Rob 501 - Mathematics for Robotics
HW #9

Prof. Grizzle

Due Nov. 29, 2018
3PM via Gradescope

Remarks:

- If you have been hearing about "quadratic programs" (QPs), well, Problem 2 is a simple case of a QP. You also solved one in HW 7 and in HW 1 (go back and look at Problem 6 of HW 1). We will do more advanced QPs at the end of the term.

1. Consider the finite dimensional vector space $(\mathcal{X}, \mathbb{R})$, where $\mathcal{X} = \text{span}\{1, t, t^2, \sin(\pi t)\}$. Equip it with the inner product $< f, g >:= \int_0^2 f(\tau)g(\tau)d\tau$. When working the problem, feel free to use MATLAB to compute any required integrals, and you may compute them symbolically or numerically, as you wish.

   (a) Find the vector of minimum norm that satisfies $< f, t >= 2$.

   (b) Find the vector of minimum norm that satisfies $< f, t >= 2$ and $< f, \sin(\pi t) >= \pi$.

2. **Underdetermined Equations:** We consider $Ax = b$, where $b \in \mathbb{R}^p$, $x \in \mathbb{R}^n$, $n > p$. Use the normal equations derived in HW #8 to solve the following problems. The key is to interpret the rows of $Ax = b$ in terms of inner product conditions that look like $< x, y_i >= c_i$.

   (a) We assume an inner product on $\mathbb{R}^n$ defined by $< x, z >:= x^\top z$, and thus $||x|| = (x^\top x)^{1/2}$. Show that if the rows of A are linearly independent, then

   $$\widehat{x} := \arg \min_{Ax=b} ||x||$$

   is given by $\widehat{x} = A^\top \left(AA^\top\right)^{-1} b$

   (b) We assume an inner product on $\mathbb{R}^n$ defined by $< x, z >:= x^\top Q z$, where $Q \succ 0$, and thus $||x|| = (x^\top Q x)^{1/2}$. Show that if the rows of A are linearly independent, then

   $$\widehat{x} := \arg \min_{Ax=b} ||x||$$

   is given by $\widehat{x} = Q^{-1}A^\top \left(AQ^{-1}A^\top\right)^{-1} b$.

   Remark: A QP is an optimization problem of the form

   $$\widehat{x} := \begin{array}{c} \arg \min \quad ||x||^2. \\ A_{eq}x = b_{eq} \\ A_{in}x \leq b_{in} \end{array}$$

   The key addition is that inequality constraints, $A_{in}x \leq b_{in}$, can also be included. We'll talk more about this the last day of lecture.

3. We did one version of the QR factorization in lecture (we assumed A had linearly independent columns). Scan MATLAB's documentation of the `qr` function and note its *economy* version, `[Q,R]=qr(A,0)`. Using Gram Schmidt, compute the QR factorization of the matrix A below "by hand" (you can do any vector multiplications and such in MATLAB[1]). Compare your answer to MATALB's `[Q,R]=qr(A)` and `[Q,R]=qr(A,0)` and discuss similarities and differences.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

4. The following problem arises in camera calibration:

$$\widehat{x} = \arg \min_{x^\top x = 1} x^\top A^\top A x.$$

Show that if A is real[2], then $\widehat{x}$ is given by the last column of V where $A = U\Sigma V^\top$ is the SVD of A.

5. Compute a rank 2 approximation of

$$A = \begin{bmatrix} 4.041 & 7.046 & 3.014 \\ 10.045 & 17.032 & 7.027 \\ 16.006 & 27.005 & 11.048 \end{bmatrix}.$$

In particular, find ΔA of smallest norm such that $\widehat{A} := A + \Delta A$ has rank 2. The matrix norm being used is

$$||M||_2 = \sqrt{\lambda_{max}(M^\top M)}.$$

It is enough to give $\widehat{A}$, the rank 2 approximation, in your solution.

