BACKHOE KINEMATICS

For use in the design of the Haptic Backhoe project

Joe Frankel

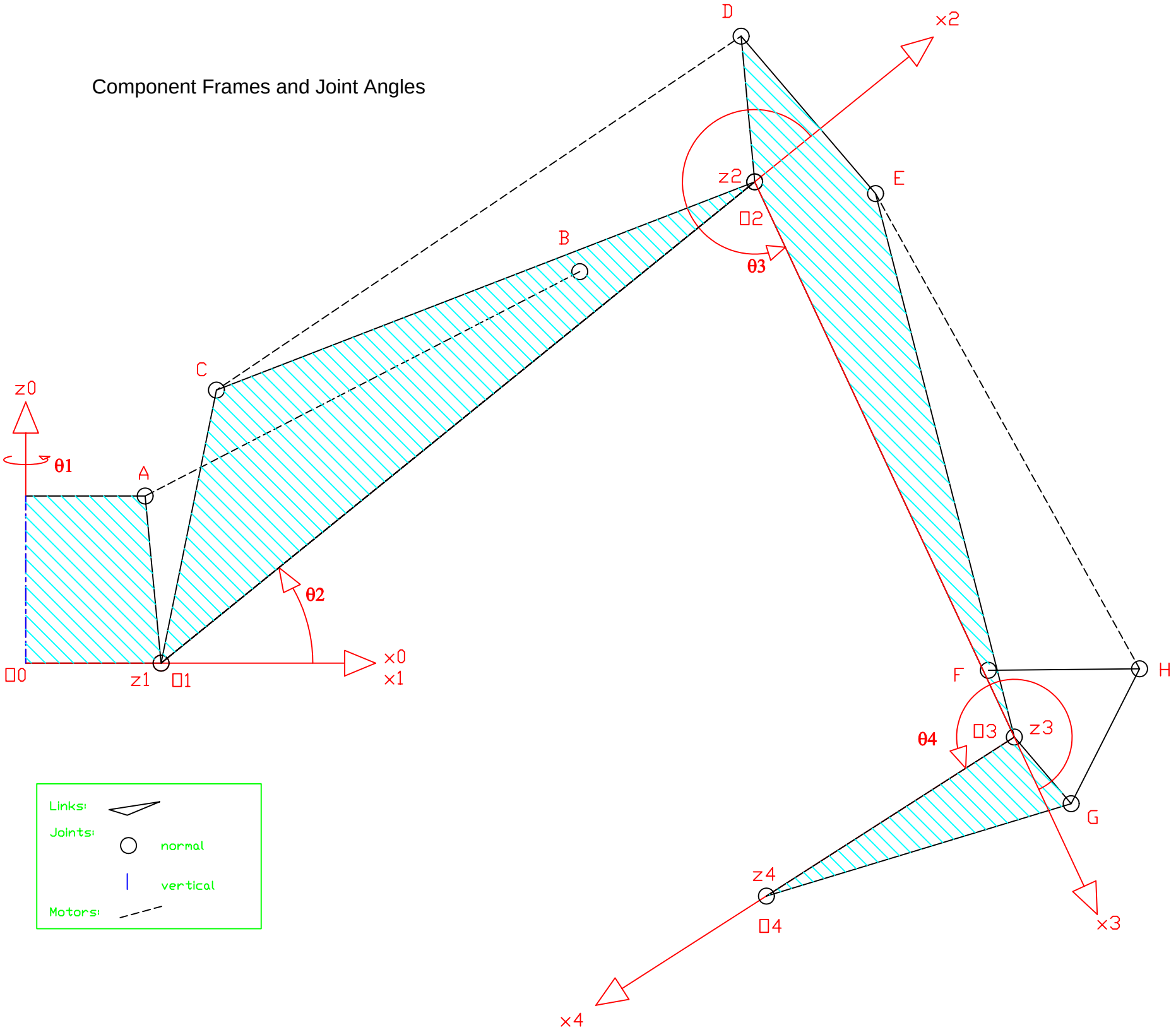
Georgia Institute of Technology

Intelligent Machines Dynamics Laboratory

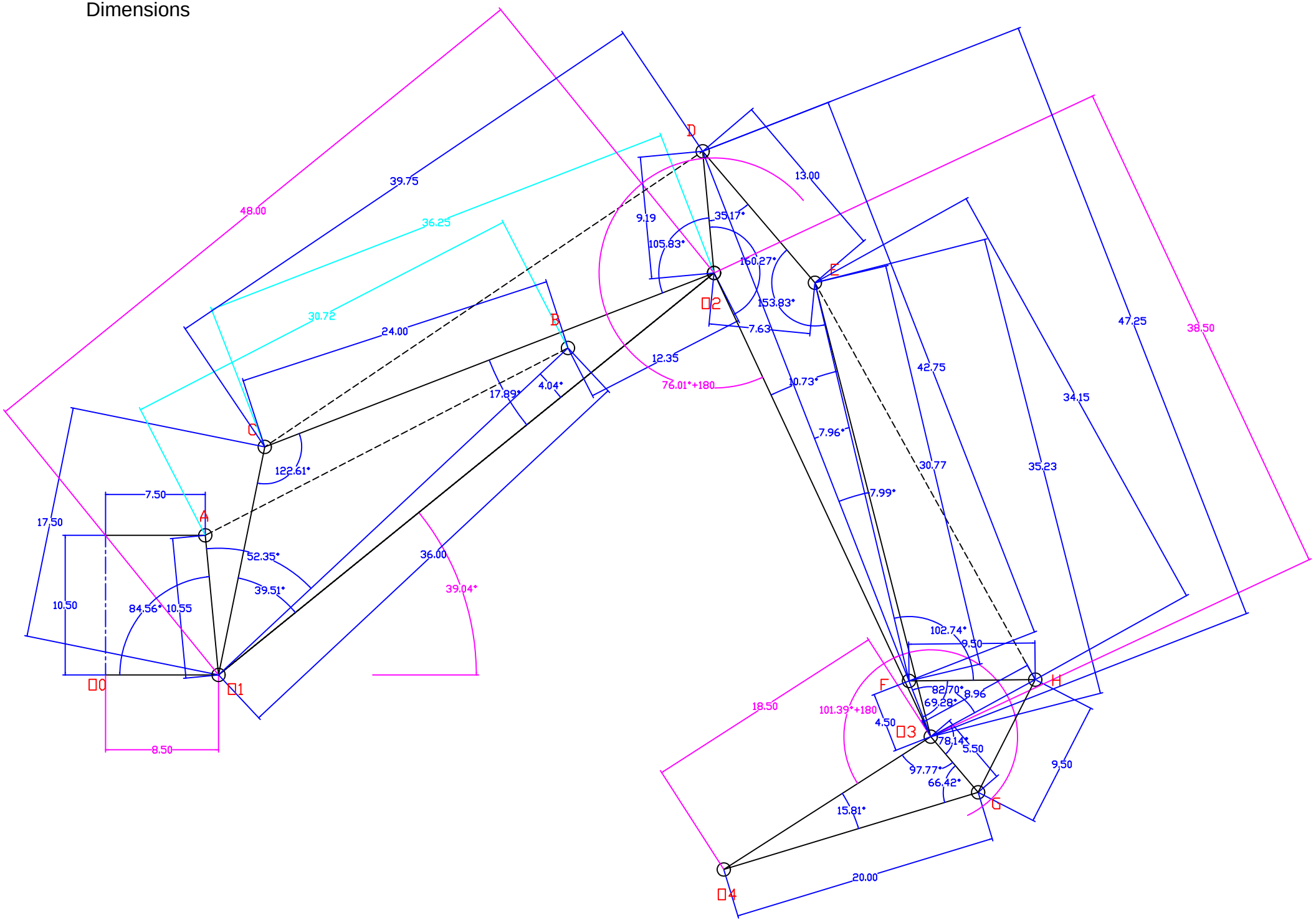
March 6, 2003

CONTENTS

Component Frames and Joint Angles



Dimensions


A
B
C
D
E
F
G
H
□0
□1
□2
□3
□4
48.00
39.75
36.25
30.72
24.00
13.00
9.19
35.17°
105.83°
150.27°
153.83°
7.63
12.35
76.01°+180
17.89°
4.04°
122.61°
7.50
17.50
52.35°
39.51°
84.56°
10.55
10.50
36.00
39.04°
8.50
10.73°
42.75
47.25
38.50
34.15
7.96°
30.77
35.23
7.99°
102.74°
9.50
82.70°
8.96
69.28°
101.39°+180
18.50
4.50
78.14°
5.50
97.77°
66.42°
9.50
15.81°
20.00

3-2-03 REVERSE DISPLACEMENT ANALYSIS

GIVEN: Bucketdisp => x, y, z & orientation

FIND: $\theta_1, \theta_2, \theta_3, \theta_4$

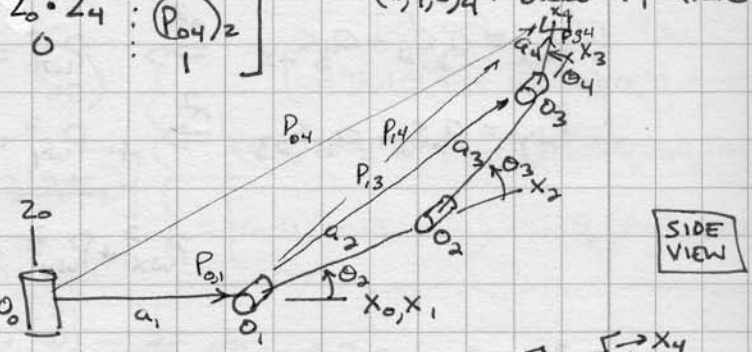
SOLUTION:

