Metody Numeryczne – Zadanie NUM5

Bartosz Bochniak

Wstęp

Polecenie

Rozwiąż układ równań

$$\begin{pmatrix} d & 0.5 & 0.1 & & & & \\ 0.5 & d & 0.5 & 0.1 & & & \\ 0.1 & 0.5 & d & 0.5 & 0.1 & & \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & 0.1 & 0.5 & d & 0.5 \\ & & & & 0.1 & 0.5 & d \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ N-1 \\ N \end{pmatrix}$$

dla *N* = 200 za pomocą metod Jacobiego i Gaussa-Seidela, gdzie *d* jest elementem diagonalnym. Dla różnych wartości *d* i punktów startowych przedstaw graficznie różnicę pomiędzy dokładnym rozwiązaniem a jego przybliżeniami w kolejnych iteracjach. Odpowiednio dobierając zakres parametrów, porównaj dwie metody. Czy procedura iteracyjna zawsze jest zbieżna?

Rozwiązanie

Metody Jacobiego oraz Gaussa-Seidela, są metodami iteracyjnymi, co oznacza, że wynik „dokładny” jest otrzymywany dla ilości kroków dążącej do nieskończoności. Ponieważ jest to niepraktyczne, metoda iteracyjna jest dobra wtedy, jeżeli wynik osiągnięty po niewielkiej ilości kroków zbliży się do wyniku ścisłego w granicach błędu zaokrąglenia.

Z tego typu metod korzysta się najczęściej, gdy zastosowanie faktoryzacji prowadziłoby do wypełnienia macierzy rzadkiej.

Wzór Jacobiego

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{N} a_{ij}\, x_j^{(k)}}{a_{ii}}$$

Wzór Gaussa-Seidela

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{N} a_{ij}\, x_j^{(k)}}{a_{ii}}$$

...gdzie k reprezentuje k-tą iterację algorytmu. ($a_{ii} = d$)

Macierz zadana w poleceniu zadania jest macierzą z pięcioma diagonalami, czyli jest macierzą rzadką dla dużych rozmiarów *N*. Uwzględniając to do obu wzorów przy implementacji, uniknięte zostaje wiele redundantnych obliczeń związanych z mnożeniem przez zero, co znacznie przyspiesza czas działania algorytmu.

Zbieżność metod iteracyjnych

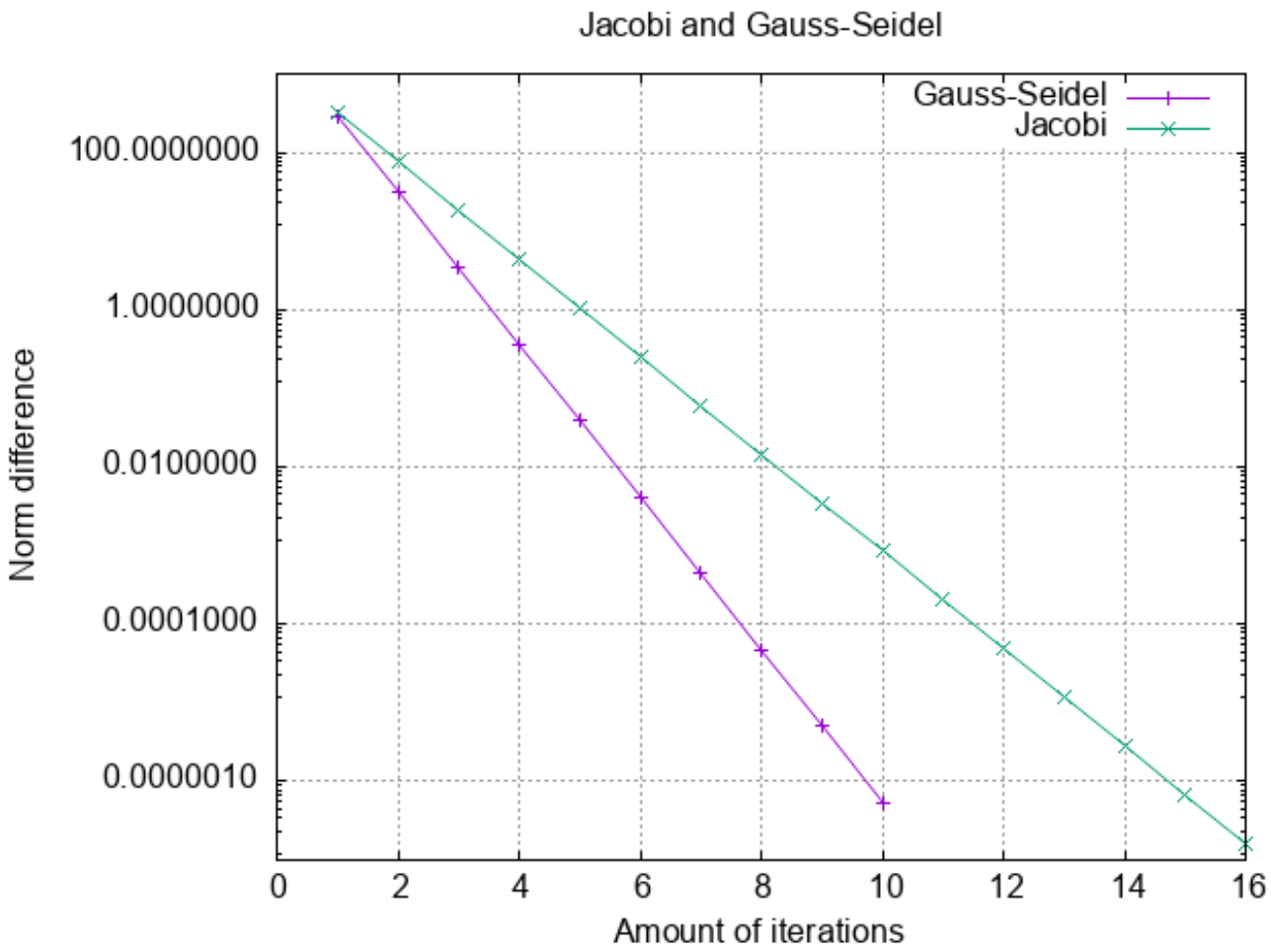
Obie metody iteracyjne są zbieżne pod warunkiem, że macierz jest **silnie diagonalnie dominująca**, co oznacza, że:

$$|d| > \forall_i \sum_{i \neq j} |a_{ij}|$$

Metoda Gaussa-Seidela natomiast jest także zbieżna, gdy macierz jest symetryczna oraz dodatnio określona.

Wzór Gaussa-Seidela jest analogiczny do wzoru Jacobiego, aczkolwiek korzysta on także z „najnowszych" przybliżeń, co czyni go skuteczniejszym. Wynika z tego, że metoda Gaussa-Seidela powinna osiągnąć dokładniejszą zbieżność z wynikiem niż metoda Jacobiego, czego dowodem jest wykres poniżej:

$$d = 5.0 \qquad x_0 = (0, \dots, 0)^T$$



Przykładowe wyniki

Poniżej podane zostaną wyniki dla czterech różnych punktów startowych per jedną wartość *d*, dla trzech wartości *d*. Wszystkie dane będą notowane nad wynikami oraz wykresami zbieżności algorytmu, natomiast pod koniec czterech sekcji wyników zostanie podany wynik dokładny uzyskany za pomocą biblioteki Eigen. Każdy wykres został utworzony za pomocą aplikacji gnuplot.

