Zadanie Numeryczne 6

Zadanie polegało na rozwiązaniu układu równań Ax = b metoda gradientów sprzężonych.

$$A = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}$$

Program jest napisany w języku c++ aby skompilować i uruchomić można skorzystać z komendy

g++ ZN6.cpp

Metoda gradientów sprzężonych jest metoda iteracyjną, dokładny wynik udało się uzyskać już po drugiej iteracji. Rozwiązaniem układu jest wektor:

$$x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Metodę gradientów sprzężonych można stosować do macierzy symetrycznych i dodatnio określonych. Algorytm dla metody gradientów sprzężonych polega na obliczeniu wartości α, przybliżonej wartości x oraz dwóch wektorów p i r.

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k},$$
$$r_{k+1} = r_k - \alpha_k A p_k,$$
$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k},$$
$$p_{k+1} = r_{k+1} + \beta_k p_k.$$

Początkowe wartości $p_{0=}r_0$, $r_0 = b - Ax_0$, gdzie x_0 jest odgadniętym wektorem x ja zastosowałem wektor zerowy. Algorytm zaczynamy od obliczenia wartości α, następnie liczymy pierwsze przybliżenie x. Aby dostać dokładniejszy wynik obliczany wektor reszty r, wartość parametru β oraz następny wektor kierunki poszukiwań p. Następne iteracje zaczynamy od policzenia wartości parametru α już z nowymi wartościami p i r. Kolejne iteracje dają nam coraz lepsze przybliżenie rozwiązania.

```cpp
#include <iostream>

using namespace std;
int main()
{

	double A[3][3] = { 4, -1, 0,
	                                -1, 4, -1,
	                                 0, -1, 4 };

	double b[3] = { 2, 6 , 2 };
	double x[3] = { 0,0,0 };
	double r[3] = { 2,6,2 };
	double p[3] = { 2,6,2 };
	double rr[3];
	double pp[3];
	double alfa;
	double beta;

	int t = 1;
	int T = 2;

	//wykonuje obliczenia interacyjnie o zadana ilosc razy
	while (t <= T) {

		alfa = (r[0] * r[0] + r[1] * r[1] + r[2] * r[2]) /
			((p[0] * A[0][0] + p[1] * A[0][1] + p[2] * A[0][2]) * p[0] +
			(p[0] * A[1][0] + p[1] * A[1][1] + p[2] * A[1][2]) * p[1] +
			(p[0] * A[2][0] + p[1] * A[2][1] + p[2] * A[2][2]) * p[2]);

		for (int i = 0; i < 3; i++) {
			x[i] = x[i] + alfa * p[i];
		}

		for (int i = 0; i < 3; i++) {
			rr[i] = r[i] - ((A[0][i] * p[0] + A[1][i] * p[1] + A[2][i] * p[2])
			* alfa);
		}

		beta = (rr[0] * rr[0] + rr[1] * rr[1] + rr[2] * rr[2]) /
			(r[0] * r[0] + r[1] * r[1] + r[2] * r[2]);

		for (int i = 0; i < 3; i++) {
			p[i] = rr[i] + beta * p[i];

			r[i] = rr[i];
		}


		t++;
		}

for (int i = 0; i < 3; i++) {
			cout << x[i] << endl;

		}
return 0;
}
```