# EEEN30150: STATIC EQUATIONS

*Dr Paul F. Curran*

**Conor Igoe**

13360786

# Contents

## Declaration

I declare that the work described in this report was done by the person named above, and that the description and comments in this report are my own work, except where otherwise acknowledged. I understand the definition of plagiarism. I have read and understand the consequences of plagiarism as discussed in the School Policy on Plagiarism, the UCD Plagiarism Policy and the UCD Briefing Document on Academic Integrity and Plagiarism.

Signed: . . . . . .

Date: 09 / 03 / 2017 . . . . . .

# Primary Problem 2: Jansen Mechanical Linkage

## Deriving the System of Equations

For the Cartesian geometric system shown in the reference diagram in Figure 1, the following non-linear equations can be derived relating the angles $\theta_1$ through $\theta_8$, using the independent variable and constants as described in Table 1. Note that throughout this assignment, all angles are defined with respect to the horizontal, and are expressed in radians. Note also that a positive angle indicates counter-clockwise rotation from the horizontal.

$$l_i \sin(\theta_i) + l_1 \sin(\theta_1) = -a + l_2 \sin(\theta_2) \tag{1}$$

$$l_i \cos(\theta_i) + l_1 \cos(\theta_1) = -b + l_2 \cos(\theta_2) \tag{2}$$

$$l_i \sin(\theta_i) + l_1 \sin(\theta_1) + l_3 \sin(\theta_3) = -a + l_4 \sin(\theta_4) \tag{3}$$

$$l_i \cos(\theta_i) + l_1 \cos(\theta_1) + l_3 \cos(\theta_3) = -b + l_4 \cos(\theta_4) \tag{4}$$

$$l_i \sin(\theta_i) + l_6 \sin(\theta_6) = -a + l_7 \sin(\theta_7) \tag{5}$$

$$l_i \cos(\theta_i) + l_6 \cos(\theta_6) = -b + l_7 \cos(\theta_7) \tag{6}$$

$$l_i \sin(\theta_i) + l_1 \sin(\theta_1) + l_3 \sin(\theta_3) + l_5 \sin(\theta_5) = -a + l_7 \sin(\theta_7) + l_8 \sin(\theta_8) \tag{7}$$

$$l_i \cos(\theta_i) + l_1 \cos(\theta_1) + l_3 \cos(\theta_3) + l_5 \cos(\theta_5) = -b + l_7 \cos(\theta_7) + l_8 \cos(\theta_8) \tag{8}$$

Table 1: Constants and Independent Variables

| Notation | Value | Description |
|---|---|---|
| $\theta_i$ | Independent | Crank Angle |
| $l_i$ | 15 | Length of link connecting joints 0 and A |
| $l_1$ | 50 | Length of link connecting joints A and B |
| $l_2$ | 41.5 | Length of link connecting joints 1 and B |
| $l_3$ | 55.8 | Length of link connecting joints B and C |
| $l_4$ | 40.1 | Length of link connecting joints 1 and C |
| $l_5$ | 39.4 | Length of link connecting joints C and E |
| $l_6$ | 61.9 | Length of link connecting joints A and D |
| $l_7$ | 39.3 | Length of link connecting joints 1 and D |
| $l_8$ | 36.7 | Length of link connecting joints E and D |
| $a$ | 7.8 | Vertical distance from joint 1 to joint 0 |
| $b$ | 38 | Horizontal distance from joint 1 to joint 0 |

Using one of the two known fixed joints in Figure 1, from elementary trigonometry we can express the vertical distance from this chosen fixed joint to the position of any other joint in terms of the lengths and angles of various links. Given that position in a Cartesian coordinate system is conservative, we can equate

this vertical distance to the vertical distance found through any other choice of links that ends at the same joint. This results in an expression relating various angles in the Jansen linkage mechanism. Given that orthogonal components of a Cartesian co-ordinate system are by definition independent, the same approach can be applied to horizontal distances, obtaining another relation for the same angles. Equation (1) shows this approach in practice: the vertical distance from joint 0 to joint B via joint A is equated to the vertical distance of the "path" via joint 1. This results in a non-linear relation involving the two unknowns $\theta_1$ and $\theta_2$. The same analysis using horizontal distances results in another relation involving $\theta_1$ and $\theta_2$, shown in Equation (2). This approach can be applied to multiple pairs of "paths", building a system of 8 non-linear relations involving all 8 angles $\theta_1$ through $\theta_8$.

Taking all terms to the left hand side in Equations (1) though (8), we obtain a system of 8 independent non-linear equations in 8 unknowns, as described in Equation (9).

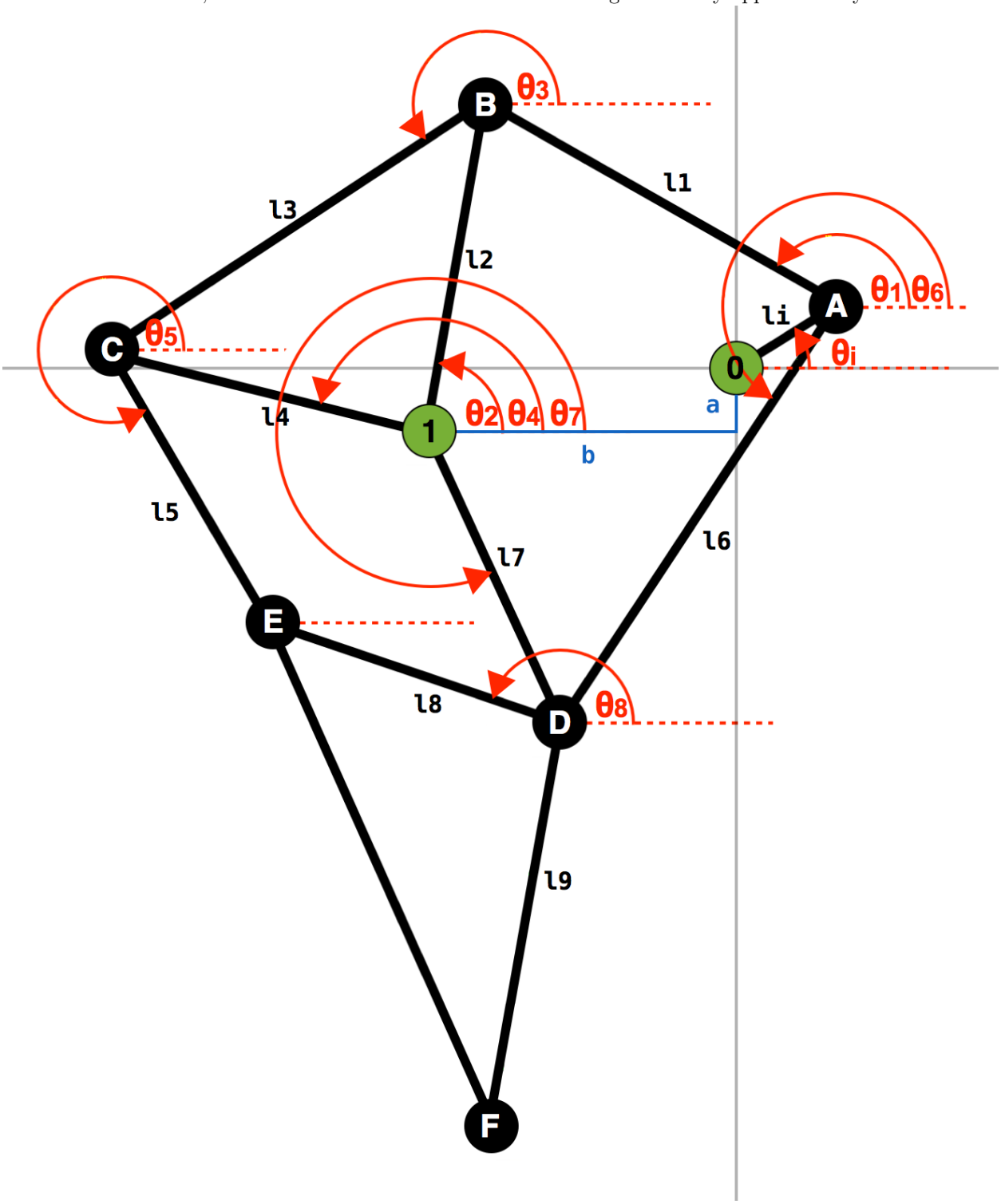
$$\mathbf{x} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \end{bmatrix}$$

