Υλοποιήσεις Ψηφιακών Φίλτρων

Α. Εισαγωγή
Οποιοδήποτε γραμμικό χρονικά αμετάβλητο σύστημα διακριτού χρόνου χαρακτηρίζεται πλήρως από τη συνάρτηση μεταφοράς του η οποία έχει τη γενική μορφή:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-n_a}}$$

Βέβαια, τα πολυώνυμα αριθμητή και παρονομαστή μπορούν να παραγοντοποιηθούν ως προς τις ρίζες τους και έτσι η συνάρτηση μεταφοράς να γραφεί στη μορφή:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0(1 - \rho_1 z^{-1})(1 - \rho_2 z^{-1}) \cdots (1 - \rho_k z^{-1})}{(1 - r_0 z^{-1})(1 - r_1 z^{-1}) \cdots (1 - r_n z^{-1})}$$

Ακόμα, αν η συνάρτηση μεταφοράς αντιστοιχεί σε αιτιατό σύστημα είναι εφικτή και η ακόλουθη διάσπαση σε άθροισμα απλούστερων κλασμάτων:

$$H(z) = \frac{A_0}{1 - r_0 z^{-1}} + \frac{A_1}{1 - r_1 z^{-1}} + \cdots + \frac{A_n}{1 - r_n z^{-1}}$$

Οι παραπάνω μορφές αποτελούν μερικά παραδείγματα από τα οποία μπορούμε να οδηγηθούμε σε διαφορετικές υλοποιήσεις του συστήματος στο οποίο αντιστοιχούν.

Οι διάφορες υλοποιήσεις που προκύπτουν κάθε φορά διαφέρουν ως προς ένα σύνολο χαρακτηριστικών τα οποία συμπεριλαμβάνουν:
1. Την υπολογιστική πολυπλοκότητα (πλήθος προσθέσεων και πολλαπλασιασμών)
2. Το πλήθος των στοιχείων καθυστέρησης (μνήμη) που χρησιμοποιούν
3. Το βαθμό παραλληλοποίησης ο οποίος επιτυγχάνεται
4. Το βαθμό pipelining που επιτυγχάνεται
5. Την ευαισθησία της υλοποίησης ως προς σφάλματα κβαντισμού.

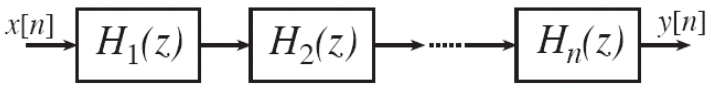
Β. Υλοποίηση από σε σειρά συνδεσμολογία απλούστερων συστημάτων
Στην περίπτωση όπου η συνάρτηση μεταφοράς μπορεί να γραφεί στη μορφή:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0(1 - \rho_1 z^{-1})(1 - \rho_2 z^{-1}) \cdots (1 - \rho_k z^{-1})}{(1 - r_0 z^{-1})(1 - r_1 z^{-1}) \cdots (1 - r_n z^{-1})}$$

τότε εύκολα μπορούμε να κάνουμε τη διάσπαση:

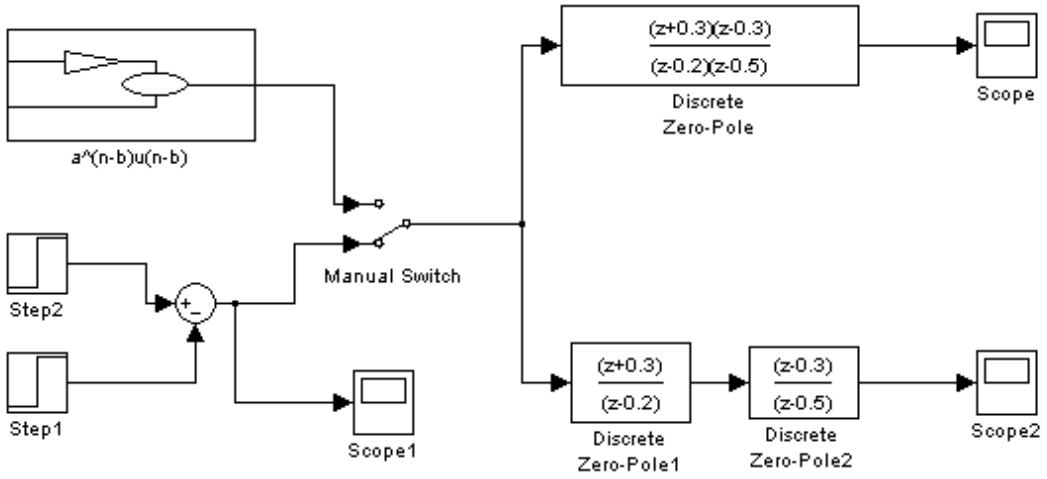
$$H(z) = H_1(z) \cdot H_2(z) \cdots H_n(z)$$

και έτσι να υλοποιήσουμε το σύστημά μας όπως παρουσιάζεται στο παρακάτω σχήμα.



