Numerical Optimization - Handin 6

Martin Simon Haugaard - CDL966

April 4, 2016

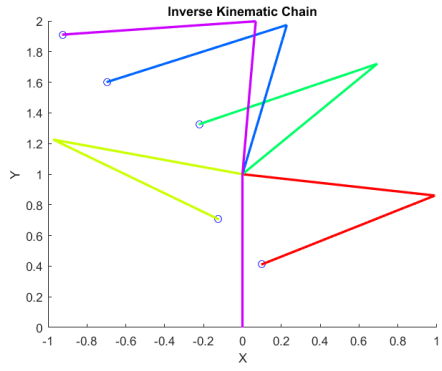
1 Programming

This week it's all about implementing constraints in the Inverse Kinematics framework. I've chosen to stay with *Steepest Descent* approch, and focus purely on implementing the constraints. My implementation works initially like any Steepest Descent method, by calculating a steepest descent direction

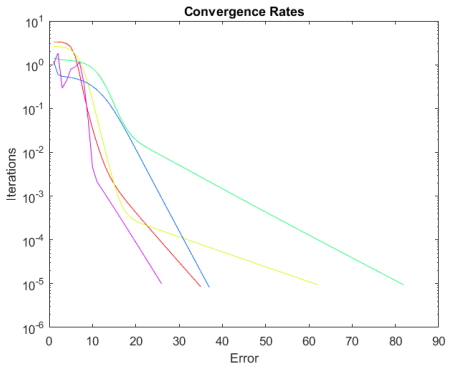
$$p_k = -J' * (e - goal)$$

With Jacobian, J, desired location $goal$ and e is the end-effector position.

Once the direction is computed, a desired location is computed by calculating a series of angles, which would best achieve the desired location. Figure 1a shows how this will look, when there are no constraints, or if the constraints applied are large enough not to have an effect (which is the case for this plot). To verify that the steepest descent is



(a) Unconstricted Steepest Descent Inverse Kinematic



(b) Convergence Plot for Unconstricted Steepest Descent

Figure 1

in effect, the convergence plot is seen in Figure 1b. It can be seen that the convergence plot error

$$error = 0.5 * ||r||^2$$

where $r = e - goal$, is almost logaritmic, which is the desired effect we want.

To add the constraints an alteration was made to the Line Search algorithm in use. After calculating the ideal angles, these are sent through a projection script, which will make sure they are all within the given constraints, as described in (16.69). Because of the constraints, an alteration is needed on the Backtracking Line Search algorithm (P. 37).

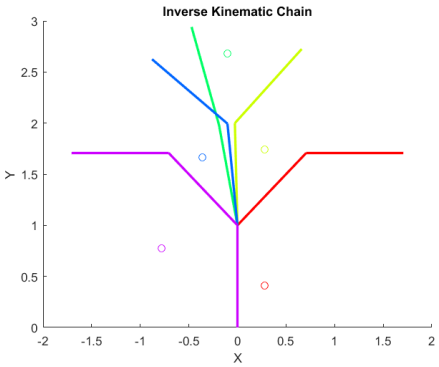
Instead of the original:

$$\text{While } f(x_k) + c\alpha\nabla f_k^T p_k <= f(x_k + \alpha p_k)\{\dots\}$$

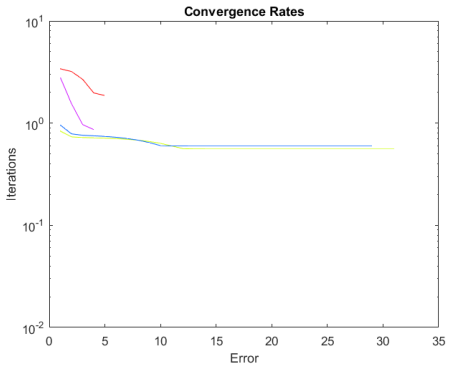
A loop taking the projecting into account is needed:

$$\text{While } error_{projected} + c\alpha\nabla f_k * (\theta_{projected} - \theta_{org}) <= error_{org}$$

This loop guarentees that the projected angles are still approching the desired location, but allows the new angles to be projections.