$$f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \\ f_4(\mathbf{x}) \\ f_5(\mathbf{x}) \\ f_6(\mathbf{x}) \\ f_7(\mathbf{x}) \\ f_8(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} l_i \sin(\theta_i) + l_1 \sin(\theta_1) + a - l_2 \sin(\theta_2) \\ l_i \cos(\theta_i) + l_1 \cos(\theta_1) + b - l_2 \cos(\theta_2) \\ l_i \sin(\theta_i) + l_1 \sin(\theta_1) + l_3 \sin(\theta_3) + a - l_4 \sin(\theta_4) \\ l_i \cos(\theta_i) + l_1 \cos(\theta_1) + l_3 \cos(\theta_3) + b - l_4 \cos(\theta_4) \\ l_i \sin(\theta_i) + l_6 \sin(\theta_6) + a - l_7 \sin(\theta_7) \\ l_i \cos(\theta_i) + l_6 \cos(\theta_6) + b - l_7 \cos(\theta_7) \\ l_i \sin(\theta_i) + l_1 \sin(\theta_1) + l_3 \sin(\theta_3) + l_5 \sin(\theta_5) + a - l_7 \sin(\theta_7) - l_8 \sin(\theta_8) \\ l_i \cos(\theta_i) + l_1 \cos(\theta_1) + l_3 \cos(\theta_3) + l_5 \cos(\theta_5) + b - l_7 \cos(\theta_7) - l_8 \cos(\theta_8) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (9)$$

Using an iterative method with a sufficiently accurate initial estimate, it is possible to approximate the values for which the vector of unknowns $\mathbf{x}$ satisfies our system of equations. Equation (10) shows the multi-dimensional Newton-Raphson method used in this assignment to approximate such values. Note that the subscript $n$ in $\mathbf{x}_n$ denotes the estimate arising from the $n$th iteration of the iterative method when using an independently derived vector of initial estimates $\mathbf{x}_0$. Note, too, that $Df(\mathbf{x}_n)$ denotes the Jacobian of the matrix $f(\mathbf{x}_n)$.

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (Df(\mathbf{x}_n))^{-1} f(\mathbf{x}_n) \quad (10)$$

Figure 1: Reference Diagram for the Jansen Linkage Mechanism. Joints coloured green are fixed. Note that joint 0 is defined to be at the origin of our coordinate system. As this diagram is intended to serve as a reference for notation, units on axis lines are omitted and the diagram is only approximately to scale.



## Determining the Jacobian Matrix for the Multi-Dimensional Newton Raphson-Method

In order to use the Multi-Dimensional Newton-Raphson method, equation (10) requires that we calculate the Jacobian of matrix $f(\mathbf{x})$, shown in equation (11).

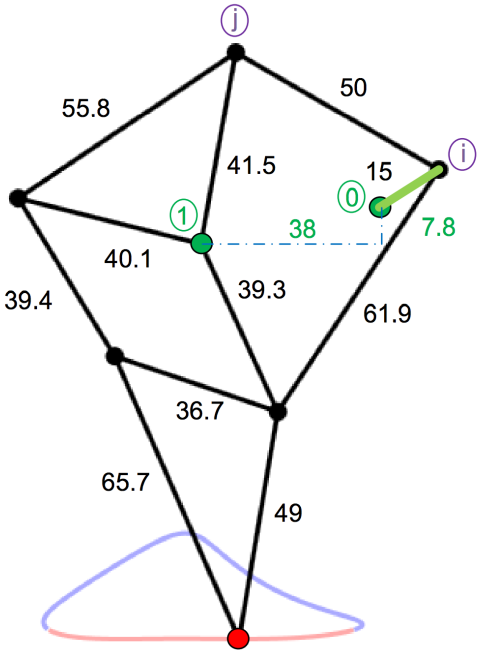
$$Df(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \theta_3} & \frac{\partial f_1}{\partial \theta_4} & \frac{\partial f_1}{\partial \theta_5} & \frac{\partial f_1}{\partial \theta_6} & \frac{\partial f_1}{\partial \theta_7} & \frac{\partial f_1}{\partial \theta_8} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \theta_3} & \frac{\partial f_2}{\partial \theta_4} & \frac{\partial f_2}{\partial \theta_5} & \frac{\partial f_2}{\partial \theta_6} & \frac{\partial f_2}{\partial \theta_7} & \frac{\partial f_2}{\partial \theta_8} \\ \frac{\partial f_3}{\partial \theta_1} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial \theta_3} & \frac{\partial f_3}{\partial \theta_4} & \frac{\partial f_3}{\partial \theta_5} & \frac{\partial f_3}{\partial \theta_6} & \frac{\partial f_3}{\partial \theta_7} & \frac{\partial f_3}{\partial \theta_8} \\ \frac{\partial f_4}{\partial \theta_1} & \frac{\partial f_4}{\partial \theta_2} & \frac{\partial f_4}{\partial \theta_3} & \frac{\partial f_4}{\partial \theta_4} & \frac{\partial f_4}{\partial \theta_5} & \frac{\partial f_4}{\partial \theta_6} & \frac{\partial f_4}{\partial \theta_7} & \frac{\partial f_4}{\partial \theta_8} \\ \frac{\partial f_5}{\partial \theta_1} & \frac{\partial f_5}{\partial \theta_2} & \frac{\partial f_5}{\partial \theta_3} & \frac{\partial f_5}{\partial \theta_4} & \frac{\partial f_5}{\partial \theta_5} & \frac{\partial f_5}{\partial \theta_6} & \frac{\partial f_5}{\partial \theta_7} & \frac{\partial f_5}{\partial \theta_8} \\ \frac{\partial f_6}{\partial \theta_1} & \frac{\partial f_6}{\partial \theta_2} & \frac{\partial f_6}{\partial \theta_3} & \frac{\partial f_6}{\partial \theta_4} & \frac{\partial f_6}{\partial \theta_5} & \frac{\partial f_6}{\partial \theta_6} & \frac{\partial f_6}{\partial \theta_7} & \frac{\partial f_6}{\partial \theta_8} \\ \frac{\partial f_7}{\partial \theta_1} & \frac{\partial f_7}{\partial \theta_2} & \frac{\partial f_7}{\partial \theta_3} & \frac{\partial f_7}{\partial \theta_4} & \frac{\partial f_7}{\partial \theta_5} & \frac{\partial f_7}{\partial \theta_6} & \frac{\partial f_7}{\partial \theta_7} & \frac{\partial f_7}{\partial \theta_8} \\ \frac{\partial f_8}{\partial \theta_1} & \frac{\partial f_8}{\partial \theta_2} & \frac{\partial f_8}{\partial \theta_3} & \frac{\partial f_8}{\partial \theta_4} & \frac{\partial f_8}{\partial \theta_5} & \frac{\partial f_8}{\partial \theta_6} & \frac{\partial f_8}{\partial \theta_7} & \frac{\partial f_8}{\partial \theta_8} \end{bmatrix} =$$

