Computational Fluid Dynamics
http://www.nd.edu/~gtryggva/CFD-Course/

Computational Fluid Dynamics

Lecture 2
January 23, 2017

Grétar Tryggvason

Computational Fluid Dynamics

Accuracy

Computational Fluid Dynamics

It is clear that although the numerical solution is qualitatively similar to the analytical solution, there are significant quantitative differences.

The derivation of the numerical approximations for the derivatives showed that the error depends on the size of h and Δt.

First we test for different Δt.

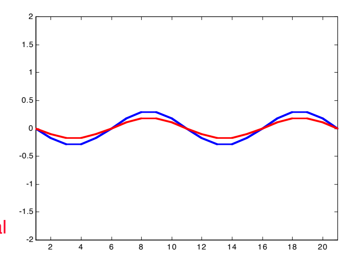
Number of time steps= T/Δt

Computational Fluid Dynamics
Accuracy

Evolution for
U=1;
D=0.05;
k=1

N=21
Δt=0.05

m=11; time=0.50



Exact

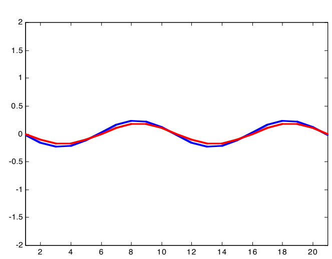
Numerical

Computational Fluid Dynamics
Accuracy

Repeat with a smaller time-step
Δt=0.025

N=21

m=21; time=0.50



Exact

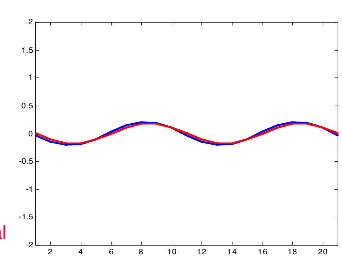
Numerical

Computational Fluid Dynamics
Accuracy

Repeat with a smaller time-step
Δt=0.0125

N=21

m=41; time=0.50



Exact

Numerical

Computational Fluid Dynamics

How accurate solution can we obtain?

Take
Δt = 0.0005
and
N=200

Computational Fluid Dynamics
Accuracy

Very fine spatial resolution and a small time step

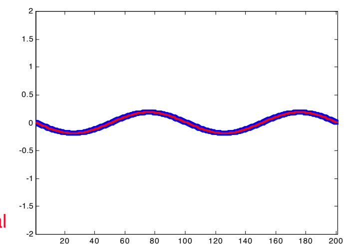
U=1;
D=0.05;
k=1

N=200
Δt=0.0005

Exact

Numerical

m=1001; time=0.50



Computational Fluid Dynamics
Accuracy

Quantifying the Error
Order of Accuracy

Examine the spatial accuracy by taking a very small time step, Δt = 0.0005 and vary the number of grid points, N, used to resolve the spatial direction.

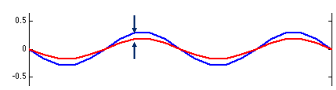
The grid size is $h = L/N$ where $L = 1$ for our case

Computational Fluid Dynamics
Accuracy

Exact

Numerical

time=0.50



N=11; E = 0.1633
N =21; E = 0.0403
N =41; E = 0.0096
N =61; E = 0.0041
N =81; E = 0.0022
N =101; E = 0.0015
N =121; E = 0.0011
N =161; E = 9.2600e-04

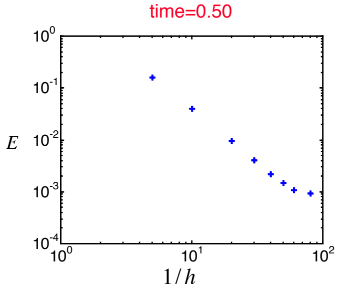
$$E = h\sqrt{\sum_{j=1}^{N}(f_j - f_{exact})^2}$$

Computational Fluid Dynamics
Accuracy

Accuracy. Effect of spatial resolution

dt=0.0005
N=11 to N=161

time=0.50



Computational Fluid Dynamics
Accuracy

If the error is of second order:

$$E = Ch^2 = C\left(\frac{1}{h}\right)^{-2}$$

Taking the log:

$$\ln E = \ln\left(C\left(\frac{1}{h}\right)^{-2}\right) = \ln C - 2\ln\left(\frac{1}{h}\right)$$

