MATH 552 **Scientific Computing II** **Spring 2016**

Homework Set 4 - SOLUTIONS

1. For N even define a discrete grid x_h for $[0, 2\pi]$ by $x_k = k*h$ where $h = 2\pi/N$ and $0 \leq k \leq N-1$. Show

$$e^{i(\frac{N}{2}+j)x_k} = e^{-i(\frac{N}{2}-j)x_k}$$

for $j = 1, 2, \ldots, \frac{N}{2} - 1$. This shows that on such a grid, wave numbers greater than $\frac{N}{2}$ are aliased to wave numbers below $\frac{N}{2}$.

ANS: There are a number of ways to show this result, one of which is by first noting that

$$e^{iNx_k} = e^{iN\frac{2\pi k}{N}} = e^{i2\pi k} = \cos(2k\pi) + i\sin(2k\pi) = 1.$$

Multiplying on the right by this expression gives

$$e^{-i(\frac{N}{2}-j)x_k}e^{iNx_k} = e^{-i\frac{N}{2}x_k}e^{ijx_k}e^{iNx_k} = e^{i(\frac{N}{2}+j)x_k}.$$

An alternative would be to simply subtract the two expressions, and using a trig identity to arrive at 0.

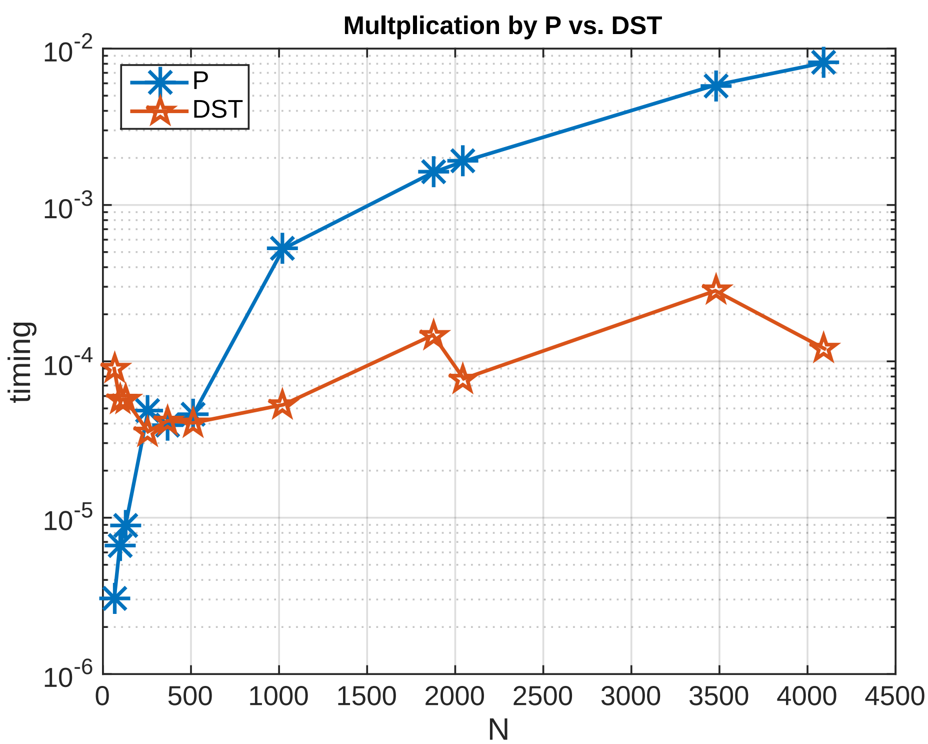
2. Given a real vector $v = (v_1, \ldots, v_{N-1})^T$ the *Discrete Sine Transform* of v is given by $\hat{v} = P^{-1}v$, where P is an $(N-1)\times(N-1)$ matrix with $P_{i,j} = (2/\sqrt{2N})\sin(ij\pi/N)$ for $i, j = 1, 2, \ldots, N-1$.

   We showed that $\hat{v}$ could be computed using an FFT, implemented by the M-file *dst.m*. For a randomly chosen v and values of N given by

