ECE-520 Control Systems II

Recursive Least Squares

A Report is due May 14

Notes:

1. You may work in groups of two on this project, but I am likely to more generous to students working alone.
2. This is not a class assignment. You are not to share solutions with other groups/people.
3. You may ask me for clarification, but not help.
4. Your derivations can be hand written in an attachment. The result of the report must be computer generated with the figures included, with figures numbers and captions. *The derivations are the most important parts of this project.*
5. I would start this early. You are likely to get stuck and it will be a good idea to put it away for awhile and then come back to it.

Introduction and Motivation.

In this assignment, we will derive the recursive least squares algorithm in a somewhat general context, and then implement it in Matlab and try a few simple cases. Although the recursive least squares algorithm may seem to be of little use, it is a precursor to the Kalman filter (which is very widely used) and is one of the simplest adaptive methods to utilize. Although there will seem to be alot of equations, we will break this up into small pieces.

Consider a single input-single output system (such as you may have used in a classical control course). In general, you can modify the system behavior somewhat by employing an appropriate compensator (lead, lead-lag, PID, etc) once the correct transfer function is known. However, sometimes the transfer function may change (rapidly) over time and you will need to be able to adapt your compensator (or the input to the plant) based on your current estimate of the system transfer function. As a specific example, consider a drilling process. When the drill is not touching the workpiece it might have one transfer function, when it first starts drilling it may have a different transfer function, and as the drill bit heats up, it may have a third transfer function. In order to try to "optimize" the drilling process, we need to determine estimates of the transfer function. Hence, for this specific example, we assume the process output $y(k)$ (which might be the speed of the drill bit) is

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + ... + a_m y(k-m) + b_0 u(k) + b_1 u(k-1) + ... + b_n u(k-n)$$

where $y(k-i)$ are the outputs at previous time steps, $u(k-i)$ are the inputs, and the coefficients $a_1, ...a_m, b_0, ...b_n$ are the coefficients to be determined.

$\boxed{\textbf{In what follows, you may not assume that a vector has an inverse!!}}$

$\boxed{\textbf{You will loose MANY points if you do...}}$

Derivation

We will now derive the recursive least squares in a general setting, and then apply it to the above problem. Assume, in general, the output at time k, $y(k)$, is given as

$$\begin{aligned} y(k) &= \phi_1(k)\theta_1 + \phi_2(k)\theta_2 + ... + \phi_n(k)\theta_n \\ &= \underline{\phi}^t(k)\underline{\theta} \end{aligned}$$

where

$$\begin{aligned} \underline{\phi}^t(k) &= [\phi_1(k)\ \phi_2(k)\ ...\ \phi_n(k)] \\ \underline{\theta}^t &= [\theta_1\ \theta_2\ ...\ \theta_n] \end{aligned}$$

a) Assume we are trying to model the system given by difference equation

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + b_0 u(k) + b_1 u(k-1)$$

(*) What are good choices for $\underline{\phi}(k)$ and $\underline{\theta}$?

b) Let's define the error at time step k as

$$e(k) = y(k) - \hat{y}(k) = y(k) - \underline{\phi}^t(k)\hat{\underline{\theta}}$$

where $\hat{\underline{\theta}}$ is the current estimate of the parameters in $\underline{\theta}$ and $\hat{y}(k) = \underline{\phi}^t(k)\hat{\underline{\theta}}$ is the estimate of the true output $y(k)$ at time k. We now want to determine $\hat{\theta}$ to minimize a simple function of the error. Lets look at the error measure

$$V(\hat{\underline{\theta}}, k) = \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i} e^2(i)$$

Here λ is a forgetting or deweighting factor between 0 and 1. With this parameter, we can emphasize new data and "discount" or "deweight" old data. For $\lambda = 1$ we have the straight least squares algorithm.
(*) Show that the minimum is given by

$$\hat{\underline{\theta}} = \left[\sum_{i=1}^{i=k} \lambda^{k-i}\underline{\phi}(i)\underline{\phi}^t(i)\right]^{-1} \sum_{i=1}^{i=k} \lambda^{k-i} y(i)\underline{\phi}(i)$$

c) In order to get a recursive algorithm, we need to apply a few tricks. Let's define

$$P(k) = \left\{\sum_{i=1}^{i=k} \lambda^{k-i}\underline{\phi}(i)\underline{\phi}^t(i)\right\}^{-1}$$