$$\begin{bmatrix} l_1\cos(\theta_1)\ , & -l_2\cos(\theta_2)\ , & 0\ , & 0\ , & 0\ , & 0\ , & 0\ , & 0 \\ -l_1\sin(\theta_1)\ , & l_2\sin(\theta_2)\ , & 0\ , & 0\ , & 0\ , & 0\ , & 0\ , & 0 \\ l_1\cos(\theta_1)\ , & 0\ , & l_3\cos(\theta_3)\ , & -l_4\cos(\theta_4)\ , & 0\ , & 0\ , & 0\ , & 0 \\ -l_1\sin(\theta_1)\ , & 0\ , & -l_3\sin(\theta_3)\ , & l_4\sin(\theta_4)\ , & 0\ , & 0\ , & 0\ , & 0 \\ 0\ , & 0\ , & 0\ , & 0\ , & 0\ , & l_6\cos(\theta_6)\ , & -l_7\cos(\theta_7)\ , & 0 \\ 0\ , & 0\ , & 0\ , & 0\ , & 0\ , & -l_6\sin(\theta_6)\ , & l_7\sin(\theta_7)\ , & 0 \\ l_1\cos(\theta_1)\ , & 0\ , & l_3\cos(\theta_3)\ , & 0\ , & l_5\cos(\theta_5)\ , & 0\ , & -l_7\cos(\theta_7)\ , & -l_8\cos(\theta_8) \\ -l_1\sin(\theta_1)\ , & 0\ , & -l_3\sin(\theta_3)\ , & 0\ , & -l_5\sin(\theta_5)\ , & 0\ , & l_7\sin(\theta_7)\ , & l_8\sin(\theta_8) \end{bmatrix} \tag{11}$$

## Determining the Initial Estimates for the Netwon-Raphson Method for the First Crank Angle

A crude analysis of the dimensions of the diagram in Figure 2 — taken from from the assignment brief — confirms that the dimensions of the linkages in the given image are roughly to scale. Using the ruler tool in the digital photo editing software Photoshop, and accounting for the constant scaling factor between the linear scale of pixels in Photoshop measurements and the linear scale of the linkage dimensions supplied in the brief, the lengths of the linkages in the given image were determined to match closely to the numerical dimensions. This constant scaling factor was determined from Equation (12). Table 2 shows the actual and scaled measurements of the lengths in the given image obtained using the above method, the numerical dimensions from Table 1, and a % difference as determined using Equation (13). From the results in Table 2, with percentage differences in the range of $\pm 2\%$, it is evident that the included image in the assignment brief is indeed roughly to scale. This analysis served as justification for the use of the given image as a sufficiently accurate source for initial estimates.[1]

Figure 2: Given diagram from assignment brief.




$$k = \frac{1}{N} \sum_{i=0}^{N-1} \frac{x_i}{y_i} \tag{12}$$

where $k$ is the Photoshop Scaling Factor
$N$ is the number of linkage lengths
$x_i$ is the given numerical value for the $i$th linkage length (unitless)
$y_i$ is the measured value for the $i$th linkage length in Photoshop (pixels)

[1] It was assumed that a % difference in the range of $\pm 2\%$ was sufficiently small for the purposes of obtaining initial estimates, though, admittedly, no analytical basis was established a priori for this assumption.

Table 2: Measurements & Comparisons of linkage lengths in Figure 2

| Linkage | Numerical Value ($x_i$) | Measurement ($y_i$) | Scaled Measurement ($ky_i$) | % Difference ($d_i$) |
|---|---|---|---|---|
| $l_i$ | 15 | 155.9 | 14.64 | 2.4% |
| $l_1$ | 50 | 537.94 | 50.52 | -1.03% |
| $l_2$ | 41.5 | 435.41 | 40.89 | 1.5% |
| $l_3$ | 55.8 | 589.10 | 55.32 | 0.9% |
| $l_4$ | 40.1 | 429.29 | 40.31 | -0.5% |
| $l_5$ | 39.4 | 417.14 | 39.17 | 0.6% |
| $l_6$ | 61.9 | 656.77 | 61.68 | 0.4% |
| $l_7$ | 39.3 | 424.65 | 39.88 | -1.5% |
| $l_8$ | 36.7 | 401.93 | 37.74 | -2.8% |

$$d_i = \frac{x_i - ky_i}{x_i} \times 100\% \tag{13}$$

where $d_i$ is the percentage difference between the given numerical dimension and the scaled measurement for the $i$th linkage

