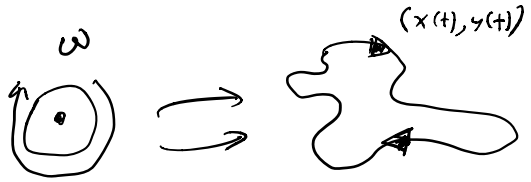
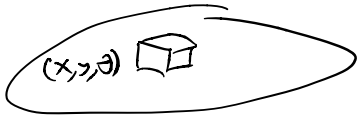
Linkage systems



- Mobility

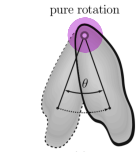


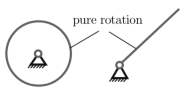
planer system

6-DoF Spatial system

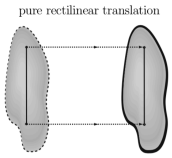
Connecting rigid bodies together through joints removes DOF.

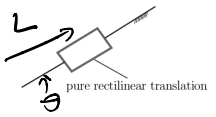
Mobility of a linkage system is the DOF



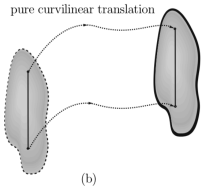


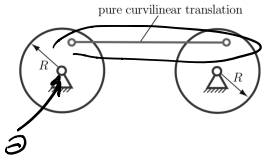
Revolute joints



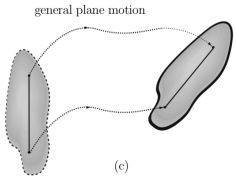


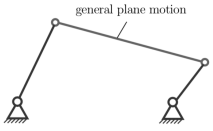
← Prismatic joint



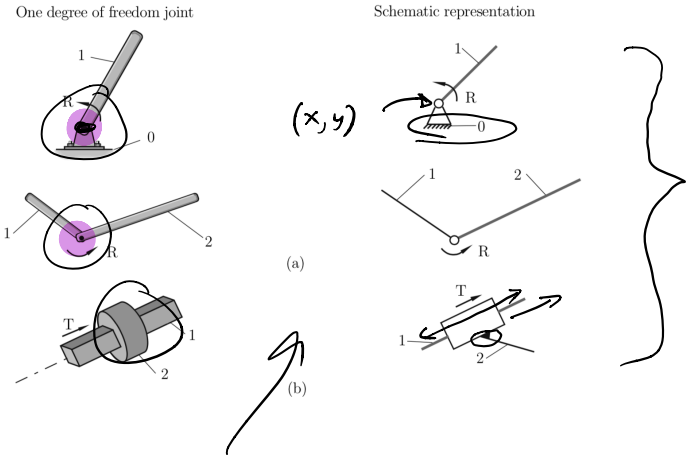


Single DOF



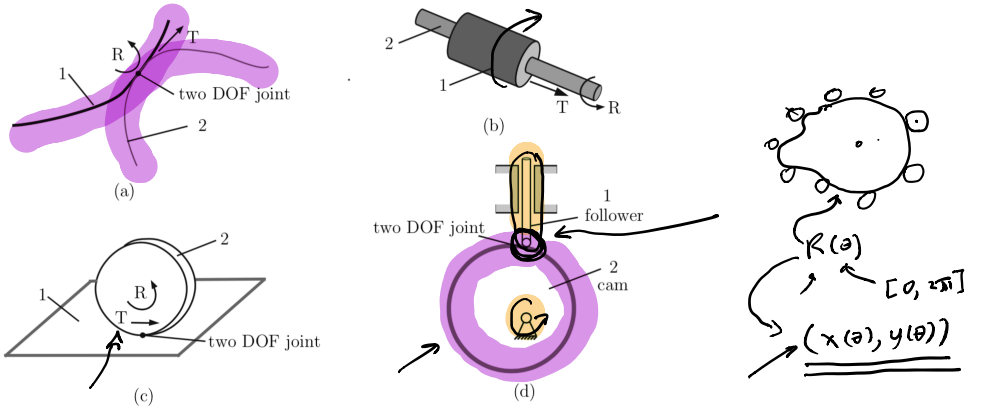


Planar systems

