Instructor: Emre Mengi

Math 504 (Fall 2011)

Study Guide for Weeks 8-10
This homework concerns the following topics

- IEEE Floating Point Arithmetic (Trefethen&Bau, Lecture 13)
- Condition Numbers (Trefethen&Bau, Lectures 12&18)
- Backward Error Analysis (Trefethen&Bau, Lectures 14-15)
- Gaussian Elimination (Trefethen&Bau, Lecture 20)
- Gaussian Elimination with Partial Pivoting, GEPP (Trefethen&Bau, Lecture 21)
- Backward Error Analysis of GEPP (Trefethen&Bau, Lecture 22)

Homework 4 (Assigned on November 30th, Wednesday; Due on December 16th, Friday by 17:00)
Please return the solutions only to the questions marked with (*) and (**). The rest of the questions are provided to practice on your own. Questions marked with (**) are Matlab questions. Please attach Matlab outputs, routines and figures, whichever are necessary.

1.(*) Let $x = 2$, $y = 2^{24}$, $w = -1$ and $z = 2^{-23}$. In parts **(b)** and **(c)** $\oplus$ and $\otimes$ denote the addition and multiplication in IEEE single precision. (Note: machine precision in single precision is 2^{-24}.)

(a) Find the floating point number in IEEE single precision that is closest to and greater than x. Repeat this for y and z.

(b) Perform the following operations; **(i)** $x \oplus y$, **(ii)** $x \oplus z$ and **(iii)** $(x \oplus y) \otimes (x \oplus z)$.

(c) Verify that $(w \oplus x) \oplus z \neq w \oplus (x \oplus z)$. This shows that the addition in IEEE floating point arithmetic is not associative.

(d) Is the addition in IEEE floating point arithmetic commutative? In other words does the equality $x_1 \oplus x_2 = x_2 \oplus x_1$ hold for any two floating point numbers x_1, x_2 in single or double precision?

2. (Modified from Watkins, **Exercise 2.5.7**) Matlab performs all of the computations below in double precision arithmetic. You should type "`format long e`" before running the m-files below to view the outputs in double precision.

(a) A Matlab m-file `accuracy.m` is provided on the course website and its content is given below.

```
function [] = accuracy()

a = 1;
u = 1;

b = a+u;

while b ~= a
   u = 0.5*u;
   b = a+u;
end

u

return
```

Please save the m-file. In Matlab go to the directory in which you saved the file. You can call the m-file by typing "`accuracy()`" in this directory. The m-file prints out the value stored in u right before termination. What does this value of u correspond to in IEEE double precision?

(b) Now consider the Matlab m-file `accuracy2.m` provided on the course website.

```
function [] = accuracy2()

a = 1;

while a ~= inf
   b = a;
   a = 2*a;
end

b


return
```

Once again save the m-file, go to the directory in which you saved the file and run the m-file by typing "`accuracy2()`". The m-file prints out the value stored in b before termination. What does this value of b correspond to in IEEE double precision?

(c) Finally the Matlab m-file `accuracy3.m` is provided on the course website.

```
function [] = accuracy3()

a = 1;

while a ~= 0
   b = a;
   a = 0.5*a;
end

b

return
```

Save and run this m-file. What does the value of b printed out at termination correspond to in IEEE double precision arithmetic?

3. Consider a floating point standard where the numbers are represented using the normalized binary floating point respresentation $S \times 2^E$. In these standards suppose four bits (binary digits) are reserved for the significand $S = (1.b_1b_2b_3b_4)_2$ and four bits for the exponent E. For the exponent four bits are used to represent 14 integer exponent values (the sequence of all zeros are reserved for the subnormalized form; the sequence of all ones are reserved for ∞ and NaN) ranging in the interval $[-6, 7]$.

(a) What is the largest floating point number in these standards?

(b) What is the maximal relative error possible due to the representation of a number (also called the machine precision or unit round-off error) in these standards?

