CME211 Project Conjugate Gradient Solver

Xuyi Long

November 25, 2022

1 Project Summary

This CG Solver is an iterative algorithm that uses Conjugate Gradient method to solve for a linear system of equation Au = b, where A is a input matrix, u is the solution vector, b is a vector of zeros. The CG solver takes a CSR matrix, a initial guess of solution vector u0, and right hand side b, and a tolerance threshold value as input parameters. The program continues to execute until the solution value converged within the specified tolerance threshold. To reduce redundant code in the program, common vector and matrix operations that are implemented in the CG algorithm are defined in matevcops.cpp file with a corresponding include file matvecops.hpp for prototypes. The section below decomposes the CG solver and provides details of each matrix and vector operation used in the algorithm.

2 CG Solver Vector Matrix Operations

2.1 double L2norm(vector)

This function calculates the norm of a input vector and returns norm value in double.

```
double L2norm(std::vector<double>  r0) {
    double norm_square = 0;
    double norm = 0;
    for (int i =0; i < r0.size(); i++) {
        norm_square = norm_square + pow(r0[i], 2);
    }
    norm = sqrt(norm_square);
    return norm;
}
```

2.2 double vect_mult(v1,v2)

The function takes two vectors and calculates the dot product.

```
double vect_mult(std::vector<double> v1, std::vector<double> v2) {
  double result =0;
  for (int i =0; i < v1.size(); i++) {
  double new_val = v1[i]*v2[i];
  result +=new_val;
  }
  return result;
}
```

2.3 vector vect_add(v1,v2);

This function takes two vectors and sums them up to get the resultant vector.

```
std::vector<double> vect_add(std::vector<double> v1, std::vector<double> v2) {
  std::vector<double> result;
  for (int i =0; i < v1.size(); i++) {
    double new_val = v1[i] + v2[i];
    result.push_back(new_val);
  }
  return result;
}
```

2.4 vector scal_vec mult(vec, num)

This function multiplies a vector to a number and returns a resultant vector.

```
std::vector<double> scal_vec_mult(std::vector<double> vec, double num) {
  std::vector<double> result;
  for (int i =0; i < vec.size(); i++) {
  double new_val = vec[i]*num;
  result.push_back(new_val);
  }
  return result;
}
```

2.5 vector mat_vec_mult(val, row_ptr, col_idx, u)

This function multiplies a CSR matrix and a solution vector and returns a resultant vector.

```
std::vector<double> mat_vec_mult(std::vector<double> &val, std::vector<int> &row_ptr,
                              std::vector<int> &col_idx, std::vector<double> &u) {

std::vector<double> result(row_ptr.size()-1,0);
  for (int i = 0; i < row_ptr.size()-1; i++) {
    for (int j = row_ptr[i]; j < row_ptr[i+1]; j++) {
      result[i] += val[j]* u[col_idx[j]];
    }
  }
  return result;
}
```