Nama : Firman Gani Heriansyah

NIM : 21120122130043

Penjabaran Dekomposisi Crout

Link github :

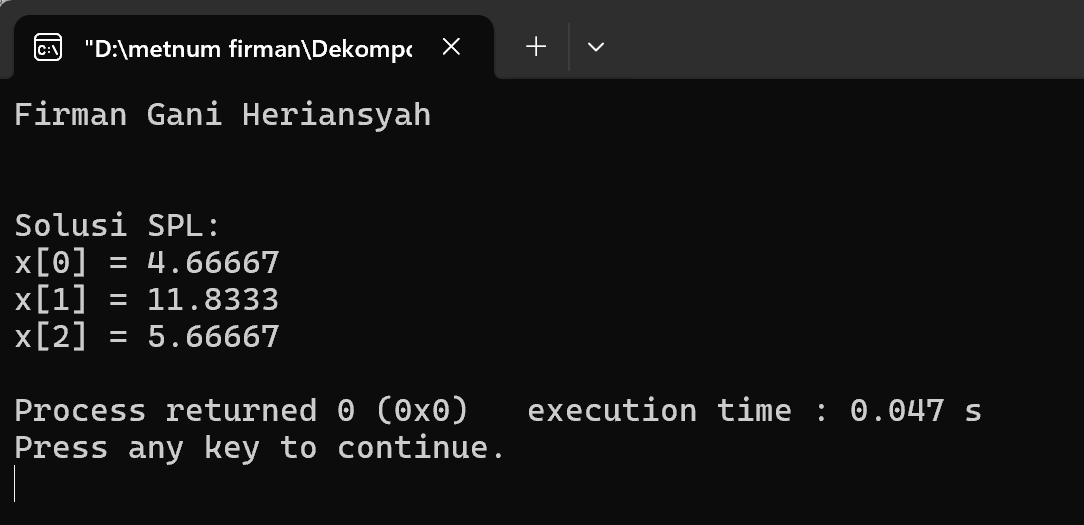
|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  // Fungsi untuk melakukan dekomposisi Crout  void croutDecomposition(vector<vector<double>>& A, vector<vector<double>>& L, vector<vector<double>>& U) {  int n = A.size();  cout <<"Firman Gani Heriansyah"<<endl;  cout <<"\n"<<endl;  // Inisialisasi matriks L dan U dengan nilai 0  L = vector<vector<double>>(n, vector<double>(n, 0));  U = vector<vector<double>>(n, vector<double>(n, 0));  // Proses dekomposisi Crout  for (int i = 0; i < n; i++) {  // Mengisi diagonal utama matriks U  U[i][i] = 1;  // Menghitung elemen-elemen matriks L dan U  for (int j = i; j < n; j++) {  double sum = 0;  for (int k = 0; k < i; k++) {  sum += L[j][k] \* U[k][i];  }  L[j][i] = A[j][i] - sum;  }  for (int j = i + 1; j < n; j++) {  double sum = 0;  for (int k = 0; k < i; k++) {  sum += L[i][k] \* U[k][j];  }  U[i][j] = (A[i][j] - sum) / L[i][i];  }  }  }  // Fungsi untuk menyelesaikan SPL dengan dekomposisi Crout  vector<double> solveSPL(vector<vector<double>>& A, vector<double>& b) {  int n = A.size();  vector<vector<double>> L, U;  // Melakukan dekomposisi Crout  croutDecomposition(A, L, U);  // Penyelesaian SPL dengan dekomposisi Crout  vector<double> y(n, 0), x(n, 0);  // Solusi y dari Ly = b  for (int i = 0; i < n; i++) {  double sum = 0;  for (int j = 0; j < i; j++) {  sum += L[i][j] \* y[j];  }  y[i] = (b[i] - sum) / L[i][i];  }  // Solusi x dari Ux = y  for (int i = n - 1; i >= 0; i--) {  double sum = 0;  for (int j = i + 1; j < n; j++) {  sum += U[i][j] \* x[j];  }  x[i] = (y[i] - sum) / U[i][i];  }  return x;  } |

Fungsi croutDecomposition digunakan untuk melakukan dekomposisi matriks koefisien menjadi matriks lower triangular (L) dan upper triangular (U). Kemudian, fungsi solveSPL menggunakan hasil dekomposisi untuk menyelesaikan SPL dengan matriks L dan U yang telah dihasilkan. Prosesnya terdiri dari dua tahap: pertama, menghitung solusi 𝑦 dari matriks lower triangular menggunakan substitusi maju, dan kedua, menghitung solusi 𝑥 dari matriks upper triangular menggunakan substitusi mundur. Hasil akhirnya adalah solusi SPL yang diberikan dalam bentuk vector 𝑥.

|  |
| --- |
| //-----Kode Testing-----//  int main() {  // Contoh SPL  vector<vector<double>> A = {{6, -4, 5}, {-4, 2, -3}, {1, -2, 6}};  vector<double> b = {9, -12, 15};  // Menyelesaikan SPL dengan dekomposisi Crout  vector<double> x = solveSPL(A, b);  // Menampilkan solusi  cout << "Solusi SPL:" << endl;  for (int i = 0; i < x.size(); i++) {  cout << "x[" << i << "] = " << x[i] << endl;  }  return 0;  } |

Pada kode testing tersebut, matriks koefisien 𝐴 dari SPL dan vektor 𝑏 yang berisi konstanta pada sisi kanan persamaan diinisialisasi dengan nilai-nilai tertentu. Kemudian, fungsi solveSPL() dipanggil dengan parameter matriks 𝐴 dan vektor 𝑏 untuk mendapatkan solusi SPL. Fungsi solveSPL() akan menghitung solusi SPL dan mengembalikan vektor solusi 𝑥. Setelah itu, program akan mencetak solusi-solusi tersebut ke layar dengan menggunakan loop for. Terakhir, program mencetak teks "Solusi SPL:" diikuti dengan nilai-nilai solusi dari SPL ke layar.

Hasil Output



Link Github : <https://github.com/Frmngh/Metodenumerik>