Bucketdisp → Rdisp → theta

$$\text{Bucketdisp} = B = \begin{bmatrix} X_0 \cdot X_4 & X_0 \cdot Y_4 & X_0 \cdot Z_4 & (P_{04})_x \\ Y_0 \cdot X_4 & Y_0 \cdot Y_4 & Y_0 \cdot Z_4 & (P_{04})_y \\ Z_0 \cdot X_4 & Z_0 \cdot Y_4 & Z_0 \cdot Z_4 & (P_{04})_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$(x,y,z)_0$: base frame

$(x,y,z)_4$: bucket tip frame

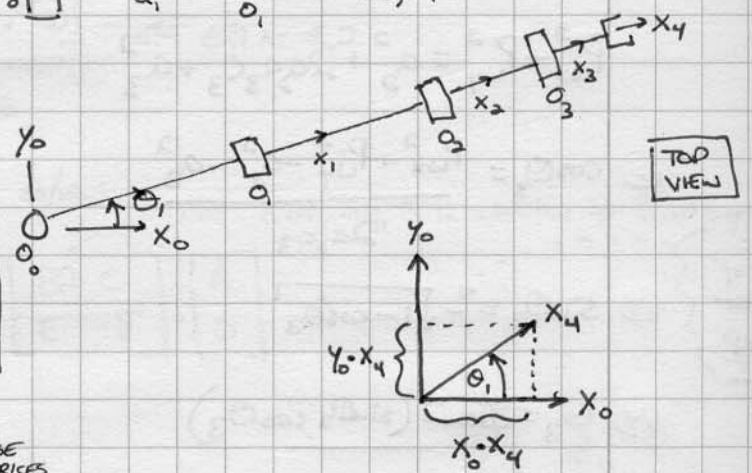


θ_1 is constant along boom, bucket, & stick:

$$\tan\theta_1 = \frac{Y_0 \cdot X_4}{X_0 \cdot X_4}$$

NOTE: errors occur for $B(2,1) = -0.000$ use $\text{atan}(B(2,1)/B(1,1))$ instead

① $\theta_1 = \text{atan2}(B(2,1), B(1,1))$



Frame 0→1 is a 2-rotation by θ_1:

② $R_1^0 = R_y(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ 0 & 1 & 0 \\ -\sin\theta_1 & 0 & \cos\theta_1 \end{bmatrix}$

$R_x(90°)$ ← oops! forget this...

$R_x(90°)$ $R_y(\theta_1)$

Origin 1 wrt to origin 0:

③ $P_{0,1}^0 = \begin{bmatrix} a_1 \cos\theta_1 \\ a_1 \sin\theta_1 \\ 0 \end{bmatrix}$ ⇒ ④ $P_{0,1}^1 = \left(R_1^0\right)^T P_{0,1}^0$

INVERSE = TRANSPOSE FOR ROTATION MATRICES

End-effector (bucket tip) wrt origin 0:

⑤ $P_{0,4}^1 = \left(R_1^0\right)^T P_{0,4}^0 = \left(R_1^0\right)^T \begin{bmatrix} B(1,4) \\ B(2,4) \\ B(3,4) \end{bmatrix}$

Wrist wrt origin 4 (bucket tip)

⑥ $P_{3,4}^0 = -a_4 \begin{bmatrix} X_0 \cdot X_4 \\ Y_0 \cdot X_4 \\ Z_0 \cdot X_4 \end{bmatrix} = -a_4 \begin{bmatrix} B(1,1) \\ B(2,1) \\ B(3,1) \end{bmatrix}$

↑ WRIST LIES ALONG $-X_4$ AXIS

IN frame 1, wrist position is:

⑦ $P_{3,4}^1 = \left(R_1^0\right)^T P_{3,4}^0$

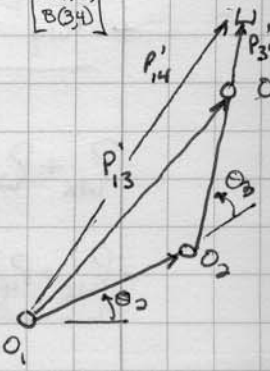
Bucket tip wrt frame 1:

⑧ $P_{1,4}^0 = P_{0,4}^0 - P_{0,1}^0$ ⇒ ⑨ $P_{1,4}^1 = \left[\left(R_1^0\right)^T P_{0,4}^0\right] - \left[R(\theta_1)^T P_{0,1}^0\right]$

② $\begin{bmatrix} B(1,4) \\ B(2,4) \\ B(3,4) \end{bmatrix}$ ② ③

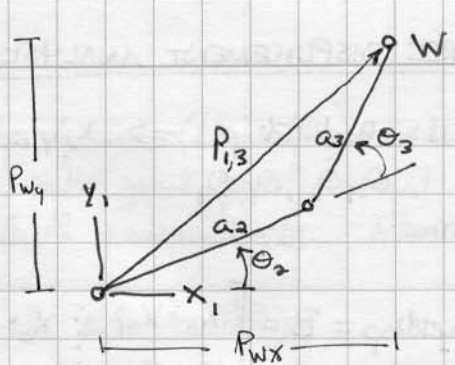
Wrist position in frame 1:

⑩ $P_{1,3}^1 = P_{1,4}^1 - P_{3,4}^1$



SOLVE FOR θ_3:

$$P'_{1,3} = \begin{bmatrix} a_2\cos\theta_2 + a_3\cos(\theta_2+\theta_3) \\ a_2\sin\theta_2 + a_3\sin(\theta_2+\theta_3) \\ 0 \end{bmatrix}$$



(11) Let $(P'_{1,3})_x = P_{WX}$, $(P'_{1,3})_y = P_{WY}$, then

$$\begin{cases} P_{WX} = a_2c_2 + a_3c_{23} \\ P_{WY} = a_2s_2 + a_3s_{23} \end{cases}$$

$$\Rightarrow P_{WX}^2 = a_2^2c_2^2 + 2a_2a_3c_2c_{23} + a_3^2c_{23}^2$$

$$+ \; P_{WY}^2 = a_2^2s_2^2 + 2a_2a_3s_2s_{23} + a_3^2s_{23}^2$$

$$P_{WX}^2 + P_{WY}^2 = a_2^2(c_2^2+s_2^2) + 2a_2a_3(c_2c_{23}+s_2s_{23}) + a_3^2(c_{23}^2+s_{23}^2)$$

TRIG: $\cos x\cos y + \sin x\sin y = \cos(x-y)$;

Let $x=\theta_2$, $y=\theta_2+\theta_3$, then

$\cos\theta_2\cos(\theta_2+\theta_3) + \sin(\theta_2\sin(\theta_2+\theta_3) = \cos(\theta_2-(\theta_2+\theta_3$

$= \cos(-\theta_3) = \cos\theta_3$, so

$$c_2c_{23} + s_2s_{23} = c_3$$

$$P_{WX}^2 + P_{WY}^2 = a_2^2 + 2a_2a_3c_3 + a_3^2$$

(12) $$\cos\theta_3 = \frac{P_{WX}^2 + P_{WY}^2 - a_2^2 - a_3^2}{2a_2a_3}$$

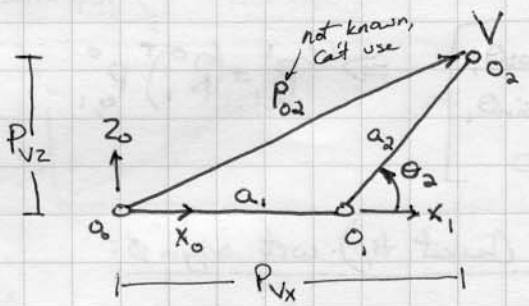
NOTE: "elbow up" position always

(13) $$\sin\theta_3 = \pm\sqrt{1-\cos^2\theta_3}$$

(14) $$\theta_3 = \operatorname{atan2}(\sin\theta_3, \cos\theta_3)$$

SOLVE FOR θ_2:

$$P^0_{0,2} = \begin{bmatrix} P_{Vx} \\ P_{Vy} \\ P_{Vz} \end{bmatrix} = \begin{bmatrix} a_1c_1 + a_2c_1c_2 \\ a_1s_1 + a_2s_1c_2 \\ a_2s_2 \end{bmatrix}$$



Try equations from above, now that θ_3 is known:

1ST EQN: $$P_{WX} = a_2c_2 + a_3c_{23}$$

$$+ \; P_{WY} = a_2s_2 + a_3s_{23}$$

$$P_{WX} + P_{WY} = a_2(c_2+s_2) + a_3(c_{23}+s_{23})$$

$$P_{WX} + P_{WY} = a_2(c_2+s_2) + a_3\left[c_2(c_3+s_3) + s_2(c_3-s_3)\right]$$

TRIG: $\cos(x+y) = \cos x\cos y - \sin x\sin y$

$\sin(x+y) = \sin x\cos y + \cos x\sin y$

Let $x=\theta_2$, $y=\theta_3$, then

$\left[\cos(\theta_2+\theta_3) + \sin(\theta_2+\theta_3)\right]$

$= \left[(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) + (\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3)\right]$

$= \left[\cos\theta_2(\cos\theta_3+\sin\theta_3) + \sin\theta_2(\cos\theta_3-\sin\theta_3)\right]$

$$\underbrace{P_{wx}+P_{wy}}_{(15)\ A} = C_2\underbrace{\left[a_2+a_3(C_3+S_3)\right]}_{(16)\ B} + S_2\underbrace{\left[a_2+a_3(C_3-S_3)\right]}_{(17)\ C} \quad \Leftarrow \text{EQN. \#1}$$