6. This problem is just for fun. It will not show up on any exam. Download the file `zebra.jpg` from the MATLAB folder. Everything I know about image compression, I learned in 15 minutes with a google search[3] on "image svd". I invite you to do the same! These commands show how you can read an image into MATLAB, covert it to double precision arithmetic, perform a numerical operation on the matrix, convert it back to binary grey scale, and print it to a figure:

```
A = imread('zebra.jpg');
%throw away color information for this example
A=A(:,:,1);
% show the image
figure(1)
imshow(A)
%convert to double precision
B=double(A);
%Let's do a matrix operation; you will find something more imaginative to do
C=fliplr(B);
%convert back to binary
C=uint8(C);
%show the image
figure(2)
imshow(C)
```

[1] This is just to encourage you to do one QR factorization yourself without relying on MATLAB's built-in function.

[2] When we write down `arg min`, we should also have conditions so that the answer is unique. For this problem, we need the smallest e-value of $A^\top A$ to be unique, and even then, it is only unique up to a sign (i.e., if $\widehat{x}$ is an answer, then so is $-\widehat{x}$). Unfortunately, in the literature, you typically see the problem given as stated.

[3] You might also try the last page of `http://persson.berkeley.edu/18.335/lec3handout2pp.pdf` and glance at `http://www.math.ucla.edu/~wittman/PCMI/pcmi1.pdf`

(a) Use the SVD to produce a low-rank approximation to the image and display it. Can you reduce the rank by 75% and still have an acceptable image? Less? No, need more? The purpose is to illustrate visually that the SVD can help you to find and eliminate redundant data. Turn in the original image and your low rank approximation; state the rank of your approximation.

(b) Download a "wholesome" image[4] from the internet and repeat the above.

7. Estimating derivatives of functions numerically. In this problem, we introduce a method to estimate the Jacobian of a function numerically.

(a) Compute the Jacobian of the following function $f : \mathbb{R}^3 \to \mathbb{R}$ analytically and evaluate it at the point $x^* = [1\ 3\ -1]^\top$

$$f(x_1, x_2, x_3) = 3x_1 \left[2x_2 - (x_3)^3\right] + \frac{(x_2)^4}{3}. \tag{1}$$

Recall that the gradient of a scalar valued function is normally written as a row vector

$$\frac{\partial f(x)}{\partial x} := [\frac{\partial f(x)}{\partial x_1} \quad \frac{\partial f(x)}{\partial x_2} \quad \frac{\partial f(x)}{\partial x_3}].$$

(b) In calculus, you learned that $\frac{\partial f(x^*)}{\partial x_i} := \lim_{\delta \to 0} \frac{f(x^* + \delta e_i) - f(x^*)}{\delta}$, where e_i, $i = 1:3$ are the natural basis vectors, and thus, for a fixed $\delta > 0$, you might estimate the derivative by

$$\frac{\partial f(x^*)}{\partial x_i} \approx \frac{f(x^* + \delta e_i) - f(x^*)}{\delta}$$

It turns out that better numerical accuracy is usually obtained by a *symmetric difference*[5]

$$\frac{\partial f(x^*)}{\partial x_i} \approx \frac{f(x^* + \delta e_i) - f(x^* - \delta e_i)}{2\delta}. \tag{2}$$

Compute a numerical approximation to the Jacobian of the function (1) using symmetric differences and report the value(s) you used for δ. You are not obliged to use the same δ for each partial derivative. Use the same x^* as in part (a).

(c) When you already know the answer, it is easy to play with δ and come up with a good numerical approximation to the derivative. What will you do when you do not know the answer before hand? Let's find out! Download the function `funcPartC.p` from the MATLAB folder. It is a hidden function $f : \mathbb{R}^5 \to \mathbb{R}$. Report its Jacobian at $x^* = [1, 1, 1, 1, 1]$.

Here is how to call the function

```
x0=[1 2 3 4 5];
f=funcPartC(x0)
f = 2.1281e+01
```

[4] Be considerate! Animals and scenery tend to be non-controversial.

[5] It can be shown that a symmetric difference is EXACT for a quadratic function. You might want to try that out.

8. **One step update of a robot's state using a Kalman filter**. You are given a simple robot that moves in one dimension, say along the X-axis.

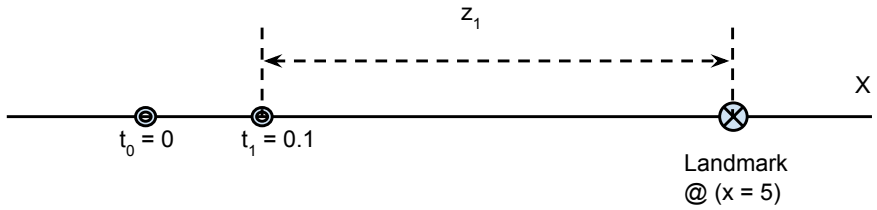

Figure 1: Robot's motion is along the X-axis. The landmark is a fixed point at $x = 5$m.

Your robot starts at a position[6] x_0, which is normally distributed $x_0 \sim \mathcal{N}(1, 0.25)$, i.e. $\mu_0 = 1$ and $\Sigma_0 = \sigma^2 = 0.25$. The state of your robot at the next time step is given by the following equation

$$x_1 = x_0 + u_1 \cdot \delta_t \quad \text{(state propagation model)} \tag{3}$$

where u_1, the action taken, is also normally distributed[7], $u_1 \sim \mathcal{N}(10, 16)$, i.e. $\hat{u}_1 = 10$, $R = 16$ and $\delta_t = 0.1$ is the time step.

Fortunately, your robot has a reasonably accurate time of flight sensor[8] which can measure the robot's distance from a fixed landmark. Based on the physics of the situation, you expect the output of your robot's sensor (i.e., the measurement) to be related to the robot's position by

$$\hat{z}_t = \frac{2}{c}(5 - x_t), \tag{4}$$

where x_t is the robot's position at time t, c is the speed of light[9] and $\hat{z}_t$ is the corresponding expected output[10](in seconds) from the sensor.

Because your sensor is a real device, it has error in its measurement; the actual sensor output is modeled to be, $z_1 \sim \mathcal{N}(\hat{z}_t, Q)$, where Q is the uncertainty in the measurement. In this case, the manufacturer tells you that your LiDAR has uncertainty, $Q = 10^{-18}$.

Problem: Suppose at time $t = 0.1s$ your robot takes action u_1 and moves according to (3). Your sensor outputs $z_1 = 2.2 \times 10^{-8}$s. Estimate the position x_1 of your robot at time $t = 0.1$ and the uncertainty in its position. Give your answer in the form of a normal distribution $x_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$.

[6] We have used the notation, $x \sim \mathcal{N}(\mu, \Sigma)$, in previous HW sets.

[7] Why would the control signal be random? Well, what if the feet or the wheels of your robot are slipping in sand or wet grass? Then the control signal you command to the motor is not the action that will be applied to the body of the robot!

[8] Understanding how LiDAR works https://www.youtube.com/watch?v=EYbhNSUnIdU. More information about LiDARs specifically in mobile robots can be found here, https://www.clearpathrobotics.com/2015/04/robots-101-lasers/

[9] Use $c = 3 \times 10^8$ m/s

[10] The expected output, $\mathbb{E}\{z\} = \mu_z$, is the mean value

Hints

Hints: Prob. 1 Apply Problem 6 from HW #8. You may enjoy the symbolic toolbox for doing the calculations.

```
syms t pi
f=t^2;
g=exp(pi*t);
a=0; b=2;
G11=int(f*f,a,b);
G11=simple(G11)
```

Commands such as `inv` and `numden` work in the symbolic toolbox. Also checkout `simplify` and `pretty`.

Hints: Prob. 2 Decompose A by its rows,

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}.$$

Define $\tilde{A} = AQ^{-1}$ (why is Q invertible?), and define $v_i \in \mathbb{R}^n$ by

$$v_i = (a_i Q^{-1})^\top = Q^{-1} a_i^\top,$$

where we have used the fact that Q is symmetric. Show that

$$Ax = b \Leftrightarrow < v_i, x >= b_i, \quad 1 \leq i \leq p.$$

Now, work out those normal equations from last week! If the above derivation confuses you, set $Q = I$ and you will get it! That is why I broke the problem into two parts.