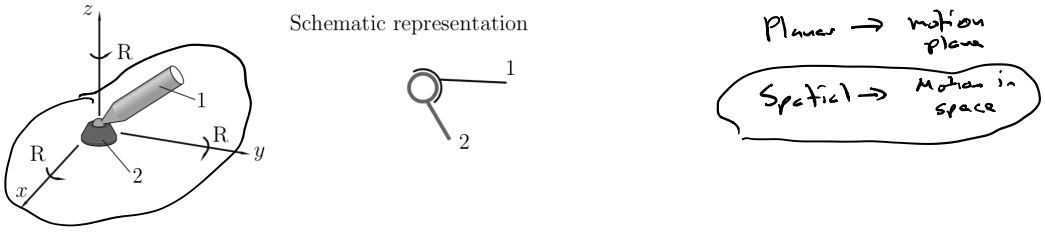


1-DOF joints

2 DOF joint



3-DOF joint ⟶ Spatial joint



Planar → motion plane

Spatial → Motion in space

Mobility equation: Gruebler equation
Chebyshev condition

$$M = \lambda(l - j - 1) + \sum_{i}^{j} f_i$$

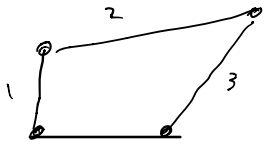
- λ: 3: for planar systems; 3: for spherical systems; 6: Generic spatial linkages
- l: # links in mechanism
- j: # joint in mechanism
- f_i: DOF of each joint

Planar systems: $f_i = 1$ $\lambda = 3$

$M = 3 \cdot (l-1) - 3j + j$

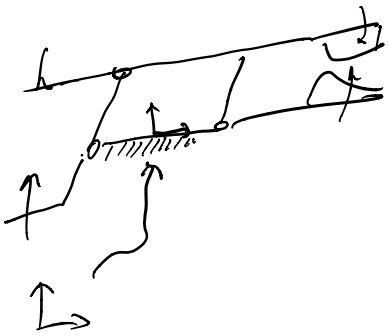
$M = 3 \cdot (l-1) - 2j$

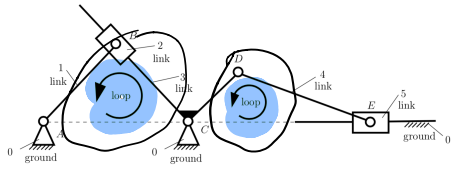
Total # of links

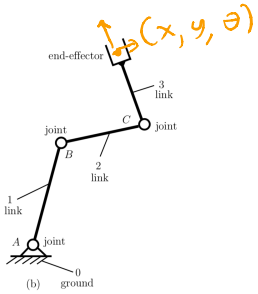


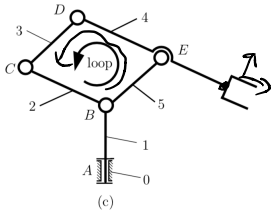
$M = 3 \cdot n - 2 \cdot j$ ← $l - 1 = n$

