Gaussian Elimination vs. Jacobi Iteration
Project Report

by William R. Fraser
for MCS 471 at UIC
November 21, 2008

1 Introduction

The goal of this project is to find when it is beneficial to solve a linear system of equations $Ax = b$ using the Jacobi iteration instead of using Gaussian elimination. Specifically, for what kinds of matrices can using the Jacobi iteration yield a residual equivalent to that of Gaussian elimination in fewer operations?

2 Implementation

I implemented both methods in C++ using 64-bit (double-precision) floating-point numbers. For storing the matrices, I used the boost::multi_array multidimensional array library from Boost.org. Source code was compiled with various versions of GCC (with -O3 -funsafe-loop-optimizations optimization flags), and run on 32-bit x86 and 64-bit AMD64 Linux 2.6 systems.

The Gaussian elimination routine does not use pivoting, because only diagonally-dominant matrices are used here, making pivoting unnecessary (the pivotal element is always sufficiently large because it is the largest in the row, by definition). Pivoting would also kill Gaussian elimination’s good runtime.

Rather than compute whether the Jacobi iteration will converge or not, a divergence detection scheme is employed: at each iteration, the residual is computed. If the residual does not decrease for 15 consecutive iterations, the iteration is deemed to be diverging, and is terminated, returning its results at the iteration prior to its divergence.

3 Algorithm Comparisons

3.1 Gaussian Elimination

Benefits:

1. Flexibility.
   For any system, if there is a solution, Gaussian elimination is guaranteed to find it.

2. Speed.
   Gaussian elimination’s runtime is almost exactly or slightly better than $n^3$ operations, $n$ being the dimension of the square coefficient matrix. Furthermore, if the matrix contains many zeroes or ones on the diagonal, the runtime is dramatically reduced, as it can skip large amounts of the diagonalizing process.

3. Exact.
   If pivoting is done properly, the answer Gaussian elimination generates is highly precise. In my trials, the residual was always very close to the smallest representable number in the system.

Problems:

1. Not Iterative.
   If it is desired that a rough solution be computed quickly, Gaussian elimination cannot do this. It must be allowed to run through the full algorithm, arriving at an exact answer. Interrupting it mid-computation yields a partially inverted matrix of very little value.
3.2 Jacobi Iteration

Benefits:

1. Iterative.
   The Jacobi method first generates inexact results and subsequently refines its results at each iteration, with the residuals converging at an exponential rate. For many applications, this is highly desirable.

Problems:

1. Inflexible.
   The Jacobi method only works on matrices $A$ for which $\rho(A) < 1$, or $\|A\| < 1$, or $\forall i \left( a_{ii} > \sum_{i \neq j} a_{ij} \right)$ holds. This makes it inapplicable to a large set of problems. Furthermore, determining whether a matrix satisfies the previous conditions is expensive to compute.

2. Large Set-Up Time.
   The Jacobi method cannot immediately begin producing results. Before it can begin its iteration, a matrix $-D^{-1}(L+U)$ must be computed. For large input matrices, this may not be a trivial operation, as it takes $O(n^2)$ time to perform this matrix multiplication. The result is a significant lag before any results can be output.

4 Analysis 1: Large Matrices

This test is on large (1000x1000) matrices of various kinds. All the matrices are made of random double-precision values in the range $[0, 1]$. For all matrices, the same completely random 1000x1 column vector was used as the $b$ matrix. Three classes of square matrices were generated:

1. “Completely Random”: All entries are completely random.

2. “Diagonally Dominant $x\%$ Zeroes”: Same as “Completely Random”, but the diagonal entries are set to the sum of the whole row plus a random value. Additionally, $x\%$ of the off-diagonal entries are set to zero. Various samplings of $x$ are used, with a bias towards the large end, as it is more interesting.

3. “Strictly Diagonal”: All the off-diagonal entries are zero, and the diagonal entries are completely random. This is the same as “Diagonally Dominant 100% Zeroes”.

For each matrix, first the system is solved using Gaussian elimination, which yields an operation count and the residual it was able to reach (as measured using the infinity norm operation). Then the Jacobi method is applied, using the Gaussian elimination method’s residual as a stopping point – at each iteration, the residual is computed, and if less than the residual from Gaussian elimination, the iteration is stopped. This yields an operation count, a residual, and an iteration count. These values are tabulated in the table on the following page (Table 1), except for the “Completely Random” matrix, since the Jacobi iteration diverged immediately. These values are also plotted on a linear-log graph on the following page (Figure 1).

5 Analysis 2: 3-Dimensional Plot

In this analysis, smaller matrices varying both in size and in zero density as in Analysis 1 are solved using Gaussian elimination and the Jacobi method. For this analysis, the desired residual on the Jacobi iteration is set to $1.0 \times 10^{-15}$ for all runs of the tests. Square matrices of sizes from 10 to 300 are used with a step size of 10. For each size, for zero densities between 0% and 80% a step size of 10% is used, between 80% and 98% a step size of 1% is used, and between 98% and 100% a step size of 0.1% is used. For each matrix size, the same $N$x1 column vector is used as the solution vector.