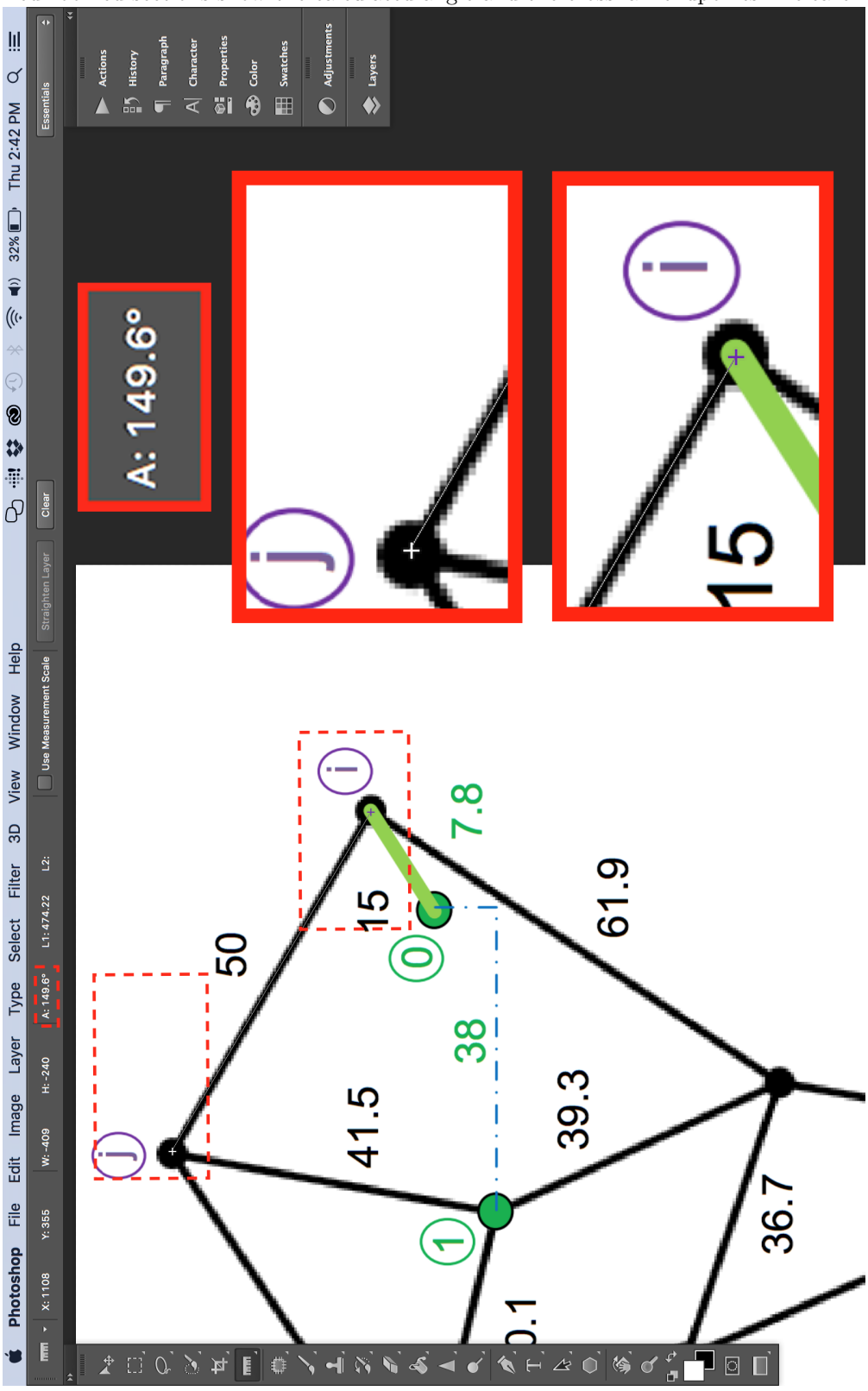
Using the image in Figure 2 once again with the ruler tool in Photoshop, the angles $\theta_1$ through $\theta_8$ were measured for the given crank angle in the image of $\theta_i \approx 1.57$ rad. Figure 3 shows a screenshot of the software in use whilst measuring angle $\theta_1$. Table 3 lists the measured angle values obtained using this approach. Note that Photoshop's angle units are in degrees. To convert to radians, the relation in Equation (14) was used.

$$1^\circ = \frac{2\pi}{360} \text{ rad} \tag{14}$$

Table 3: Measurements of angles of joints in Figure 2

| Angle | Measurement (rad) |
|---|---|
| $\theta_i$ | 1.57 |
| $\theta_1$ | 2.62 |
| $\theta_2$ | 1.40 |
| $\theta_3$ | 3.73 |
| $\theta_4$ | 2.90 |
| $\theta_5$ | 5.26 |
| $\theta_6$ | 4.12 |
| $\theta_7$ | 5.14 |
| $\theta_8$ | 2.81 |

Figure 3: Screenshot of Photoshop in use determining $\theta_1$ from Figure 2. Note that the ruler tool calculates the distance (in pixels) and angle (in degrees) between two points in the image, indicated by the crosshair icons. The red zoomed sections show the calculated angle and the crosshair endpoints in clearer detail



## Determining the Roots of the System of Equations for the first Crank Angle

Using an initial estimate $\mathbf{x}_0$ for the roots of $f(\mathbf{x})$ for a given crank angle $\theta_i$, we can use the multi-dimensional Newton-Raphson iterative method in Equation (10) to obtain the next estimate of the roots, $\mathbf{x}_1$. This process is repeated $n$ times, using estimate $\mathbf{x}_n$ to obtain estimate $\mathbf{x}_{n+1}$. The process terminates when the absolute value of each function in our system of functions $f(\mathbf{x})$ evaluated at the latest estimate of the root is sufficiently close to 0, or when the algorithm has iterated a pre-defined maximum number of times.

Listing 1 shows the code used to define a MATLAB function to implement this algorithm. Each root approximation was calculated using a somewhat arbitrary[2] tolerance of $10^{-5}$ and iteration limit of 10.

Listing 1: MATLAB function-M file describing a function to determine an approximation for the angles $\theta_1$ through $\theta_8$ using the Newton-Raphson iterative method in Equation (10) with the system of equations $f(\mathbf{x})$ in Equation (9) and the Jacobian matrix $Df(\mathbf{x})$ in Equation (11)

```
function [t1, t2, t3, t4, t5, t6, t7, t8] = Jansen_Newton_Raphson(ti, t1_0, t2_0,
    t3_0, t4_0, t5_0, t6_0, t7_0, t8_0)
%[t1, t2, t3, t4, t5, t6, t7, t8] = JANSEN_NEWTON_RAPHSON(ti, t1_0, t2_0, t3_0,
    t4_0, t5_0, t6_0, t7_0, t8_0):
%function to approximate the vector of unknowns X for the Jansen Linkage Mechanism
%in problem 2 of MP1 using the multi-dimensional Newton-Raphson method with
%initial estimates X_0 and crank angle ti.
%
%Input ti = crank angle (rad)
%Input t1_0 = initial estimate angle theta_1 (rad)
%Input t2_0 = initial estimate angle theta_2 (rad)
%Input t3_0 = initial estimate angle theta_3 (rad)
%Input t4_0 = initial estimate angle theta_4 (rad)
%Input t5_0 = initial estimate angle theta_5 (rad)
%Input t6_0 = initial estimate angle theta_6 (rad)
%Input t7_0 = initial estimate angle theta_7 (rad)
%Input t8_0 = initial estimate angle theta_8 (rad)
%Output t1 = approximate angle theta_1 (rad)
%Output t2 = approximate angle theta_2 (rad)
%Output t3 = approximate angle theta_3 (rad)
%Output t4 = approximate angle theta_4 (rad)
%Output t5 = approximate angle theta_5 (rad)
%Output t6 = approximate angle theta_6 (rad)
%Output t7 = approximate angle theta_7 (rad)
%Output t8 = approximate angle theta_8 (rad)

%   Version 1: created 09/03/2017. Author: Conor Igoe
%   This MATLAB function M-file is not flexible. It works for the Jansen Linkage
%   mechansim in problem 2 of MP1 only.
%
%   The limit on the number of allowable iterations and the acceptable
%   tolerance for the Newton-Raphson (NR) algorithm are internally
%   generated.
```