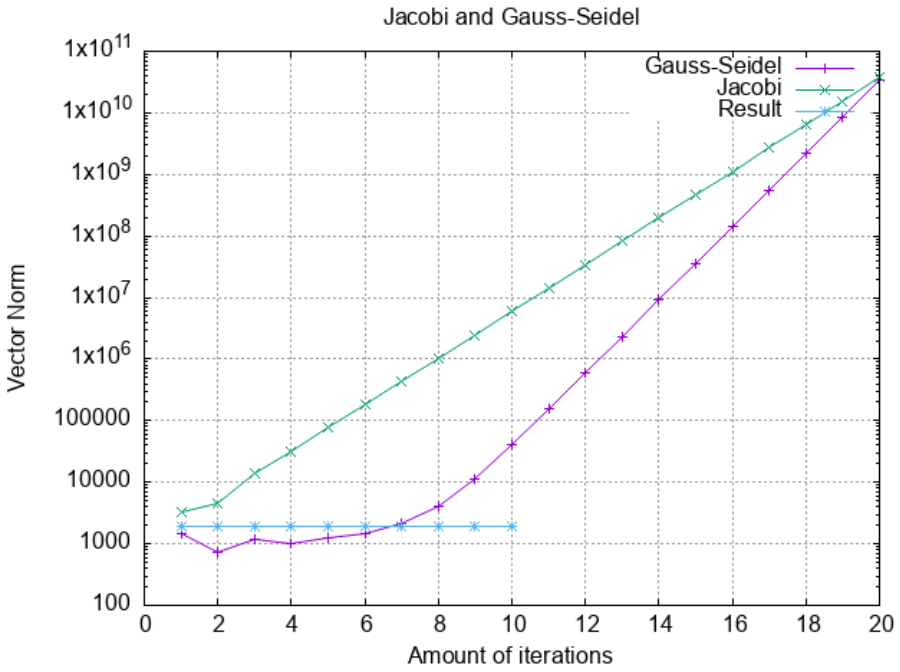
$$d = 0.5 \qquad x_0 = (0, \dots, 0)^T$$

```
Algorytm Jacobiego dla tej macierzy (o d = 0.5) jest rozbiezny. Wywolujemy 20 iteracji.
Punkt startowy dla powyzszego algorytmu: wektor z samymi 0.
Wynik:

(-2.65939e+07, -4.93243e+07, -7.26499e+07, -9.59313e+07, -1.19271e+08, -1.42657e+08, -1.66092e+08, -1.89573e+08, -2.13094e+08, -2.36649e+08,
-2.60231e+08, -2.83834e+08, -3.07452e+08, -3.31081e+08, -3.54718e+08, -3.78359e+08, -4.02002e+08, -4.25647e+08, -4.49294e+08, -4.7294e+08,
-4.96587e+08, -5.20234e+08, -5.43881e+08, -5.67528e+08, -5.91175e+08, -6.14822e+08, -6.38469e+08, -6.62116e+08, -6.85763e+08, -7.0941e+08,
-7.33057e+08, -7.56704e+08, -7.80351e+08, -8.03998e+08, -8.27645e+08, -8.51292e+08, -8.74939e+08, -8.98586e+08, -9.22233e+08, -9.4588e+08,
-9.69527e+08, -9.93174e+08, -1.01682e+09, -1.04047e+09, -1.06411e+09, -1.08776e+09, -1.11141e+09, -1.13506e+09, -1.1587e+09, -1.18235e+09,
-1.206e+09, -1.22964e+09, -1.25329e+09, -1.27694e+09, -1.30058e+09, -1.32423e+09, -1.34788e+09, -1.37153e+09, -1.39517e+09, -1.41882e+09,
-1.44247e+09, -1.46611e+09, -1.48976e+09, -1.51341e+09, -1.53705e+09, -1.5607e+09, -1.58435e+09, -1.608e+09, -1.63164e+09, -1.65529e+09,
-1.67894e+09, -1.70258e+09, -1.72623e+09, -1.74988e+09, -1.77352e+09, -1.79717e+09, -1.82082e+09, -1.84447e+09, -1.86811e+09, -1.89176e+09,
-1.91541e+09, -1.93905e+09, -1.9627e+09, -1.98635e+09, -2.00999e+09, -2.03364e+09, -2.05729e+09, -2.08094e+09, -2.10458e+09, -2.12823e+09,
-2.15188e+09, -2.17552e+09, -2.19917e+09, -2.22282e+09, -2.24646e+09, -2.27011e+09, -2.29376e+09, -2.31741e+09, -2.34105e+09, -2.3647e+09,
-2.38835e+09, -2.41199e+09, -2.43564e+09, -2.45929e+09, -2.48293e+09, -2.50658e+09, -2.53023e+09, -2.55388e+09, -2.57752e+09, -2.60117e+09,
-2.62482e+09, -2.64846e+09, -2.67211e+09, -2.69576e+09, -2.7194e+09, -2.74305e+09, -2.7667e+09, -2.79035e+09, -2.81399e+09, -2.83764e+09,
-2.86129e+09, -2.88493e+09, -2.90858e+09, -2.93223e+09, -2.95587e+09, -2.97952e+09, -3.00317e+09, -3.02681e+09, -3.05046e+09, -3.07411e+09,
-3.09776e+09, -3.1214e+09, -3.14505e+09, -3.1687e+09, -3.19234e+09, -3.21599e+09, -3.23964e+09, -3.26328e+09, -3.28693e+09, -3.31058e+09,
-3.33423e+09, -3.35787e+09, -3.38152e+09, -3.40517e+09, -3.42881e+09, -3.45246e+09, -3.47611e+09, -3.49975e+09, -3.5234e+09, -3.54705e+09,
-3.5707e+09, -3.59434e+09, -3.61799e+09, -3.64164e+09, -3.66528e+09, -3.68893e+09, -3.71258e+09, -3.73622e+09, -3.75987e+09, -3.78352e+09,
-3.80717e+09, -3.83081e+09, -3.85446e+09, -3.87811e+09, -3.90175e+09, -3.9254e+09, -3.94905e+09, -3.97269e+09, -3.99634e+09, -4.01999e+09,
-4.04364e+09, -4.06728e+09, -4.09093e+09, -4.11458e+09, -4.13822e+09, -4.16187e+09, -4.18551e+09, -4.20914e+09, -4.23274e+09, -4.25626e+09,
-4.2796e+09, -4.30257e+09, -4.32477e+09, -4.34553e+09, -4.36364e+09, -4.37724e+09, -4.38345e+09, -4.37811e+09, -4.35561e+09, -4.30866e+09,
-4.22849e+09, -4.10517e+09, -3.92827e+09, -3.68801e+09, -3.37646e+09, -2.98873e+09, -2.52477e+09, -1.98948e+09, -1.39131e+09, -7.64402e+08)
```

```
Algorytm Gaussa-Seidela dla tej macierzy (o d = 0.5) jest rozbiezny. Wywolujemy 20 iteracji.
Punkt startowy dla powyzszego algorytmu: wektor z samymi 0.
Wynik:

(438.873, -9017.74, 45274, -151700, 410032, -963324, 2.04412e+06, -4.00756e+06, 7.36807e+06, -1.28363e+07,
2.13533e+07, -3.41171e+07, 5.25984e+07, -7.85395e+07, 1.13936e+08, -1.60997e+08, 2.22084e+08, -2.99628e+08, 3.96036e+08, -5.13573e+08,
6.54254e+08, -8.19714e+08, 1.01111e+09, -1.229e+09, 1.4733e+09, -1.7432e+09, 2.03714e+09, -2.35282e+09, 2.68723e+09, -3.03673e+09,
3.39711e+09, -3.7637e+09, 4.13153e+09, -4.49543e+09, 4.85019e+09, -5.19068e+09, 5.51203e+09, -5.8097e+09, 6.07962e+09, -6.31827e+09,
6.52274e+09, -6.69079e+09, 6.82087e+09, -6.91211e+09, 6.96434e+09, -6.978e+09, 6.95416e+09, -6.89442e+09, 6.80082e+09, -6.67583e+09,
6.52221e+09, -6.34297e+09, 6.14129e+09, -5.92041e+09, 5.6836e+09, -5.4341e+09, 5.17505e+09, -4.90943e+09, 4.64007e+09, -4.36956e+09,
4.10027e+09, -3.83435e+09, 3.57365e+09, -3.31979e+09, 3.07416e+09, -2.83786e+09, 2.6118e+09, -2.39665e+09, 2.19288e+09, -2.0008e+09,
1.82052e+09, -1.65204e+09, 1.49523e+09, -1.34983e+09, 1.21552e+09, -1.0919e+09, 9.78494e+08, -8.74815e+08, 7.8033e+08, -6.94487e+08,
6.16731e+08, -5.46503e+08, 4.83252e+08, -4.26439e+08, 3.75545e+08, -3.3007e+08, 2.89538e+08, -2.535e+08, 2.21534e+08, -1.93243e+08,
1.68263e+08, -1.46254e+08, 1.26904e+08, -1.09927e+08, 9.50629e+07, -8.2074e+07, 7.07462e+07, -6.08856e+07, 5.23183e+07, -4.48879e+07,
3.84555e+07, -3.28961e+07, 2.81001e+07, -2.3969e+07, 2.04168e+07, -1.73671e+07, 1.47531e+07, -1.25157e+07, 1.0604e+07, -8.97256e+06,
7.58272e+06, -6.39997e+06, 5.39532e+06, -4.54267e+06, 3.82045e+06, -3.20908e+06, 2.69267e+06, -2.25657e+06, 1.88922e+06, -1.5797e+06,
1.3197e+06, -1.10108e+06, 917946, -764237, 635863, -528273, 438701, -363710, 301500, -249438,
206428, -170419, 140822, -116000, 95730.6, -78669.3, 64858.7, -53159, 43802.2, -35790.8,
29493.2, -24008.6, 19804.1, -16043.8, 13266.1, -10677.8, 8869.51, -7074.46, 5922.68, -4662.54,
3954.02, -3053.04, 2643.05, -1982.16, 1772.83, -1271.65, 1197.02, -801.483, 817.25, -491.1,
567.621, -286.629, 404.125, -152.155, 297.472, -63.8073, 228.224, -5.77197, 183.524, 32.4013,
154.899, 57.6022, 136.72, 74.163, 125.353, 86.2908, 119.183, 89.0976, 112.49, 120.024,
103.62, 11.1939, 268.329, 277.315, -840.817, 1075.76, 2122.33, -7731.73, 9018.88, 9360.43,
-52401.5, 90810.4, -44721.3, -169021, 530594, -818724, 698208, -59362.5, -640721, 652993)
```



