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?”

The idea is to change $\Theta_1$ so that (X,Y) are as close to (X*,Y*) as possible. Then change $\Theta_2$. Then change $\Theta_3$. Then change $\Theta_1$. Then change $\Theta_2$. Then change $\Theta_3$. Then change $\Theta_1$.

...
Changing Θ1

Holding Θ2 and Θ3 constant, rotate Θ1 towards (X*,Y*) so that the dashed purple lines line up.

Changing Θ2

Holding Θ1 and Θ3 constant, rotate Θ2 towards (X*,Y*) so that the dashed purple lines line up.
Changing Θ3

Holding Θ1 and Θ2 constant, rotate Θ3 towards (X*,Y*) so that the dashed purple lines line up.

Now, do it again -- Changing Θ1
Now, do it again -- Changing Θ2

Now, do it again -- Changing Θ2

Now, do it again -- Changing Θ3

Now, do it again -- Changing Θ3
Computing how much to change a rotation by
(in this example, we are changing $\theta_2$)

Where we are now: $(X_2, Y_2)$

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

* $\theta^* = \text{atan2}(Y' - Y_2, X' - X_2)$;
* $\theta_2 = \text{atan2}(Y_3 - Y_2, X_3 - X_2)$;
* $\Delta \theta_2 = \theta^* - \theta_2$

Use the C/C++ atan2() function:  Do not use the C/C++ atan() function:

$\theta^* = \text{atan}(Y' - Y_2 / (X' - X_2))$;
$\theta_2 = \text{atan}(Y_3 - Y_2 / (X_3 - X_2))$;
$\Delta \theta_2 = \theta^* - \theta_2$