Homework #2 Report

John Yearsley

1. Both the upper and lower triangular solvers take $O(n^2)$ time to solve an upper and lower triangular $N \times N$ matrix respectively. This stems from the fact that each solver utilizes a nested for loop mechanism (one loop to iterate the N rows of the matrix and another loop to iterate the N columns of the matrix). While there are operations that happen within each for loop (e.g. look up value, multiply two numbers, etc.), these operations are additive. Hence, asymptotically the solvers looks like $O(n^2)$.

2.
   ```
void solve_upper_triangular(double* out, double* U, double* b, int N)
{
  for (int i=0; i<N; ++i)
    out[i] = 0;

  out[N-1] = b[N-1]/U[3*N-1];
  for (int i=N-2; i>-1; --i){
    out[i] = b[i];
    for (int j=i+1; j<N; ++j){
      out[i] -= out[j]*U[i*N + j];
    }
    out[i] = out[i]/U[i*N+i];
  }
}
```

3.
```
int gauss_seidel(double* out, double* A, double* b, int N, double epsilon)
{

  double upper_A[N*N];
  double lower_A[N*N];
  double xold[N];
  double lower_A_xold[N];
  double rhsvecdiff[N];
  double xnew[N];
  double diff[N];
  double error;
  int num_iter=0;

  for (int i=0; i<N; ++i)
    xold[i] = 0;

  for (int i=0; i<N; ++i){
    for (int j=i; j<N; ++j){
      upper_A[i*N + j] = A[i*N + j];
    }
  }

  for (int i=0; i<N*N; ++i)
    lower_A[i] = A[i] - upper_A[i];

  mat_vec(lower_A_xold, lower_A, xold, N, N);
  vec_sub(rhsvecdiff, b, lower_A_xold, N);
  solve_upper_triangular(xnew, upper_A, rhsvecdiff, N);
  vec_sub(diff, xnew, xold, N);
  error = vec_norm(diff, N);

  while(error > epsilon){
    num_iter += 1;
    for (int i=0; i<N; ++i)
      xold[i] = xnew[i];

    mat_vec(lower_A_xold, lower_A, xold, N, N);
    vec_sub(rhsvecdiff, b, lower_A_xold, N);
    solve_upper_triangular(xnew, upper_A, rhsvecdiff, N);
    vec_sub(diff, xnew, xold, N);
    error = vec_norm(diff, N);
  }

  for (int i=0; i<N; ++i)
    out[i] = xnew[i];

  return num_iter;
}
```