4.(*) Let $z = x^T y$ where $x, y \in \mathbb{R}^n$. Show that the computed product $\tilde{z}$ in IEEE floating point arithmetic satisfies

$$\tilde{z} = x^T(y + \delta y)$$

where $\|\delta y\|_1 \leq n\epsilon_{\text{mach}}\|y\|_1$.

5.(*) Find the relative condition number for the solution $x \in \mathbb{C}^n$ of the linear system $Ax = b$ with respect to perturbations in $b \in \mathbb{C}^n$. Assume $A \in \mathbb{C}^{n\times n}$ is fixed.
(Note: in the definition of the relative condition number use the vector 2-norm.)

6. Find the absolute and relative condition numbers of the product

$$C(B) = A \cdot B$$

with respect to perturbations in $B \in \mathbb{C}^{m\times n}$ where $A \in \mathbb{C}^{p\times m}$ is fixed.
(Note : in the definitions of the condition numbers use the matrix 2-norm.)

7. Let $A = QR$ be a full QR factorization of $A \in \mathbb{C}^{m\times n}$. Find the relative condition number of the factor R with respect to perturbations in A assuming Q is fixed. Once again use the matrix 2-norms in the definitions.

8.(**) Consider

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1.0002 \\ 1 & 1.0002 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ 0.0001 \\ 4.0001 \end{bmatrix}.$$

(a) Find the exact solution x to the least squares problem $Ax \approx b$. Do not use Matlab.

(b) What are the relative condition numbers for the following problems? Numerical answers are acceptable (i.e., now you can use Matlab).

(i) y as a function of b

(ii) x as a function of b

(iii) y as a function of A

(iv) x as a function of A

(c) Give examples of perturbations δb that approximately attain the relative condition numbers for problems **(i)**-**(ii)** in part **(b)**. Again numerical answers are acceptable.

9. Let us focus on the least squares problems $Ax \approx b_1$ and $Ax \approx b_2$ with

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 2 \\ -2 \\ 10^{-6} \end{bmatrix}, \quad \text{and} \quad b_2 = \begin{bmatrix} 2 \\ 2 \\ 10^{-6} \end{bmatrix}$$

(a) Consider the solutions x_1 and x_2 to the least squares problems $Ax \approx b_1$ and $Ax \approx b_2$. In theory which of the solutions x_1 and x_2 would you expect to be more sensitive to perturbation in b?

(b) By slightly perturbing b_1 and b_2 estimate the relative condition numbers for the solutions of the least squares problems $Ax \approx b_1$ and $Ax \approx b_2$ with respect to b using Matlab. Do you observe that these practical estimates for the relative condition numbers are consistent with what is expected in theory?

10.(*) For each of the problems below **(i)** determine the backward error if this is possible, and **(ii)** indicate whether the computations can be done in a backward stable manner in IEEE floating point arithmetic. Also for simplicity assume in all parts x is representable in IEEE floating point arithmetic.

(a) $x \oplus 2$ as a function of $x \in \mathbb{R}$

(b) $2 \otimes x$ as a function of $x \in \mathbb{R}$

(c) $x \otimes x$ as a function of $x \in \mathbb{R}$

(d) the product xx^T as a function of $x \in \mathbb{R}^n$ performed in IEEE floating point arithmetic

11. Suppose that the sum $f(A) = A + A^T$ where $A \in \mathbb{R}^{n \times n}$ is computed in IEEE floating point arithmetic, and $\tilde{f}(A)$ denotes the computed value of the sum. For simplicity you can assume that $\mathrm{fl}(A) = A$.

Show that

$$\tilde{f}(A) = (A + \delta A) + (A + \delta A)^T$$

for some δA such that

$$\|\delta A\|_1 \leq \epsilon_{\text{mach}} \|A\|_1.$$

12. (Trefethen&Bau, Exercise 15.2)
Consider an algorithm for the problem of computing the (full) SVD of a matrix. The data for this problem is a matrix A, and the solution is three matrices U (unitary), Σ (diagonal), and V (unitary) such that $A = U\Sigma V^*$. (We are speaking here of explicit matrices U and V, not implicit representations as products of reflectors.)