Wyniki dla macierz o d = 0.5 są zdecydowanie rozbieżne.

Przeprowadzone testy dla $x_0' = (1, \dots, 1)^T; x_0'' = b = (1, 2, \dots, N)^T; x_0''' = (rand(), \dots)^T$ dają podobne rezultaty jeżeli chodzi o zbieżność. Wyniki są obarczone gigantycznymi błędami.

```
Wynik za pomoca biblioteki Eigen:

(141.248, -131.718, -37.6535, 160.614, -67.4623, -105.773, 150.761, 11.7572, -146.264, 104.501,
89.2698, -149.289, 33.8574, 146.309, -113.655, -43.2666, 169.207, -47.7365, -107.365, 152.74,
32.6036, -142.153, 101.414, 107.936, -138.592, 28.3581, 160.071, -97.1255, -47.9282, 176.554,
-27.5712, -108.146, 153.744, 53.3092, -136.968, 97.7195, 125.954, -126.833, 22.7684, 172.836,
-79.8097, -52.1399, 182.8, -7.08897, -108.047, 153.819, 73.7844, -130.693, 93.495, 143.25,
-114.042, 17.1799, 184.567, -61.7746, -55.8197, 187.957, 13.6215, -107.018, 153.017, 93.9401,
-123.318, 88.8229, 159.758, -100.254, 11.6842, 195.237, -43.0921, -58.8893, 192.04, 34.4698,
-105.011, 151.395, 113.69, -114.841, 83.7884, 175.417, -85.5114, 6.37232, 204.823, -23.8386,
-61.2744, 195.075, 55.3645, -101.989, 149.019, 132.95, -105.268, 78.4793, 190.171, -69.8635,
1.33386, 213.314, -4.09436, -62.9056, 197.094, 76.2139, -97.9186, 145.958, 151.64, -94.6115,
72.9854, 203.971, -53.3656, -3.34333, 220.703, 16.0569, -63.7187, 198.136, 96.9266, -92.7757,
142.285, 169.684, -82.8935, 67.3975, 216.775, -36.079, -7.57395, 226.992, 36.5285, -63.6551,
198.246, 117.412, -86.5429, 138.08, 187.009, -70.1416, 61.8072, 228.547, -18.0704, -11.2758,
232.19, 57.2319, -62.6624, 197.476, 137.582, -79.2102, 133.423, 203.549, -56.3913, 56.3063,
239.258, 0.588537, -14.3703, 236.315, 78.0765, -60.6944, 195.885, 157.349, -70.7756, 128.401,
219.242, -41.6847, 50.9856, 248.886, 19.8215, -16.7831, 239.39, 98.9711, -57.712, 193.537,
176.63, -61.2444, 123.101, 234.032, -26.0707, 45.9349, 257.42, 39.5484, -18.4447, 241.448,
119.824, -53.6827, 190.501, 195.343, -50.6298, 117.612, 247.87, -9.60454, 41.2422, 264.851,
59.6855, -19.2905, 242.526, 140.544, -48.5816, 186.85, 213.413, -38.9523, 112.026, 260.713,
7.65286, 36.9927, 271.183, 80.145, -19.2558, 242.647, 161.137, -42.7842, 184.258, 224.298)
```

