

Επιστημονικός Υπολογισμός
Εργασία 3
Παράδοση 13 Μαΐου 2009
Ονοματεπώνυμο: Κάλλης Παύλος
ΑΕΜ: 792

Μέρος 2ο

Άσκηση 1: Υπολογίστε τους πρώτους επτά όρους της σειράς Taylor για την $f(x) = -\log(1 - x)$.

```
clear x;  
syms x;  
y=taylor('-log(1-x)',x,7)
```

Αποτελέσματα:

y =

$$x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \frac{1}{5}x^5 + \frac{1}{6}x^6$$

Υπολογίστε τους επτά πρώτους όρους της σειράς Taylor για την $g(x) = e^x/\cos(x) + \tan^3(x)$.

```
clear x;  
syms x;  
z=taylor('exp(x)/cos(x)+(tan(x))^3',x,7)
```

Αποτελέσματα:

z =

$$1 + x + x^2 + \frac{5}{3}x^3 + \frac{1}{2}x^4 + \frac{13}{10}x^5 + \frac{19}{90}x^6$$

Άσκηση 2: Υπολογίστε τους μιγαδικούς συντελεστές Fourier a_{-3}, \dots, a_3 για την συνάρτηση που ορίζεται από την $f(x) = x^2 - 4$ στο $[-\pi, \pi]$ και το οποίο είναι 2π περιοδική (δηλαδή $f(x) = f(x + 2\pi)$ για κάθε x).

```
clear x; syms x;  
int((x^2-4),x,-pi,pi)/(2*pi)
```

Αποτελέσματα:

ans =

$$\frac{1}{2} \cdot (2/3 \pi^3 - 8 \pi) / \pi$$

Άσκηση 3: Υπολογίστε τους συντελεστές b_0, b_1, b_3, b_5 για τη συνάρτηση $f(x)$ του προηγούμενου παραδείγματος. Στη συνέχεια χρησιμοποιήστε το Equation 1 για να υπολογίσετε τις $f(0), f(1), f(2), f(3), f(4), f(5)$ και $f(6)$.

```
clear all;clc;

%calculate b0
b(1)=(1/6)*(1*exp(-i*2*pi*0/6*1)+1*exp(-i*2*pi*0/6*3)+1*exp(-i*2*pi*0/6*5));

%calculate b1
b(2)=(1/6)*(1*exp(-i*2*pi*1/6*1)+1*exp(-i*2*pi*1/6*3)+1*exp(-i*2*pi*1/6*5));

%calculate b2
b(3)=(1/6)*(1*exp(-i*2*pi*2/6*1)+1*exp(-i*2*pi*2/6*3)+1*exp(-i*2*pi*2/6*5));

%calculate b3
b(4)=(1/6)*(1*exp(-i*2*pi*3/6*1)+1*exp(-i*2*pi*3/6*3)+1*exp(-i*2*pi*3/6*5));

%calculate b4
b(5)=(1/6)*(1*exp(-i*2*pi*4/6*1)+1*exp(-i*2*pi*4/6*3)+1*exp(-i*2*pi*4/6*5));

%calculate b5
b(6)=(1/6)*(1*exp(-i*2*pi*5/6*1)+1*exp(-i*2*pi*5/6*3)+1*exp(-i*2*pi*5/6*5));

for(i=0:5)
    fprintf('b(%d)=%g\n',i,b(i+1));
end

pause;

f=[
% f(0)
(1/6)*(b(1)*exp(i*2*pi*0/6*0)+b(2)*exp(i*2*pi*1/6*0)+b(3)*exp(i*2*pi*2/6*0+i*2*pi*3/6*0)+b(4)*exp(i*2*pi*4/6*0)+b(5)*exp(i*2*pi*5/6*0));
% f(1)
(1/6)*(b(1)*exp(i*2*pi*0/6*1)+b(2)*exp(i*2*pi*1/6*1)+b(3)*exp(i*2*pi*2/6*0+i*2*pi*3/6*1)+b(4)*exp(i*2*pi*4/6*1)+b(5)*exp(i*2*pi*5/6*1));
% f(2)
(1/6)*(b(1)*exp(i*2*pi*0/6*2)+b(2)*exp(i*2*pi*1/6*2)+b(3)*exp(i*2*pi*2/6*2+i*2*pi*3/6*2)+b(4)*exp(i*2*pi*4/6*2)+b(5)*exp(i*2*pi*5/6*2));
% f(3)
(1/6)*(b(1)*exp(i*2*pi*0/6*3)+b(2)*exp(i*2*pi*1/6*3)+b(3)*exp(i*2*pi*2/6*3+i*2*pi*3/6*3)+b(4)*exp(i*2*pi*4/6*3)+b(5)*exp(i*2*pi*5/6*3));
% f(4)
(1/6)*(b(1)*exp(i*2*pi*0/6*4)+b(2)*exp(i*2*pi*1/6*4)+b(3)*exp(i*2*pi*2/6*4+i*2*pi*3/6*0)+b(4)*exp(i*2*pi*4/6*4)+b(5)*exp(i*2*pi*5/6*4));
% f(5)
(1/6)*(b(1)*exp(i*2*pi*0/6*5)+b(2)*exp(i*2*pi*1/6*5)+b(3)*exp(i*2*pi*2/6*5+i*2*pi*3/6*5)+b(4)*exp(i*2*pi*4/6*5)+b(5)*exp(i*2*pi*5/6*5));
];
```

```

% f(6)=f(0)
f(6)=f(1);

for (i=0:5)
    fprintf('f(%d)=%g\n',i,f(i+1));
end

