Laboratorul 02.

Semnale şi sisteme de bază

La curs am discutat despre semnale de bază şi sisteme. În continuare vom face câteva exerciţii legate de aceste noţiuni.

Exerciţiul 1 [2p]

Pentru a transmite simboluri (ex: litere din alfabet) modemurile PC-urilor folosesc două frecvenţe (1600 Hz şi 1800 Hz) şi mai multe niveluri de amplitudine. O transmisiune se face într-o perioadă de timp T (interval de transmisie) şi este egală cu suma a două semnale de amplitudine diferită:

x(t) = A1 * sin (2*pi*f1*t) + A2 * sin (2*pi*f2*t)

1. Care este cel mai mic interval de transmisie care are sens să fie folosit cu frecvenţele de mai sus? Cu alte cuvinte, cât ar trebui să fie T astfel încât semnalul să aibă un număr întreg de cicluri? [1p]
2. Afişaţi cu ajutorul Octave semnalul produs de modem pe parcursul mai multor intervale de transmisie consecutive. Aşa cum am învăţat în laboratorul trecut adăugaţi titlu şi etichete plotului. [1p]

Exerciţiul 2 [2.5p]

La curs am văzut că putem descompune semnalele într-o sumă de mai multe semnale de bază (ramp, step etc.). Pentru acest exerciţiu veţi încerca să folosiţi semnalele 'step' şi 'ramp' pentru a crea semnalul reprezentat cu negru în acest slide: building_signals.pdf

Pentru a face asta în Octave va trebui să lucrăm cu semnale discrete, nu continue (vom discuta despre acest aspect în cursurile viitoare). În loc să lucrăm cu semnale reprezentate in intervalul [0,1] ca în slide vom folosi semnale ce se întind peste 100 de puncte.

Puteţi folosi următoarea funcţie pentru a crea un semnal 'ramp' peste N puncte:

ramp.m

```
function y=ramp(N)
%RAMP Returns a ramp signal of a given number of samples
%   [y] = RAMP(N)
%   returns the ramp signal for the given samples

y = zeros(1, N);
for t=1:N
    y(t) = t-1;
end
```

Task-ul vostru este să creaţi un semnal combinat, ca cel din slide, dar folosind secvenţe discrete, cu o formulă ca cea de mai jos:

s(i) = r(i) - r(i-T) - T*u(i-T)

unde i este un index de la 0 la N (în loc de un număr real de la 0 la 1), T este întârzierea, s este semnalul rezultat, r este semnalul 'ramp' (eventual întârziat cu T) şi u este semnalul 'unit step' (întârziat aici cu T).

- r(i-T) este doar o notaţie care marchează faptul că folosim un semnal 'ramp' întârziat cu T. Nu trebuie să apelaţi funcţia ramp cu argumentul (i-T) pentru că nu va funcţiona. Va trebui să întârziaţi semnalul 'de mână' ca mai jos.

  - Folosim factorul T în faţa lui u(i-T) pentru că este amplitudinea la care r(i) a ajuns până în acel moment şi vrem să avem semnalul final la 0.

Pentru asta ar trebui să:

1. creaţi un semnal 'unit step' numit ustep.m, care practic întoarce o secvenţă de N valori de 1 [0.5p]
2. să setaţi numărul de puncte la N = 200 şi delay-ul la T = 100
3. să creaţi cele 3 semnale (care urmează să fie combinate) folosind 'ramp' şi 'ustep' cu N şi T de mai sus. [1p]
    - Puteţi crea secvenţa de input ca:

      ```
      x=1:N;
      ```

    - Puteţi crea primul semnal ca:

      ```
      s1 = ramp(N)
      ```

    - Puteţi întârzia un semnal cu T în felul următor:

      ```
      [zeros(1,T), s(1:N-T)];
      ```

4. combinaţi cele 3 semnale
5. afişaţi toate cele 4 semnale (cele 3 individuale şi combinaţia lor) [1p]
    - folosiţi culori diferite (eventual grosimi de linie diferite) pentru fiecare semnal şi afişaţi legenda pentru a diferenţia semnalele.

Folosiţi 'help plot' sau 'doc plot' pentru a vedea cum se plotează un semnal. De exemplu pentru a plota s1 cu o linie verde de grosime 2 puteţi folosi codul următor:

```
plot(x, s1, 'g-', 'LineWidth', 2);
```

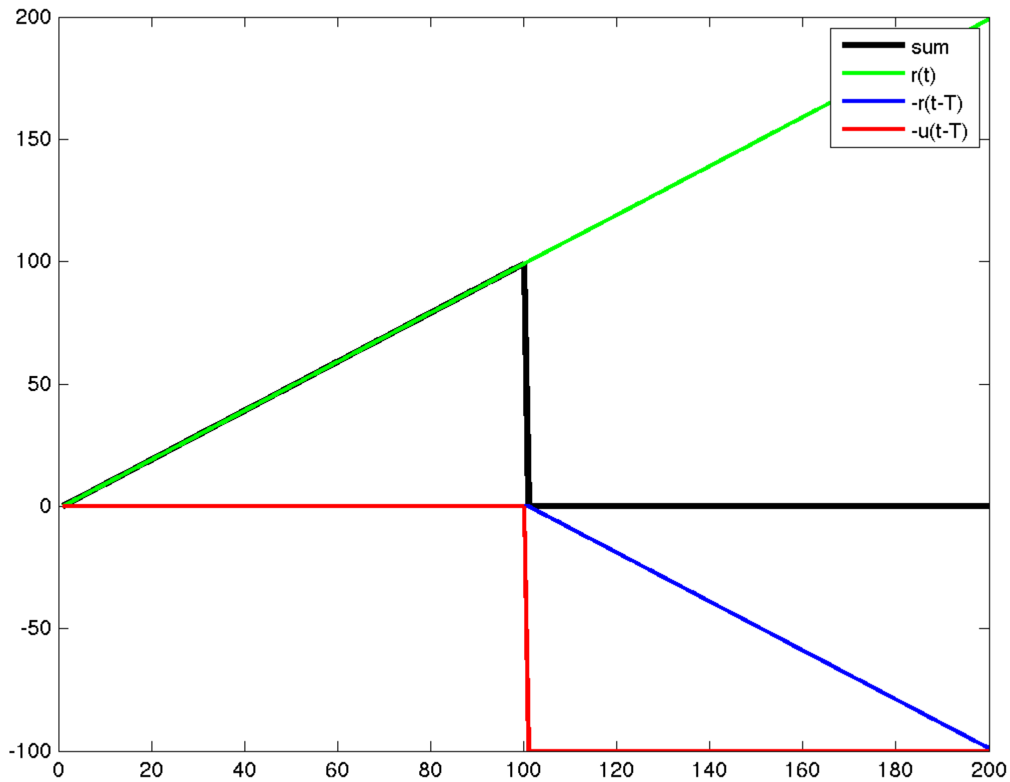
Pentru a afişa mai multe semnale în aceeaşi figură cu ajutorul comenzii plot puteţi folosi

```
hold on;
```