[2] As we have not established the nature (number of roots, critical points etc.) of each of the functions in $f(\mathbf{x})$ in this report, we cannot comment on the rate of convergence. It was found, empirically, that an iteration limit of 10 was more than sufficient to ensure convergence for the chosen tolerance level and initial conditions. It was noted that a lower iteration limit would be sufficient for more accurate initial estimates, as in the case of all sets of root approximations following the first during pseudo-static analysis (see the following subsection).

```
% ------------------------------------------------------------------------

% Check input and output arguments
if (nargin ~= 9), error('Incorrect number of input arguments.'); end
if (nargout ~= 8), error('Incorrect number of output arguments.'); end

% ------------------------------------------------------------------------

% Internal parameter ITERATION_LIMIT = maximum number of steps permitted.
% Internal parameter TOLERANCE = minimum acceptable value for |f_dash(x)|
%                                evaluated at the approximate roots.
% Internal parameters li, l1, l2, l3, l4, l5, l6, l7, l8 = linkage lengths (
    unitless)
% Internal parameter a = fixed joint vertical separation
% Internal parameter b = fixed joint horizontal separation

ITERATION_LIMIT = 10;
TOLERANCE = 10^-5;

li = 15;
l1 = 50;
l2 = 41.5;
l3 = 55.8;
l4 = 40.1;
l5 = 39.4;
l6 = 61.9;
l7 = 39.3;
l8 = 36.7;

a = 7.8;
b = 38;

% ------------------------------------------------------------------------

% System of equations to be used in NR
f1 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*sin(ti) + l1*sin(t1) + a - l2*sin(
    t2) );
f2 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*cos(ti) + l1*cos(t1) + b - l2*cos(
    t2) );
f3 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*sin(ti) + l1*sin(t1) + l3*sin(t3)
    + a - l4*sin(t4) );
f4 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*cos(ti) + l1*cos(t1) + l3*cos(t3)
    + b - l4*cos(t4) );
f5 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*sin(ti) + l6*sin(t6) + a - l7*sin(
    t7) );
f6 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*cos(ti) + l6*cos(t6) + b - l7*cos(
    t7) );
f7 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*sin(ti) + l1*sin(t1) + l3*sin(t3)
    + l5*sin(t5) + a - l7*sin(t7) - l8*sin(t8) );
f8 = @(ti, t1, t2, t3, t4, t5, t6, t7, t8) ( li*cos(ti) + l1*cos(t1) + l3*cos(t3)
    + l5*cos(t5) + b - l7*cos(t7) - l8*cos(t8) );

% Jacobian of non-linear system of equations to be used in NR
```

```
d_f1_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l1*cos(t1) );
d_f1_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l2*cos(t2) );
d_f1_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f1_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f1_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f1_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f1_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f1_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );

d_f2_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l1*sin(t1) );
d_f2_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l2*sin(t2) );
d_f2_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f2_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f2_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f2_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f2_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f2_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );

d_f3_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l1*cos(t1) );
d_f3_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f3_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l3*cos(t3) );
d_f3_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l4*cos(t4) );
d_f3_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f3_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f3_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f3_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );

d_f4_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l1*sin(t1) );
d_f4_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f4_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l3*sin(t3) );
d_f4_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l4*sin(t4) );
d_f4_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f4_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f4_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f4_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );

d_f5_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f5_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f5_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f5_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f5_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f5_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l6*cos(t6) );
d_f5_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l7*cos(t7) );
d_f5_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );

d_f6_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f6_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f6_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f6_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f6_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f6_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l6*sin(t6) );
d_f6_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l7*sin(t7) );
d_f6_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
```

```
130
d_f7_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l1*cos(t1) );
d_f7_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f7_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l3*cos(t3) );
d_f7_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
135 d_f7_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l5*cos(t5) );
d_f7_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f7_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l7*cos(t7) );
d_f7_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l8*cos(t8) );

140 d_f8_t1 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l1*sin(t1) );
d_f8_t2 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f8_t3 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l3*sin(t3) );
d_f8_t4 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f8_t5 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( -l5*sin(t5) );
145 d_f8_t6 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( 0 );
d_f8_t7 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l7*sin(t7) );
d_f8_t8 = @(t1, t2, t3, t4, t5, t6, t7, t8) ( l8*sin(t8) );

% ------------------------------------------------------------------------

150
X = [t1_0; t2_0; t3_0; t4_0; t5_0; t6_0; t7_0; t8_0];       % Initialize initial
    guess
iterations = 0;                                             % Initialize
    iterations counter

for count = 1:ITERATION_LIMIT + 1
155     if count == ITERATION_LIMIT + 1
            error('The Iteration limit has been reached and did not converge')
            break
        end

160     % Update root estimate
        t1 = X(1);
        t2 = X(2);
        t3 = X(3);
        t4 = X(4);
165     t5 = X(5);
        t6 = X(6);
        t7 = X(7);
        t8 = X(8);

170     % Check if we have met the desired tolerance
        if abs(f1(ti, t1, t2, t3, t4, t5, t6, t7, t8)) < TOLERANCE && abs(f2(ti, t1,
            t2, t3, t4, t5, t6, t7, t8)) < TOLERANCE && abs(f3(ti, t1, t2, t3, t4, t5,
             t6, t7, t8)) < TOLERANCE && abs(f4(ti, t1, t2, t3, t4, t5, t6, t7, t8)) <
             TOLERANCE && abs(f5(ti, t1, t2, t3, t4, t5, t6, t7, t8)) < TOLERANCE &&
            abs(f6(ti, t1, t2, t3, t4, t5, t6, t7, t8)) < TOLERANCE && abs(f7(ti, t1,
            t2, t3, t4, t5, t6, t7, t8)) < TOLERANCE && abs(f8(ti, t1, t2, t3, t4, t5,
             t6, t7, t8)) < TOLERANCE
            break
        end
```

```
    % Calculate f(x)
    f = [      f1(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f2(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f3(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f4(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f5(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f6(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f7(ti, t1, t2, t3, t4, t5, t6, t7, t8);
             f8(ti, t1, t2, t3, t4, t5, t6, t7, t8)];

    % Calculate D_f_dash(x)
    D_f = [    [    d_f1_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f1_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
                [   d_f2_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f2_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
                [   d_f3_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f3_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
                [   d_f4_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f4_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
                [   d_f5_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f5_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
                [   d_f6_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                   d_f6_t2(t1, t2, t3, t4, t5, t6, t7, t8)
```

```
                    d_f6_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f6_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f6_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f6_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f6_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f6_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
185                 [   d_f7_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f7_t8(t1, t2, t3, t4, t5, t6, t7, t8)];
                [   d_f8_t1(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t2(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t3(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t4(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t5(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t6(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t7(t1, t2, t3, t4, t5, t6, t7, t8)
                    d_f8_t8(t1, t2, t3, t4, t5, t6, t7, t8)]];

    % Calculate next approximation
    X = X - (D_f)\f;
190
    iterations = iterations + 1;
end
end
```

## Determining the Roots of the System of Equations for Successive Crank Angles

Under the assumption that a small change in the crank angle gives rise to sufficiently small changes in angles $\theta_1$ through $\theta_8$, then it is valid to use the approximations computed before the change in crank angle as the initial estimates for the Newton-Raphson method. For a fixed tolerance, this assumption enforces a maximum step size in changes in crank angle as the crank angle is incremented to trace the path of joint F during pseudo-static analysis.

