Shared Memory Implementation of Jacobi with Dominant Row

Gabriel Dimitriu

1 Introduction

Let

$$Ax = a \tag{1}$$

be our system of linear equations, with $A = (a_{ij})_{i,j \in \{1,...,m\}}$, $a = (a_1, ..., a_m)$ and $x = (x_1, ..., x_m)^T$.

Suppose $a_{ii} \neq 0$ for any $i \in \{1, ..., m\}$ then we can note $D = \begin{pmatrix} a_{11} & 0 \\ 0 & a_{mm} \end{pmatrix}$ and $D^{-1} = \begin{pmatrix} 1/a_{11} & 0 \\ 0 & 1/a_{mm} \end{pmatrix}$.

The system (1) could be transform in $D^{-1}Ax = D^{-1}a$ or

$$\left(I - \left(\underbrace{I - D^{-1}A}_{B} \right) \right) x = \underbrace{D^{-1}a}_{b}$$

which is equivalent with

$$(I - B)x = b \tag{2}$$

Theorem 1 *If*

$$|a_{ii}| > \sum_{j=1, j \neq i}^{m} |a_{ij}|, \forall i \in \{1, ..., m\} \tag{3}$$

then the system (2) has the unique solution z and $\forall x^{(0)} \in R^m$ the string $(x^{(n)})_{n \in N}$ with $x^{(n+1)} = Bx^{(n)} + b$ convert to z and it takes palaces the following relations:

$$\left\| x^{(n)} - z \right\|_{\infty} \leq \frac{q}{1-q} \left\| x^{(n)} - x^{(n-1)} \right\|_{\infty} \leq \frac{q^n}{1-q} \left\| x^{(1)} - x^{(0)} \right\|_{\infty} \tag{4}$$

where

$$q = \max_{1 \leq i \leq m} \sum_{j=1, j \neq i}^{m} \left| \frac{a_{ij}}{a_{ii}} \right|$$

2 Implementation of generator C

Where in conformity with system (1) :mat is matrix A of coefficients, y is the free vector a,x is the solution x of the system, err is the error and rez is the correct solution.

Because I want to have a random solution but also to can verified the solution and the system to be compatible with systems solved by Jacobi with dominant row I use the following generator which will assure me that I have the following solution rez=(1,2,...,m) for a m dimensional problem.

```
for(i=0;i<dim;i++) rez[i]=(double)i+1;
for(i=0;i<dim;i++)
    for(j=0;j<dim;j++) if(i!=j) {
            mat[i][j]=20000*rand()/(double)RAND_MAX;
            if((rand()/(double)RAND_MAX)<0.5) mat[i][j]=-mat[i][j]; }
for(i=0;i<dim;i++) {
    temp=0.0;
    for(j=0;j<dim;j++) if(j!=i) temp+=fabs(mat[i][j]);
    mat[i][i]=temp+(20000+temp)*rand()/(double)RAND_MAX+0.00001;
    if((rand()/(double)RAND_MAX)<0.5) mat[i][i]=-mat[i][i]; }
for(i=0;i<dim;i++) {
    y[i]=0.0; x[i]=0.0;
    for(j=0;j<dim;j++) y[i]+=mat[i][j]*rez[j]; }
```

3 Results

The Jacobi with dominant row without load balance is compiled with Omni 1.6 for OpenMP and with the original GCC with pthread implementation from Fedora Core 1 with maximum optimization "-O3". The pthread implementation is a transcription of OpenMP implementation.

The programs ran on a dual pentium II at 500MHz with 256MB RAM and LINUX Fedora Core 1 and resulted the following graph. In the graph with red is plotted the results from Omni and with blue is plotted the results from pthread.

The results are a little different because I have my own implementation of barrier which is not fully optimized as the barrier from directive **single** from OpenMP standard.