24-682 COMPUTER-AIDED ENGINEERING Spring 11

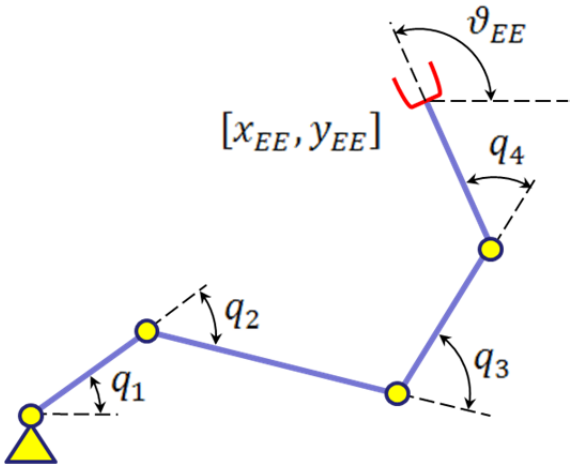
Carnegie Mellon University

PROBLEM SET 9

Due: **4/6/2011 (Wed) 12:30 PM @ SH 214**
Issued: 3/28/2011 (Mon)
Weight: 3% of total grade
Note: * **Attach the last page of the problem set as the cover page of your paper.**

PS9-1 Kinematically Redundant Planar Manipulator: Direct and Inverse Kinematic

The figure below illustrates a four link planar manipulator. Each link has length l_i. The joint angles, i.e. the relative angles between two adjacent links, are indicated with q_i (positive if counterclockwise). The *placement* of the end-effector is determined by three parameters: the two coordinates x_{EE} and y_{EE} and its absolute orientation ϑ_{EE}.



Determine the analytic relationship between a set of joint angles (also called a *configuration*) and the end-effector placement (this relationship is called *direct kinematic*). Write then a computationally efficient MATLAB function `ps9_1_DK( )` (you will call this function many times …) that calculates the end-effector *placement* given the manipulator *configuration*:

$$[x_{ee}, y_{EE}, \vartheta_{EE}] = \mathrm{DK}\,(q_1, q_2, q_3, q_4)$$

The inverse problem is to determine the manipulator *configuration* given an end-effector *placement* (this relationship is called *inverse kinematic*. The four link manipulator is kinematically redundant in the plane, i.e. there are infinite many *configurations* that correspond to the same *placement*. In other words there are an infinite number of solutions to this problem:

$$\boldsymbol{q} = [q_1, q_2, q_3, q_4] = \mathrm{IK}\,(x_{ee}, y_{EE}, \vartheta_{EE})$$

Use an optimization framework to write a MATLAB function `ps9_1_IK()` that solves for a feasible *configuration* given an end-effector *placement* while minimizing the norm of the *configuration* vector $\boldsymbol{q}$:

$$\begin{aligned} &\text{minimize: } \|\boldsymbol{q}\|_2 \\ &\text{subject to: } [x_{ee}, y_{EE}, \vartheta_{EE}] = \mathrm{f}(\boldsymbol{q}) \end{aligned}$$

The parameter to use for the manipulator are:

```
l = [1.2 0.8 0.8 0.4];        % Links lengths [m]
qL = -[0.9 1.0 1.0 1.2]*pi;   % Joints lower limits [rad]
qU = -qL;                     % Joints upper limits [rad]
```

Make sure that your optimization routine constraints the solution to be within the given joint limits.

In your hand-in directory on AFS, make a new directory called ps9-1 (in lower case) and upload your MATLAB code. Name the two functions "ps9_1_DK.m" and "ps9_1_IK.m", and include the title and your name, "Direct/Inverse Kinematic by YOUR_NAME," at the top of your code.

Submit a printout of your MATLAB code. Also, submit the explanation of the optimization method used -- your explanation should be at least one-page long, and the results given by your inverse kinematic routine for the following two *placements*:

```
PlaS = [1.8, -0.3, -pi/2];    % Pick up placement
PlaE = [-0.5, 2.2, pi];       % Drop off placement
```

Hint: one tool you can use to solve this homework is `fmincon()` available in the Matlab Optimization Toolbox. We suggest you to carefully read the help of this built-in function to well understand how to specify variable boundaries and equality/inequality non-linear constraints.

