Lecture 4a – Programming a Single-Block Simulation Code

- **Engineers develop and use various simulation codes to perform analysis and design**
- **As the complexity of physical problems increases, the likelihood that these analyses are used on parallel computers greatly increases**
  - Knowing how to develop and/or use parallel computers to do these engineering tasks will give you skills that will enhance your career
- **One of the objectives of this course is to take you through the steps that are used to develop a parallel simulation code**
  - Develop a single-block structured simulation code
  - Develop a data-structure to support multiple blocks
  - Develop a multi-block structured simulation code
  - Develop a domain-decomposition capability
  - Develop a parallel, multi-block structured simulation code

Programming a Simulation Code

- **Numerical engineering analyses range in complexity from**
  - simple algebraic tasks → solving a differential equation → solving multiple differential equations → solving multiple sets of differential equations for different disciplines
- **Many simple algebraic analyses, such as data reduction, are "embarrassingly parallel" in that the data of a given point does not depend on other points**
- **However, most differential equation solvers rely on finite-differencing, finite-volume, or finite-element numerical schemes that require that the data at a given point is dependent on its surrounding set of points.**
- **You will learn much more by developing a parallel simulation code for this type of problem**

Programming a Simulation Code

- **For this class, you will be asked to program a parallel simulation code that solves the heat conduction equation.**
  - A single, second-order, partial differential equation for the thermal field of a structure
  - If you have NOT had any background/experience in numerical methods, don't worry….I will go through the Poisson/Laplace equation with you and give the basics needed to perform this task

Programming a Simulation Code

- **Many engineering analyses require the solution of a single differential equation.**
- **A good example of this is the Poisson/Laplace equation used for:**
  - Stream-function (incompressible fluid dynamics)
  - Potential flow (incompressible, irrotational fluid dynamics)
  - Heat conduction (heat transfer)
  - Wave or Helmholtz equation (acoustics)
  - Grid generation (smoothing a computational grid)

Example: The Stream Function For Irrotational (Inviscid) Flow

- **For irrotational flow, the curl of the velocity is zero**

$$\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}=0 \quad let's\ define: \quad u=\frac{\partial \psi}{\partial y} \quad and \quad v=-\frac{\partial \psi}{\partial x}$$

- **Substituting the definitions of the velocity components in terms of the stream function leads to:**

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

- **Laplace's equation with boundary conditions:**
  - At infinity: $\psi = U_\infty y + const$
  - At solid body: $\psi = const$

What is the Stream Function Physically?

- **Lines of constant ψ are streamlines of the flow!**
- **If an elemental arc-length of a 2-D streamline is to be parallel to the velocity vector, then the velocity components must be in proportion to:**

$$\frac{dx}{u}=\frac{dy}{v}$$

or $$u\,dy - v\,dx = 0$$

- **Substituting the stream function for the velocity components leads to:**

$$\frac{\partial \psi}{\partial x}dx+\frac{\partial \psi}{\partial y}dy=0=d\psi \qquad or$$

ψ = constant along a streamline

Example: The Velocity Potential

- **We can define another function, called the velocity potential, ϕ, that satisfies the irrotationality condition (instead of continuity as the stream function did).**
  - Note that the stream function only forced us to 2-dimensional problems but with no other constraints
    - (must use 2 stream functions for 3-D flow!)
  - Introduction of the velocity potential restricts us to irrotational flow but we can deal with 3-D flows directly!
    - However, there aren't many 3-D irrotational flows in reality

$$u=\frac{\partial \phi}{\partial x} \qquad v=\frac{\partial \phi}{\partial y} \qquad w=\frac{\partial \phi}{\partial z} \qquad or \qquad V=\nabla\phi$$

$$curl\ V=\nabla\times V=0$$

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial \phi}{\partial x} & \frac{\partial \phi}{\partial y} & \frac{\partial \phi}{\partial z} \end{vmatrix} = \left(\frac{\partial^2\phi}{\partial y\partial z}-\frac{\partial^2\phi}{\partial z\partial y}\right)\hat{i}+\left(\frac{\partial^2\phi}{\partial z\partial x}-\frac{\partial^2\phi}{\partial x\partial z}\right)\hat{j}+\left(\frac{\partial^2\phi}{\partial x\partial y}-\frac{\partial^2\phi}{\partial y\partial x}\right)\hat{k}=0$$

Velocity Potential

- **We know that the velocity potential automatically satisfies the irrotationality condition.**
- **Let's see what happens to continuity:**
  - For incompressible flow:

$$\nabla\cdot V=\nabla\cdot(\nabla\phi)=0 \quad or$$

$$\nabla^2\phi=0=\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}$$

- **Laplace's equation which is fairly easy to solve**

Orthogonality of Streamlines and Potential Lines

- **By comparing the stream function with the velocity potential, we see**

$$u = \frac{\partial \psi}{\partial y} = \frac{\partial \phi}{\partial x}$$

$$v = -\frac{\partial \psi}{\partial x} = \frac{\partial \phi}{\partial y}$$

or $d\psi = \left(\frac{\partial \psi}{\partial y} dy\right)\hat{i} + \left(\frac{\partial \psi}{\partial x} dx\right)\hat{j} = du\,dy - dv\,dx = 0$ along line of constant ψ

$$d\phi = \left(\frac{\partial \phi}{\partial x} dx\right)\hat{i} + \left(\frac{\partial \phi}{\partial y} dy\right)\hat{j} = du\,dx + dv\,dy = 0 \text{ along line of constant } \phi$$

- **The condition for orthogonality between ψ and ϕ is that**

$$\left(\frac{dy}{dx}\right)_{\phi=constant} = -\frac{u}{v} = -\frac{1}{(dy/dx)_{\psi=constant}}$$

Example: Heat Conduction

- **The temperature distribution through a solid is governed by the transient heat conduction equation**

$$\rho c_p \frac{\partial T}{\partial t} = k\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right]$$

With boundary conditions:

At edges: $T(x,y) = known$ or

$$\frac{\partial T}{\partial n}(x,y) = known$$

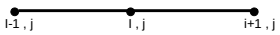
- **For steady state:**

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

Laplace's equation again

Finite Difference Representation

- **One can represent derivatives of differential equations as finite-differences at some level of accuracy based upon series expansions**
- **Usually, we would use standard second-order accurate (error of O[Δx³, Δy³]), central difference operators**



$$\frac{\partial \phi}{\partial x} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})} + O\left[\Delta x^3\right] \quad central$$

$$= \frac{\phi_{i+1,j} - \phi_{i,j}}{(x_{i+1,j} - x_{i,j})} + O\left[\Delta x^2\right] \quad forward$$

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2} + O\left[\Delta x^4\right]$$

Finite Difference Representation

- **So we could take a Laplace equation, like the steady heat conduction equation, and write it in terms of finite-differences (for simple plates)**

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

$$\left[\frac{T^n_{i+1.j} - 2T^n_{i,j} + T^n_{i-1,j}}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2} + \frac{T^n_{i.j+1} - 2T^n_{i,j} + T^n_{i,j-1}}{(y_{i+\frac{1}{2},j} - y_{i-\frac{1}{2},j})^2}\right] = 0$$

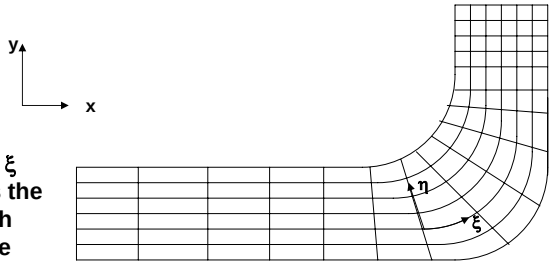
We can take the $T_{i,j}$ term and write it at the new time level giving

$$T^{n+1}_{i,j} = \frac{1}{2}\left[\frac{T^n_{i+1,j} + T^n_{i-1,j}}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2} + \frac{T^n_{i,j+1} + T^n_{i,j-1}}{(y_{i,j+\frac{1}{2}} - y_{i,j-\frac{1}{2}})^2}\right]\left[\frac{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2 (y_{i,j+\frac{1}{2}} - y_{i,j-\frac{1}{2}})^2}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2 + (y_{i,j+\frac{1}{2}} - y_{i,j-\frac{1}{2}})^2}\right]$$

and we could write the RHS terms to all exist at time level n (making the scheme EXPLICIT). This is the point-Jacobi numerical method.

Finite Difference Representation

- **This "direct" type of finite differencing works fine if the geometry is very simple (rectangular).**
- **If the geometry is distorted or has large variations in angle, then the finite difference equations must be written in the curvilinear coordinate system OR the equations must be integrated and discretized using a finite-volume procedure.**



Think of ξ and η as the arclength along the lines

You can see from this example how a "direct" finite difference could lead to a divide by zero when the differencing lines change direction

Finite Difference Representation

- **So in the curvilinear coordinate system (ξ,η) where ξ and η are functions of x and y**

$$\frac{\partial}{\partial \xi} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \xi}$$
$$\frac{\partial}{\partial \eta} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \eta}$$

or

$$\begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

Jacobian, J

- **And we can find the inverse to the Jacobian matrix to give**

$$J^{-1} = \frac{1}{|J|}\begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \qquad |J| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}$$