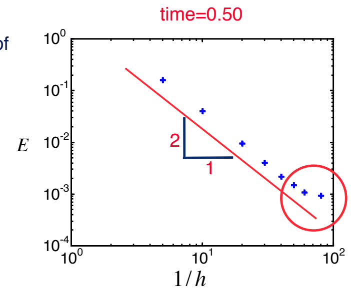
On a log-log plot, the E versus $(1/h)$ curve should therefore have a slope -2

Computational Fluid Dynamics
Accuracy

Accuracy. Effect of spatial resolution

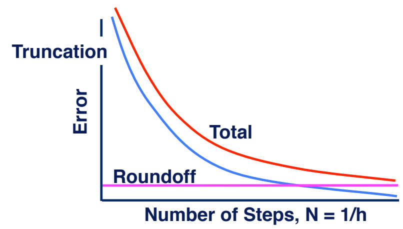
dt=0.0005

N=11 to N=161



Computational Fluid Dynamics
Accuracy

Why is does the error deviate from the line for the highest values of N?



Computational Fluid Dynamics
Summary

Finite difference approximations by Taylor expansion

Approximating a partial differential equation

Showed, by numerical experiments, that accuracy increases as the resolution is increased

Showed that the error behaves in a way that should be predictable

Computational Fluid Dynamics

Stability

Computational Fluid Dynamics
Stability

As long as accuracy is reasonable, integration at larger time steps is more efficient and desirable.

Can we increase the time step indefinitely?

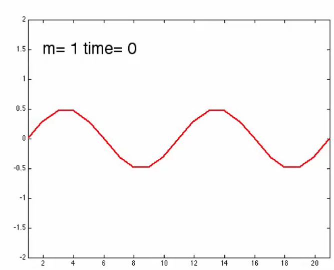
Let's repeat the 1-D advection-diffusion equation with larges time step.

Use Δt = 0.2, instead of Δt = 0.05

Computational Fluid Dynamics
Stability

Evolution for
U=1;
D=0.05;
k=1

N=21
Δt=0.2



Exact

Numerical

Computational Fluid Dynamics
Stability

Instead of decaying as it should, the amplitude of the numerical solution keeps increasing.

Indeed, if we continued the calculations, we would eventually produce numbers larger than the computer can handle. This results in an “overflow” or “NaN” (Not a Number).

Computational Fluid Dynamics

Ordinary Differential Equation Stability

Computational Fluid Dynamics
ODE—Example

Take:

$$\frac{df}{dt} = -f \quad \text{with initial condition} \quad f(0) = 1$$

The exact solution is

$$f(t) = e^{-t}$$

Forward Euler

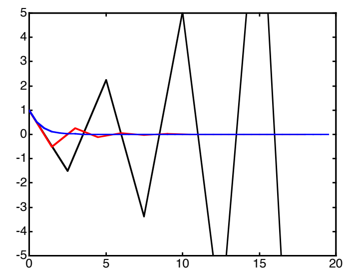
$$f^{n+1} = f^n - f^n \Delta t \quad \rightarrow \quad f^{n+1} = (1 - \Delta t) f^n$$

$$\left| \frac{f^{n+1}}{f^n} \right| \le 1 \quad \text{only if} \quad \Delta t \le 2$$

Computational Fluid Dynamics
ODE—Example

$$f^{n+1} = (1 - \Delta t) f^n$$

Δt=0.5
Δt=1.5
Δt=2.5



$$\left| \frac{f^{n+1}}{f^n} \right| \le 1 \quad \text{only if} \quad \Delta t \le 2$$

Computational Fluid Dynamics
ODE—Example

However

$$f^{n+1} = (1-\Delta t) f^n = (1-\Delta t)^2 f^{n-1}$$
$$= \ldots\ldots\ldots\ldots = (1-\Delta t)^n f^1$$

Obviously, f oscillates unless $\Delta t \le 1$

Backward Euler

$$f^{n+1} = f^n - f^{n+1} \Delta t \quad \rightarrow \quad \frac{f^{n+1}}{f^n} = \frac{1}{(1+\Delta t)}$$

$$\left| \frac{f^{n+1}}{f^n} \right| \le 1 \quad \text{for all } \Delta t$$

Computational Fluid Dynamics

Stability Analysis of the Advection-Diffusion Equation: Von Neumann Method

Generally, stability analysis of the full nonlinear system of equations is too involved to be practical, and we study a model problem that in some way mimics the full equations. The linear advection-diffusion equation is one such model equation, and we will apply von Neumann's method to check the stability of a simple finite difference approximation to that equation.