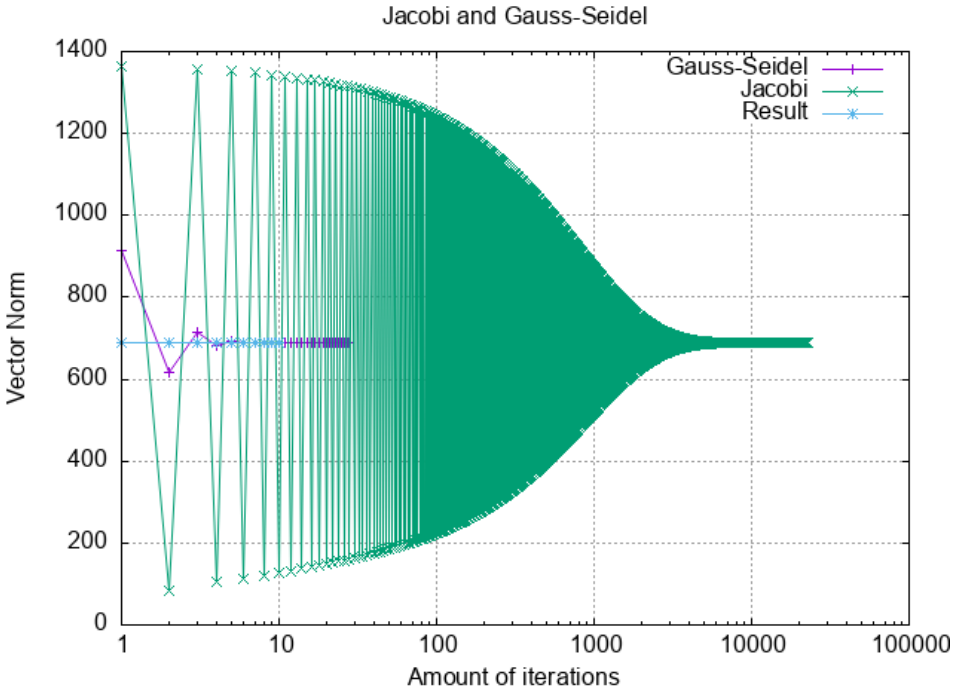
$$d = 1.201 \qquad x_0 = (0, \dots, 0)^T$$

```
Zbieznosc po iteracji dla Jacobiego dla d = (1.201): 22735
Punkt startowy dla powyzszego algorytmu: wektor z samymi 0.
Wynik:

(0.374034, 0.852627, 1.24472, 1.66617, 2.08286, 2.49876, 2.91551, 3.33194, 3.74843, 4.16493,
4.58142, 4.99792, 5.41441, 5.8309, 6.2474, 6.66389, 7.08038, 7.49688, 7.91337, 8.32986,
8.74636, 9.16285, 9.57934, 9.99584, 10.4123, 10.8288, 11.2453, 11.6618, 12.0783, 12.4948,
12.9113, 13.3278, 13.7443, 14.1608, 14.5773, 14.9938, 15.4102, 15.8267, 16.2432, 16.6597,
17.0762, 17.4927, 17.9092, 18.3257, 18.7422, 19.1587, 19.5752, 19.9917, 20.4082, 20.8247,
21.2411, 21.6576, 22.0741, 22.4906, 22.9071, 23.3236, 23.7401, 24.1566, 24.5731, 24.9896,
25.4061, 25.8226, 26.2391, 26.6556, 27.0721, 27.4885, 27.905, 28.3215, 28.738, 29.1545,
29.571, 29.9875, 30.404, 30.8205, 31.237, 31.6535, 32.07, 32.4865, 32.903, 33.3195,
33.7359, 34.1524, 34.5689, 34.9854, 35.4019, 35.8184, 36.2349, 36.6514, 37.0679, 37.4844,
37.9009, 38.3174, 38.7339, 39.1504, 39.5668, 39.9833, 40.3998, 40.8163, 41.2328, 41.6493,
42.0658, 42.4823, 42.8988, 43.3153, 43.7318, 44.1483, 44.5648, 44.9813, 45.3978, 45.8142,
46.2307, 46.6472, 47.0637, 47.4802, 47.8967, 48.3132, 48.7297, 49.1462, 49.5627, 49.9792,
50.3957, 50.8122, 51.2287, 51.6451, 52.0616, 52.4781, 52.8946, 53.3111, 53.7276, 54.1441,
54.5606, 54.9771, 55.3936, 55.8101, 56.2266, 56.6431, 57.0596, 57.4761, 57.8925, 58.309,
58.7255, 59.142, 59.5585, 59.975, 60.3915, 60.808, 61.2245, 61.641, 62.0575, 62.474,
62.8905, 63.307, 63.7234, 64.1399, 64.5564, 64.9729, 65.3894, 65.8059, 66.2224, 66.6389,
67.0554, 67.4719, 67.8884, 68.3049, 68.7214, 69.1379, 69.5544, 69.9708, 70.3873, 70.8038,
71.2203, 71.6368, 72.0533, 72.4698, 72.8863, 73.3028, 73.7193, 74.1358, 74.5523, 74.9688,
75.3853, 75.8017, 76.2182, 76.6347, 77.0512, 77.4677, 77.8842, 78.3007, 78.7172, 79.1339,
79.5486, 79.9715, 80.377, 80.7808, 81.3623, 81.1547, 82.7847, 83.8168, 69.5365, 130.6)
```

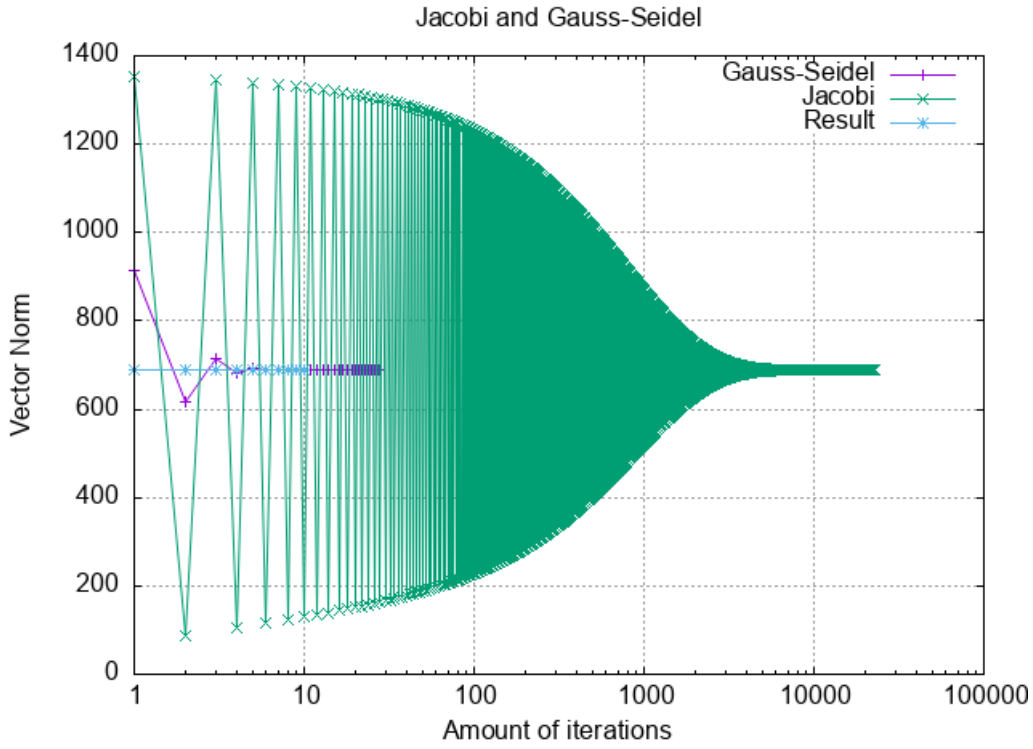
```
Zbieznosc po iteracji dla Gaussa-Seidela dla d = (1.201): 28
Punkt startowy dla powyzszego algorytmu: wektor z samymi 0.
Wynik:

(0.374034, 0.852627, 1.24472, 1.66617, 2.08286, 2.49876, 2.91551, 3.33194, 3.74843, 4.16493,
4.58142, 4.99792, 5.41441, 5.8309, 6.2474, 6.66389, 7.08038, 7.49688, 7.91337, 8.32986,
8.74636, 9.16285, 9.57934, 9.99584, 10.4123, 10.8288, 11.2453, 11.6618, 12.0783, 12.4948,
12.9113, 13.3278, 13.7443, 14.1608, 14.5773, 14.9938, 15.4102, 15.8267, 16.2432, 16.6597,
17.0762, 17.4927, 17.9092, 18.3257, 18.7422, 19.1587, 19.5752, 19.9917, 20.4082, 20.8247,
21.2411, 21.6576, 22.0741, 22.4906, 22.9071, 23.3236, 23.7401, 24.1566, 24.5731, 24.9896,
25.4061, 25.8226, 26.2391, 26.6556, 27.0721, 27.4885, 27.905, 28.3215, 28.738, 29.1545,
29.571, 29.9875, 30.404, 30.8205, 31.237, 31.6535, 32.07, 32.4865, 32.903, 33.3195,
33.7359, 34.1524, 34.5689, 34.9854, 35.4019, 35.8184, 36.2349, 36.6514, 37.0679, 37.4844,
37.9009, 38.3174, 38.7339, 39.1504, 39.5668, 39.9833, 40.3998, 40.8163, 41.2328, 41.6493,
42.0658, 42.4823, 42.8988, 43.3153, 43.7318, 44.1483, 44.5648, 44.9813, 45.3978, 45.8142,
46.2307, 46.6472, 47.0637, 47.4802, 47.8967, 48.3132, 48.7297, 49.1462, 49.5627, 49.9792,
50.3957, 50.8122, 51.2287, 51.6451, 52.0616, 52.4781, 52.8946, 53.3111, 53.7276, 54.1441,
54.5606, 54.9771, 55.3936, 55.8101, 56.2266, 56.6431, 57.0596, 57.4761, 57.8925, 58.309,
58.7255, 59.142, 59.5585, 59.975, 60.3915, 60.808, 61.2245, 61.641, 62.0575, 62.474,
62.8905, 63.307, 63.7234, 64.1399, 64.5564, 64.9729, 65.3894, 65.8059, 66.2224, 66.6389,
67.0554, 67.4719, 67.8884, 68.3049, 68.7214, 69.1379, 69.5544, 69.9708, 70.3873, 70.8038,
71.2203, 71.6368, 72.0533, 72.4698, 72.8863, 73.3028, 73.7193, 74.1358, 74.5523, 74.9688,
75.3853, 75.8017, 76.2182, 76.6347, 77.0512, 77.4677, 77.8842, 78.3007, 78.7172, 79.1339,
79.5486, 79.9715, 80.377, 80.7808, 81.3623, 81.1547, 82.7847, 83.8168, 69.5365, 130.6)
```