Εικόνα 10: Σε σειρά υλοποίηση συστήματος

Για παράδειγμα, μπορείτε να φορτώσετε το αρχείο series.mdl με χρήση του περιβάλλοντος Simulink. Τότε θα εμφανιστεί το διάγραμμα του ακόλουθου σχήματος:



Εικόνα 11: Το μοντέλο του αρχείου series.mdl

Από το παραπάνω μοντέλο μπορείτε να επαληθεύσετε πως οι δυο κλάδοι υλοποιούν το ίδιο ακριβώς γραμμικό σύστημα. Τρέχοντας το μοντέλο με το διακόπτη στην κάτω θέση επιβεβαιώνεται πως τα συστήματα έχουν την ίδια κρουστική απόκριση ενώ με το διακόπτη στην πάνω θέση επιβεβαιώνεται πως τα συστήματα δίνουν την ίδια έξοδο.

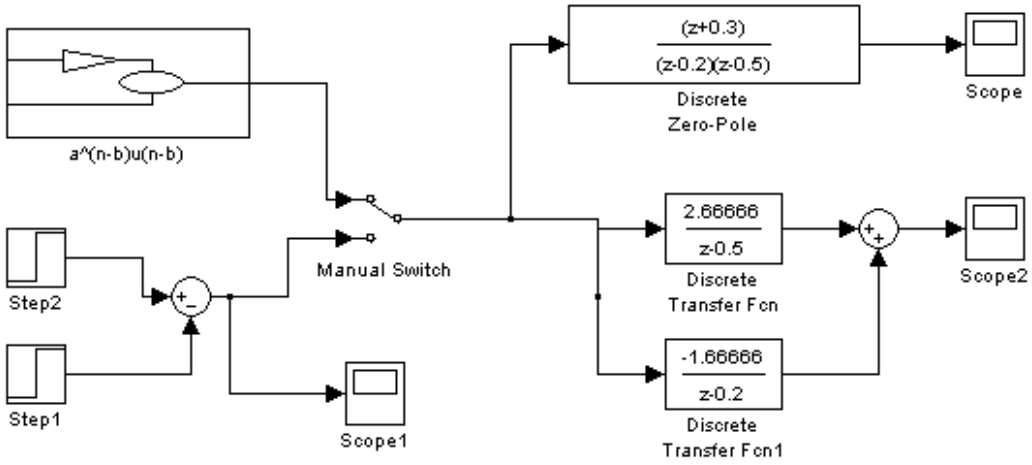
- Σχεδιάστε ένα κατωπερατό IIR φίλτρο Butterworth τάξης 5 μέσα από το περιβάλλον του Matlab με χρήση της εντολής **[B,A]=butter(5,0.5).** Στη συνέχεια να υπολογίσετε τις ρίζες του πολυωνύμου του αριθμητή και του παρονομαστή με χρήση της συνάρτησης **roots( ).**
- Ανοίξτε το περιβάλλον του Simulink και υλοποιήστε το κατωπερατό Butterworth φίλτρο με χρήση ενός block «Discrete Zero-Pole» (δίνοντας τις ρίζες που επέστρεψε η συνάρτηση **roots**) και στη συνέχεια με χρήση τριών συστημάτων «Discrete Zero-Pole» στη σειρά τοποθετημένων. Επιβεβαιώστε την ίδια λειτουργία των δυο υλοποιήσεων. Πως ομαδοποιήσατε τους πόλους και τα μηδενικά του φίλτρου;

Γ. Υλοποίηση από παράλληλη σύνδεση απλούστερων συστημάτων

Στην περίπτωση όπου μπορούμε να εκφράσουμε τη συνάρτηση μεταφοράς ενός συστήματος στη μορφή:

$$H(z) = \frac{A_0}{1 - r_0 z^{-1}} + \frac{A_1}{1 - r_1 z^{-1}} + \cdots + \frac{A_n}{1 - r_n z^{-1}}$$

τότε μπορούμε να υλοποιήσουμε το σύστημά μας συνδέοντας παράλληλα τα **n** υποσυστήματα και οδηγώντας τις εξόδους τους σε έναν αθροιστή. Για παράδειγμα, φορτώστε το αρχείο parallel.mdl



Εικόνα 12: Το μοντέλο του αρχείου parallel.mdl

Από το παραπάνω μοντέλο μπορείτε να επαληθεύσετε πως οι δυο κλάδοι υλοποιούν το ίδιο ακριβώς γραμμικό σύστημα. Τρέχοντας το μοντέλο με το διακόπτη στην κάτω θέση επιβεβαιώνεται πως τα συστήματα έχουν την ίδια κρουστική απόκριση ενώ με το διακόπτη στην πάνω θέση επιβεβαιώνεται πως τα συστήματα δίνουν την ίδια έξοδο.