## Determining the Position of Joints for a Given Crank Angle

Using the vector of initial estimates $\mathbf{x}_0$ for a given crank angle $\theta_i$ with the Multi-Dimensional Newton-Raphson algorithm, the resulting angle approximations $\theta_1$ through $\theta_8$ can be used to determine the position of joints through elementary trigonometry. Equations (15) through (20) describe the the position of joints A though F in terms of angles $\theta_i$, $\theta_1$ through $\theta_8$ and constants $a$, $b$ and $c$. In the following set of equations, $x$ and $y$ denote horizontal and vertical positions, respectively. The subscripts of $x$ and $y$ denote the specific joint for which the position describes.

$$x_A = l_i \cos\theta_i \qquad y_A = l_i \sin\theta_i \tag{15}$$

$$x_B = l_1 \cos\theta_1 + x_A \qquad y_B = l_1 \sin\theta_1 + y_A \tag{16}$$

$$x_C = l_3 \cos\theta_3 + x_B \qquad y_C = l_3 \sin\theta_3 + y_B \tag{17}$$

$$x_D = -38 + l_7 \cos\theta_7 \qquad y_D = -7.8 + l_7 \sin\theta_7 \tag{18}$$

$$x_E = l_8 \cos\theta_8 + x_D \qquad y_E = l_8 \sin\theta_8 + y_D \tag{19}$$

$$x_F = l_9 \cos(\theta_8 + c) + x_D \qquad y_F = l_9 \sin(\theta_8 + c) + y_D \tag{20}$$

where $l_9 = 49$ denotes the length of the link joining joint D and F

Note that the constant angle $c$ in the equations in Equation (20) was defined to be the interior angle at joint D between links $l_8$ and $l_9$. The angle can be easily determined using the cosine rule, shown in Equation (21).

$$l_{10}^2 = l_8^2 + l_9^2 - 2l_8 l_9 \cos(c)$$

$$c = \arccos\left(\frac{l_8^2 + l_9^2 - l_{10}^2}{2l_8 l_9}\right) = 1.729556 \text{ rad} \tag{21}$$

where $l_{10} = 65.7$ denotes the length of the link joining joint E and F

Listing 2 shows a Matlab function-M file that calculates the position of joints A through F for given angles $\theta_i$ and $\theta_1$ through $\theta_8$.

Listing 2: MATLAB function-M file to calculate position of joints from a given set of angles

```
function [positions] = find_joint_positions(angles)
%[positions] = FIND_JOINT_POSITIONS(angles):
%function to calculate the position of joints A through F for given angles
%
%Input angles = [ti t1 t2 t3 t4 t5 t6 t7 t8] given angles of Linkage system (rad)
%Output positions = [[JointA_x JointA_y]
%                    [JointB_x JointB_y]
%                    [JointC_x JointC_y]
%                    [JointD_x JointD_y]
%                    [JointE_x JointE_y]
%                    [JointF_x JointF_y]
%                    [Joint0_x Joint0_y]
%                    [Joint1_x Joint1_y]] joint positions (unitless)

%   Version 1: created 09/03/2017. Author: Conor Igoe
%   This MATLAB function M-file is not flexible. It works for the Jansen
%   Linkage mechansim in problem 2 of MP1 only.

% ------------------------------------------------------------------------

% Check input and output arguments
if (nargin ~= 1), error('Incorrect number of input arguments.'); end
if (nargout ~= 1), error('Incorrect number of output arguments.'); end

% ------------------------------------------------------------------------

% Internal parameters li, l1, l2, l3, l4, l5, l6, l7, l8, l0 = linkage lengths (
    unitless)
% Internal parameter a = fixed joint vertical separation (unitless)
% Internal parameter b = fixed joint horizontal separation (unitless)
% Internal parameter c = fixed internal angle EDF (rad)
```

```
li = 15;
l1 = 50;
l2 = 41.5;
l3 = 55.8;
l4 = 40.1;
l5 = 39.4;
l6 = 61.9;
l7 = 39.3;
l8 = 36.7;
l9 = 49;

a = 7.8;
b = 38;

c = 1.729556;

% ------------------------------------------------------------------------

% Extract angles
ti = angles(1);
t1 = angles(2);
t2 = angles(3);
t3 = angles(4);
t4 = angles(5);
t5 = angles(6);
t6 = angles(7);
t7 = angles(8);
t8 = angles(9);

% Joint A
positions(1, 1)  = li*cos(ti);
positions(1, 2)  = li*sin(ti);

% Joint B
positions(2, 1)  = l1*cos(t1) + positions(1, 1);
positions(2, 2)  = l1*sin(t1) + positions(1, 2);

% Joint C
positions(3, 1)  = l3*cos(t3) + positions(2, 1);
positions(3, 2)  = l3*sin(t3) + positions(2, 2);

% Joint D
positions(4, 1)  = -b + l7*cos(t7);
positions(4, 2)  = -a + l7*sin(t7);

% Joint E
positions(5, 1)  = l8*cos(t8) + positions(4, 1);
positions(5, 2)  = l8*sin(t8) + positions(4, 2);

% Joint F
positions(6, 1)  = l9*cos(t8 + c) + positions(4, 1);
positions(6, 2)  = l9*sin(t8 + c) + positions(4, 2);
```

```
% Joint 0
positions(7, 1) = 0;
positions(7, 2) = 0;

% Joint 1
positions(8, 1) = -b;
positions(8, 2) = -a;

end
```

## Visualization

To visualize the Jansen linkage mechanism during simulation, the MATLAB function-M file in Listing 3 was used. On a high level, this code:

1. Increments the crank angle from a given starting angle using a give step size
2. For each of these crank angles, determines the corresponding approximate angles $\theta_1$ through $\theta_8$ by calling `Jansen_Newton_Raphson()`
3. Finds the positions of all joints from the approximated angles by calling `find_joint_positions()`
4. Adds the latest position of joint P to the matrix keeping track of the trace of joint P
5. Plots the history of positions of point P using the `line()` function
6. Draws faces, joints and links of the Jansen mechanism using the `line()` and `patch()` functions

