1 Zadanie numeryczne N13

Szymon Pach

1.1 Opis metody

Zadaniem jest znalezienie wszystkich rozwiązań danego układu 2 równań nieliniowych z 2 niewiadomymi dla danych ograniczeń przedziałów. W tym celu zastosowałem metodę iteracyjną Newtona, gdzie kolejne przybliżenia uzyskuje się wg wzoru: $x_{k+1} = x_k - J^{-1}(x_k)g(x_k)$, gdzie J jest Jakobianem funkcji g, natomiast g jest funkcją, której szukamy miejsca zerowego. Jako punkty startowe przyjąłem kolejne pary liczby (x, y) z krokiem $h = 0.1$, gdzie $x \in [-3, 3]$, $y \in [-3, 3]$. Przybliżenia były zapisywane do wektora wynikowego tylko wtedy, jeśli rozwiązanie różniło się od pozostałych przybliżeń zapisanych w wektorze.

1.2 Instrukcja uruchomienia

Program został napisany w języku Matlab. W środowisku Matlab należy przejść w zakładce *Current folder* do folderu, w którym znajduje się program o nazwie *main.m*, a następnie wpisać w oknie konsoli *main*, a następnie nacisnąć *enter*.

1.3 Rozwiązanie

Rozwiązaniem są następujące pary liczb, gdzie pierwsza kolumna odpowiada za współrzędne *x*, druga za współrzędne *y*.

V =

```
  -0.569246228123221  -0.574454645248109
  0.154218232635811  -0.154225503695219 -
   0.525521713907784   2.561834369565612
  0.470642505628747   1.708284450321633 -
   0.678889859287618   2.600420107178453
   0.578444741708951   0.584105254540553
```

1.4 Analiza błędów

Układ równań został rozwiązany z następującymi błędami.

Błędy w normie euklidesowej

errs_2 =

1.0e-14 *

0.378006077244768

0.011102230246252

0.113220977340074

0.035108334685767

0.064974136686045

0.022887833992611

Błędy w normie maksymalnej

errs_inf =

1.0e-14 *

0.327515792264421

0.011102230246252

0.111022302462516

0.033306690738755

0.061062266354384

0.022204460492503

Obliczenia wykonywałem w pakiecie 32-bitowym. Dokładność rozwiązania wynosi ok. 10^{-14}, więc rozwiązania są dosyć dokładne, biorąc pod uwagę, że można było uzyskać rozwiązania co najwyżej z dokładnością równą ok. 10^{-16}.

1.5 Opis złożoności algorytmu

Maksymalna liczba iteracji dla danych punktów startowych wynosi 50. Mamy $61^2 = 3721$ punktów startowych, więc ogólnie co najwyżej zostanie wykonanych $50 \cdot 3721 = 186050$ iteracji. Oznacza to, że co najwyżej tyle razy trzeba będzie obliczyć wartość funkcji, ułożyć i rozwiązać układ równań.

1.6 Możliwa optymalizacja

W obecnej sytuacji układ równań jest rozwiązywany w każdej iteracji, co jest dość kosztowne. Można by było zmienić macierz J co kilka kroków co przyspieszyłoby obliczenia. Jest to szczególnie efektywne przy dużym wymiarze układu równań, my mamy tylko 2 równania, więc optymalizacja ta nie polepszyłaby znacznie czasu obliczeń.

1.7 Kod źródłowy

```
warning('off','all');
close all; clear; clc;

tol = 1e-6;
tol2 = 2 * tol;
h = 0.1;

V = [inf; inf];
n = 1;

for x = (-3) : h : 3
  for y = (-3) : h : 3
    v = [x; y];
    for iter = 1 : 50
      d = (v(1) + 1) ^ 2 + (v(2) - 1) ^ 2;
      g = [v(1) ^ 2 - sin(v(2) ^ 2); log(d) - 0.98];
      M = [2 * v(1), -2 * v(2) * cos(v(2) ^ 2);
           (2 * (v(1) + 1)) / d, (2 * (v(2) - 1)) / d];
      z = M \ g;
      v = v - z;
      if norm(z, 2) < tol
                  if min( abs( sum( diag(v) * ones(2, n) - V ) ) ) > tol2
          n = n + 1;
          V = [V v];
        end
        break;
      end
    end
  end
end

V = V(:,2:n);
n = n - 1;
```

```
errs_2 = zeros(n, 1);
errs_inf = zeros(n, 1);
for i = 1 : n
   v = V(:,i);
   err = abs([v(1) ^ 2 - sin(v(2) ^ 2); log((v(1) + 1) ^ 2 + (v(2) - 1) ^ 2) - 0.98]);
   errs_2(i) = norm(err, 2);
   errs_inf(i) = norm(err, inf);
end

V = V';

display(V);
display(errs_2);
display(errs_inf);
```