This file is provided for educational purposes as guidance for the use of the software tool. It is not guarateed to be free from errors or ommissions. The methods and assumptions may not apply in all cases. Engineers and designers should seek design and calculation approval from a relevant organisation.
Coventry University - Nov 2013 (David Sandells)

This is an example to show you how SMath can calculate the movement of kinematic mechanisms.

Plugins : You will need the following plugins (tools>plugins) enabled to view this file:-
Math Region, Picture Region, Plot Region, Text Region, Area Region
SMath Studio Files Plugin
Special Functions
You can find plugins here:- http://smath.info/?extensions=SMathStudio_Desktop
http://en.smath.info/forum/yaf_topics32_Extensions.aspx

Plotting functions (for use with making graphs)

$size := 15$

$sym := "+"$

$col := "Red"$

Settings for symbol, size and colour of the plotted jointgs.

$pt(vec) := \left(vec_1 \; vec_2 \; sym \; size \; col \right)$

Function to plot a point based on a position vector.

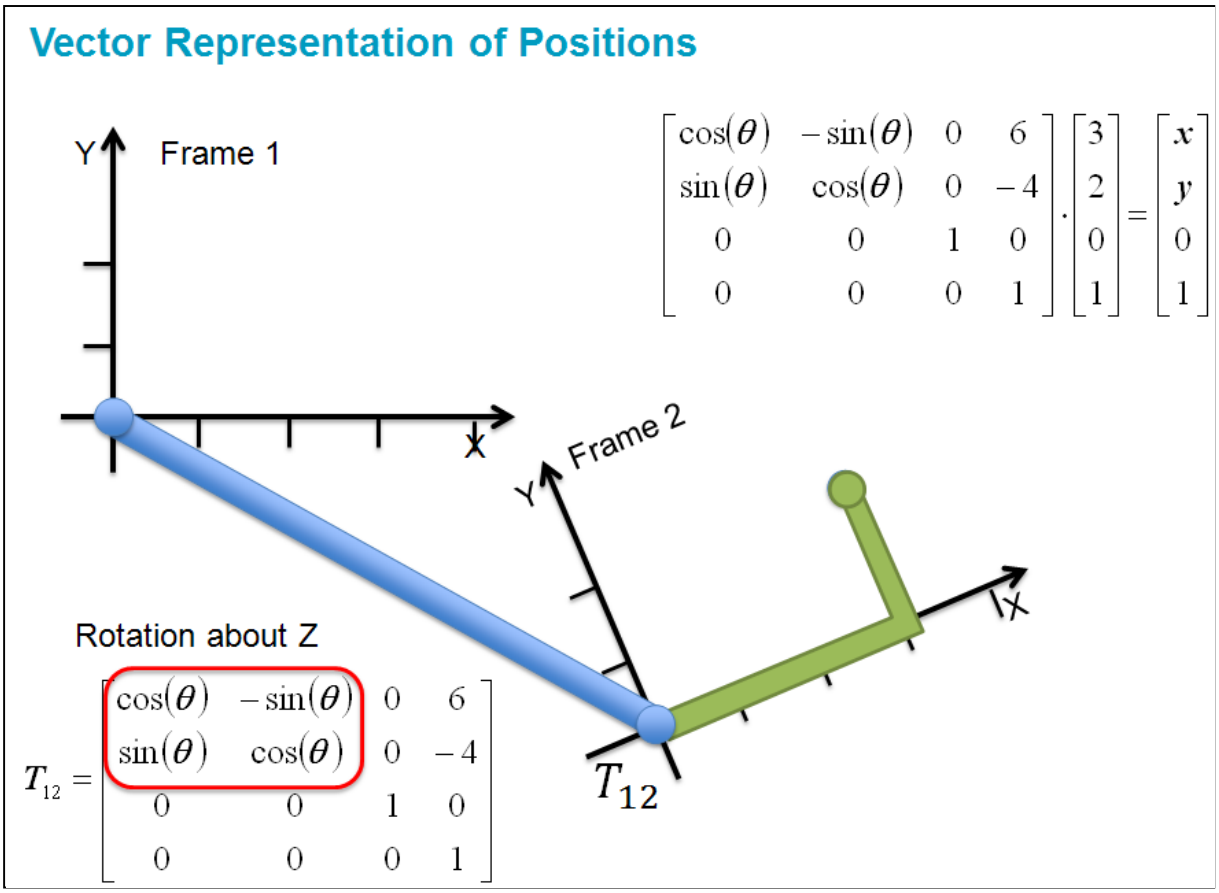
$whl(vec) := \left(vec_1 \; vec_2 \; "." \; 145 \; "Black" \right)$

Function to plot a wheel shape (a very big dot!) based on a position vector.

