Redundancy resolution based on optimization

Jane Li

Assistant Professor
Mechanical Engineering Department, Robotic Engineering Program
Worcester Polytechnic Institute
Quiz (10 pts)

• (6 pts) Explain the optimization/tradeoff underlying the damped least square method

\[
\min_{\dot{\mathbf{q}}} \frac{\mu^2}{2} \|\dot{\mathbf{q}}\|^2 + \frac{1}{2} \|\mathbf{x} - J\dot{\mathbf{q}}\|^2 = H(\dot{\mathbf{q}})
\]

• (2 pts) List two metrics that measure the distance from singularity

• (2 pts) How to guarantee the secondary task will not interfere the primary task
To render robust behavior when crossing the singularity, we can add a small constant along the diagonal of \((J(q)^T J(q))\) to make it invertible when it is singular.
Distance to singularity

• Manipulability index – Jacobian matrix determinant

\[ \mu = \sqrt{|J J^T|} \]

• Which is indeed

\[ \mu = \prod_{i=1}^{M} \sigma_i \]

• Is it a good measurement?
Distance to singularity

- Manipulability index – condition number
  \[ \kappa = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \]

- Alternatively, can use isotropy
  \[ \text{Isotropy} = \frac{\sigma_{\text{min}}}{\sigma_{\text{max}}} \]

- Is it good enough?
Distance to singularity

- Manipulability index – the smallest singular value
  \[ \sigma_{\text{min}} \]
- Direction of velocity disadvantage
- Is it good enough?
Distance to singularity

• Manipulability index

\[ \mu' = \sum_{i=1}^{M} \sqrt{|J_i J_i^T|} \]

• What does it imply?
  • Manipulability of every sub-manipulator (non-redundant)
The Null-space of Jacobian

- Secondary tasks is satisfied in the **null-space** of the Jacobian pseudo-inverse

- In linear algebra, the **null-space** of a matrix \( A \) is the set of vectors \( V \) such that, for any \( v \) in \( V \), \( 0 = A^Tv \).

- \( V \) is orthogonal to the range of \( A \)
The Null-space of Jacobian

- Given the null space of Jacobian, the secondary task will not disturb the primary task

- The null-space projection matrix for the Jacobian pseudo-inverse is:

\[ N(q) = I - J(q)\dagger J(q) \]
The Null-space of Jacobian

- Project a **task space velocity vector** into the null-space

\[
\dot{q} = J(q)^\dagger \dot{x} + (I - J(q)^\dagger J(q)) J_c(q)^\dagger \dot{x}_c
\]

Primary task

Secondary task
Redundancy resolution based on optimization
Still a problem ...

- Methods for redundancy resolution has been studied for decades, yet there are still unsolved problems

- Multi-objective Optimization
  - What are the optimization criteria?
  - How to assign weighting coefficients?
Robot manipulator – Performance to optimize

- Manipulability
- Force/velocity transmission efficiency
- Energy
- Motion smoothness
- Task accuracy

<table>
<thead>
<tr>
<th>Performance indices</th>
<th>Formula</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determinant of Jacobian (1984)</td>
<td>$w_n = \sqrt{JJ^T}$</td>
<td>Uniformity of the torque-velocity gain</td>
</tr>
<tr>
<td>Condition number (1982)</td>
<td>$\kappa = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}$</td>
<td>Variance in velocity/force transmission</td>
</tr>
<tr>
<td>Isotropy (1987)</td>
<td>$I_{so} = \frac{\sigma_{\text{min}}}{\sigma_{\text{max}}}$</td>
<td>same as condition number</td>
</tr>
<tr>
<td>Min eigen-value of Jacobian (1987)</td>
<td>$I_{so} = \frac{\sigma_{\text{min}}}{\sigma_{\text{max}}}$</td>
<td>Efficiency of force/velocity transmission</td>
</tr>
<tr>
<td>Dynamic Manipulability (1985)</td>
<td>$G = J^{-T}MJ^{-1}$</td>
<td>Uniformity of this torque-acceleration gain</td>
</tr>
<tr>
<td>Distance from singularity (1987)</td>
<td>$H = \left</td>
<td>\prod_{i} \Delta_i \right</td>
</tr>
<tr>
<td>Acceleration radius (1988)</td>
<td>$\tau = M(\theta) + C(\theta, \dot{\theta})\dot{\theta}$</td>
<td>acceleration capability of the end-effector</td>
</tr>
<tr>
<td>Force transmission ratio (1988)</td>
<td>$\alpha = [(u^TJJ^Tu)^{1/2}]$</td>
<td>Force gain along task-compatibility direction</td>
</tr>
<tr>
<td>Velocity transmission ratio (1988)</td>
<td>$\beta = [u^T(JJ^T)u]^{1/2}$</td>
<td>Velocity along task-compatibility direction</td>
</tr>
<tr>
<td>Min Jerk model (1984)</td>
<td>$\min(\frac{\partial^3x}{\partial t^3})$</td>
<td>Motion smoothness</td>
</tr>
<tr>
<td>Min (commanded) torque-change (1985,1989)</td>
<td>$\min(\frac{\partial T}{\partial x})$</td>
<td>Motion smoothness</td>
</tr>
<tr>
<td>Min work model (1983)</td>
<td>$\min(W)$</td>
<td>Energy</td>
</tr>
<tr>
<td>Min variance model (1989)</td>
<td>$\min[\text{var}(x - x_d)]$</td>
<td>Task accuracy</td>
</tr>
</tbody>
</table>
Common Objectives for Redundant Resolution

