Inverse Kinematics

Forward Kinematics solves the problem “if I know the link transformation parameters, where are the links?”.

Inverse Kinematics (IK) solves the problem “If I know where I want the links to be (X*, Y*), what link transformation parameters will put them there?”
Inverse Kinematics (IK):
Things Need to Move – What Parameters Will Make Them Do That?

Remember the Idea Behind Newton’s Method?

“Pick an initial guess of the input parameter and keep refining it until the answer it produces is close enough”

1. \[ \Delta y = \frac{dy}{dx} \Delta x \]
2. \[ \Delta x = -\frac{y_{\text{have}}}{dy/dx} \]
3. \[ x = x + \Delta x \]

Now we just need to do it with more than one input parameter.
Jacobian Method

Our input parameters (in this case) are the three rotation angles. If \((X^*, Y^*)\) are where we want the end of the third link to end up, then:

\[
X^* = f(\theta_1, \theta_2, \theta_3) \quad \text{and} \quad Y^* = g(\theta_1, \theta_2, \theta_3)
\]

If you have a good guess for \((\theta_1, \theta_2, \theta_3)\), which currently produce an \((X^*, Y^*)\) that is not quite what you want, then you can refine your values for \((\theta_1, \theta_2, \theta_3)\) and try again. The fundamental equations for this are:

\[
X^* - X_{approx}^* = \Delta X = \frac{\partial X^*}{\partial \theta_1} \Delta \theta_1 + \frac{\partial X^*}{\partial \theta_2} \Delta \theta_2 + \frac{\partial X^*}{\partial \theta_3} \Delta \theta_3
\]

\[
Y^* - Y_{approx}^* = \Delta Y = \frac{\partial Y^*}{\partial \theta_1} \Delta \theta_1 + \frac{\partial Y^*}{\partial \theta_2} \Delta \theta_2 + \frac{\partial Y^*}{\partial \theta_3} \Delta \theta_3
\]

The matrix:

\[
\begin{bmatrix}
\frac{\partial X^*}{\partial \theta_1} & \frac{\partial X^*}{\partial \theta_2} & \frac{\partial X^*}{\partial \theta_3} \\
\frac{\partial Y^*}{\partial \theta_1} & \frac{\partial Y^*}{\partial \theta_2} & \frac{\partial Y^*}{\partial \theta_3}
\end{bmatrix}
\]

is called the Jacobian, and is abbreviated as \([ J ]\):

\[
\begin{bmatrix}
\Delta \theta_1 \\
\Delta \theta_2 \\
\Delta \theta_3
\end{bmatrix} = [ J ] \begin{bmatrix}
\Delta X \\
\Delta Y
\end{bmatrix}
\]
Solving the Equations

Note that \([J]\) is not a square matrix, so this system of equations cannot be solved for directly. But, if we pre-multiply by the transpose of \([J]\), we get:

\[
\begin{pmatrix}
\Delta \theta_1 \\
\Delta \theta_2 \\
\Delta \theta_3
\end{pmatrix} = \begin{pmatrix}
\Delta X \\
\Delta Y
\end{pmatrix}
\]

which is solvable because it is 3-equations-3-unknowns.

It is not obvious, but this is the Least Squares formulation. It will give an optimum \((\Delta \theta_1, \Delta \theta_2, \Delta \theta_3)\) to make \((X^*, Y^*)\) move closer to the desired values.

Iterating to get a Solution (note similarity to Newton’s Method)

Differentiate the equations \(X^* = f(\theta_1, \theta_2, \theta_3)\) and \(Y^* = g(\theta_1, \theta_2, \theta_3)\)

Pick a starting \(\theta_1, \theta_2, \text{ and } \theta_3\)

1. Compute \(X^*_{\text{approx}}\) and \(Y^*_{\text{approx}}\) from \((\theta_1, \theta_2, \theta_3)\)
2. Compute \(\Delta X = X^*-X^*_{\text{approx}}\) and \(\Delta Y = Y^*-Y^*_{\text{approx}}\)
3. If \(|\Delta X|\) and \(|\Delta Y|\) are “small enough”, we’re done

4. Compute:

\[
\frac{\partial X^*}{\partial \theta_1} \cdot \frac{\partial X^*}{\partial \theta_2} \cdot \frac{\partial X^*}{\partial \theta_3} \cdot \frac{\partial Y^*}{\partial \theta_1} \cdot \frac{\partial Y^*}{\partial \theta_2} \cdot \frac{\partial Y^*}{\partial \theta_3}
\]