după primul plot:

```
figure;
plot(x, s1, 'g-', 'LineWidth', 2);
hold on;
plot(...)
```

Ar trebui să obtineţi ceva similar imaginii de mai jos:



Exerciţiul 3 [2p]

La curs am văzut că datorită egalității lui Euler putem scrie o exponenţială complexă ca o sumă de sin şi cos:

$$e^{j \cdot t} = \cos(t) + j \cdot \sin(t)$$

De asemenea:

$$e^{-j \cdot t} = \cos(t) - j \cdot \sin(t)$$

Adunând aceste 2 ecuaţii şi împărţind la 2 obţinem:

$$\cos(t) = \frac{e^{j \cdot t} + e^{-j \cdot t}}{2}$$

Încercaţi să arătaţi asta în Octave, făcând următoarele:

- Folosiţi secvenţa de input

```
t = [0, pi/6, pi/4, pi/3, pi/2];
```

- Afişaţi exponenţiala complexă $s_1 = e^{j \cdot t}$, e.g.

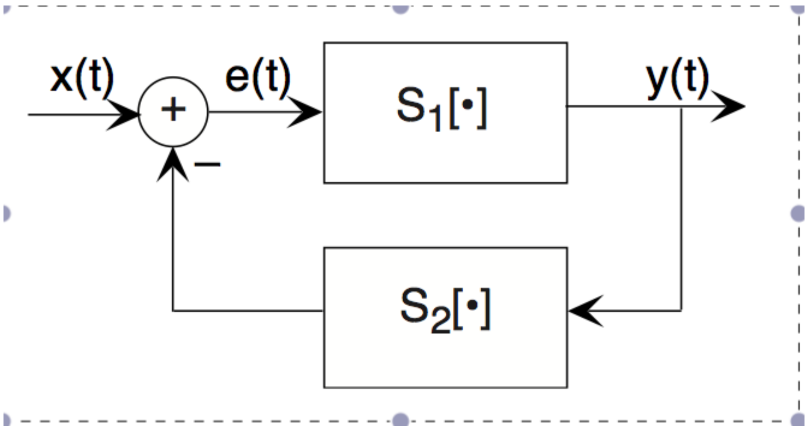
```
plot(exp(1i*t), 'ro');
```

- Afişaţi exponenţiala complexă $s_2 = e^{-j \cdot t}$ cu o altă culoare
- Calculaţi media celor 2 exponenţiale $s_s = \frac{s_1 + s_2}{2}$, i.e. cos(t)
- Afisaţi secvenţa rezultată în imaginar folosid real(ss) ca valori x şi imag(ss) ca valori y

- Verificaţi dacă s1, s2 şi ss arată cum v-aţi fi aşteptat!

Exerciţiul 4 [3.5p]

Avem un sistem de feedback precum cel din imaginea următoare:



Să presupunem că folosim acest sistem pentru sistemul de pilot automat al maşinii, unde x(t) este o constantă ce reprezintă viteza dorită, iar y(t) este viteza maşinii măsurată de vitezometru. În această aplicaţie, sistemul 2 este sistemul identitate (intrare = ieşire).

Să construim acest sistem având în vedere următoarele constrângeri:

- viteza initială a maşinii este 7
- valoarea iniţială a secvenţei de feedback (output-ul lui S2), f, este 0
- valoarea iniţială pentru secvenţa de diferenţă, e, este 0
- valoarea de input a sistemului de pilot automat, x, este o secvenţă de tipul [60, 60, …, 60]
- primul sistem, S1, primeste 2 input-uri: viteza curentă şi diferenţa e(i). Bazându-se pe acestea, actualizează viteza curentă după cum urmează:
  - Dacă e(i) > 10, atunci y(i+1) = y(i) + 10
  - altfel dacă e(i) > 0, atunci y(i+1) = y(i) + 1
  - altfel dacă e(i) == 0, atunci y(i+1) = y(i)

Construiţi sistemul S1 ca o funcţie Octave cu 2 parametrii (viteza_curentă, e), care afişează următoarea viteză curentă ca mai sus. În Octave puteţi folosi o instrucţiune for pentru asta:

```
N = 20;
y = zeros(1,N);
y(1) = 7;
for i=1:N-1
  ...
  y(i+1) = S1(y(i), e(i));
end
```

Rulaţi sistemul de N = 20 ori şi afişaţi outputul sistemului.

ps/labs/02.txt · Last modified: 2020/10/28 17:39 by ionut.gorgos