Finite Difference Representation

- **So we have**

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{1}{|J|}\begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix}$$

or

$$\frac{\partial}{\partial x} = \frac{1}{|J|}\left[\frac{\partial}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial}{\partial \eta}\frac{\partial y}{\partial \xi}\right]$$
$$\frac{\partial}{\partial y} = \frac{1}{|J|}\left[-\frac{\partial}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial}{\partial \eta}\frac{\partial x}{\partial \xi}\right]$$

$$|J| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}$$

Finite Difference Representation

- **And for second derivatives, we can use the chain rule of differentiation to get:**

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{1}{|J|^2}\left[(y_\eta)^2 \phi_{\xi\xi} - 2y_\xi y_\eta \phi_{\xi\eta} + (y_\xi)^2 \phi_{\eta\eta} + (y_\eta y_{\xi\eta} - y_\eta y_{\eta\eta})\phi_\xi + (y_\xi y_{\xi\eta} - y_\eta y_{\xi\xi})\phi_\eta\right]$$

$$\frac{\partial^2 \phi}{\partial y^2} = \frac{1}{|J|^2}\left[(x_\eta)^2 \phi_{\xi\xi} - 2x_\xi x_\eta \phi_{\xi\eta} + (x_\xi)^2 \phi_{\eta\eta} + (x_\eta x_{\xi\eta} - x_\xi x_{\eta\eta})\phi_\xi - (x_\xi x_{\xi\eta} - x_\eta x_{\xi\xi})\phi_\eta\right]$$

- **You can see that the governing equation becomes much more complicated in the transformed coordinate system!**

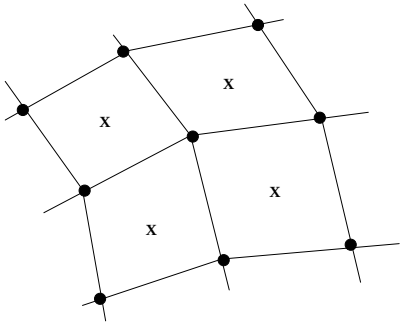
Finite Volume Representation

- **We can avoid dealing with Jacobians directly by *integrating* the governing equations instead of using finite-differences. This results in a control (finite) volume procedure.**
- **From Green's theorem (integrating in the counter-clockwise direction):**

$$\int_{Vol}\left[\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right]dVol = 0$$

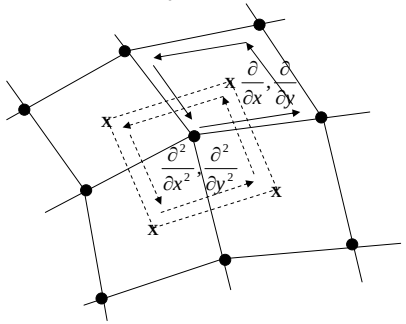
$$\frac{\partial \phi}{\partial x} = \frac{1}{Vol}\oint_{CV}\phi\, dy$$

$$\frac{\partial \phi}{\partial y} = -\frac{1}{Vol}\oint_{CV}\phi\, dx$$



Finite Volume Representation

- **We can apply the finite volume discretization to the steady heat conduction equation by applying Green's integral twice**
  - Around the primary control volume (with the nodes as the verticies) to find the first derivative
  - Then around the secondary control volume (with the cell-centers as the verticies) to find the second derivative



Finite Volume Representation

- **So let's just look at the x-derivative terms for now**

$$\left.\frac{\partial T}{\partial x}\right|_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2Vol_{i+\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}(T_{i,j}+T_{i+1,j})(y_{i+1,j}-y_{i,j})+(T_{i+1,j}+T_{i+1,j+1})(y_{i+1,j+1}-y_{i+1,j})+\\(T_{i,j+1}+T_{i+1,j+1})(y_{i,j+1}-y_{i+1,j+1})+(T_{i,j}+T_{i,j+1})(y_{i,j}-y_{i,j+1})\end{array}\right]$$

$$\left.\frac{\partial T}{\partial x}\right|_{i-\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2Vol_{i-\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}(T_{i-1,j}+T_{i,j})(y_{i,j}-y_{i-1,j})+(T_{i,j}+T_{i,j+1})(y_{i,j+1}-y_{i,j})+\\(T_{i-1,j+1}+T_{i,j+1})(y_{i-1,j+1}-y_{i,j+1})+(T_{i-1,j}+T_{i-1,j+1})(y_{i-1,j}-y_{i-1,j+1})\end{array}\right]$$