$lne(vec1, vec2) := \begin{pmatrix} vec1_1 & vec1_2 \\ vec2_1 & vec2_2 \end{pmatrix}$

Function to plot a line between two points defined by vectors.

Definition of Transforms



$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 6 \\ \sin(\theta) & \cos(\theta) & 0 & -4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

$$T_{12} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 6 \\ \sin(\theta) & \cos(\theta) & 0 & -4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation from Frame 1 to Frame 2 is a combined translation and rotation:-

The following is a symbolic equation (bold equal signs).
It is for show and is not actually calculated

$$T_{12} \equiv \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 6 \\ \sin(\theta) & \cos(\theta) & 0 & -4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \equiv Tr(6, -4, 0) \cdot R_Z(\theta)$$

Combined Matrix *Translation Matrix* *Rotation Matrix* *Using the functions to create the matricies*

Location of each of the points in the mechanism is found by moving to the right coordinate frame and then moving to a vector position within that frame of reference. If we are interested in the origin then the vector position is (0,0,0,1). (The extra '1' is needed to make the matrix calculations work).

$$Pt1 := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Point 1 is origin (don't go anywhere)

$$Pt2 := Tr(6, -4, 0) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Point 2 is the end of the first arm - translate to the bottom of the arm and then don't go anywhere else.
In fact this is the same as a vector (6,-4,0,1)

$$Pt3(\theta) := Tr(6, -4, 0) \cdot R_Z(\theta) \cdot \begin{pmatrix} 3 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

point 3 is the right angle of the second arm
Translate then rotate to get into the second coordinate frame (frame 2) and then go to position 3,0,0.
This depends on the rotation angle so is not a constant and therefore is defined as a function of the rotation angle.

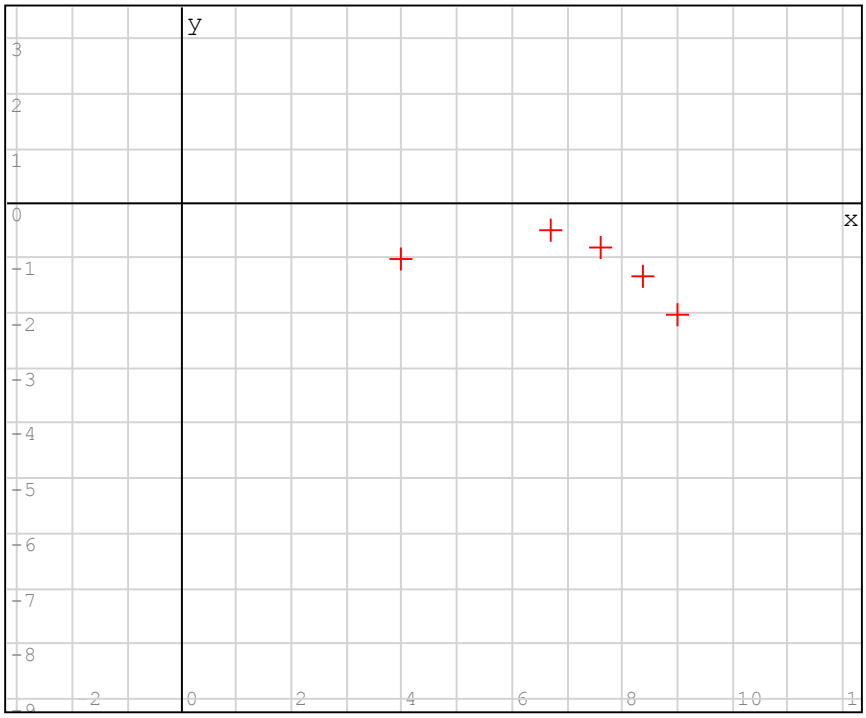
$$Pt4(\theta) := Tr(6, -4, 0) \cdot R_Z(\theta) \cdot \begin{pmatrix} 3 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

point 3 is the end point of the second arm
Translate then rotate to get into the second coordinate frame (frame 2) and then go to position 3,2,0.
As before defined as a function of the rotation angle.