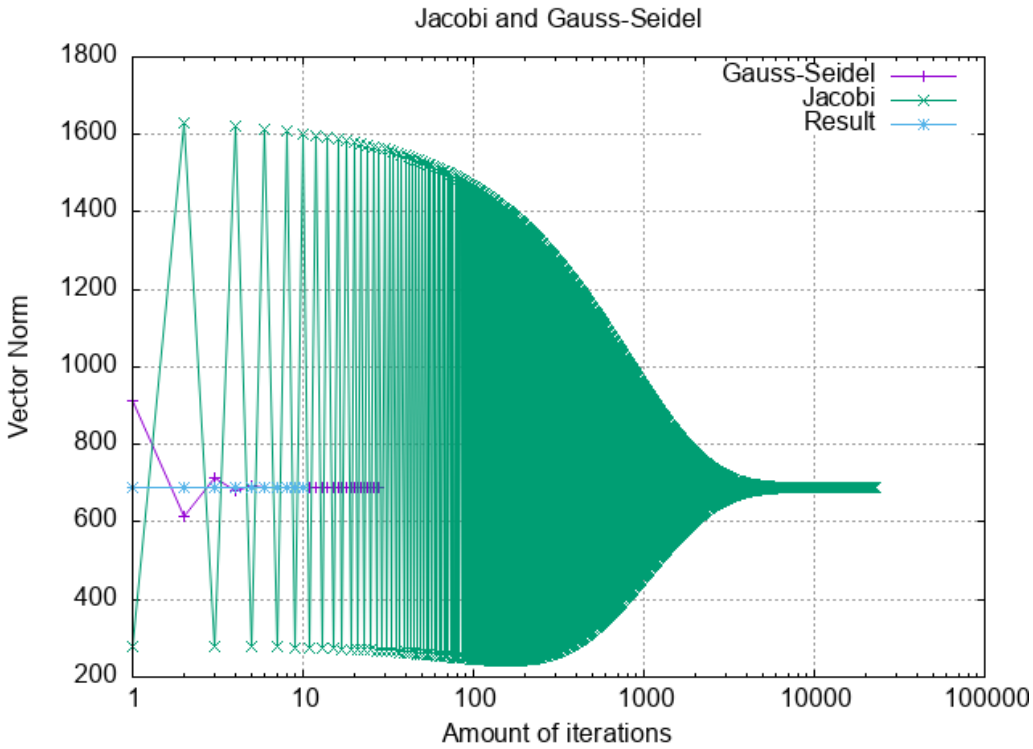


Dla każdego następnego x_0 dla d = 1.201 wyniki są takie same, dlatego dla oszczędzenia miejsca, pozostaną umieszczone tylko wykresy zbieżności od ilości iteracji oraz dokładny wynik od Eigen.

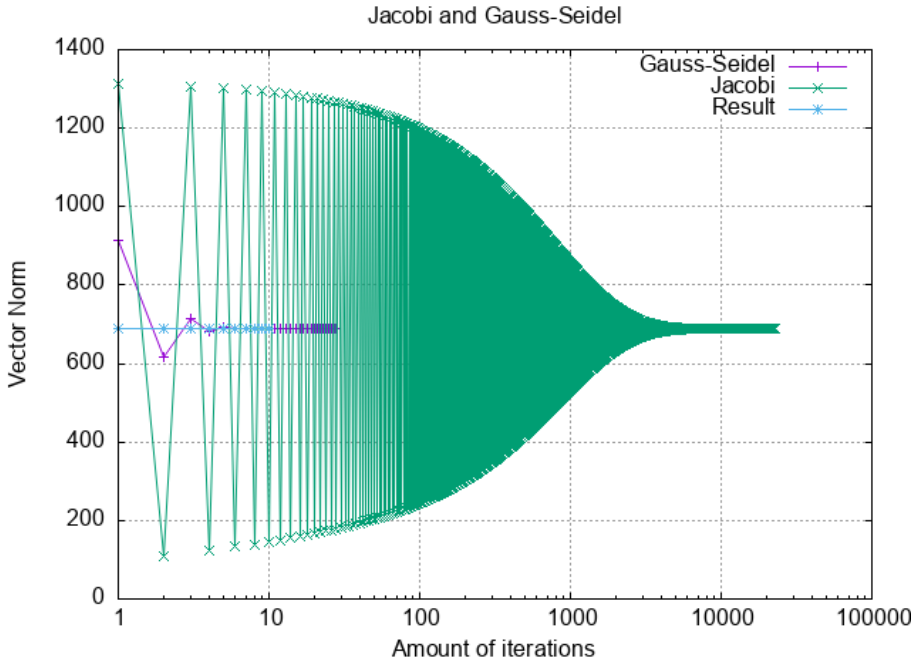
$$d = 1.201 \qquad x_0 = (1, \dots, 1)^T$$



