Sprawozdanie

Metody numeryczne 2

Temat 1, Zadanie nr 6

Mateusz Śliwakowski, F4

10/15/2018

1 Treść zadania

Metoda iteracji prostej SOR dla układu $Ax = b$ z macierzą rozrzedzoną.

2 Opis metody

2.1 Wprowadzenie - metody iteracyjne

Służą do rozwiązywania układów równań liniowych

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

Będziemy je zapisywać w postaci $Ax = b$ zakładając:

$$A = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{21} & a_{22} & \dots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \ x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \ b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

W metodzie iteracyjnej tworzony jest ciąg kolejnych przybliżeń $(x^{(1)}, x^{(2)}, \dots)$ taki, że o ile metoda jest zbieżna $\|x^{(k)} - x\| \xrightarrow{k\to\infty} 0$.

Wiele metod iteracyjnych (w tym metoda SOR) ma postać
$x^{(k+1)} = Bx^{(k)} + C$, gdzie $B \in \mathbb{R}^{n\times n}$, a $C \in \mathbb{R}^n$. Postać B i C definiuje konkretną metodę.

2.2 Metoda SOR (Successive Over-Relaxation)

Zdefiniujmy macierze:

$$L = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ a_{21} & 0 & 0 & \dots & 0 \\ a_{31} & a_{32} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots 0 & \end{bmatrix},$$

$$D = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix},$$

$$U = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n-1} & a_{1n} \\ 0 & 0 & \dots & a_{2n-1} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1n} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Wprowadźmy parametr relaksacji $\omega \in \mathbb{R}$. Wzór macierzowy metody SOR ma postać:

$$x^{(k+1)} = \underbrace{(D + \omega L)^{-1}((1-\omega)D - \omega U)}_{B_{SOR}} x^k + \underbrace{(D + \omega L)^{-1} b\omega}_{C_{SOR}}$$

Zapis niemacierzowy:

repeat
  for $i = 1, \dots, n$ **do**
    $x_i^{k+1} = (1-\omega)x_i^{(k)} + \omega(b - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)})/a_{ii}$
  end for
until <uzyskamy zbieżność>

Na potrzeby implementacji zadania posłużymy się zapisem niemacierzowym.

3 Warunki i założenia

1. Macierz wejściowa jest w formacie, który pozwala przyspieszyć obliczenia oraz zmniejszyć złożoność pamięciową dla macierzy rozrzedzonych. Każda kolumna tej macierzy zawiera indeks wiersza, indeks kolumny oraz wartość niezerowego pola macierzy A.
2. Diagonala macierzy wejściowej nie może zawierać zerowego elementu (mielibyśmy wtedy dzielenie przez zero).
3. Aby metoda mogła być zbieżna: $\omega \in (0, 2)$ (wniosek z *lematu Kahana*)
4. Algorytm może zakończyć się w dwóch przypadkach:
   (a) Zostanie spełniony warunek $\|x^{(k)} - x\| < \epsilon$, gdzie ϵ to zadana dokładność).
   (b) Przekroczymy maksymalną liczbę iteracji.

4 Implementacja metody

4.1 Funkcja *sorSparse*

Będzie to funkcja, która rozwiązuje zadany problem.

```
function [x, k] = sorSparse(A, b, w, xFirst, epsilon,
    maxIterations)
```

Parametry wejściowe:

- A - macierz wejściowa w postaci rozrzedzonej (jak w punkcie *Warunki i założenia*),
- b - wektor wyrazów wolnych,
- ω - parametr relaksacji metody SOR (domyślnie 1),
- $xFirst$ - przybliżenie początkowe (domyślnie wektor złożony z samych jedynek),
- $epsilon \in \mathbb{R}$ - określa warunek stopu (domyślnie 1e−5),
- $maxIterations$ - maksymalna liczba iteracji, jaką może przeprowadzić algorytm (domyślnie 1e4).

Parametry wyjściowe:

- x - rozwiązanie układu równań liniowych $Ax = b$,
- k - liczba iteracji.

Na początku definiujemy parametry domyślne dla funkcji. Następnie sprawdzamy czy na głównej diagonali nie występuje zero, co uniemożliwiłoby poprawne działanie algorytmu (wystąpiłoby dzielenie przez zero w pętli obliczającej kolejne przybliżenia). W celu ułatwienia dalszej pracy sortujemy tablicę wejściową względem pierwszego wiersza. Definiujemy wektory pomocnicze, określające ile niezerowych wyrazów znajduje się w każdym wierszu macierzy A (nierozrzedzonej) oraz od jakiego indeksu w macierzy w postaci rozrzedzonej rozpoczynają się ciągi wyrazów z kolejnych wierszy. Następnie wykonywana jest pętla główna metody SOR, gdzie obliczamy kolejne przybliżenia x do czasu, gdy norma z różnicy kolejnych przybliżeń będzie mniejsza od *epsilon* lub przekroczymy maksymalną dopuszczalną liczbę iteracji.

4.2 Pozostałe funkcje

W celu przeprowadzenia testów definiujemy kilka pomocniczych funkcji:

1. $sorNormal$ - implementacja metody SOR dla macierzy nierozrzedzonej, wprowadzanej w postaci macierzy $n \times n$.
2. $matrixToSparse$ - konwerter macierzy z postaci $n \times n$ na postać opisaną w punkcie *Warunki i założenia*.
3. $generateSquareMatrix$ - elastyczny generator macierzy kwadratowej. Użyjemy go do utworzenia testów od dużym rozmiarze.

4. *displayComparison* - wyświetla porównanie podanych metod iteracji oraz wbudowanej funkcji Matlaba. Wyświetla czas działania każdej funkcji, porównuje normy z wyników oraz w przypadku metod iteracyjnych wyświetla liczbę wykonanych iteracji.

5 Przykłady i wnioski

5.1 Przykłady ze względu na wielkość macierzy

Na początku zobaczymy rezultat działania algorytmu dla macierzy o różnych rozmiarach dla parametru ω równego 1.25. Resztę parametrów pozostawimy domyślną, gdyż sprawdzenie wpływu każdego z nich wymagałoby utworzenia zbyt dużej ilości testów. W ramkach przedstawiony zostanie nieznacznie sformatowany wynik działania funkcji *displayComparison*.

1. Macierz 5×5 - jedyna jaką wprowadzimy ręcznie podczas naszych testów:

$$A = \begin{bmatrix} 4 & 0 & 0 & 0 & 3 \\ 3 & 5 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 2 & 0 & 0 & 6 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

   Otrzymujemy:

   Metoda SOR dla macierzy rozrzedzonej:
   Liczba iteracji: 26
   Czas: 0.21186
   Zwykła metoda SOR:
   Liczba iteracji: 26
   Czas: 0.038119
   Wbudowana funkcja Matlaba: Czas: 0.046239
   Norma z różnicy wyników metod SOR: 0
   Norma z różnicy metody SOR i wbudowanej funkcji Matlaba: $9.9478e-06$

   Jak widać nawet d;a macierzy rozrzedzonej małych rozmiarów różnica wydajności pomiędzy metodami SOR jest zauważalna. Obydwie metody uzyskały lepszy wynik również od funkcji Matlaba, lecz warto zauważyć, że uzyskany przez nią wynik jest wynikiem dokładnym.

2. Macierz 25×25 - wygenerowana automatycznie, 95% elementów to zera, reszta elementów to liczby rzeczywiste. Zaznaczmy, że na diagonali umieszczone są elementy znaczni większe. Wektor wyrazów wolnych również wypełniamy losowo, liczbami z przedziału $(-0.5, 0.5)$.

Metoda SOR dla macierzy rozrzedzonej: Liczba iteracji: 11 Czas: 0.0043368 Zwykła metoda SOR: Liczba iteracji: 11 Czas: 0.0050096 Wbudowana funkcja Matlaba: Czas: 0.0081887 Norma z różnicy wyników metod SOR: 0 Norma z różnicy metody SOR i wbudowanej funkcji Matlaba: $1.3348e-06$

Jak widać nawet d;a macierzy rozrzedzonej małych rozmiarów różnica wydajności pomiędzy metodami SOR jest zauważalna. Obydwie metody uzyskały lepszy wynik również od funkcji Matlaba, lecz warto zauważyć, że uzyskany przez nią wynik jest wynikiem z dokładnością maszynową.

3. Macierz 5000×5000 - wygenerowana automatycznie, 99% elementów to zera. Pozostałe parametry analogicznie jak w punkcie wyżej.

Metoda SOR dla macierzy rozrzedzonej: Liczba iteracji: 13 Czas: 0.22015 Zwykła metoda SOR: Liczba iteracji: 13 Czas: 9.6931 Wbudowana funkcja Matlaba: Czas: 0.91442 Norma z różnicy wyników metod SOR: 0 Norma z różnicy metody SOR i wbudowanej funkcji Matlaba: $1.0588e-06$

W tym przykładzie widać wyraźnie, że dla macierzy rozrzedzonych o dużym rozmiarze funkcja przystosowana do pracy z macierzami tego typu jest zdecydowanie bardziej optymalna. Uzyskany czas jest około 44 razy lepszy niż ten uzyskany przez klasyczną metodę SOR. Warto zauważyć że wbudowany algorytm Matlaba znakomicie sobie radzi również z dużymi macierzami - podając wynik z dokładnością maszynową jest zaledwie 4 razy wolniejszy od naszej funkcji.

4. Macierz 100×100 - wygenerowana tak aby suma wartości bezwzględnych znajdujących się na diagonali była mała.

Metoda SOR dla macierzy rozrzedzonej: Metoda jest rozbieżna Czas: 1.41 $\vdots$

Co nie jest zaskoczeniem - metoda SOR okazała się rozbieżna w tym przypadku.

5.2 Przykłady ze względu na parametr relaksacji

Sprawdzimy jaka wartość parametru ω jest optymalna dla macierzy w różnych rozmiarach. Testować będziemy za pomocą pętli:

```
sizes = [5,25,100,500];
for i=1:4
    A = generateSquareMatrix(sizes(i), 0.97, 1, 10, 0)
        ;
    b = 0.5 - rand(sizes(i),1);

    A = matrixToSparse(A);

    bestk = 1e5;
    bestw = 0;

    for w=0.02:0.02:1.98
        [x,k] = sorSparse(A,b,w, zeros(sizes(i),1), 10
            e-8, 10e4);
        if(k < bestk)
            bestk = k;
            bestw = w;
        end
    end

    D = ['Dla rozmiaru: ', int2str(sizes(i)), 'x',
        int2str(sizes(i)) ' najlepsze w: ', int2str(
        bestw), ', liczba iteracji: ', int2str(bestk)];
    disp(D);
end
```

Otrzymaliśmy wyjście:

```
Dla rozmiaru: 5x5 najlepsze w: 1, liczba iteracji: 3
Dla rozmiaru: 25x25 najlepsze w: 1, liczba iteracji: 5
Dla rozmiaru: 100x100 najlepsze w: 0.98, liczba iteracji: 8
Dla rozmiaru: 500x500 najlepsze w; 1, liczba iteracji: 10
```

Okazuję się, że najczęściej, najlepszym wyborem jest $\omega = 1$. Jest to szczególna ω gdyż wtedy pętla metody SOR przyjmuje taką samą postać, jak metoda Gaussa-Seidela. Aby nie zaciemniać rezultatu działania testu wyciąłem komunikaty o rozbieżności, jednak warto zwrócić uwagę iż dla ω bliskiej 0 lub 2 metoda potrzebuje dużej ilości iteracji, lub nawet jest rozbieżna.

5.3 Wnioski

1. Na początku warto odpowiedzieć na pytanie: czy warto programować własną metodę, kiedy algorytm Matlaba jest dobrze napisany i dokładnie przetestowany? Jak pokazują przykłady, jeśli nie zależy nam na wysokiej

dokładności oraz nasza macierz jest rozrzedzona, własny algorytm potrafi być szybszy od algorytmu Matlaba.

2. Warto zauważyć iż pracując na macierzach przechowywanych w postaci rozrzedzonej ($3 \times l$, gdzie l to liczba niezerowych elementów) jesteśmy w stanie przeprowadzać obliczenia na dużo większych macierzach niż bylibyśmy w stanie dla macierzy przechowywanych w tablicach $n \times n$.

3. Pośród testowanych przypadków trudno znaleźć dobre zastosowanie dla samodzielnie napisanej, klasycznej metody SOR - dla dużych macierzy jest ona wolniejsza od funkcji Matlaba nawet o rząd wielkości.

4. Parametr ω znacząco wpływa na zbieżność. Z przeprowadzonych testów wynika, że najlepiej dla macierzy rozrzedzonych wybierać $\omega \approx 1$.