$$\left.\frac{\partial T}{\partial x}\right|_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{1}{2Vol_{i-\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}(T_{i-1,j-1}+T_{i,j-1})(y_{i,j-1}-y_{i-1,j-1})+(T_{i,j-1}+T_{i,j})(y_{i,j}-y_{i,j-1})+\\(T_{i-1,j}+T_{i,j})(y_{i-1,j}-y_{i,j})+(T_{i-1,j-1}+T_{i-1,j})(y_{i-1,j-1}-y_{i-1,j})\end{array}\right]$$

$$\left.\frac{\partial T}{\partial x}\right|_{i+\frac{1}{2},j-\frac{1}{2}} = \frac{1}{2Vol_{i+\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}(T_{i,j-1}+T_{i+1,j-1})(y_{i+1,j-1}-y_{i,j-1})+(T_{i+1,j-1}+T_{i+1,j})(y_{i+1,j}-y_{i+1,j-1})+\\(T_{i,j}+T_{i+1,j})(y_{i,j}-y_{i+1,j})+(T_{i,j-1}+T_{i,j})(y_{i,j-1}-y_{i,j})\end{array}\right]$$

Finite Volume Representation

- **And likewise, the second derivative is:**

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{i,j} = \frac{1}{Vol_{i,j}}\left[\begin{array}{l}\frac{\left(\frac{\partial T_{i+\frac{1}{2},j+\frac{1}{2}}}{\partial x}+\frac{\partial T_{i-\frac{1}{2},j+\frac{1}{2}}}{\partial x}\right)}{2}\left(y_{i+\frac{1}{2},j+\frac{1}{2}}-y_{i-\frac{1}{2},j+\frac{1}{2}}\right)+\frac{\left(\frac{\partial T_{i-\frac{1}{2},j+\frac{1}{2}}}{\partial x}+\frac{\partial T_{i-\frac{1}{2},j-\frac{1}{2}}}{\partial x}\right)}{2}\left(y_{i-\frac{1}{2},j+\frac{1}{2}}-y_{i-\frac{1}{2},j-\frac{1}{2}}\right)+\\\frac{\left(\frac{\partial T_{i-\frac{1}{2},j-\frac{1}{2}}}{\partial x}+\frac{\partial T_{i+\frac{1}{2},j-\frac{1}{2}}}{\partial x}\right)}{2}\left(y_{i-\frac{1}{2},j-\frac{1}{2}}-y_{i+\frac{1}{2},j-\frac{1}{2}}\right)+\frac{\left(\frac{\partial T_{i+\frac{1}{2},j-\frac{1}{2}}}{\partial x}+\frac{\partial T_{i+\frac{1}{2},j+\frac{1}{2}}}{\partial x}\right)}{2}\left(y_{i+\frac{1}{2},j-\frac{1}{2}}-y_{i+\frac{1}{2},j+\frac{1}{2}}\right)\end{array}\right]$$