Consider the 1-D advection-diffusion equation:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = D\frac{\partial^2 f}{\partial x^2}$$

In finite-difference form:

$$\frac{f_j^{n+1} - f_j^n}{\Delta t} + U\frac{f_{j+1}^n - f_{j-1}^n}{2h} = D\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{h^2}$$

Look at the evolution of a small perturbation

$$f_j^n = \varepsilon_j^n$$

The evolution of the perturbation is governed by:

$$\frac{\varepsilon_j^{n+1} - \varepsilon_j^n}{\Delta t} + U\frac{\varepsilon_{j+1}^n - \varepsilon_{j-1}^n}{2h} = D\frac{\varepsilon_{j+1}^n - 2\varepsilon_j^n + \varepsilon_{j-1}^n}{h^2}$$

Write the error as a wave (expand as a Fourier series):

$$\varepsilon_j^n = \varepsilon^n(x_j) = \sum_{k=-\infty}^{\infty} \varepsilon_k^n e^{ikx_j}$$

Dropping the subscript

$$\varepsilon_j^n = \varepsilon^n e^{ikx_j}$$

Recall:

$$e^{ikx} = \cos kx + i\sin kx$$

The error at node j is:

$$\varepsilon_j^n = \varepsilon^n e^{ikx_j}$$

The error at j+1 and j -1 can be written as

$$\varepsilon_{j+1}^n = \varepsilon^n e^{ikx_{j+1}} = \varepsilon^n e^{ik(x_j+h)} = \varepsilon^n e^{ikx_j} e^{ikh}$$

$$\varepsilon_{j-1}^n = \varepsilon^n e^{ikx_{j-1}} = \varepsilon^n e^{ik(x_j-h)} = \varepsilon^n e^{ikx_j} e^{-ikh}$$

Substituting

$$\varepsilon_j^n = \varepsilon^n e^{ikx_j} \qquad \varepsilon_{j+1}^n = \varepsilon^n e^{ikx_j} e^{ikh} \qquad \varepsilon_{j-1}^n = \varepsilon^n e^{ikx_j} e^{-ikh}$$

into

$$\frac{\varepsilon_j^{n+1} - \varepsilon_j^n}{\Delta t} + U\frac{\varepsilon_{j+1}^n - \varepsilon_{j-1}^n}{2h} = D\frac{\varepsilon_{j+1}^n - 2\varepsilon_j^n + \varepsilon_{j-1}^n}{h^2}$$

yields

$$\frac{\varepsilon^{n+1}\cancel{e^{ikx_j}} - \varepsilon^n \cancel{e^{ikx_j}}}{\Delta t} + U\frac{\varepsilon^n}{2h}(e^{ikh}\cancel{e^{ikx_j}} - e^{-ikh}\cancel{e^{ikx_j}}) =$$

$$D\frac{\varepsilon^n}{h^2}(e^{ikh}\cancel{e^{ikx_j}} - 2\cancel{e^{ikx_j}} + e^{-ikh}\cancel{e^{ikx_j}})$$

The equation for the error is:

$$\frac{\varepsilon^{n+1} - \varepsilon^n}{\Delta t} + U\frac{\varepsilon^n}{2h}(e^{ikh} - e^{-ikh}) = D\frac{\varepsilon^n}{h^2}(e^{ikh} - 2 + e^{-ikh})$$

Solving for the ratio of the errors:

$$\frac{\varepsilon^{n+1}}{\varepsilon^n} = 1 - \frac{U\Delta t}{2h}(e^{ikh} - e^{-ikh}) + \frac{D\Delta t}{h^2}(e^{ikh} - 2 + e^{-ikh})$$

