Norwegian University of Science
and Technology
Department of Mathematical
Sciences

TMA4180 Optimization Theory
Spring 2013

Exercise set "Project"

Instructions

- The deadline for the exercise is Monday April 22, at 16:00.
- You can work in groups of up to 2 students.
- The report should be short, not more than 5-6 pages in LaTeX (or equivalent). Hand written reports are OK, but write readable. The report can be in English or Norwegian.
- Do not include listings of your Matlab code. But describe your algorithms (or refer to the relevant algorithm in the textbook).
- Write your student number(s) on the top of the report. Do not use your name.
- Send the report by email to Markus, or put it in my mailbox in the 7th floor, SBII.
- You are free to collaborate between groups, but do not copy from each other. If you use information from books, internet, etc., always give citations.
- You are encouraged to play around with algorithms, parameters, initial values, sub-algorithms, etc. Solve more test problems, and maybe more difficult problems. Use different algorithms, and compare them. Innovative, relevant and correct work can to some extent compensate for less work on other tasks.
- The project counts for 20% of the grade. If you do not do the project, you can still take the exam, but the best grade you can achieve is a C.

Tutorials: These two weeks, the tutorials are moved to MA24 (in the basement of the green building behind Realfagsbygget), from 16:15-18:00. Do you need more help, you are free to contact us outside the tutorial hours, but please make an appointment first.

1 Introduction

In this project, you will develop some matlab code to calculate the inverse kinematics of robots.

We will assume that we are dealing with a planar robot consisting only of a chain of $n-1$ rigid *segments* of fixed length connected by revolute joints as seen in Figure 1. Furthermore,

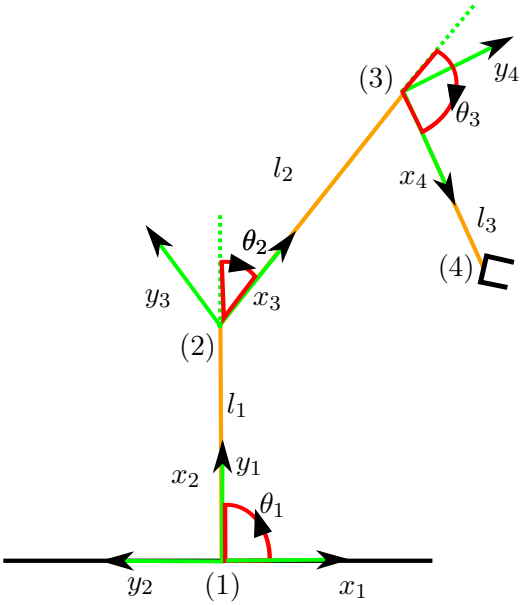

Figure 1: Notation for our planar multi-joint robot. The robot starts on the ground with joint (1). The hand or tool, which is freely orientable, is located at joint (4). In every joint but the last one, we have a local rectangular coordinate system oriented along the next robot segment. The pair (x_1, y_1) denotes the coordinate axes of the usual cartesian coordinate system in $\mathbb{R}^2$. The coordinate vector x_2 points along the first robot segment with y_2 complementing it to an orthogonal coordinate system and so on. The origin of every coordinate system (x_i, y_i) for $i > 1$, is located at the joint-position of joint $i-1$. Note that in this case y_1 and x_2 coincide, as the robot's first joint makes precisely a $\pi/2$ rotation and the robot starts in the origin. The joint angles θ_i denote the angles between successive robot segments, i.e., θ_i is the angle between x_i and x_{i+1}. The first angle θ_1 is then the angle between the x-axis in $\mathbb{R}^2$ and the first robot segment. Angles are measured counter-clockwise. This means that in this figure, θ_1 is positive, while θ_2 and θ_3 are negative. Lengths of the segments are denoted by l_i.

we will assume that the first segment is fixed in the origin of $\mathbb{R}^2$ (on the ground) and that the "hand" or "tool" is attached to the last joint and can freely rotate (so we only care about the position of the last joint and not its orientation). The *configuration space* $\mathcal{C}$ of the robot is then simply the set of possible hand positions: $\mathcal{C} \subset \mathbb{R}^2$.

Planar revolute joints are parametrized by an angle θ_i as seen in Figure 1. The set of all possible robot poses can then be parametrized by the vector space product of all joint-rotations, which we will call the *joint space* $\mathcal{J}$:

$$\mathcal{J} = \underbrace{\mathcal{S}^1 \times \cdots \times \mathcal{S}^1}_{n-1},$$

where n denotes the number of joints and $\mathcal{S}^1$ is the torus $[0, 2\pi)$.

