Νευρο-ασαφής Έλεγχος

9ο Εξάμηνο 2023 – 2024

Assignment 2 – Solutions

Ζαρίφης Στέλιος – el20435
Email: el20435@mail.ntua.gr

Contents

1 Ανάλυση χρονοσειρών της κατανάλωσης ηλεκτρικής ενέργειας στην Ελλάδα: AR Model Estimation με Cross-Validation

1.1 Οπτικοποίηση των δεδομένων

Τρέχουμε τον βοηθητικό κώδικα MATLAB για να φορτώσουμε τη χρονοσειρά και τη σχεδιάζουμε σε γράφημα.

```
el_load;
figure;
plot(el_lo)
title('Initial␣Data');
```

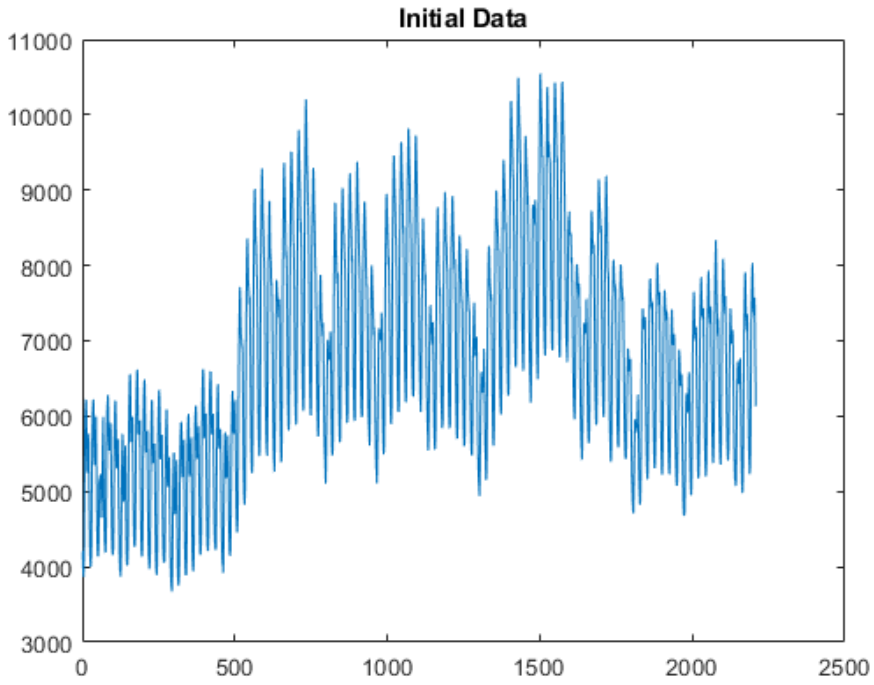


Figure 1: Initial Time Series Data

Μεγεθύνοντας στο γράφημα, μπορούμε να παρατηρήσουμε ότι η χρονοσειρά δεν είναι τυχαία, αλλά υπάρχουν μοτίβα που επαναλαμβάνονται. Λογικό είναι λοιπόν να προσπαθήσουμε να ”μάθουμε” αυτά τα μοτίβα με ένα AR μοντέλο.

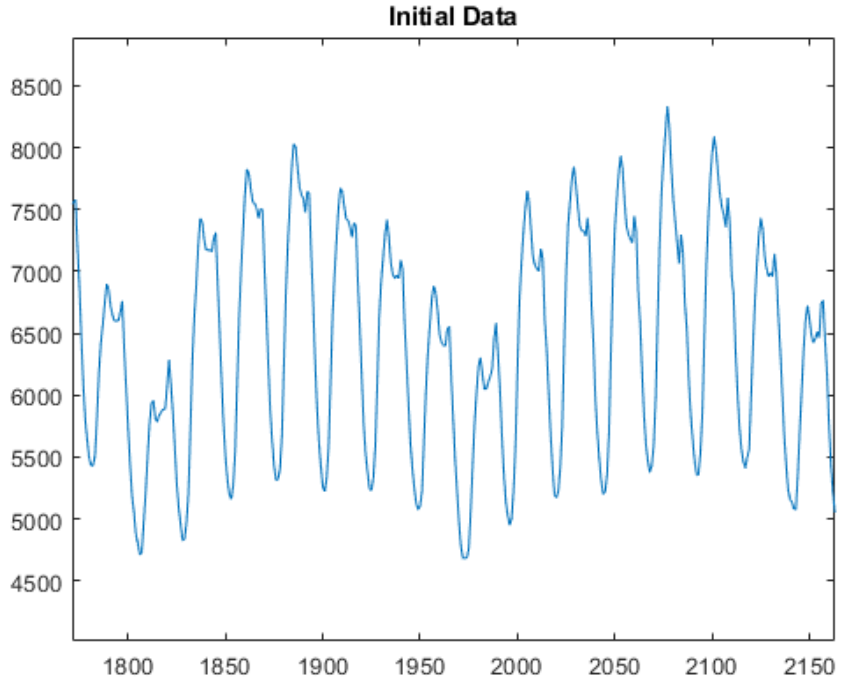


Figure 2: Initial Time Series Data Zoomed

Autoregressive (AR) Models Τα Autoregressive (AR) μοντέλα είναι μια κατηγορία μοντέλων για χρονοσειρές που περιγράφουν τις σχέσεις μέσα σε υπακολουθίες παρατηρήσεων. Συγκεκριμένα, ένα AR μοντέλο αναπαριστά κάθε τιμή ως γραμμικό συνδυασμό παρελθόντων τιμών.
Η μορφή ενός AR(p) μοντέλου είναι:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \epsilon_t$$

όπου:

- Y_t η παρατήρηση της χρονικής τιμής t.
- c μια σταθερά.
- $\phi_1, \phi_2, \ldots, \phi_p$ οι autoregressive συντελεστές.
- $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ οι (p το πλήθος) παρελθούσες τιμές.
- ϵ_t ο error όρος τη στιγμή t.

Παρατήρηση: Για να είναι ακριβές ένα AR μοντέλο, η χρονοσειρά που επιδιώκει να περιγράψει πρέπει να είναι stationary, δηλαδή οι στατιστικές της ιδιότητες να μη μεταβάλονται με το πέρασμα του χρόνου.

1.2 Deseasonalization Χρονοσειράς

Αρχικά προβάλουμε τη διακύμανση του φορτίου ανά ημέρα και υπολογίζουμε το μέσο όρο της χρονοσειράς σε κάθε ώρα. Αφαιρούμε αυτόν το μέσο όρο (ημερήσια τάση) από τα δεδομένα μας.

```
%% Step 2: Perform Data Preprocessing (Deseasonalization)
jj_max = length(el_lo) / 24;
```

```
% Rebuild as a matrix with day and time
for ii = 1:24
        for jj = 1:jj_max
                el_lo_mat(ii, jj) = el_lo((jj - 1) * 24 + ii);
        end
end

figure;
plot(el_lo_mat)
title('Hourly␣Electricity␣Load');

% Compute the mean load for each time of the day
el_lo_mean = mean(el_lo_mat.');

% Compute the deseasonalized load
for ii = 1:24
        for jj = 1:jj_max
                el_lo_des((jj - 1) * 24 + ii) = el_lo_mat(ii, jj) - el_lo_mean(
        end
end

figure;
plot(el_lo_des)
title('Deseasonalized␣Electricity␣Load');
```