```

Αποτελέσματα:

```

b(0)=0.5
b(1)=-9.25186e-017
b(2)=-1.85037e-016
b(3)=-0.5
b(4)=3.88578e-016
b(5)=-1.75785e-016

```

```

f(0)=1.85037e-017
f(1)=-1.03911e+008
f(2)=-1.2957e+017
f(3)=-1.61564e+026
f(4)=-2.0146e+035
f(5)=1.85037e-017

```

Άσκηση 4:

```

>> Jblur2=J;
>> for i=1:843
    for j=1:843
        if (log(1+J(i,j))<.5*(log(1+J(1,1)))) Jblur2(i,j)=0;
        end
    end
end;
>> Iblur2=ifft2(Jblur2)/max(max(abs(ifft2(Jblur2))));
>> imshow(Iblur2)

```

Τρέξτε αυτό το κώδικα και δείτε πως επηρεάζει τη δεδομένη εικόνα. Τώρα πειραματιστείτε με το συντελεστή 0.5 που χρησιμοποιείται στην εντολή if. Σε ποια τιμή έχετε χάσει το περισσότερο μέρος από τις λεπτομέρειες της εικόνας; Τώρα τρέξτε τον παραπάνω κώδικα αλλά αλλάξτε την ανισότητα στην εντολή if από «less than» σε «greater than».

Πειραματιστείτε με το συντελεστή 0.5 και ειδικότερα φέρτε το συντελεστή αρκετά κοντά στο 1.

Τι βλέπετε;

Όσο μεγαλώνει ο συντελεστής 0.5 και πλησιάζει προς τη μονάδα, τόσο περισσότερο χάνονται οι λεπτομέρειες της εικόνας. Μόλις φτάσει κοντά στο 1 έχουμε χάσει τις περισσότερες λεπτομέρειες της εικόνας και πλέον δε μπορούμε να διακρίνουμε την εικόνα λόγω εφαρμογής του φίλτρου blur. Αντιθέτως

όσο πλησιάζει στο 0 τόσο μειώνεται το φίλτρο blur.

Αν αλλάξουμε το '<' σε '>' τότε παρατηρούμε πως επηρεάζεται η φωτεινότητα της εικόνας. Όσο ο συντελεστής αυξάνει και πλησιάζει στο 1 τα σκούρα pixels γίνονται φωτεινότερα και ανεβαίνει η φωτεινότητα της εικόνας. Μόλις φτάσει στο 1 η εικόνα έχει την αρχική της φωτεινότητα.

Μέρος 2ο

Άσκηση 1: Να βρεθούν οι 4^{ες} ρίζες του 1

Σχεδιάστε όλες τις 4^{ες} ρίζες της μονάδος. Υπόμνημα: χρησιμοποιείτε την συνάρτηση του MATLAB plot() με μιγαδικό όρισμα.

ΒΑΛΤΕ ΤΟΝ ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

```
clear all;clc;clf;
w=exp(i*2*pi/4);
for j=1:4
    z(j)=w^(j-1);
end
plot(z,'*r');
axis([-2 2 -2 2]);
```

Άσκηση 2: Ιδιότητες του πίνακα Fourier

Γράψτε ένα MATLAB script που να κατασκευάζει έναν πίνακα Fourier, **F**, για δεδομένο *N*. Δείξτε ότι ο **F** είναι:

- συμμετρικός,
- ορθογώνιος,
- δεν έχει ορθογώνιες στήλες,
- και έχει ένα πολύ εύκολο αντίστροφο.