$$N = [64, 96, 128, 256, 368, 512, 1024, 1874, 2048, 3477, 4096]$$

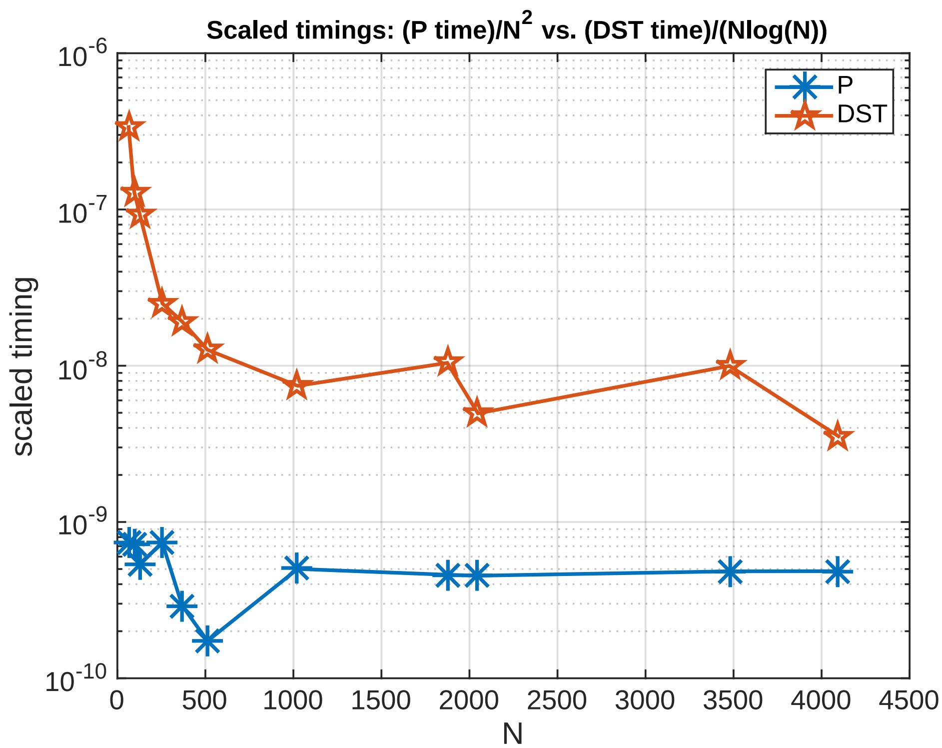
   compute $\hat{v}$ by direct matrix-vector multiplication and also by *dst.m*. Using *tic* and *roc* in MATLAB, plot the cpu time on a *semilogy* plot, and discuss the results. To obtain a reasonably accurate timing, execute each method 500 times and then take the average.

   ANS: Below is the graph. For both multiplication by P and using the DST, except for N small, the DST is clearly faster. In the case of small N, we can see that the DST is actually slower. This is most likely due to the fact that for small N matrix-vector multiplication is extremely fast because of the presence of L1 cache, a small amount of memory close to the arithmetic units with low latency, i.e. data moves in and out very rapidly.



   Let us try to get a clearer picture of the timings. Below is a graph of timings with the time divided by N^2 in the case of multiplication by P case, and divided by $N \log N$ in the DST case. Focusing on the larger values of N, on this log scale the multiplication by P is nearly a constant, confirming the expected N^2 scaling. For the DST it is not as clear because the optimality of the DST varies with N, the best case being when N is a power of 2, which is quite apparent in the results.

   Here is a copy of the code:




```
N = [64 96 128 256 368 512 1024 1874 2048 3477 4096]';
time_P    = zeros(length(N),1);
time_dst = zeros(length(N),1);
num_trials = 500;

for i = 1:length(N)
    n = N(i)
    v = rand(n-1,1);
    [P,x] = createP(n);

    tic
    for j = 1:num_trials
        v = P*v;
    end
    time_P(i) = toc/num_trials;
    tic
    for j = 1:num_trials
        v = dst(v);
    end
    time_dst(i) = toc/num_trials;
end
```

3. Consider the 2-point one-dimensional BVP

$$\begin{cases} -u'' + u = (\pi^2 \sin \pi x - 2\pi \cos \pi x)e^x \\ u(0) = u(1) = 0\,. \end{cases}$$

The exact solution is $u(x) = e^x \sin(\pi x)$.