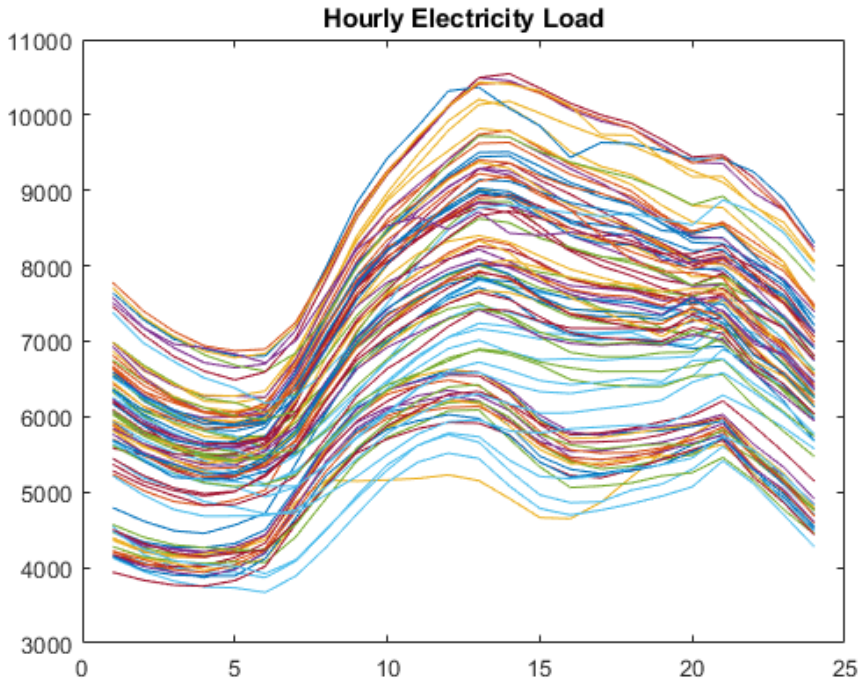


Figure 3: Stacked Hourly Data

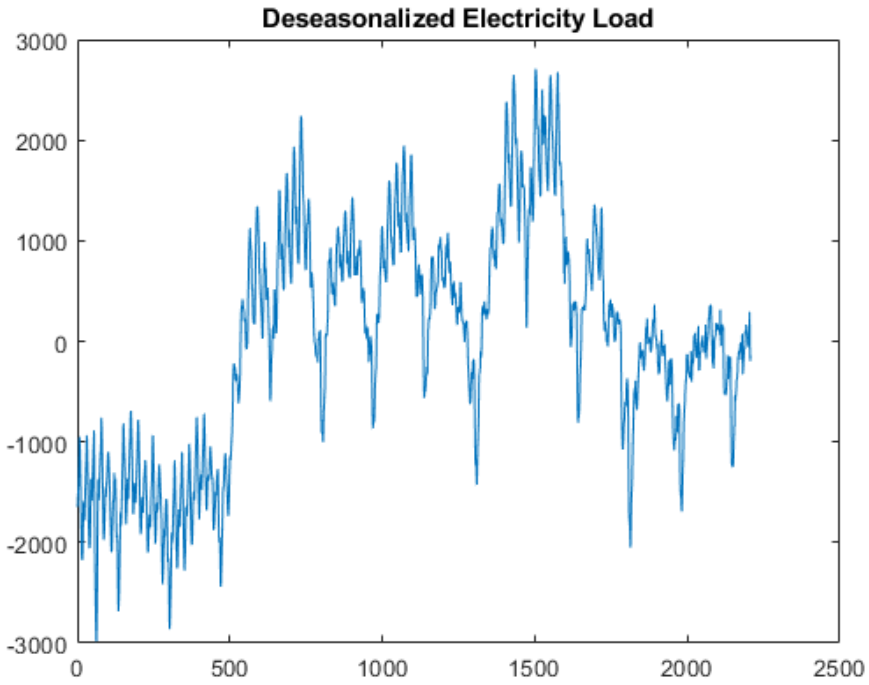


Figure 4: Data After Subtracting the Daily Mean Value

Μετά την αφαίρεση της ημερήσιας τάσης, μπορούμε να παρατηρήσουμε εβδομαδιαία και μηνιαία τάση. Για παράδειγμα χαμηλή κατανάλωση τα Σαββατοκύριακα και υψηλή τους καλοκαιρινούς μήνες.

1.3 Επιλογή AR Model Order και Εκπαίδευση

Στον παρακάτω κώδικα εφαρμόζουμε cross-validation για να επιλέξουμε το βέλτιστο autoregressive order ώστε να περιγράψουμε τα δεδομένα.

1. Δοκιμές Orders: Εξετάζουμε orders από 1 ως 10.
2. Cross-validation: Για διάφορα training-validation splits (50% - 10%, 60% - 10%, 70% - 10%, 80% - 10%, 90% - 10%, πάντα το training γίνεται σε παρατηρήσεις που συνέβησαν πριν τις παρατηρήσεις του validation, λόγω της ακολουθιακότητας των δεδομένων), εξετάζουμε AR μοντέλα από κάθε order.
3. Επιλογή order: Υπολογίζουμε το mean squared error (MSE) για κάθε order στο validation set. Το order με το χαμηλότερο MSE στο validation set επιλέγεται ως το βέλτιστο για το συγκεκριμένο split.
4. Συνολικό Βέλτιστο Order: Μετά την εξέταση όλων των cross-validation splits, επιλέγουμε το συνολικό βέλτιστο AR model order ως εκείνον που επιλέχθηκε βέλτιστος στα περισσότερα spilits.

```
%% Step 3: Choose the Order of the AR Model using Cross-Validation
% Define a range of AR model orders to consider
order_range = 1:10;

% Define the percentage splits for training-validation
split_percentages = [50, 60, 70, 80, 90];
val_percentage = 10;
```

```
% Initialize variables to store the count of each order as the best
order_counts = zeros(1, length(order_range));

% Perform cross-validation for each split percentage
for split_percentage = split_percentages
        fprintf('Training-Validation Split: %d-%d\n', split_percentage, val_perc

        % Calculate the number of observations for training and validation
        train_split_index = round(split_percentage / 100 * length(el_lo_des));
        val_split_index = round((split_percentage + val_percentage) / 100 * leng
        num_train = train_split_index;
        num_val = val_split_index;

        % Initialize variable to store the best order for this split
        best_order_for_split = -1;

        % Initialize variable to store the lowest MSE for this split
        lowest_mse_for_split = inf;

        % Split data into training and validation sets
        training_data = el_lo_des(1:train_split_index);
        validation_data = el_lo_des(train_split_index + 1:val_split_index);
        fprintf('Length of train data: %d\n', length(training_data))
        fprintf('Length of val data: %d\n', length(validation_data))
        % Perform cross-validation
        for i = 1:length(order_range)
                order = order_range(i);
                % Fit an AR model of the selected order to the training data
                ar_model = ar(training_data, order);

                % Use the AR model to simulate future values on the validation
                simulated_data = filter(ar_model.A, 1, validation_data);

                % Calculate the MSE for the current order
                mse = mean((validation_data - simulated_data).^2);
                % Update the best order and lowest MSE if the current MSE is lo
                if mse < lowest_mse_for_split
                        fprintf('Found better order: %d\n', order)
                        lowest_mse_for_split = mse;
                        best_order_for_split = order;
                end
        end

        % Update the count for the best order
        fprintf("Best order for split: %d\n", best_order_for_split)
        order_counts(best_order_for_split) = order_counts(best_order_for_split)
end

% Find the overall best order based on the global count
```

```
[~, overall_best_order] = max(order_counts);

fprintf('Overall Best AR Model Order: %d\n', overall_best_order);

%% Step 4: Estimate AR Model Parameters using the overall best order
training_data = el_lo_des(1:end-24);  % Exclude the last day for training
overall_best_ar_model = ar(training_data, overall_best_order);

%% Step 5: Simulate future values on the last day
simulated_data_last_day = filter(overall_best_ar_model.A, 1, el_lo_des(end-23:en

%% Step 6: Plot Actual and Predicted Data for the Last Day
figure;
plot(el_lo_des(end-23:end), 'b', 'DisplayName', 'Actual Data (Last Day)');
hold on;
plot(simulated_data_last_day, 'r', 'DisplayName', 'Predicted Data (Overall Best
title('Actual vs. Predicted Data for the Last Day');
legend;
xlabel('Time');
ylabel('Value');
```

Το αποτέλεσμα φαίνεται εδώ:

```
Training-Validation Split: 50-10
Length of train data: 1104
Length of val data: 221
Found better order: 1
Best order for split: 1
Training-Validation Split: 60-10
Length of train data: 1325
Length of val data: 221
Found better order: 1
Best order for split: 1
Training-Validation Split: 70-10
Length of train data: 1546
Length of val data: 220
Found better order: 1
Best order for split: 1
Training-Validation Split: 80-10
Length of train data: 1766
Length of val data: 221
Found better order: 1
Best order for split: 1
Training-Validation Split: 90-10
Length of train data: 1987
Length of val data: 221
Found better order: 1
Best order for split: 1
Overall Best AR Model Order: 1
```

Παρατηρούμε ότι παρόλο που όσο μεγαλύτερο είναι το order, τόσο μικρότερο είναι το σφάλμα εκπαίδευσης, επιλέχθηκε order 1 καθώς ήταν το βέλτιστο γενικά, χάρη στη διαδικασία Cross-Validation. Εξετάζοντας διαφορετικά splits εμποδίζουμε το μοντέλο από το να απομνημονεύσει τα training δεδομένα. Το αποτέλεσμα είναι η αποφυγή του overfitting και το μοντέλο να γενικεύει καλύτερα. Βλέπουμε και την εκτίμηση για την τελευταία ημέρα του dataset.

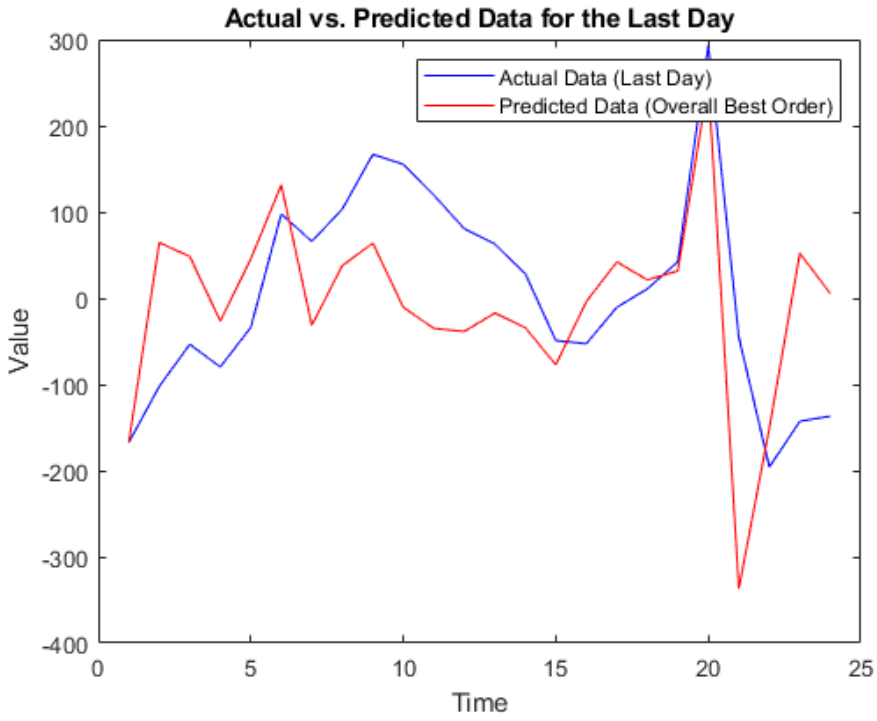


Figure 5: Last Day Load Prediction

Παρατήρηση: Αν είχαμε εκπαιδεύσει το μοντέλο πάνω σε όλο το dataset εκτός της τελευταίας ημέρας, χωρίς Cross-Validation, μπορεί η πρόβλεψη να είναι ακριβέστερη, όπως φαίνεται παρακάτω, αλλά όπως είπαμε πριν, το μοντέλο δε γενικεύει καλά.

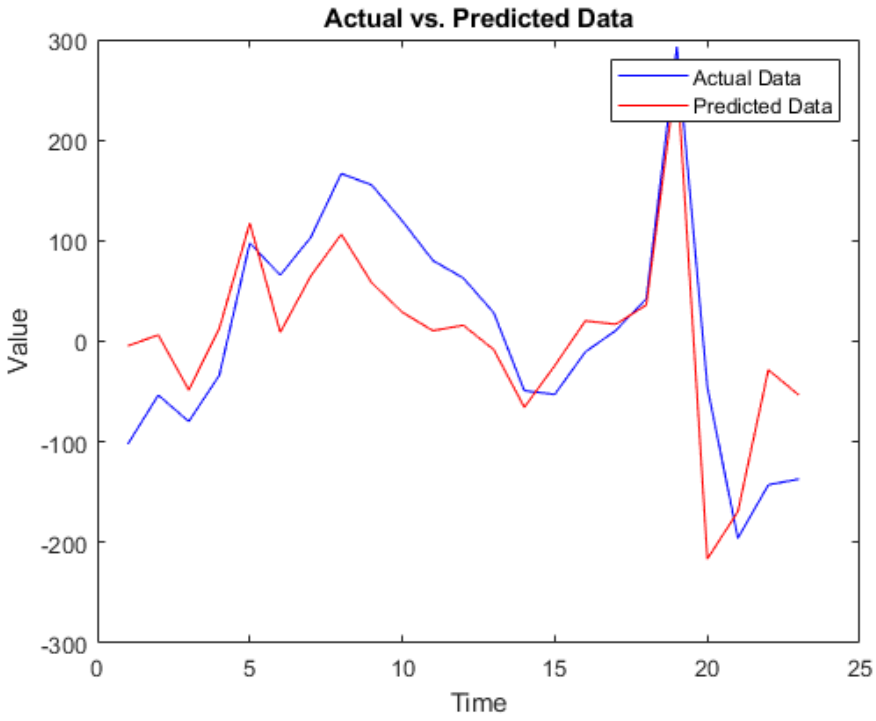


Figure 6: Last Day Load Prediction Without Cross-Validation

2 Εκτίμηση Παραμέτρων με Least Squares & Lasso

Έστω το γραμμικό μοντέλο με ανεξάρτητες μεταβλητές $x_i \in \mathbb{R}^{80}$:

$$y_i = c^T x_i + 0.5\epsilon_i, c^T = [\text{rand}(1,10)\,\text{zeros}(1,60)\,\text{rand}(1,10)]$$

Και ϵ_i ομοιόμορφα κατανεμημένες ανεξάρτητες κανονικές τυχαίες μεταβλητές (με μοναδιαίο variance).

2.1 Εκτίμηση Παραμέτρων με Least Squares

Η μέθοδος least squares επιδιώκει να εκτιμήσει τις παραμέτρους του γραμμικού μοντέλου ελαχιστοποιώντας το άθροισμα των τετραγώνων των διαφορών μεταξύ παρατηρήσεων και εκτιμήσεων. Για ένα γραμμικό μοντέλο με μορφή $y = Xc + \epsilon$, όπου y είναι το διάνυσμα των παρατηρήσεων, X ο πίνακας εισόδου, c το διάνυσμα των παραμέτρων που θα εκτιμηθεί, και ϵ το διάνυσμα των errors, η λύση των ελαχίστων τετραγώνων για το c προκύπτει:

$$c_{\text{LS}} = (X^T X)^{-1} X^T y$$

Με τον παρακάτω κώδικα, κατασκευάζουμε το γραμμικό μοντέλο και προβάλουμε σε ένα γράφημα την εκτίμηση με Least Squares.

```
rng(42);

% Generate data
c = [randn(1, 10), zeros(1, 60), randn(1, 10)];
x = randn(80, 100);
y = c * x + 0.5 * randn(100, 1)';

% x = x', y = y' -> swap
% Least Squares Estimation
F_inv = inv(x * x');
c_ls = F_inv * (x * y');

% Plot true and least squares estimated coefficients (c and c_ls)
figure;
plot(c, 'b', 'DisplayName', 'True Coefficients');
hold on;
plot(c_ls, 'r', 'DisplayName', 'Least Squares Estimated Coefficients');
title('True and Least Squares Estimated Coefficients');
xlabel('Index');
ylabel('Coefficient Value');
legend;
```

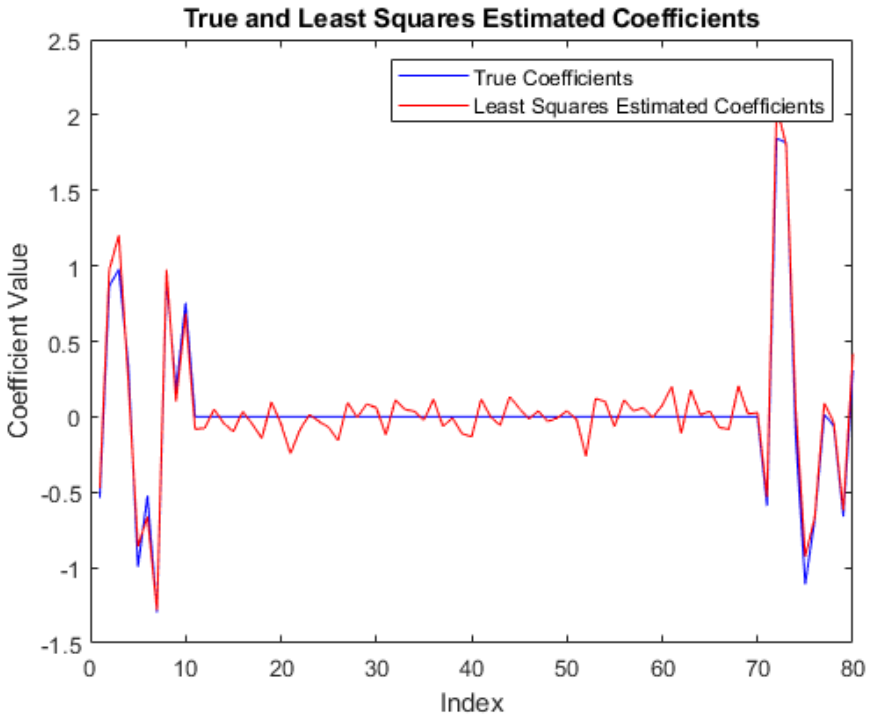


Figure 7: Least Squares Estimation

Παρατηρούμε ότι στην προσπάθειά μας να μάθουμε τις σχέσεις που παράγουν τα πρώτα 10 και τελευταία 10 σημεία, προκύπτουν σχέσεις και στα ενδιάμεσα 60. Αυτές οι σχέσεις δεν απεικονίζουν την πραγματικότητα, αλλά προκύπτουν στην προσπάθεια ελαχιστοποίησης του αθροίσματος των τετραγώνων. Το κάθε y_i εξαρτάται από όλα τα x_i και εκείνα τα y_i που είναι μη μηδενικά απαιτούν τέτοιους συντελεστές που δεν είναι δυνατόν να παράξουν 60 μηδενικά δείγματα στη μέση.

2.2 Εκτίμηση Παραμέτρων με Lasso

Η μέθοδος Lasso (Least Absolute Shrinkage and Selection Operator) εισάγει έναν όρο κανονικοποίησης στην εκτίμηση least squares:

$$\text{minimize} \left\{ \frac{1}{2N} \|y - Xc\|_2^2 + \lambda \|c\|_1 \right\}$$

Όπου λ είναι η παράμετρος κανονικοποίησης που ελέγχει την ισχύ του penalty πάνω στις απόλυτες τιμές των συντελεστών. Το penalty του Lasso οδηγεί σε πιο αραιό διάνυσμα συντελεστών, δηλαδή μια μορφή feature selection.

```
% Lasso Estimation
[B, FitInfo] = lasso(x', y, 'CV', 10);

fprintf("Plot Lasso results\n")
% Plot Lasso results
figure;
lassoPlot(B, FitInfo, 'PlotType', 'CV');
legend('show');
title('Lasso Results');
```