The *Forward Kinematic Problem* is concerned with determining the positions of the robot's joints, in particular the last one, given a robot's joint angles $\{\theta_i\}_{i=1}^{n-1}$; i.e., we are interested

in finding a function

$$F : \mathcal{J} \to \mathcal{C} \tag{1}$$

Under the given assumptions, F takes on a particularly simple form (see Appendix A for a derivation):

$$F(\theta) = \begin{pmatrix} \sum_{i=1}^{n-1} l_i \cos\left(\sum_{j=1}^{i} \theta_j\right) \\ \sum_{i=1}^{n-1} l_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \end{pmatrix} \tag{2}$$

Note that the position of the k'th joint can be calculated by the same expression by simply summing up only from $i = 1$ to $k - 1$.

2 First Task - Forward Kinematics

Convince yourself that Equation (2) is correct.

1 Implement a Matlab function that, for given lengths $\{l_i\}_{i=1}^{n-1}$ and joint angles $\{\theta_i\}_{i=1}^{n-1}$, calculates the robot's n joint positions in global cartesian coordinates. Test the function for the following cases:

1. $l = (1, 1)$, $\theta = (\frac{\pi}{2}, 0)$
2. $l = (1, 1)$, $\theta = (\frac{\pi}{2}, \frac{\pi}{2})$
3. $l = (1, 2, 2, 1)$, $\theta = (\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2}, -\frac{\pi}{2})$

Write a plotting function to visualize the robot for a given configuration!

2 Sketch the set $\mathcal{C}$ of all reachable positions with a 2-joint robot with joint lengths l_1 and l_2.

3 Second Task - Inverse Kinematics 1

We now turn to the *inverse* problem to the forward kinematics just considered: *given a target point $t \in \mathcal{C}$, find a joint-configuration $\theta \in \mathcal{J}$ such that $F(\theta) = t$.*

Note that this is an ill-posed problem: A solution is not necessarily unique, and does not necessarily depend continuously on the position t. More generally we're just given a point $t \in \mathbb{R}^2$, so we don't even know if a solution exists at all, so we will consider the more general problem: Given $t \in \mathbb{R}^2$

$$\min_{\theta \in \mathcal{J}} d(\theta) := \frac{1}{2} \|t - F(\theta)\|_{\mathbb{R}^2}^2 \tag{3}$$

.

For now we will assume that there are no constraints on the joint rotations.

3 Calculate the gradient of d in (3).

4 Implement a steepest descent method to solve Problem (3). Note that you will need to use a line search method to determine the optimal step size, for example the backtracking line search from Exercise 3. Use the following settings:

1. $l = (1, 1)$, $t = (0, 2)$, $\theta^0 = (0, 0)$
2. $l = (1, 1)$, $t = (0, 1)$, $\theta^0 = (0, 0)$
3. $l = (1, 1)$, $t = (0, 1)$, $\theta^0 = (\pi, 0)$
4. $l = (2, 2, 1, 1, 1)$, $t = (2, 4)$, $\theta^0 = (0, 0, 0, 0, 0)$

where θ^0 is the starting point for the iteration. You can also use your own settings, but have at least one test case with at least 5 joints.

Count the number of iterations it takes to converge to a reasonable solution (choose suitable abort criteria).

Also find an example where the robot can't reach the target position.

Hint: You will need to use small step sizes.

4 Third Task - Inverse Kinematics 2

5 Calculate the Hessian of d in (3). Note that the Hessian of F is a 3-dimensional matrix!

6 Based on your code from task 4 and the expressions you derived in task 5, implement now a Newton method solver for Problem (3). Use the same linesearch algorithm.

In the inverse kinematics problem, the Hessian will often be indefinite, i.e, we will find many saddle points. (Why?) In order to still use the Newton method under these circumstances, we will use a modified method as specified in algorithm 3.2 ("Line Search Newton with Modification") in your textbook. Use algorithm 3.3, ("Cholesky with Added Multiple of the Identity") from your textbook to determine the perturbation of the Hessian. You can use Matlab's chol function.

We will make one more modification to the standard Newton method: Begin your optimization with 10 normal gradient descent (with backtracking) steps before using the Newton method. Such an *initialization* procedure can prevent the Newton method from too quickly convering to a local minimum close to the iteration's starting point.