Lets try using the function to find out where the end point (point 4) will be for different rotation angles.

$$Pt4(0\ deg) = \begin{pmatrix} 9 \\ -2 \\ 0 \\ 1 \end{pmatrix} \quad Pt4(90\ deg) = \begin{pmatrix} 4 \\ -1 \\ 0 \\ 1 \end{pmatrix} \quad Pt4(180\ deg) = \begin{pmatrix} 3 \\ -6 \\ 0 \\ 1 \end{pmatrix} \quad Pt4(270\ deg) = \begin{pmatrix} 8 \\ -7 \\ 0 \\ 1 \end{pmatrix}$$

Plot of the end point (Point 4) for different rotation angles

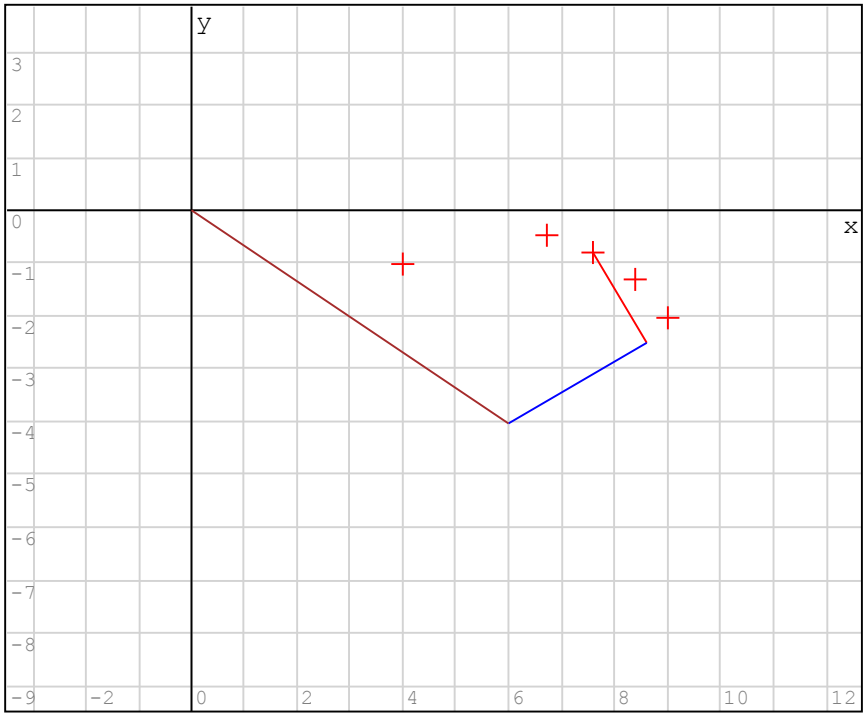


$$\begin{cases} pt(Pt4(0\ deg)) \\ pt(Pt4(15\ deg)) \\ pt(Pt4(30\ deg)) \\ pt(Pt4(45\ deg)) \\ pt(Pt4(90\ deg)) \end{cases}$$

Using graphs like this can help visualise the movement of the mechanism. However, you have to add a term for each point you want to plot.

We can use the line and point ploting functions above to generate data sets that can be plotted on the graph to show the position of the mechanism at a particular point.

Arm1:= lne(Pt1 , Pt2) Arm2a:= lne(Pt2 , Pt3(30 *deg*)) Arm2b:= lne(Pt3(30 *deg*) , Pt4(30 *deg*))



pt(Pt4(0 *deg*))
pt(Pt4(15 *deg*))
pt(Pt4(30 *deg*))
pt(Pt4(45 *deg*))
pt(Pt4(90 *deg*))
Arm1
Arm2a
Arm2b