Remark: Squaring the norm does not change the answer[11] to the optimization problem because it is a strictly monotonically increasing function.

$$\widehat{x} := \arg \min_{Ax=b} ||x|| = \arg \min_{Ax=b} ||x||^2 = \arg \min_{Ax=b} x^\top Q x.$$

The corresponding QP is always written using the form on the right.

Hints: Prob. 3 Nothing exciting here. Just pointing out that there exist multiple versions of the QR factorization.

Hints: Prob. 4 We have seen this problem before. Recall HW #2, Prob. 7-(b). How does the e-vector of $A^\top A$ corresponding to the minimum e-value relate to the columns of V? (See statement of SVD Theorem given in lecture).

Hints: Prob. 5 See the handout SVD_Rob501 in the resource folder...the one everyone slept through! If this were not very useful and practical stuff, I would not pester you about it. I know, about this point in the term, everyone is pretty tired.

[11] Yes, the value is squared, but the argument achieving the minimum, i.e., the $\widehat{x}$, does not change.

Hints: Prob. 6 `k=YourValue; C=U(:,1:k)*S(1:k,1:k)*V(:,1:k)';` Too much said. The problem is for fun, but the message is interesting!

Hints: Prob. 7 Oh, the age-old problem, how to tune parameters in algorithms? In this case, a *rule of thumb* is, if decreasing δ by a factor of 2 does not significantly change the estimate, then you can stop.

`Remark on exactness for a quadratic function:` Let $f(x) = ax^2 + bx + c$ be a quadratic. Performing the symmetric difference quotient about a point x^*, we have

$$\begin{aligned}
\frac{f(x^*+h) - f(x^*-h)}{2h} &= \frac{a(x^*+h)^2 + b(x^*+h) + c - (a(x^*-h)^2 + b(x^*-h) + c)}{2h} \\
&= \frac{ax^{*2} + 2ax^*h + ah^2 + bx^* + bh + c - ax^{*2} + 2ax^*h - ah^2 - bx^* + bh - c}{2h} \\
&= \frac{4ax^*h + 2bh}{2h} \\
&= \frac{2h(2ax^* + b)}{2h} \\
&= 2ax^* + b,
\end{aligned}$$

where we recognize the last line as the derivative at x^*.

Hints: Prob. 8 You have all the values for the Kalman filter available to you. Recall that you have seen in the handout on Gaussian Random Variables that if x_1 and x_2 are uncorrelated and $Y = Ax_1 + Bx_2$, then

$$\begin{aligned}
\mu_y &= A\mu_{x_1} + B\mu_{x_2} \\
\Sigma_y &= A\Sigma_{x_1}A^\top + B\Sigma_{x_2}B^\top
\end{aligned}$$

In this problem, you can see that $A = 1$ and $B = 0.1$, which will allow you to compute $\hat{x}_{1|0}$ and $P_{1|0}$. The rest of the data is available to you. Plug the values into the Kalman filter equations and note that you want to compute $\hat{x}_{1|1}$ and $P_{1|1}$, i.e. $x_1 \sim \mathcal{N}(\hat{x}_{1|1}, P_{1|1})$

Kalman Filter Equations with action model:
Prediction Step:

$$\begin{aligned}
\hat{x}_{k+1|k} &= A\hat{x}_{k|k} + B\hat{u}_1 \\
P_{k+1|k} &= AP_{k|k}A^\top + BRB^\top
\end{aligned}$$

Measurement Update Step:

$$\begin{aligned}
K_{k+1} &= P_{k+1|k}C^\top(CP_{k+1|k}C^\top + Q)^{-1} \\
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - \hat{z}_{k+1|k}) \\
P_{k+1|k+1} &= P_{k+1|k} - K_{k+1}CP_{k+1|k}
\end{aligned}$$

$\hat{z}_{k+1|k}$ is the expected output if the position and measurement were not stochastic, i.e. using (4),

$$\hat{z}_{k+1|k} = \frac{2}{c}(5 - \hat{x}_{k+1|k})$$