

COMPUTATIONAL THERMAL ANALYSIS

PRESENTATION MEDIA SAMPLE

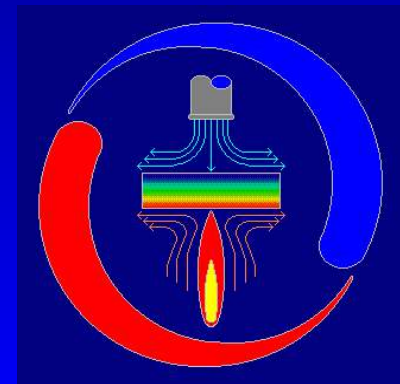
Selection Featured in Lecture 5b
© Dean S. Schrage

Engineering Short Course

www.TITANAlgorithms.com/courses.html

Instructor: Dean S. Schrage

Admin@TITANAlgorithms.com



Semi-Direct Solution Methods

ADI and CGC

ADI

- **ADI:** *Alternating Direction Implicit solution methods describe an approximate **splitting** of the heat flow into the X, Y, Z directions*
 - describe the energy flow in each direction with separate equations
 - these equations are **tri-diagonal**
 - solve the tri-diagonal matrix equations using exact **hybrid inversion**
 - **couple** the advance of equations
- Several ADI schemes have been developed over the years:
 - splitting
 - f-factor
 - Brian and Douglas
- **Splitting Method:** *an ideal method to become familiar with the concepts highlighted above*
 - algorithm is easiest to implement with fewest numerical operations
 - however, limited to systems with narrow **conductance band**

ADI SOLUTION METHOD

Generic Method Setup

- Discretize the discretized energy transport equation using implicit time differencing¹:

$$\rho C_p \delta V_i \left(\frac{T_i^{(n)} - T_i^{(n-1)}}{\Delta t} \right) = \delta \hat{Q}(T^{(n)}, \mathbf{G}_{j,j}, i, \mathbf{e}_X, \mathbf{e}_Y, \mathbf{e}_Z) + \dot{S}_i$$

- Note that integrated heat flux operator δQ works on X, Y, Z directions
 - convert the equations to the standard matrix equation


$$\mathbf{A} \mathbf{T}^{(n)} = \mathbf{b}$$

- \mathbf{A} is banded and sparse
- rewrite as linear system with source vector \mathbf{b}
 - holds capacitance and source heating effects
 - off-diagonal heat flows for higher order methods (Brian)

¹ The ADI method is not limited to implicit differencing and can be applied to a Crank-Nicholson differencing with an adjustment in the source vector \mathbf{b} .

ADI SOLUTION METHOD SETUP

Splitting the Matrix

$$\mathbf{A} \mathbf{T}^{(n)} = \mathbf{b}$$

- Split \mathbf{A} matrix and source vector \mathbf{b} according to sweep directions X, Y, Z :

$$\mathbf{A}_X \mathbf{U} = \hat{\mathbf{b}}_X(\mathbf{T}^{(n-1)})$$

$$\mathbf{A}_Y \mathbf{V} = \hat{\mathbf{b}}_Y(\mathbf{T}^{(n-1)}, \mathbf{U})$$

$$\mathbf{A}_Z \mathbf{T}^{(n)} = \hat{\mathbf{b}}_Z(\mathbf{T}^{(n-1)}, \mathbf{U}, \mathbf{V})$$

- each split matrix $\mathbf{A}_X, \mathbf{A}_Y, \mathbf{A}_Z$ is tri-diagonal
- advance $\varphi^{(n-1)}, \mathbf{U}, \mathbf{V}$ towards solution of $\varphi^{(n)}$
- note the source vector \mathbf{b} can also be a function of the ADI **sub-states**
 - \mathbf{U}, \mathbf{V} intermediate temperature vectors en route to $\mathbf{T}^{(n)}$
 - Brian method