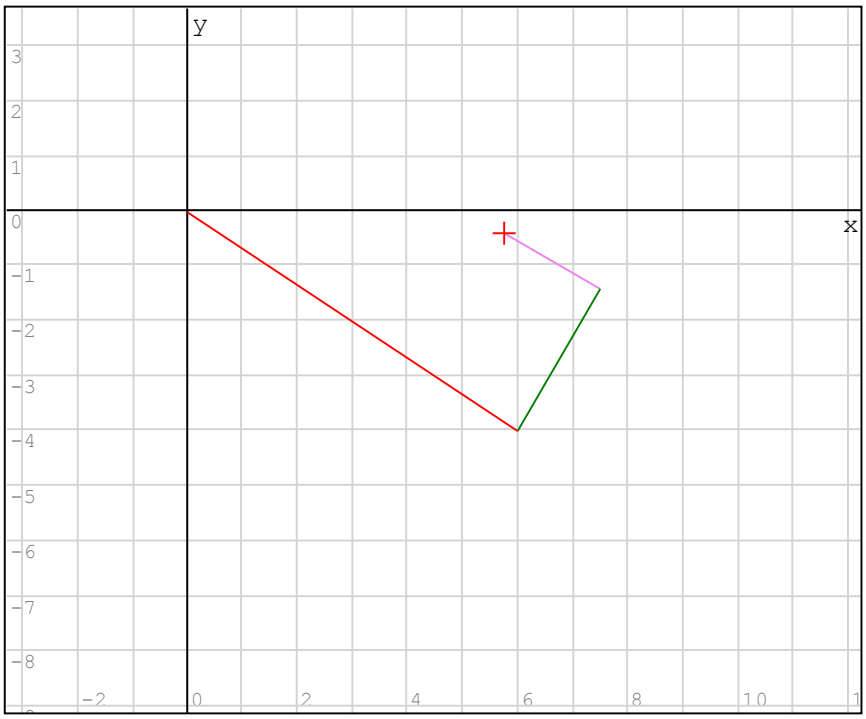
As an alternative, the graph below uses animation to show the movement of the mechanism.
Functions are used to allow production of different data for different angles
The input "t" is replaced by the chosen animation list variable (Listθ).
To set this up by right click on the plot and selecting "ani" variable under the animate sub-menu.

The "Frame Rate" sub menu controls the animation speed etc.

for k := 1, k < 20, k := k + 1
  $List_{\theta_k} := 0\,deg + 10\,deg \cdot k$

This 'for loop' sets up a list of angles to be used in the animated graph. The list is shown off the page to the right.

Arm1 := lne(Pt1, Pt2) Arm2a(θ) := lne(Pt2, Pt3(θ)) Arm2b(θ) := lne(Pt3(θ), Pt4(θ))



pt(Pt4(t))
Arm1
Arm2a(t)
Arm2b(t)

More Complex mechanism

In addition to the rotation of the second arm we can also allow the first arm to rotate by and angle we will call α.

$$Pt1 := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$Pt2(\alpha) := R_Z(\alpha) \cdot Tr(6, -4, 0) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Now we need to rotate the first coordinate frame by angle α before we move down the arm.

$$Pt3(\alpha, \theta) := R_Z(\alpha) \cdot Tr(6, -4, 0) \cdot R_Z(\theta) \cdot \begin{pmatrix} 3 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$Pt4(\alpha, \theta) := R_Z(\alpha) \cdot Tr(6, -4, 0) \cdot R_Z(\theta) \cdot \begin{pmatrix} 3 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

With this mechanism we can set an additional constraint that the end point (point 4) must slide along the x axis.

Therefore as we vary α we need to find the angle θ that will keep the end point on the x axis (i.e. y=0)

We can use the "Solve" function (Part of the special functions plugin) to achieve this.

'Plugins' can be found through the SMath forum website.
They are *.dll files that need to be copied in to the right directory (usually c:\smath\smath studio\plugins). When done, the plugin will load automatically the next time SMath is started.

We are trying to find the angle that will produce a y=0 coordinate. Therefore we need to define a function that only provides the y value (the second element of the vector result).

$$fn(\alpha, \theta) := Pt4(\alpha, \theta)_2$$

Example :- $Pt4(10\,deg, 5\,deg) = \begin{pmatrix} 8.9836 \\ -0.189 \\ 0 \\ 1 \end{pmatrix}$ $fn(10\,deg, 5\,deg) = -0.189$

The 'solve' function will tell us what values make the function output=0

$$solve(fn(5\,deg, \theta), \theta, -180\,deg, 180\,deg) = \begin{pmatrix} 35.079 \\ 67.5409 \end{pmatrix} deg$$

Solve finds mutiple values of θ that make the fn=0

This time we need to animate both the α and θ values so we will generate a separate animation variable (ani) to control both.