$$d = 1.201 \qquad x_0 = b = (1, 2, \dots, N)^T$$



$$d = 1.201 \qquad x_0 = (rand(\,), \ldots)^T$$



Wynik za pomocą biblioteki Eigen:

```
Wynik za pomoca biblioteki Eigen:

(0.374034, 0.852627, 1.24472, 1.66617, 2.08286, 2.49876, 2.91551, 3.33194, 3.74843, 4.16493,
4.58142, 4.99792, 5.41441, 5.8309, 6.2474, 6.66389, 7.08038, 7.49688, 7.91337, 8.32986,
8.74636, 9.16285, 9.57934, 9.99584, 10.4123, 10.8288, 11.2453, 11.6618, 12.0783, 12.4948,
12.9113, 13.3278, 13.7443, 14.1608, 14.5773, 14.9938, 15.4102, 15.8267, 16.2432, 16.6597,
17.0762, 17.4927, 17.9092, 18.3257, 18.7422, 19.1587, 19.5752, 19.9917, 20.4082, 20.8247,
21.2411, 21.6576, 22.0741, 22.4906, 22.9071, 23.3236, 23.7401, 24.1566, 24.5731, 24.9896,
25.4061, 25.8226, 26.2391, 26.6556, 27.0721, 27.4885, 27.905, 28.3215, 28.738, 29.1545,
29.571, 29.9875, 30.404, 30.8205, 31.237, 31.6535, 32.07, 32.4865, 32.903, 33.3195,
33.7359, 34.1524, 34.5689, 34.9854, 35.4019, 35.8184, 36.2349, 36.6514, 37.0679, 37.4844,
37.9009, 38.3174, 38.7339, 39.1504, 39.5668, 39.9833, 40.3998, 40.8163, 41.2328, 41.6493,
42.0658, 42.4823, 42.8988, 43.3153, 43.7318, 44.1483, 44.5648, 44.9813, 45.3978, 45.8142,
46.2307, 46.6472, 47.0637, 47.4802, 47.8967, 48.3132, 48.7297, 49.1462, 49.5627, 49.9792,
50.3957, 50.8122, 51.2287, 51.6451, 52.0616, 52.4781, 52.8946, 53.3111, 53.7276, 54.1441,
54.5606, 54.9771, 55.3936, 55.8101, 56.2266, 56.6431, 57.0596, 57.4761, 57.8925, 58.309,
58.7255, 59.142, 59.5585, 59.975, 60.3915, 60.808, 61.2245, 61.641, 62.0575, 62.474,
62.8905, 63.307, 63.7234, 64.1399, 64.5564, 64.9729, 65.3894, 65.8059, 66.2224, 66.6389,
67.0554, 67.4719, 67.8884, 68.3049, 68.7214, 69.1379, 69.5544, 69.9708, 70.3873, 70.8038,
71.2203, 71.6368, 72.0533, 72.4698, 72.8863, 73.3028, 73.7193, 74.1358, 74.5523, 74.9688,
75.3853, 75.8017, 76.2182, 76.6347, 77.0512, 77.4677, 77.8842, 78.3007, 78.7172, 79.1339,
79.5486, 79.9715, 80.377, 80.7808, 81.3623, 81.1547, 82.7847, 83.8168, 69.5365, 130.6)
```

Wynik dla wartości *d* będącej blisko wartości normy obliczany przez metodę Jacobiego wymaga nieoptymalnie wielkiej ilości iteracji aby zbliżyć się do wyniku (25 000+), natomiast metoda Gaussa-Seidela wymaga mniej niż 30 iteracji, dzięki temu, że uwzględnia ona otrzymywane wyniki natychmiastowo w następujących obliczeniach. Ta znaczna różnica ukazuje przewagę metody Gaussa-Seidela dla warunków, gdzie macierz jest **nie-silnie diagonalnie dominująca.**

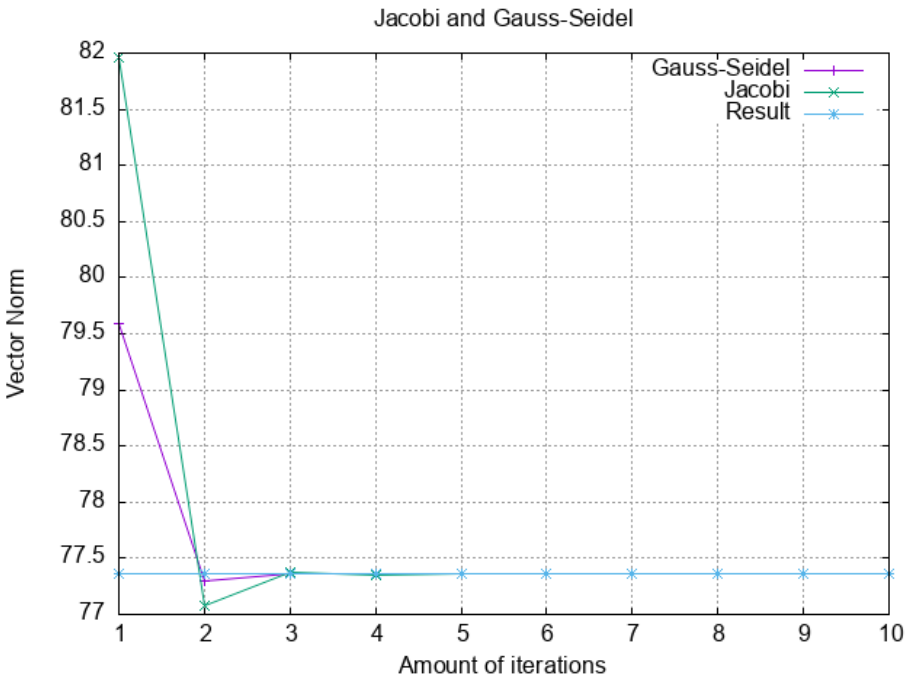
$$d = 20.0 \qquad x_0 = (0, \dots, 0)^T$$

```
Zbieznosc po iteracji dla Jacobiego dla d = (20): 8
Punkt startowy dla powyzszego algorytmu: wektor z samymi 0.
Wynik:

(0.0469338, 0.0943455, 0.14151, 0.188679, 0.235849, 0.283019, 0.330189, 0.377358, 0.424528, 0.471698,
0.518868, 0.566038, 0.613208, 0.660377, 0.707547, 0.754717, 0.801887, 0.849057, 0.896226, 0.943396,
0.990566, 1.03774, 1.08491, 1.13208, 1.17925, 1.22642, 1.27358, 1.32075, 1.36792, 1.41509,
1.46226, 1.50943, 1.5566, 1.60377, 1.65094, 1.69811, 1.74528, 1.79245, 1.83962, 1.88679,
1.93396, 1.98113, 2.0283, 2.07547, 2.12264, 2.16981, 2.21698, 2.26415, 2.31132, 2.35849,
2.40566, 2.45283, 2.5, 2.54717, 2.59434, 2.64151, 2.68868, 2.73585, 2.78302, 2.83019,
2.87736, 2.92453, 2.9717, 3.01887, 3.06604, 3.11321, 3.16038, 3.20755, 3.25472, 3.30189,
3.34906, 3.39623, 3.4434, 3.49057, 3.53774, 3.58491, 3.63208, 3.67925, 3.72642, 3.77358,
3.82075, 3.86792, 3.91509, 3.96226, 4.00943, 4.0566, 4.10377, 4.15094, 4.19811, 4.24528,
4.29245, 4.33962, 4.38679, 4.43396, 4.48113, 4.5283, 4.57547, 4.62264, 4.66981, 4.71698,
4.76415, 4.81132, 4.85849, 4.90566, 4.95283, 5, 5.04717, 5.09434, 5.14151, 5.18868,
5.23585, 5.28302, 5.33019, 5.37736, 5.42453, 5.4717, 5.51887, 5.56604, 5.61321, 5.66038,
5.70755, 5.75472, 5.80189, 5.84906, 5.89623, 5.9434, 5.99057, 6.03774, 6.08491, 6.13208,
6.17925, 6.22642, 6.27358, 6.32075, 6.36792, 6.41509, 6.46226, 6.50943, 6.5566, 6.60377,
6.65094, 6.69811, 6.74528, 6.79245, 6.83962, 6.88679, 6.93396, 6.98113, 7.0283, 7.07547,
7.12264, 7.16981, 7.21698, 7.26415, 7.31132, 7.35849, 7.40566, 7.45283, 7.5, 7.54717,
7.59434, 7.64151, 7.68868, 7.73585, 7.78302, 7.83019, 7.87736, 7.92453, 7.9717, 8.01887,
8.06604, 8.11321, 8.16038, 8.20755, 8.25472, 8.30189, 8.34906, 8.39623, 8.4434, 8.49057,
8.53774, 8.58491, 8.63208, 8.67925, 8.72642, 8.77358, 8.82075, 8.86792, 8.91509, 8.96226,
9.00943, 9.0566, 9.10377, 9.15094, 9.19811, 9.2453, 9.29231, 9.3372, 9.42717, 9.71763)
```