- Tracking end-effector trajectory → primary task

- Obstacle avoidance
  - Pseudoinverse – Incorporate obstacle as secondary constraints
  - Artificial potential field – repulsive obstacle + attractive target

- Motion limits
  - Position, velocity, acceleration
  - Avoid vibration, improve motion smoothness
To be consistent and predictable, robot motion needs to be repetitive in both task and configuration space

- Close path in task space $\rightarrow$ close path in configuration space

Unpredictable robot behavior

- Joint angle drift
- Readjusting the manipulators' configuration with self-motion at every cycle $\rightarrow$ inefficient
Methods

- Baseline = Closed-loop pseudo-inverse

- Define a cost function to optimize for motion repetition, and solve it using
  - Genetic Algorithm [1]
  - Dynamical quadratic programming [2]

- Continuous pseudo-inverse and global redundancy resolution [3]
Closed-loop pseudo-inverse

• Compute the joint position through time integration pseudo-inverse

\[ \Delta q = J^\dagger \Delta x \]

Unpredictable, not repeatable arm configurations
Closed-loop pseudo-inverse + Genetic Algorithm

\[ \Delta q = J^\dagger \Delta x \]

\[ \Delta x^* = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_m \\ \Delta x_{m+1} \\ \vdots \\ \Delta x_n \end{bmatrix} \]

Generated by GA

\[ J^* = \begin{bmatrix} -l_1S_1 & -l_2S_2 & \cdots & -l_nS_n \\ l_1C_1 & l_2C_2 & \cdots & l_nC_n \end{bmatrix} \]

\[ \begin{bmatrix} j_{(m+1)1} \\ \vdots \\ j_{(m+1)n} \\ \vdots \\ j_{n1} \\ \vdots \\ j_{nn} \end{bmatrix} \]

1. Begin
2. \( T = 0 \)
3. calculate \( \Delta x = x_{\text{ref}} - x_{\text{ref}} \cdot J \)
4. initialize random population
5. \( P(T) = \left[ \begin{bmatrix} J^{(1)} : \Delta x^{(1)} \end{bmatrix}, \ldots, \begin{bmatrix} J^{(N)} : \Delta x^{(N)} \end{bmatrix} \right] \)
6. get \( \Delta q = J^{-1} (q) \Delta x^* \) and \( q = f \Delta q \)
7. evaluate \( P(T) \)
8. repeat
9.
10.
11.
12. get \( \Delta q = J^{-1} (q) \Delta x^* \) and \( q = f \Delta q \)
13. evaluate \( P(T) \)
14. \( T = T + 1 \)
15. until termination condition is TRUE
16. get new \( q \)
17. End

Use GA to update \( P(T) \)

Cost function?
Weighted least squares

Minimize the displacement between initial and current joint configurations over a time step
## Simulation Result