(a) Explain what it would mean for this algorithm to be backward stable.

(b) In fact, for a simple reason, this algorithm cannot be backward stable. Explain.

13.(*) Suppose that the upper triangular linear system

$$Rx = b$$

is solved by back substitution. It can be shown that in IEEE floating point arithmetic the computed value $\tilde{x}$ satisfies

$$(R + \delta R)\tilde{x} = b$$

for some δR such that

$$\frac{\|\delta R\|}{\|R\|} = O(\epsilon_{\text{mach}}).$$

(See Lecture 17, in particular Thm 17.1 on page 122, in Trefethen&Bau.)

Perform a backward error analysis to deduce a tight upper bound for the forward error

$$\frac{\|\tilde{x} - x\|}{\|x\|}.$$

14. Let $A = QR$ be an exact QR factorization for a matrix $A \in \mathbb{R}^{n\times n}$. Suppose also that the QR factorization can be computed by an algorithm such that

$$Q\tilde{R} = A + \delta A$$

where $\tilde{R}$ is the computed upper triangular factor and $\frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{mach}})$.
Derive a tight upper bound for the forward error

$$\frac{\|\tilde{R} - R\|}{\|R\|}.$$

The choice of norm above is up to you. But it simplifies the matters to use the 2-norm or the Frobenius norm.

15.(*) Given a non-singular matrix $A \in \mathbb{R}^{n\times n}$ and a vector $y_0 \in \mathbb{R}^n$ Define the sequence of vectors $\{y_k\}$ in $\mathbb{R}^n$ for $k \geq 1$ as

$$Ay_k = y_{k-1}.$$

(a) Calculate an LU factorization for

$$A = \begin{bmatrix} 1 & 4 \\ 3 & 1 \end{bmatrix}$$

without using pivoting.

(b) Suppose $y_0 = \begin{bmatrix} -5 \\ 7 \end{bmatrix}$. Calculate the vectors $y_1, y_2 \in \mathbb{R}^n$ where A is as given in part **(a)** by using your LU factorization from part **(a)**, and forward and back substitutions.

(c) Devise an algorithm for the efficient computation of $y_1, y_2, \ldots, y_n$ for a general matrix $A \in \mathbb{R}^{n\times n}$. You can assume that it is possible to reduce A into an upper triangular matrix by only applying row-replace operations. Provide also the total flop count for your algorithm.

16. The kth principal submatrix of $A \in \mathbb{C}^{n\times n}$ is the upper left-most $k \times k$ portion of A. Show that A has an LU factorization of the form $A = LU$ where L is unit lower triangular with ones along the diagonal, and U is invertible and upper triangular if and only if all principal submatrices of A are invertible.

17. Consider the matrices

$$A_1 = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} \quad \text{and } A_2 = \begin{bmatrix} 1 & 1 & 3 \\ 2 & 2 & 0 \\ 3 & 0 & 1 \end{bmatrix}.$$

You are expected to perform all the computations in both parts by hand.

(a) Compute the factorizations of the form $PA = LU$ for each one of A_1 and A_2 by applying the LU factorization algorithm with partial pivoting. Recall that in the factorization $PA = LU$ the matrix P is a permutation matrix, L is a unit lower triangular matrix (*i.e.* a lower triangular matrix with 1s on the main diagonal) and U is an upper triangular matrix.

(b) Use the factorizations computed in part **(a)** to solve the linear systems

$$A_1 x = \begin{bmatrix} -2 \\ 4 \\ -4 \end{bmatrix} \quad \text{and} \quad A_2 x = \begin{bmatrix} -2 \\ 8 \\ 7 \end{bmatrix}.$$

18. Modify the Gaussian elimination with partial pivoting for the numerical solution of a linear system of the form $Tx = b$ where $T \in \mathbb{C}^{n\times n}$ is tridiagonal (i.e., $t_{ij} = 0$ for all i, j such that $|i - j| > 1$ and $b \in \mathbb{C}^n$). The number of flops required by your algorithm must be $O(n)$. Write down a pseudocode for your algorithm.