Computational Fluid Dynamics
Stability

Dividing by the error amplitude at n:

amplification factor

$$\frac{\varepsilon^{n+1}}{\varepsilon^{n}}=1-\frac{U\Delta t}{2h}(e^{ikh}-e^{-ikh})+\frac{D\Delta t}{h^2}(e^{ikh}-2+e^{-ikh})$$

$$=1-\frac{U\Delta t}{h}i\sin kh+\frac{D\Delta t}{h^2}2(\cos kh-1)$$

$$=1-4\frac{D\Delta t}{h^2}\sin^2 k\frac{h}{2}-i\frac{U\Delta t}{h}\sin kh$$

Using:

$$e^{ikh}+e^{-ikh}=2\cos kh;\quad e^{ikh}-e^{-ikh}=i2\sin kh;\qquad 2\sin^2\theta=1-\cos 2\theta$$

Computational Fluid Dynamics
Stability

The ratio of the error amplitude at n+1 and n is:

$$\frac{\varepsilon^{n+1}}{\varepsilon^{n}}=1-4\frac{D\Delta t}{h^2}\sin^2 k\frac{h}{2}-i\frac{U\Delta t}{h}\sin kh$$

Stability requires that

$$\left|\frac{\varepsilon^{n+1}}{\varepsilon^{n}}\right|\le 1$$

Since the amplification factor is a complex number, and k, the wave number of the error, can be anything, the determination of the stability limit is slightly involved.

We will look at two special cases: (a) $U=0$ and (b) $D=0$

Computational Fluid Dynamics
Stability

(a) Consider first the case when $U=0$, so the problem reduces to a pure diffusion

$$\frac{\varepsilon^{n+1}}{\varepsilon^{n}}=1-4\frac{D\Delta t}{h^2}\sin^2 k\frac{h}{2}$$

Since $\sin^2()\le 1$ the amplification factor is always less than 1, and we find that it is bigger than -1 if

$$-1\le 1-4\frac{D\Delta t}{h^2}\le 1$$

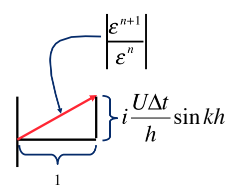
$$\boxed{\frac{D\Delta t}{h^2}\le\frac{1}{2}}$$

Computational Fluid Dynamics
Stability

(b) Consider now the other limit where $D=0$ and we have a pure advection problem.

$$\frac{\varepsilon^{n+1}}{\varepsilon^{n}}=1-i\frac{U\Delta t}{h}\sin kh$$

Since the amplification factor has the form 1+i() the absolute value of this complex number is always larger than unity and the method is unconditionally unstable for this case.



Computational Fluid Dynamics
Stability

For the general case we must investigate the stability condition in more detail. We will not do so here, but simply quote the results:

$$\boxed{\frac{\Delta tD}{h^2}\le\frac{1}{2}\quad\text{and}\quad\frac{U^2\Delta t}{D}\le 2}$$

Notice that high velocity and low viscosity lead to instability according to the second restriction.

Computational Fluid Dynamics
Stability

For a two-dimensional problem, assume an error of the form

$$\varepsilon_{i,j}^{n}=\varepsilon^{n}e^{i(kx_i+ly_j)}$$

A stability analysis gives:

$$\boxed{\frac{D\Delta t}{h^2}\le\frac{1}{4}\quad\text{and}\quad\frac{(|U|+|V|)^2\Delta t}{D}\le 4}$$

For a three-dimensional problem we get:

$$\boxed{\frac{D\Delta t}{h^2}\le\frac{1}{6}\quad\text{and}\quad\frac{(|U|+|V|+|W|)^2\Delta t}{D}\le 8}$$

Computational Fluid Dynamics
Stability

Stability – Now you know!

Convergence – the solution to the finite-difference equation approaches the true solution to the PDE having the same initial and boundary conditions as the mesh is refined.

Lax's Equivalence Theorem

Given a properly posed initial value problem and a finite-difference approximation to it that satisfies the consistency condition, stability is the necessary and sufficient condition for convergence.

Computational Fluid Dynamics

The Modified Equation

Computational Fluid Dynamics
Consistency

Using the finite difference approximation, we are effectively solving an equation that is slightly different than the original partial differential equations.

Does the finite difference equation approach the partial differential equation in the limit of zero Δt and h?