mobile links



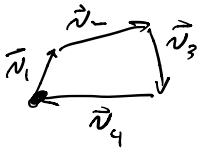






Closed-kinematic chains

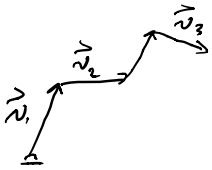
$f(\vec{x}) = 0$



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

Open-chain linkages

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



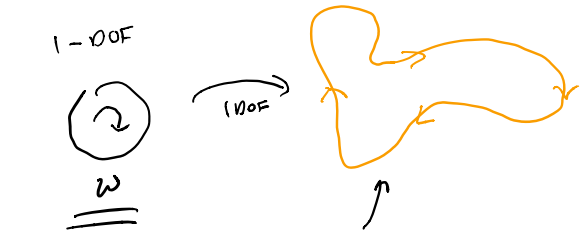
Mixed systems

Planar mobility

$$M = 3 \cdot (l - 1) - 2 \cdot j$$

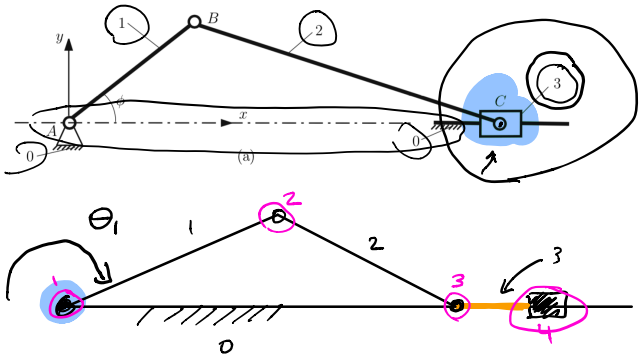
$$= 3 \cdot n - 2j$$

Motor | Mechanism | Output



RRRP

$M = 3 \cdot (\ell - 1) - 2 \cdot j$



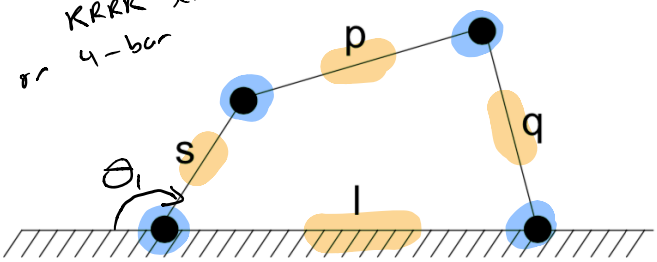
$\ell = 4$

$j = 4$

$M = 3 \cdot 3 - 2 \cdot 4$

$= 1$

KRRR linkage or 4-bar



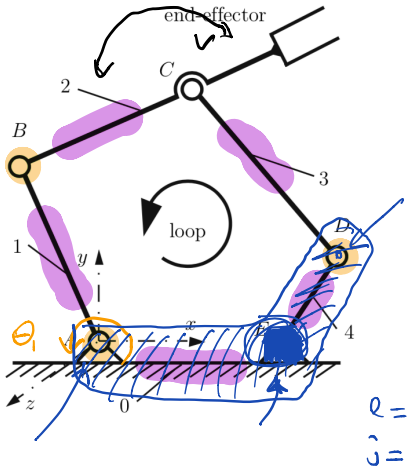
$\ell = 4$

$j = 4$

$M = 3 \cdot 3 - 2 \cdot 4$

$= 1$

5-bar linkage



$\ell = 5$

$j = 5$

$M = 3 \cdot (4) - 2 \cdot 5$

$= 2$

$\ell = 4 \rightarrow M = 1$

$j = 4$

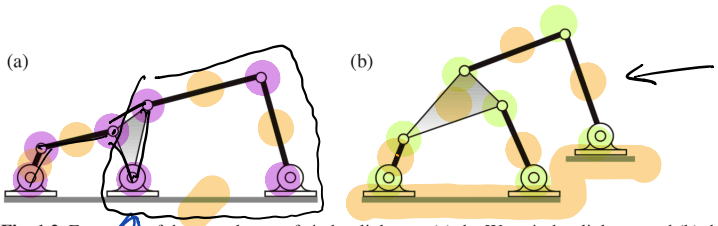


Fig. 1.2 Examples of the two classes of six bar linkages: (a) the Watt six-bar linkage; and (b) the Stephenson six-bar linkage.

$l = 6$

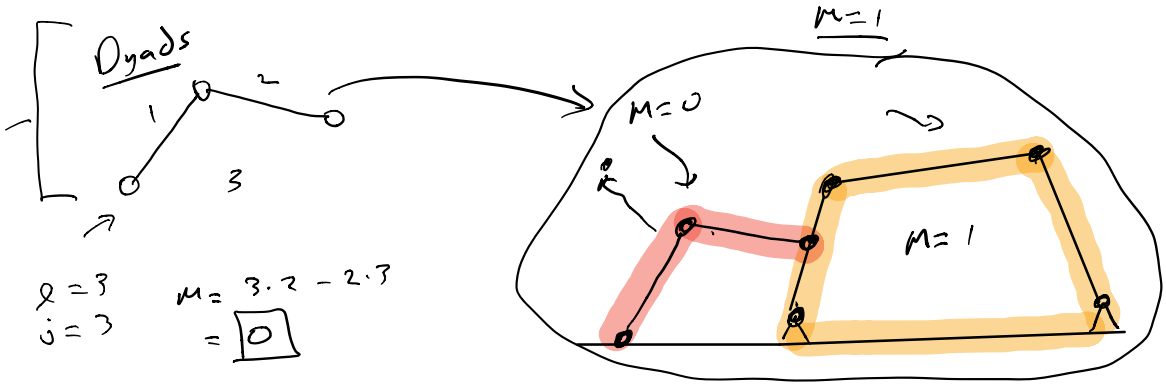
$j = 7$

$m = 1$

$l = 6$ $\quad m = 3.5 - 2.7$

$j = 7$ $\quad = 15 - 14$

$= \boxed{1}$



$l = 3$

$j = 3$

$m = 3.2 - 2.3$

$= \boxed{0}$

Dyads can be added to mechanism and don't change the net mobility.