Use the same robot and target settings as in task 4.

Try using a larger step size compared to the gradient descent method. Compare the number of iterations with the gradient descent method.

5 Fourth Task - Constraints

7 Sketch the set $\mathcal{C}$ of all reachable positions with a 2-joint robot with joint lengths l_1 and l_2, where we restrict the joint angles θ_1 and θ_2 to lie within the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

8 Set up the KKT-conditions for

$$\min_{\theta \in \mathcal{J}} d(\theta) := \frac{1}{2}\|t - F(\theta)\|^2_{\mathbb{R}^2} \tag{4}$$

$$\alpha_i \leq \theta_i \leq \beta_i, \qquad i = 1, \ldots, n-1, \tag{5}$$

for given bounds $\{\alpha_i\}_{i=1}^{n-1}$ and $\{\beta_i\}_{i=1}^{n-1}$.

Do the Lagrange multipliers have any special meaning in this setting?

9 Modify your algorithm from task 4 or 6, to enforce the following constraints on the joint angles via a projection method as discussed in class (see also the lecture notes no. 13 from the course webpage).

Use the following settings: $l = (2, 2, 1, 1, 1)$, $t = (2, 4)$, $\theta_0 = (0, 0, 0, 0, 0)$. We will always use the constraint $0 \leq \theta_1 \leq \pi$. Compare four different cases:

1. $-\frac{\pi}{2} \leq \theta_i \leq \frac{\pi}{2}$, $i \in 2, \ldots, n-1$
2. $-\frac{\pi}{4} \leq \theta_i \leq \frac{\pi}{4}$, $i \in 2, \ldots, n-1$
3. $-\frac{\pi}{16} \leq \theta_i \leq \frac{\pi}{16}$, $i \in 2, \ldots, n-1$

10 (Optional) How would you modify Problem (4) to prevent the robot from "crashing into the ground", i.e., intersecting the x-axis? Use the matlab optimization toolbox to develop an inverse kinematics function that solves this problem.

A Forward Kinematics

With the assumptions specified in the introduction, we can find a particularly simple expression for F. We start by introducing a set of rectangular coordinate systems, starting with the global cartesian system (x_1, y_1), with joint 1 located in the origin. For every joint i we set up a local rectangular coordinate system (x_{i+1}, y_{i+1}) with the origin in joint i, oriented such that the positive x_i axis lies along the direction of the i'th robot segment. We do not set up a coordinate system for the last joint. See Figure 1 for an example of our notation.

The hand obviously has (x_n, y_n) coordinates $(l_{n-1}, 0)$, but we are interested in its (x_1, y_1) coordinates.

We can calculate this recursively. Given a point q in the plane with (x_i, y_i) and (x_{i-1}, y_{i-1}) coordinates (a_i, b_i) and (a_{i-1}, b_{i-1}) respectively, we get

$$\begin{pmatrix} a_{i-1} \\ b_{i-1} \end{pmatrix} = \begin{pmatrix} \cos\theta_{i-1} & -\sin\theta_{i-1} \\ \sin\theta_{i-1} & \cos\theta_{i-1} \end{pmatrix} \begin{pmatrix} a_i \\ b_i \end{pmatrix} + \begin{pmatrix} l_{i-2} \\ 0 \end{pmatrix},$$

where θ_{i-1} is the counterclockwise angle from the x_{i-1} axis to the x_i axis. We define $l_0 = 0$.

This can be written as

$$\begin{pmatrix} a_{i-1} \\ b_{i-1} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\theta_{i-1} & -\sin\theta_{i-1} & l_{i-2} \\ \sin\theta_{i-1} & \cos\theta_{i-1} & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{A_{i-1}} \begin{pmatrix} a_i \\ b_i \\ 1 \end{pmatrix}$$

We then get for the hand position

$$\begin{aligned} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} &= A_1 A_2 \dots A_{n-1} \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} \\ &= A_1 A_2 \dots A_{n-1} \begin{pmatrix} l_{n-1} \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^{n-1} l_i \cos\left(\sum_{j=1}^{i} \theta_j\right) \\ \sum_{i=1}^{n-1} l_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \\ 1 \end{pmatrix} \end{aligned}$$

or simply:

$$F(\theta) = \begin{pmatrix} \sum_{i=1}^{n-1} l_i \cos\left(\sum_{j=1}^{i} \theta_j\right) \\ \sum_{i=1}^{n-1} l_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \end{pmatrix}$$