<table>
<thead>
<tr>
<th></th>
<th>CLGA $r = 0.7$</th>
<th>CLGA $r = 1.0$</th>
<th>CLGA $r = 2.0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3R$</td>
<td>9.96E–04</td>
<td>8.84E–04</td>
<td>1.08E–03</td>
</tr>
<tr>
<td>$4R$</td>
<td>7.12E–04</td>
<td>7.38E–04</td>
<td>5.70E–04</td>
</tr>
<tr>
<td>$5R$</td>
<td>6.73E–04</td>
<td>5.42E–04</td>
<td>6.15E–04</td>
</tr>
<tr>
<td>$6R$</td>
<td>5.98E–04</td>
<td>4.81E–04</td>
<td>8.57E–04</td>
</tr>
<tr>
<td>$7R$</td>
<td>1.26E–03</td>
<td>5.44E–04</td>
<td>5.39E–04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CLP $r = 0.7$</th>
<th>CLP $r = 1.0$</th>
<th>CLP $r = 2.0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3R$</td>
<td>1.35E+01</td>
<td>6.41E+00</td>
<td>5.80E–01</td>
</tr>
<tr>
<td>$4R$</td>
<td>8.2E+00</td>
<td>4.4E+00</td>
<td>5.8E–01</td>
</tr>
<tr>
<td>$5R$</td>
<td>7.2E+00</td>
<td>2.2E+00</td>
<td>4.4E–01</td>
</tr>
<tr>
<td>$6R$</td>
<td>5.4E+00</td>
<td>4.9E+00</td>
<td>3.0E–01</td>
</tr>
<tr>
<td>$7R$</td>
<td>4.2E+00</td>
<td>2.4E+00</td>
<td>2.0E–01</td>
</tr>
</tbody>
</table>
Multi-objective optimization

- Formulation of Optimization Problem

\[
\text{minimize} \quad \frac{(\dot{\theta} + p)^T(\dot{\theta} + p)}{2} \\
\text{subject to} \quad J_e(\theta)\dot{\theta} = \dot{r}_d \\
J_o \dot{\theta} \leq b_o \\
\zeta^- \leq \dot{\theta} \leq \zeta^+ \\
\]

- Repetitive motion
- Tracking EE trajectory
- Obstacle constraints
- Joint limit
Formulation of Optimization Problem

\[
\text{minimize} \quad \frac{(\dot{\theta} + p)^T(\dot{\theta} + p)}{2} \\
\text{subject to} \quad J_e(\theta)\dot{\theta} = \dot{r}_d \\
J_o \dot{\theta} \leq b_o \\
\zeta^- \leq \dot{\theta} \leq \zeta^+ \\
\|
\dot{\theta}(t) + \eta(\theta(t) - \theta(0))\|_2^2 \\
\eta > 0 \in \mathbb{R}
\]

Repetitive motion

\[
p = \eta(\theta(t) - \theta(0))
\]

\[
\z(t) = \theta(t) - \theta(0) \\
\dot{z}(t) = -\eta z(t) \\
\|z(t)\|_2 = \exp(-\eta t)\|z(0)\|_2 \to 0
\]

\[
\theta(t) = \theta(0), \quad t \to \infty
\]
Dynamical quadratic programming

\[
\begin{align*}
\text{minimize} & \quad \frac{(\dot{\theta} + p)^T(\dot{\theta} + p)}{2} \\
\text{subject to} & \quad J_e(\theta)\dot{\theta} = \dot{r}_d \\
& \quad J_o \dot{\theta} \leq b_o \\
& \quad \zeta^- \leq \dot{\theta} \leq \zeta^+ 
\end{align*}
\]

\[
\begin{align*}
\text{minimize} & \quad \frac{x^TQx}{2} + p^Tx \\
\text{subject to} & \quad Ax = d \\
& \quad Cx \leq b \\
& \quad \zeta^- \leq x \leq \zeta^+
\end{align*}
\]

- Dynamical quadratic program (DQP) with equality, inequality, and bound constraints
- Can be solved by piecewise-linear projection equation (PLPE) neural network
Simulation Result [1]

Simulation motion

Motion of each joint in task space
Simulation Result

Joint angles

Minimal link-obstacle distance
Practical needs in robot control
Continuous, globally consistent redundancy resolution
Continuity and global consistency

• Continuity of redundancy resolution
  • Starting joint configuration was chosen "badly", then the robot tracking a simple path could get stuck when it hits joint limits.

• Globally consistent redundancy resolution
  • When tracking a cyclic path (forward and backward), the robot should return to the same joint configuration that it started from
Pathwise Redundancy Resolution
Algorithm 2 PRM-Path-Resolution($y, N$)