(a) Constricted Steepest Descent Inverse Kinematic



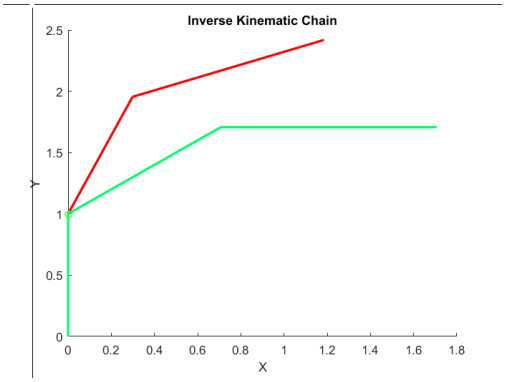
(b) Convergence Plot for constricted Steepest Descent

Figure 2

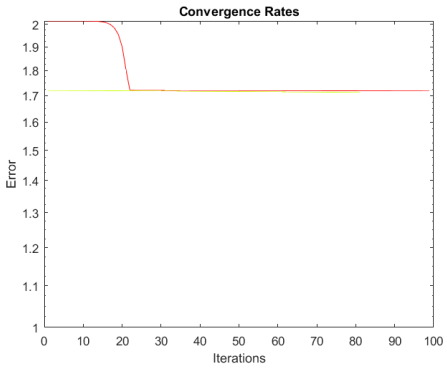
In Figure 3 the constraining angle were set to $\frac{\pi}{4}$ and $-\frac{\pi}{4}$ for all the joints. This allows for a 90 degree angle in total in which the joints can move. Observing the red and purple plots, the angle between these is indeed 90 degrees. And for both of these examples they all their angles are at the maximum allowed angle for each of the joints, just in opposite directions. The convergence plots for these are almost linear when plotted on a logaritmic scale, however, the error is generally a lot bigger. Which makes sense since we are adding limits as to how close to the ideal we allow it to be.

As for the rest of the three plots, they struggle a bit more trying to approch the minimum. As seen by the increased number of iterations conducted, however, they do manage to make what seems to be the best possible approximation, given the constraints.

A general flaw I found when using Steepest Descent, was the dependencies on the initial guess. When trying to reach a point, outside the reachable area, depending on the initial guess, the approximation may be fairly good, or horribly wrong.



(a) Constricted Steepest Descent Inverse Kinematic



(b) Convergence Plot for constricted Steepest Descent

Figure 3

Figure 3a shows two plots, both with the desired location $x^* = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. The Green plot has initial angles $[0; 0; 0]$ while the red plot reuses the green plot's end final angles. It can be seen two different results are given. The cause likely being that Steepest Descent lacks the smarts to realize that it's already in an ideal position when starting, and it naively tries to improve upon the initial guess. It's a general error I tend to have, I can't, at the moment, find a good way of making the initial angles, without having a few of my results clearly being affected by poor start location.

2 Exercises

12.13

$$
\begin{aligned}
(x_1 - 1)^2 + (x_2 - 1)^2 &\leq 2, \\
(x_1 - 1)^2 + (x_2 + 1)^2 &\leq 2, \\
x_1 &\geq 0
\end{aligned}
$$

is the same as writing constraints

$$
\begin{aligned}
c_1(x) = 2 - (x_1 - 1)^2 - (x_2 - 1)^2 &\geq 0, \\
c_2(x) = 2 - (x_1 - 1)^2 - (x_2 + 1)^2 &\geq 0, \\
c_3(x) = x_1 &\geq 0
\end{aligned}
$$

Giving

$$\nabla c_1(x) = \begin{bmatrix} -2(x_1 - 1) \\ -2(x_2 - 1) \end{bmatrix}$$
$$\nabla c_2(x) = \begin{bmatrix} -2(x_1 - 1) \\ -2(x_2 + 1) \end{bmatrix}$$
$$\nabla c_3(x) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

All the sets are active in $x^* = (0, 0)^T$ meaning the dimensions of the problem is 3. However, the dimensions of the problem is limited to 2. For LICQ to be linear independent

$$\nabla c_1(x), i \in A(x)$$

Which is not the case, hence LICQ is not satisfied.

For MFCQ, there has to be a vector $w \in R^n$ such that $\nabla c_1(x^*)^T w > 0$ for all $i \in A(x^*) \cap I$, this however does hold $(w = (1, 0))$. Thus MFCQ is satisfied.