Paralelní LU rozklad

Lukáš Michalec

Katedra fyziky, Přírodovědecká fakulta Univerzity J.E. Purkyně v Ústí n.L.
ročník, specializace

Abstract

Seminární práce se zabývá řešení soustavy lineárních rovnic pomocí LU dekompozice a zrychlení výpočtu pomocí paralelizace této metody.

1 Úvod

K jedním z nejstarším matematickým problémům patří řešení soustavy lineárních rovnic. S tímto problémem se můžeme například setkat v lineárním programování, předpovědích či odhadováním.

Řešení tohoto problému je několik, například Gaussova eliminace, Cramerov pravidlo, pomocí inverzní matice, metodou nejmenších čterců a LU rozklad.

2 Teorie

Takovou soustavou lineárních rovnic může být například soustava:

$$\begin{aligned} 3x + 5y - z &= 2 \\ 4x + 1y + 2z &= 1 \\ 1x - 2y + 1z &= 5 \end{aligned} \tag{1}$$

V této soustavě máme tři promněné (x,y,z). Graficky si tuto soustavu můžeme představi jako soustavu tří ploch a my hledáme bod, ve kterém se dané plochy protínají.

Místo rovnicového zápisu, můžeme použít i maticový zápis:

$$A\vec{x} = \vec{b} \tag{2}$$

kde **A** je matice koeficinetů, **x** je vektor promněných a **b** je vektor požadavků. Pokud přepíšeme soustavu 1, dostaneme:

$$\begin{pmatrix} 3 & 5 & -1 \\ 4 & 1 & 2 \\ 1 & -2 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 5 \end{pmatrix} \tag{3}$$

2.1 LU rozklad

Metoda LU dekompozice je založná takovém rozkladu matice A, aby se zjednodušil výpočet výsledného řešení. LU rozklad jde použít jen na regulární čtvercové řešení, to znamená, že rovnice mají právě jedno řešení a to řešení existuje. Naším úkol je rozložit matici A na matice L a U, tak aby:

$$A = L * U \tag{4}$$

kde matice **L** je dolní trojuhelníková a matice **U** je horní trojuhelníková. Tímto rozkladem dostaneme rovnici:

$$L * U * \vec{x} = \vec{b} \tag{5}$$

součin matice U a vektoru x si označíme $U * \vec{x} = \vec{z}$, takže dostaneme:

$$L * \vec{z} = \vec{b} \tag{6}$$

Prvním krokem je potom už dopředná substituce v rovnici: $L * \vec{z} = \vec{b}$ a spočteme $\vec{z}$.

Dalším krokem potom bude zpětná substituce, dosahíme za $\vec{z}$ a spočteme $\vec{x}$: $U * \vec{x} = \vec{z}$

3 Výpočet LU rozkladu

Algoritmus rozkladu si ukážeme na konkrétním příkladě. Mějme matici:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 4 & 2 & 0 \end{pmatrix} \tag{7}$$

z této matice, potřebujeme získat matice L a U:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \tag{8}$$

Pokud si rozepíšeme násobení matic L a U, tak aby výsledkem byla matice A, tak dostaneme soustavu jednoduchých rovnic:

$$\begin{aligned} u_{11} * 1 &= 1 \rightarrow u_{11} = 1 \\ u_{12} * 1 &= 2 \rightarrow u_{12} = 2 \\ u_{13} * 1 &= 3 \rightarrow u_{13} = 3 \\ l_{21} * u_{11} &= 2 \rightarrow l_{21} = 2 \\ &\vdots \end{aligned} \tag{9}$$

což můžeme zapsat algoritmem jako:

```
DO k = 1, n-1
  DO i = k+1, n
    a(i,k) = a(i,k) / a(k,k)
  END DO

  DO i = k+1, n
    a(i,j) = a(i,j) - a(i,k) * a(k,j)
  END DO
END DO
```

Pro ušetření místa, zapíšeme matice L a U do matice A.

3.1 Dopředná substituce

LU rozkladem jsme dostali matice L a U:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 2 & 2.5 & 1 \end{pmatrix}, U = \begin{pmatrix} 2 & -1 & 7 \\ 0 & 2 & -23 \\ 0 & 0 & 41.5 \end{pmatrix} \tag{10}$$

a dále budeme řešit:

$$L * \vec{y} = \vec{b} \tag{11}$$

Zde opět dostaneme soustavu rovnic:

$$\begin{aligned} y_1 &= 11 \\ 4y_1 + y_2 &= 25 \\ 2y_1 + 2.5y_2 + y_3 &= 16 \end{aligned} \tag{12}$$

opět, řešení této soustavy je triviální a pokud to přepíšeme do fortranovského kódu:

```
DO i = 2, n
  DO j = 1, i - 1
    y(i) = y(i) - a(i,j) * y(j)
  ENDDO
ENDDO
```

3.2 Zpětná substituce

Posledním krokem je zpětná substituce, kde řešíme rovnici:

$$U * \vec{x} = \vec{y} \tag{13}$$

$$\begin{pmatrix} 2 & -1 & 7 \\ 0 & 2 & -23 \\ 0 & 0 & 41.5 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{pmatrix} 11 \\ -19 \\ 41.5 \end{pmatrix} \tag{14}$$

opět, když jsi rozepíšeme násobení matice a vektoru, tak dostaneme triviální soustavu rovnic:

$$\begin{aligned} x_3 &= 1 \\ 2x_2 - 23x_3 &= -19 \\ 2x_1 - x_2 + 7x_3 &= 11 \end{aligned} \tag{15}$$

Fortranovský kód vypadá následovně:

```
DO i = n, 1, -1
  DO j = i + 1, n
    x(i) = y(i) - a(i, j) * x(j)
  ENDDO
  x(i) = x(i) / a(i,i)
ENDDO
```

A tímto jsme získali řešení soustavy lineárních rovnic.

3.3 Paralelizace metody LU

Paralelizace této metody spočívá v distribuci cyklů mezi všemy procesory. Takovou ukázkou distribuce je následující příklad:

```
DO i = myrank, n-1, nproc
        print *,"procesor␣",myrank,"␣dela␣",i,".␣iteraci"
ENDDO
```

výstupem potom je:

```
procesor 0 dela 0. iteraci
procesor 1 dela 1. iteraci
procesor 2 dela 2. iteraci
procesor 0 dela 3. iteraci
procesor 1 dela 4. iteraci
...
```

A jelikož všechny tři kroky LU dekompozice jsou iterační, tak lze distribuci cyklů provést u všech kroků.

3.3.1 Paralelizace rozkladu LU

```
DO k = 1, n
  IF (map(k) == myrank) THEN
    DO i = k+1, n
      a(i,k) = a(i,k) / a(k,k)
    ENDDO
  ENDIF

  CALL MPI_BCAST(a(k,k), n-k+1, MPI_REAL, map(k),MPI_COMM_WORLD, ier

  DO j = k+1, n
    IF (map(j) == myrank) THEN
      DO i = k+1, n
        a(i,j) = a(i,j) - a(i,k) * a(k,j)
      ENDDO
    ENDIF
  ENDDO
ENDDO
```

Zde byla distribuce cyklů zajištěna pomocí podmínek v cyklu, kvůli synchronizaci. Každý procesor si nejdříve spočítá vlastní část L matice, pak si ty procesory mezi s sebou ty hodnoty vymění a pak pokračují u části U matice.

3.3.2 Paralelizace dopředné distribuce

```
DO i = 2, n
  s = 0.0
  DO j = 1 + myrank, i - 1, nprocs
    s = s + a(i,j) * y(j)
  ENDDO

  CALL MPI_ALLREDUCE(s, ss, 1, MPI_REAL, MPI_SUM,MPI_COMM_WORLD, ier
  y(i) = y(i) - ss
ENDDO
```

V této části, kdy počítáme vektor $\vec{y}$, tak jsou iterace na sobě závislé, protože postupně dosazujeme vypočtené promněné do dalších rovnic. Proto jsme zde

jen zparalelizovali sumu násobků, která je potřeba k výpočtu. Toto urychlení se projevý hlavně u hodně velkých matic.

3.3.3 Paralelizace zpětné distribuce

```
DO i = n, 1, -1
  s = 0.0
  IF (map(i+1) <= myrank) THEN
    ii = i + 1 + myrank - map(i+1)
  ELSE
    ii = i + 1 + myrank - map(i+1) + nprocs
  ENDIF

  DO j = ii, n, nprocs
    s = s + a(i, j) * b(j)
  ENDDO

  CALL MPI_ALLREDUCE(s, ss, 1, MPI_REAL, MPI_SUM,MPI_COMM_WORLD, ier

  b(i) = (b(i) - ss) / a(i,i)
ENDDO
```

v této části jsme opět rozparalelizovali sumu násobků,která je potřebná k dosazování do rovnic. Jen bylo složitější zpočtení, co který proces má násobit a od jakého indexu začít.

4 Výsledky a diskuze

Výsledkem máme program, který je schopný pomocí LU dekompozice spočítat řešení soustavy lineárních rovnic.

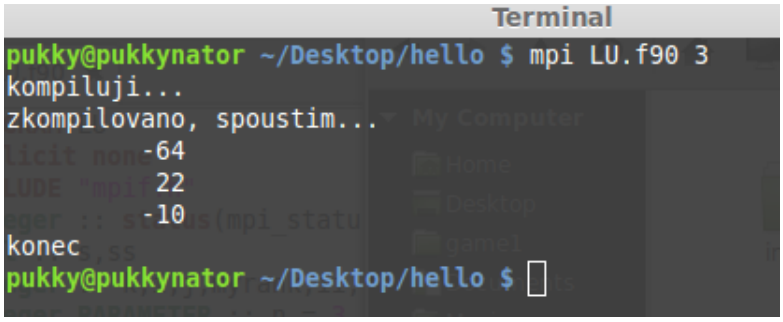

Figure 1: Ukázka výstupu programu

Pokud by jsme změřili rychlost výpočtu u velkých matic, zjistili bychom, že jsme ušetřili více času.

5 Závěr

Tato úloha nám ukázala, jakým lehkým způsobem se dá urychlit iterační výpočet a jak vůbec funguje metoda LU dekompozice.