Sprawozdanie z listy 5. - Obliczenia naukowe

Jakub Bachanek

10 stycznia 2021

1 Opis problemu

Celem zadania jest znalezienie efektywnego sposobu na rozwiązanie problemu

$$\boldsymbol{Ax} = \boldsymbol{b}$$

dla danej macierzy współczynników $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $\boldsymbol{b} \in \mathbb{R}^n$. Macierz $\mathbf{A}$ jest macierzą rzadką, tj. mającą dużą liczbę elementów zerowych, i blokową o strukturze:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{C}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \boldsymbol{B}_2 & \boldsymbol{A}_2 & \boldsymbol{C}_2 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_3 & \boldsymbol{A}_3 & \boldsymbol{C}_3 & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{B}_{v-2} & \boldsymbol{A}_{v-2} & \boldsymbol{C}_{v-2} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \boldsymbol{B}_{v-1} & \boldsymbol{A}_{v-1} & \boldsymbol{C}_{v-1} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{B}_v & \boldsymbol{A}_v \end{pmatrix}$$

gdzie $v = \frac{n}{\ell}$, zakładając, że n jest podzielne przez ℓ, gdzie ℓ jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków):

- $\boldsymbol{A}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 1, ..., v$ jest macierzą gęstą
- $\boldsymbol{B}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 2, ..., v$ jest postaci:

$$\boldsymbol{B}_k = \begin{pmatrix} 0 & \cdots & 0 & b_1^k \\ 0 & \cdots & 0 & b_2^k \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & b_\ell^k \end{pmatrix}$$

- $\mathbf{0} \in \mathbb{R}^{\ell \times \ell}$ to kwadratowa macierz zerowa stopnia ℓ

– $\boldsymbol{C}_k \in \mathbb{R}^{\ell\times\ell}$, $k = 1, ..., v-1$ jest macierzą diagonalną:

$$\boldsymbol{C}_k = \begin{pmatrix} c_1^k & 0 & 0 & \cdots & 0 \\ 0 & c_2^k & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{\ell-1}^k & 0 \\ 0 & \cdots & 0 & 0 & c_\ell^k \end{pmatrix}$$

1.1 Przechowywanie macierzy i algorytmy

Z uwagi na wymagania czasowe i pamięciowe nie jest możliwe pamiętanie macierzy $\boldsymbol{A}$ jako tablicy $n \times n$, również standardowe algorytmy będą nieoptymalne dla takiej specyfikacji macierzy. Z tego powodu użyjemy specjalnej struktury `SparseMatrixCSC` z biblioteki `SparseArrays` w języku `Julia`. Korzystając z niej zaoszczędzimy miejsce, ponieważ przechowuje ona tylko elementy niezerowe. Założymy też, że dostęp do elementu macierzy jest w czasie stałym (powszechnie wiadomo, że tak nie jest). Do rozwiązania układów równań liniowych $\boldsymbol{Ax} = \boldsymbol{b}$ użyjemy specjalnie zmodyfikowanych algorytmów:

- Metody eliminacji Gaussa bez wyboru elementu głównego
- Metody eliminacji Gaussa z częściowym wyborem elementu głównego
- Wyznaczenia rozkładu $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ metodą eliminacji Gaussa bez wyboru elementu głównego, a następnie obliczenia $\boldsymbol{LUx} = \boldsymbol{b}$
- Wyznaczenia rozkładu $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ metodą eliminacji Gaussa z częściowym wyborem elementu głównego, a następnie obliczenia $\boldsymbol{LUx} = \boldsymbol{b}$

2 Metoda eliminacji Gaussa

Rozwiązanie układu równań liniowych za pomocą tego algorytmu przebiega w dwóch etapach. W pierwszym za pomocą operacji elementarnych wyłącznie na wierszach doprowadza się macierz $\boldsymbol{A}$ do postaci schodkowej. Powszechnie stosuje się następujący sposób: w k-tym kroku eliminujemy zmienną x_k z równań od $k+1$ do n-tego poprzez pomnożenie k-tego równania przez $I_{jk} = \frac{a_{i\,k}^{(k)}}{a_{k\,k}^{(k)}}$, gdzie $i = k+1, ..., n$, i odjęcie od pozostałych. Wtedy po $n-1$ krokach dostaniemy układ z macierzą górno trójkątną, którą następnie wykorzystujemy w kolejnym etapie w algorytmie podstawiania wstecz. Opiera się on na iteracyjnym obliczaniu

$$x_k = \frac{b_k - \sum_{j=k+1}^{n} u_{kj}}{u_{kk}}$$

dla $k = n-1, ..., 1$.

2.1 Rozwiązanie bez wyboru elementu głównego

Analizując strukturę macierzy $\boldsymbol{A}$ zauważamy, że łatwo operując na indeksach możemy wymusić działanie algorytmu tylko na elementach niezerowych. Osiągniemy to jeżeli będziemy iterować od $k+1$ do $k+(l-k \bmod l)$ (zamiast do n) w celu eliminacji zmiennej x_k z odpowiadającego równania. Podobna zależność istnieje dla algorytmu podstawiania wstecz, gdzie pomijamy te indeksy, które leżą powyżej diagonali macierzy $\boldsymbol{C}$.

```
Dane: A, n, l, b
Wynik: x
for k = 1 to n − 1 do
    counter ← k + (l − k mod l)
    for i = k + 1 to counter do
        mult ← A[i, k]/A[k, k]
        A[i, k] ← 0
        for j = k + 1 to min(n, k + l) do
            A[i, j] ← A[i, j] − mult ∗ A[k, j]
        end
        b[i] ← b[i] − mult ∗ b[k]
    end
end
for i = n 1 do
    sum ← 0
    for j = i + 1 to min(i + l, n) do
        sum ← sum + A[i, j] ∗ x[j]
    end
    x[i] ← (b[i] − sum)/A[i, i]
end
return x
```

2.2 Rozwiązanie z wyborem elementu głównego

Częściowy wybór w eliminacji Gaussa pozwoli zabezpieczyć algorytm przed zagrożeniami takimi jak istnienie zera na diagonali lub niestabilność numeryczna. Ta technika polega na znalezieniu takiego elementu, że

$$|a_{pk}^{(k)}| = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|$$

i przestawieniu w macierzy $\boldsymbol{A}^{(k)}$ wiersza p-tego z k-tym oraz elementu p-tego z k-tym w wektorze $\boldsymbol{b}$. Dalsze kroki są takie same jak poprzednio, jedynie musimy pamiętać indeksy zmian w wektorze permutacji p.

```
Dane: A, n, l, b
Wynik: x
for i = 1 to n do
  | p[i] ← i
end
for k = 1 to n − 1 do
  | counter ← k + (l − k mod l)
  | max_r ← k
  | max_v ← abs(A[k, k])
  | for i = k to counter do
  |   | if abs(A[k, p[i]]) > max_v then
  |   |   | max_r ← i
  |   |   | max_v ← abs(A[k, p[i]])
  |   | end
  | end
  | p[k] ↔ p[max_r
  | for i = k + 1 to counter do
  |   | mult ← A[p[i], k]/A[p[k], k]
  |   | A[p[i], k] ← 0
  |   | for j = k + 1 to min(n, k + 2 ∗ l) do
  |   |   | A[p[i], j] ← A[p[i], j] − mult ∗ A[p[k], j]
  |   | end
  |   | b[p[i]] ← b[p[i]] − mult ∗ b[p[k]]
  | end
end
for i = n 1 do
  | sum ← 0
  | for j = i + 1 to min(i + 2 ∗ l + 1, n) do
  |   | sum ← sum + A[p[i], j] ∗ x[j]
  | end
  | x[i] ← (b[p[i]] − sum)/A[p[i], i]
end
return x
```

2.3 Złożoność algorytmów

Zauważmy, że pętla z iteracją po k wykona się n razy, a inne pętle co najwyżej ℓ razy, co daje nam złożoność $O(\ell^2 n)$. Jest to prawdziwe zarówno dla wersji z wyborem jak i bez wyboru.

3 Rozkład $\boldsymbol{LU}$

Rozkład $\boldsymbol{LU}$ macierzy $\boldsymbol{A}$ to przedstawienie jej w postaci iloczynu $\boldsymbol{A} = \boldsymbol{LU}$, gdzie $\boldsymbol{L}$ jest macierzą dolną trójkątną, a $\boldsymbol{U}$ macierzą górno trójkątną. Do uzyskania tego rozkładu można użyć metody eliminacji Gaussa, wtedy $\boldsymbol{U}$ uzyskujemy przez przekształcenie $\boldsymbol{A}$ w trakcie wykonywania algorytmu, a $\boldsymbol{L}$ jest uzyskana poprzez zapisywanie mnożników $I_{ij}^{(k)}$ w tej samej macierzy $\boldsymbol{A}$ w miejscu zerowanych elementów $a_{ij}^{(k)}$. Znając rozkład $\boldsymbol{A} = \boldsymbol{LU}$ zadanie $\boldsymbol{Ax} = \boldsymbol{b}$ sprowadzamy do rozwiązania dwóch układów trójkątnych $\boldsymbol{Ly} = \boldsymbol{b}$ i $\boldsymbol{Ux} = \boldsymbol{y}$, co możemy zrobić stosując wspomniany algorytm podstawiania wstecz i jego analogię – podstawiania w przód.

W trakcie tworzenia rozkładu korzystamy z tych samych technik ograniczania zakresów iteracji jedynie po elementach dla nas istotnych, odrzucamy te, gdzie z góry wiemy, że występuje tam zero.

Poniższe algorytmy przedstawiają sposób rozwiązywania $\boldsymbol{Ly} = \boldsymbol{b}$ oraz $\boldsymbol{Ux} = \boldsymbol{y}$ kolejno dla wariantu bez wyboru jak i z nim.

Dane: A, n, l, b
Wynik: x
for $i = 1$ **to** n **do**
  $sum \leftarrow 0$
  for $j = max(1, (l * \lfloor (i-1)/l \rfloor))$ **to** $i - 1$ **do**
    $sum \leftarrow sum + A[i, j] * y[j]$
  end
  $y[i] \leftarrow b[i] - sum$
end
for $i = n\ 1$ **do**
  $sum \leftarrow 0$
  for $j = i + 1$ **to** $min(n, i + l)$ **do**
    $sum \leftarrow sum + A[i, j] * x[j]$
  end
  $x[i] \leftarrow (y[i] - sum)/A[i, i]$
end
return x

Dane: A, n, l, b
Wynik: x
for $i = 1$ **to** n **do**
  $sum \leftarrow 0$
  for $j = max(1, (l * \lfloor(i-1)/l\rfloor))$ **to** $i - 1$ **do**
  | $sum \leftarrow sum + A[p[i], j] * y[j]$
  end
  $y[i] \leftarrow b[p[i]] - sum$
end
for $i = n$ 1 **do**
  $sum \leftarrow 0$
  for $j = i + 1$ **to** $min(n, i + 2 * l)$ **do**
  | $sum \leftarrow sum + A[i, p[j]] * x[j]$
  end
  $x[i] \leftarrow (y[i] - sum)/A[p[i], i]$
end
return x

3.1 Złożoność algorytmów

W związku z tym, że algorytmy rozkładu ***LU*** i rozwiązywania równań wykorzystują jedynie lekko zmodyfikowane wcześniej przytoczone metody eliminacji Gaussa i podstawiania wstecz i w przód, dostajemy ponownie złożoność $O(n)$.

4 Wyniki

Poniższa tabela przedstawia wyniki błędów względnych dla metod eliminacji Gaussa.

rozmiar n macierzy	eliminacja Gaussa	eliminacja Gaussa z częściowym wyborem
16	5.102800490722269e-16	2.8844440295753465e-16
10000	8.252489306370072e-15	4.556420758958771e-16
50000	1.8398915733251618e-13	4.783590363830398e-15

Poniższa tabela przedstawia wyniki błędów względnych dla rozkładów LU.

rozmiar n macierzy	rozkład LU	rozkład LU z częściowym wyborem
16	9.879581447690234e-16	2.64771316376407e-16
10000	9.673417173856611e-15	4.529750757309705e-16
50000	2.0040929265984656e-13	4.866635623115654e-15

Analizując wyniki widzimy, że częściowy wybór zawsze generuje mniejsze błędy względne, jednak obie techniki wykazują się dobrą dokładnością obliczeń.

5 Wnioski

Dla pewnych specyficznych problemów jesteśmy w stanie zmodyfikować niektóre algorytmy, specjalnie dopasowując je do znanych informacji o problemie, osiągając w ten sposób znacznie wyższą wydajność obliczeń.