CS 421: Fall 1997

Solutions to Assignment 1

1.
```
function A=lehmer(n)
% A=lehmer(n)
% The Lehmer matrix of order n defined by A(i,j)=i/j for j>=i
% and A(i,j)=j/i for j<i

A=eye(n,n);
for i=1:n
    for j=i+1:n
        A(i,j)=i/j;
        A(j,i)=i/j;
    end
end
```

   A version without loops:

```
function A=lehmer(n)
A=triu((1:n)'*(1./(1:n)))+tril((1./(1:n)')*(1:n),-1);
```

   The Lehmer matrix A has following properties:

   - It is symmetric and positive definite.
   - The inverse A^{-1} is tridiagonal and SPD.
   - In the LU-factorization $PA = LU$, L is the lower triangular part of A, $U = DL^T$ where D is a diagonal matrix, and P is the identity.
   - The eigenvalues are positive (because A is SPD), distinct, and their sum is n (because trace$(A) = n$). The eigenvectors form an orthogonal basis.

2. (a) Using four-digit decimal arithmetic (with rounding to nearest):

$$\begin{aligned} 4\pi &\approx (4.000 \times 10^0) \times (3.142 \times 10^0) \approx 1.257 \times 10^1 \\ r^2 &\approx (6.470 \times 10^3)^2 \approx 4.186 \times 10^7 \\ A = 4\pi r^2 &\approx (1.257 \times 10^1) \times (4.186 \times 10^7) \approx 5.262 \times 10^8 \text{ km}^2 \end{aligned}$$

   (b) Difference when radius is increased by 1 km:

$$\begin{aligned} (r')^2 &\approx (6.471 \times 10^3)^2 \approx 4.187 \times 10^7 \\ A' = 4\pi(r')^2 &\approx (1.257 \times 10^1) \times (4.187 \times 10^7) \approx 5.263 \times 10^8 \\ \Delta A = A' - A &\approx 1.000 \times 10^5 \text{ km}^2 \end{aligned}$$

(c) Approximation of difference based on $dA/dr = 8\pi r$:

$$\begin{aligned} 8\pi &\approx (8.000 \times 10^0) \times (3.142 \times 10^0) \approx 2.514 \times 10^1 \\ 8\pi r &\approx (2.514 \times 10^1) \times (6.470 \times 10^3) \approx 1.627 \times 10^5 \\ \Delta A \approx 8\pi r h &\approx (1.627 \times 10^5) \times (1.000 \times 10^0) \approx 1.627 \times 10^5 \text{ km}^2 \end{aligned}$$

This differs from the result in (b) by almost two thirds.

(d) Repeating (a)-(c) in six-digit decimal arithmetic:

$$\begin{aligned} 4\pi &\approx (4.00000 \times 10^0) \times (3.14159 \times 10^0) \approx 1.25664 \times 10^1 \\ r^2 &\approx (6.47000 \times 10^3)^2 \approx 4.18609 \times 10^7 \\ A = 4\pi r^2 &\approx (1.25664 \times 10^1) \times (4.18609 \times 10^7) \approx 5.26041 \times 10^8 \text{ km}^2 \end{aligned}$$

$$\begin{aligned} (r')^2 &\approx (6.47100 \times 10^3)^2 \approx 4.18738 \times 10^7 \\ A' = 4\pi (r')^2 &\approx (1.25664 \times 10^1) \times (4.18738 \times 10^7) \approx 5.26203 \times 10^8 \\ \Delta A = A' - A &\approx 1.62000 \times 10^5 \text{ km}^2 \end{aligned}$$

$$\begin{aligned} 8\pi &\approx (8.00000 \times 10^0) \times (3.14159 \times 10^0) \approx 2.51327 \times 10^1 \\ 8\pi r &\approx (2.51327 \times 10^1) \times (6.47000 \times 10^3) \approx 1.62609 \times 10^5 \\ \Delta A \approx 8\pi r h &\approx (1.62609 \times 10^5) \times (1.00000 \times 10^0) \approx 1.62609 \times 10^5 \text{ km}^2 \end{aligned}$$

Clearly the result in (c) is closer than (b) to the exact answer.

(e) The "exact" algorithm (b) gives an inaccurate answer because of cancellation: The difference between A and A' is of of the same order as the rounding errors of these quantities when using four-digit accuracy. The error in (c) comes both from rounding and the approximation made in the algorithm. The error in the formula $\Delta A \approx 8\pi r h$ is $4\pi h^2$ resulting in a relative error $4\pi h^2/8\pi r h = h/2r \approx 7.7 \times 10^{-5}$. So for this formula, using more than five-digit precision does not improve the accuracy of the answer. This example shows that it can be better to base an algorithm on an approximation than an exact formula.

(f) Use Matlab to do the computation for $h = 10^0, 10^{-1}, \ldots, 10^{-16}$ km:

```
r=6470;
j=(0:16);
h=10.^(-j);
d1=4*pi*(r+h).^2-4*pi*r.^2;
d2=8*pi*r*h;
fprintf('     h                    d1                    d2\n')
fprintf('------------------------------------------------------------\n')
fprintf('%12.1e %20.8e %20.8e\n',[h; d1; d2])
fprintf('------------------------------------------------------------\n')
```

which generated the output

```
      h                    d1                    d2
------------------------------------------------------------
     1.0e+00       1.62621402e+05       1.62608836e+05
     1.0e-01       1.62610092e+04       1.62608836e+04
     1.0e-02       1.62608961e+03       1.62608836e+03
     1.0e-03       1.62608848e+02       1.62608836e+02
     1.0e-04       1.62608837e+01       1.62608836e+01
     1.0e-05       1.62608832e+00       1.62608836e+00
     1.0e-06       1.62608862e-01       1.62608836e-01
     1.0e-07       1.62608027e-02       1.62608836e-02
     1.0e-08       1.62607431e-03       1.62608836e-03
     1.0e-09       1.62720680e-04       1.62608836e-04
     1.0e-10       1.62720680e-05       1.62608836e-05
     1.0e-11       1.54972076e-06       1.62608836e-06
     1.0e-12       1.78813934e-07       1.62608836e-07
     1.0e-13       0.00000000e+00       1.62608836e-08
     1.0e-14       0.00000000e+00       1.62608836e-09
     1.0e-15       0.00000000e+00       1.62608836e-10
     1.0e-16       0.00000000e+00       1.62608836e-11
------------------------------------------------------------
```

The relative difference between A and A' is

$$\frac{A'-A}{A} \approx \frac{8\pi r h}{4\pi r^2} = \frac{2h}{r}.$$

For $\Delta A = A' - A$ we loose all accuracy when this becomes smaller than machine epsilon, which happens when h is of the order 10^{-13}.

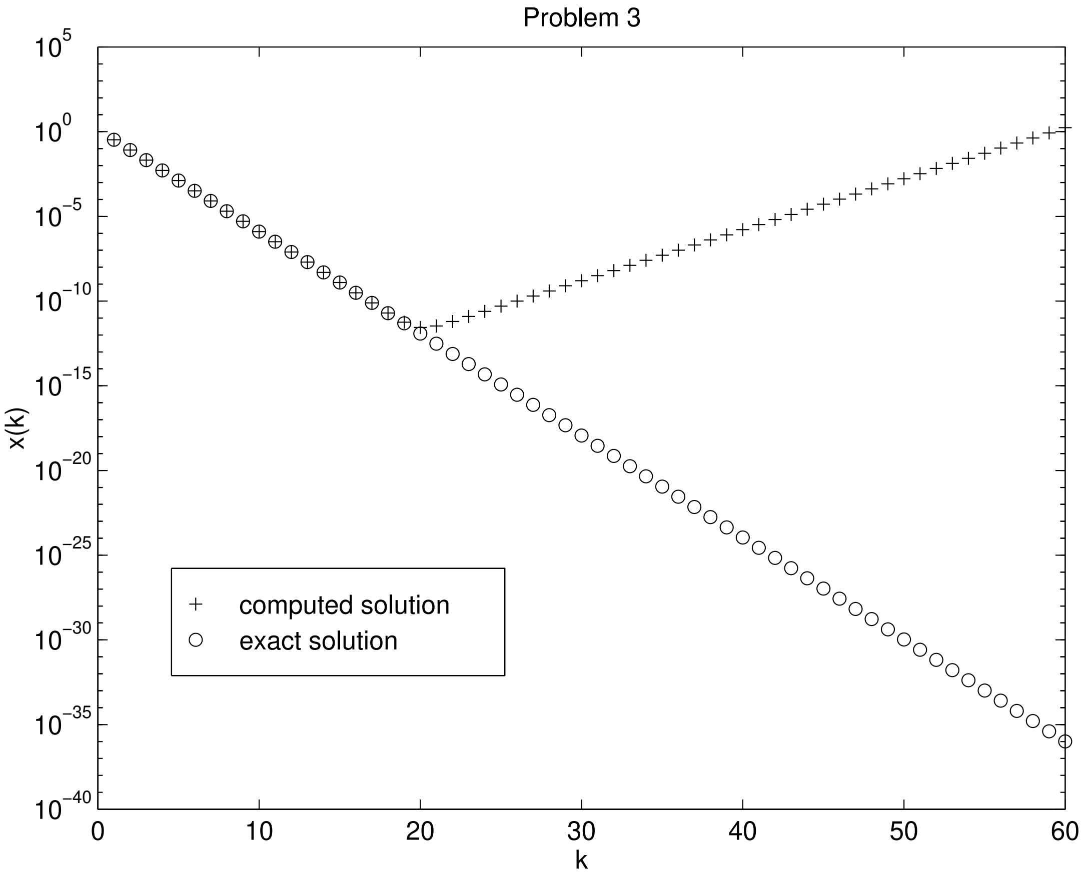
3. prob3.m:

```
% Compute the sequence defined by
%    x(k+1)=2.25*x(k)-0.5*x(k-1), x(1)=1/3, x(2)=1/12
% and make a semilog plot to compare it with exact solution.

n=60;
x=zeros(1,n);       % Allocate memory for x
x(1:2)=[1/3 1/12];
for k=2:n-1
    x(k+1)=2.25*x(k)-0.5*x(k-1);
end
y=4.^(1-(1:n))/3;   % The exact solution

semilogy(1:n,x,'+',1:n,y,'o')
xlabel('k')
ylabel('x(k)')
```

```
legend('computed solution','exact solution')
title('Problem 3')
```



The exact solution decreases monotonically, but after about 20 steps the computed solution starts increasing. The explanation of this behavior lies in the fact that the general solution of the difference equation is $x_k = \alpha(\frac{1}{4})^k + \beta 2^k$. The coefficient β is zero if and only if $x_2 = \frac{1}{4}x_1$. But 1/3 and 1/12 are not represented precisely in binary floating point arithmetic, causing an initial rounding error. So for the computed solution $x_{k+1} \neq \frac{1}{4}x_k$ and this amounts to a nonzero β. To begin with, the rounding errors are small, but they are amplified by a factor of 2 at each step. And in the long run they dominate the computation, since the rate of growth of the errors is bigger than of the exact solution.

There is a rounding error introduced at each step and some students thought it was the accumulation of these that was responsible for the growth of the error. But if the errors were not amplified, we would not get an exponential growth.