PS9-2 Path Planning: Cycle Time Optimization

Your task is to find the near-minimum traveling time trajectory for the manipulator to move from a pickup *placement* to a drop off *placement* while avoiding an obstacle, which is represented by a planar disk.

The trajectory can be simplified as a sequence of n *configurations*. Each *configuration* of the manipulator is a vector $\boldsymbol{q}^{(i)} = \left[q_1^{(i)}, q_2^{(i)}, q_3^{(i)}, q_4^{(i)}\right]$. The time required to go from one *configuration* to the next in the trajectory can be simplified by:

$$\Delta t^{(i)} = \max_{k=1,\ldots,4} \left\{ \frac{\left| q_k^{(i+i)} - q_k^{(i)} \right|}{\omega_k} \right\}$$

where ω_k is the angular velocity of the k-th joint. For the purpose of this problem you will use:

```
omega = [1.0 2.0 2.0 3.0];    % Joint angular velocity [rad/s]
```

Furthermore, to obtain a smooth trajectory, we constraint each step to be smaller than a given constant value (different for each joint):

$$\left|q_k^{(i+i)} - q_k^{(i)}\right| \leq \Delta q_{k,\mathrm{MAX}} \quad \forall i \in \{1,\ldots,n\} \quad \forall k \in \{1,\ldots,4\}$$

Write a MATLAB function `ps9_2_T()` that calculates a minimum traveling time trajectory of 10 *configurations* (n=10) to connect a pickup *placement* to a drop off *placement* while keeping the joints within their limits and not allowing each step to be larger than the value:

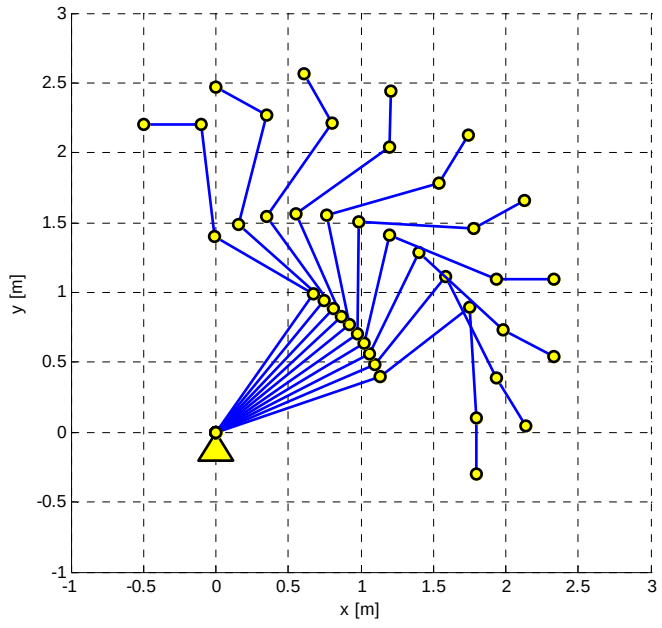
```
Dqmax = pi./[12 10 10 8];     % Maximum step size [rad]
```

Make sure that your objective function, linear constraints, and non-linear constraints are properly stated

Hint: In this case the variables of your optimization routine are **10 x 4 = 40!** This is the reason why in practical applications local optimization approaches are used instead of a unique global optimization as we do in this homework.

In your hand-in directory on AFS, make a new directory called ps9-2 (in lower case) and upload your MATLAB code. Name the file "ps9_2_T.m" and include the title and your name, "Minimum Traveling Time Trajectory Calculation by YOUR_NAME," at the top of your code.

