AE483 Homework #4:
Quad-Rotor Collision Avoidance

(due at the beginning of class on Monday, November 16)

1. *Be able to find the distance between two spheres.* One sphere has center $q \in \mathbb{R}^3$ and radius $r > 0$. The other sphere has center $p \in \mathbb{R}^3$ and radius $s > 0$.

   (a) Find the distance between these two spheres.

   (b) Find the gradient of this distance with respect to q.

   (c) Implement (a) and (b) as the function `[d,dgrad] = SphereSphere(q,r,p,s)` in the MATLAB script `hw4code.m`.

2. *Be able to find the distance between a sphere and a plane.* The sphere has center $q \in \mathbb{R}^3$ and radius $r > 0$. The plane (i.e., a flat surface) passes through the point $p \in \mathbb{R}^3$ and is normal to the unit vector $z \in \mathbb{R}^3$.

   (a) Find the distance between the sphere and the plane.

   (b) Find the gradient of this distance with respect to q.

   (c) Implement (a) and (b) as the function `[d,dgrad] = SpherePlane(q,r,p,z)` in the MATLAB script `hw4code.m`.

3. *Be able to find the distance between a sphere and a hot dog.* The sphere has center $q \in \mathbb{R}^3$ and radius $r > 0$. The "hot dog" is the set of all points closer than $s > 0$ to a line segment with endpoints $p_1 \in \mathbb{R}^3$ and $p_2 \in \mathbb{R}^3$.

   (a) Find the distance between the sphere and the hot dog.

   (b) Find the gradient of this distance with respect to q.

   (c) Implement (a) and (b) as the function `[d,dgrad] = SphereHotDog(q,r,p1,p2,s)` in the MATLAB script `hw4code.m`.

4. *Be able to compute the gradient of an attractive potential function.* As we saw in class, the gradient of

$$f_{\text{att}}(q) = \begin{cases} \frac{1}{2} k_{\text{att}} \|q - q_{\text{goal}}\|^2 & \text{if } \|q - q_{\text{goal}}\| \leq b_{\text{att}} \\ k_{\text{att}} b_{\text{att}} \left(\|q - q_{\text{goal}}\| - \frac{1}{2} b_{\text{att}} \right) & \text{otherwise} \end{cases}$$

   with respect to q is

$$\nabla_q f_{\text{att}}(q) = \begin{cases} k_{\text{att}} \left(q - q_{\text{goal}} \right)^T & \text{if } \|q - q_{\text{goal}}\| \leq b_{\text{att}} \\ k_{\text{att}} b_{\text{att}} \left(\frac{q - q_{\text{goal}}}{\|q - q_{\text{goal}}\|} \right)^T & \text{otherwise.} \end{cases}$$

   Implement this computation as `gradfatt = GetAttractiveGradient(drone,goal,param)` in the MATLAB script `hw4code.m`.

5. *Be able to compute the gradient of a repulsive potential function.* Let $d_i(q, A_i)$ be the distance between a sphere of center q and a convex set $A_i \subset \mathbb{R}^3$. As we saw in class, the gradient of

$$f_{\text{rep,i}}(q) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{d_i(q, A_i)} - \frac{1}{b_{\text{rep}}} \right)^2 & \text{if } d_i(q, A_i) \leq b_{\text{rep}} \\ 0 & \text{otherwise} \end{cases}$$

with respect to q is

$$\nabla_q f_{\text{rep,i}}(q) = \begin{cases} -k_{\text{rep}} \left(\frac{1}{d_i(q,A_i)} - \frac{1}{b_{\text{rep}}} \right) \left(\frac{1}{d_i(q,A_i)} \right)^2 \nabla_q d_i(q, A_i) & \text{if } d_i(q, A_i) \leq b_{\text{rep}} \\ 0 & \text{otherwise.} \end{cases}$$

By linearity, the gradient of

$$f_{\text{rep}}(q) = \sum_{i=1}^{n_{\text{obs}}} f_{\text{rep,i}}(q)$$

is

$$\nabla_q f_{\text{rep}}(q) = \sum_{i=1}^{n_{\text{obs}}} \nabla_q f_{\text{rep,i}}(q).$$

Implement this computation as `gradfrep = GetRepulsiveGradient(drone,goal,param)` in the MATLAB script `hw4code.m`. (Helpfully, this function already loops through all the obstacles—you need only figure out how to update $\nabla_q f_{\text{rep}}(q)$ each time through the loop.)

6. *Be able to implement gradient descent.* We have seen how to compute both attractive and repulsive potential functions. The total potential function is simply

$$f(q) = f_{\text{att}}(q) + f_{\text{rep}}(q).$$

To locally minimize $f(q)$, we can apply gradient descent:

$$\dot{q}(t) = -k_{\text{descent}} \nabla_q f(q)^T,$$

where $k_{\text{descent}} > 0$ is a constant gain. We can approximate gradient descent in discrete time using Euler's method:

$$q(t + \Delta t) \approx q(t) - \Delta t \left(k_{\text{descent}} \nabla_q f(q)^T \right),$$

where $\Delta t > 0$ is a constant time step. When implementing gradient descent, we often cap the step size as follows:

$$q(t + \Delta t) = \begin{cases} q(t) - \Delta t \left(k_{\text{descent}} \nabla_q f(q)^T \right) & \text{if } \| \Delta t \left(k_{\text{descent}} \nabla_q f(q)^T \right) \| \leq b_{\text{descent}} \\ q(t) - b_{\text{descent}} \left(\frac{\nabla_q f(q)^T}{\| \nabla_q f(q)^T \|} \right) & \text{otherwise,} \end{cases}$$

where $b_{\text{descent}} > 0$ is a constant maximum step size. Implement this computation as

```
res = DoGradientDescent(world,drone,goal,obst,param)
```

in the MATLAB script `hw4code.m`.

7. *Be able to tune the parameters governing a potential field approach to collision avoidance.* Four parameters govern the behavior of the potential field approach to collision avoidance that you have now implemented in `hw4code.m`:

$$k_{\text{att}}, b_{\text{att}}, k_{\text{rep}}, b_{\text{rep}}.$$

   (a) Choose values for these parameters so that the "quad-rotor" (i.e., the blue sphere) reaches the "goal" (i.e., the green sphere) with "SUCCESS!" in the MATLAB script `hw4code.m` in both Scenario #1 and Scenario #2 (uncomment the appropriate lines of `hw4code.m` to try these scenarios). Try to choose values that make the result "look nice" (e.g., that minimize chattering or other strange behavior) and be relatively fast. Explain why you chose the values that you did. (What bad things happen with different values?)

   (b) Once you have chosen parameters, apply them also to Scenario #3. In this case, spherical obstacles are scattered randomly throughout the workspace. Try executing the code several times—you should find that, occasionally, the quad-rotor will get stuck (and the code will return "FAILURE!"). Why does it get stuck?