19.(**) Given a non-singular $n \times n$ dense matrix A (dense in this context means most of the entries, for instance more than half of the entries, of A are nonzero). The standard approach to solve the linear system $Ax = b$ is as follows.

1. Compute an LU factorization $PA = LU$ by partial pivoting.
2. Permute the right-hand side matrix, that is form $\hat{b} = Pb$.
3. Let $\hat{x} := Ux$. Solve the lower triangular system $L\hat{x} = \hat{b}$ by forward substitution.
4. Solve the upper triangular system $Ux = \hat{x}$ by back substitution.

(a) Write a Matlab function to compute the LU factorization of A by partial pivoting. Your function must take a matrix A as input and return a lower triangular matrix L and an upper triangular matrix U as well as a vector P as outputs. The vector P represents the permutation matrix. In particular at the kth iteration if the kth and lth rows are swapped, then keep this information as $P(k) = l$. Your Matlab routine should look like

```
function [P,L,U] = lu_factor_pivot(A)

return;
```

(b) Implement a Matlab routine to apply a given permutation matrix P to a vector b, that is implement a routine of the form

```
function [bhat] = permuteb(P,b)

return;
```

where $bhat = Pb$.

(c) Write another Matlab function to solve the system $Ax = b$ following the steps described above. Here is how your function should look like

```
function x =  linear_sys_solver(A,b)

return;
```

The routines for forward and back substitution are provided on the course website. Have a careful look and make sure you understand them. For the computation of an LU factorization and for the permutation of the vector b use the routines that you implemented in parts **(a)** and **(b)**.

20.(**) Cleve Moler's book "numerical computing with Matlab" is freely available at

`http://www.mathworks.com/moler/lu.pdf`

In particular Question 2.3 in Moler's book concerns the calculation of the net forces on a plane truss, that requires the solution of a linear system. Solve the linear system in Matlab using your linear system solver based on LU factorization from Question 19.(c).

21. Solve Question 2.4 in Moler's book concerning the voltages and currents on an electrical circuit. Use your linear system solver from Question 19.(c).

22.(**) One important application where very large linear systems arises is the numerical solution of partial differential equations. Consider the Poisson equation

$$u_{xx}(x, y) + u_{yy}(x, y) = 0$$

where $x, y \in [0, 1]$ with the boundary conditions

$$u(x, 0) = x^2 + 1, \;\; u(0, y) = -y^2 + 1, \;\; u(x, 1) = x^2, \;\; u(1, y) = 2 - y^2.$$

The solution sought $u(x, y)$ is a twice differentiable function. Above u_{xx} and u_{yy} denotes the partial derivatives of u with respect to x and y twice, respectively.

One can numerically estimate the solution using finite differences. In particular

$$u_{xx}(x, y) \approx \frac{u(x + h, y) - 2u(x, y) + u(x - h, y)}{h^2}$$

and

$$u_{yy}(x, y) \approx \frac{u(x, y + h) - 2u(x, y) + u(x, y - h)}{h^2}.$$

Indeed the finite difference formulas on the right-hand sides approach the corresponding partial derivatives as $h \to 0$.

Now use a two dimensional mesh $\{u_{ij}\}$ where $i = 0, \ldots, n$ and $j = 0, \ldots, n$ to estimate the solution $u(x, y)$. In particular u_{ij} corresponds to the numerical estimate for $u(ih, jh)$ where $h = 1/n$. Using the finite difference formulas we may approximate the Poisson equation

$$u_{xx}(ih, jh) + u_{yy}(ih, jh) = 0$$

by

$$\frac{u((i+1)h, jh) - 2u(ih, jh) + u((i-1)h, jh)}{h^2} + \frac{u((i+1)h, jh) - 2u(ih, jh) + u((i-1)h, jh)}{h^2} = 0$$

or

$$\frac{u_{(i+1)j} - 2u_{ij} + u_{(i-1)j}}{h^2} + \frac{u_{i(j+1)} - 2u_{ij} + u_{i(j-1)}}{h^2} = 0$$

equivalently

$$u_{(i+1)j} + u_{i(j+1)} - 4u_{ij} + u_{(i-1)j} + u_{i(j-1)} = 0.$$

Choose $h = 0.025$ and set up a linear system with the unknowns $\{u_{ij}\}$ where $i = 1, \ldots, n-1$ and $j = 1, \ldots, n-1$. Estimate the solution of the Poisson equation by solving the resulting linear system. Use your linear system solver from Question 19.(c).