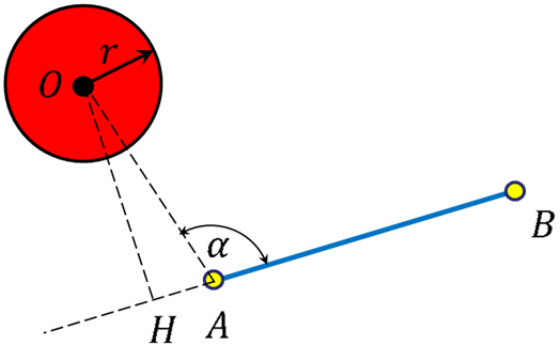
Submit a printout of your MATLAB code, a plot of the trajectory (similar to the picture below), the list of the calculated configurations for the near-optimal trajectory, and the near-optimal travelling time calculated using for the pickup *placement* and the drop off *placement* the value given to you in PS9-1.

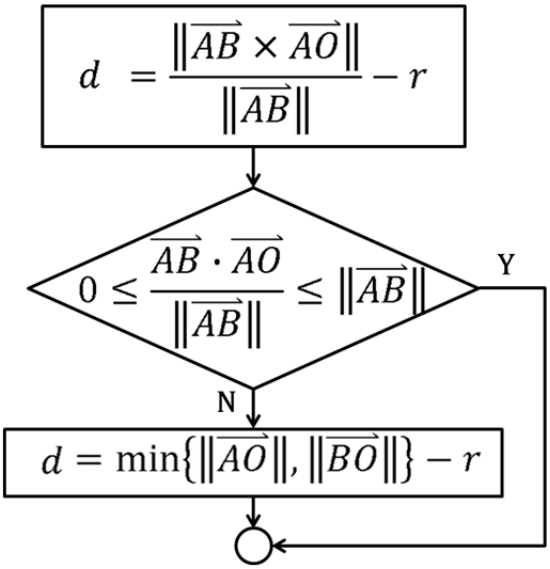


PS9-3 Path Planning: Cycle Time Optimization with Obstacle Avoidance

In this third part we add an obstacle between the pickup location and the drop off location. The obstacle is a disk centered at x = 2 m, y = 2 m and with radius 0.7 m. In order to avoid this obstacle we will add a non-convex additional term to the objective function we used in PS9-2. This term is called *repulsive potential* and it is a function of the distance between the manipulator and the obstacle: the smaller is this distance, the larger becomes the repulsive potential.

First, write an efficient MATLAB function `ps9_3_OD()` that returns the smallest distance between the fixed obstacle and the manipulator in a given *configuration*. You can use this simple procedure to calculate the minimum distance between a disk and a manipulator link:





You can use also your own and possibly more efficient approach to calculate the minimum distance. Notice that this function should be very efficient since it is going to be called by `fmincon()` at each objective function evaluation.

Second, write a MATLAB function `ps9_3_TO()` that calculates a minimum traveling time trajectory of **10** *configurations* to connect a pickup *placement* to a drop off *placement* while keeping the joints within their limits, not allowing each step to be larger than the maximum value defined in PS9-2, and avoiding the obstacle.

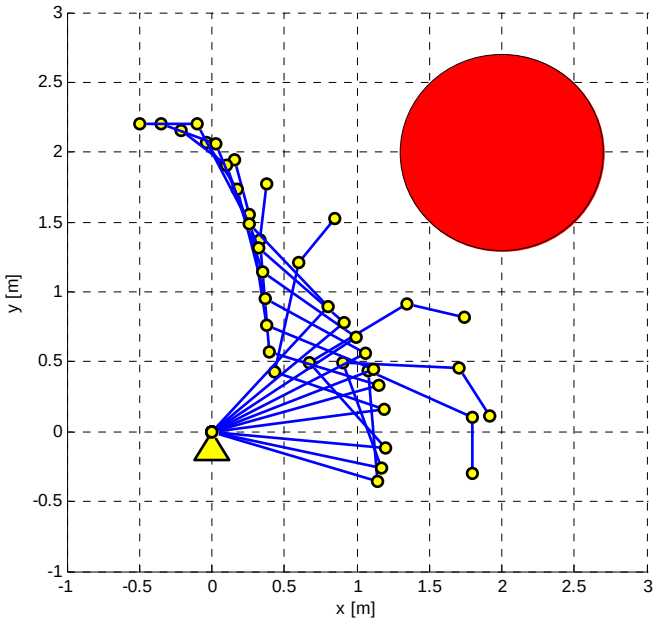
Hint: the repulsive component of your objective function can be (not mandatory, though):