$ani := 1..20$

We can then generate a list of α angle values

for $k \in ani$ This means for k equal to every value in ani in turn.

$List_{\alpha_k} := \left(5\ deg + \frac{k}{20} \cdot 50\ deg\right)$

And using the solve method we can find a matching list of θ values. Note we have limited the angles through which solve can search (-140,+60) which should mean that we only get one value of θ for each value of α. You can see that this is ok in the results shown off the right hand side.

for $k \in ani$

$List_{\theta_k} := solve\left(fn\left(List_{\alpha_k}, \theta\right), \theta, -140\ deg, 60\ deg\right)$ *Find a value of θ between -140 and +60 degs that makes the fn=0*

The lines for the graph now need to functions of both α and θ.

$Arm1(\alpha) := lne(Pt1, Pt2(\alpha))$

$Arm2a(\alpha, \theta) := lne(Pt2(\alpha), Pt3(\alpha, \theta))$

$Arm2b(\alpha, \theta) := lne(Pt3(\alpha, \theta), Pt4(\alpha, \theta))$

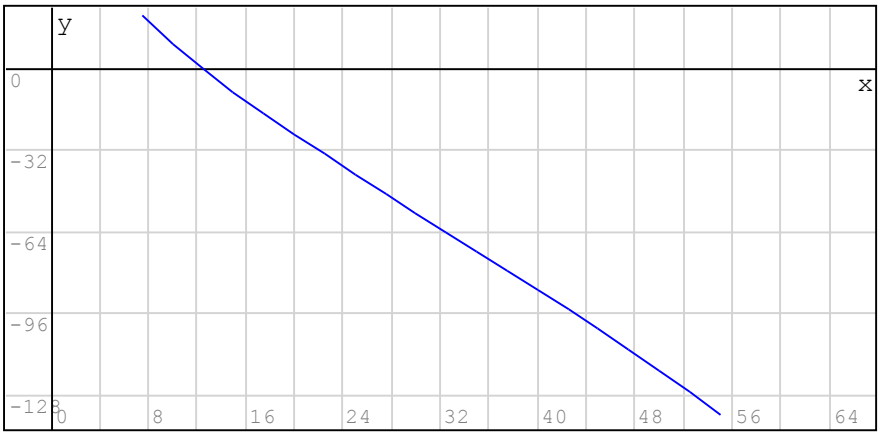
This graph animates using the 'ani' variable to step through each result in the list and display it.

$$\begin{cases} pt\left(Pt4\left(List_{\alpha_t}, List_{\theta_t}\right)\right) \\ Arm1\left(List_{\alpha_t}\right) \\ Arm2a\left(List_{\alpha_t}, List_{\theta_t}\right) \\ Arm2b\left(List_{\alpha_t}, List_{\theta_t}\right) \end{cases}$$

The 't' is replaced by each value of 'ani' and this calls each element of the result list in turn

We can now plot the the result data sets against one another.

Arm2 Angle (deg) v Arm1 Angle (deg)



$$\text{augment}\left(\frac{\text{List}_{\alpha}}{deg}, \frac{\text{List}_{\theta}}{deg}\right)$$

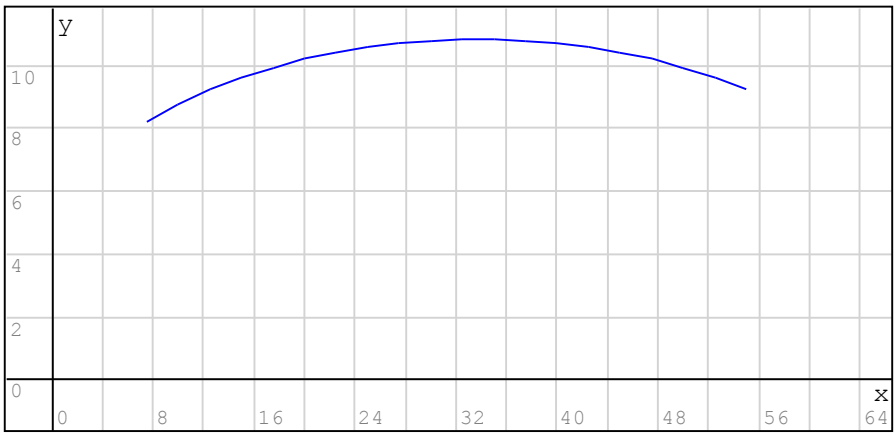
Here calculate the x position from the angle data and plot that.