23. (Trefethen&Bau, 20.2 and 21.2)
When one works on the solution of banded linear systems, it is desirable not to destroy the banded structure (or at least to increase the bandwidth by a small amount) of the system for efficient computation.

Suppose $A \in \mathbb{C}^{n \times n}$ is banded with bandwidth p, that is $a_{ij} = 0$ whenever $|i - j| > p$.

(a) What can you say about the sparsity patterns of the factor L and U if you calculate the LU factorization without pivoting? (Assume A is reducible into an upper triangular matrix by applying only row-replace operations.)

(b) Repeat part **(a)** but when LU factorization is calculated using partial pivoting.

24. (Trefethen&Bau, 21.6)
Suppose $A \in \mathbb{C}^{n \times n}$ is *strictly diagonally dominant*, that is for all k,

$$|a_{kk}| > \sum_{j=1, j \neq k}^{n} |a_{jk}|.$$

Show that if LU factorization with partial pivoting is used, no row-interchange operation is performed. (In other words LU factorization with or without partial pivoting are the same for strictly diagonally dominant matrices.)

25. Recall that the growth factor for $A \in \mathbb{C}^{n\times n}$ is defined as

$$\rho_{\max} = \frac{\max_{ij} |u_{ij}|}{\max_{ij} |a_{ij}|}$$

where u_{ij} denotes the entries of the U factor in the LU factorization $A = LU$. It can be shown that without pivoting $\rho_{\max} \leq 2^{n-1}$. Write down a 7×7 matrix for which $\rho_{\max} = 2^6$.

26. On the course webpage the Matlab routine `random_growth_factor.m` is provided. (It calls the auxiliary routine `growth_factor.m` .) When this routine is called as

```
>> random_growth_factor(max_dim, m);
```

It plots the growth factors for m random matrices with sizes varying between one and `max_dim`. The horizontal and vertical axes represent the size and the growth factor of the matrix, respectively. The red curve corresponds to $f(n) = \sqrt{n}$.

Run the routine for 400 matrices of size at most 200. Attach the plots generated. Can you observe that for a matrix $A \in \mathbb{C}^{n\times n}$ typically the growth factor does not exceed $\sqrt{n}$.

27.(*) Let $A, B \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$. Suppose that you solve the linear system

$$(A + B)x = b$$

for $x \in \mathbb{C}^n$ in IEEE floating point arithmetic by first adding matrices A and B followed by an LU factorization with pivoting, a forward substitution and finally a back substitution. Denote the computed solution by $\hat{x}$.

(a) Determine the backward error w.r.t. the 1-norm, i.e., find $\|\Delta\|_1$ where $\Delta \in \mathbb{C}^{n\times n}$ satisfies

$$(A + B + \Delta)\hat{x} = b.$$

Your answer should be in terms of A, B, n, $\rho_{\max}$ and ϵ_{mach}.

(b) Find an upper bound for the forward error

$$\frac{\|\hat{x} - x\|_1}{\|x\|_1}.$$

28. Given $A \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$ and a positive integer m. Consider the following linear system

$$A^m x = b. \tag{1}$$

You need to solve such a linear system for instance for the inverse iteration for the calculation of the smallest eigenvalue of A in modulus. Assume that an LU factorization for A is computed with partial pivoting initially. Then the linear systems

$$Ax_k = x_{k-1}$$

are solved $k = 1, \dots, m$ by exploiting the LU factorization and based on forward and back substitutions. Here $x_0 = b$ and in exact arithmetic x_m is the solution for $A^m x = b$.

