Shared Memory Implementation of Jacobi with Dominant Collumn

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 $a_{jj} \neq 0$ and $a_{ii} > \sum_{i=1}^{m} a_{ij}$ 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 $AD^{-1}Dx = a$ or

$$\left(I - \left(I - \underbrace{AD^{-1}}_{C}\right)\right) Dx = a$$

which is equivalent with

$$(I - C)Dx = a \tag{2}$$

Let consider the following system

$$(I - C)y = a \tag{3}$$

Theorem 1 *If $a_{jj} \neq 0$ and $a_{ii} > \sum_{i=1}^{m} a_{ij}$ for any $i \in \{1, ..., m\}$ the w exist and is unique so $(I - C)w = a$ and the Jacobi method is convergent for the system (3) and the evaluation error for the original system is*

$$\left\| x^{(n)} - z \right\|_1 = \left\| D^{-1}(y^{(n)} - w \right\|_1 \leq \frac{1}{\min_{1\leq j\leq m} |a_{jj}|} \frac{q}{1-q} \left\| y^{(n)} - y^{(n-1)} \right\|_1$$

Where: $\|x\|_1 = \sum_{i=1}^{m} |x_i|$ and $q = \|C\|_1 = \max_{1\leq j\leq m} \sum_{i=1, i\neq j}^{m} \left|\frac{a_{ij}}{a_{jj}}\right| < 1$.

2 Implementation of generator in C

Where in conformity with system (1) we have: mat is matrix A of coefficients,y is the free vector a,x is the solution x of the system,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 column 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[j][i]);
    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]; }
//generate the free term
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 column 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.