Computational Fluid Dynamics
Consistency

Consider the 1-D advection-diffusion equation

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = D\frac{\partial^2 f}{\partial x^2}$$

and its finite-difference approximation

$$\frac{f_j^{n+1} - f_j^n}{\Delta t} + U\frac{f_{j+1}^n - f_{j-1}^n}{2h} = D\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{h^2}$$

The discrepancy between the two equations can be found by deriving the modified equation.

Computational Fluid Dynamics
Consistency

Substituting

$$\frac{f_j^{n+1} - f_j^n}{\Delta t} = \frac{\partial f(t)}{\partial t} + \frac{\partial^2 f(t)}{\partial t^2}\frac{\Delta t}{2} + \cdots$$

$$\frac{f_{j+1}^n - f_{j-1}^n}{2h} = \frac{\partial f(x)}{\partial x} + \frac{\partial^3 f(x)}{\partial x^3}\frac{h^2}{6} + \cdots$$

$$\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{h^2} = \frac{\partial^2 f(x)}{\partial x^2} + \frac{\partial^4 f(x)}{\partial x^4}\frac{h^2}{12} + \cdots$$

into the finite difference equation

$$\frac{f_j^{n+1} - f_j^n}{\Delta t} + U\frac{f_{j+1}^n - f_{j-1}^n}{2h} = D\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{h^2}$$

Computational Fluid Dynamics
Consistency

Results in the Modified Equation:

$$\underbrace{\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} - D\frac{\partial^2 f}{\partial x^2}}_{\text{Original Equation}} = \underbrace{-\frac{\partial^2 f(t)}{\partial t^2}\frac{\Delta t}{2} - U\frac{\partial^3 f(x)}{\partial x^3}\frac{h^2}{6} + D\frac{\partial^4 f(x)}{\partial x^4}\frac{h^2}{12}}_{\text{Error terms}} + \cdots$$

Shorthand:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} - D\frac{\partial^2 f}{\partial x^2} = O(\Delta t, h^2)$$

In this case, the error goes to zero as $h \to 0$ and $\Delta t \to 0$, so the approximation is said to be CONSISTENT

Although most finite difference approximations are consistent, innocent-looking modifications can sometimes lead to approximations that are not!

The Frankel-Dufort is an example of an non-consistent scheme.

You will examine it in the homework

HW: Examine the Frankel-Dufort method

Solve the diffusion equation

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}$$

Using the Leapfrog time integration method and standard finite-difference approximation for the spatial derivative gives:

$$\frac{f_j^{n+1} - f_j^{n-1}}{2\Delta t} = \frac{D}{h^2}\left[f_{j+1}^n - 2f_j^n + f_{j-1}^n\right]$$

Leapfrog method

Now modify it slightly:

$$\frac{f_j^{n+1} - f_j^{n-1}}{2\Delta t} = \frac{D}{h^2}\left[f_{j+1}^n - 2f_j^n + f_{j-1}^n\right]$$

Replace by: $f_j^n = \frac{1}{2}\left(f_j^{n+1} + f_j^{n-1}\right)$

This gives:

$$f_j^{n+1} = f_j^{n-1} + 2\frac{\Delta t\, D}{h^2}\left(f_{j+1}^n - f_j^{n+1} - f_j^{n-1} + f_{j-1}^n\right).$$

Which is easily solved for f at the new time step

In the HW, you will examine the error!

The MODIFIED EQUATION is obtained by substituting the expression for the finite difference approximations, including the error terms, into the finite difference equation. For a CONSISTENT finite difference approximation the error terms go to zero as $h \to 0$ and $\Delta t \to 0$.

The modified equation can often be used to infer the nature of the error of the finite difference scheme. More about that later.

Two-Dimensional Advection-Diffusion Equation

We will use the model equation:

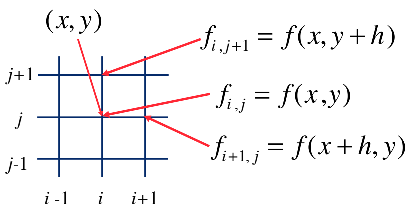
$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} + V\frac{\partial f}{\partial y} = D\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right)$$

to demonstrate how to solve a partial equation (initial value problem) numerically.

The extension to two-dimensions is relatively straight forward, once the one-dimensional problem is fully understood.

