Robotics: Science and Systems

Optimization II

Zhibin Li
School of Informatics
University of Edinburgh

Content

- Constrained optimization
  - Constrained least square
  - Quadratic Optimization (QP)
- *Nonlinear optimization
  - Application for dynamic walking
  - Application for task space planning of a robot arm

Constrained linear least squares

Least squares with constraints

Simple least squares with equality constraints:

$$\text{minimize } \|\mathbf{Ax-b}\|^2$$

$$\text{subject to } \mathbf{C}_{eq}\mathbf{x=d}$$

where $\|\mathbf{Ax-b}\|^2$ is the objective function, and $\mathbf{Cx=d}$ is the equality constraint.

Additional constraints may also exist:

Inequality constraint, must satisfy $\mathbf{C}_{ineq}\mathbf{x} \leq \boldsymbol{\alpha}$

The vector $\mathbf{x}$ must satisfy the vector inequalities $lb \leq \mathbf{x} \leq ub$, *lb* and *ub* are lower and upper bounds.

Matlab exercise

Function: **lsqlin**

Solve constrained linear least-squares problems with bounds or linear constraints.

Exercise with the examples here.

Comparison with Tikhonov regularisation

Constrained Least Squares explicitly specifies the constraints, while Tikhonov regularisation only minimize the deviation from a *nominal*.

When the range of solution is large, and boundary limits are far away to “hit”, both may yield the same results. However, if constraints are *hard*, then explicit constraints are necessary.

Quadratic Programming (QP)

Quadratic Programming (QP)

QP is quadratic optimization problem that optimizes (minimizes or maximizes) a quadratic function of variables subject to linear constraints on these variables.

The objective of QP is to find vector $\mathbf{x} \in \mathbf{R}^n$, that will

$$
\begin{aligned}
\min \quad & 1/2 \cdot \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\
\text{st} \quad & \mathbf{A} \cdot \mathbf{x} \le \mathbf{b} \\
& \mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq} \\
& lb \le \mathbf{x} \le ub
\end{aligned}
$$

where $\mathbf{A} \cdot \mathbf{x} \le \mathbf{b}$ means that every element of the vector $\mathbf{A} \cdot \mathbf{x}$ is less than or equal to the corresponding element of vector $\mathbf{b}$. (All vectors are column vectors by default)

Quadratic Programming (QP)

The objective of QP is to find vector $\mathbf{x} \in \mathbf{R}^n$, that will

$$\begin{aligned} &\text{minimize } 1/2 \cdot \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ &\text{subject to } \mathbf{A} \cdot \mathbf{x} \le \mathbf{b} \\ &\qquad \mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq} \\ &\qquad lb \le \mathbf{x} \le ub \end{aligned}$$

c: m-dimensional real number column vector
H: n × n Hessian matrix, symmetric* and positive (semi-)definite
A, $\mathbf{A}_{eq}$: m × n real matrix A
b, $\mathbf{b}_{eq}$: m-dimensional real number column vector

* In linear algebra, a symmetric matrix is a square matrix that is equal to its transpose, so $\mathbf{H} = \mathbf{H}^T$.

Quadratic Programming (QP)

Note that every equality constraint $\mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq}$ can be equivalently replaced by a pair of inequality constraints

$$\mathbf{A}_{eq} \cdot \mathbf{x} \le \mathbf{b}_{eq}$$

$$-\mathbf{A}_{eq} \cdot \mathbf{x} \le \mathbf{b}_{eq}$$

Therefore, equality constraints are somewhat redundant mathematically. However, the concept of equality constraints are useful when it maps the real world problems into mathematical formulations.

Quadratic Programming (QP)

Common methods for QP solutions:

1. Interior-point
2. Active-set
3. Trust-region-reflective

Matlab: quadprog

C++: qpOASES

LS and QP

Non-negative least squares (NNLS) is a constrained version of the least squares problem. The NNLS problem is equivalent to a quadratic programming problem.

$$1/2\|\mathbf{Ax}-\mathbf{b}\|^2$$

$$=?$$

An example how to transform a minimization problem (LS) into a QP form.

Quiz: whiteboard exercise: quadratic multiplication of vectors.

LS and QP

Non-negative least squares (NNLS) is a constrained version of the least squares problem. The NNLS problem is equivalent to a quadratic programming problem.

$$1/2\|\mathbf{Ax}-\mathbf{b}\|^2$$

$$=1/2(\mathbf{Ax}-\mathbf{b})^T(\mathbf{Ax}-\mathbf{b})$$

$$=1/2(\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-\mathbf{x}^T\mathbf{A}^T\mathbf{b}-\mathbf{b}^T\mathbf{Ax}+\mathbf{b}^T\mathbf{b})$$

$$=1/2(\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-2\mathbf{c}^T\mathbf{Ax}+\mathbf{b}^T\mathbf{b})$$

Note that $(\mathbf{Ax})^T\mathbf{b}=\mathbf{b}^T(\mathbf{Ax})$, since this is dot product of two vectors: multiply corresponding elements of each element, then add up the products. The result is a scalar value.

LS and QP

$$1/2\|\mathbf{Ax}-\mathbf{b}\|^2$$

$$=1/2(\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-2\mathbf{b}^T\mathbf{Ax}+\mathbf{b}^T\mathbf{b})$$

$$=1/2\cdot\mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x} - (\mathbf{b}^T\mathbf{A})\cdot\mathbf{x} + 1/2\cdot\mathbf{b}^T\mathbf{b}$$

Since $\mathbf{b}^T\mathbf{b}$ is a non-negative quantity, so minimizing $1/2\|\mathbf{Ax}-\mathbf{b}\|^2$ is equal to minimizing $1/2\cdot\mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x} - (\mathbf{b}^T\mathbf{A})\cdot\mathbf{x}$, treating $\mathbf{b}^T\mathbf{b}$ as a non-negative offset.

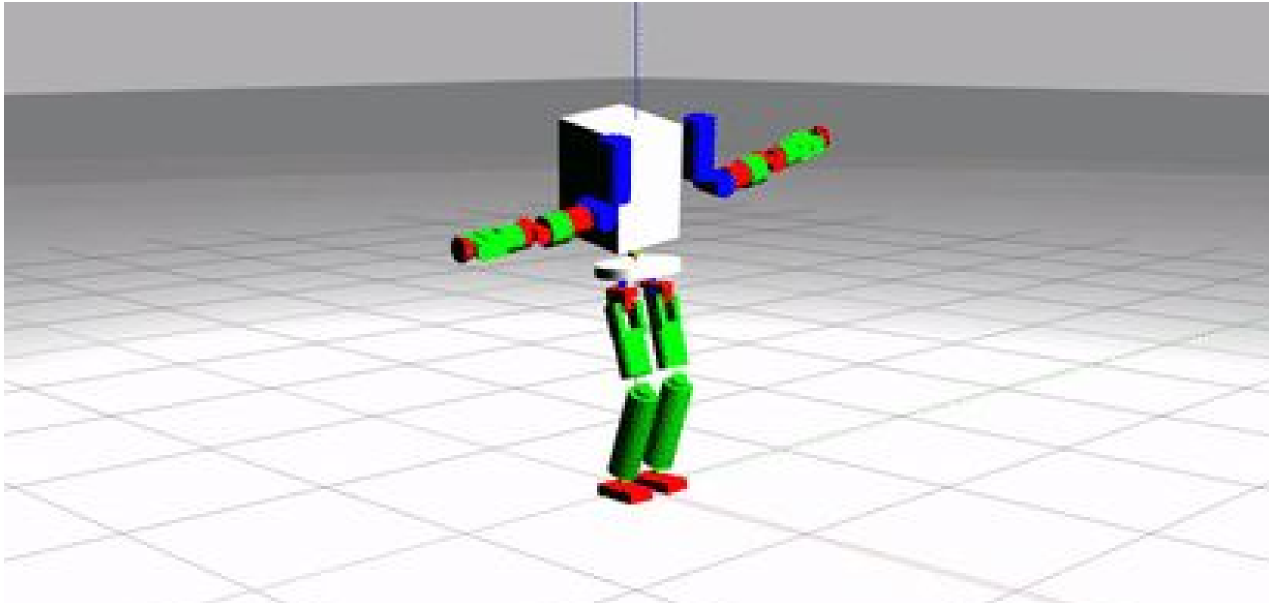
Hence, it is equivalent to a QP problem:

$$f(\mathbf{x})=1/2\cdot\mathbf{x}^T\mathbf{Hx} + \mathbf{c}^T\cdot\mathbf{x}$$

where $\mathbf{H}=(\mathbf{A}^T\mathbf{A})$ and $\mathbf{c}^T=-\mathbf{b}^T\mathbf{A}$.

Applications of QP

Using optimization to coordinate all joints (whole body control) of a humanoid robot. (Detailed formulation will be covered by Robot Learning and Sensorimotor Control, semester 2)



*Nonlinear optimization

*Nonlinear optimization

Often, there are problems that are nonlinear, thus require nonlinear optimizers.

Cost function ***J*** can be more complex, and are not necessarily the norm of vectors, eg forward kinematics.

- Matlab: optimization toolbox has solvers for linear, quadratic, integer, and nonlinear optimization problems. For example, fmincon is a very powerful tool, if the problem is well formulated, usually fmincon can find the solution.
- C++: NLopt

Matlab functions for unconstrained optimization

Functions

1. fminsearch: Find minimum of unconstrained multivariable function using derivative-free method
2. fminunc: Find minimum of unconstrained multivariable function

Note: these are actually nonlinear programming solvers.

Matlab functions for constrained optimization

Functions

1. fminbnd: find minimum of single-variable function on fixed interval
2. fmincon: find minimum of constrained nonlinear multivariable function
3. fseminf: find minimum of semi-infinitely constrained multivariable nonlinear function

Formulating constraints: QP case

Usually qpOASES expects QPs to be formulated in the following standard form:

$$
\begin{aligned}
\min_{x} \quad & \tfrac{1}{2} x^T H x + x^T g(w_0) \\
\text{s.t.} \quad & \mathrm{lb}A(w_0) \leq Ax \leq \mathrm{ub}A(w_0)\,, \\
& \mathrm{lb}(w_0) \leq \ x \ \leq \mathrm{ub}(w_0)\,,
\end{aligned}
$$

where we can express the equality constraint by setting **lbA** and **ubA** the same.

Formulating constraints: fmincon case

However, fmincon has different interface or formalization of the constraints, as shown below:

fmincon R2017b

Find minimum of constrained nonlinear multivariable function collapse all in page

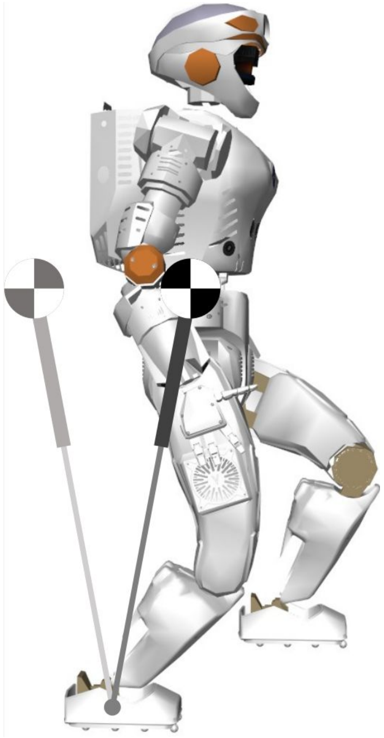
Nonlinear programming solver.

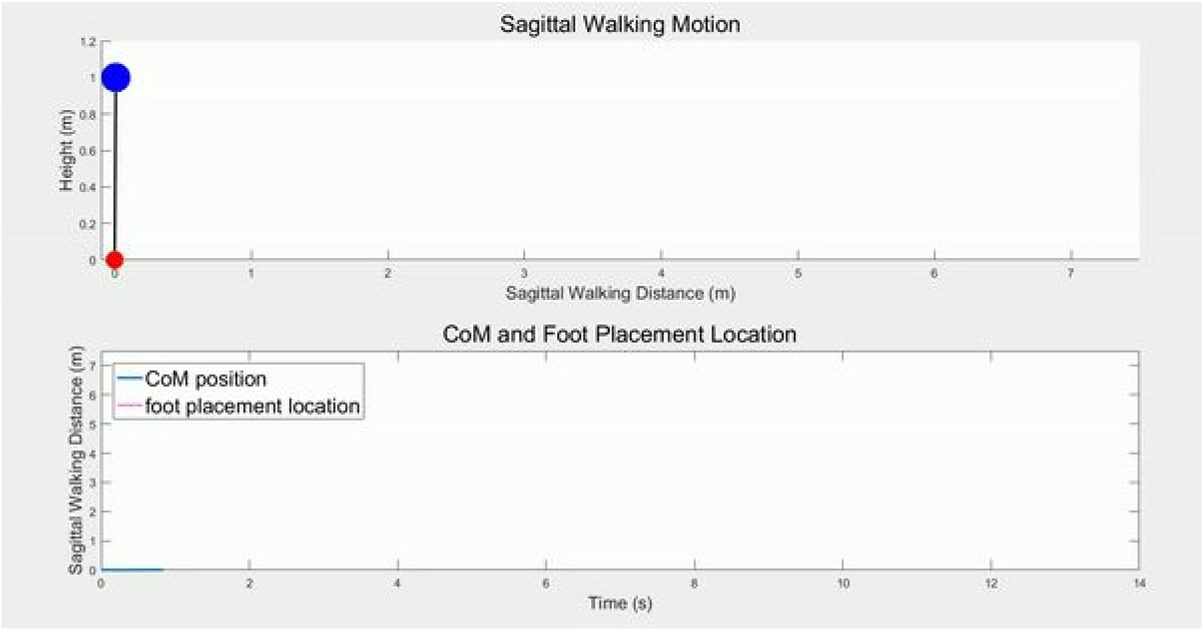
Finds the minimum of a problem specified by

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub, \end{cases}$$

Quiz: how to formulate inequality constraints **lbA** ≤ **A·x** ≤ **ubA** in the form **A·x** ≤ **b**?

Application of nonlinear optimization (fmincon)





Task space control using optimization

Question: how to formulate inverse kinematics problem considering the constraints, eg joint angle limit?

Problem Formulation?

Application of nonlinear optimization (fmincon)

Fmincon example: task space control, eg inverse kinematics, using optimization

Problem Formulation

Recall Tikhonov regularization:

$$\boldsymbol{f}(\mathbf{x}) = \| \mathbf{A}\,\mathbf{x} - \mathbf{b} \|_{\mathbf{P}}^{2} + \| \mathbf{x} - \mathbf{x}_{0} \|_{\mathbf{Q}}^{2},$$

Our problem formulation of inverse kinematics problem considering the constraints is somewhat similar.

Application of nonlinear optimization (fmincon)

Problem Formulation

$$f(\mathbf{x}) = \| \mathbf{F(q)} - \mathbf{y}^{\mathbf{d}} \|_{\mathbf{Qp}}^{2} + \| \mathbf{O(q)} - \mathbf{vec}^{\mathbf{d}} \|_{\mathbf{Qr}}^{2} + \| \mathbf{q} - \mathbf{q}_0 \|_{\mathbf{Qj}}^{2},$$

$$\mathbf{q}_{\mathbf{min}} \leq \mathbf{q} \leq \mathbf{q}_{\mathbf{max}}$$

q: joint configuration

$\mathbf{q}_0$, $\mathbf{q}_{\mathbf{min}}$, $\mathbf{q}_{\mathbf{max}}$: initial, minimum, maximum joint configurations

F(q), O(q): forward kinematics, returns end effector position and orientation

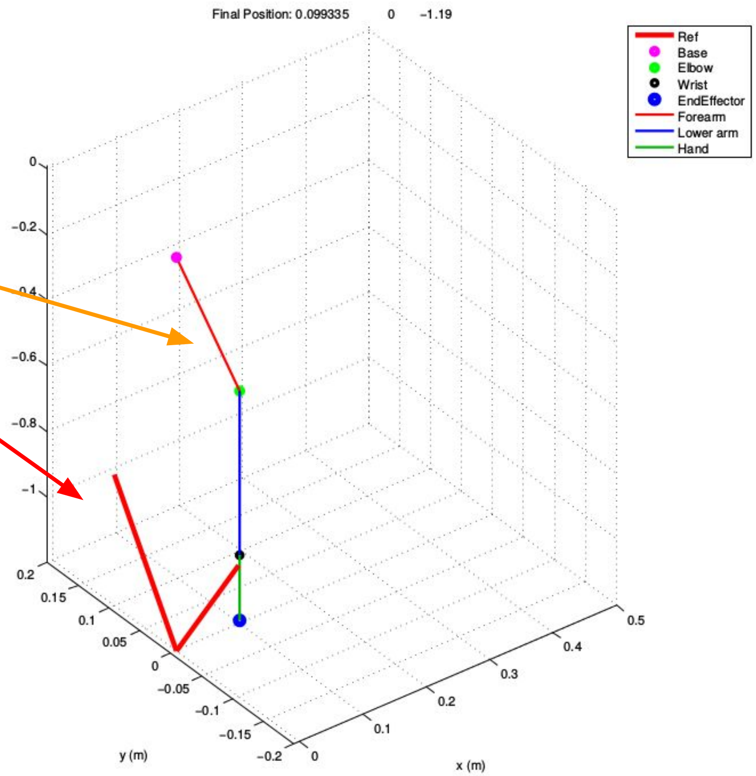
$\mathbf{y}^{\mathbf{d}}$, $\mathbf{vec}^{\mathbf{d}}$: desired task space or workspace position, orientation

$\mathbf{Q}_{\mathbf{p}}$, $\mathbf{Q}_{\mathbf{r}}$, $\mathbf{Q}_{\mathbf{j}}$ are the weighting matrices for workspace position, rotation, and joint/configuration space respectively.

Application of nonlinear optimization (fmincon)

Initial configuration of robot arm at $\mathbf{q}_0$

V shape is the workspace trajectory



Application of nonlinear optimization (fmincon)

Optimization only over end-effector position and minimum change of joint angles.

$$f(\mathbf{x}) = \|\, \mathbf{F(q)} - \mathbf{y}^{\mathbf{d}} \,\|_{\mathbf{Qp}}^2 + \|\, \mathbf{q} - \mathbf{q}_0 \|_{\mathbf{Qj}}^2$$

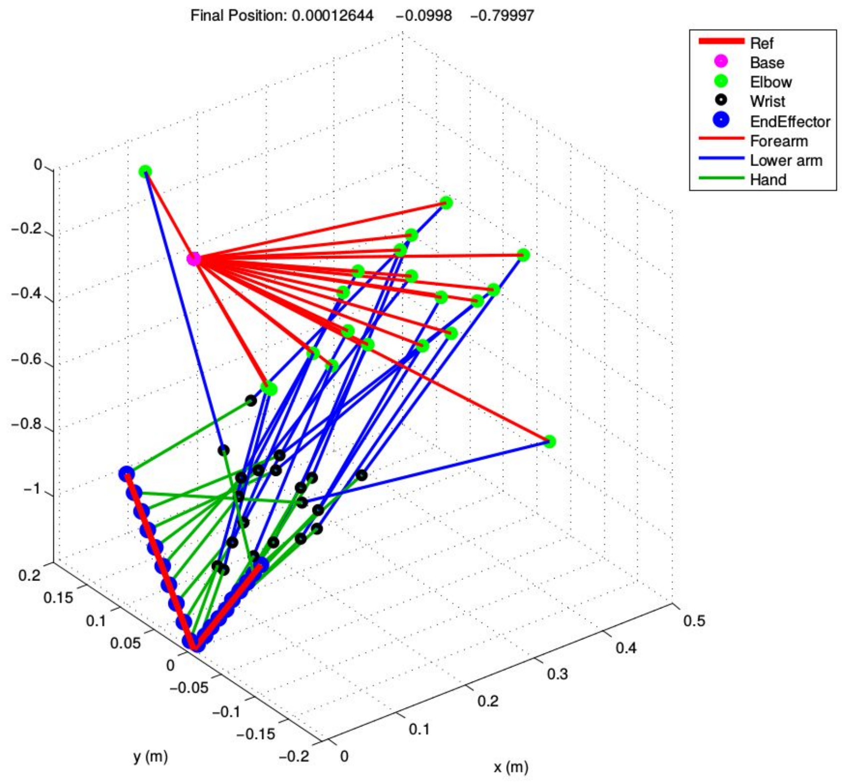
$\mathbf{Q_p}$ = 1;

$\mathbf{Q_r}$ = 0; (orientation is NOT controlled)

$\mathbf{Q_j}$ = 1e-6;

$\mathbf{q_{min}}$: -pi/2

$\mathbf{q_{max}}$: pi/2



Application of nonlinear optimization (fmincon)

Optimization only over end-effector position and minimum change of joint angles.

$$f(\mathbf{x})=\| \mathbf{F(q)} - \mathbf{y}^d \|_{\mathbf{Qp}}^2 + \| \mathbf{q} - \mathbf{q}_0\|_{\mathbf{Qj}}^2$$

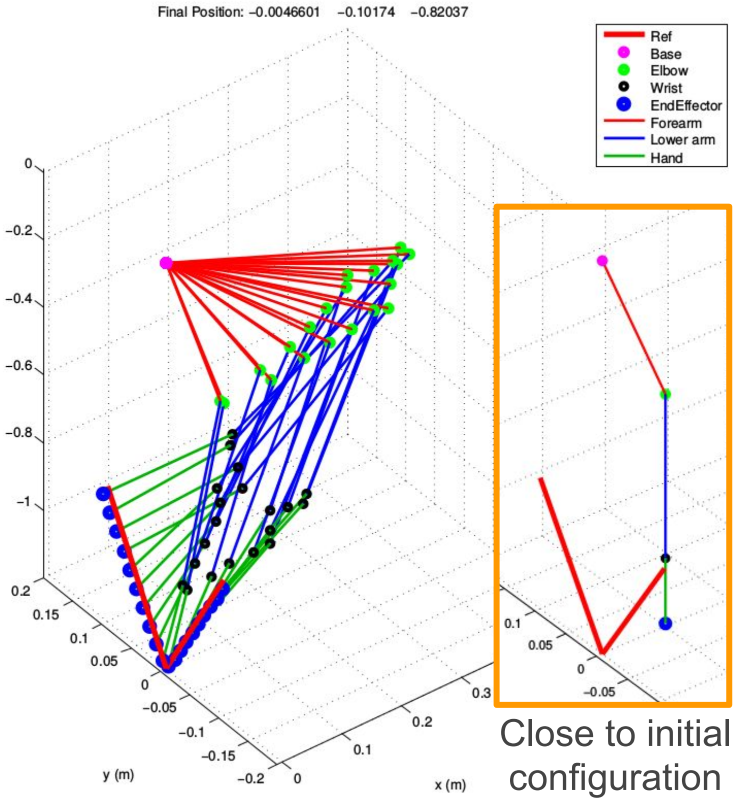
$\mathbf{Q}_p$ = 1;

$\mathbf{Q}_r$ = 0; (orientation is NOT controlled)

$\mathbf{Q}_j$ = 5e-3; ⇐ Increase weight for $\mathbf{q}_0$

$\mathbf{q}_{min}$: -pi/2

$\mathbf{q}_{max}$: pi/2



Close to initial configuration

Application of nonlinear optimization (fmincon)

Optimization of all terms:

$$f(\mathbf{x}) = \| \mathbf{F(q)} - \mathbf{y}^d \|_{\mathbf{Qp}}^2 + \| \mathbf{O(q)} - \mathbf{vec}^d \|_{\mathbf{Qr}}^2$$

$$+ \| \mathbf{q} - \mathbf{q}_0 \|_{\mathbf{Qj}}^2,$$

$\mathbf{Q_p}$ = 1;

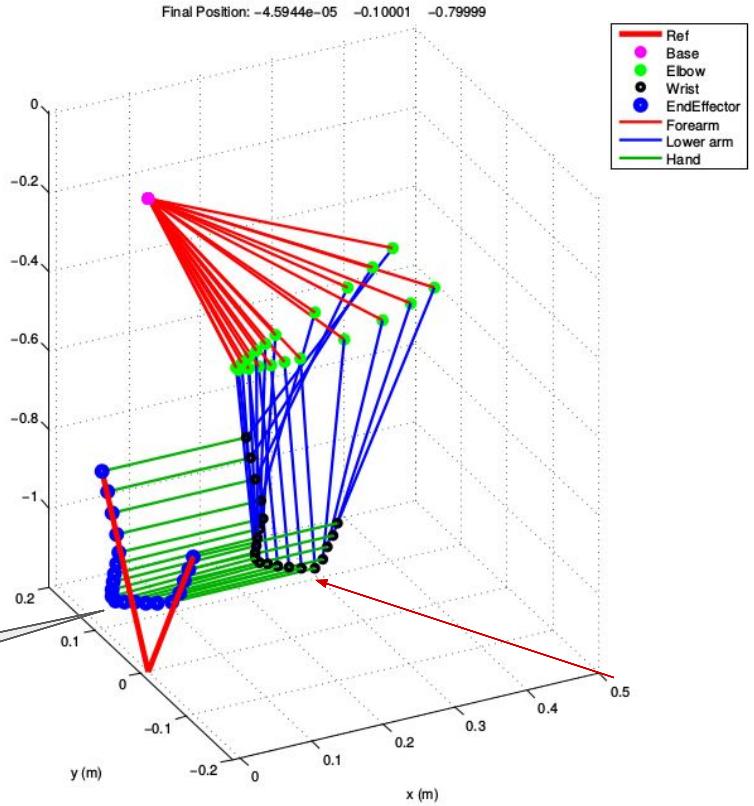
$\mathbf{Q_r}$ = 1; (note: orientation is controlled)

$\mathbf{Q_j}$ = 1e-6;

$\mathbf{q_{min}}$: -pi/2

$\mathbf{q_{max}}$: pi/2

What is the problem here?



Application of nonlinear optimization (fmincon)

Optimization of all terms:

$$f(\mathbf{x}) = \| \mathbf{F(q)} - \mathbf{y^d} \|_{\mathbf{Qp}}^2 + \| \mathbf{O(q)} - \mathbf{vec^d} \|_{\mathbf{Qr}}^2$$

$$+ \| \mathbf{q} - \mathbf{q}_0 \|_{\mathbf{Qj}}^2,$$

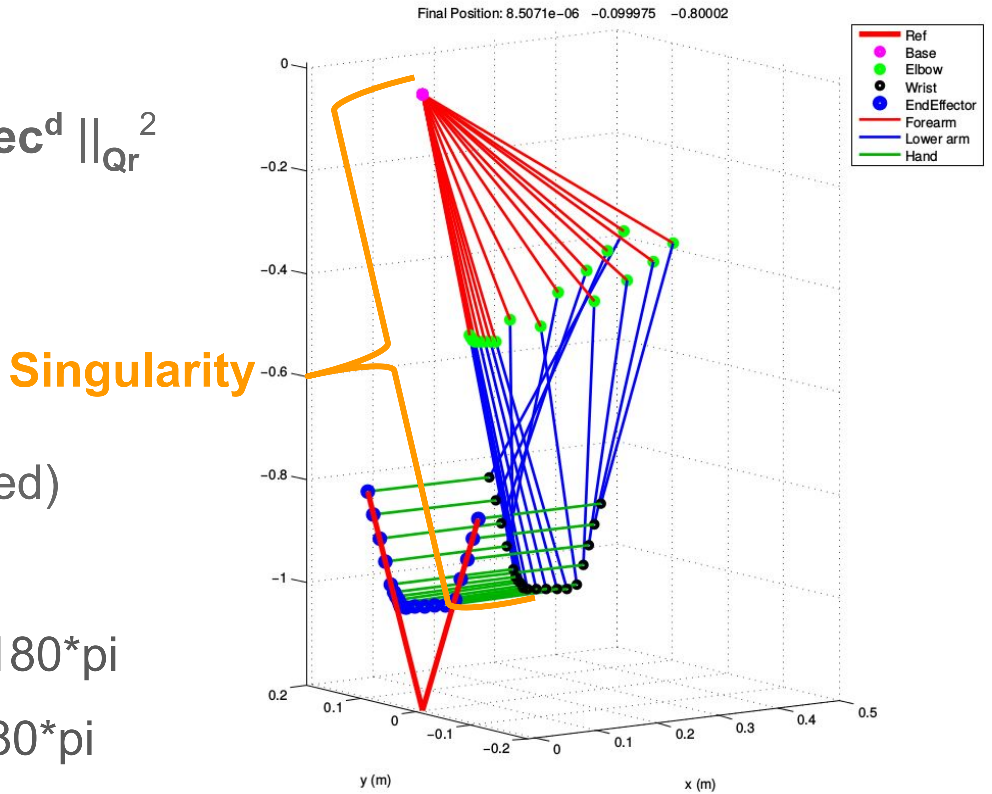
$\mathbf{Q_p}$ = 1;

$\mathbf{Q_r}$ = 1; (note: orientation is controlled)

$\mathbf{Q_j}$ = 1e-6;

$\mathbf{q_{min}}$: [-150; -90; -90; 0; -90; -150]/180*pi

$\mathbf{q_{max}}$: [150; 90; 90; 150; 90; 150]/180*pi



Application of nonlinear optimization (fmincon)

Same setting for optimization, only change the task space **V** trajectory to be *within* workspace.

The optimization scheme works nicely, considering the minimum deviation from its home position, minimum error of pose (position & orientation), as well as the joint limits!