Matlab κι εφαρμογές στην Γραμμική Άλγεβρα

Έστω V διανυσματικός χώρος. Ένας **γραμμικός συνδυασμός** μιας λίστας διανυσμάτων $(u_1, \ldots, u_m)$ στον V είναι ένα διάνυσμα της μορφής

$$a_1 u_1 + \cdots + a_m u_m ,$$

Το σύνολο όλων των γραμμικών συνδυασμών των $(u_1, \ldots, u_m)$ λέγεται παραγόμενος χώρος των $(u_1, \ldots, u_m)$ και δηλώνεται με
span$(u_1, \ldots, u_m)$.

Για παράδειγμα, το διάνυσμα (7, 2, 9) είναι ένας γραμμικός συνδυασμός των (2, 1, 3), (1, 0, 1) γιατί (7, 2, 9) = 2(2, 1, 3) + 3(1, 0, 1).

Πως μπορούμε να υλοποιήσουμε στο Matlab έναν αλγόριθμο που να μας δίνει αν ένα δοσμένο διάνυσμα v ανήκει στο span μιας λίστας δοσμένης διανυσμάτων $(u_1, \ldots, u_m)$?

Προφανώς το πρόβλημα ανάγεται στην ύπαρξη της λύσης του παρακάτω γραμμικού συστήματος εξισώσεων:

$$v = U a$$

όπου v το διάνυσμα v σε στήλη, ο πίνακας U έχει στην m στήλη τις συνιστώσες του διανύσματος u_m.

Η συνάρτηση span

```
function span(v, varargin)
% Βρίσκει αν ένα διάνυσμα v είναι στο span μιας λίστας διανυσμάτων.
A = [ ];
n = length(varargin);
for i=1:n
    u = varargin{i};
    u = u';
    A = [A u(:)];
end
v = v';
v = v(:);
if rank(A) == rank([A v])
disp(' Given vector is in the span.')
else
disp(' Given vector is not in the span.')
end
```

Για παράδειγμα:

```
>> v=[7 2 9];u1=[2 1 3];u2=[1 0 1];
>> span(v,u1,u2)
 Given vector is in the span.
```

- Άλλο παράδειγμα

```
>> v=[7 2 9];u1=[1 0 0];u2=[0 1 0];
>> span(v,u1,u2)
 Given vector is not in the span.
```

- Ένα παράδειγμα με πίνακες

```
>> v=eye(3);
>> u1=[0 0 0;0 1 0;0 0 1];u2=[1 0 0;0 0 0;0 0 1];u3=[1 0 0;0 1 0;0 0 0];
>> span(v,u1,u2,u3)
 Given vector is in the span
```

Η διαδικασία ορθοκανονικοποίησης Gram-Schmidt

Θεωρούμε έναν διανυσματικό χώρο V με το συνηθισμένο Ευκλείδιο εσωτερικό. Μια λίστα διανυσμάτων (e_1 , . . . , e_m) λέγεται ορθοκανονική αν

- ανα δύο όλα τα διανύσματα είναι ορθογώνια και
- κάθε διάνυσμα έχει μέτρο 1

δηλαδή στο Matlab

dot(e_i,e_j)=0 αν i≠j

dot(e_i,e_j)=1 αν i=j

Κάθε ορθοκανονική λίστα διανυσμάτων είναι γραμμικώς ανεξάρτητη.

- >>e1=[1/2 1/2 1/2 1/2];e2=[1/2 1/2 -1/2 -1/2];e3=[1/2 -1/2 -1/2 1/2]; e4=[-1/2 1/2 -1/2 1/2];

>>dot(e1,e1)

ans

1

Το κύριο θεώρημα για ορθοκανονικές βάσεις

Έστω $(e_1,...e_n)$ μια ορθοκανονική βάση του V. Τότε για το τυχαίο $v \in V$ ισχύει

$$v = \langle v , e_1 \rangle e_1 + \cdots + \langle v , e_n \rangle e_n$$

και

$$\|v\|^2 = |\langle v , e_1 \rangle|^2 + \cdots + |\langle v , e_n \rangle|^2$$

Παράδειγμα: $e_1=(1,0)$, $e_2=(0,1)$. Τότε το τυχαίο διάνυσμα $v=(v_1,v_2)$

$$v = \langle v , e_1 \rangle e_1 + \langle v , e_2 \rangle e_2 = v_1 e_1 + v_2 e_2$$

και

$$\|v\|^2 = v_1^2 + v_2^2$$

Gram-Schmidt: Αν $(v_1,...v_n)$ είναι μια λίστα γραμμικώς ανεξάρτητων διανυσμάτων στο V, τότε υπάρχει μια ορθοκανονική βάση $(e_1,...e_n)$ στο V τέτοια ώστε

$\mathrm{span}(v_1,...v_n) = \mathrm{span}(e_1,...e_n)$.