Listing 3: MATLAB function-M file to render a 2D visualization of the Jansen linkage mechanism

```
function [positions, trace] = plot_jansen(ti, t_estimates,
    input_angle_step_size, cycles)
%[positions] = PLOT_JANSEN(angles):
%function to plot the Jansen Mechanism and trace of joint P from a starting
    crank angle over a number
%of crank angles, as determined by the step size and number of cycles
%
%Input ti = starting crank angle (rad)
%Input t_estimates = [t1_0 t2_0 t3_0 t4_0 t5_0 t6_0 t7_0 t8_0] theta angle
    initial estimates (rad)
%Input input_angle_step_size = step size (rad)
%Input cycles = number of desired cycles to simulate pseudo-statically
%Output positions = [[JointA_x JointA_y]
%                    [JointB_x JointB_y]
%                    [JointC_x JointC_y]
%                    [JointD_x JointD_y]
%                    [JointE_x JointE_y]
%                    [JointF_x JointF_y]
%                    [Joint0_x Joint0_y]
%                    [Joint1_x Joint1_y]] final positions for joints A:F , 0 &
     1

%   Version 1: created 09/03/2017. Author: Conor Igoe
%   This MATLAB function M-file is not flexible. It works for the Jansen
```

```
%   Linkage mechanism in problem 2 of MP1 only.

% ------------------------------------------------------------------------

% Check input and output arguments
% Argument check
if (nargin ~= 5), error('Incorrect number of input arguments.'); end
if (nargout ~= 2), error('Incorrect number of output arguments.'); end

% Input check
if (cycles <= 0), error('Cycles must be positive'); end
if (input_angle_step_size <= 0), error('Step size must be positive'); end

% Close all open figures
close all

% Assign initial estimates
t1_0 = t_estimates(1);
t2_0 = t_estimates(2);
t3_0 = t_estimates(3);
t4_0 = t_estimates(4);
t5_0 = t_estimates(5);
t6_0 = t_estimates(6);
t7_0 = t_estimates(7);
t8_0 = t_estimates(8);

% Preallocate trace matrix
trace = zeros(2, floor(2*cycles*pi/input_angle_step_size));

% Counter for trace indexing
count = 1;

% ------------------------------------------------------------------------

% Loop through input angles,
for tinput = ti:input_angle_step_size:ti + 2*cycles*pi + input_angle_step_size
    % Calculate angles using NR
    [t1, t2, t3, t4, t5, t6, t7, t8] = Jansen_Newton_Raphson(tinput, t1_0,
        t2_0, t3_0, t4_0, t5_0, t6_0, t7_0, t8_0);
    angles = [tinput, t1, t2, t3, t4, t5, t6, t7, t8];

    % Find positions of nodes from angles
    positions = find_joint_positions(angles);

    jointA = [positions(1,1) , positions(1,2)];
    jointB = [positions(2,1) , positions(2,2)];
    jointC = [positions(3,1) , positions(3,2)];
    jointD = [positions(4,1) , positions(4,2)];
    jointE = [positions(5,1) , positions(5,2)];
    jointF = [positions(6,1) , positions(6,2)];
    joint0 = [positions(7,1) , positions(7,2)];
    joint1 = [positions(8,1) , positions(8,2)];
```

```
    % In-loop plotting; clear previous plot; set axis for correct scaling
    drawnow
    clf
    axis([-115 115 -115 115])

    % Plot trace
    trace(1,count) = jointF(1);
    trace(2,count) = jointF(2);
    line(trace(1,1:count), trace(2,1:count));

    patch('Faces', [1 2 3], 'Vertices', [joint1; jointB; jointC])
    patch('Faces', [1 2 3], 'Vertices', [jointD; jointE; jointF])

    line([joint0(1) jointA(1)], [joint0(2) jointA(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40, 'MarkerEdgeColor', [0
         0 0])
    line([jointA(1) jointB(1)], [jointA(2) jointB(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([jointB(1) jointC(1)], [jointB(2) jointC(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([jointE(1) jointD(1)], [jointE(2) jointD(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([jointA(1) jointD(1)], [jointA(2) jointD(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([jointB(1) joint1(1)], [jointB(2) joint1(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([joint1(1) jointD(1)], [joint1(2) jointD(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([joint1(1) jointC(1)], [joint1(2) jointC(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40, 'MarkerEdgeColor', [0
         0 0])
    line([jointE(1) jointF(1)], [jointE(2) jointF(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([jointD(1) jointF(1)], [jointD(2) jointF(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)
    line([jointC(1) jointE(1)], [jointC(2) jointE(2)], 'Color',[0.5 0.5 0.5],
        'LineWidth', 3, 'Marker', '.', 'MarkerSize', 40)

    % Update initial estimates for NR for next input angles
    t1_0 = t1;
    t2_0 = t2;
    t3_0 = t3;
    t4_0 = t4;
    t5_0 = t5;
    t6_0 = t6;
    t7_0 = t7;
    t8_0 = t8;

    % Increment counter for trace indexing
    count = count + 1;
end
end
```

## Analysis

It was noted that, depending on the required accuracy, it is possible to replace the system of 8 equations with 4 systems of 2 equations, and solve for the unknown angles $\theta_1$ through $\theta_8$ sequentially. The major caveat with such an approach, of course, is that such a system no longer mathematically represents the combined system of 8 equations. To illustrate this, consider the following 4 systems of 2 equations in 2 unknowns:

$$g^a(\theta_1,\theta_2)=\begin{bmatrix}g_1(\theta_1,\theta_2)\\g_2(\theta_1,\theta_2)\end{bmatrix}=\begin{bmatrix}l_i\sin(\theta_i)+l_1\sin(\theta_1)+a-l_2\sin(\theta_2)\\\\l_i\cos(\theta_i)+l_1\cos(\theta_1)+b-l_2\cos(\theta_2)\end{bmatrix} \tag{22}$$

$$g^b(\theta_3,\theta_4)=\begin{bmatrix}g_3(\theta_3,\theta_4)\\g_4(\theta_3,\theta_4)\end{bmatrix}=\begin{bmatrix}l_i\sin(\theta_i)+l_1\sin(\theta_1)+l_3\sin(\theta_3)+a-l_4\sin(\theta_4)\\\\l_i\cos(\theta_i)+l_1\cos(\theta_1)+l_3\cos(\theta_3)+a-l_4\cos(\theta_4)\end{bmatrix} \tag{23}$$

$$g^c(\theta_6,\theta_7)=\begin{bmatrix}g_5(\theta_6,\theta_7)\\g_6(\theta_6,\theta_7)\end{bmatrix}=\begin{bmatrix}l_i\sin(\theta_i)+l_6\sin(\theta_6)+a-l_7\sin(\theta_7)\\\\l_i\cos(\theta_i)+l_6\cos(\theta_6)+b-l_7\cos(\theta_7)\end{bmatrix} \tag{24}$$

$$g^d(\theta_5,\theta_8)=\begin{bmatrix}g_7(\theta_5,\theta_8)\\g_8(\theta_5,\theta_8)\end{bmatrix}=\begin{bmatrix}l_i\sin(\theta_i)+l_1\sin(\theta_1)+l_3\sin(\theta_3)+l_5\sin(\theta_5)+a-l_7\sin(\theta_7)-l_8\sin(\theta_8)\\\\l_i\cos(\theta_i)+l_1\cos(\theta_1)+l_3\cos(\theta_3)+l_5\cos(\theta_5)+a-l_7\cos(\theta_7)-l_8\cos(\theta_8)\end{bmatrix} \tag{25}$$

Equations (22) through (25) are extractions from the system of 8 equations in 8 unknowns in Equation (9). By ordering the sequence of numerical approximations, it is possible to determine, for example, $\theta_1$ from Equation (22) for use in calculating $\theta_3$ and $\theta_4$ from Equation (23). A similar argument can be extended to the determination of each angle in the 4 systems of 2 equations.

The shortcomings of such an approach become evident upon inspection of the Jacobian matrices necessary for use in the Newton-Raphson method. Consider, now, the 2x2 Jacobian matrix that would be derived when determining $\theta_3$ and $\theta_4$:

$$D(g^b(\theta_3,\theta_4))=\begin{bmatrix}\frac{\partial g_3}{\partial\theta_3} & \frac{\partial g_3}{\partial\theta_4}\\\\\frac{\partial g_4}{\partial\theta_3} & \frac{\partial g_4}{\partial\theta_4}\end{bmatrix} \tag{26}$$

As $\theta_1$ is now a fixed constant—a direct result of isolating the 8 equations into 4 pairs of 2—, during an iteration of the Multi-Order Newton-Raphson method, it is clear that the next estimates of $\theta_3$ and $\theta_4$ will not contain any partial derivative terms with respect to $\theta_1$. Given that is has been determined in Equation (11) that such terms are, in general, non-zero, this results in the the convergence point for $\theta_3$ and $\theta_4$ to necessarily differ from the point arrived at by means of the full system of 8 equations.