2ND EQN:

$$P_{wx} = a_2C_2 + a_3C_{23}$$

$$-P_{wy} = a_2S_2 + a_3S_{23}$$

$$P_{wx}-P_{wy} = a_2(C_2-S_2) + a_3(C_{23}-S_{23})$$

$$P_{wx}-P_{wy} = a_2(C_2-S_2) + a_3\left[C_2(C_3-S_3) - S_2(S_3+C_3)\right]$$

TRIG:

$\cos(x+y) = \cos x\cos y - \sin x\sin y$

$\sin(x+y) = \sin x\cos y + \cos x\sin y$

Let $x=\theta_2$, $y=\theta_3$, then

$$\left[\cos(\theta_2+\theta_3) - \sin(\theta_2+\theta_3)\right]$$

$$= \left[(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) - (\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3)\right]$$

$$= \left[\cos\theta_2(\cos\theta_3 - \sin\theta_3) - \sin\theta_2(\sin\theta_3 + \cos\theta_3)\right]$$

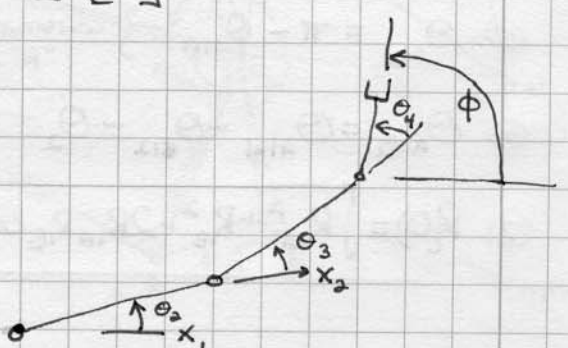
$$\underbrace{P_{wx}-P_{wy}}_{(18)\ D} = C_2\underbrace{\left[a_2+a_3(C_3-S_3)\right]}_{(19)\ E=C} - S_2\underbrace{\left[a_2+a_3(S_3+C_3)\right]}_{(20)\ F=B} \quad \Leftarrow \text{EQN. \#2}$$

Now arrange two eqns. into a matrix & solve:

NOTE: already using B in subroutine for Bucket disp

$$(21)\quad \begin{bmatrix} A \\ D \end{bmatrix} = \begin{bmatrix} B & C \\ E & -F \end{bmatrix}\begin{bmatrix} C_2 \\ S_2 \end{bmatrix} \Rightarrow \begin{bmatrix} C_2 \\ S_2 \end{bmatrix} = \begin{bmatrix} B & C \\ E & -F \end{bmatrix}^{-1}\begin{bmatrix} A \\ D \end{bmatrix}$$

$$(22)\quad \theta_2 = \text{atan2}(S_2, C_2)$$

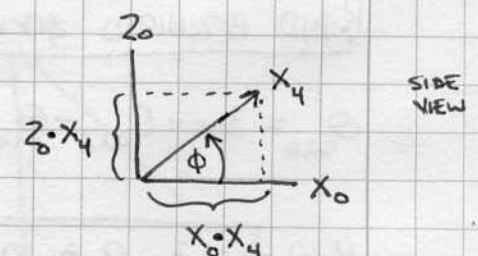


SOLVE FOR θ_4:

$$\phi = \theta_2+\theta_3+\theta_4;$$

$$\tan\phi = \left(\frac{Z_0 \cdot X_4}{X_0 \cdot X_4}\right) \Rightarrow \phi = \text{atan2}(B(3,1), B(1,1)) \quad (23)$$

$$(24)\quad \theta_4 = \phi - \theta_2 - \theta_3$$



[This solution path loosely follows the derivation laid out in "Modeling and Control of Robot Manipulators", Sciavicco and Siciliano, p.67-68 (1996)
Solution of Three-Link Planar Arm (almost) → has a 4th link!]

... Now put into Matlab ...

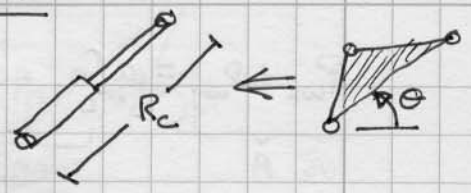
3-2-03

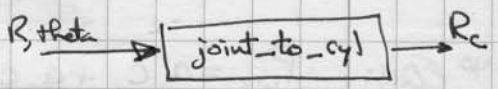
3-3-03 JOINT ANGLE TO CYLINDER POSITION CONVERSION

GIVEN: theta = $[\theta_1\ \theta_2\ \theta_3\ \theta_4]$

FIND: $R_c = [R_c(1)\ R_c(2)\ R_c(3)\ R_c(4)]$

SOLUTION:





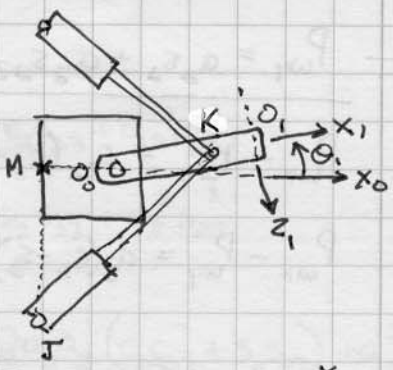
FIND REQUIRED SWING CYLINDER LENGTH FROM θ_1:

① $R_{OJ} = \sqrt{R_{OM}^2 + R_{JM}^2}$

② $\theta_{JOM} = \text{atan}\left(\frac{R_{JM}}{R_{OM}}\right)$

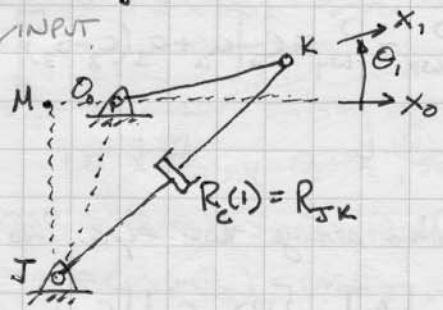
③ $\theta_{XOOJ} = \pi - \theta_{JOM}$

NOTE: these calcs should not be performed at every step: the values are constant → add to "R" matrix



④ $R_c(1) = \sqrt{R_{OJ}^2 + R_{OK}^2 - 2R_{OJ}R_{OK}\cos(\theta_{XOOJ} + \theta_1)}$

OUTPUT ← $R_c(1)$; θ_1 ← INPUT



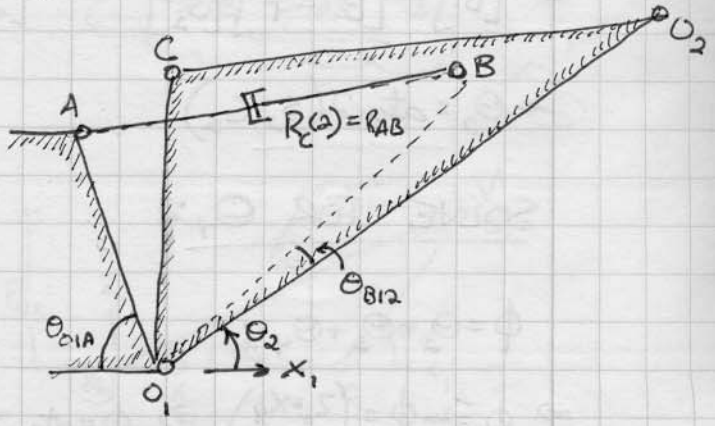
FIND REQUIRED BOOM CYLINDER LENGTH FROM θ_2:

⑤ $\theta_{A1x1} = \pi - \theta_{01A}$ } CONSTANT: add to "R" matrix

⑥ $\theta_{A1B} = \theta_{A1x1} - \theta_{B12} - \theta_2$ ← INPUT

⑦ $R_c(2) = \sqrt{R_{1A}^2 + R_{1B}^2 - 2R_{1A}R_{1B}\cos\theta_{A1B}}$

OUTPUT ← $R_c(2)$

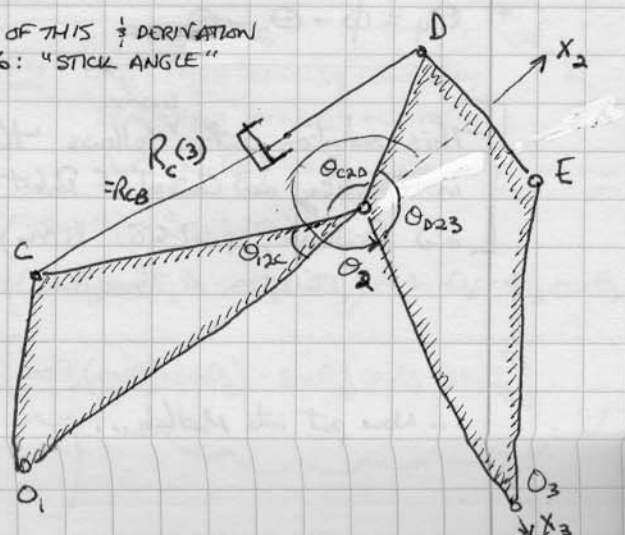


FIND REQUIRED STICK CYLINDER LENGTH FROM θ_3:

⑧ $\theta_{C2D} = 3\pi - \theta_{D23} - \theta_{12C} - \theta_3$ ← INPUT

← SEE INVERSE OF THIS ½ DERIVATION ON PAGE 76: "STICK ANGLE"