Repeat Question 27 but for the numerical solution of (1) as described in the paragraph above.

29.(**) The purpose of this question is to gain insight into the cause of the numerical error of the LU factorization with partial pivoting in practice. In theory when a linear system $Ax = b$ is solved by means of the LU factorization with partial pivoting, the overall procedure is backward stable. The computed solution $\hat{x}$ satisfies $(A + \delta A)\hat{x} = b$ for some δA with

$$\|\delta A\|_1 \leq 3n^3 \rho_{\max} \epsilon_{\text{mach}} \|A\|_1. \tag{2}$$

Here $\rho_{\max}$ is the growth factor as defined in Question 26. In the worst case $\rho_{\max} = 2^{n-1}$. But in practice $\rho_{\max}$ exceeds $\sqrt{n}$ only for extreme examples. The consequence of the backward error (2) is a bound for the forward error given by

$$\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq 3n^3 \rho_{\max} \epsilon_{\text{mach}} \|A\|_1 \|A^{-1}\|_1. \tag{3}$$

Recall that above the quantity $\kappa(A) = \|A\|_1 \|A^{-1}\|_1$ is called the condition number of the matrix A.

On the course website the following Matlab m-files are made available.

- `plot_growth_factor.m` (plots $\rho_{\max}$ with respect to the size of the matrix n for a given class of matrices)
- `plot_condition_factor.m` (plots the condition number $\kappa(A) = \|A\|_1 \|A^{-1}\|_1$ with respect to the size of the matrix n for a given class of matrices)
- `growth_factor` (computes the growth factor of a matrix; this is an auxiliary m-file called by `plot_growth_factor.m`)

You will only need to call the first two m-files above (unless you would like to entertain yourself with further numerical experiments), but you need to save all three on your computer. Both of the first two routines take three parameters. The first two are integers *lb*, *ub* which determine the interval $[lb, ub]$ in which the size of the matrix n varies. The third is a string (such as 'lotkin', 'kahan', 'leslie', 'randn') identifying the family of matrices. The question is "what are the allowable families?" There is a built-in Matlab routine called `gallery`. It generates matrices from various families. You can type `help gallery` for the names of the specific families. Most of the families that can be generated by `gallery` are allowable families for `plot_growth_factor.m` and `plot_condition_factor.m`. But you will specifically experiment with 'lotkin', 'ris', 'riemann' and 'chebvand'. (Though yet again feel free to entertain yourself with others.) For instance the commands

```
>> plot_growth_factor(5,50,'riemann');
>> plot_condition_number(5,50,'riemann');
```

plot $\rho_{\max}$ and $\kappa(A)$ for the Riemann matrices with sizes varying in between 5 and 50.

(a) Plot the growth factors for 'lotkin', 'ris', 'riemann' and 'chebvand' with sizes varying in $[5, 50]$. Do you observe $\rho_{\max}$ exceeding $\sqrt{n}$ significantly?

(b) Plot the condition numbers for 'lotkin', 'ris', 'riemann' and 'chebvand' with sizes varying in $[5, 10]$. Order these families according to their condition numbers.

(c) Consider the 10×10 linear systems $Ax = b$ with A equal to one of 'lotkin', 'ris', 'riemann' and 'chebvand'. For each of the matrices 'lotkin', 'ris', 'riemann' and 'chebvand' of sizes 10×10 repeat the following. Set x equal to the vector of ones, then define $b = Ax$. Finally attempt to solve the system $A\hat{x} = b$.

- **(i)** Based on equation (3) and your observations in part **(b)** in theory how many of the decimal places of x (the exact solution) and $\hat{x}$ (the computed solution) would you expect to match?
- **(ii)** Now perform the suggested computations in Matlab. You can generate the specific matrices by typing

  ```
  >> A = gallery(matname,10);
  ```

  where matname is one of `'lotkin'`, `'ris'`, `'riemann'` and `'chebvand'`. Do you observe the accuracy that you expect in **(i)**? (Note : To see the computed results in double precision you should type `>> format long e;` in Matlab.)