Το περιβάλλον Matlab παρέχει μια συνάρτηση με την οποία μπορούμε να υπολογίσουμε την ανάλυση σε απλά κλάσματα μιας συνάρτησης μεταφοράς. Η συνάρτηση αυτή είναι η **residuez** και μπορεί να χρησιμοποιηθεί όπως παρουσιάζεται στη συνέχεια:

```
>> [R,P,K]=residuez(poly([-0.3]),poly([0.2 0.5]))

R =

   2.666666666666667e+000
  -1.666666666666668e+000


P =

   4.999999999999999e-001
   2.000000000000000e-001


K =

    [ ]

>>
```

Παραπάνω φαίνεται ο τρόπος με τον οποίο υπολογίσαμε την ανάλυση σε απλά κλάσματα για μια συνάρτηση η οποία έχει το μηδενικό -0.3 και τους πόλους 0.2 και 0.5. Η συνάρτηση poly επιστρέφει τους συντελεστές ενός πολυωνύμου από τις ρίζες του και η συνάρτηση residuez επιστρέφει τους συντελεστές των αριθμητών και των παρονομαστών των απλών κλασμάτων.

- Να υπολογίσετε την ανάλυση σε απλά κλάσματα μιας συνάρτησης μεταφοράς η οποία έχει το μηδενικό 0.3 και τους πόλους 0.2 και 0.5. Να υλοποιήσετε το σύστημα στο οποίο αντιστοιχεί:
    1. Με τη χρήση ενός block «Discrete Transfer Fcn». Για να υπολογίσετε τους συντελεστές των πολυωνύμων της συνάρτησης μεταφοράς μπορείτε να χρησιμοποιήσετε τη συνάρτηση poly.
    2. Με τη χρήση ενός block «Discrete Zero-Pole», δίνοντας τα μηδενικά και τους πόλους της συνάρτησης μεταφοράς.
    3. Με τη χρήση 2 block «Discrete Transfer Fcn» τα οποία θα συνδέσετε παράλληλα.
    4. Επιβεβαιώστε πως όλα τα παραπάνω συστήματα έχουν την ίδια κρουστική απόκριση.

- Να πειραματιστείτε με την περίπτωση όπου η συνάρτηση μεταφοράς έχει μιγαδικούς συζυγείς πόλους. Τι μορφή έχουν τότε τα «απλούστερα» κλάσματα;
- Τι συμβαίνει όταν ο βαθμός του πολυωνύμου του αριθμητή γίνει ίσος ή μεγαλύτερος από το βαθμό του πολυωνύμου του παρονομαστή;

Δ. Απευθείας Υλοποίηση (Direct Form)

Ως γνωστόν, η συνάρτηση μεταφοράς ενός γραμμικού συστήματος ταυτίζεται με το λόγο του μετασχηματισμού Z μιας οποιασδήποτε εξόδου του προς το μετασχηματισμό Z της αντίστοιχης εισόδου του:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-n_a}} = \frac{Y(z)}{X(z)}$$

Έτσι έχουμε:

$$Y(z) \cdot A(z) = X(z) \cdot B(z)$$

δηλαδή:

$$Y(z)\cdot(1+a_1 z^{-1}+a_2 z^{-2}+\cdots+a_{n_a} z^{-n_a}) = X(z)\cdot(b_0+b_1 z^{-1}+b_2 z^{-2}+\cdots+b_{n_b} z^{-n_b})$$

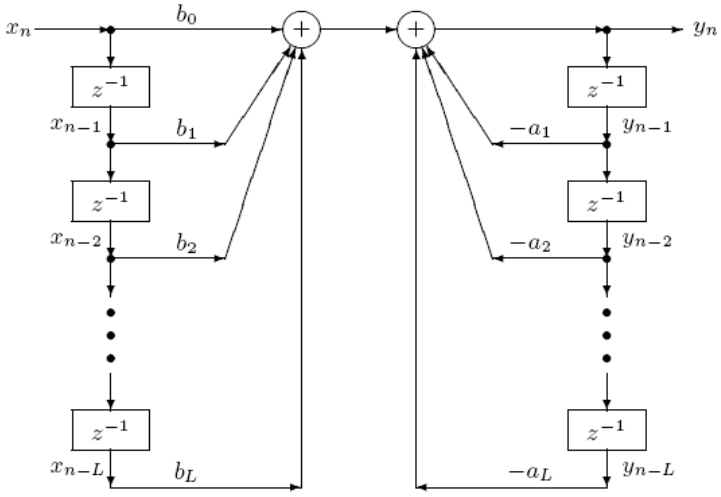
Λαμβάνοντας τώρα αντίστροφο μετασχηματισμό Z και στα δυο μέλη της παραπάνω σχέσης θα έχουμε:

$$y[n] + \sum_{i=1}^{n_a} a_i y[n-i] = \sum_{i=0}^{n_b} b_i x[n-i]$$

και τελικά:

$$y[n] = \sum_{i=0}^{n_b} b_i x[n-i] - \sum_{i=1}^{n_a} a_i y[n-i]$$

Η τελευταία σχέση οδηγεί στην υλοποίηση του επομένου σχήματος την οποία ονομάζουμε «απευθείας» υλοποίηση,

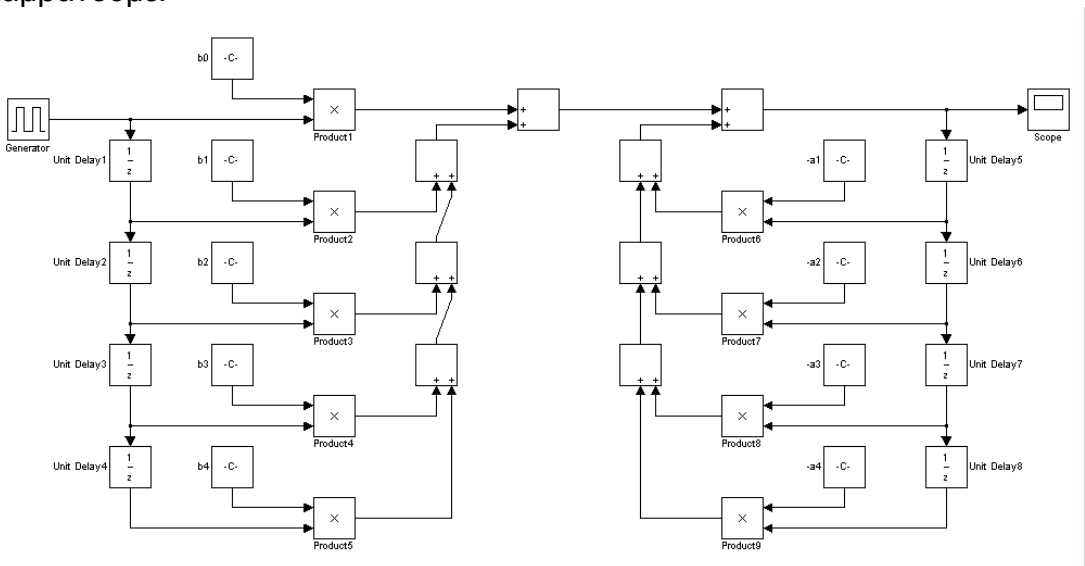


Εικόνα 13: Απευθείας υλοποίηση συστήματος

Το αρχείο makemod.m που έχετε κατεβάσει αποτελεί μια συνάρτηση η οποία δέχεται ως είσοδο ένα όνομα αρχείου, τους συντελεστές του πολυωνύμου A και τους συντελεστές του πολυωνύμου B και παράγει ως έξοδο ένα μοντέλο Simulink το οποίο υλοποιεί την αντίστοιχη συνάρτηση μεταφοράς με χρήση της απευθείας υλοποίησης. Για παράδειγμα, με τις ακόλουθες εντολές:

```
>> [B,A]=butter(4,0.5);
>> makemod('test1.mdl',B,A);
>> test1
```

Λαμβάνουμε:



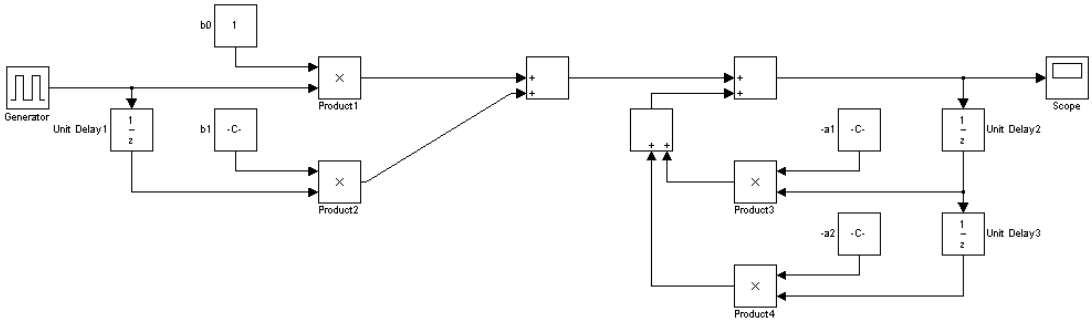
Εικόνα 14: Απευθείας υλοποίηση ενός κατωπερατού ψηφιακού φίλτρου Butterworth τάξης 4