⑨ $R_c(3) = \sqrt{R_{2C}^2 + R_{2D}^2 - 2R_{2C}R_{2D}\cos\theta_{C2D}}$

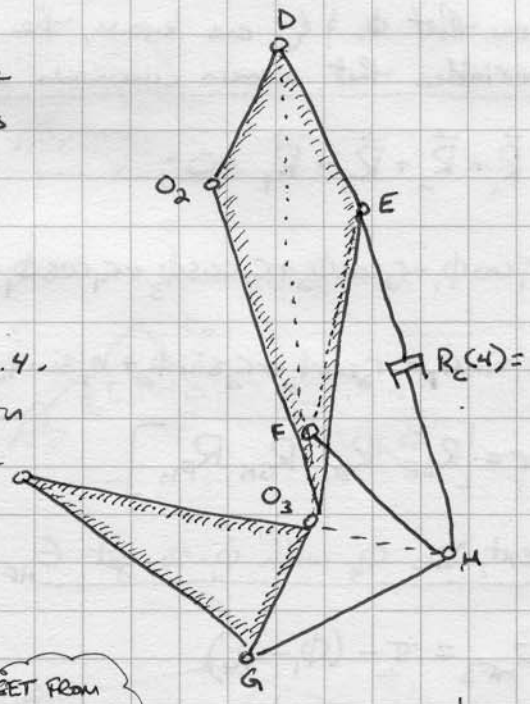


FIND REQUIRED BUCKET CYLINDER LENGTH FROM Θ_4:

To find the sublink configuration, solve the vector loop equation in x & y coordinates (bottom right sketch, this page)

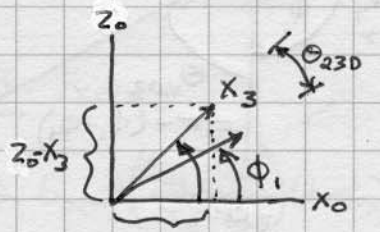
$$\vec{R}_1 + \vec{R}_2 + \vec{R}_3 + \vec{R}_4 = \emptyset ;$$

This will have 8 variables, r_i & ϕ_i, $i = 1 \ldots 4$. All four vector lengths r_i are fixed, known quantities. Angles ϕ_1 and ϕ_2 can be found from forward displacement analysis of the known joint orientations:



(10) $$J_{pos3} = \begin{bmatrix} X_0 \cdot X_3 & X_0 \cdot Y_3 & X_0 \cdot Z_3 & (P_{03})_x \\ Y_0 \cdot X_3 & Y_0 \cdot Y_3 & Y_0 \cdot Z_3 & (P_{03})_y \\ Z_0 \cdot X_3 & Z_0 \cdot Y_3 & Z_0 \cdot Z_3 & (P_{03})_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

← GET FROM "Jdisp" function

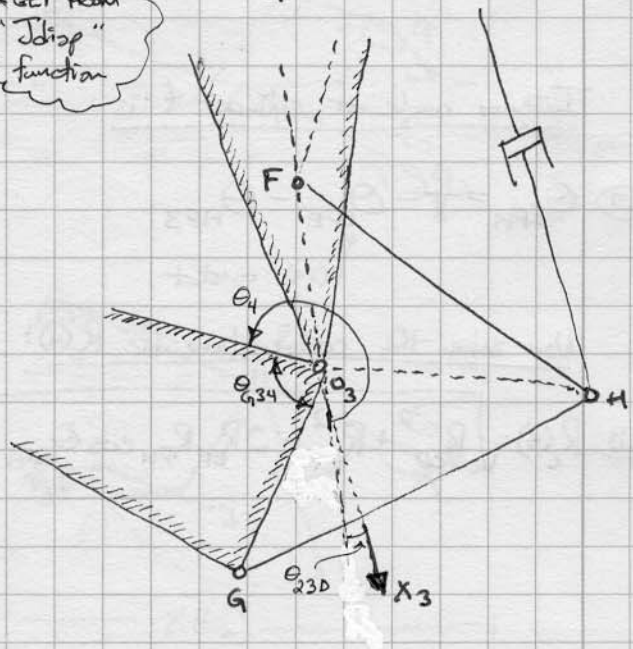


$$\phi_1 = \operatorname{atan}\left(\frac{X_0 \cdot X_3}{X_0 \cdot X_3}\right) - \Theta_{23D}$$

$X_0 \cdot X_3$ ← $J_{pos3}(3,1)$; $X_0 \cdot X_3$ ← $J_{pos3}(1,1)$

(12) $$\phi_1 = \operatorname{atan2}(J_{pos3}(3,1), J_{pos3}(1,1)) - \Theta_{23D}$$

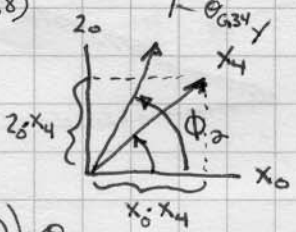
(11) $$\Theta_{23D} = \cos^{-1}\left(\frac{R_{23}^2 + R_{3D}^2 - R_{2D}^2}{2 R_{23} R_{3D}}\right)$$



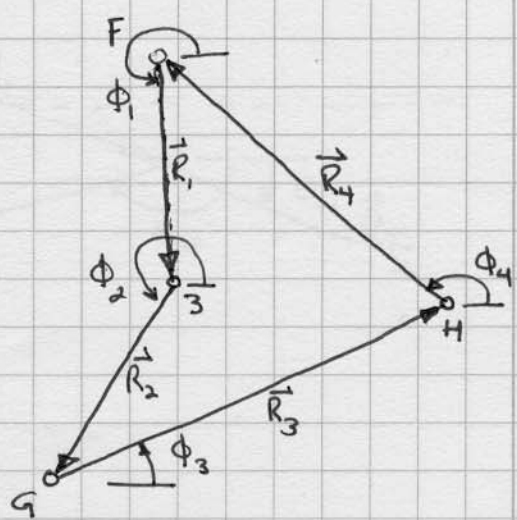
(13) $$J_{pos4} = \begin{bmatrix} X_0 \cdot X_4 & Y_0 \cdot Y_4 & X_0 \cdot Z_4 & (P_{04})_x \\ Y_0 \cdot X_4 & Y_0 \cdot Y_4 & Y_0 \cdot Z_4 & (P_{04})_y \\ Z_0 \cdot X_4 & Z_0 \cdot Y_4 & Z_0 \cdot Z_4 & (P_{04})_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\phi_2 = \operatorname{atan}\left(\frac{Z_0 \cdot X_4}{X_0 \cdot X_4}\right) + \Theta_{G34}$$

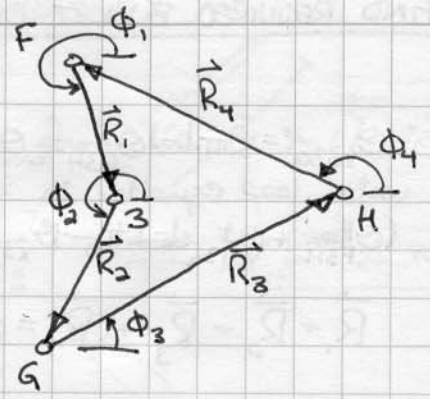
R(4,8) ← Θ_{G34}



(14) $$\phi_2 = \operatorname{atan2}(J_{pos4}(3,1), J_{pos4}(1,1)) + \Theta_{G34}$$



Now that ϕ_1 & ϕ_2 are known, the only variables that remain unknown are ϕ_3 & ϕ_4:



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

SCALAR EQNS:

(15) $r_1\cos\phi_1 + r_2\cos\phi_2 + r_3\cos\phi_3 + r_4\cos\phi_4 = 0$

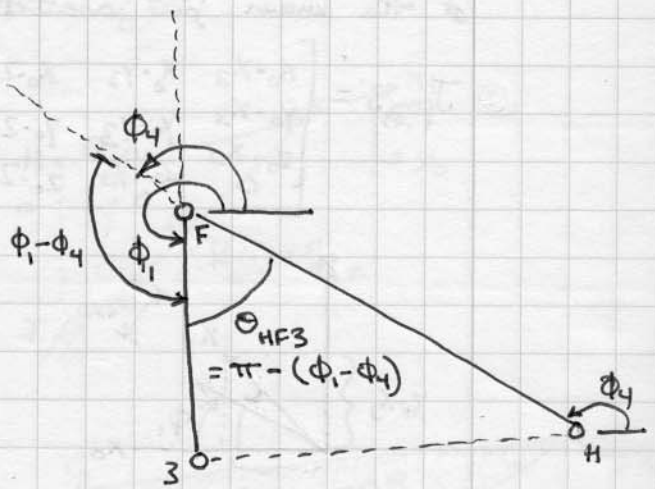
$r_1\sin\phi_1 + r_2\sin\phi_2 + r_3\sin\phi_3 + r_4\sin\phi_4 = 0$

} SOLVE w/ MATLAB "fminsearch" FUNCTION

$r = [R_{3F}\ R_{3G}\ R_{GH}\ R_{FH}]$

Next, use ϕ_4 with ϕ_1 to get θ_{HF3}:

(16) $\theta_{HF3} = \pi - (\phi_1 - \phi_4)$



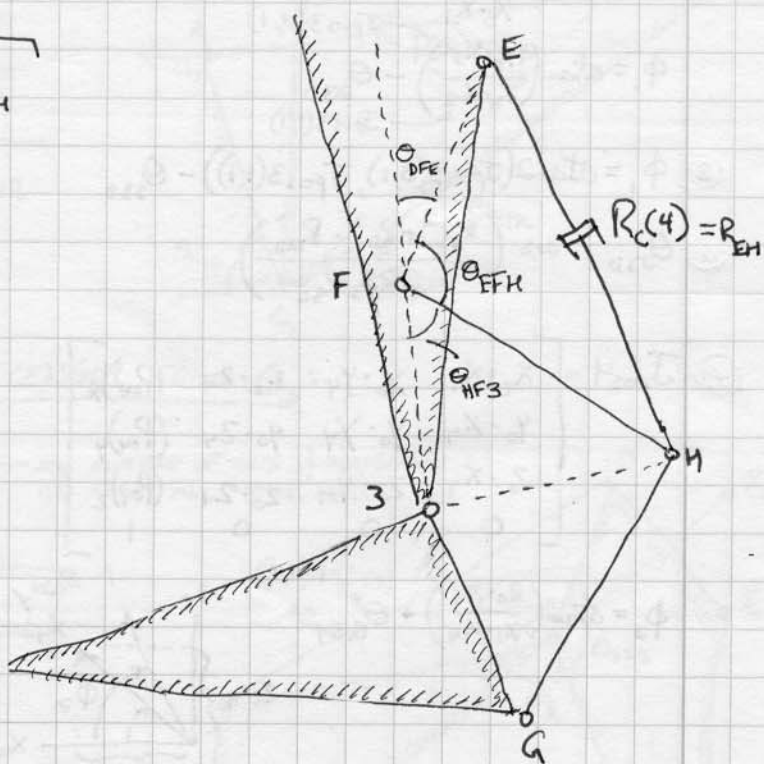
Interior angle of cylinder 4 is:

(17) $\theta_{EFH} = \pi - \theta_{DFE} - \theta_{HF3}$

θ_{DFE}: R(3,9) : constant

Now solve the cosine law for $R_c(4)$:

(18) $R_c(4) = \sqrt{R_{EF}^2 + R_{FH}^2 - 2R_{EF}R_{FH}\cos\theta_{EFH}}$

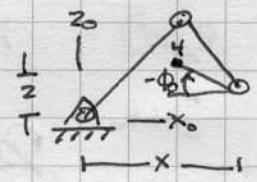


3-3-03

3-5-03 DIGGING SIMULATION: BUCKET TRAJECTORY

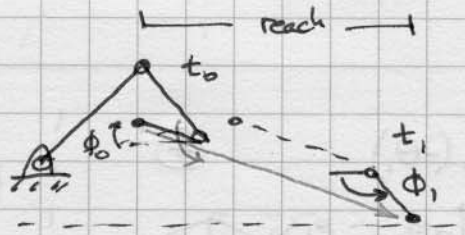
STEP 1: Start

$$START = [t_0 \; X_0 \; Y_0 \; Z_0 \; \phi_0]$$



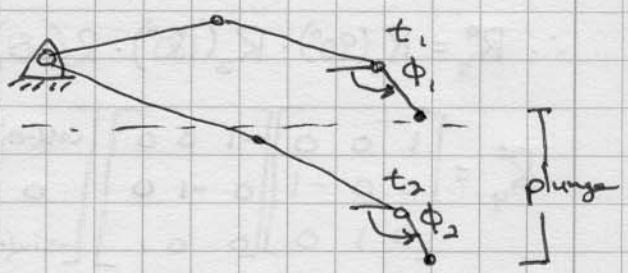
STEP 2: Reach

$$x(t) = X_0 + \left(\frac{reach - X_0}{t_1 - t_0}\right) t$$
$$z(t) = Z_0 + \left(\frac{Z_0 - ground}{t_1 - t_0}\right) t$$
$$\phi(t) = \phi_0 + \left(\frac{\phi_1 - \phi_0}{t_1 - t_0}\right) t$$
$$t_0 < t < t_1$$



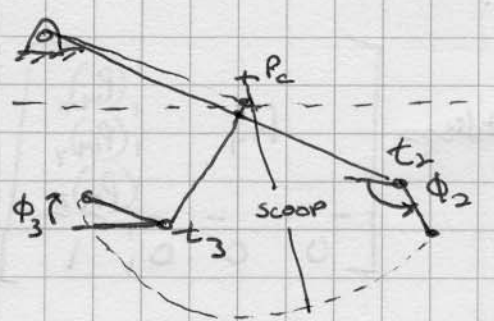
STEP 3: Plunge

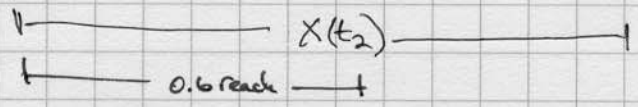
$$x(t) = x(t_1)$$
$$z(t) = -ground + \left(\frac{ground - plunge}{t_2 - t_1}\right)(t - t_1)$$
$$\phi(t) = \phi_1 + \left(\frac{\phi_2 - \phi_1}{t_2 - t_1}\right)(t - t_1)$$
$$t_1 < t < t_2$$

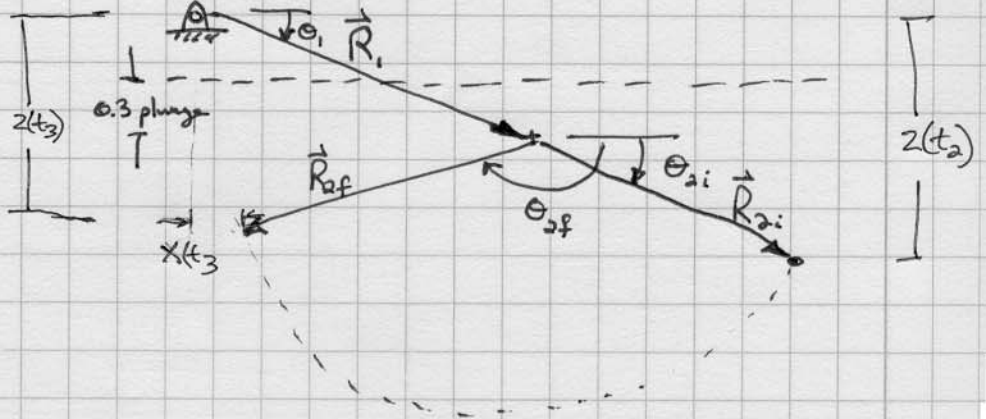


STEP 3: Scoop

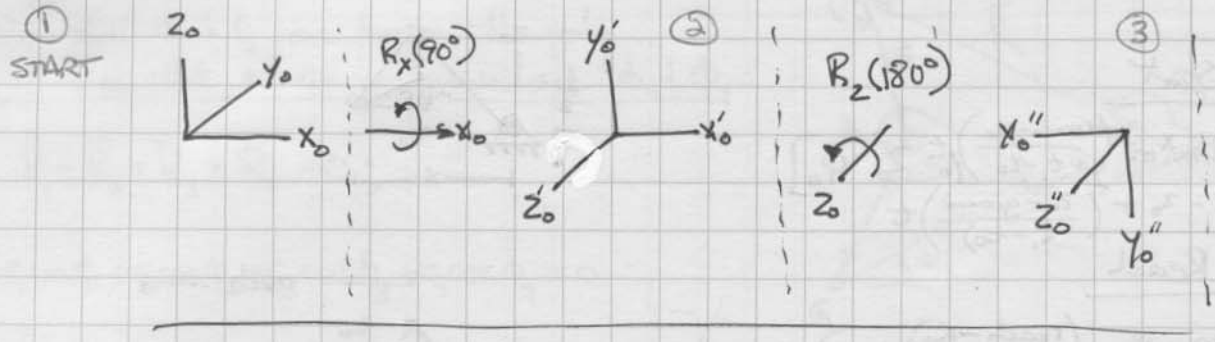
$$\theta_2(t) = \theta_{2i} + \left(\frac{\theta_{2f} - \theta_{2i}}{t_3 - t_2}\right)(t - t_2)$$
$$\theta_1 = \tan^{-1}\left(\frac{-0.3\,plunge}{0.6\,reach}\right) = const.$$
$$x(t) = r_1 \cos\theta_1 + r_2 \cos\theta_2(t)$$
$$z(t) = r_1 \sin\theta_1 + r_2 \sin\theta_2(t)$$
$$\phi(t) = \phi_2 + \left(\frac{\phi_3 - \phi_2}{t_3 - t_2}\right)(t - t_2)$$
$$t_2 < t < t_3$$