Computational Fluid Dynamics
Multidimensional Equations

For a two-dimensional flow discretize the variables on a two-dimensional grid



Computational Fluid Dynamics
Multidimensional Equations

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} + V\frac{\partial f}{\partial y} = D\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right)$$

The discrete equation is:

$$\frac{f_{i,j}^{n+1} - f_{i,j}^n}{\Delta t} = -U\left(\frac{f_{i+1,j}^n - f_{i-1,j}^n}{2h}\right) - V\left(\frac{f_{i,j+1}^n - f_{i,j-1}^n}{2h}\right) +$$
$$D\left(\frac{f_{i+1,j}^n + f_{i-1,j}^n - 2f_{i,j}^n}{h^2} + \frac{f_{i,j+1}^n + f_{i,j-1}^n - 2f_{i,j}^n}{h^2}\right)$$

Computational Fluid Dynamics
Multidimensional Equations

Solve for $f_{i,j}^{n+1}$

$$f_{i,j}^{n+1} = f_{i,j}^n + \Delta t\left[-U\left(\frac{f_{i+1,j}^n - f_{i-1,j}^n}{2h}\right) - V\left(\frac{f_{i,j+1}^n - f_{i,j-1}^n}{2h}\right) + \right.$$
$$\left. D\left(\frac{f_{i+1,j}^n + f_{i-1,j}^n + f_{i,j+1}^n + f_{i,j-1}^n - 4f_{i,j}^n}{h^2}\right)\right]$$

or

$$f_{i,j}^{n+1} = f_{i,j}^n - \frac{\Delta t U}{2h}\left(f_{i+1,j}^n - f_{i-1,j}^n\right) - \frac{\Delta t V}{2h}\left(f_{i,j+1}^n - f_{i,j-1}^n\right)$$
$$+ \frac{\Delta t D}{h^2}\left(f_{i+1,j}^n + f_{i-1,j}^n + f_{i,j+1}^n + f_{i,j-1}^n - 4f_{i,j}^n\right)$$

Accuracy: $O\left(\Delta t, h^2\right)$

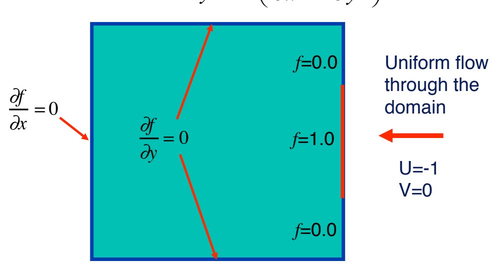
A stability analysis gives: $\frac{\Delta t D}{h^2} \le \frac{1}{4}$ and $\frac{(|U|+|V|)^2 \Delta t}{D} \le 4$

Computational Fluid Dynamics

Example

Computational Fluid Dynamics
Multidimensional Equations

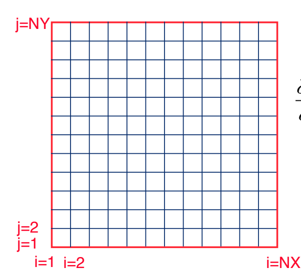
$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} + V\frac{\partial f}{\partial y} = D\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right)$$



Computational Fluid Dynamics
Multidimensional Equations

$$f_{i,j}^{n+1} = f_{i,j}^n - \frac{\Delta t U}{2h}\left(f_{i+1,j}^n - f_{i-1,j}^n\right) + \frac{\Delta t D}{h^2}\left(f_{i+1,j}^n + f_{i-1,j}^n + f_{i,j+1}^n + f_{i,j-1}^n - 4f_{i,j}^n\right)$$

$f_{i,j}$ stored at each grid point



$\frac{\partial f}{\partial n}$ or f given on the boundary

Computational Fluid Dynamics
Multidimensional Equations

Boundary conditions

Where f is given, we simply specify its value

Where the normal derivative is specified, we approximate the value at the boundary by one-sided differences

At the $i=1$ boundary, for example, $\frac{\partial f}{\partial y} = 0$

and by using $\frac{\partial f}{\partial y} \approx \frac{f_{i,2}^n - f_{i,1}^n}{h} = 0$

