Math 4610 – Exam 2 Solutions

Jake Daniels A02307117

December 2, 2022

1) Explain the difference between concurrent processes and parallel processes in computing.

Response: Concurrent processes are ones where multiple things are happening at one time, but tasks are not being parallel processed, instead there is one than more processes happening at the same time. Meaning one task does not fully end before the next begins. Whereas, parallel processes are ones where the work is split up into smaller sub-tasks and done simultaneously. Concurrency increases the amount of work done at one time, where parallelism increases the computational speed.

2) From a mathematical point of view describe the difference between a data fitting problem where the size of the matrix is larger than the number of coefficients being determined or fit and the case where the size of the matrix is the same as the number coefficients to be determined. Hint: Think about determining the order of convergence from computed errors when approximating derivatives or when approximating definite integrals.

Response: For this problem lets think about the problem $A\vec{x} = \vec{y}$. For the first half let A be an m x n matrix where $m > n$, and $\vec{x}$ be be the vector for our coefficients of length n. Meaning we have n coefficients and m equations to solve. The rank of a matrix is defined to be the number of linearly independent rows or columns. If the $m \geq rank(A) > n$ then we cannot pick the coefficients such that they will fully solve our system. If $rank(A) = n$ then we can pick the coefficients such that they will be able to solve all of our rows for unique values. If $rank(A) \leq n$ then we can solve our system, but some of our coefficients will not be unique and instead defined by each other so we will have infinitely many solutions. Now lets think about if A is an n x n matrix. Then the $rank(A) \leq n$ so the same reasoning applies above and we can fully determine our system.

3) Explain the differences between direct methods and iterative methods for the solution of linear systems of equations. Use examples of direct methods like Gaussian elimination with back substitution or LU factorization on the direct solution side and Jacobi and Gauss-Seidel on the iterative method side.

Response: The main difference is that direct methods deals with a formula to find the answer while iterative methods uses a guess which converge to a approximate value. This means that direct methods give precise answer in finite steps while iterative method have infinite successive steps which improve the approximated value and are not guaranteed to give an answer in a finite amount of steps. For our eigenvalue problem Gaussian elimination with back substitution is an example of a direct method. In this method we convert a matrix to its row-echelon form and then back substitute the value we find into the row above it and so on and so forth to solve our linear system. Meaning there is no approximation involved and we can find a solution in a finite amount of steps. Whereas for an iterative method like Jacobi Iteration we approximate the solution to our system and use a residual vector to evaluate our error and see when we are close enough to the solution we want. Therefore we are not guaranteed to get an exact solution in a finite amount of steps, but we can get close.

4) Describe how the residual vector can be used to compute the error in an approximate solution of linear systems of equations. How does this compare to the exact solution and when is the residual preferable to estimating the exact error from the approximation.

Response: The residual vector is defined as the following:

$$\vec{r_k} = \vec{b} - A\vec{x_k}$$

We are trying to solve the problem $A\vec{x_k} = \vec{b}$, so the residual vector is really telling us how well $\vec{x_k}$ satisfies the equation, meaning it is able to help us calculate the error. We commonly use the norm of the residual vector to calculate the error. However, the residual vector is going to get smaller and smaller, so we run into the issue of losing accuracy when trying to compute the error, so sometimes instead of using the norm we use the dot product to compute error instead. It is convenient to use the residual vector to calculate error when we are already calculating it as part of our code, like in Jacobi iteration, but otherwise we may want to use another method.

5) How are the normal equations used in the determination of the rate of convergence of an algorithm for fitting errors?

Response: We can take our data set with the goal of fitting it to a polynomial or a linear function, $y = ax + b$. However that requires us to solve a system of equations $A\vec{x} = \vec{y}$, where A holds the data for our x-values, $\vec{x}$ is the vector for our coefficients, and $\vec{y}$ is the vector that holds the data for our y-values. A is also called a Vandermonde matrix. We can solve this system by multiplying both sides by A^T as follows:

$$A^T A\vec{x} = A^T\vec{y}$$

More specifically the normal equation is:

$$\vec{x} = (A^T A)^{-1}(A^T \vec{y})$$

This is useful because computing the transpose of a matrix is much easier than computing the inverse. Except we are still going to eventually take the inverse. For examples where we are fitting data to a linear function, $y = ax + b$ this is easy to compute since we can reduce A to be a 2 x 2 matrix so the inverse is easy to compute. When fitting to an n^{th} degree polynomial that inverse is not going to be so nice. Thinking about this in terms of error, usually we write our error term as the following:

$$e_h \leq Ch^p$$

Where p is our rate of convergence. If we take the log of both sides we can change this equation to the following:

$$log(e_h) \leq log(C) + plog(h)$$

If we define $log(e_h) = y$ and $log(h) = x$ then we just have a linear equation where p is the slope and $log(C)$ is the y-intercept. Meaning if we fit the data to a line using the normal equations we can find p the rate of convergence by finding thel slope!

6) Describe the pros and cons of using Jacobi iteration and Gauss-Seidel iteration in the solution of linear systems of equations. Your description should include terms like efficiency, accuracy, and robust.

Response: Starting with Jacobi iteration:

(a) *efficiency:* The disadvantage of Jacobi method is that, even after the modified value of a variable is evaluated in the present iteration, it is not used until the next iteration. In other words, the value of all the variables which are used in current iteration are from the previous iteration, hence increase the number of iterations to reach the exact solution. However Jacobi iteration can be coded in parelle unlike Gauss Seidel which allows it to make up for this issue.

(b) *accuracy:* We can use the residual vector to calculate the accuracy.

(c) *robust:* This method is not very robust as it only converges for certain matrices, those being matrices who are diagonally dominant and have nonzero entries on the diagonal.

Now for Gauss-Seidel:

(a) *efficiency:* This method is more efficient than Jacobi iteration as it requires less iterations to achieve an appropriate approximation for some accuracy. However, it cannot be parallelized since it is a recursion relation.

(b) *accuracy:* Same as below except our residual is a little different.

(c) *robust:* Same conditions are required as above. However, something interesting to note is that both will always converge when all eigenvalues have a magnitude less than one.

7) Describe the mathematics of the power method. That is, give a brief explanation of the mathematics behind the power method for computing the largest eigenvalue of the matrix.

Response: Consider an $n \times n$ matrix A that has n linearly independent real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and the corresponding eigenvectors $\vec{v_1}, \vec{v_2}, ..., \vec{v_n}$. Since the eigenvalues are scalars, we can rank them so that $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$ (actually, we only require $|\lambda_1| > |\lambda_2|$, other eigenvalues may be equal to each other). By definition any vector in our space can be written as a linear combination of eigenvectors. That is, we can write a vector $\vec{x_0}$ as:

$$\vec{x_0} = c_1\vec{v_1} + c_2\vec{v_2} + \cdots + c_n\vec{v_n}$$

Where $c_1 \neq 0$, if it is zero we can simply choose a different initial vector. Now we can multiply by our matrix A to get:

$$A\vec{x_0} = c_1 A\vec{v_1} + c_2 A\vec{v_2} + \cdots + c_n A\vec{v_n}$$

Recall that by definition $A\vec{v_i} = \lambda_i\vec{v_i}$ so we can rewrite the above line as:

$$A\vec{x_0} = c_1\lambda_1\vec{v_1} + c_2\lambda_2\vec{v_2} + \cdots + c_n\lambda_n\vec{v_n} = c_1\lambda_1[\vec{v_1} + \frac{c_2\lambda_2}{c_1\lambda_1}\vec{v_2} + \cdots + \frac{c_n\lambda_n}{c_1\lambda_1}\vec{v_n}] = c_1\lambda_1\vec{x_1}$$

Where $\vec{x_1} = \vec{v_1} + \frac{c_2\lambda_2}{c_1\lambda_1}\vec{v_2} + \cdots + \frac{c_n\lambda_n}{c_1\lambda_1}\vec{v_n}$. This finishes the first iteration, to start the second we can multiply by A again and after some algebra we find:

$$A\vec{x_1} = \lambda_1[\vec{v_1} + \frac{c_2\lambda_2^2}{c_1\lambda_1^2}\vec{v_2} + \cdots + \frac{c_n\lambda_n^2}{c_1\lambda_1^2}\vec{v_n}] = \lambda_1\vec{x_2}$$

We can extend this, multiplying k more times to find the following relation:

$$A\vec{x_{k+1}} = \lambda_1[\vec{v_1} + \frac{c_2\lambda_2^k}{c_1\lambda_1^k}\vec{v_2} + \cdots + \frac{c_n\lambda_n^k}{c_1\lambda_1^k}\vec{v_n}] = \lambda_1\vec{x_k}$$

Since $|\lambda_1| > |\lambda_i| \forall 1 < i \leq n, \implies \lim_{k\to\infty} \frac{\lambda_i^k}{\lambda_1^k} = 0$. Which means as we take more and more iterations $A\vec{x_{k+1}}$ will converge to $\lambda_1\vec{v_1}$, which using Rayleigh's Quotient, we can find λ_1 the largest eigenvalue!

8) What is the inverse power method for finding the smallest eigenvalue of the smallest eigenvalue of a matrix.

Response: The inverse power method uses the same idea as above except with the goal of finding the smallest eigenvalue. By definition, if λ is an eigenvalue for the matrix A, then $\frac{1}{\lambda}$ is an eigenvalue for A^{-1}, the inverse of A. So the inverse power method is the process of applying the power method to the inverse of our matrix, so that we can find the largest eigenvalue of A^{-1} which will minimize $\frac{1}{\lambda}$ and therefore be the smallest eigenvalue of our matrix A. This requires us to change some of our code for the power method. In the power method we use matrix multiplication for our iterations, but we don't want to take the inverse of A. So whenever we would have had to do that matrix multiplication, we instead change it to a problem of solving a linear system of equations. Where we use the methods talked about above, Jacobi iteration and Gauss Seidel, to do this.

9) What is the Rayleigh quotient and how is it used to approximate eigenvalues of a matrix?

Response: The Rayleigh quotient is defined as the following:

$$\lambda = \frac{\vec{v}^T A\vec{v}}{\vec{v}^T\vec{v}}$$

Where A is the matrix whose eigenvalues we want to approximate, $\vec{v}$ is an approximation for the eigenvector, and λ is our approximation of the eigenvalue of A. λ will be the largest eigenvalue when $\vec{v}$ is the corresponding eigenvector. This property is very useful for our regular and inverse power methods as we want to converge to the largest eigenvalue so all we need to do is have our approximation for the eigenvector converge to the corresponding eigenvalue.

10) Suppose you want to compute an approximate value for the eigenvalue of a matrix that is closest to a given real number, say μ. Describe how to do this using a variant of the power method.

Response: For this, we use the shifted inverse power method. Instead of looking for eigenvalues of our matrix, A, we will instead look for eigenvalues of the matrix, $A - \mu I$, where I is the identity matrix. By definition the eigenvalues of a diagonal matrix are just the diagonal entries. Meaning that the eigenvalues of μI is μ with a certain multiplicity, meaning we can rewrite the above matrix as:

$$(A - \mu I)\vec{x} = (\lambda - \mu)\vec{x}$$

Where λ is an eigenvalue of A. So if we want to find an eigenvalue that is closest to μ we can apply the inverse power method to this shifted matrix, so that the eigenvalue with the smallest magnitude of $A - \mu I$ will be the eigenvalue of A that is closest to μ, minus μ. Which the inverse power method should return and after some algebra we can return any eigenvalue of A that we want! This allows us to find all the eigenvalues of a matrix, when we choose a correct shift such that the method converges to the eigenvalue we want. Finding the correct shift is the main downside of this process.

11) What is Moore's law for computer performance? Does this still apply and how can parallel computation be used to compensate for the so-called "Power Wall" in the design of hardware?

Response: Moore's law is the observation that the number of transistors in a dense integrated circuit (IC) doubles about every two years. Moore's law is an observation and projection of a historical trend. Rather than a law of physics, it is an empirical relationship linked to gains from experience in production. This is falling apart as we have reached our limit of the amount of transistors we can put on a chip, meaning our hardware has hit its limit. Which lead us to why parallel processing is so important, as this allows us to use our existing software in a more efficient way.

12) Describe the three ways to interact with OpenMP to achieve parallelization of serial code.

Response: Directives are things that we put into our code like the pragma statement we use to tell our compiler to run a specific section of our code in parallel, and to leave the rest to run regularly. Intrinsic functions are functions created by the creators of OpenMP like the get thread number or set thread number functions. We use these in our code to manipulate OpenMP to work in our favor and perform the same task every time. Environment variables are things we can put into the console to set things like the thread number outside of our actual code. Emphasis on outside of your code.