5. Form the Jacobian \([J]\)
6. Solve the system of equations:

\[
\begin{pmatrix}
\Delta \theta_1 \\
\Delta \theta_2 \\
\Delta \theta_3
\end{pmatrix} = \begin{pmatrix}
\Delta X \\
\Delta Y
\end{pmatrix}
\]

7. Refine:

\[
\theta_1 = \theta_1 + \Delta \theta_1; \theta_2 = \theta_2 + \Delta \theta_2; \theta_3 = \theta_3 + \Delta \theta_3
\]
The Calculus Product Rule Tells us How to Differentiate Scalar Variables that have been Multiplied

\[ c = ab \]

\[ \frac{dc}{d\theta} = \frac{d(ab)}{d\theta} = \frac{da}{d\theta}b + a\frac{db}{d\theta} \]

The Calculus Product Rule Tells us How to Differentiate Matrices that have been Multiplied

\[ [M] = [T][R_\theta] \]

\[ \frac{d[M]}{d\theta} = \frac{d([T][R_\theta])}{d\theta} = \frac{d[T]}{d\theta} [R_\theta] + [T] \frac{d[R_\theta]}{d\theta} \]

\[ \frac{d[R_\theta]}{d\theta} = \frac{d}{d\theta} \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\sin \theta & -\cos \theta & 0 & 0 \\ \cos \theta & -\sin \theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \]
Numerical Differentiation

Rather than explicitly differentiating an equation, it is often easier to calculate the derivative numerically. This is known as the **Central Difference** method, where you look both forward and backwards to see how the dependent variable is changing with respect to the independent variable:

\[
\frac{\partial X}{\partial \theta} \approx \frac{X(\theta + \Delta \theta) - X(\theta - \Delta \theta)}{2\Delta \theta}
\]

Pick a delta that is small, but not so small that floating point accuracy becomes an issue.

If there are other independent variables, hold them constant:

\[
\frac{\partial X}{\partial \theta_1} \approx \frac{X(\theta_1 + \Delta \theta_1, \theta_2, \theta_3) - X(\theta_1 - \Delta \theta_1, \theta_2, \theta_3)}{2\Delta \theta_1}
\]
**Numerical Differentiation**

In case you ever need it, here is how to compute the second derivative:

\[
\frac{\partial^2 X}{\partial \theta^2} \approx \frac{X(\theta + \Delta \theta) - 2X(\theta) + X(\theta - \Delta \theta)}{(\Delta \theta)^2}
\]

As before, if there are other independent variables, hold them constant:

---

**Another IK Approach:**

**Cyclic Coordinate Descent (CCD) Method**

The idea is to change $\Theta_1$ so that $(X,Y)$ are as close to $(X^*,Y^*)$ as possible.

1. Then change $\Theta_2$.
2. Then change $\Theta_3$.
3. Then change $\Theta_1$.
4. Then change $\Theta_2$.
5. Then change $\Theta_3$.
6. Then change $\Theta_1$.

\[\ldots\]
Changing $\Theta_1$

Holding $\Theta_2$ and $\Theta_3$ constant, rotate $\Theta_1$ so that the blue lines line up.
Changing $\Theta_2$

Holding $\Theta_1$ and $\Theta_3$ constant, rotate $\Theta_2$ so that the blue lines line up.
Changing $\Theta_3$

Holding $\Theta_1$ and $\Theta_2$ constant, rotate $\Theta_3$ so that the blue lines line up.
Now, do it again -- Changing $\Theta_1$
Now, do it again -- Changing $\Theta_2$
Now, do it again -- Changing Θ3

(X,Y)

(X*,Y*)
Computing how much to change a rotation
(in this example, we are changing $\theta_2$)

Where we are now: $(X_3,Y_3)$

Where we want to be: $(X',Y')$

Use the C/C++ $\text{atan2}$ function:  

\[ \theta' = \text{atan2}\left( Y' - Y_2, X' - X_2 \right); \]
\[ \theta = \text{atan2}\left( Y_3 - Y_2, X_3 - X_2 \right); \]

\[ \Delta \theta_2 = \theta' - \theta \]

Do not use the C/C++ $\text{atan}$ function:  

\[ \theta' = \text{atan}\left( \frac{Y' - Y_2}{X' - X_2} \right); \]
\[ \theta = \text{atan}\left( \frac{Y_3 - Y_2}{X_3 - X_2} \right); \]

\[ \Delta \theta_2 = \theta' - \theta \]