Το παραπάνω μοντέλο Simulink το οποίο δημιουργήθηκε αυτόματα μπορούμε να το εξομοιώσουμε άμεσα και να δούμε στην έξοδό του την κρουστική του απόκριση.

Παρόμοια, με τις εντολές:

```
>> makemod('test2.mdl',poly([-0.3]),poly([0.2 0.5]));
>> test2
```

Λαμβάνουμε:



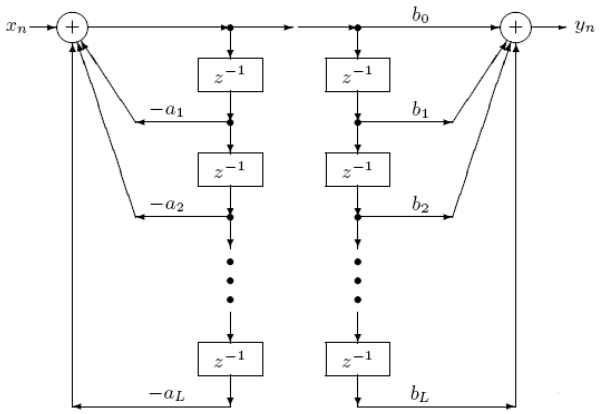
Εικόνα 15: Απευθείας υλοποίηση ενός συστήματος με συγκεκριμένα μηδενικά και πόλους

Το οποίο μπορεί άμεσα να εξομοιωθεί κα να δούμε την κρουστική του απόκριση.

- Με χρήση της συνάρτησης makemod δημιουργείστε μοντέλα για διάφορες συναρτήσεις μεταφοράς και εξομοιώνοντας τα δείτε τις αντίστοιχες κρουστικές αποκρίσεις.
- Υλοποιήστε κάθε συνάρτηση μεταφοράς με ένα block «Discrete Transfer Fcn». Συγκρίνετε τις διαφορετικές υλοποιήσεις ως προς την έξοδό τους.
- Τι γίνεται στην περίπτωση όπου σχεδιάσουμε ένα ασταθές σύστημα; Σχεδιάστε ένα ασταθές σύστημα και οδηγήστε την είσοδό του με ένα φραγμένο σήμα. Προσπαθήστε με τη βοήθεια ενός Scope να βρείτε σε ποια σημεία του κυκλώματος έχουμε φραγμένο σήμα και σε ποια σημεία το σήμα δεν φράσσεται.

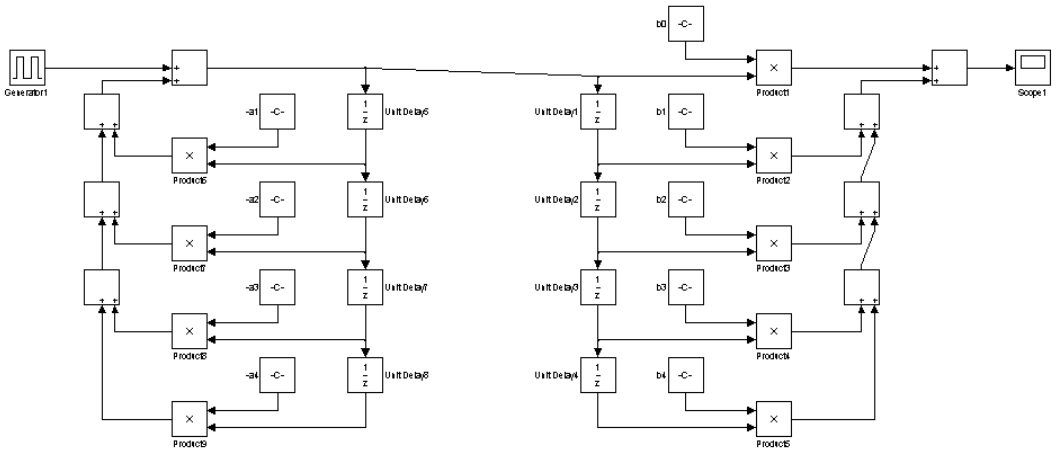
Ε. Απευθείας Υλοποίηση, πρώτα ο παρονομαστής (Direct Form II)