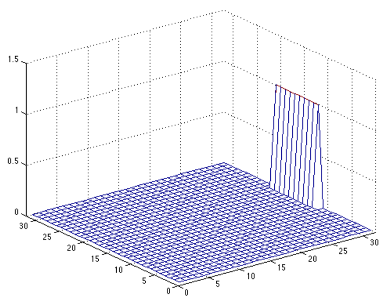
we find that: $f_{i,2}^n = f_{i,1}^n$

Computational Fluid Dynamics
Multidimensional Equations

```
% two-dimensional unsteady diffusion by the FTCS scheme
%-----------------------------------------------------------
n=32;m=32;nstep=120;D=0.025;length=2.0;h=length/(n-1);
dt=1.0*0.125*h*h/D;f=zeros(n,m);fo=zeros(n,m);time=0.0;
u=-0.0; v=-1.0; f(12:21,n)=1.0;
for l=1:nstep,l,time
hold off;mesh(f); axis([0 n 0 m 0 1.5]);pause;
  fo=f;
  for i=2:n-1, for j=2:m-1
    f(i,j)=fo(i,j)-(0.5*dt*u/h)*(fo(i+1,j)-fo(i-1,j))-...
                          (0.5*dt*v/h)*(fo(i,j+1)-fo(i,j-1))+...
    (D*dt/h^2)*(fo(i+1,j)+ fo(i,j+1)+fo(i-1,j)+fo(i,j -1)-4*fo(i,j));
 end,end
 for i=1:n, f(i,1)=f(i,2);end;for j=1:m,f(1,j)=f(2,j);f(m,j)=f(m-1,j);end;
 time=time+dt;
end;
```

Computational Fluid Dynamics
Multidimensional Equations

The unsteady evolution of the solution



Computational Fluid Dynamics

Multidimensional Boundary Value Problems (Steady-State)

Computational Fluid Dynamics
Boundary Value Problems

Consider the Poisson Equation:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = S$$

This equation has a solution if f or $\partial f / \partial n$ is specified on the boundary

Use standard finite differences to discretize:

$$\frac{f_{i+1,j} + f_{i-1,j} - 2f_{i,j}}{h^2} + \frac{f_{i,j-1} + f_{i,j+1} - 2f_{i,j}}{h^2} = S_{i,j}$$

Computational Fluid Dynamics
Boundary Value Problems

For uniform grids:

$$\frac{f_{i+1,j} + f_{i-1,j} - 2f_{i,j}}{h^2} + \frac{f_{i,j-1} + f_{i,j+1} - 2f_{i,j}}{h^2} = S_{i,j}$$

can be written as

$$\frac{f_{i+1,j} + f_{i-1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2} = S$$

Solve for $f_{i,j}$:

$$f_{i,j} = \frac{1}{4}\left(f_{i+1,j} + f_{i-1,j} + f_{i,j-1} + f_{i,j+1} - h^2 S_{i,j}\right)$$

Computational Fluid Dynamics
Boundary Value Problems

Solve for $f_{i,j}$ and use the right hand side to compute a new value. Denote the old values by α and the new ones with α+1

$$f_{i,j}^{\alpha+1} = \frac{1}{4}\left(f_{i+1,j}^{\alpha} + f_{i-1,j}^{\alpha} + f_{i,j-1}^{\alpha} + f_{i,j+1}^{\alpha} - h^2 S_{i,j}\right)$$

This iteration process—Jacobi iteration—is very robust but many iterations are required to reach an accurate solution.

Computational Fluid Dynamics
Boundary Value Problems

The iteration must be carried out until the solution is sufficiently accurate. To measure the error, define the residual:

$$R_{i,j} = \frac{f_{i+1,j} + f_{i-1,j} + f_{i,j-1} + f_{i,j+1} - 4f_{i,j}}{h^2} - S_{i,j}$$

At steady-state the residual should be zero. The point-wise residual or the average absolute residual can be used, depending on the problem. Often, simpler criteria, such as the change from one iteration to the next is used

Computational Fluid Dynamics
Boundary Value Problems

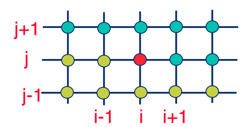
Although the Jacobi iteration is a very robust iteration technique, it converges VERY slowly.

We therefore seek a way to ACCELERATE the convergence to steady-state, making use of the fact that it is only the steady-state that is of interest.

Here we introduce the Gauss-Seidler method and the Successive Over-Relaxation (SOR) method.