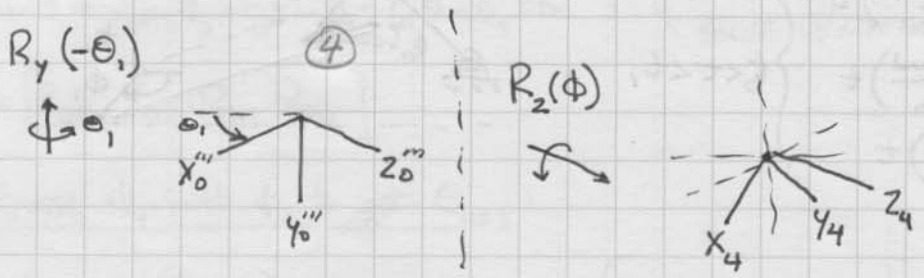






BUCKET DISPLACEMENT MATRIX IN BASE FRAME COORDINATES

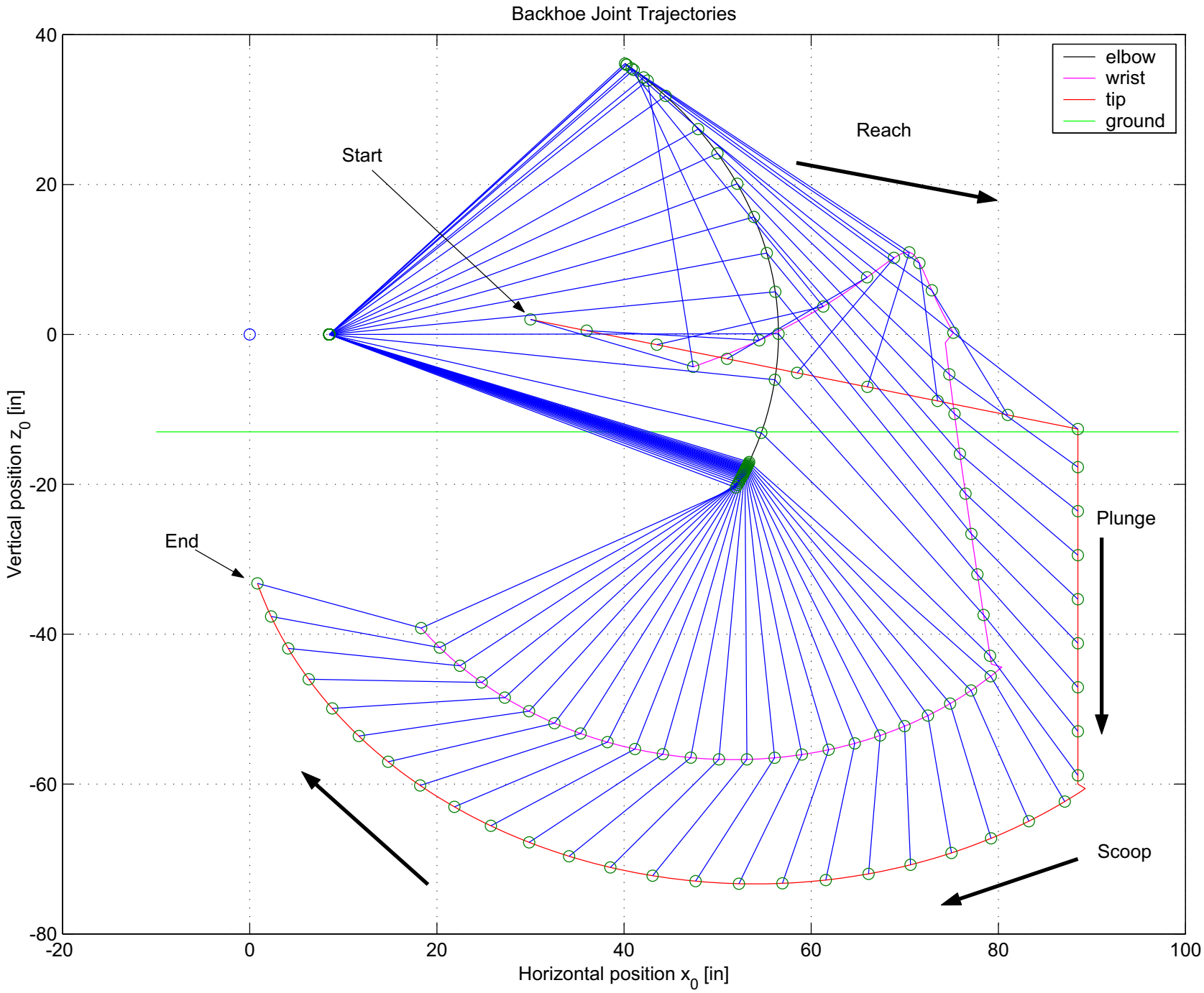


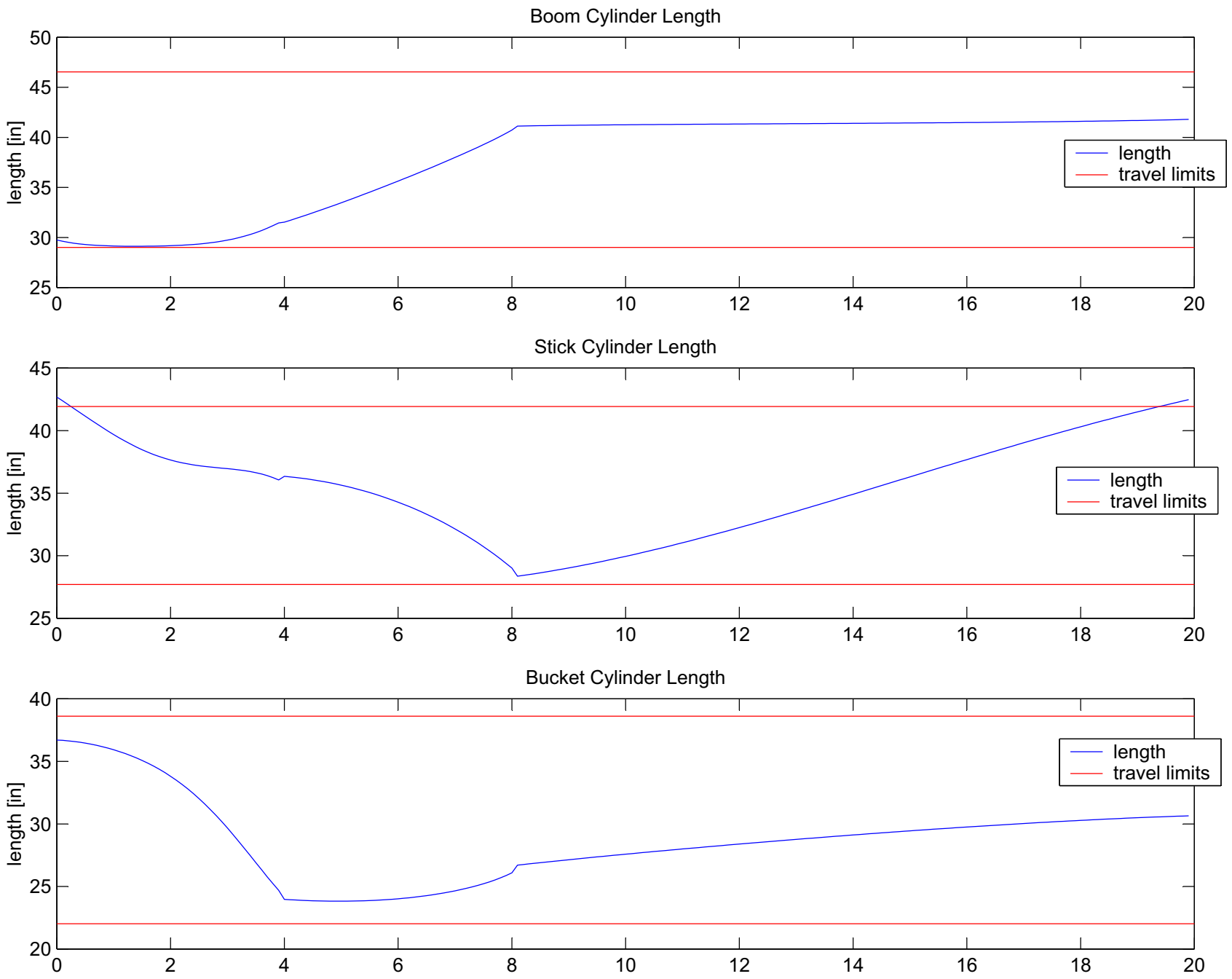


$$\therefore R_4^0 = R_x(90°) \cdot R_z(180°) \cdot R_y(-\theta_1) \cdot R_z(\phi)$$

$$R_4^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\theta_1) & 0 & \sin(-\theta_1) \\ 0 & 1 & 0 \\ -\sin(-\theta_1) & 0 & \cos(-\theta_1) \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Bucketdisp} = \left[\begin{array}{ccc|c} & R_4^0 & & \begin{matrix}(P_{04})_x \\ (P_{04})_y \\ (P_{04})_z\end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \Leftarrow \text{END-EFFECTOR DISPLACEMENT}$$


Backhoe Joint Trajectories
elbow
wrist
tip
ground
Start
End
Reach
Plunge
Scoop
Vertical position z_0 [in]
Horizontal position x_0 [in]
40
20
0
-20
-40
-60
-80
-20
0
20
40
60
80
100