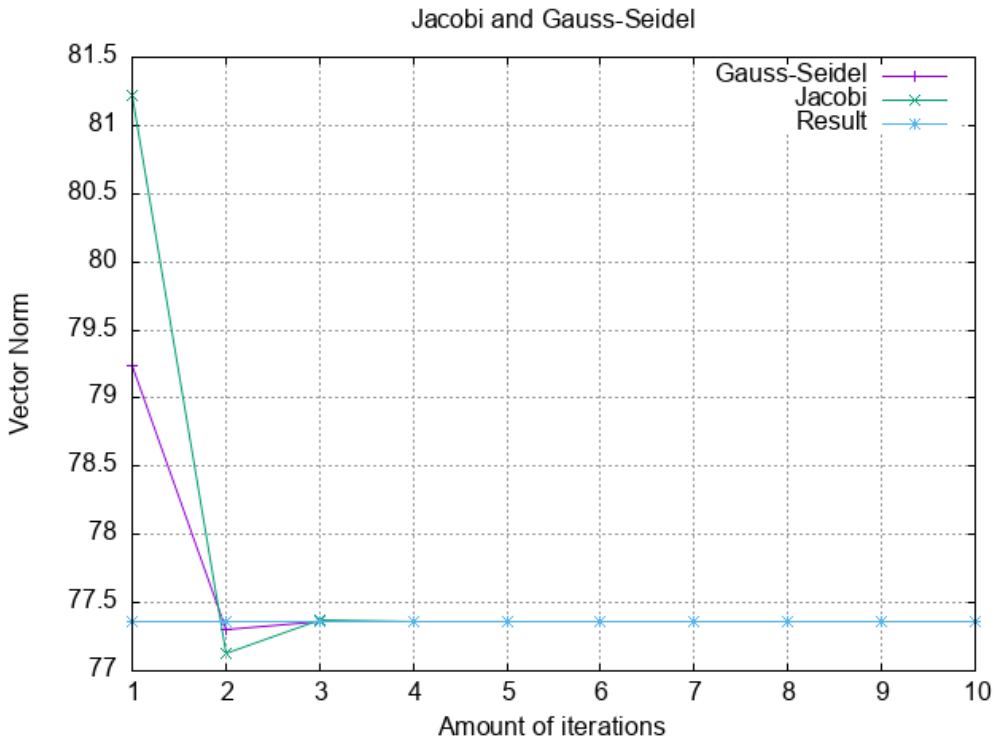
```
Zbieznosc po iteracji dla Gaussa-Seidela dla d = (20): 6
Punkt startowy dla powyzszego algorytmu: wektor z samymi 0.
Wynik:

(0.0469338, 0.0943455, 0.14151, 0.188679, 0.235849, 0.283019, 0.330189, 0.377358, 0.424528, 0.471698,
0.518868, 0.566038, 0.613208, 0.660377, 0.707547, 0.754717, 0.801887, 0.849057, 0.896226, 0.943396,
0.990566, 1.03774, 1.08491, 1.13208, 1.17925, 1.22642, 1.27358, 1.32075, 1.36792, 1.41509,
1.46226, 1.50943, 1.5566, 1.60377, 1.65094, 1.69811, 1.74528, 1.79245, 1.83962, 1.88679,
1.93396, 1.98113, 2.0283, 2.07547, 2.12264, 2.16981, 2.21698, 2.26415, 2.31132, 2.35849,
2.40566, 2.45283, 2.5, 2.54717, 2.59434, 2.64151, 2.68868, 2.73585, 2.78302, 2.83019,
2.87736, 2.92453, 2.9717, 3.01887, 3.06604, 3.11321, 3.16038, 3.20755, 3.25472, 3.30189,
3.34906, 3.39623, 3.4434, 3.49057, 3.53774, 3.58491, 3.63208, 3.67925, 3.72642, 3.77358,
3.82075, 3.86792, 3.91509, 3.96226, 4.00943, 4.0566, 4.10377, 4.15094, 4.19811, 4.24528,
4.29245, 4.33962, 4.38679, 4.43396, 4.48113, 4.5283, 4.57547, 4.62264, 4.66981, 4.71698,
4.76415, 4.81132, 4.85849, 4.90566, 4.95283, 5, 5.04717, 5.09434, 5.14151, 5.18868,
5.23585, 5.28302, 5.33019, 5.37736, 5.42453, 5.4717, 5.51887, 5.56604, 5.61321, 5.66038,
5.70755, 5.75472, 5.80189, 5.84906, 5.89623, 5.9434, 5.99057, 6.03774, 6.08491, 6.13208,
6.17925, 6.22642, 6.27358, 6.32075, 6.36792, 6.41509, 6.46226, 6.50943, 6.5566, 6.60377,
6.65094, 6.69811, 6.74528, 6.79245, 6.83962, 6.88679, 6.93396, 6.98113, 7.0283, 7.07547,
7.12264, 7.16981, 7.21698, 7.26415, 7.31132, 7.35849, 7.40566, 7.45283, 7.5, 7.54717,
7.59434, 7.64151, 7.68868, 7.73585, 7.78302, 7.83019, 7.87736, 7.92453, 7.9717, 8.01887,
8.06604, 8.11321, 8.16038, 8.20755, 8.25472, 8.30189, 8.34906, 8.39623, 8.4434, 8.49057,
8.53774, 8.58491, 8.63208, 8.67925, 8.72642, 8.77358, 8.82075, 8.86792, 8.91509, 8.96226,
9.00943, 9.0566, 9.10377, 9.15094, 9.19811, 9.2453, 9.29231, 9.3372, 9.42717, 9.71763)
```