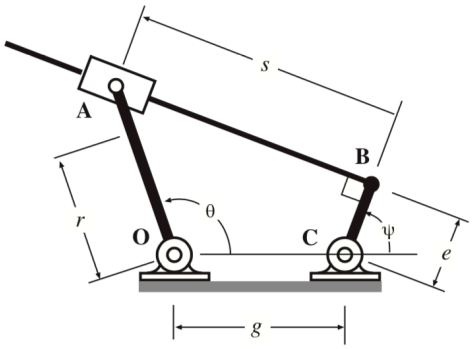
1-DOF planar systems

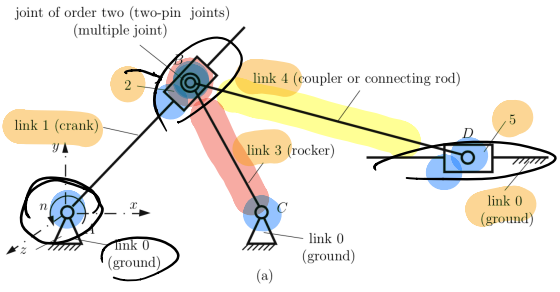
$$m = 1 = 3 \cdot (l-1) - 2j$$

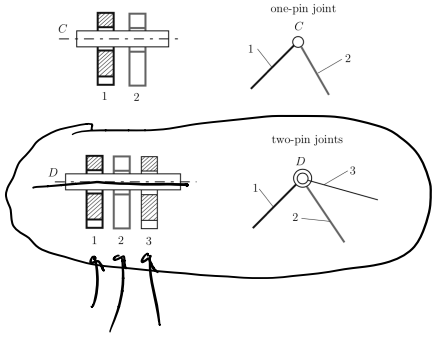
l even for m = 1

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

$$4 + 2j = 3l \rightarrow \quad l = \frac{4+2j}{3}$$







$l = 6$

$j = 7$

$M = 3(6-1) - 2 \cdot 7$

$= 3 \cdot 5 - 2 \cdot 7$

$= 1$

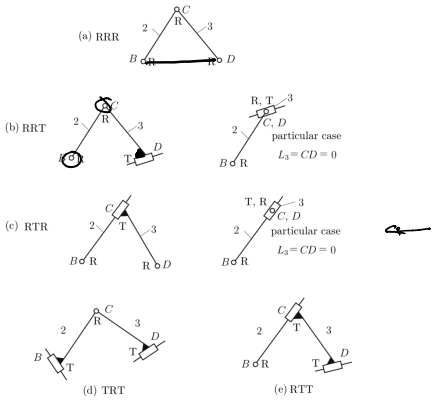
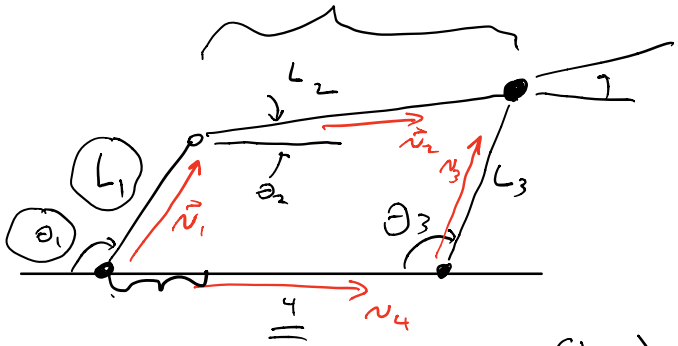


Fig. 1.9 Types of dyads: (a) RRR, (b) RRT, (c) RTR, (d) TRT, and (e) RTT

4-bar linkage



Any link that can fully rotate → crank

Any that oscillates → rocker

Rocker

Shortest + Longest < a+b

crank

Shortest + Longest > a+b

Closed chain → $f(\vec{x}) = 0$

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

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

$x: L_1 \cos\theta_1 + L_2 \cos\theta_2 - L_4 - L_3 \cos\theta_3 = 0$

$y: L_1 \sin\theta_1 + L_2 \sin\theta_2 - 0 - L_3 \sin\theta_3 = 0$

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

```
%% 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(__)
```

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.

```
%% 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
```