ECE-520: Discrete-Time Control Systems

Recursive Least Squares Project

Due IN CLASS January 11, 2007

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 a lot 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 controller (lag, 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 controller (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.

Notes:

- In what follows, you may not assume that a vector has an inverse!!
- If we have and expression like $\sum_i p(i) = \sum_i q(i)$ we cannot multiply both of these by $a(i)$ call them equal, i.e., we cannot therefore conclude that $\sum_i p(i)a(i) = \sum_i q(i)a(i)$. You are an idiot if you do...and more importantly, you will loose alot of points!!

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

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

where

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

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

$$y(k) = ay(k-1) + by(k-2) + cu(k) + du(k-1)$$

(*) 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

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

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

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

(*) 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**.

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 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.

(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

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

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 turn it in. Do your estimates converge to the correct values?

(i) For the same simulation as in (h), try various values of λ, from $\lambda = 0.3$ to $\lambda = 1.0$. Do these behave as you would expect? What are the benefits and drawbacks of making λ larger or smaller? Things to look for here at the behavior of the system when the plant changes and the time it takes for the recursive least squares algorithm to reach the correct values. Print the graphs, label them, and turn them in. Be sure to turn in at least three plots. One of the plots should show that your answers are clearly correct.

(j) Often a value of λ between 0.9 and 1.0 is used for the recursive least squares algorithm. Of course this partly depends on how fast the system you are trying to track is moving. Try at least three values of λ between 0.9 and 1.0 and comment on the tradeoffs.

(k) Now add a second test case,

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

Assume the input is white noise as before, with an input of 600 samples. 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? What are the trade-offs in the values of λ? Again, print the output graphs and turn them in.

(l) Let's look at a system that is evolving slowly 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 400 samples of white noise and try $\lambda = 0.5, 0.7, 0.9$ and $\lambda = 1.0$. Does the system track well? What is the tradeoff between the various values of λ ? Turn in all four plots, and be sure you plotted the signals we were trying to track for this part. (The plotting commands near the end of the code should be uncommented to get a plot of the coefficients and the estimates...)

(m) If you look through the code, there are a couple of things we did not talk about, both of which have to do with the *covariance*, or P matrix. First of all, we need to be sure the matrix does not become singular, which probably will not happen for most of our cases. This tends to happen when our estimates have converged to the correct values, which is not as likely to happen if the input is noise. The second thing is that sometimes we can get better results if we regularly *restart* the P to force our algorithm to converge faster. This is like increasing the gain in a proportional controller, it amplifies the error and usually makes the system respond faster. However, it can also cause oscillations and other bad things to happen if we are not careful. Modifying P is not really part of the recursive least squares algorithm, but it is often done to improve the performance. The line of code

```
 if (rem(i,1000) == 0 ) P_last=100.0*eye(4); end;
```

is where we can require the P matrix to be updated. Rerun your examples from the previous parts for large values of λ (near or at 1) with P being updated every 10 or 20 samples. Turn in these plots and comment on the changes in the performance of the algorithm.

Summary

There are some things you should be aware of:

- The noise is actually a *random process*, and each time you run the simulations you get a different *realization* of a random process that have the same statistical properties. To really analyze the system you would need to run multiple simulations and average the results.
- In most real systems, the control input u would change as the model of the system changed. We did not do that, though it should be obvious how to do it.
- To really do a recursive least squares like we are doing, we need a *persistently exciting* input of a sufficient order. That is, our input must have sufficient spectral content to allow us to determine the system. The easiest way to think about this is to try and imagine trying to experimentally determine the Bode plot for an unknown system. In lab we know the systems are second order systems with no zeros, but we still need to look at a large number of input frequencies to get an accurate estimate of the Bode plot. Because we used white noise we did not have to worry about this. It does become more of a problem in a real control system.
- Besides least squares, there are many other (and better) adaptive methods for estimating the parameters.

Project Report

Your report should be neat, and should be computer generated (except for your derivation, which should be an appendix). The graphs should be included within the (word, LaTeX, etc.) document (not just stapled to it) and should have figure numbers and captions.