for $k \in ani$

$$\text{List}_{x_k} := \text{Pt4}\left(\text{List}_{\alpha_k}, \text{List}_{\theta_k}\right)_1$$

For each pair of angles calculate the x position (element '1' of the vector)

End point distance along x axis v Arm1 angle (deg)



Plot of input angle v output position

$$\text{augment}\left(\frac{\text{List}_{\alpha}}{deg}, \text{List}_{x}\right)$$

We can now take this data further. Lets assume that the mechanism is driven by a motor turning arm1 with 10Nm of torque. If the joints had zero friction then no energy is lost in them and therefore the output force must be enought to balance the input energy.

$Energy_{in} = Torque \cdot Angle$

$Energy_{out} = Force \cdot distance$

Note that if the joints are not perfect then we need to calculate the energy lost along the way and substract it.

$ani2 := 1 \,..\, (length(ani) - 1)$

If we are calculating the difference in distance then we will end up with a shorter vector.

for $k \in ani2$

$Ergy_{in_k} := \left(List_{\alpha_{k+1}} - List_{\alpha_k}\right) \cdot 10\, N\, m$

List of input energies (Torque*Angle)

for $k \in ani2$

$Frc_{out_k} := \dfrac{Ergy_{in_k}}{\left(List_{x_{k+1}} - List_{x_k}\right) m}$

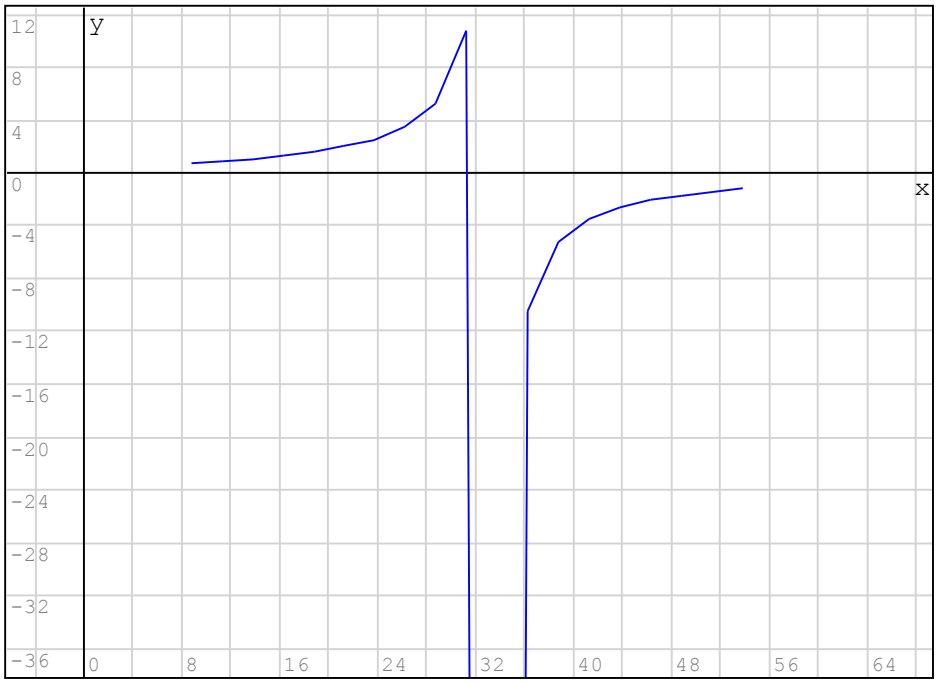
List of output forces (energy/distance)

for $k \in ani2$

$List_{\alpha 2_k} := \left(\dfrac{List_{\alpha_k} + List_{\alpha_{k+1}}}{2}\right)$

Shorter list of input angles for plotting with which takes the average position between the points

Force generated (N) v Arm1 angle (deg)



A graph like this could help with mechanism sizing and design.
If we want a more accurate result then we need to add more calculation points.

$augment\left(\dfrac{List_{\alpha 2}}{deg}, \dfrac{Frc_{out}}{N}\right)$