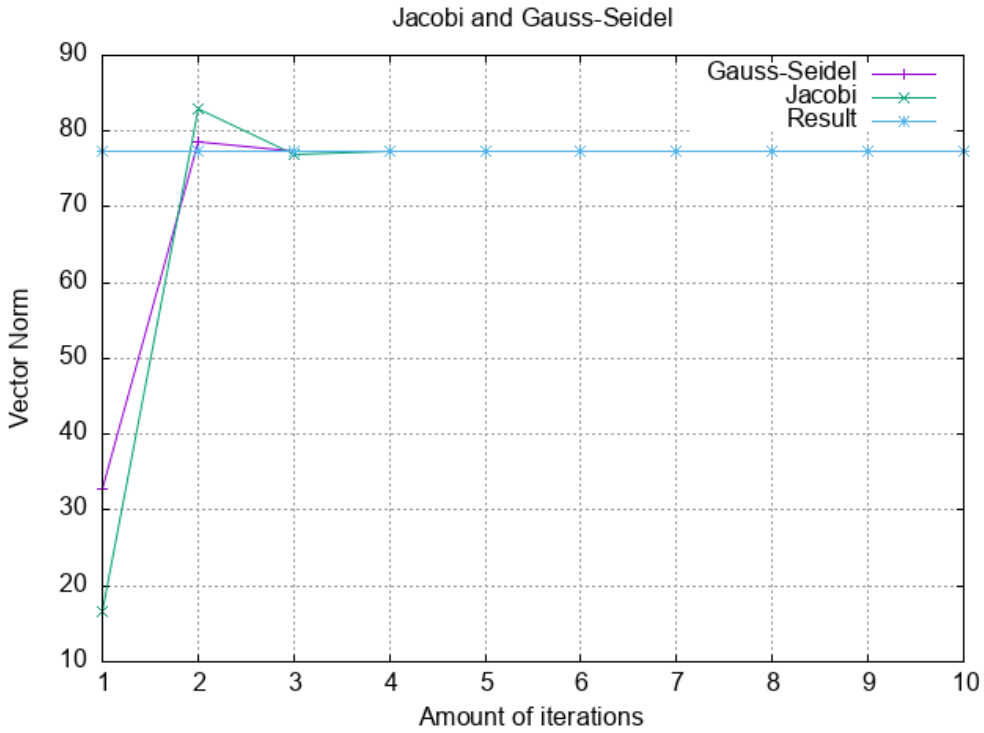


Analogicznie, każdy następny wynik jest taki sam, uwzględnione zostaną tylko wykresy zbieżności.

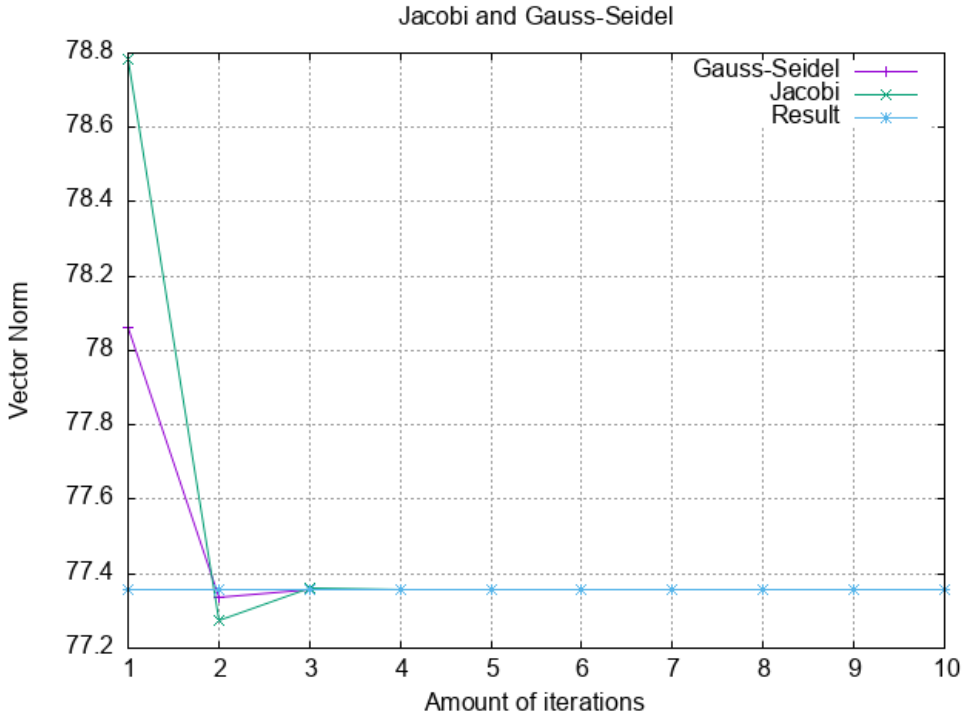
$$d = 20.0 \qquad x_0 = (1, \dots, 1)^T$$



$$d = 20.0 \qquad x_0 = (1, 2, \dots, N)^T$$



$$d = 20.0 \qquad x_0 = (rand(), \dots)^T$$



Wynik za pomocą biblioteki Eigen:

```
Wynik za pomoca biblioteki Eigen:

(0.0469338, 0.0943455, 0.14151, 0.188679, 0.235849, 0.283019, 0.330189, 0.377358, 0.424528, 0.471698,
0.518868, 0.566038, 0.613208, 0.660377, 0.707547, 0.754717, 0.801887, 0.849057, 0.896226, 0.943396,
0.990566, 1.03774, 1.08491, 1.13208, 1.17925, 1.22642, 1.27358, 1.32075, 1.36792, 1.41509,
1.46226, 1.50943, 1.5566, 1.60377, 1.65094, 1.69811, 1.74528, 1.79245, 1.83962, 1.88679,
1.93396, 1.98113, 2.0283, 2.07547, 2.12264, 2.16981, 2.21698, 2.26415, 2.31132, 2.35849,
2.40566, 2.45283, 2.5, 2.54717, 2.59434, 2.64151, 2.68868, 2.73585, 2.78302, 2.83019,
2.87736, 2.92453, 2.9717, 3.01887, 3.06604, 3.11321, 3.16038, 3.20755, 3.25472, 3.30189,
3.34906, 3.39623, 3.4434, 3.49057, 3.53774, 3.58491, 3.63208, 3.67925, 3.72642, 3.77358,
3.82075, 3.86792, 3.91509, 3.96226, 4.00943, 4.0566, 4.10377, 4.15094, 4.19811, 4.24528,
4.29245, 4.33962, 4.38679, 4.43396, 4.48113, 4.5283, 4.57547, 4.62264, 4.66981, 4.71698,
4.76415, 4.81132, 4.85849, 4.90566, 4.95283, 5, 5.04717, 5.09434, 5.14151, 5.18868,
5.23585, 5.28302, 5.33019, 5.37736, 5.42453, 5.4717, 5.51887, 5.56604, 5.61321, 5.66038,
5.70755, 5.75472, 5.80189, 5.84906, 5.89623, 5.9434, 5.99057, 6.03774, 6.08491, 6.13208,
6.17925, 6.22642, 6.27358, 6.32075, 6.36792, 6.41509, 6.46226, 6.50943, 6.5566, 6.60377,
6.65094, 6.69811, 6.74528, 6.79245, 6.83962, 6.88679, 6.93396, 6.98113, 7.0283, 7.07547,
7.12264, 7.16981, 7.21698, 7.26415, 7.31132, 7.35849, 7.40566, 7.45283, 7.5, 7.54717,
7.59434, 7.64151, 7.68868, 7.73585, 7.78302, 7.83019, 7.87736, 7.92453, 7.9717, 8.01887,
8.06604, 8.11321, 8.16038, 8.20755, 8.25472, 8.30189, 8.34906, 8.39623, 8.4434, 8.49057,
8.53774, 8.58491, 8.63208, 8.67925, 8.72642, 8.77358, 8.82075, 8.86792, 8.91509, 8.96226,
9.00943, 9.0566, 9.10377, 9.15094, 9.19811, 9.2453, 9.29231, 9.3372, 9.42717, 9.71763)
```

Dla metody Jacobiego wyniki są osiągane po 8 iteracjach, natomiast dla metody Gaussa-Seidela po 6. Fakt, że metoda Gaussa-Seidela jest wydajniejsza potwierdzają także powyższe wykresy, na których widocznie ta metoda zbiega szybciej do funkcji stałej reprezentującej wynik osiągnięty za pomocą biblioteki Eigen, niż metoda Jacobiego. Najbardziej to widać dla przypadku, gdzie *d* jest bliskie wartości normy.