Finite Volume Representation

- **If we substituted the first derivatives, we get:**

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{i,j} = \frac{1}{2Vol_{i,j}}\left[\begin{array}{l}
\left(\frac{1}{2Vol_{i+\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}\left(\boldsymbol{T_{i,j}}+T_{i+1,j}\right)\left(y_{i+1,j}-y_{i,j}\right)+\left(T_{i+1,j}+T_{i+1,j+1}\right)\left(y_{i+1,j+1}-y_{i+1,j}\right)+\\ \left(T_{i,j+1}+T_{i+1,j+1}\right)\left(y_{i,j+1}-y_{i+1,j+1}\right)+\left(\boldsymbol{T_{i,j}}+T_{i,j+1}\right)\left(y_{i,j}-y_{i,j+1}\right)\end{array}\right]+\right.\\
\left.\left(\frac{1}{2Vol_{i-\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i-1,j}+\boldsymbol{T_{i,j}}\right)\left(y_{i,j}-y_{i-1,j}\right)+\left(\boldsymbol{T_{i,j}}+T_{i,j+1}\right)\left(y_{i,j+1}-y_{i,j}\right)+\\ \left(T_{i-1,j+1}+T_{i,j+1}\right)\left(y_{i-1,j+1}-y_{i,j+1}\right)+\left(T_{i-1,j}+T_{i-1,j+1}\right)\left(y_{i-1,j}-y_{i-1,j+1}\right)\end{array}\right]\right)\left(y_{i+\frac{1}{2},j+\frac{1}{2}}-y_{i-\frac{1}{2},j+\frac{1}{2}}\right)+\\
\left(\frac{1}{2Vol_{i-\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i-1,j}+\boldsymbol{T_{i,j}}\right)\left(y_{i,j}-y_{i-1,j}\right)+\left(\boldsymbol{T_{i,j}}+T_{i,j+1}\right)\left(y_{i,j+1}-y_{i,j}\right)+\\ \left(T_{i-1,j+1}+T_{i,j+1}\right)\left(y_{i-1,j+1}-y_{i,j+1}\right)+\left(T_{i-1,j}+T_{i-1,j+1}\right)\left(y_{i-1,j}-y_{i-1,j+1}\right)\end{array}\right]+\right.\\
\left.\left(\frac{1}{2Vol_{i-\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i-1,j-1}+T_{i,j-1}\right)\left(y_{i,j-1}-y_{i-1,j-1}\right)+\left(T_{i,j-1}+\boldsymbol{T_{i,j}}\right)\left(y_{i,j}-y_{i,j-1}\right)+\\ \left(T_{i-1,j}+\boldsymbol{T_{i,j}}\right)\left(y_{i-1,j}-y_{i,j}\right)+\left(T_{i-1,j-1}+T_{i-1,j}\right)\left(y_{i-1,j-1}-y_{i-1,j}\right)\end{array}\right]\right)\left(y_{i-\frac{1}{2},j+\frac{1}{2}}-y_{i-\frac{1}{2},j-\frac{1}{2}}\right)+\\
\left(\frac{1}{2Vol_{i-\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i,j-1}+T_{i+1,j-1}\right)\left(y_{i+1,j-1}-y_{i,j-1}\right)+\left(T_{i+1,j-1}+\boldsymbol{T_{i,j}}\right)\left(y_{i+1,j}-y_{i,j-1}\right)+\\ \left(\boldsymbol{T_{i,j}}+T_{i+1,j}\right)\left(y_{i,j}-y_{i+1,j}\right)+\left(T_{i,j-1}+\boldsymbol{T_{i,j}}\right)\left(y_{i,j-1}-y_{i,j}\right)\end{array}\right]+\right.\\
\left.\left(\frac{1}{2Vol_{i+\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i,j-1}+T_{i+1,j-1}\right)\left(y_{i+1,j-1}-y_{i,j-1}\right)+\left(T_{i+1,j-1}+T_{i+1,j}\right)\left(y_{i+1,j}-y_{i+1,j-1}\right)+\\ \left(\boldsymbol{T_{i,j}}+T_{i+1,j}\right)\left(y_{i,j}-y_{i+1,j}\right)+\left(T_{i,j-1}+\boldsymbol{T_{i,j}}\right)\left(y_{i,j-1}-y_{i,j}\right)\end{array}\right]\right)\left(y_{i+\frac{1}{2},j-\frac{1}{2}}-y_{i+\frac{1}{2},j+\frac{1}{2}}\right)+\\
\left(\frac{1}{2Vol_{i+\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(\boldsymbol{T_{i,j}}+T_{i+1,j}\right)\left(y_{i+1,j}-y_{i,j}\right)+\left(T_{i+1,j}+T_{i+1,j+1}\right)\left(y_{i+1,j+1}-y_{i+1,j}\right)+\\ \left(T_{i,j+1}+T_{i+1,j+1}\right)\left(y_{i,j+1}-y_{i+1,j+1}\right)+\left(\boldsymbol{T_{i,j}}+T_{i,j+1}\right)\left(y_{i,j}-y_{i,j+1}\right)\end{array}\right]\right)
\end{array}\right]$$

It is easier to do this as 2 steps, however!! I'll explain this more later.

Point-Jacobi Finite Volume Representation

X- second derivative term only:

$$T_{i,j}^{n+1}\left[\begin{array}{l}\frac{\left(y_{i+1,j}-y_{i,j+1}\right)}{Vol_{i+\frac{1}{2},j+\frac{1}{2}}}+\\ \frac{\left(y_{i,j+1}-y_{i-1,j}\right)}{Vol_{i-\frac{1}{2},j+\frac{1}{2}}}+\\ \frac{\left(y_{i-1,j}-y_{i,j-1}\right)}{Vol_{i-\frac{1}{2},j-\frac{1}{2}}}+\\ \frac{\left(y_{i,j-1}-y_{i+1,j}\right)}{Vol_{i+\frac{1}{2},j-\frac{1}{2}}}\end{array}\right] = \frac{1}{2Vol_{i,j}}\left[\begin{array}{l}
\left(\frac{1}{2Vol_{i+\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i+1,j}\right)\left(y_{i+1,j}-y_{i,j}\right)+\left(T_{i+1,j}+T_{i+1,j+1}\right)\left(y_{i+1,j+1}-y_{i+1,j}\right)+\\ \left(T_{i,j+1}+T_{i+1,j+1}\right)\left(y_{i,j+1}-y_{i+1,j+1}\right)+\left(T_{i,j+1}\right)\left(y_{i,j}-y_{i,j+1}\right)\end{array}\right]+\right.\\
\left.\left(\frac{1}{2Vol_{i-\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i-1,j}\right)\left(y_{i,j}-y_{i-1,j}\right)+\left(T_{i,j+1}\right)\left(y_{i,j+1}-y_{i,j}\right)+\\ \left(T_{i-1,j+1}+T_{i,j+1}\right)\left(y_{i-1,j+1}-y_{i,j+1}\right)+\left(T_{i-1,j}+T_{i-1,j+1}\right)\left(y_{i-1,j}-y_{i-1,j+1}\right)\end{array}\right]\right)\left(y_{i+\frac{1}{2},j+\frac{1}{2}}-y_{i-\frac{1}{2},j+\frac{1}{2}}\right)+\\
\left(\frac{1}{2Vol_{i-\frac{1}{2},j+\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i-1,j}\right)\left(y_{i,j}-y_{i-1,j}\right)+\left(T_{i,j+1}\right)\left(y_{i,j+1}-y_{i,j}\right)+\\ \left(T_{i-1,j+1}+T_{i,j+1}\right)\left(y_{i-1,j+1}-y_{i,j+1}\right)+\left(T_{i-1,j}+T_{i-1,j+1}\right)\left(y_{i-1,j}-y_{i-1,j+1}\right)\end{array}\right]+\right.\\
\left.\left(\frac{1}{2Vol_{i-\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i-1,j-1}+T_{i,j-1}\right)\left(y_{i,j-1}-y_{i-1,j-1}\right)+\left(T_{i,j-1}\right)\left(y_{i,j}-y_{i,j-1}\right)+\\ \left(T_{i-1,j}\right)\left(y_{i-1,j}-y_{i,j}\right)+\left(T_{i-1,j-1}+T_{i-1,j}\right)\left(y_{i-1,j-1}-y_{i-1,j}\right)\end{array}\right]\right)\left(y_{i-\frac{1}{2},j+\frac{1}{2}}-y_{i-\frac{1}{2},j-\frac{1}{2}}\right)+\\
\left(\frac{1}{2Vol_{i-\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i,j-1}+T_{i+1,j-1}\right)\left(y_{i+1,j-1}-y_{i,j-1}\right)+\left(T_{i+1,j-1}+T_{i+1,j}\right)\left(y_{i+1,j}-y_{i+1,j-1}\right)+\\ \left(T_{i+1,j}\right)\left(y_{i,j}-y_{i+1,j}\right)+\left(T_{i,j-1}\right)\left(y_{i,j-1}-y_{i,j}\right)\end{array}\right]+\right.\\
\left.\left(\frac{1}{2Vol_{i+\frac{1}{2},j-\frac{1}{2}}}\left[\begin{array}{l}\left(T_{i,j-1}+T_{i+1,j-1}\right)\left(y_{i+1,j-1}-y_{i,j-1}\right)+\left(T_{i+1,j-1}+T_{i+1,j}\right)\left(y_{i+1,j}-y_{i+1,j-1}\right)+\\ \left(T_{i+1,j}\right)\left(y_{i,j}-y_{i+1,j}\right)+\left(T_{i,j-1}\right)\left(y_{i,j-1}-y_{i,j}\right)\end{array}\right]\right)\left(y_{i+\frac{1}{2},j-\frac{1}{2}}-y_{i+\frac{1}{2},j+\frac{1}{2}}\right)
\end{array}\right]$$

Finite Volume Representation

- **Now, if we write the value of $T_{i,j}$ to be at iteration level n+1 and all other values of T to be at iteration n, and we add a similar term for the y-second derivative, we have developed a general form of the point-Jacobi numerical method (in finite-volume form)**
- **For the previous second derivative of T wrt x, we get:**

Finite Volume Representation

- **You could derive a similar term for the y-second derivative and add this to the previous term for the x-second derivative to get a general form of the point-Jacobi iteration.**
- **The resulting discretization would work for <u>any arbitrary orientation</u> of the block**

A Similar Poisson Problem

- **Now, instead of solving the "steady" heat conduction equation, let's solve for the temperature field as a transient problem**

$$\rho c_p \frac{\partial T}{\partial t} = k\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right]$$

$$\frac{\partial T}{\partial t} = \alpha\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right]$$

- **This changes the "characteristics" of the equation and forces us to use different numerical algorithms**