Computational Fluid Dynamics
Boundary Value Problems

The Jacobi iteration can be improved somewhat by using new values as soon as they become available.




```
for j=1:m
for i=1:n
    iterate
end
end
```

$$f_{i,j}^{\alpha+1} = \frac{1}{4}(f_{i+1,j}^{\alpha} + f_{i-1,j}^{\alpha+1} + f_{i,j+1}^{\alpha} + f_{i,j-1}^{\alpha+1} - h^2 S_{i,j})$$

From a programming point of view, Gauss-Seidler iteration is even simpler than Jacobi iteration since only one vector with f values is needed.

Computational Fluid Dynamics
Boundary Value Problems

The Gauss-Seidler iteration can be accelerated even further by various acceleration techniques. The simplest one is the Successive Over-Relaxation (SOR) iteration

$$f_{i,j}^{\alpha+1} = \frac{\beta}{4}(f_{i+1,j}^{\alpha} + f_{i-1,j}^{\alpha+1} + f_{i,j+1}^{\alpha} + f_{i,j-1}^{\alpha+1} - h^2 S_{i,j}) + (1-\beta) f_{i,j}^{\alpha}$$

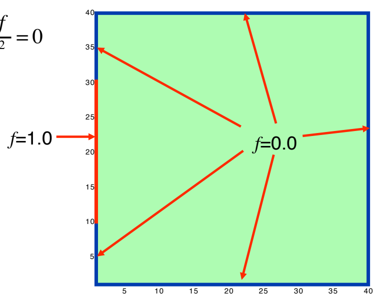
The SOR iteration is very simple to program, just as the Gauss-Seidler iteration. The user must select the coefficient. It must be bounded by 1<β<2. β=1.5 is usually a good starting value.

Computational Fluid Dynamics

Example

Computational Fluid Dynamics
Boundary Value Problems

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$



Computational Fluid Dynamics
Boundary Value Problems

```
% two-dimensional steady-state problem by SOR
n=40;m=40;iterations=5000;length=2.0;h=length/(n-1);
T=zeros(n,m);bb=1.7;
T(10:n-10,1)=1.0;
for l=1:iterations,
   for i=2:n-1, for j=2:m-1
      T(i,j)=bb*0.25*(T(i+1,j)+...
             T(i,j+1)+T(i-1,j)+T(i,j-1))+(1.0-bb)*T(i,j);
   end,end
   % find residual
   res=0;
   for i=2:n-1, for j=2:m-1
      res=res+abs(T(i+1,j)+...
             T(i,j+1)+T(i-1,j)+T(i,j-1)-4*T(i,j))/h^2;
   end,end
   l,res/((m-2)*(n-2)) % Print iteration and residual
   if (res/((m-2)*(n-2)) < 0.001), break,end
end;
contour(T);
```

Computational Fluid Dynamics
Boundary Value Problems

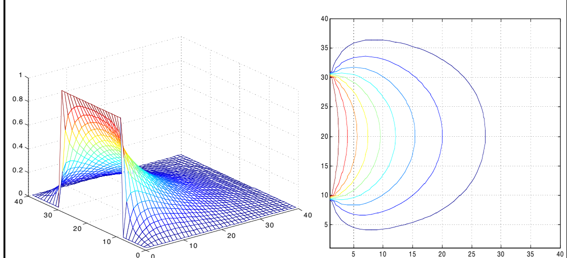
The program is easily modified for the Jacobi and the Gauss-Seidler iteration:

Average absolute error: 0.001

	Number of iterations
Jacobi:	1989
Gauss-Seidler:	986
SOR (β =1.5):	320
SOR (β = 1.7):	162
SOR (β = 1.9):	91
SOR (β = 1.95):	202

Computational Fluid Dynamics
Boundary Value Problems

The converged solution: $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$



Computational Fluid Dynamics
Summary

Accuracy—smaller time step and finer resolution should get us the exact solution

Stability—introduced the von Neumann method. Fairly mechanical process, we will provide more insight by the finite volume point of view

The modified equations helps us see how the approximate and the exact equation differ and if the former is consistent with the latter

Multidimensional advection-diffusion equation. Essentially the same as the one-dimensional problem

Iterative methods for boundary value problems. Elementary approaches to steady state problems