$$\gamma \cdot \sum_{i=1}^{n} \frac{1}{\max\left\{\mathrm{d}\left(\boldsymbol{q}^{(i)}\right) - \mathrm{d}_{\mathrm{MIN}}, \epsilon\right\}}$$

where γ is a damping factor, ϵ is a positive number to avoid a division by zero, $\mathrm{d}_{\mathrm{MIN}}$ is a safety distance you want to keep from the obstacle, and $\mathrm{d}\left(\boldsymbol{q}^{(i)}\right)$ is the output of the function `ps9_3_OD()` you implemented in PS9-2. Notice that if $\mathrm{d}\left(\boldsymbol{q}^{(i)}\right) - \mathrm{d}_{\mathrm{MIN}}$ becomes less than ϵ, you may want to slightly modify the above expression to have a linear behavior (instead of constant) of the *repulsive potential* to avoid local minima.

In your hand-in directory on AFS, make a new directory called ps9-3 (in lower case) and upload your MATLAB code. Name the two files "ps9_3_OD.m" and "ps9_3_TO.m" and include the title and your name, "Minimum Traveling Time Trajectory Calculation with Obstacle by YOUR_NAME," at the top of your code.

Submit a printout of your MATLAB code, a plot of the trajectory (similar to the picture below), the list of the calculated configurations for the near-optimal trajectory, and the near-optimal travelling time calculated using for the pickup *placement* and the drop off *placement* the value given to you in PS9-1.



EXTRA Morphology Optimization (10 bonus points)

In this last part your task is to optimize the morphology of the manipulator in order to minimize again the cycle time. In other words, the manipulator has to travel from the pickup *placement* to the drop off *placement*, to avoid the obstacle, and to perform the task in the shortest time as possible. The question is: what are the best links dimensions to do that?

Hint: Now the number of variables of the optimization problem is going to be 10x4 + 4 = 44. You can use the code you wrote for PS9-3 to solve the problem. You just need to consider the link lengths not as parameter but as variables of the problem.

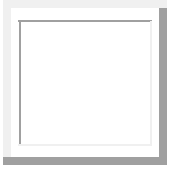
Write a MATLAB function `ps9_4_TOL()` that calculates a minimum traveling time trajectory of 10 *configurations* to connect a pickup *placement* to a drop off *placement* while keeping the joints within their limits, not allowing each step to be larger than the maximum value defined in PS9-2, and avoiding the obstacle defined in PS9-3. The limits to use for the link dimension are:

```
lL = [0.8 0.4 0.4 0.2];     % Link lower limits [m]
lU = [1.8 1.2 1.2 0.6];     % Link upper limits [m]
```

In your hand-in directory on AFS, make a new directory called ps9-4 (in lower case) and upload your MATLAB code. Name the file “ps9_4_TOL.m” and include the title and your name, “Morphology optimization by YOUR_NAME,” at the top of your code.

Submit a printout of your MATLAB code, a plot of the trajectory (similar to the plots you did for PS9-3), the list of the calculated configurations for the near-optimal trajectory, the near-optimal travelling time calculated, and your optimal four link lengths using for the pickup *placement* and the drop off *placement* the value given to you in PS9-1. Submit also a brief description to illustrate how the link lengths have changed, and how the traveling time has improved compared to your previous results in PS9-3?

PS9



The first letter of your LAST name

_____ First Name

_____ Last Name

How many hours did you spend to complete this problem set? _____ Hours

How many no-penalty late days do you want to use for this problem set? _____

PS9-1 (30 pts)	PS9-2 (30 pts)	PS9-3 (40 pts)	Following Instructions (5 pts)	Bonus Points (10 pts)	Total (110 pts)

24-682 COMPUTER-AIDED ENGINEERING Spring 11

Carnegie Mellon University

PROBLEM SET 9

Due: **4/6/2011 (Wed) 12:30 PM @ SH 214**
Issued: 3/28/2011 (Mon)
Weight: 3% of total grade
Note: * **Attach the last page of the problem set as the cover page of your paper.**