Η υλοποίηση της διαδικασίας Gram-Schmidt στο Matlab

```
function V = gs(A)
% Gram-Schmidt ορθοκανονικοποίηση διανυσμάτων που δίνονται
% στις στήλες ενός πίνακα  A. Τα ορθοκανονικοποιημένα διανύσματα
% δίνονται στις στήλες ενός πίνακα V.
[m,n] = size(A);
for k=1:n
V(:,k) = A(:,k);
for j=1:k-1
R(j,k) = V(:,j)'*A(:,k);
V(:,k) = V(:,k) - R(j,k)*V(:,j);
end
R(k,k) = norm(V(:,k));
V(:,k) = V(:,k)/R(k,k);
end
```

```
>> A=[1 0;1 2];
>> E=gs(A)
E =
   0.707106781186547  -0.707106781186547
   0.707106781186547   0.707106781186548
>> E'*E
ans =
   1.000000000000000   0.000000000000000
   0.000000000000000   1.000000000000000
```

Ορθογώνιες προβολές και προβλήματα ελαχιστοποίησης

Έστω U ένα υποσύνολο ενός διανυσματικού χώρου V, τότε το ορθογώνιο συμπλήρωμα του U, συμβολίζεται με $U^{\perp}$ είναι το σύνολο όλων των διανυσμάτων του V τα οποία είναι ορθογώνια σε κάθε διάνυσμα του U.

$$U^{\perp} = \{ v \in V : < v , u > = 0 \text{ για κάθε } u \in U \}$$

Κάθε διάνυσμα του V μπορεί να γραφεί με μοναδικό τρόπο ως το ευθύ άθροισμα

$$v = u + w$$

όπου $u \in U$ και $w \in U^{\perp}$. Η ορθογώνια προβολή του V στον U είναι ακριβώς αυτή η αντιστοίχηση $P^{u}(v)=u$.

Αν $(e_1 , . . . , e_m)$ είναι μια ορθοκανονική βάση του U, τότε

$$P^{u}(v) = < v , e_1 > e_1 + \cdot \cdot \cdot + < v , e_m > e_m$$

Πρόταση: Έστω U υπόχωρος του V και $v \in V$. Τότε για κάθε $u \in U$

$$\| v - P^{u}(v) \| \leq \| v - u \|$$

Προβλήματα ελαχιστοποίησης

Θεωρούμε το πρόβλημα εύρεσης πολυωνύμου u με πραγματικούς συντελεστές και βαθμού το πολύ 5 το οποίο στο διάστημα [-π, π] προσεγγίζει την συνάρτηση sin(x) όσο το δυνατόν καλύτερα, με την έννοια

$$\int |\sin(x) - u(x)|^2 \, dx \text{ στο } [-\pi, \pi]$$

είναι όσο το δυνατόν μικρότερο.

- Για να λύσουμε το πρόβλημα αυτό θεωρούμε το διανυσματικό χώρο των συνεχών πραγματικών συναρτήσεων C[-π, π] στο [-π, π] με εσωτερικό γινόμενο

$$\langle f(x), g(x) \rangle = \int f(x)\, g(x)\, dx \quad \text{(η ολοκλήρωση από } x=-\pi \text{ μέχρι } x=\pi\text{).}$$

Έστω v=sin(x) και U ο υπόχωρος του C[-π, π] που αποτελείται από πολυώνυμα με πραγματικούς συντελεστές με βαθμό το πολύ 5.

- Το πρόβλημα διατυπώνεται ώς εξής:

Να βρεθεί $u \in U$ τέτοιο που $\|v-u\|$ να είναι όσο το δυνατόν μικρότερο.

Υλοποίηση στο Matlab

```
function V = gspoly(A,x1,x2)
% Ορθοκανονικοποίηση Gram-Schmidt βάσης πολυωνύμων που δίνεται
% στο διάνυσμα A. Η ορθοκανονικοποιημένη βάση
% δίνεται στο διάνυσμα V.
[m,n] = size(A);
for k=1:n
   V(k) = A(k);
for j=1:k-1
   R(j,k) = int(V(j)*A(k),x1,x2);
   V(k) = V(k) - R(j,k)*V(j);
end
   R(k,k) = (int(V(k)*V(k),x1,x2))^(1/2);
   V(k) = V(k)/R(k,k);
end
```

```
%Γραφικά αποτελέσματα της προσέγγισης της sin x από
%ορθοκανονική βάση πολυωνύμων τα οποία δίνονται από
%τη διαδικασία ορθοκανονικοποίησης Gram-Schmidt με
%την συνάρτηση gspoly Προσοχή!!! πρέπει να τρέξω
%                    >> E=gspoly(A)
syms x;
[m,n] = size(A);
PU=int(sin(x)*E(1),-pi,pi)*E(1);
for j=2:n
    PU=PU+int(sin(x)*E(j),-pi,pi)*E(j);
end
close all
 u=vpa(PU,6);
plot1=ezplot(u,[-pi,pi]);
hold on
set(plot1,'Color','red','LineWidth',3);
plot2=ezplot(sin(x),[-pi,pi]);
set(plot2,'Color','blue','LineWidth',1);
title 'The approximation of sin x by 5th order polynomials in [-pi,pi]'
hold off
```

Γραφικά αποτελέσματα της προσέγγισης

```
>> A=[1 x x^2 x^3];
>> E=gspoly(A,-pi,pi);
>> polysine
```

Μια πιο καλή προσέγγιση!!!

```
>> A=[1 x x^2 x^3 x^4 x^5];
>> E=gspoly(A,-pi,pi);
>> polysine
```

- **Άσκηση:** Με την συνάρτηση gspoly στο διάστημα [0,1] να κατασκευασθεί μια ορθοκανονική βάση με εφαρμογή της διαδικασίας Gram-Schmidt στην βάση (1, x, x^2).

  Αφού τροποποιήσετε κατάλληλα το script M-file polysine (ονομάστε το polyexp) να παρασταθούν γραφικά τα αποτελέσματα της προσέγγισης της συνάρτησης exp(x) στο διάστημα [0,1] με την ορθοκανονική βάση που βρήκατε.