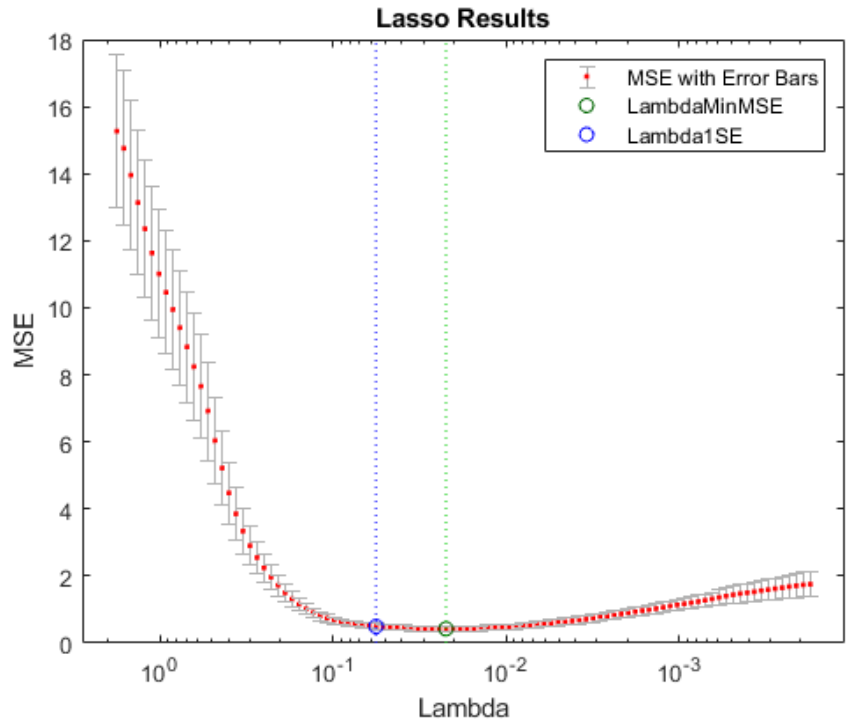


Figure 8: Lasso Estimation Error

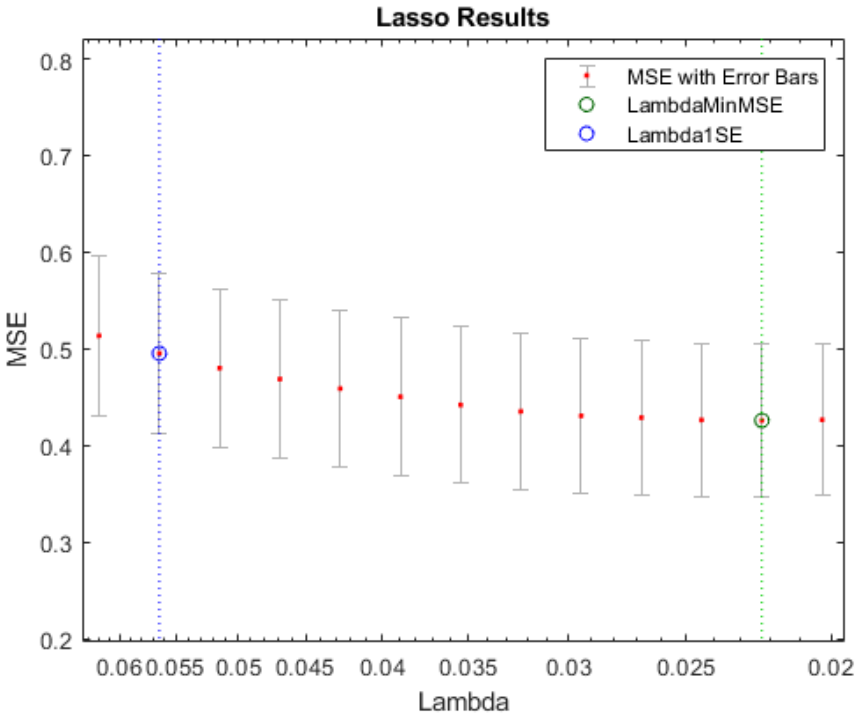


Figure 9: Lasso Estimation Error Zoomed

Μεγεθύνοντας, βλέπουμε πως η συνάρτηση του MATLAB για την εκτίμηση Lasso υπολογίζει το λ για το οποίο έχουμε ελάχιστο σφάλμα και την αβεβαιότητα κάθε μέτρησης. Τέλος διαλέγει το μεγαλύτερο λ για το οποίο έχουμε σφάλμα που βρίσκεται εντός της εκτίμησης της αβεβαιότητας. Ακόμα παρατηρούμε ότι με $\lambda \to 0$ η εκτίμηση Lasso εκφυλίζεται σε εκτίμηση Least Squares ενώ με μεγάλο λ το σφάλμα εκρήγνυται διότι το

πολύ μεγάλο penalty δεν επιτρέπει μεγάλες αλλαγές στις εκιμήσεις των συντελεστών, δηλαδή ο εκτιμητής δεν προσαρμόζεται καλά.
Στον κώδικα εφαρμόζουμε Lasso εκτίμηση με 10-fold cross-validation (έτσι προκύπτει και η εκτίμηση της αβεβαιότητας που αναφέρθηκε).

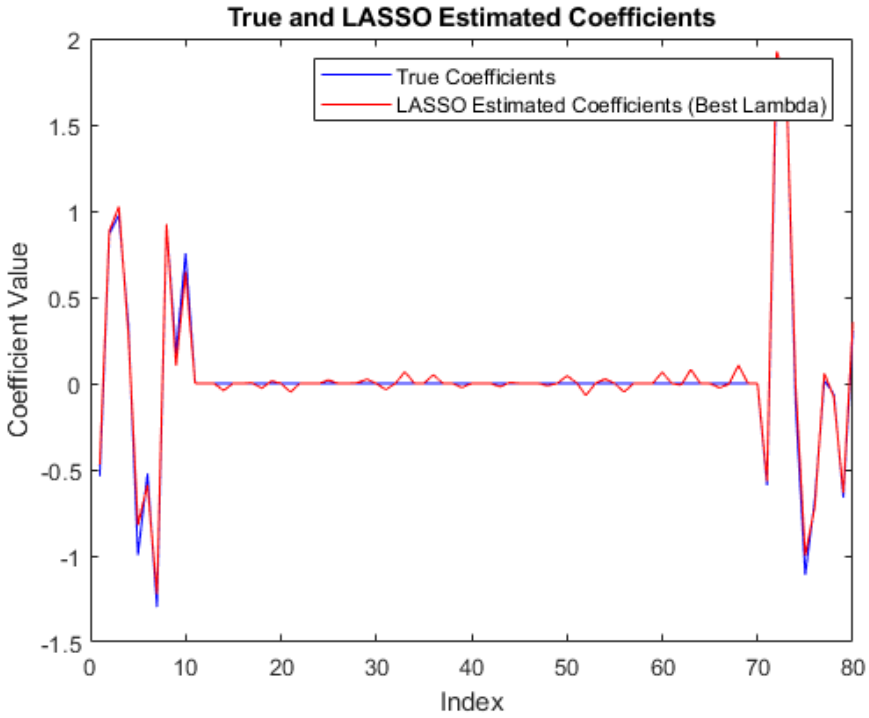


Figure 10: Lasso Estimation

Παρατηρούμε πολύ πιο ακριβή εκτίμηση συντελεστών.

3 Εκτίμηση Παραμέτρων σε Σύστημα με Εκκρεμές: Gradient Descend και Σύγκλιση

Οι εξισώσεις που περιγράφουν το σύστημα είναι

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\alpha_1 \sin x_1 - \alpha_2 x_2 + bu \end{cases}$$

3.1 Εκτιμητής σε Σύστημα χωρίς Θόρυβο

Έχουμε το σύστημα χωρίς θόρυβο στην παρατήρηση της εξόδου

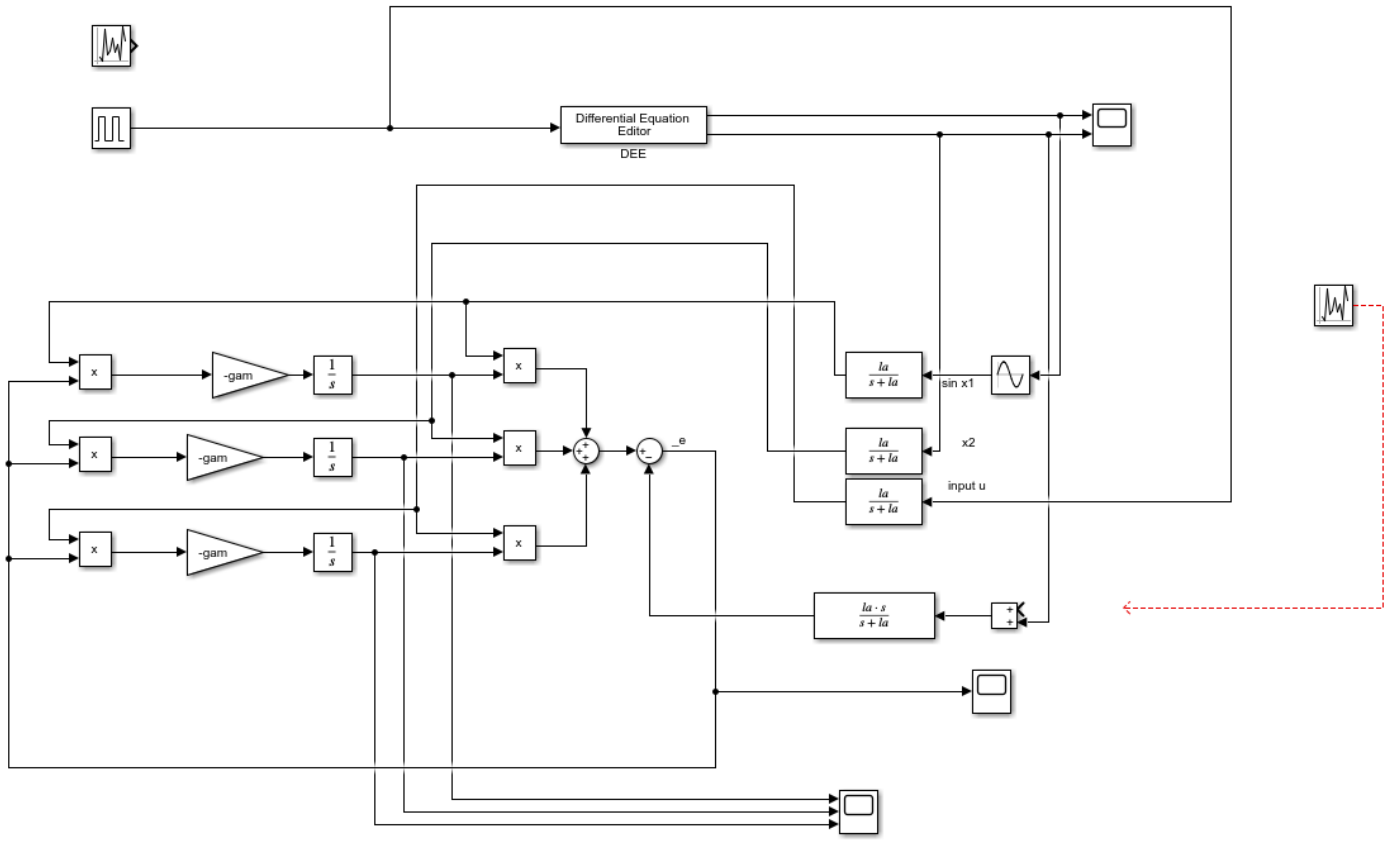


Figure 11: Estimator Without Noise

Διεγείρουμε το σύστημα με τετραγωνική παλμοσειρά πλάτους 0.2 και περιόδου 200:

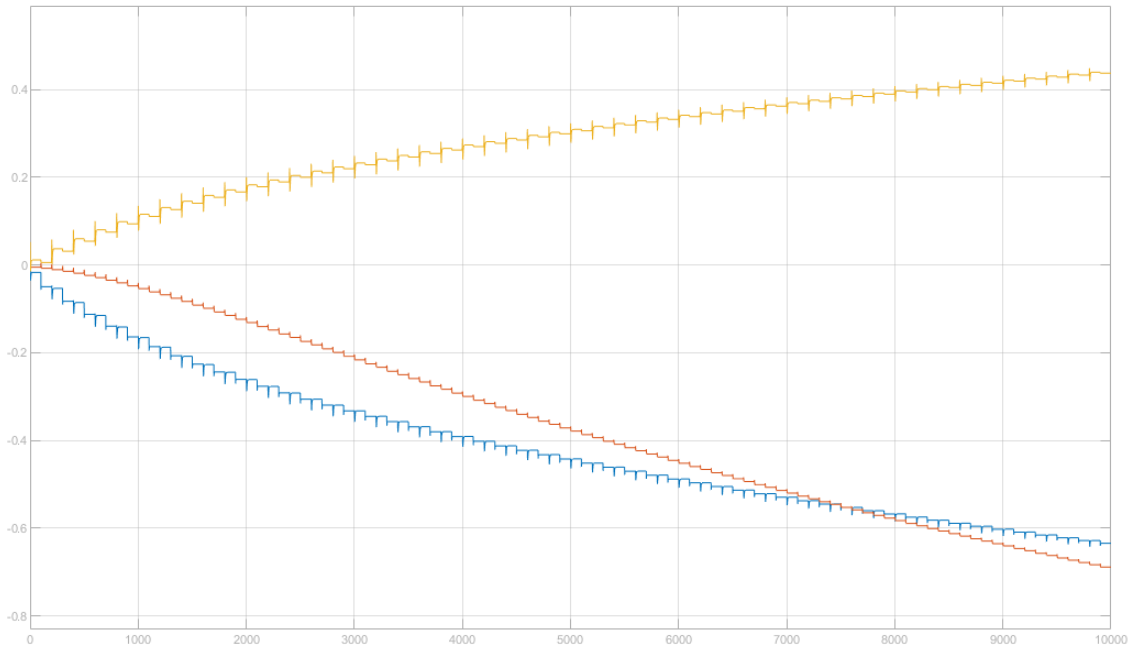


Figure 12: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$

Μειώνουμε την περίοδο για να αυξήσουμε το ρυθμό μάθησης ώστε να παρατηρήσουμε τη σύγκλιση (όσο περισσότερο διεγείρουμε το σύστημα, τόσο περισσότερο θα μάθουμε τις παραμέτρους):

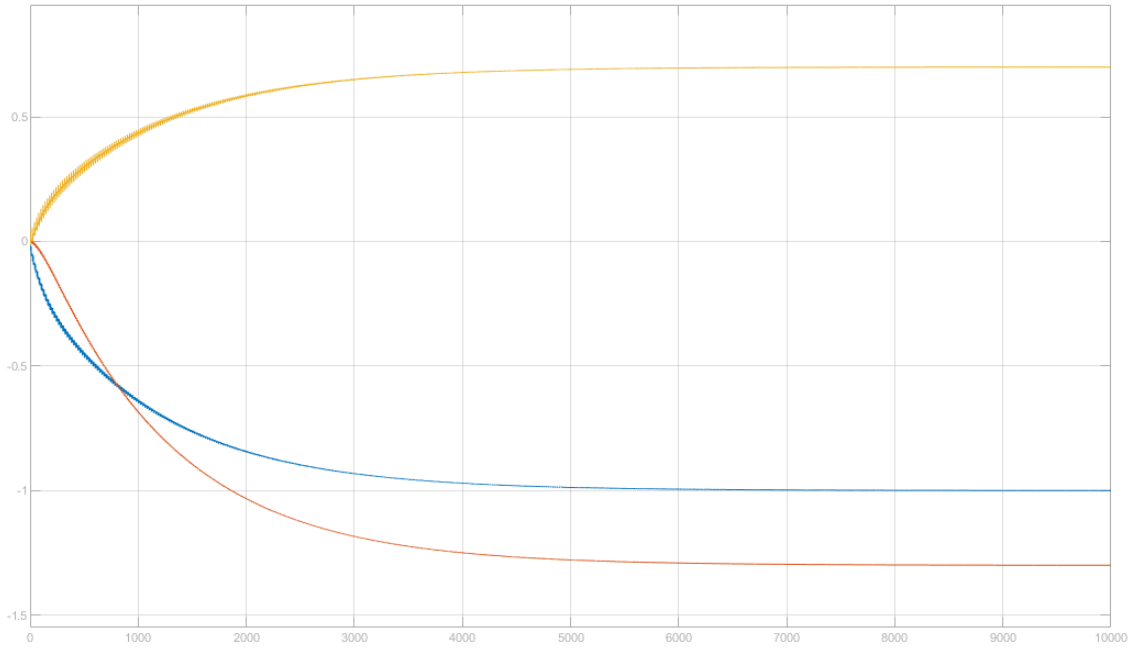


Figure 13: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$

Και βλέπουμε πως το σύστημα με κάθε παλμό μαθαίνει και καλύτερα τις παραμέτρους:

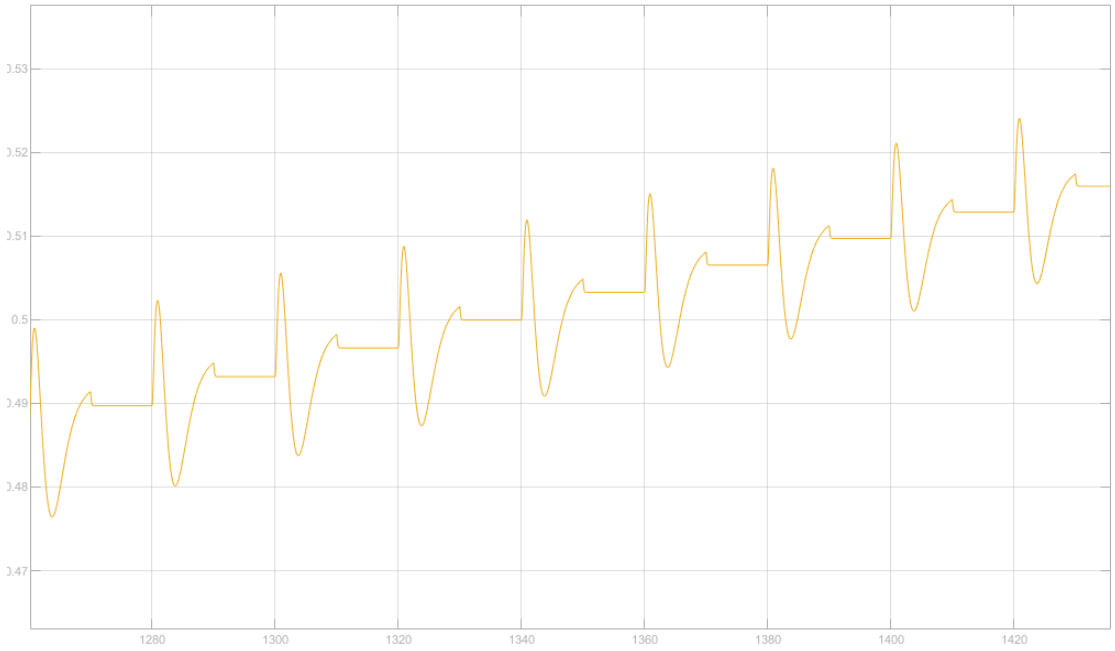


Figure 14: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ - zoomed

Δοκιμάζουμε με $\lambda = 100, \gamma = 20$ και παρατηρούμε πως το σύστημα μαθαίνει πιο γρήγορα:

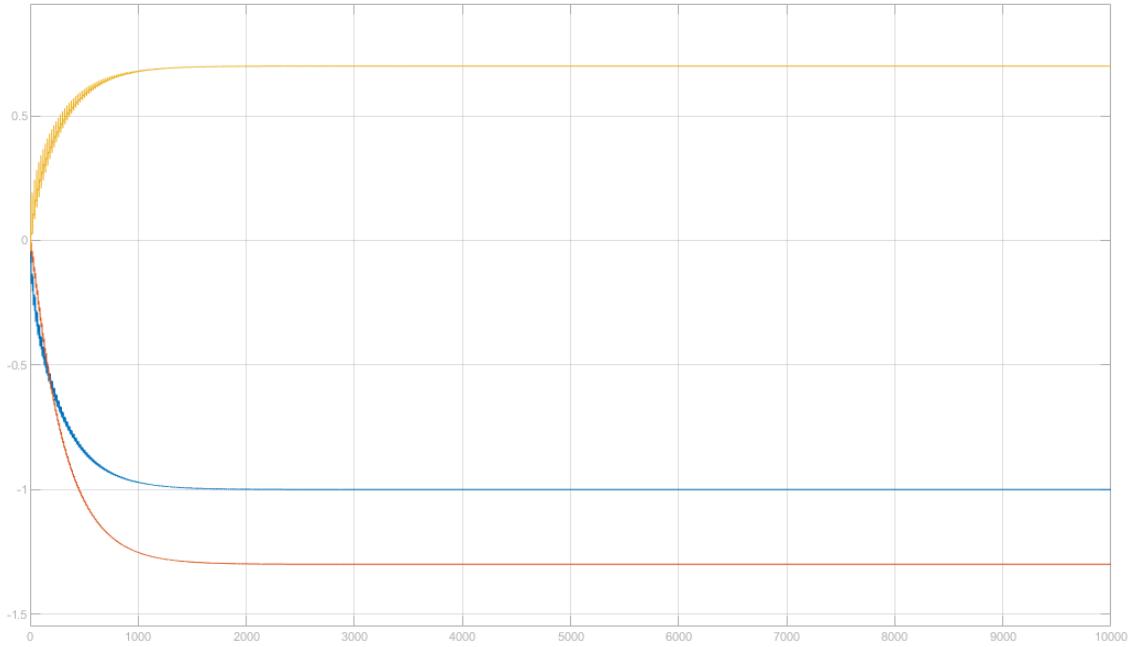


Figure 15: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 100, \gamma = 20$

Φαίνεται ότι σε αυτό το ιδανικό σύστημα, όσο πιο μεγάλες τιμές έχουν οι παράμετροι, τόσο πιο γρήγορη θα είναι η σύγκλιση. Η επιλογή των παραμέτρων όμως σε ένα πραγματικό σύστημα εξαρτάται από το θόρυβο, όπως θα δούμε στη συνέχεια.

3.2 Εκτιμητής σε Σύστημα με Θόρυβο

Έστω το σύστημα με θόρυβο στη μέτρηση της ταχύτητας:

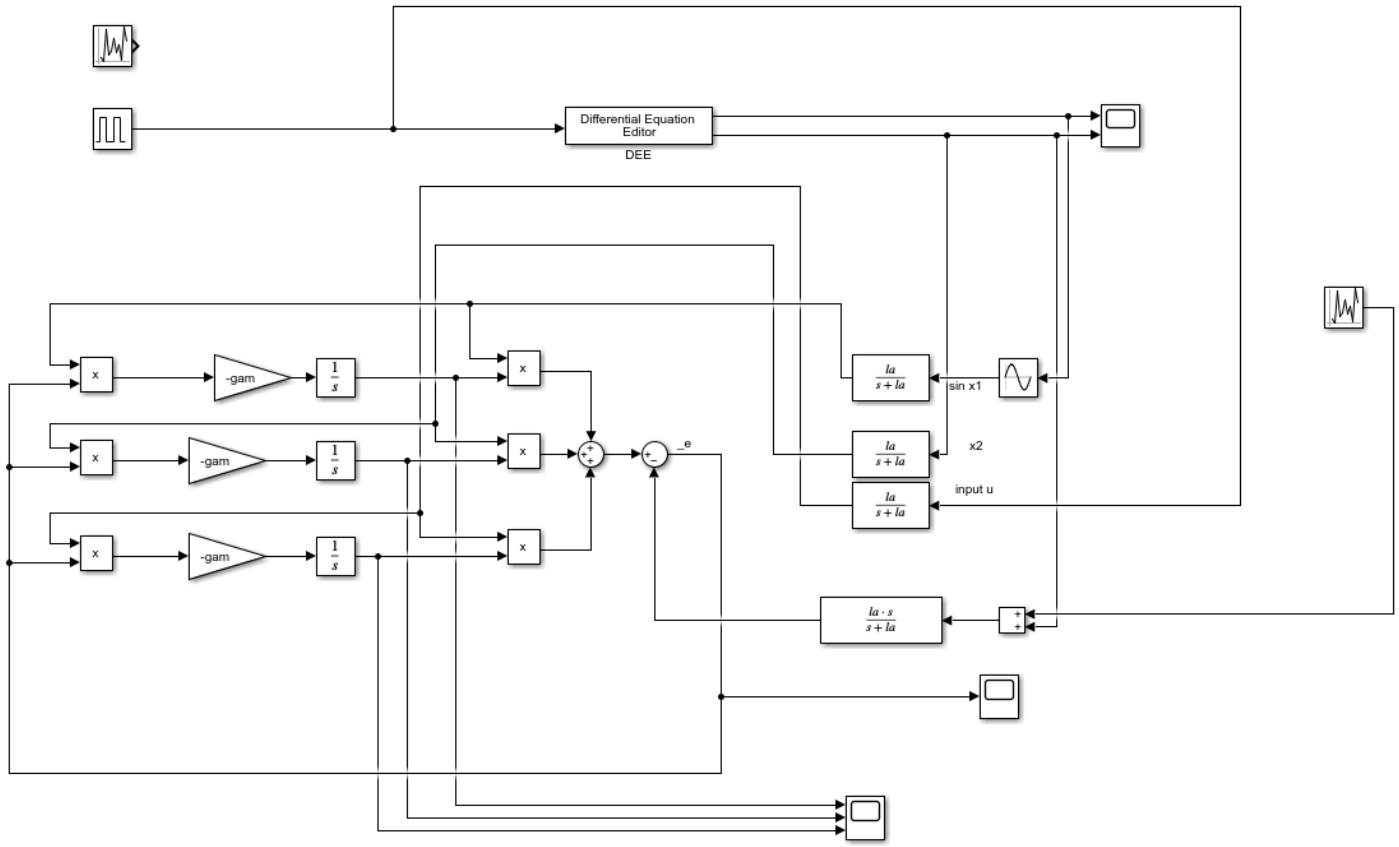


Figure 16: Estimator With Noise

Δοκιμάζουμε τις τελευταίες παραμέτρους ($\lambda = 100, \gamma = 20$) στο σύστημα με θόρυβο και βλέπουμε πως το σφάλμα είναι μεγάλο:

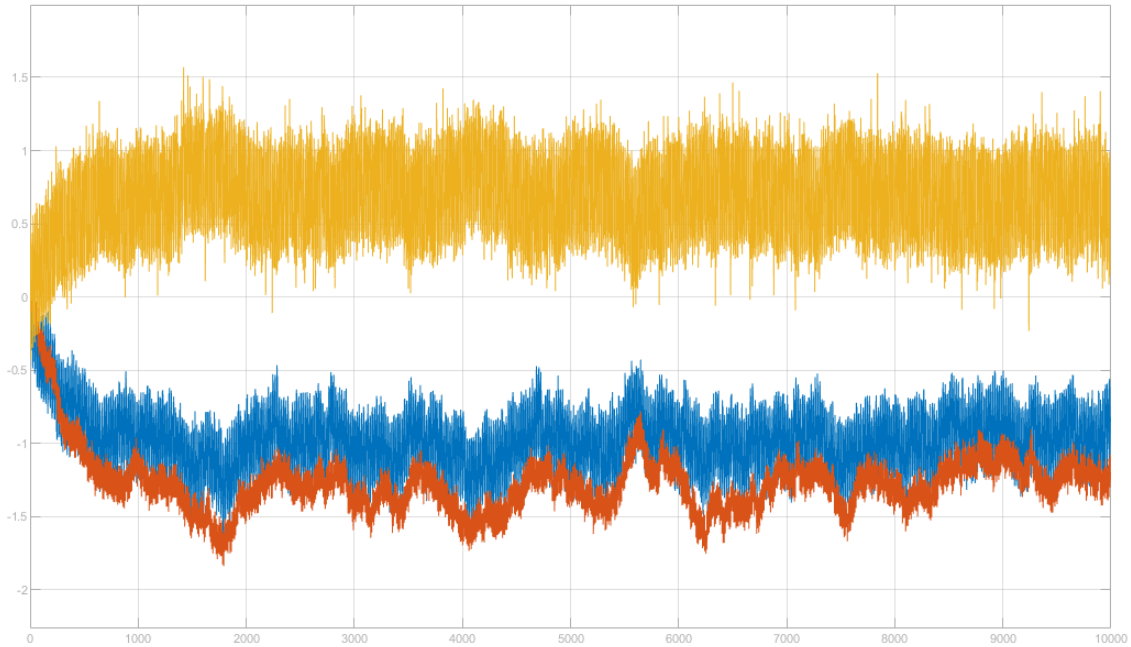


Figure 17: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 100, \gamma = 20$ and noise

Μειώνουμε τις παραμέτρους πάλι σε $\lambda = 10, \gamma = 5$ και παρατηρούμε μικρότερο σφάλμα:

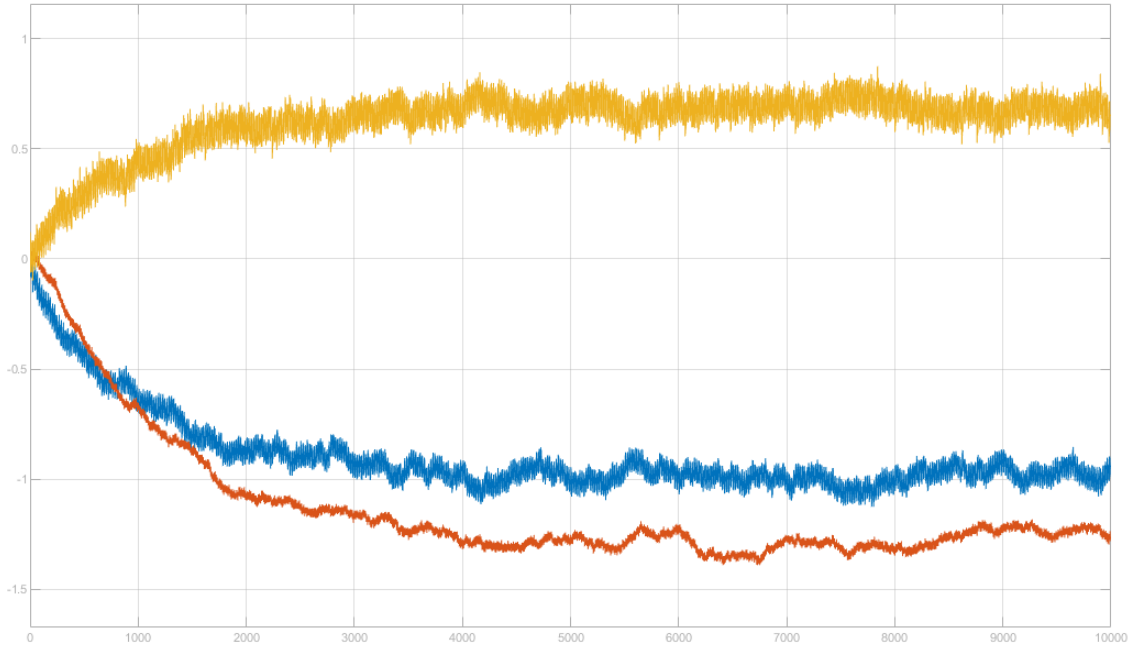


Figure 18: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and noise

3.3 Σύγκλιση για μεταβλητή είσοδο

Τώρα, η διέγερση αντί για παλμοσειρά θα είναι τυχαία:

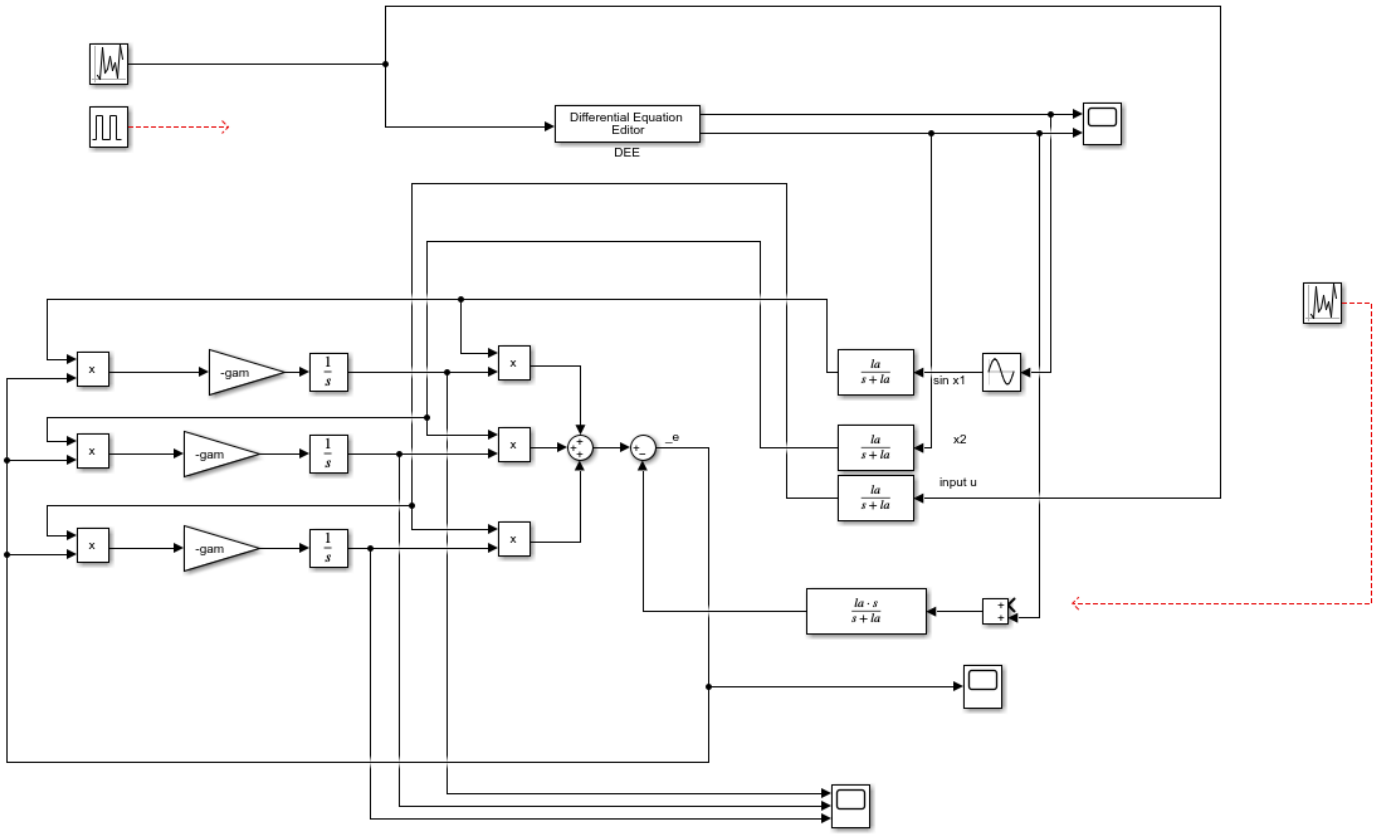


Figure 19: Estimator With Random Excitement

Για $\lambda = 10, \gamma = 5$ έχουμε:

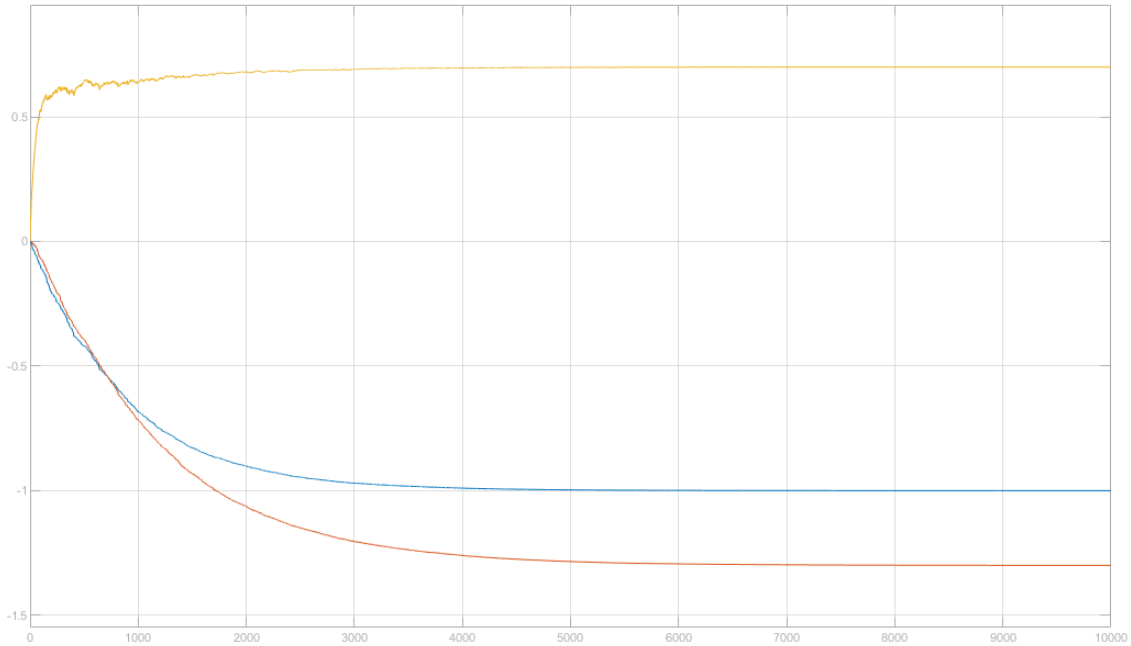


Figure 20: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and Random Excitement

Και για $\lambda = 100, \gamma = 20$ έχουμε:

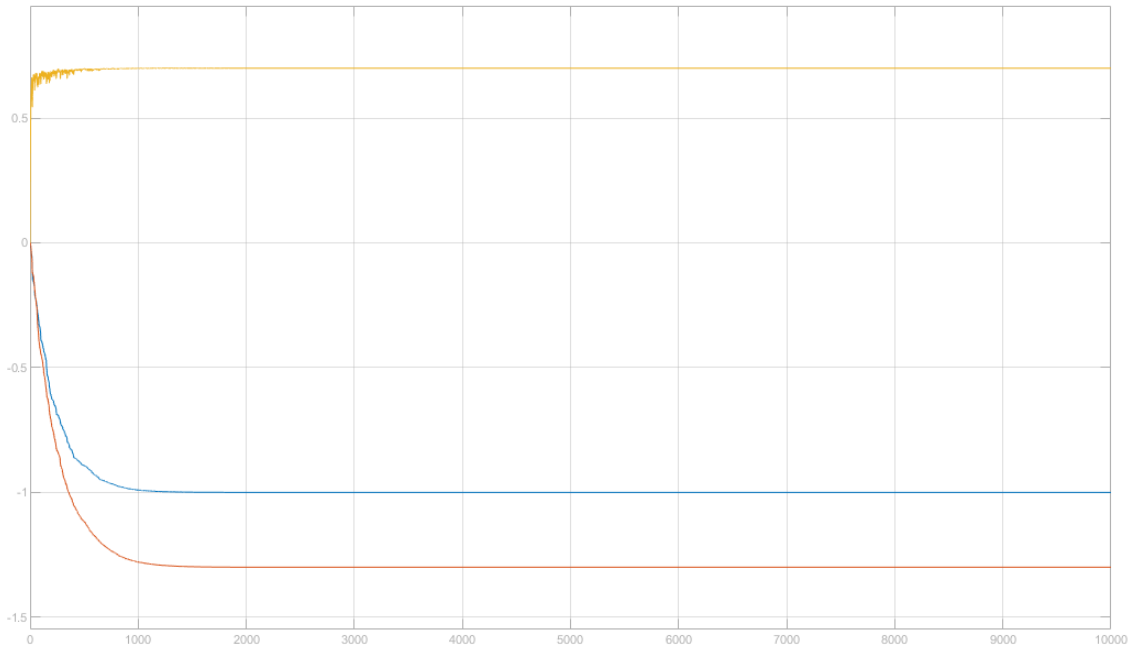
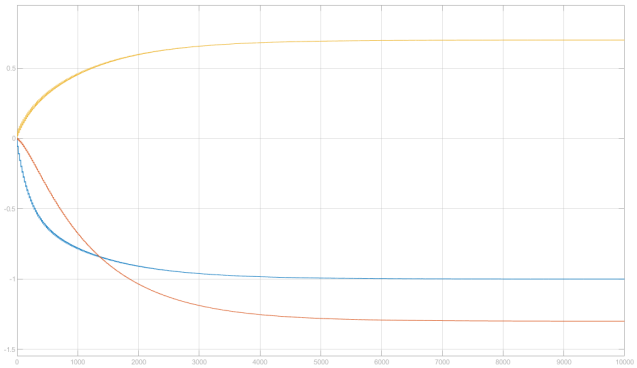


Figure 21: Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 100, \gamma = 20$ and Random Excitement

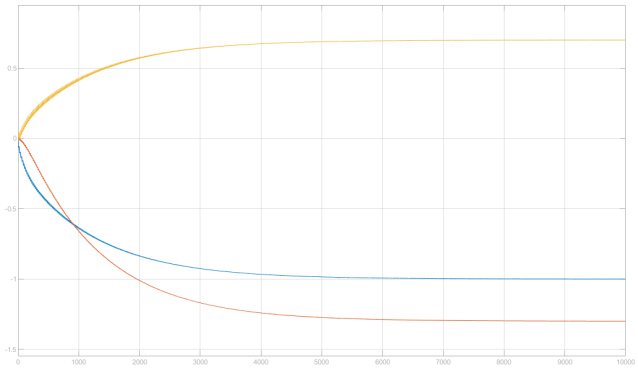
Παρατηρούμε ότι με τυχαία διέγερση μαθαίνει εξίσου καλά αν όχι καλύτερα. Συνεπώς, αρκεί οποιαδήποτε διέγερση για να μάθουμε τις παραμέτρους του συστήματος.

3.4 Ταχύτητα Σύγκλισης και Περίοδος Παλμοσειράς

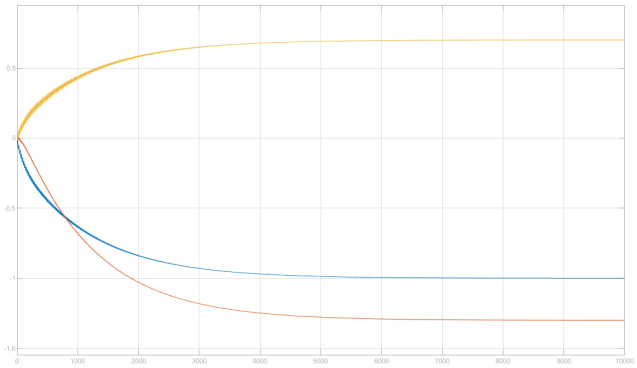
Δοκιμάζουμε περιόδους $10, 30, 70, 90$. Παρατηρούμε ταχύτερη σύγκλιση για μικρότερη περίοδο - μεγαλύτερη συχνότητα. Διαισθητικά είναι σα να διεγείρουμε περισσότερες φορές το σύστημα ανά μονάδα χρόνου, άρα ταχύτερα καταλαβαίνουμε τις σχέσεις και μπορούμε να το μάθουμε.



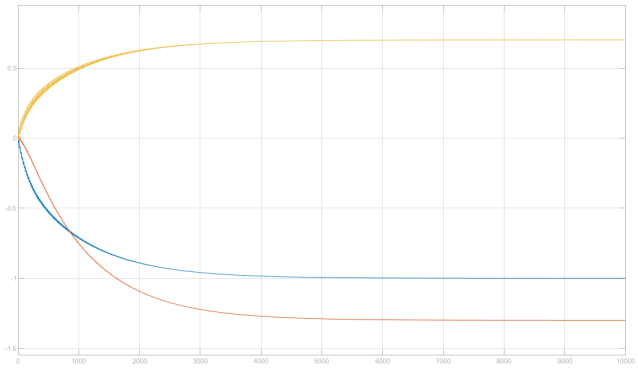
(α΄) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse period 10



(β΄) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse period 30



(γ΄) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse period 70

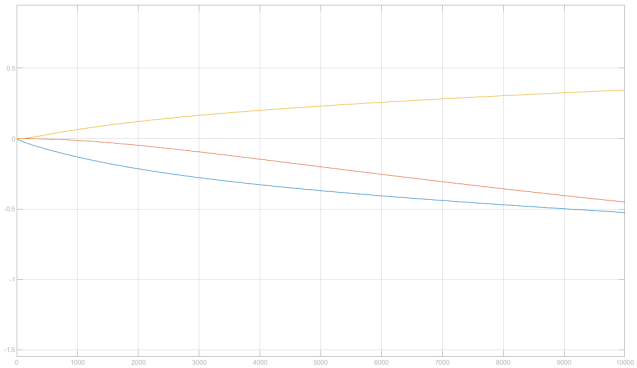


(δ΄) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse period 90

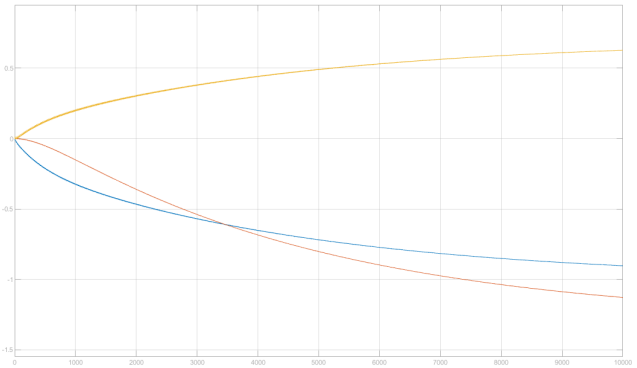
Figure 22: Estimations for different pulse periods

3.5 Ταχύτητα Σύγκλισης και Πλάτος Παλμοσειράς

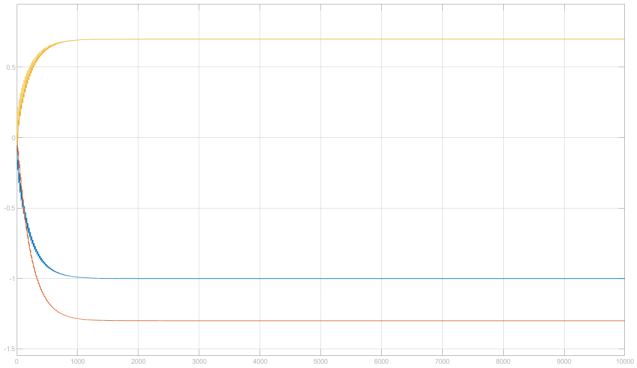
Εξετάζουμε πλάτη $0.05, 0.1, 0.5, 1$. Βλέπουμε ότι με μεγαλύτερα πλάτη αποκτούμε γρηγορότερα κατανόηση των παραμέτρων του συστήματος. Διαισθητικά, ασυμπτωτικά για πλάτος 0 δεν έχουμε διέγερση, άρα δεν μπορούμε να μάθουμε το σύστημα.



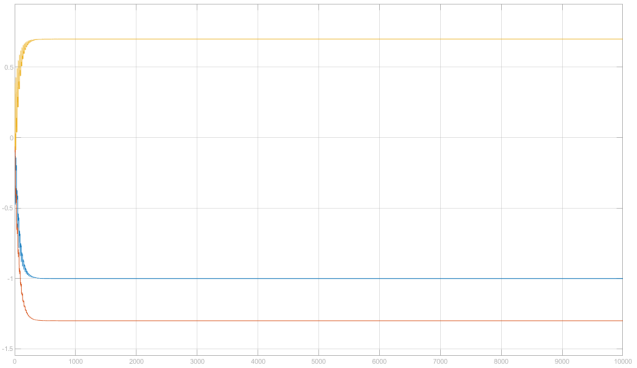
(α′) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse amplitude 0.05



(β′) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse amplitude 0.1



(γ′) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse amplitude 0.5



(δ′) Estimation with $\alpha_1 = 1, \alpha_2 = 1.3, b = 0.7, \lambda = 10, \gamma = 5$ and pulse amplitude 1

Figure 23: Estimations for different pulse amplitudes