Boom Cylinder Length
length [in]
50
45
40
35
30
25
0
2
4
6
8
10
12
14
16
18
20
length
travel limits
Stick Cylinder Length
length [in]
45
40
35
30
25
0
2
4
6
8
10
12
14
16
18
20
length
travel limits
Bucket Cylinder Length
length [in]
40
35
30
25
20
0
2
4
6
8
10
12
14
16
18
20
length
travel limits

```
%=========================================================================
%
%            BACKHOE DIGGING SIMULATION
%
%
%             Joe Frankel
%             Spring 2003
%
%
% This program simulates the digging motion of the backhoe and plots the
% bucket trajectory and corresponding cylinder positions
%
%=========================================================================

clear all; close all; clc;

% Define digging dimensions
reach=90;
plunge=60;
scoop=45;
N=200;

% Retreive desired 6xN twist trajectory
traj=dig(reach,plunge,scoop,N);

% Set up link dimensions
R=dimensions(1);
a=[R(1,1) R(2,2) R(3,3) R(4,1)];

% Cylinder limitations
Rclim=[7.5 29 27.6990 22.0242;  % <- minimums (Rc2min=30.7529in, calculated)
   16.6208 46.5434 41.9298 38.6144]; % <- maximums

% Loop through each set of data points
k=0;
for i=1:N

   % Compute Bucketdisp matrix from current values of traj
   % Note rotation matrices from base frame to e-e frame
   t(i)=traj(i,1);
   P04x=traj(i,2);
   P04y=traj(i,3);
   P04z=traj(i,4);
   phi=traj(i,5)*pi/180;
   T1=atan2(P04y,P04x);
   Rx90=[1 0 0;
      0 0 -1;
      0 1 0];
   Rz180=[-1 0 0;
      0 -1 0;
      0 0 1];
   RymT1=[cos(-T1) 0 sin(-T1);
      0 1 0;
      -sin(-T1) 0 cos(-T1)];
   Rzp=[cos(phi) -sin(phi) 0;
      sin(phi) cos(phi) 0;
      0 0 1];
   R04=Rx90*Rz180*RymT1*Rzp;
   P04=[P04x P04y P04z]';
   Bucketdisp=[R04 P04;
   0 0 0 1];

   % Solve for joint angles in the current configuration
   theta=Rdisp(Bucketdisp,a)*180/pi;

   % Generate Joint positions for the current configuration
   for j=2:4
      Jpos{j}=Jdisp(R,theta,j);
   end
   J2x(i)=Jpos{2}(1,4);
   J2z(i)=Jpos{2}(3,4);
   J3x(i)=Jpos{3}(1,4);
   J3z(i)=Jpos{3}(3,4);
   J4x(i)=Jpos{4}(1,4);
   J4z(i)=Jpos{4}(3,4);
```

```
   % Create a set of link positions every few points
   if mod(i,5)==0 | i==1
      k=k+1;
      Xpos{k}=[a(1) J2x(i) J3x(i) J4x(i)];
      Zpos{k}=[0 J2z(i) J3z(i) J4z(i)];
   end

   % Compute cylinder lengths in this configuration
   Rc=joint_to_cyl(R,theta);
   Rc2(i)=Rc(2);
   Rc3(i)=Rc(3);
   Rc4(i)=Rc(4);
end

% Plot the joint trajectories
plot(J2x,J2z,'k',J3x,J3z,'m',J4x,J4z,'r',[-10 max(J4x)+10],[-13 -13],'g')
title('Backhoe Joint Trajectories')
xlabel('Horizontal position x_0 [in]')
ylabel('Vertical position z_0 [in]')
hold on;grid on
plot(0,0,'O')
xlim=([-10 110]);
ylim=([-60 60]);
for h=1:k
   plot(Xpos{h},Zpos{h},Xpos{h},Zpos{h},'O')
end
legend('elbow','wrist','tip','ground')

% Plot the joint lengths
figure
subplot(3,1,1)
plot(t,Rc2,'b',[0 20],[Rclim(1,2) Rclim(1,2)],'r',...
   [0 20],[Rclim(2,2) Rclim(2,2)],'r')
ylabel('length [in]')
ylim=([Rclim(1,2)-10 Rclim(2,2)+10]);
title('Boom Cylinder Length')
legend('length','travel limits')

subplot(3,1,2)
plot(t,Rc3,'b',[0 20],[Rclim(1,3) Rclim(1,3)],'r',...
   [0 20],[Rclim(2,3) Rclim(2,3)],'r')
ylabel('length [in]')
ylim=([Rclim(1,3)-10 Rclim(2,3)+10]);
title('Stick Cylinder Length')
legend('length','travel limits')

subplot(3,1,3)
plot(t,Rc4,'b',[0 20],[Rclim(1,4) Rclim(1,4)],'r',...
   [0 20],[Rclim(2,4) Rclim(2,4)],'r')
ylabel('length [in]')
ylim=([Rclim(1,4)-10 Rclim(2,4)+10]);
title('Bucket Cylinder Length')
legend('length','travel limits')
```

```
function R=dimensions(a)

% ====================== LINK DIMENSIONS =========================
% All dimensions in inches. Note that only center-to-center joint
% distances ar required. See drawings for dimensions and point labels.

% ----Link 1----

% Dimension constants from length measurements
R01=8.5;
R0N=10.5;
RAN=7.5;
RJM=10; % <--??
R0M=6; % <--??
R0K=5.5; % <--??

% Calculated interior angle and dimension constants
R1A=sqrt(R0N^2+(R01-RAN)^2);
T01A=atan(R0N/(R01-RAN));

% ----Link 2 (Boom)----

% Dimension constants from length measurements
R1C=17.5;
R12=48;
R2C=36.25;
R1B=36;
R2B=12.35;

% Calculated interior angle and dimension constants
TB12=acos((R1B^2+R12^2-R2B^2)/(2*R1B*R12));
T12C=acos((R12^2+R2C^2-R1C^2)/(2*R12*R2C));

% ----Link 3 (Stick)----

% Dimension constants from length measurements
R2D=9+3/16;
RDE=13;
R23=38.5;
R2E=7+5/8;
R3D=47.25;
REF=30.77;
R3F=4.5;

% Calculated interior angle and dimension constants
RDF=R3D-R3F;
TD23=acos((R2D^2+R23^2-R3D^2)/(2*R2D*R23));
TDFE=acos((RDF^2+REF^2-RDE^2)/(2*RDF*REF));

% ----Link 4 (Bucket)----

% Dimension constants from length measurements
R34=18.5;
R3G=5.5;
R4G=20.01;

% Calculated interior angle and dimension constants
TG34=acos((R3G^2+R34^2-R4G^2)/(2*R3G*R34));

% ----Bucket sub-links----
RFH=9.5;
RGH=9.5;

% Place all dimensions and angles into matrix R
% --Columns 1-7 are lengths [in]
% --Columns 8-9 are angles [rad]
% --Row(i) contains dimensions for Link(i)

R=[R01 R0N RAN RJM R1A R0M R0K T01A 0;
   R1C R12 R2C R1B R2B 0 0 TB12 T12C;
   R2D RDE R23 R2E R3D REF RDF TD23 TDFE;
   R34 R3G R4G RFH RGH 0 0 TG34 0;];

% Output angle dimensions in degrees if desired
Rd=[R(:,1:7) R(:,8:9)*180/pi];
```

```
function theta=Rdisp(B,a)

% This function computes the reverse displacement analysis of the backhoe

% Solve for theta(1) directly from B matrix
theta(1)=atan(B(2,1)/B(1,1));

% Frame 0->1: Rotate +90 deg about x, then theta(1) about y
Rx90=[1 0 0;0 0 -1;0 1 0];
Ryt1=[cos(theta(1)) -sin(theta(1)) 0;
   sin(theta(1)) cos(theta(1)) 0;
   0 0 1];
R01=Rx90*Ryt1;

% Frame 1 wrt frame 0
p001=a(1)*[cos(theta(1)) sin(theta(1)) 0]';
p101=R01'*p001;

% Frame 4 wrt frame 3 and 1
p104=R01'*[B(1,4) B(2,4) B(3,4)]';
p034=a(4)*[B(1,1) B(2,1) B(3,1)]';
p134=R01'*p034;
p114=R01'*[B(1,4) B(2,4) B(3,4)]'-R01'*p001;

% Frame 3 wrt frame 1
p113=p114-p134;

% Solve triangle 123 for theta(3)
pWx=p113(1);
pWy=p113(2);
c3=(pWx^2+pWy^2-a(2)^2-a(3)^2)/(2*a(2)*a(3));
s3=-sqrt(1-c3^2);
theta(3)=atan2(s3,c3);

% Use results of last step to set up system of 2 eqns and solve for theta(2)
A=pWx+pWy;
B2=a(2)+a(3)*(c3+s3);
C=a(2)+a(3)*(c3-s3);
D=pWx-pWy;
E=C;
F=B2;
G=inv([B2 C;E -F])*[A D]';
c2=G(1);
s2=G(2);
theta(2)=atan2(s2,c2);

% Solve for theta(4)
phi=atan2(B(3,1),B(1,1));
theta(4)=phi-theta(2)-theta(3);
```

```
function Jpos=Jdisp(R,theta,j)

% This function computes end-effector displacement

theta=theta*pi/180;
alpha=[pi/2 0 0 0];
d=[0 0 0 0];
a=[R(1,1) R(2,2) R(3,3) R(4,1)];

% Compute Homogeneous Transformation Matrices
Jpos=eye(4);
for i=1:j
   st=sin(theta(i));
   ct=cos(theta(i));
   sa=sin(alpha(i));
   ca=cos(alpha(i));
   A{i}=[ct -st*ca st*sa a(i)*ct;
      st ct*ca -ct*sa a(i)*st;
      0 sa ca d(i);
      0 0 0 1];
   Jpos=Jpos*A{i};
end
```

```
function Rc=joint_to_cyl(R,theta)

% This function computes the required cylinder lengths for a given set of joint angles

% Extract relevant dimensions from the 4x9 R matrix in the sublink area
%
%  R=[R01 R0N RAN RJM R1A R0M R0K T01A 0;
%     R1C R12 R2C R1B R2B 0 0 TB12 T12C;
%     R2D RDE R23 R2E R3D REF RDF TD23 TDFE;
%     R34 R3G R4G RFH RGH 0 0 TG34 0;];
%

%1.
R0M=R(1,6);
RJM=R(1,4);
R0K=R(1,7);

%2.
T01A=R(1,8);
TB12=R(2,8);
R1A=R(1,5);
R1B=R(2,4);

