Turn in your Matlab/Scilab script on Blackboard by the end of Wednesday 11/5.

Goal: Compress a given signal using Fourier series, achieving certian quality of compression in terms of L^2 norm.

Data: function $f(t)$ given as samples at $t = 0, 0.02, 0.04, \ldots, 1$. Posted at `http://goo.gl/r15Ow6` (link also available on Blackboard). Use the data in the row where ID matches the last two digits of your SUID number. It should be easy to copy it: e.g., copy-paste the entire row and delete the ID number. Spaces are valid separators of numbers in Matlab/Scilab.

Answer the following questions, based on output:

(1) What threshold did you use to achieve compression quality about 95%? How many coefficients were kept?
(2) What threshold did you use to achieve compression quality about 90%? How many coefficients were kept?
(3) Which of the two compressed signals was visibly closer to the original?

You can include the answers as comments when submitting the file on Blackboard, or as comments within the script.

Method: Use the cosine Fourier series: $f(t) = \frac{1}{2}A_0 + \sum_{n=1}^{\infty} A_n \cos \pi n t$ where

$$A_n = 2 \int_0^1 f(t) \cos \pi n t \, dt \tag{1}$$

Use the trapezoidal rule to compute the integral in (1), thus obtaining A_n for $n = 0, 1, \ldots, 15$.

The trapezoidal rule for the integral of function $g(t)$ sampled at $t = a, a+h, a+2h, \ldots, a+Nh$ is

$$\int_a^{a+Nh} g(t)\, dt \approx h \left(\frac{g(a) + g(a+Nh)}{2} + \sum_{k=1}^{N-1} g(a+kh) \right)$$

In our situation $a = 0$, $h = 0.02$, and $N = 50$.

To compress the data, we keep only coefficients A_n about a certain threshold (to be chosen by you), replacing the rest by zeros. That is, the new coefficients are

$$C_n = \begin{cases} A_n & \text{if } |A_n| > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

Reconstruct the signal from the compressed data as the sum

$$\frac{1}{2}C_0 + \sum_{n=1}^{15} C_n \cos \pi n t$$

For comparison, plot the original and reconstructed functions together.

To quantify the quality of compression, use the Parseval equality. First, calculate

$$\int_0^1 f(t)^2 \, dt \tag{2}$$

using the trapezoidal rule. Then calculate the sum on the other side of Parseval's equality, using only the coefficients you kept:

$$(C_0/2)^2 + \frac{1}{2}\sum_{n=1}^{15} C_n^2 \tag{3}$$

The ratio of (3) to (2) is a measure of the quality of compression: the higher it is, the closer the compressed signal to the original one.

Sample Matlab program (Differences with Scilab are noted at the end.)

```
data = [ ... copied from the given spreadsheet ...  ]
A = zeros(1,16);
C = zeros(1,16);
t = 0:0.02:1;
for n=1:16
    product = data.*cos(pi*(n-1)*t);
    A(n) = 2*0.02*(sum(product)-(product(1)+product(end))/2);
end
```

The computation of $A(n)$ uses trapezoidal rule: add all values of the function, then subtract half of the endpoint values because only half of those should be included.

Note that the indexing is off by one, because zero index is not allowed in Matlab. So, $A(1)$ represents A_0 in mathematical notation.

```
disp('Coefficients:');
disp(A);
threshold =       ... you choose this ...
for n=1:16
    if abs(A(n))>threshold
        C(n)=A(n);
    end
end
compressed = (C(1)/2)*ones(size(t));
for n=2:16
    compressed = compressed + C(n)*cos(pi*(n-1)*t);
end
disp('Number of coefficients kept (out of 16):');
disp(nnz(C));
```

The coefficients A_n are displayed to aid in choosing a suitable threshold. You can choose it by trial and error, beginning with, for example, half the magnitude of the largest coefficient. Adjust it to get the desired quality (95% or 90%).

The quality of compression is calculated as follows:

```
TotalEnergy = 0.02*(sum(data.^2)-(data(1)^2+data(end)^2)/2);
EnergyKept = (C(1)/2)^2+1/2*sum(C(2:16).^2);
```

```
disp('Compression quality, in percent');
disp(100*EnergyKept/TotalEnergy);
```

Finally, plot the compressed signal (in red) for comparison with the original (in blue):

```
clf()
plot(t,data)
hold on
plot(t,compressed,'r')
hold off
```

Notes for Scilab users:

- Replace `pi` with `%pi`
- Replace `product(end)` with `product($)`
- Replace `ones(size(t))` with `ones(t)`
- Remove `hold on` and `hold off`