1: Initialize empty roadmap $\mathcal{R} = (V, E)$
2: if $q(0)$ and $q(1)$ are given then
3:     Add $(0, q(0))$ and $(1, q(1))$ to $V$
4: else
5:     Sample $O(N)$ start configurations using $\text{SampleF}(y(0))$
6:     Sample $O(N)$ goal configurations using $\text{SampleF}(y(1))$
7: for $i = 1, \ldots, N$ do
8:     Sample $t_{\text{sample}} \sim U([0, 1])$
9:     Sample $q_{\text{sample}} \leftarrow \text{SampleF}(y(t_{\text{sample}}))$
10:    if $q_{\text{sample}} \neq \text{nil}$ then add $(t_{\text{sample}}, q)$ to $V$
11: for all nearby pairs of vertices $(t_u, q_u), (t_v, q_v)$ with $t_u < t_v$ do
12:    if Visible($y, t_u, t_v, q_u, q_v$) then
13:        Add the (directed) edge to $E$
14: Search $\mathcal{R}$ for a path from $t = 0$ to $t = 1$

Add start and end points in configuration space
Algorithm 2 PRM-Path-Resolution(y, N)

1: Initialize empty roadmap $\mathcal{R} = (V, E)$
2: if $q(0)$ and $q(1)$ are given then
3: \hspace{1em} Add $(0, q(0))$ and $(1, q(1))$ to $V$
4: else
5: \hspace{1em} Sample $O(N)$ start configurations using $\text{SampleF}(y(0))$
6: \hspace{1em} Sample $O(N)$ goal configurations using $\text{SampleF}(y(1))$
7: for $i = 1, ..., N$ do
8: Sample $t_{sample} \sim U([0, 1])$
9: Sample $q_{sample} \leftarrow \text{SampleF}(y(t_{sample}))$
10: if $q_{sample} \neq \text{nil}$ then add $(t_{sample}, q_{sample})$ to $V$
11: for all nearby pairs of vertices $(t_{1}, q_{1})$, $(t_{2}, q_{2})$ do
12: \hspace{1em} if $\text{Visible}(y, t_{1}, t_{2}, q_{1}, q_{2})$ then
13: \hspace{2em} Add the (directed) edge to $E$
14: Search $\mathcal{R}$ for a path from $t = 0$ to $t = 1$

- $\text{SampleF}(y)$ first samples a random configuration $q_{\text{rand}} \in \mathcal{C}$ and then uses $\text{Solve}(y, q_{\text{rand}})$. If the result is $\text{nil}$ or infeasible, then $\text{nil}$ is returned.

- $\text{Solve}(y, q_{\text{init}})$ solves a root-finding problem $f(q) = y$ numerically using $q_{\text{init}}$ as the initial point. If it fails, it returns $\text{nil}$. It is assumed that the result $q$ lies close to $q_{\text{init}}$. 
Algorithm 2 PRM-Path-Resolution\((y, N)\)

1: Initialize empty roadmap \(\mathcal{R} = (V, E)\)
2: if \(q(0)\) and \(q(1)\) are given then
3: \hspace{1em} Add \((0, q(0))\) and \((1, q(1))\) to \(V\)
4: else
5: \hspace{1em} Sample \(O(N)\) start configurations using \(\text{SampleF}(y(0))\)
6: \hspace{1em} Sample \(O(N)\) goal configurations using \(\text{SampleF}(y(1))\)
7: for \(i = 1, \ldots, N\) do
8: \hspace{1em} Sample \(t_{\text{sample}} \sim U([0, 1])\)
9: \hspace{1em} Sample \(q_{\text{sample}} \leftarrow \text{SampleF}(y(t_{\text{sample}}))\)
10: \hspace{1em} if \(q_{\text{sample}} \neq \text{nil}\) then add \((t_{\text{sample}}, q)\) to \(V\)
11: for all nearby pairs of vertices \((t_u, q_u), (t_v, q_v)\) with \(t_u < t_v\) do
12: \hspace{1em} if Visible\((y, t_u, t_v, q_u, q_v)\) then
13: \hspace{1em} \hspace{1em} Add the (directed) edge to \(E\)
14: Search \(\mathcal{R}\) for a path from \(t = 0\) to \(t = 1\)

Sampling in the time domain – every node added subject to the manifold constraints
Local planner – directed edges restrict forward progress along the time domain
PRM-Path Resolution

- Local planner

Algorithm 1 Visible(y, t_s, t_g, q_s, q_g)

1: if \( d(q_s, q_g) \leq \epsilon \) then return “true”
2: Let \( y_m \leftarrow y((t_s + t_g)/2) \) and \( q_m \leftarrow (q_s + q_g)/2 \)
3: Let \( q \leftarrow \text{Solve}(y_m, q_m) \)
4: if \( q = \text{nil} \) or \( q \notin \mathcal{F} \) then return “false”
5: if \( \max(d(q, q_s), d(q, q_g)) > c \cdot d(q_s, q_g) \) then return “false”
6: if Visible(y, t_s, t_m, q_s, q_m) and Visible(y, t_m, t_g, q_m, q_g) then return “true”
7: return “false”