For each of the 1410 generated matrices, both Gaussian elimination and the Jacobi method are used to solve the system. This time we are only looking at the operation count as a measure of runtime. The data from these tests has been plotted as a 3-dimensional plot in Mathematica, and is on the following page (Figure 2).

6 Interpretation

As you can see from the two graphs, the Jacobi method is only better than Gaussian elimination for matrices with a large percentage of zeroes ($\sim 83.5\%$ or more).
<table>
<thead>
<tr>
<th>Matrix Type</th>
<th>Matrix Size</th>
<th>Gaussian Elimination Operations</th>
<th>Gaussian Elimination Residual</th>
<th>Jacobi Iteration Operations</th>
<th>Jacobi Iteration Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonally Dominant (0% Zeroes)</td>
<td>1000x1000</td>
<td>1001002000 (1.001002 $n^3$)</td>
<td>$3.5527136788 \times 10^{-15}$</td>
<td>31335303000 ($31.335303$ $n^3$)</td>
<td>1.1102230246 $\times 10^{-14}$ (hit divergence detector)</td>
</tr>
<tr>
<td>50% Zeroes</td>
<td>1000x1000</td>
<td>999621621 (0.999622 $n^3$)</td>
<td>$4.3298697960 \times 10^{-15}$</td>
<td>840394000 ($8.403940$ $n^3$)</td>
<td>4.3298697960 $\times 10^{-15}$ (hit divergence detector)</td>
</tr>
<tr>
<td>75% Zeroes</td>
<td>1000x1000</td>
<td>994395400 (0.994395 $n^3$)</td>
<td>$3.7747582837 \times 10^{-15}$</td>
<td>2113110000 ($2.113110$ $n^3$)</td>
<td>3.6637359813 $\times 10^{-15}$</td>
</tr>
<tr>
<td>80% Zeroes</td>
<td>1000x1000</td>
<td>995098102 (0.995098 $n^3$)</td>
<td>$4.1078251911 \times 10^{-15}$</td>
<td>1351349000 ($1.351349$ $n^3$)</td>
<td>3.9968028887 $\times 10^{-15}$</td>
</tr>
<tr>
<td>83.5% Zeroes</td>
<td>1000x1000</td>
<td>993489495 (0.993489 $n^3$)</td>
<td>$4.3298697960 \times 10^{-15}$</td>
<td>942941000 ($0.942941$ $n^3$)</td>
<td>4.2188474936 $\times 10^{-15}$</td>
</tr>
<tr>
<td>85% Zeroes</td>
<td>1000x1000</td>
<td>990142151 (0.990142 $n^3$)</td>
<td>$3.5527136788 \times 10^{-15}$</td>
<td>780779000 ($0.780779$ $n^3$)</td>
<td>3.2196467714 $\times 10^{-15}$</td>
</tr>
<tr>
<td>87.5% Zeroes</td>
<td>1000x1000</td>
<td>984468483 (0.984468 $n^3$)</td>
<td>$3.6637359813 \times 10^{-15}$</td>
<td>538537000 ($0.538537$ $n^3$)</td>
<td>3.4416913763 $\times 10^{-15}$</td>
</tr>
<tr>
<td>93.75% Zeroes</td>
<td>1000x1000</td>
<td>970514543 (0.970515 $n^3$)</td>
<td>$3.8857805862 \times 10^{-15}$</td>
<td>153152000 ($0.153152$ $n^3$)</td>
<td>3.8302694350 $\times 10^{-15}$</td>
</tr>
<tr>
<td>96.875% Zeroes</td>
<td>1000x1000</td>
<td>940012071 (0.940012 $n^3$)</td>
<td>$3.5527136788 \times 10^{-15}$</td>
<td>54053000 ($0.054053$ $n^3$)</td>
<td>2.6614994930 $\times 10^{-15}$</td>
</tr>
<tr>
<td>98.4375% Zeroes</td>
<td>1000x1000</td>
<td>884430518 (0.884404 $n^3$)</td>
<td>$4.2188474936 \times 10^{-15}$</td>
<td>28027000 ($0.028027$ $n^3$)</td>
<td>2.2204460493 $\times 10^{-15}$</td>
</tr>
<tr>
<td>99.218750% Zeroes</td>
<td>1000x1000</td>
<td>740976234 (0.740976 $n^3$)</td>
<td>$2.7755575616 \times 10^{-15}$</td>
<td>18017000 ($0.018017$ $n^3$)</td>
<td>1.6653345369 $\times 10^{-15}$</td>
</tr>
<tr>
<td>Strictly Diagonal (100% Zeroes)</td>
<td>1000x1000</td>
<td>1000000 (0.001000 $n^3$)</td>
<td>$1.1102230246 \times 10^{-16}$</td>
<td>3002000 ($0.003002$ $n^3$)</td>
<td>1.1102230246 $\times 10^{-16}$</td>
</tr>
</tbody>
</table>
Figure 1: Linear-Log Plot of 1000x1000 Matrix Data of Various Zero Density, Gaussian Elimination in Blue, Jacobi Method in Red.
Figure 2: Gaussian Elimination (blue) and Jacobi Method (red) Applied to Matrices Varying in Size and Zero Density.

The blue area is where the Jacobi method outperforms Gaussian elimination.