(*) With the above definition, show that

$$P^{-1}(k) = \lambda P^{-1}(k-1) + \underline{\phi}(k)\underline{\phi}^t(k)$$

d) Now we have the estimate at time k, $\underline{\hat{\theta}}(k)$ given by

$$\begin{aligned}\underline{\hat{\theta}}(k) &= P(k)\left\{\sum_{i=1}^{i=k}\lambda^{k-i}\underline{\phi}(i)y(i)\right\}\\ &= P(k)\left\{\lambda\sum_{i=1}^{i=k-1}\lambda^{k-1-i}\underline{\phi}(i)y(i)+\underline{\phi}(k)y(k)\right\}\end{aligned}$$

and

$$\underline{\hat{\theta}}(k-1) = P(k-1)\left\{\sum_{i=1}^{i=k-1}\lambda^{k-1-i}\underline{\phi}(i)y(i)\right\}$$

(*) Using these relationships, and the relationship derived in part (c), show that

$$\begin{aligned}\underline{\hat{\theta}}(k) &= \underline{\hat{\theta}}(k-1)+P(k)\underline{\phi}(k)\left\{y(k)-\underline{\phi}^t(k)\underline{\hat{\theta}}(k-1)\right\}\\ &= \underline{\hat{\theta}}(k-1)+K(k)\left\{y(k)-\underline{\phi}^t(k)\underline{\hat{\theta}}(k-1)\right\}\end{aligned}$$

where $K(k) = P(k)\underline{\phi}(k)$. Hence the estimate at time k, $\underline{\hat{\theta}}(k)$ is equal to the previous estimate at time $k-1$, $\underline{\hat{\theta}}(k-1)$, plus a correction term which is proportional to the error in the current estimate, $y(k)-\underline{\phi}(k)\underline{\hat{\theta}}(k-1)$.

e) Starting from

$$P(k) = \left\{\lambda P^{-1}(k-1)+\underline{\phi}(k)\underline{\phi}^t(k)\right\}^{-1}$$

(*) show that

$$P(k) = \frac{1}{\lambda}\left\{P(k-1)-P(k-1)\underline{\phi}(k)(\lambda+\underline{\phi}^t(k)P(k-1)\underline{\phi}(k))^{-1}\underline{\phi}^t(k)P(k-1)\right\}$$

Hint: Use the matrix inversion lemma

$$(A+BCD)^{-1} = A^{-1}-A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}$$

This is true if A, C, and $C^{-1}+DA^{-1}B$ are nonsingular square matrices.

f) Starting from

$$P(k) = \frac{1}{\lambda}\left\{P(k-1)-P(k-1)\underline{\phi}(k)(\lambda+\underline{\phi}^t(k)P(k-1)\underline{\phi}(k))^{-1}\underline{\phi}^t(k)P(k-1)\right\}$$

(*) show that

$$K(k) = P(k-1)\underline{\phi}(k)(\lambda+\underline{\phi}^t(k)P(k-1)\underline{\phi}(k))^{-1}$$

Hint: For us, $\underline{\phi}^t(k)P(k-1)\underline{\phi}(k)$ is a **scalar**, *λ is also a scalar.*

g) Combining the results from part (f) and part (e),
(*) show that

$$P(k) = \left\{I-K(k)\underline{\phi}^t(k)\right\}P(k-1)/\lambda$$

Hint: This should be very short work. If you are writing alot, you've missed it.

Summary Hence the recursive equations we need to implement are

$$\begin{aligned} \underline{\hat{\theta}}(k) &= \underline{\hat{\theta}}(k-1) + K(k)\left\{y(k) - \underline{\phi}^t(k)\underline{\hat{\theta}}(k-1)\right\} \\ K(k) &= P(k-1)\underline{\phi}(k)(\lambda + \underline{\phi}^t(k)P(k-1)\underline{\phi}(k))^{-1} \\ P(k) &= \left\{I - K(k)\underline{\phi}^t(k)\right\} P(k-1)/\lambda \end{aligned}$$

In order to perform the recursions, we need an initial value for $P(0)$ and an initial estimate for $\underline{\phi}(0)$. We will assume $P(0) = I$ and $\underline{\phi}(0) = \underline{0}$

Simulations

On the course web site is a Matlab routine to implement the recursive least squares algorithm. If, while running the Matlab routine, you see the message "singular to working precision" the results are garbage, even if they look nice. Also, to better see some of the results, use commands like

```
grid;
axis([0 400 -2 4]);
```

to have Matlab put a grid on a plot and scale the axis. (This will limit "x" from 0 to 400, and "y" from -2 to 4.) Note also that each time we run a system with noise, we will probably get different results. That's because noise is a *random process* and each time we get a sequence of noise values we are getting a different *realization* of the noise. You can tell Matlab to stop this but I don't want you to do this.