- \( \text{Solve}(y, q_{init}) \) solves a root-finding problem \( f(q) = y \) numerically using \( q_{init} \) as the initial point. If it fails, it returns \( \text{nil} \). It is assumed that the result \( q \) lies close to \( q_{init} \).
Approximate global redundancy resolution

• Assign a single robot configuration to each target point
• Pointwise global resolution
• Constraint-satisfaction-based resolution
Algorithm 3: Pointwise-Global-Resolution\((G_W, N_q)\)

1. Initialize empty roadmap \(R_C = (V_C, E_C)\).
2. for each \(y \in V_W\) do
   \[N(y)\] is the neighborhood of a vertex \(y\) in the workspace graph.
3. Let \(Q_{seed} \leftarrow \bigcup_{w \in N(y)} Q[w]\).
4. for each \(q_s \in Q_{seed}\) do
5. Run \(q \leftarrow \text{Solve}(y, q_s)\).
6. if \(q \neq \text{nil}\) then add \(q\) to \(V_C\) and go to Step 2, proceeding to the next \(y\).
7. Run \(\text{SampleF}(y)\) up to \(N_q\) times. If any sample \(q\) succeeds, add it to \(V_C\).
8. for all edges \((y, y') \in E_W\) such that \(|Q(y)| > 0\) and \(|Q(y')| > 0\) do
9. Let \(q\) be the only member of \(Q(y)\) and \(q'\) the only member of \(Q(y')\).
10. if \(R(y, y', q, q') = 1\) then
11. Add \((q, q')\) to \(E_C\) return \(R_C\).
Pointwise global resolution

Algorithm 3 Pointwise-Global-Resolution($G_W, N_q$)

1: Initialize empty roadmap $\mathcal{R}_C = (V_C, E_C)$
2: for each $y \in V_W$ do
3:     Let $Q_{seed} \leftarrow \bigcup_{w \in N(y)} Q[w]$
4:     for each $q_s \in Q_{seed}$ do
5:         Run $q \leftarrow \text{Solve}(y, q_s)$
6:         if $q \neq \text{nil}$ then add $q$ to $V_C$ and go to Step 2, proceeding to the next $y$.
7:         Run SampleF($y$) up to $N_q$ times. If any sample $q$ succeeds, add it to $V_C$.
8:     for all edges $(y, y') \in E_W$ such that $|Q(y)| > 0$ and $|Q(y')| > 0$ do
9:         Let $q$ be the only member of $Q(y)$ and $q'$ the only member of $Q(y')$
10:        if $R(y, y', q, q')=1$ then
11:        Add $(q, q')$ to $E_C$
12: return $\mathcal{R}_C$

Keep only one configuration
Pointwise global resolution
Limitation of pointwise method

• Pointwise method can yield poor results
  • Several edges unnecessarily unresolved

• Constraint-satisfaction problem
  • Sample many configurations in the preimage of each workspace point
  • Connect them with feasible edges
  • Seek a “sheet” in the C-space roadmap that satisfies the constraints
Constraint-satisfaction-based resolution

- Primary error metric
  - Measures the number of unresolved edges

- Secondary error metric
  - Maximize smoothness in the redundant dimensions
Minimize the number of unsolvable edges

- Let $G_W = (V_W, E_W)$ be the workspace roadmap

$$U(g) = |E_W| - \sum_{(y, y') \in E_W} R(y, y', g[y], g[y'])$$

- Seek the mapping $g$ from task space vertices to C-space vertices

Local reachability indicator function – check for locally pairwise resolvable
Maximize pseudo-inverse smoothness

- Distance is a good proxy for smoothness.
  - Use total C-space path length to measure smoothness

\[ L(g) = \sum_{(y, y') \in E_W} d(g[y], g[y']) R(y, y', g[y], g[y']). \]
Ensure connection in C-space and task space

- Given the C-space roadmap \( R = (V_c, E_c) \), make sure

\[
E_C = \{ (q, q') \mid (Y[q], Y[q']) \in E_W \text{ and } R(Y[q], Y[q'], q, q') = 1 \} 
\]
Discontinuity boundary for 3-DOF arm

- **Pointwise solution**
- **Optimization-based solution**
Discontinuity boundary for 3-DOF arm

Pointwise solution

Optimization-based solution


  • [link](http://motion.pratt.duke.edu/redundancyresolution/)