vec(2)], 'o-', 'linewidth', 3); % link1
    hold on;
    plot([0, l2_vec(1)], [0, l2_vec(2)], 'o-', 'linewidth', 3); % link2
    plot([l2_vec(1), l3_vec(1)], [l2_vec(2), l3_vec(2)], 'o-', 'linewidth', 3); % link3
    plot([l1_vec(1), l4_vec(1)], [l1_vec(2), l4_vec(2)], 'o-', 'linewidth', 3); % link4


    axis equal;
    axis(3.5*[-1, 1, -1, 1]);

    drawnow;

end
```

fsolve

Solve system of nonlinear equations

Syntax

```
x = fsolve(fun,x0)
x = fsolve(fun,x0,options)
x = fsolve(problem)
[x,fval] = fsolve(___)
[x,fval,exitflag,output] = fsolve(___)
[x,fval,exitflag,output,jacobian] = fsolve(___)
```

2) Numerical solution

fsolve → constraint equation

Need

1) $f(\vec{x}) = 0$

2) guess x_0

check solution

$fval = f(\vec{x}_{solution})$

0 or small

Description

Nonlinear system solver

Solves a problem specified by

$$F(x) = 0$$

for x, where $F(x)$ is a function that returns a vector value.

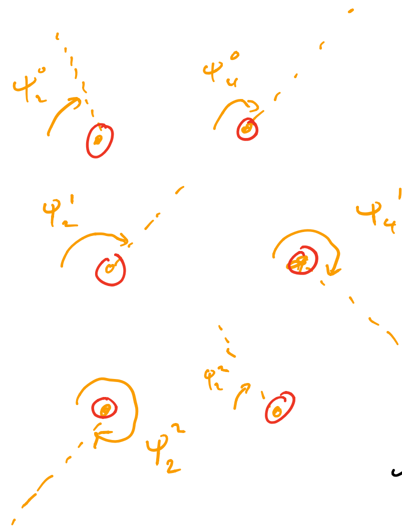
x is a vector or a matrix; see Matrix Arguments.

`x = fsolve(fun,x0)` starts at `x0` and tries to solve the equations `fun(x) = 0`, an array of zeros.

```matlab
%% Let's look at an animation!!

% Original
l1 = 2;
l2 = 1;
l3 = 2;
l4 = 1.2;


% l1 = 1;
% l2 = 2;
% l3 = .5;
% l4 = 1;

x_guess = [0, 0];

input_angle = 0;
output_angle = 0;
cnt = 1;

omega = 1;

for t = 0:0.1:100000

    theta_2 = mod(omega*t, 2*pi);

    %% Solve numerically

    [x,fval] = fsolve(@(x) four_bar_constraint(x, theta_2, l1, l2, l3, l4), [0, output_angle(end)]);

    theta_3 = x(1);
    theta_4 = x(2);

    input_angle(cnt) = theta_2;
    output_angle(cnt) = theta_4;
    cnt = cnt + 1;


    % Only plot solutions that satisfy a certain final tolerance
    if norm(fval) < 0.000001
        %% plot

        l1_vec = [l1, 0];
        l2_vec = [l2*cos(theta_2), l2*sin(theta_2)];
        l3_vec = l2_vec + [l3*cos(theta_3), l3*sin(theta_3)];
        l4_vec = l1_vec + [l4*cos(theta_4), l4*sin(theta_4)]; % -x because epsilon wrt interior angle


        clf;

        plot([0, l1_vec(1)], [0, l1_vec(2)], 'o-', 'linewidth', 3); % link1
        hold on;
        plot([0, l2_vec(1)], [0, l2_vec(2)], 'o-', 'linewidth', 3); % link2
        plot([l2_vec(1), l3_vec(1)], [l2_vec(2), l3_vec(2)], 'o-', 'linewidth', 3); % link3
        plot([l1_vec(1), l4_vec(1)], [l1_vec(2), l4_vec(2)], 'o-', 'linewidth', 3); % link4



        axis equal;
        axis(3.5*[-1, 1, -1, 1]);

        drawnow;
    end


%     pause;
end
```

$$\boxed{D_3 - \cos(\theta_2 - \theta_4) + D_2 \cos(\theta_4) - D_1 \cos(\theta_2) = 0}$$

Motion synthesis with F equation



$$\begin{cases} D_3 - \cos(\varphi_2^0 - \varphi_4^0) + D_2 \cos(\varphi_4^0) - D_1 \cos \varphi_2^0 = 0 \\ D_3 - \cos(\varphi_2^1 - \varphi_4^1) + D_2 \cos(\varphi_4^1) - D_1 \cos \varphi_2^1 = 0 \\ D_3 - \cos(\varphi_2^2 - \varphi_4^2) + D_2 \cos(\varphi_4^2) - D_1 \cos \varphi_2^2 = 0 \end{cases}$$

D_3, D_2, D_1

$$\underline{\underline{f(\underline{\underline{\vec{x}}}) = 0}}$$

$$\vec{x} = \begin{bmatrix} D_3 \\ D_2 \\ D_1 \end{bmatrix}$$