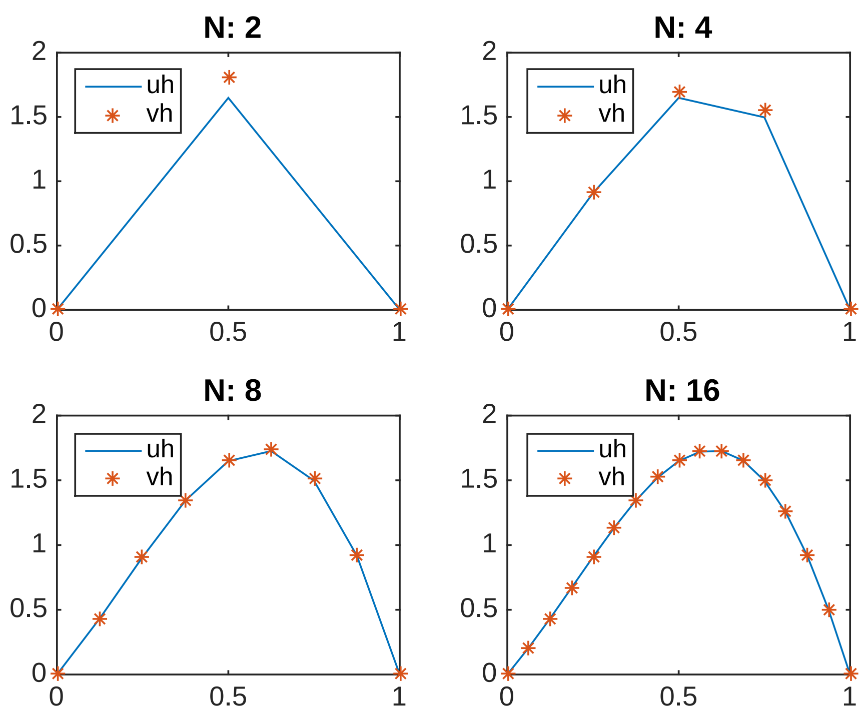
(a) Write a MATLAB script to solve the problem by the FFT method, using the *Discrete Sine Transform* as implemented by *dst.m* applied to the **2nd** order centered FD scheme, assuming $\sigma \geq 0$ is a **constant**,

$$-D^2 v_i + \sigma v_i = f_i,$$

where $D^2 = D_+ D_-$. Assume a meshsize $h = 1/2^p$, where p is a positive integer. For $p = 1 : 4$, plot the exact solution ($u(x)$ vs. x) and the numerical solution (v_i vs. x_i), including the boundary points. The 4 plots should appear separately in one figure, with axes labeled and a title for each indicating p. Investigate **subplot** in MATLAB for how to have multiple plots in a single figure window.

(b) For $p = 1 : 15$ present a table with the following data - column 1: h; column 2: $\|u_h - v_h\|_\infty$; column 3: $\|u_h - v_h\|_\infty / h^2$, where $h = 1/n$. Discuss the trends in each column. Include a copy of your code.

<u>**ANS**</u>: Here is the graph for part (a):



The table for part (b) is below. We see clear second order accuracy up to $\approx n = 2^(12) = 4096$. After that roundoff error begins to be significant and we lose the convergence to a constant in column 3.

```
       h          inf_error    inf_error/h^2
-----------------------------------------
   5.0000e-01   1.5930e-01   6.3722e-01
   2.5000e-01   5.5870e-02   8.9391e-01
   1.2500e-01   1.3945e-02   8.9248e-01
   6.2500e-02   3.5776e-03   9.1587e-01
   3.1250e-02   8.9442e-04   9.1589e-01
   1.5625e-02   2.2361e-04   9.1589e-01
   7.8125e-03   5.5926e-05   9.1630e-01
   3.9062e-03   1.3982e-05   9.1630e-01
   1.9531e-03   3.4954e-06   9.1630e-01
   9.7656e-04   8.7386e-07   9.1630e-01
   4.8828e-04   2.1843e-07   9.1617e-01
   2.4414e-04   5.4640e-08   9.1671e-01
   1.2207e-04   1.4360e-08   9.6370e-01
   6.1035e-05   6.4970e-09   1.7440e+00
   3.0518e-05   7.6755e-09   8.2415e+00
```

Here is a copy of the code:

```
n = [1:15]';
hvals = zeros(length(n),1);
err   = zeros(length(n),1);

for i = 1:length(n)
    N = 2^(n(i));
    h = 1/N;  hvals(i) = h;
    xh = h*(0:N)';
    lam = 2*(1-cos(xh(2:N)*pi))/(h^2); % eigenvalues of A_h

    uh = exp(xh).*sin(pi*xh);  % true soln u on xh grid
    fh = exp(xh).*(pi^2*sin(pi*xh)-2*pi*cos(pi*xh)); % rhs

    fh_int = fh(2:N);  % rhs is f evaluated at N-1 interior pts

    ftil = dst(fh_int);       % dst of rhs
    vtil = ftil./(lam+1); % soln in Fourier space by simple division
    vh = dst(vtil);       %  transform back to physical space
    vh = [0;vh;0];        %  set BCs

    err(i) = max(abs(uh-vh));
    if (i < 5)
     subplot(2,2,i)
     plot(xh,uh,'-',xh,vh,'*')
     legend('uh','vh','location','northwest')
```

```
     title([' N: ',num2str(N)]);
    end
end

format short e
disp(' ')
disp('       h         inf_error   inf_error/h^2')
disp('----------------------------------------')
disp([hvals  err err./(hvals.^2)])
```