Χρησιμοποιείτε το πρόγραμμα viewmatrix.m για να τυπώσετε τους πίνακες

ΒΑΛΤΕ ΤΟΝ ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

```
%Askisi 2
n=input('Doste to megethos tou pinaka: ');
step1=0;
for k=2:n
    step2=0;
    step1=k-1;
    for l=2:n
        x(k,l)=(step1+step2);
        step2=step2+step1;
    end
end

for k=1:n
    for l=1:n
        F(k,l)=i^x(k,l);
```

```

        end
    end

    disp('Pinakas F:');
    F
    disp('Pinakas F anastrofos:');
    F'
    disp('Antistrofos tou F');
    inv(F)
    viewmatrix(x);

```

Ο πίνακας F είναι συμμετρικός αφού ο F ισούται με τον ανάστροφό του.
 είναι ορθογώνιος γιατί ο ανάστροφος του ισούται με τον αντίστροφό του.
 δεν έχει ορθογώνιες στήλες γιατί κάθε στήλη του είναι γραμμικά ανεξάρτητη.
 και έχει έναν πολύ εύκολο αντίστροφο ο οποίος τυπώνεται μόλις τρέξουμε το πιο πάνω script.

σημείωση: Επειδή ο F περιέχει μιγαδικούς αριθμούς και επειδή η viewmatrix δεν υποστηρίζει εκτύπωση μιγαδικών εμφανίζουμε μόνο τους εκθέτες του I δηλαδή τον πίνακα x.

Άσκηση 3: Εφαρμογή του FT

Βρείτε τους μετασχηματισμούς Fourier για τις 3 παρακάτω συναρτήσεις:

Διακριτή συνάρτηση delta: $f = [1, 0, 0, 0]$;

Σταθερό διάνυσμα: $f = [1 \ 1 \ 1 \ 1]$;

Διακριτό ημίτονο: $f = [0 \ 1 \ 0 \ -1]$;

Ο ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

```

%askisi 3
f1=[1,0,0,0]';
f2=[1 1 1 1]';
f3=[0 1 0 -1]';

c1=inv(F)*f1
c2=inv(F)*f2
c3=inv(F)*f3

f1=[1,0,0,0]';
f2=[1 1 1 1]';
f3=[0 1 0 -1]';

c1=inv(F)*f1
c2=inv(F)*f2
c3=inv(F)*f3

```

Άσκηση 4: fft() και ifft()

Οι συναρτήσεις του MATLAB `fft()` και `ifft()` εκτελούν (γρήγορο) πολλαπλασιασμό πίνακα-διανύσματος με τους πίνακες DFT.

Δείτε τι κάνει η εντολή `fft(eye(N))`. Υπόμνημα: ξεκινήστε με $N = 4$. Τι κάνει η `ifft(eye(N))`;

Χρησιμοποιείτε τις `fft()` και `ifft()` για τις τρεις συναρτήσεις από την προηγούμενη άσκηση.

Ο ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

```
N=4;
trans1=fft(eye(N))
trans2=ifft(eye(N))

transfi1=ifft(f1)
transfi2=ifft(f2)
transfi3=ifft(f3)

transf1=fft(c1)
transf2=fft(c2)
transf3=fft(c3)
```

Η Εντολή `fft` κάνει τον πολλαπλασιασμό $f=FC$ όπου F ο πίνακας `fourier` και όπου C ο πίνακας των συντελεστών.

Η Εντολή `ifft` κάνει ακριβώς την αντίστροφη διαδικασία δηλαδή τον πολλαπλασιασμό $f=inv(F)C$ όπου F ο πίνακας `fourier` και όπου C ο πίνακας των συντελεστών.

Δηλαδή αν ξέρουμε μια συνάρτησης για να υπολογίσουμε τους συντελεστές `fourier` χρησιμοποιούμε `fft` και `ifft` για να κάνουμε το αντίστροφο.

Άσκηση 5:

Ελέγξτε την παραπάνω παραγοντοποίηση.

Ο ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

Άσκηση 6: το πρώτο βήμα του FFT στο MATLAB

Εδώ είναι το πρώτο βήμα του αντίστροφου FFT στο MATLAB που υπολογίζει το $f=Fc$. Κατανοείτε τον κώδικα και εφαρμοστέο σε κάποιο παράδειγμα :

```
f1 = ifft(c(1:2:N-1))*N/2;
f2 = ifft(c(2:2:N))*N/2;
d = w^(0:N/2-1);
f = [ f1 + d.*f2;
f1 - d.*f2 ];
```