Finite Difference Representation

- **If we write the transient heat conduction equation in terms of finite-differences with the RHS terms to all exist at time level n (making the scheme *EXPLICIT*)**

$$\frac{\partial T}{\partial t} = \alpha\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right]$$

$$\frac{T_{i,j}^{n+1} - T_{i,j}^{n-1}}{2\Delta t} = \alpha\left[\frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2} + \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(y_{i+\frac{1}{2},j} - y_{i-\frac{1}{2},j})^2}\right]$$

(Richardson's scheme) (2nd order accurate in space and time) (*uncondtionally UNSTABLE*)

$$\frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t} = \alpha\left[\frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2} + \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(y_{i+\frac{1}{2},j} - y_{i-\frac{1}{2},j})^2}\right]$$

(Simple Explicit) (1st order accurate in time, 2nd order accurate in space) (*conditionally stable*)

Finite Difference Representation

- **And a stability analysis states that**

$$\alpha\,\Delta t\left[\frac{1}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2} + \frac{1}{(y_{i+\frac{1}{2},j} - y_{i-\frac{1}{2},j})^2}\right] \le \frac{1}{2} \quad or$$

$$\Delta t \le \frac{1}{2\alpha}\left[\frac{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2\,(y_{i+\frac{1}{2},j} - y_{i-\frac{1}{2},j})^2}{(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j})^2 + (y_{i+\frac{1}{2},j} - y_{i-\frac{1}{2},j})^2}\right] \quad or\ \Delta t \le \frac{CFL}{2\alpha}\left[\frac{Vol^2}{\Delta x^2 + \Delta y^2}\right]$$

See "Computational Fluid Mechanics and Heat Transfer" by D.A. Anderson, J. C. Tannehill, and R. H. Pletcher p 115

- ***If* we created a computational grid that has constant spacing in each direction independently, these equations would reduce to:**

$$T_{i,j}^{n+1} = T_{i,j}^n + \Delta t\,\alpha\left[\frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(\Delta x)^2} + \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(\Delta y)^2}\right] \qquad \Delta t \le \frac{CFL}{2\alpha}\left[\frac{Vol^2}{(\Delta x)^2 + (\Delta y)^2}\right]$$

where $CFL \approx 0.5$

- **Thus the temperature at a given node at the next time level is completely dependent on values of the temperature field at the current time level (explicit scheme)**
- **Note that Jacobians or a control volume approach would make the RHS completely general**

Finite Difference Representation

- **You can see that this algorithm would be relatively simple to program. You can also see that the convergence rate is dependent on the stability limit for Δt.**

- **If we changed the value of T on the RHS of the heat conduction equation to be at the next time level, our discretization would change to (Laasonen's Method):**

Fully Implicit

$$T_{i,j}^{n+1} = T_{i,j}^n + \Delta t\,\alpha\left[\frac{T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i-1,j}^{n+1}}{(\Delta y)^2}\right]$$

$$T_{i,j}^{n+1}\left[1 + \frac{2\Delta t\,\alpha}{(\Delta x)^2} + \frac{2\Delta t\,\alpha}{(\Delta y)^2}\right] - \Delta t\,\alpha\left[\frac{T_{i+1,j}^{n+1} + T_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{T_{i+1,j}^{n+1} + T_{i-1,j}^{n+1}}{(\Delta y)^2}\right] = T_{i,j}^n$$

- **Which is unconditionally stable (we can use any value of Δt!) but involves inverting a 5-diagonal matrix.**

Finite Difference Representation

- **Rather than invert a single large matrix for the solution, we can break up the solution into 2 steps and solve:**

$$T_{i,j}^{n+1/2} - T_{i,j}^{n} = \frac{\Delta t}{2}\alpha\left[\frac{T_{i+1,j}^{n+1/2} - 2T_{i,j}^{n+1/2} + T_{i-1,j}^{n+1/2}}{(\Delta x)^2} + \frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{(\Delta y)^2}\right]$$

$$T_{i,j}^{n+1} - T_{i,j}^{n+1/2} = \frac{\Delta t}{2}\alpha\left[\frac{T_{i+1,j}^{n+1/2} - 2T_{i,j}^{n+1/2} + T_{i-1,j}^{n+1/2}}{(\Delta x)^2} + \frac{T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i-1,j}^{n+1}}{(\Delta y)^2}\right]$$

- **The solution for each step involves inverting a tri-diagonal matrix. The algorithm is unconditionally stable so we can use any value for Δt. This scheme is known as the *ADI (Peaceman and Rachford, and Douglas) or SLR* method**

Time-Marching Control-Volume Method

- **Another scheme involves using the control-volume approach for the RHS (spatial) discretization along with a finite-difference representation for the LHS (temporal)**

$$\int_{V_{ol,s}} \frac{\partial T}{\partial t} dV_{ol,s} = \alpha \int_{V_{ol,s}} \left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right] dV_{ol,s} = \alpha \int_{V_{ol,s}} \left[\frac{\partial}{\partial x}\frac{\partial T}{\partial x} + \frac{\partial}{\partial y}\frac{\partial T}{\partial y}\right] dV_{ol,s}$$

$$\frac{\left(T^{n+1} - T^{n}\right)V_{ol,s}}{\Delta t} = \alpha\left[\oint_{V_{ol,s}} \frac{\partial T}{\partial x} dy - \oint_{V_{ol,s}} \frac{\partial T}{\partial y} dx\right]$$

$$\frac{T^{n+1} - T^{n}}{\Delta t} = \frac{\alpha}{V_{ol,s}}\left[\oint_{V_{ol,s}} \left(\frac{1}{V_{ol,p}} \oint_{V_{ol,p}} T dy\right) dy + \oint_{V_{ol,s}} \left(\frac{1}{V_{ol,p}} \oint_{V_{ol,p}} T dx\right) dx\right]$$

Remember

$$\int_{V_{ol}} \frac{\partial \phi}{\partial x} dV_{ol} = \oint_{V_{ol}} \phi\, dy$$

$$\int_{V_{ol}} \frac{\partial \phi}{\partial y} dV_{ol} = -\oint_{V_{ol}} \phi\, dx$$

- **This approach has become very popular in CFD and simulation codes in general and is the approach that I would like for you to use in your projects**

Project-1

- **Write a computer program to solve the heat conduction equation on a single block of steel**
  - k = 18.8 W/m K, ρ = 8000 kg/m^3, c_p = 500 J/(kg K)
  - Solve the transient heat conduction equation using the general, explicit control volume scheme on a non-uniform computational grid.
- **Use a computational grid:**
  - That is IMAX x JMAX in dimensions
    - Where IMAX=JMAX=101 AND IMAX=JMAX=501 (2 cases)
  - That has the following non-uniform distribution:

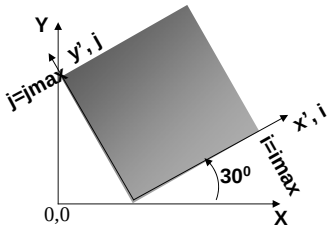
rot= 30.*3.141592654/180.

xp = cos[0.5*π(imax-i)/(imax-1)]

yp = cos[0.5*π(jmax-j)/(jmax-1)]

x(i,j) = xp*cos(rot)+(1.-yp)*sin(rot)

y(i,j) = yp*cos(rot)+xp*sin(rot)



Project-1

- **Find the temperature distribution on a 1m x 1m block with Dirichlet boundary conditions:**
  - T=5.[sin(πxp)+1.] at j=jmax
  - T=abs(cos(πxp))+1. at j=0
  - T=3.yp+2. at i=0 and i=imax

Note that xp and yp are used in these equations!

- **Iterate until the maximum residual magnitude drops below TOLER=1.0 x 10^{-5}**
  - Hint: The number of iterations will depend on the equation and the algorithm (~10,000 for 101 grid dimension)
- **Initialize the temperature field to a uniform value of 3.5**
- **Write out your converged solution to a (PLOT3D or equivalent) file that can be used for plotting or restart capability**
  - an example plot3d routine is on smartsite. Additional information can be found on the internet.

Project-1

- **Run your code on a cluster node that is not being used via qsub**
  - Compiling of code can only be performed on wopr
- **Use qsub to submit your job (that will automatically go to one of the nodes compute-0-0 through compute-0-6)**

Project-1

- **Due Wednesday October 16th (PAY CLOSE ATTENTION TO WHAT IS ASKED FOR!):**
  - Statement of problem, equations, and algorithm(s) used
  - Listing of Fortran or C code (You may program this and all projects in C if you prefer)
  - Output of run. Print out the iteration number, the magnitude of the maximum residual, and the indicies where the maximum residual was located
  - The CPU time from start of iteration to convergence

```
      integer clock_start,clock_end,clock_max,clock_rate
      real*4 wall_time
!     call system time to determine flow solver wall time
      call system_clock(count_max=clock_max,count_rate=clock_rate)
      call system_clock(clock_start)
!     determine total wall time for solver
      call system_clock(clock_end)
      wall_time=float(clock_end-clock_start)/float(clock_rate)
      print*,'solver wall clock time (seconds)',wall_time
```

  - Make sure that you link to the system library for the clock routines by adding –lrt during the linking step in your makefile
  - Plot of computational grid
  - Plot of converged temperature fields