(h) For an input $u(k)$ of 400 samples of white Gaussian noise with mean zero and variance one, utilize the recursive least squares algorithm to estimate a plant which has transfer function

$$\begin{aligned} y(k) &= 0.25y(k-1) + 0.125y(k-2) + 2.0u(k) + 1.0u(k-1) \quad k < 200 \\ y(k) &= 1.5y(k-1) - 0.5625y(k-2) + 3.0u(k) - 1.0u(k-1) \quad 200 \leq k \end{aligned}$$

Assume $\lambda = 0.5$ and plot the estimates of the coefficients. (Note the simulation is set up to run this case, I just want you to go through it...) Print the output graph out and include it in your report. Do your estimates converge to the correct values?

(i) For the same simulation as in (h), try $\lambda = 0.5$, 0.8, 0.9, 0.95, and 1.000. Do these behave as you would expect? What is the effect of making λ larger or smaller? Include these graphs and your discussion in your report. One of the plots should show that your answers are clearly correct. (You may have to scale the axis to prove this to me...)

(j) Now assume the input is $u(t) = 2\cos(2\pi t/100)$ for $t = 1, ..., 400$ and use the same values of λ as in part (i). What happens? Do the parameters converge correctly? Include these graphs and discussion in your report. To understand what is happening, try to think about identifying a transfer function using a Bode plot (its frequency response), but you are only allowed to look at one frequency. Why won't this work?

(k) Now add a third test case,

$$\begin{aligned} y(k) &= 0.25y(k-1) + 0.125y(k-2) + 2.0u(k) + 1.0u(k-1) \;\; k < 200 \\ y(k) &= 1.5y(k-1) - 0.5625y(k-2) + 3.0u(k) - 1.0u(k-1) \;\; 200 \le k < 400 \\ y(k) &= 0.0y(k-1) + 0.25y(k-2) + 0.0u(k) + 3.0u(k-1) \;\; 400 \le k \end{aligned}$$

Assume the input is white noise as before. Run the simulation for various values of λ. (At least three, one of which should show convergence to the correct parameters.) What happens? Which value of λ would you choose if you wanted to track this system and avoid the large spikes? What are the trade-offs in the values of λ? Include at least three graphs and your discussion in your report. (Note: you may have to run the simulation for a 1000 steps or so to be sure of convergence.)

(l) To model a system slowly evolving over time, lets let

$$y(k) = (0.4 + 0.5\cos(3\pi k/200))y(k-1) + (-0.5 + 0.3\sin(2\pi k/200))y(k-2) + 0.3u(k) + 0.4u(k-1)$$

Again, let the input be white noise and try $\lambda = 0.5, 0.7$ and $\lambda = 1.0$.Run the simulation for 400 time samples again. Does the system track well? What is the tradeoff between the various values of λ ? Be sure to include at least three plots in your report as well as your discussion, and be sure you plotted the signals we were trying to track for this part. (The last plotting command should be uncommented to get a plot of the coefficients and the estimates...)

(m) We can improve our results somewhat by recognizing that the P_{last} matrix can become very close to singular, which makes our results become closer and closer to garbage. To correct for this, sometimes this matrix is reset. To see what happens, remove the comments from the part of the code that resets the P_{last} matrix and retry any of the previous simulations that did not seem to work very well. Do the results improve? Include graphs of the improved results in your report (as well as some discussion).

(n) Finally, for system identification a little noise can be a great help. Add some noise to the input cosine, so the input is now $u(t) = 2\cos(2\pi t/100) + n(t)$ where $n(t)$ is white Gaussian noise with mean zero and variance σ^2. (The standard deviation is σ). Simulate the system for the same case as in part (i). What is the smallest value of the standard deviation σ that allows good tracking with the smallest spikes? (You should also keep the changes you made to part (m) in effect for this part.)