Richard Abele November 13, 2024

Numerical Methods - pSet03

1 Problem 1

A program was created to accept a command line argument that determines the size of the matrices handled in the program. The program promps the user for the values of an `n x n` matrix and a vector of dimension `n`. The program then calculates the product using the the function found in `lib/multiply.c` and it was tested using the given matrix and compared to an online calculator. The program works by iterating through the given vector and matrix by the individual components and calculating the appropriate products and sums.

To be more precise, the given vector multiplication yields the following when solved with the above function:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 11 \\ 12 \end{pmatrix} = \begin{pmatrix} 68 \\ 167 \\ 266 \end{pmatrix}$$

To solve the second part of `problem 1`, a function was created under `lib/calc_system.c` that calculates the solution vector for a system of equations when given the matrix (in the form of an upper-triangle matrix) and the result vector. This program works by solving the values of the solution vector starting from `n` and then progressing backwards to eventually reach `1`. Solving the individual values involves first subtracting the products of the known solution values and their matching matrix elements from the result vector, and then finally dividing by the matching diagonal matrix element. The validity of this algorithm was then tested by implementing a `test_sol` function that accepts the original matrix, the solution vector, and the result vector, and compares the product of the original matrix and solution vector to the result vector, returning the total error.

We are given the system of equations

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 10 \\ 5x_2 + 6x_3 &= 11 \\ 9x_3 &= 12 \end{aligned}$$

and asked to numerically calculate the solution. The implemented algorithm yields a result of

$$\begin{aligned} x_1 &= 4.80 \\ x_2 &= 0.60 \\ x_3 &= 1.33. \end{aligned}$$

These results agree with the testing function and with the results obtained using an online calculator.

2 Problem 2

For `problem 2` the initial program was slightly modified and then expanded. The program was changed to use compiled values for the matrix and target vector to make testing more practical (user inputs can be enabled by uncommenting the appropriate lines). After these small alterations were made, a function called `gaussinator` was implemented to transform a given matrix and its matching result vector to row-echelon form through Gaussian elimination, printing the transformed matrix and vector, and finally solving the system of equations.

Implementation of the function is largely based on the pseudocode provided in the exercise instructions, though some special considerations had to be taken in order to prevent critical values from being overwritten before a loop was completed (notice the `temp` variables in the code).

The algorithm is based on the following simple set of instructions:

1. Starting at the top of the matrix at $m_{1,n}$, the row and b_1 is divided by the value $m_{1,1}$.
2. Row one times value of $m_{2,1}$ is then subtracted from row two (analogous with the result vector). This in turn causes $m_2, 1$ to become zero.
3. Steps 1 and 2 are repeated for the remaining rows, starting now at row two. This results in an upper triangle matrix that can then be solved using the program from *problem*1.

In the assignment we are given the following system of equations to solve:

$$\begin{pmatrix} 2.0 & 0.1 & -0.2 \\ 0.05 & 4.2 & 0.032 \\ 0.12 & -0.07 & 5.0 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 10 \\ 11 \\ 12 \end{pmatrix}.$$

Running the diagonalization procedure yields the following:

$$\begin{pmatrix} 1.000000 & 0.500000 & -0.100000 \\ 0.000000 & 1.000000 & 0.008815 \\ 0.000000 & 0.000000 & 1.000000 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 5.000000 \\ 2.561048 \\ 2.313067 \end{pmatrix}.$$

Finally, numerically solving this system yields a solution vector of

$$\vec{x} = \begin{pmatrix} 5.104274 \\ 2.540659 \\ 2.313067 \end{pmatrix}.$$

This result agrees with the value calculated by an external tool and the error checking function yields an error of less than 10^{-6}.

3 Problem 3

Problem 3 involves finding the largest eigenvalue and its corresponding eigenvector for the following matrix

$$\begin{pmatrix} 6 & 5 & -5 \\ 2 & 6 & -2 \\ 2 & 5 & -1 \end{pmatrix}.$$

Finding the largest eigenvalue can be done relatively quickly using the power method. The power method involves multiplying a matrix M to the power of n by some starting vector $\vec{b}$ and then dividing by the product of M to the power of $n-1$ with the starting vector. In practice, one can calculate $M^n \cdot \vec{b}$ by repeatedly performing a multiplication between the matrix and a vector that results from a previous iteration of this multiplication. That is to say,

$$\vec{x}^{(n)} = M^n \cdot \vec{b} = M^{n-1} \cdot M \cdot \vec{b},$$

where $\vec{x}^{(n)}$ represents the nth iteration of the eigenvalue. After each iteration of the multiplication, it is advisable to normalize $\vec{x}$ to avoid potential divergences that may occur from repeated multiplication.

Using an error of 10^{-6} yields a solution of ≈ 6 in only eight iterations. This process if further accelerated by implementing the Aitken acceleration, a method that improves the converge of the algorithm by using information from previous iterations. The explicit form used is given by

$$\lambda_1 \approx \frac{\vec{x}_n \vec{x}_{n+2} - \vec{x}_{n+1}^2}{\vec{x}_{n+2} - 2\vec{x}_{n+1} + \vec{x}_n}$$

After a few iteratinos of the multiplication described above, $\vec{x}^{(n)}$ will also be a good approximation for the eigenvector corresponding to the calculated eigenvalue. In the case of this assignment, the normalized eigenvector was found to be

$$\lambda = \begin{pmatrix} 0.5626 \\ 0.5846 \\ 0.5846 \end{pmatrix}.$$