Αν στην υλοποίηση Direct Form αλλάξουμε τη σειρά με την οποία υπολογίζουμε τον αριθμητή και τον παρονομαστή, τότε προκύπτει η υλοποίηση του ακόλουθου σχήματος:



Εικόνα 16: Απευθείας υλοποίηση, πρώτα ο παρονομαστής

Την ανωτέρω υλοποίηση την ονομάζουμε υλοποίηση Direct Form II. Χρησιμοποιώντας το αρχείο makemod.m μπορούμε πολύ εύκολα να δημιουργήσουμε και υλοποιήσεις τύπου Direct Form II. Η μόνη τροποποίηση που πρέπει να κάνουμε στα μοντέλα εξόδου είναι η αλλαγή στη σειρά υπολογισμού αριθμητή και παρονομαστή. Για παράδειγμα από το μοντέλο test1.mdl το οποίο δημιουργήσαμε νωρίτερα (φίλτρο Butterworth) μπορούμε εύκολα να δημιουργήσουμε το ακόλουθο μοντέλο:



Εικόνα 17: Direct Form II υλοποίηση ενός κατωπερατού ψηφιακού Butterworth φίλτρου

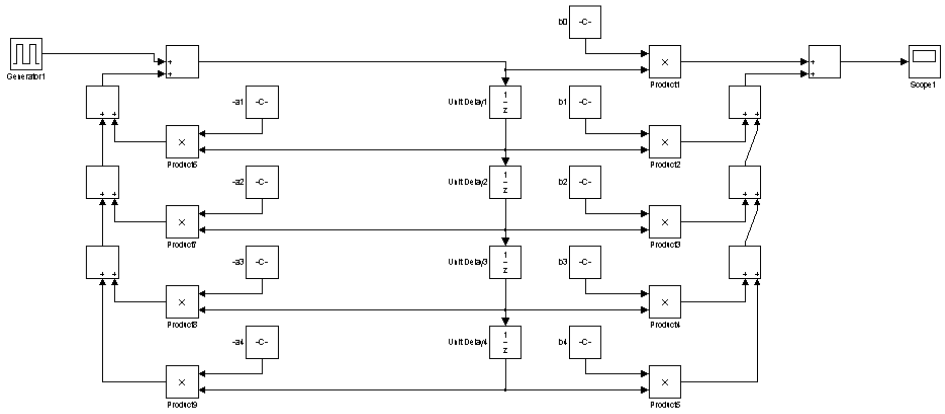
Εξομοιώνοντας το παραπάνω μοντέλο μπορούμε να επιβεβαιώσουμε πως η κρουστική του απόκριση είναι ίδια με εκείνη του μοντέλου από το αρχείο test1.mdl το οποίο δημιουργήσαμε προηγούμενα.

- Να δημιουργήσετε την υλοποίηση Direct Form II του συστήματος του μοντέλου test2.mdl και να επιβεβαιώσετε την ίδια συμπεριφορά τους.

ΣΤ. Υλοποίηση με το ελάχιστο πλήθος στοιχείων μνήμης

Από τα τελευταία 2 σχήματα που παρουσιάσαμε παραπάνω γίνεται αντιληπτό πως τα στοιχεία μνήμης που χρησιμοποιούμε μπορούν να ενωθούν, αφού και

οι δύο σειρές στοιχείων μνήμης καθυστερούν το ίδιο σήμα. Με βάση την παρατήρηση αυτή, οδηγούμαστε εύκολα στην επόμενη υλοποίηση του μοντέλου μας:



Εικόνα 18: Υλοποίηση με το ελάχιστο πλήθος στοιχείων μνήμης

- Να υλοποιήσετε τη συνάρτηση μεταφοράς με μηδενικά 0.2 0.3 0.4 και πόλους 0.1, 0.6, 0.8 και 0.5 με τον ελάχιστο αριθμό στοιχείων μνήμης.
- Να επιβεβαιώσετε τη σωστή λειτουργία του μοντέλου σας συγκρίνοντας τα αποτελέσματά σας με αυτά που δίνει ένα block «Discrete Zero-Pole».