%3.
TD23=R(3,8);
T12C=R(2,9);
R2C=R(2,3);
R2D=R(3,1);

%4.
R23=R(3,3);
R3D=R(3,5);
TG34=R(4,8);
R3F=R(3,5)-R(3,7);
R3G=R(4,2);
RGH=R(4,5);
RFH=R(4,4);
TDFE=R(3,9);
REF=R(3,6);

theta=theta*pi/180;

% ========= PART 1: Compute Swing Cylinder Length Rc(1) from theta(1) ========

R0J=sqrt(R0M^2+RJM^2);
TJ0M=atan(RJM/R0M);
Tx00J=pi-TJ0M;
Rc(1)=sqrt(R0J^2+R0K^2-2*R0J*R0K*cos(Tx00J+theta(1)));

% ========= PART 2: Compute Boom Cylinder Length Rc(2) from theta(2) ========

TA1x1=pi-T01A;
TA1B=TA1x1-TB12-theta(2);
Rc(2)=sqrt(R1A^2+R1B^2-2*R1A*R1B*cos(TA1B));

% ========= PART 3: Compute Stick Cylinder Length Rc(3) from theta(3) ========

TC2D=3*pi-TD23-T12C-theta(3);
Rc(3)=sqrt(R2C^2+R2D^2-2*R2C*R2D*cos(TC2D));

% ========= PART 4: Compute Bucket Cylinder Length Rc(4) from theta(4) ========

% Compute the two known vector angles from forward transformation matrices
Jpos3=Jdisp(R,theta*180/pi,3);
T23D=acos((R23^2+R3D^2-R2D^2)/(2*R23*R3D));
phi1=atan2(Jpos3(3,1),Jpos3(1,1))-T23D;

Jpos4=Jdisp(R,theta*180/pi,4);
phi2=atan2(Jpos4(3,1),Jpos4(1,1))+TG34;

% Enter the vector lengths from known quantities in the R matrix
r=[R3F R3G RGH RFH];

% Use the Matlab 'fminsearch' function to find the 2 unknown angles in the vector
% loop equation R1+R2+R3+R4=0 (see sketch)
```

```
guess=[pi/4 pi];
phi34=fminsearch('sublinkpos',guess,optimset('display','off'),r,phi1,phi2);
phi4=phi34(2);

% Compute upper sublink interior angle
THF3=-pi-phi1+phi4;

% Compute cylinder 4 interior angle
TEFH=pi-THF3-TDFE;

% Compute cylinder 4 length by solving cosine law
Rc(4)=sqrt(REF^2+RFH^2-2*REF*RFH*cos(TEFH));
```

```
function theta=cyl_to_joint(Rc,R)

% This function computes the joint angles of the backhoe based on
% cylinder positions measurements and link dimensions

% Extract necessary dimensions from R matrix
%
%  R=[R01 R0N RAN RJM R1A R0M R0K T01A 0;
%     R1C R12 R2C R1B R2B 0 0 TB12 T12C;
%     R2D RDE R23 R2E R3D REF RDF TD23 TDFE;
%     R34 R3G R4G RFH RGH 0 0 TG34 0;];

R1A=R(1,5);
R1B=R(2,4);
T01A=R(1,8);
TB12=R(2,8);
R2C=R(2,3);
R2D=R(3,1);
T12C=R(2,9);
TD23=R(3,8);
REF=R(3,6);
RFH=R(4,4);
TDFE=R(3,9);
R3F=R(3,5)-R(3,7);
R3G=R(4,2);
RGH=R(4,5);
TG34=R(4,8);
R23=R(3,3);
R3D=R(3,5);

% Expand cylinder lengths from Rc vector
RJK=Rc(1);
RAB=Rc(2);
RCD=Rc(3);
REH=Rc(4);

% 1. Compute theta(1) from Swing Cylinder measurement Rc(1)
theta(1)=0;

% 2. Compute theta(2) from Boom Cylinder measurement Rc(2)
TA1B=acos((R1A^2+R1B^2-RAB^2)/(2*R1A*R1B));
theta(2)=pi-T01A-TA1B-TB12;

% 3. Compute theta(3) from Stick Cylinder measurement Rc(3)
TC2D=acos((R2C^2+R2D^2-RCD^2)/(2*R2C*R2D));
theta(3)=3*pi-T12C-TC2D-TD23;

% 4. Compute theta(4) from Bucket Cylinder measurement Rc(4)
TEFH=acos((REF^2+RFH^2-REH^2)/(2*REF*RFH));
THF3=pi-TDFE-TEFH;
R3H=sqrt(R3F^2+RFH^2-2*R3F*RFH*cos(THF3));
TF3H=acos((R3F^2+R3H^2-RFH^2)/(2*R3F*R3H));
TH3G=acos((R3H^2+R3G^2-RGH^2)/(2*R3H*R3G));
T23D=acos((R23^2+R3D^2-R2D^2)/(2*R23*R3D));
theta(4)=3*pi-TF3H-TH3G-TG34-T23D;

% Verification section: use to check computed angles w/drawings

%TA1B=TA1B*180/pi
%Theta2=theta(2)*180/pi

%TC2D=TC2D*180/pi
%Theta3=theta(3)*180/pi

%TEFH=TEFH*180/pi
%TDFE=TDFE*180/pi
%THF3=THF3*180/pi
%TF3H=TF3H*180/pi
%TH3G=TH3G*180/pi
%Theta4=theta(4)*180/pi
```

```
% This program computes cylinder positions from bucket tip position and angle

clear all; close all; clc;

% Enter the displacement info here

P04x=0;
P04y=0;
P04z=-103;
phi=100*pi/180;

% Retrieve dimensions
R=dimensions(1);
a=[R(1,1) R(2,2) R(3,3) R(4,1)];

% Compute e-e displacement matrix
% Note: (x0 y0 z0) dot y4 and (x0 y0 z0) dot z4 not needed
T1=atan2(P04y,P04x);
Rx90=[1 0 0;
   0 0 -1;
   0 1 0];
Rz180=[-1 0 0;
   0 -1 0;
   0 0 1];
RymT1=[cos(-T1) 0 sin(-T1)
   0 1 0;
   -sin(-T1) 0 cos(-T1)];
Rzp=[cos(phi) -sin(phi) 0;
   sin(phi) cos(phi) 0;
   0 0 1];
R04=Rx90*Rz180*RymT1*Rzp;
P04=[P04x P04y P04z]';
Bucketdisp=[R04 P04;
   0 0 0 1];

% Compute joint angles
theta=Rdisp(Bucketdisp,a)*180/pi;

% Compute cylinder lengths
Rc=joint_to_cyl(R,theta)
```

```
function J=Jacobian(R,theta)

% This function computes the Jacobian matrix for the backhoe
% based on the link dimensions and joint angles

% Extract link lengths from R matrix
a1=R(1,1);
a2=R(2,2);
a3=R(3,3);

% Define joint angles
t1=theta(1);
t2=theta(2);
t3=theta(3);

% Unit vectors
z0=[0 0 1]';
z1=z0;
z2=z0;
z3=z0;

% Define vectors from base frame origin to origins 1-3
p0=[0 0 0]';
p1=[a1*cos(t1) a1*sin(t1) 0]';
p2=[a1*cos(t1)+a2*cos(t2)*cos(t1) a1*sin(t1)+a2*cos(t2)*sin(t1) a2*sin(t2)]';
p3=[a1*cos(t1)+a2*cos(t2)*cos(t1)+a3*cos(t2+t3)*cos(t1);
   a1*sin(t1)+a2*cos(t2)*sin(t1)+a3*cos(t2+t3)*sin(t1);
   a2*sin(t2)+a3*sin(t2+t3)];

% Choose O3 as end-effector
p=p3;

% Compute Jacobian
J=[cross(z0,p-p0) cross(z1,p-p1) cross(z2,p-p2) cross(z3,p-p3);
   z0 z1 z2 z3];
```

```
function traj=dig(reach,plunge,scoop,N)

%=========================================================================================
% This function computes an Nx5 matrix representing bucket displacement
% throughout the digging motion
%
% traj=[t(1) x(1) y(1) z(1) phi(1);
%       t(2) x(2) y(2) z(2) phi(2);
%         .    .    .     .    .   ;
%         .    .    .     .    .   ;
%         .    .    .     .    .   ;
%       t(N) x(N) y(N) z(N) phi(N);
%
% Where t is time, x,y,z are the position of the bucket tip relative to the
% base frame, and phi is the angle of the bucket wrt the -x0 axis
%
% Digging motion is defined by three intervals:
%
% 1. reach -- extend e-e on +x0 direction by distance 'reach'
% 2. plunge -- bring e-e down into soil below base frame origin by distance 'plunge'
% 3. scoop -- draw e-e back toward tractor in an arc with radius 'scoop'
%
%=========================================================================================

% Set up the initial configuration and initialize the output matrix
start=[0 30 0 2 -20];
traj=[start;zeros(N-1,5)];
x(1)=start(2);
y(1)=start(3);
z(1)=start(4);
phi(1)=start(5);

% Define the desired angles at the end of each step
phi1=140;
phi2=120;
phi3=-20;

% Set up the parameters for the scooping interval
T1=atan((-0.3*plunge)/(0.6*reach));
r1=0.6*reach/cos(T1);

T2i=atan2(-0.7*plunge,0.4*reach);
r2=(0.4*reach)/cos(T2i);

T2f=-165*pi/180;

% Define the time intervals
tf=20;
t1=0.2*tf;
t2=0.4*tf;
dt=tf/N;
t(1)=0;

% Loop through the time steps and create displacement values at each step
for i=2:N
   t(i)=t(i-1)+dt;

   % First interval: reach
   if t(i)<t1
      x(i)=x(1)+((reach-x(1))/t1)*t(i);
      y(i)=y(1);
      z(i)=z(1)-((13+z(1))/t1)*t(i);
      phi(i)=phi(1)+((phi1-phi(1))/t1)*t(i);

   % Second interval: plunge
   elseif t1<t(i) & t(i)<t2
      x(i)=x(i-1);
      y(i)=y(i-1);
      z(i)=-13+((13-plunge)/(t2-t1))*(t(i)-t1);
      phi(i)=phi1+((phi2-phi1)/(t2-t1))*(t(i)-t1);

   % Third interval: scoop
   else
      T2=T2i+((T2f-T2i)/(tf-t2))*(t(i)-t2);
      x(i)=r1*cos(T1)+r2*cos(T2);
```

```
      y(i)=y(i-1);
      z(i)=r1*sin(T1)+r2*sin(T2);
      phi(i)=phi2+((phi3-phi2)/(tf-t2))*(t(i)-t2);
   end

   % Place configuration values into output matrix
   traj(i,1)=t(i);
   traj(i,2)=x(i);
   traj(i,3)=y(i);
   traj(i,4)=z(i);
   traj(i,5)=phi(i);

end

% Plot e-e trajectory
%plot(x,z,x,-13)
%xlim([-10 max(x)+10])
```