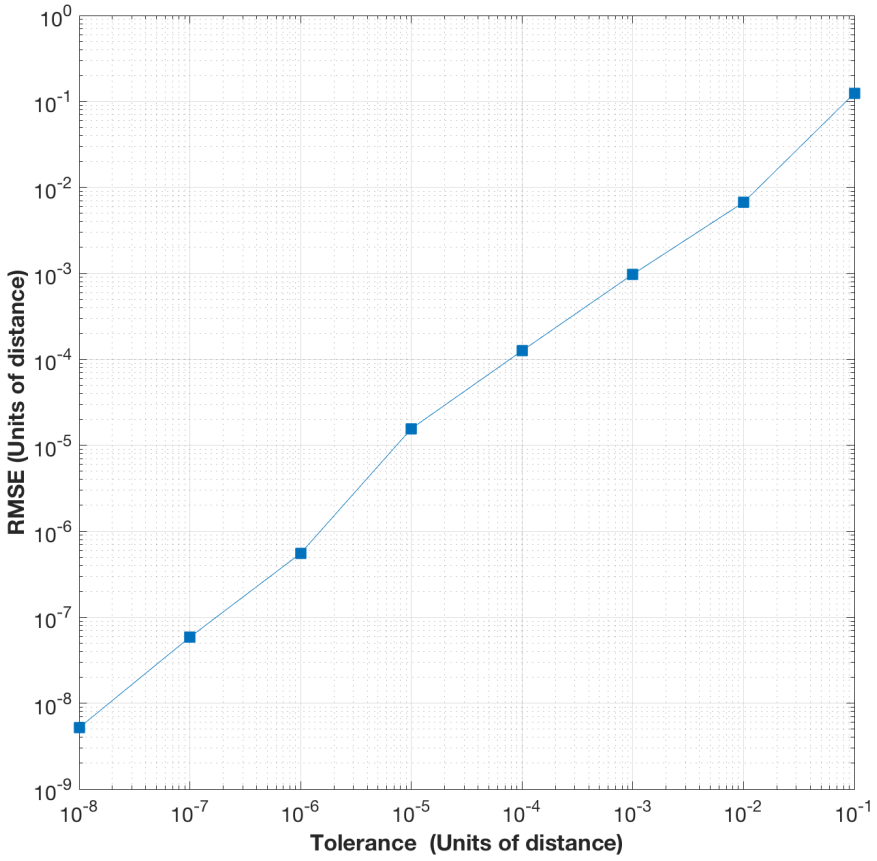
The nature of the error arising from such an approach for a given tolerance level is illustrated in the plot in Figure 4. The figure shows a plot of Root Mean Square Error (RMSE) of the sequential approach against tolerance, using the full approach as a comparison for error. The equation derived for RMSE is shown in Equation (27).

A note on notation: the superscripts full and seq on $x$ and $y$ indicate the position values obtained from the full and sequential approaches, respectively. The summing index $j$ indicates the values obtained from the $j$th crank angle step out of $M$ crank angle steps while completing one full rotational cycle.

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_{j=0}^{M-1}(\delta_j)^2} \tag{27}$$

$$\text{where } \delta_j = \sqrt{(x_P^{\text{full}} - x_P^{\text{seq}})^2 + (y_P^{\text{full}} - y_P^{\text{seq}})^2}$$

Figure 4: Log-Log plot of RMSE against the Tolerance level set in the Newton-Raphson algorithm




Though analytically inaccurate in their mathematical description of the system, the straight line plot in Figure 4 indicates that, over the range of tolerances examined, the sequential approach results in estimations that have discrepancies on the order of the desired tolerance used in the Newton-Raphson algorithm. If such discrepancies are acceptable, then the sequential version of the system of equations results in modest computational efficiency gains.

Improvements in algorithm efficiency can (very primitively) be understood by considering a single iteration of the full Newton-Rapshon algorithm with 8 systems of equations. Computing the next set of estimates requires inverting an 8x8 matrix—the most computationally taxing operation involved in an iteration of the algorithm—with a typical[1] asymptotic time complexity of $O(n^3)$. Though Big O analysis is not necessarily appropriate in this specific case—given that asymptotic performance is not of concern, knowing n to be of

size 8—, it serves a purpose in illustrating that the inversion of 4 2x2 matrices does not involve the same computational complexity as that of inverting the 8x8. More formally,

$$O(n^3) \not\equiv 4 \times O((n/4)^3)$$

In the interest of preserving the reader's sanity, the code for the sequential version is not included in the this report. The code can be found in the GitHub repository referenced in the final section of this report.

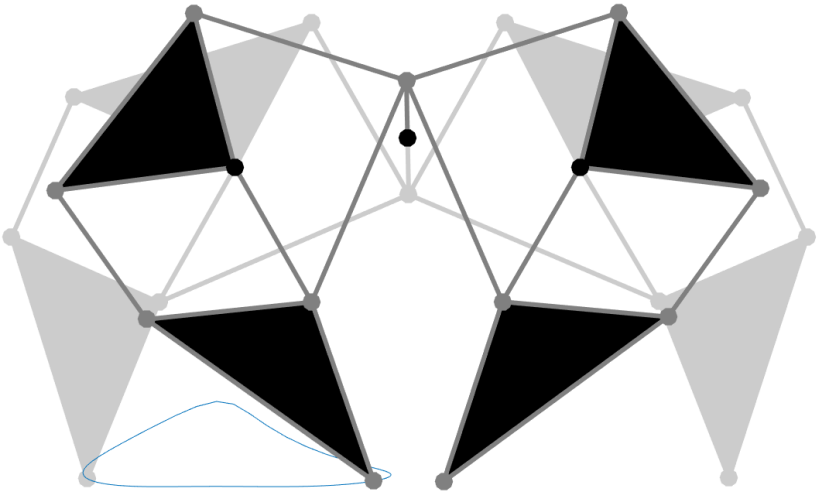
## Four-Legged Render

After implementing the Jansen linkage system for a single leg, I felt morally obligated to carry the assignment to the natural next step. Though vastly more time efficient approaches could yield the same results by utilizing already computed values for $\theta_1$ through $\theta_8$, by computing the angles $\alpha_1$ through $\alpha_8$ for another linkage system with a given crank angle $\alpha_i$ phase shifted $\pi$ rad from $\theta_i$, it is possible to plot a rendering of another "leg" behind the original.

By incrementing the input angles $\omega_i$ and $\beta_i$ of yet another set of legs in the opposite direction to that of $\theta_i$ and $\alpha_i$, and by translating the horizontal position of all joints in this pair of legs through the vertical axis, we obtain the "hind legs" of the "four -legged beest".

Note that in order to plot the legs with crank angles phase shifted $\pi$ rad from $\theta_i$, it is necessary to determine a second set of initial estimates. The estimates used in this assignment were obtained from the individual leg simulation in the previous subsections.

Figure 5: Four-Legged Jansen Beest



### Code Repository & YouTube Video

The Matlab code used for this assignment is available on
`https://github.com/cvigoe/Jansen`

A video of the renderings from Matlab of the four legged system solved pseudo-statically for several cycles of crank angle is available on
`https://www.youtube.com/watch?v=7gkSlkvqqIc&feature=youtu.be`

## References

[1] Wikipedia *Computational complexity of mathematical operations, Matrix algebra.*
`https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations`