Report

Assignment_3

Lid driven cavity problem

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    int i,j ,m,n,p;
    double dx,dy,x,y;
    m=101;
    n=101;
    p=(m-2)*(n-2);
    dy=(1.0/(n-1));
    dx=(1.0)/(m-1);
    double psi[m][n],omega[m][n],u[m][n],v[m][n],psi_prev[m][n],omega_prev[m][n], error_psi=1.0, error_omega=1.0, beta =(dx/dy),Re=100.0;
    printf("%d",m);
    int iteration=0.0;
    //Boundary conditions
    for ( i= 0; i < m; i++)
    {
        for ( j = 0; j < m; j++)
```

```
    {
        //intializing
        u[i][j]=0.0;
        v[i][j]=0.0;
        psi[i][j]=0.0;
        //bottom
        u[i][0]=0.0;
        v[i][0]=0.0;
        psi[i][0]=0.0;
        //top
        u[i][n-1]=1.0;
        v[i][n-1]=0.0;
        psi[i][n-1]=0.0;
        //left
        u[0][j]=0.0;
        v[0][j]=0.0;
        psi[0][j]=0.0;
         //right
        u[m-1][j]=0.0;
        v[m-1][j]=0.0;
        psi[m-1][j]=0.0;

    }

}
```

```
for ( i= 0; i < m; i++)
{
    for ( j = 0; j < n; j++)
    {
        //intializing
        omega[i][j]=0.0;
          //bottom
        omega[i][0]=(2.0/pow(dy,2))*(psi[i][0]-psi[i][1]);

          //top
omega[i][n-1]=(2.0/pow(dy,2))*(psi[i][n-1]-psi[i][n-1-1])-((2.0/dy)*u[i][n-1]);

           //left
            omega[0][j]=(2.0/pow(dx,2))*(psi[0][j]-psi[0+1][j]);

           //right
            omega[m-1][j]=(2.0/pow(dx,2))*(psi[m-1][j]-psi[m-1-1][j]);

    }

}
FILE*file1;
file1=fopen("output.dat","w");
// gauss- seidel methode
while (error_psi>1e-8 || error_omega>1e-8)
```

```
{
    for ( i= 0; i <m; i++)
    {
        for ( j = 0; j< n; j++)
        {
            psi_prev[i][j]=psi[i][j];
            omega_prev[i][j]=omega[i][j];
        }

    }
    // stream function
    for ( i= 1; i <m-1; i++)
    {
        for ( j = 1; j < n-1; j++)
        {

psi[i][j]=(0.5/(1.0+pow(beta,2)))*(psi[i+1][j]+psi[i-1][j]+(pow(beta,2)*(psi[i][j+1]
+psi[i][j-1]))+(pow(dx,2)*omega[i][j]));
        }

    }
    //vortocity equation
    for ( i= 1; i <m-1; i++)
    {
        for ( j = 1; j < n-1; j++)
        {
```

```
omega[i][j]=(0.5/(1.0+pow(beta,2)))*((1.0-((psi[i][j+1]-psi[i][j-1])*((beta*Re)/4.0)
))*omega[i+1][j]
                    +(1.0+((psi[i][j+1]-psi[i][j-1])*((beta*Re)/4.0)))*omega[i-1][j]

+((1.0+((psi[i+1][j]-psi[i-1][j])*(Re/(4.0*beta))))*(pow(beta,2)*omega[i][j+1]))

+((1.0-((psi[i+1][j]-psi[i-1][j])*(Re/(4.0*beta))))*(pow(beta,2)*omega[i][j-1])));
                }


        }
          for ( i= 0; i < m; i++)
    {
        for ( j = 0; j < n; j++)
        {

                omega[i][0]=(2.0/pow(dy,2))*(psi[i][0]-psi[i][1]);



omega[i][n-1]=(2.0/pow(dy,2))*(psi[i][n-1]-psi[i][n-1-1])-((2.0/dy)*u[i][n-1]);


                omega[0][j]=(2.0/pow(dx,2))*(psi[0][j]-psi[0+1][j]);


                omega[m-1][j]=(2.0/pow(dx,2))*(psi[m-1][j]-psi[m-1-1][j]);
```

```
    }

}
error_psi=0.0;
error_omega=0.0;
for ( i= 0; i < m-1; i++)
{
    for ( j= 0; j < n-1; j++)
    {
        error_psi=error_psi+pow((psi[i][j]-psi_prev[i][j]),2.0);
        error_omega=error_omega+pow((omega[i][j]-omega_prev[i][j]),2.0);
    }

}
error_psi=sqrt(error_psi/p);
error_omega=sqrt(error_omega/p);
printf("iteration=%d\t",iteration);
printf("error_psi=%.50lf\t error_omega=%.50lf\n",error_psi,error_omega);
iteration++;


}

for ( i = 0; i < m; i++)
```

```
    {
        for ( j= 0; j < n; j++)
        {
            u[i][j]=(0.5/dy)*(psi[i][j+1]-psi[i][j-1]);
            v[i][j]=(-0.5/dx)*(psi[i+1][j]-psi[i-1][j]);

        }

    }
    //fprintf(file1,"VARIABLES=
\"X\",\"Y\",\"U\",\"V\",\"PSI\",\"OMEGA\",\n");
    fprintf(file1,"ZONE I=%d, J=%d\n",m,n);
     for ( i = 0; i < m; i++)
    {
        x=i*dx;
        for ( j = 0; j < n; j++)
        {
            y=j*dy;

fprintf(file1,"%f\t%f\t%f\t%f\t%f\t%f\n",x,y,u[i][j],v[i][j],psi[i][j],omega[i][j]);
            printf("%f\t",psi[i][j]);
        }

    }
```

```
    return 0;
}
```

RESULT: