BOOK PROBLEMS

K&C 4.3: 16. (Optional but recommended: also do 17).

MORE PROBLEMS

Note on operation counts: In all problems, count ‘operations’ by counting multiplications/divisions only. You need only be precise about the leading term, e.g. $n^2 + \cdots$ as long as you can justify your answer. This allows you to avoid some tedious counting (mostly arithmetic not done in a for loop like initialization). If you want, you can use Wolfram Alpha etc. to calculate sums like

$$\sum_{k=1}^{n} k^2 + 2k = \frac{1}{6} n(n+1)(2n+7) = \frac{1}{3} n^3 + O(n^2).$$

Problem 1. Prove that if $A, B$ are both unit lower triangular, $n$ by $n$ matrices then their product $C = AB$ is also unit lower triangular.

Problem 2. a) Find the number of operations required to solve

$$Ux = b$$

using back substitution (where $U$ is $n \times n$ and upper triangular).

b) Let $A, B$ be $n \times n$ matrices and let $C = AB$. Find the number of operations required to compute $C$ via the standard formula:

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}.$$

Note: there are fancier ways to compute matrix products that can be faster.
**Problem 3.** (From Golub and Van Loan, *Matrix Computations*). Let $A$ be an invertible $n \times n$ matrix and $b \in \mathbb{R}^n$. a) Explain how to solve
\[ A^k x = b \]
without computing $A^k$. Hint: first compute the factorization $PA = LU$.

In terms of operation counts, how does this compare to computing $C = A^k$ directly and then solving $Cx = b$? In both cases, assume the inputs are $A, k$ and $b$ and the output is $x$.

b) Suppose $y, z \in \mathbb{R}^n$ and we want to compute
\[ x = y^T A^{-1} z. \]
Explain how to do this (using LU factorization) without computing the inverse $A^{-1}$.

---

**Problem 4.** The following example is due to Wilkinson\(^1\): For a given $n$, define an $n \times n$ matrix $A$ as follows:
\[
\begin{align*}
a_{ii} &= 1, & i < n \\
a_{in} &= (-1)^{i+1}, & 1 \leq i \leq n \\
a_{ij} &= (-1)^{i+j-1}, & i > j.
\end{align*}
\]

For $n = 3$,
\[
A = \begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & -1 \\
-1 & 1 & 1
\end{bmatrix}.
\]

For $n = 4$:
\[
A = \begin{bmatrix}
1 & 0 & 0 & 1 \\
1 & 1 & 0 & -1 \\
-1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1
\end{bmatrix}.
\]

For $n = 5$:
\[
A = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & -1 \\
-1 & 1 & 1 & 0 & 1 \\
1 & -1 & 1 & 1 & -1 \\
-1 & 1 & -1 & 1 & 1
\end{bmatrix}.
\]

(The main diagonal is all ones, then the diagonals alternate between $+1$ and $-1$ from the first sub-diagonal down to the bottom, and the last column is replaced by alternating 1’s and $-1$’s, starting with $+1$ at the top).

a) Find the factorization $PA = LU$ using Gaussian elimination with partial pivoting for $n = 3$ (hint: you should not have to do any pivoting).

b) Do the same for $n = 4, 5, 6, \cdots$ until you see a pattern emerge. For a general $n$, what is the last column of $U$? Explain (without proof!) what is happening.

---

\(^1\)An important numerical analyst who developd much of the framework for numerical linear algebra and analysis of numerical errors, among many other things.
Hint: For (b), you can use \([L, U, P] = lu(A)\) in MATLAB (or python equivalent), your code from the next problem, \texttt{lugui}\textsuperscript{2} or do the calculations by hand (not recommended).

---

**Computational problems**

**Problem 5.** a) Write a function \([A, p] = \texttt{lug}(A, \texttt{pivot})\) that implements Gaussian elimination with or without scaled partial pivoting (if \texttt{pivot} is \texttt{true} or \texttt{false}). Use the algorithm in p171-p172 of K&C (same as discussed in class). The output should be a permutation vector \(p\) and a single matrix \(A\) such that \(PA\) stores \(L\) and \(U\).

Do so without actually swapping rows in the matrix \(A\) (just use the permutation vector).

b) Write a function \(x = \texttt{solve}(A, p, b)\) that takes the outputs \(A\) and \(p\) from \texttt{lug} and solves \(LUx = Pb\) using forward/back substitution. Note that the input \(A\) is no longer the original matrix \(A\); it is the object that contains the (permuted) LU factorization.

c) Use your code to answer 4.3.21ab in K&C.

**What to turn in:** Code for \texttt{lug} and \texttt{solve} via online submission. Report your answer for (c) in the written part of the homework. \textit{Note: if using python, you can put both functions in one \texttt{py} file; if using MATLAB you will need two separate files.}

---

**Code considerations:** It is possible to ‘vectorize’ some of the calculations if done carefully, e.g. if \(i\) is a scalar and we set \(j = (k+1):n\) then \(A(i,j) = A(i,j) + B(i,j)\) sets \(a_{ij}\) to \(a_{ij} + b_{ij}\) for \(j = k + 1, \cdots, n\). In fact, one can do more (see \texttt{lutx} in Moler’s book) but you do not have to worry about such optimizations.

Note also that the code in Moler’s book is a bit different from the code asked for in (a) and (b) as it does row swapping, some vectorized calculations and has values stored in a different way. Don’t copy this code (although you may consult it as an example).

\textsuperscript{2}From the NCM toolbox, the MATLAB code package from Moler’s book