Z. Μετασχηματισμός Αναστροφής

Τα σχήματα τα οποία παρουσιάσαμε στα προηγούμενα, μπορούν να θεωρηθούν ως ειδικού τύπου γραφήματα. Τα γραφήματα αυτά έχουν σαν κόμβους τα στοιχεία:

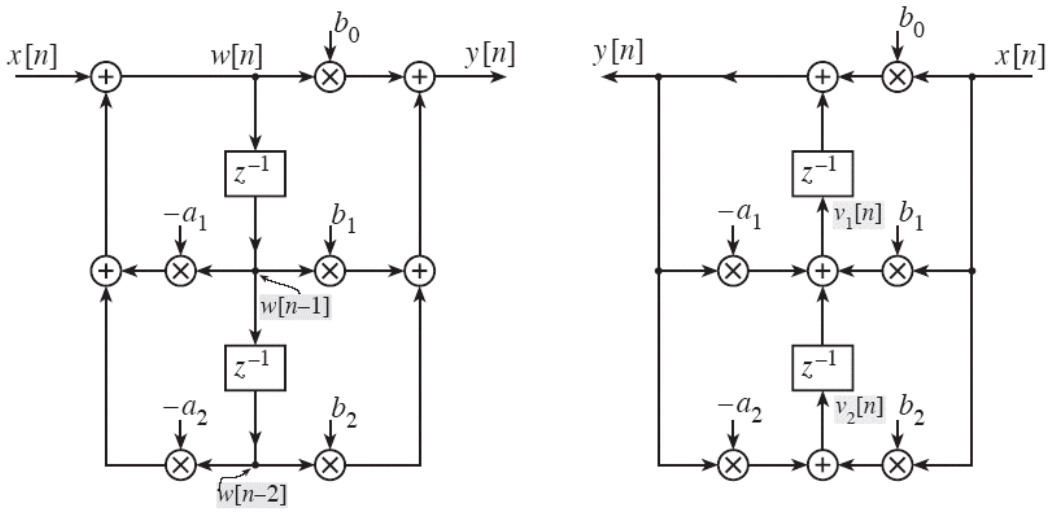
1. Στοιχεία μνήμης
2. Αθροιστές και πολλαπλασιαστές
3. Σταθερές
4. Κόμβους διαμοιρασμού σημάτων

Οι ακμές του γραφήματος αποτελούν τις γραμμές σύνδεσης ανάμεσα στους κόμβους του γραφήματος. Ένα τέτοιου είδους γράφημα ονομάζεται «Γράφημα Ροής Σήματος» (Signal Flow Graph).

Σε κάθε γράφημα ροής σήματος το οποίο έχει μια είσοδο και μια έξοδο (SISO, Single Input Single Output) μπορούμε να εκτελέσουμε τον ακόλουθο μετασχηματισμό και να οδηγηθούμε σε ένα νέο γράφημα το οποίο έχει την ίδια ακριβώς λειτουργία με το αρχικό γράφημα:

1. Αντιστρέφουμε όλες τις ακμές του γραφήματος (εκτός φυσικά από αυτές οι οποίες εξέρχονται από σταθερές)
2. Μετασχηματίζουμε όλους τους κόμβους άθροισης σε κόμβους διαμοιρασμού σήματος
3. Μετασχηματίζουμε όλους τους κόμβους διαμοιρασμού σήματος σε αθροιστές
4. Εναλλάσσουμε την είσοδο με την έξοδο

Στο επόμενο σχήμα δίνουμε ένα παράδειγμα εφαρμογής του μετασχηματισμού αναστροφής:



Εικόνα 19: Μετασχηματισμός αναστροφής

Η. Συνδυασμοί Υλοποιήσεων

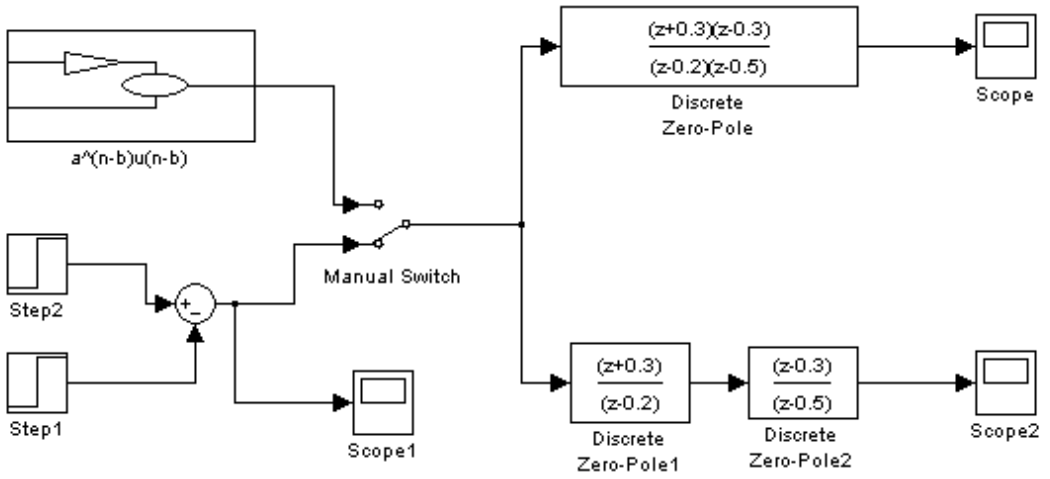
Όπως είδαμε στα προηγούμενα, μια συνάρτηση μεταφοράς μπορούμε γενικά να την απλοποιήσουμε ως άθροισμα ή γινόμενο «απλούστερων» συναρτήσεων μεταφοράς. Είδαμε τότε πως μπορούμε να την υλοποιήσουμε σε παράλληλη ή σε σειρά σύνδεση απλούστερων υποσυστημάτων. Τα υποσυστήματα αυτά, είναι με τη σειρά τους συναρτήσεις μεταφοράς οι οποίες μπορούν να υλοποιηθούν με κάποια από τις μορφές που είδαμε στα προηγούμενα:

1. Απευθείας υλοποίηση
2. Απευθείας υλοποίηση, πρώτα ο παρονομαστής
3. Υλοποίηση με το ελάχιστο πλήθος στοιχείων μνήμης

Για παράδειγμα η διάσπαση

$$H(z) = \frac{1 + 0.09z^{-2}}{1 - 0.7z^{-1} + 0.1z^{-2}} = \frac{(z - 0.3)(z + 0.3)}{(z - 0.2)(z - 0.5)} = \frac{z + 0.3}{z - 0.2} \cdot \frac{z - 0.3}{z - 0.5}$$

οδηγεί όπως είδαμε νωρίτερα στην υλοποίηση του ακόλουθου μοντέλου το οποίο βρίσκεται στο αρχείο series.mdl:

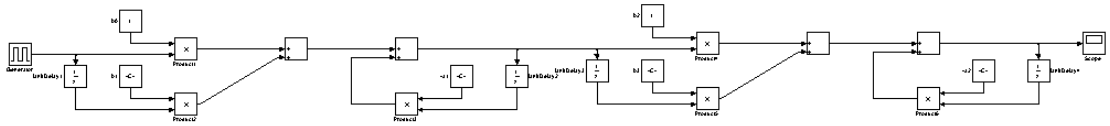


Εικόνα 20: Σε σειρά υλοποίηση συστήματος

Φυσικά, τα υποσυστήματά του μπορούμε να τα υλοποιήσουμε με βάση την υλοποίηση Direct Form. Για να δημιουργήσουμε αυτές τις υλοποιήσεις δίνουμε:

```
>> makemod('system1.mdl',poly(-0.3),poly(0.2));
>> makemod('system2.mdl',poly(0.3),poly(0.5));
>> system1
>> system2
```

Δημιουργούμε έτσι τις Direct Form υλοποιήσεις των υποσυστημάτων και τις αποθηκεύουμε ως μοντέλα στα αρχεία system1.mdl και system2.mdl. Από τα αρχεία αυτά τώρα, μπορούμε να επιλέξουμε όλα τα στοιχεία του δεύτερου υποσυστήματος και να τα αντιγράψουμε (copy/paste) στο αρχείο του πρώτου υποσυστήματος. Συνδέοντάς τα δυο υποσυστήματα σε σειρά, έχουμε υλοποιήσει το σύστημά μας ως σε σειρά συνδεσμολογία απλούστερων συστημάτων, όπου τα υποσυστήματα αυτά είναι υλοποιημένα σε μορφή Direct Form:



Εικόνα 21: Υλοποίηση σε σειρά με όλα τα υποσυστήματα να είναι υλοποιημένα σε μορφή Direct Form I

- Να υλοποιήσετε τη συνάρτηση μεταφοράς με μηδενικά 0.2 0.3 0.4 και πόλους 0.1, 0.6, 0.8 και 0.5 σε παράλληλη συνδεσμολογία. Για την ανάλυση σε απλά κλάσματα χρησιμοποιήστε τη συνάρτηση residuez και ως στοιχεία υλοποίησης χρησιμοποιήστε 4 blocks «Discrete Transfer Fcn».
- Δημιουργήστε τις Direct Form Υλοποιήσεις των τεσσάρων υποσυστημάτων με χρήση της συνάρτησης makemod
- Δημιουργήστε ένα νέο μοντέλο στο οποίο να συνδέσετε τα υποσυστήματα που σχεδιάσατε στο προηγούμενο βήμα σε παράλληλη συνδεσμολογία. Επιβεβαιώστε πως η τελευταία υλοποίηση δίνει τα ίδια αποτελέσματα με την προηγούμενη όπου χρησιμοποιήσατε τα blocks «Discrete Transfer Fcn».