ADI SOLUTION METHOD SETUP

Solving the Matrix Equation

- Solve each *tri-diagonal* system of equations using the **Thomas algorithm**
 - the **Thomas algorithm** is a recursive routine to arrive at a hybrid numerical inverse to each matrix A_n
 - **LU decomposition** of each A_n
 - solve the **lower** then the **upper** system recursively
 - very efficient in terms of speed and storage
 - refer to most numerical analysis texts for algorithm design

$$A_X U = \hat{b}_X(T^{(n-1)})$$

$$A_Y V = \hat{b}_Y(T^{(n-1)}, U) \quad \longrightarrow \quad \text{solve } U \rightarrow \text{solve } V \rightarrow \text{solve } T^{(n)}$$

$$A_Z T^{(n)} = \hat{b}_Z(T^{(n-1)}, U, V)$$

Code for Thomas Algorithm
two subroutines
minimal line length

```
TK2
8 x 12
(19t) Statement: 210 /F9
===== PROCEDURE FUNCTION: LDU_Build =====
S Statement
; LU decomposition solution to tri-diagonal system A[x] = f
; terms are:      N      number of rows
;                b,a,c  diagonals on A
;                f      source vector on RHS

call blank(L)
call blank(D)
call blank(U)

D[1] = a[1]
U[1] = c[1]
for i = 2 to N - 1
  L[i] = b[i] / D[i - 1]
  D[i] = a[i] - L[i] * U[i - 1]
  U[i] = c[i]
next i
L[N] = b[N] / D[N - 1]
D[N] = a[N] - L[N] * U[N - 1]
F1 Help F2 Cancel F5 Edit F9
```

```
TK2
8 x 12
(20t) Statement: 210 /F9
===== PROCEDURE FUNCTION: LDU_Solve =====
S Statement
; LU decomposition solution to tri-diagonal system A[x] = f
; terms are:      N      number of rows
;                L,D,U  LDU decomposition of the tridiagonal A
;                f      source vector on RHS
;                y,x    intermediate and final solution vectors

call blank(x)
call blank('y_LDU)
; Forward solver for intermediate y state:
'y_LDU[1] = f[1]
for i = 2 to N
  'y_LDU[i] = f[i] - L[i] * 'y_LDU[i - 1]
next i

; Backward solver for x state:
x[N] = 'y_LDU[N] / D[N]
for i = N - 1 to 1 step -1
  x[i] = ('y_LDU[i] - U[i] * x[i + 1]) / D[i]
next i
F1 Help F2 Cancel F5 Edit F9 Solve / Commands = Sheets ; Window switch
```

ADI SOLUTION METHOD

Splitting Method Setup

- Derive 3 orthogonal sweep equations by splitting the heat flow and associated capacitance term in each direction
 - sum of equations with the $\Delta t/3$ time step give original discretized energy equation
 - factor 3 on the flux term
 - necessary to produce proper scaling with presence of source
 - often erroneously neglected

$$(X\text{-Sweep}) \quad (\rho \delta V C_p)_i \frac{U_i - T_i^{(n-1)}}{\frac{\Delta t}{3}} = 3 \delta Q(\mathbf{U}, \mathbf{G}_{ji}, i, \mathbf{e}_x) + \dot{S}_i$$

$$(Y\text{-Sweep}) \quad (\rho \delta V C_p)_i \frac{V_i - U_i}{\frac{\Delta t}{3}} = 3 \delta Q(\mathbf{V}, \mathbf{G}_{ji}, i, \mathbf{e}_y) + \dot{S}_i$$

$$(Z\text{-Sweep}) \quad (\rho \delta V C_p)_i \frac{T_i^{(n)} - V_i}{\frac{\Delta t}{3}} = 3 \delta Q(\mathbf{T}^{(n)}, \mathbf{G}_{ji}, i, \mathbf{e}_z) + \dot{S}_i$$

ADI-SPLITTING METHOD

General Sweep Equation

- Derive general sweep equation which can be solved directly using the **Thomas algorithm**

$$p_{i_n} \Psi_{i-1} + q_{i_n} \Psi_i + r_{i_n} \Psi_{i+1} = b_{i_n} \quad \text{for } n = X, Y, Z$$

- tri-diagonal equation with coefficient vectors **p, q, r, b**
 - each coefficient (**p, q, r, b**) is unique to the CV- i number and the sweep direction n
- ψ is a pointer to the sweep-specific ADI state variable
 - $\psi = [T^{(n-1)}, U, V, T^{(n)}]^T$ where **U, V** are intermediate temperatures
 - pointer convention used to create extendable structured code
 - example: *make a pass on sweep $n = 2 = Y$*
 - *point to sweep variable $\psi[2+1] = V$*
 - that is, solve for vector **V** which is the temperature at a fractional time level (e.g. $n - 1/3$ for splitting)

ADI-SPLITTING METHOD

Coefficient Extraction

sweep direction	p_i	q_i	r_i	b_i
x	$-3 G_{i-1,x,i}$	$\frac{(\rho \delta V C_p)_i}{\Delta t/3} + 3 (G_{i-1,x,i} + G_{i+1,x,i})$	$-3 G_{i+1,x,i}$	$\dot{S}_i + \frac{(\rho \delta V C_p)_i}{\Delta t/3} T_i^{(n-1)}$
y	$-3 G_{i-1,y,i}$	$\frac{(\rho \delta V C_p)_i}{\Delta t/3} + 3 (G_{i-1,y,i} + G_{i+1,y,i})$	$-3 G_{i+1,y,i}$	$\dot{S}_i + \frac{(\rho \delta V C_p)_i}{\Delta t/3} U_i^{(n-\frac{2}{3})}$
z	$-3 G_{i-1,z,i}$	$\frac{(\rho \delta V C_p)_i}{\Delta t/3} + 3 (G_{i-1,z,i} + G_{i+1,z,i})$	$-3 G_{i+1,z,i}$	$\dot{S}_i + \frac{(\rho \delta V C_p)_i}{\Delta t/3} V_i^{(n-\frac{1}{3})}$

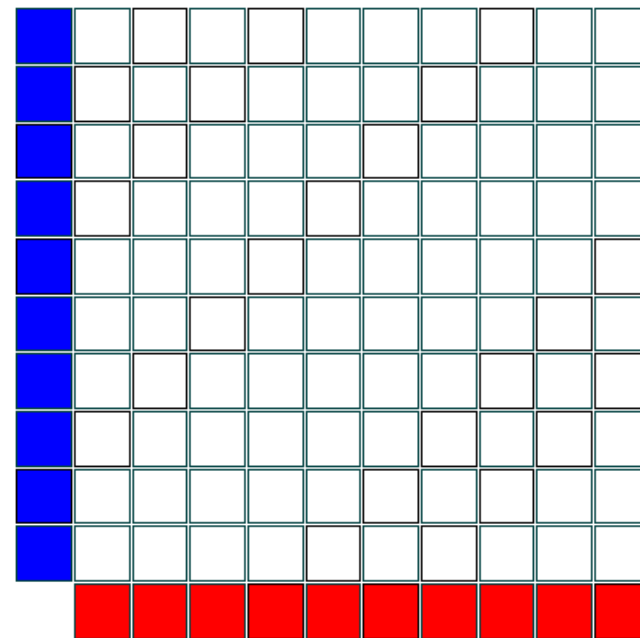
ADI-SPLITTING METHOD

Application Study - Plate With 2 Boundary Conditions

- Plate geometry with 10 x 10 discrete CV
 - 2 fixed temperature boundary conditions
 - 100 and 0 °C

- Apply the splitting method to solve the transient temperature
 - solve to steady state
 - compare to SOR
 - evaluate work-units (normalized wall clock time)

➔ *ADI will propagate inward each of the boundary temperatures along the direction normal to the surface*



Simple 2D plate discretization, 1 cm x 1 cm CV size, Aluminum, $k = 225 \text{ W/m K}$

ANIMATION DISPLAY
ADI Splitting Solution 5 second Time Step

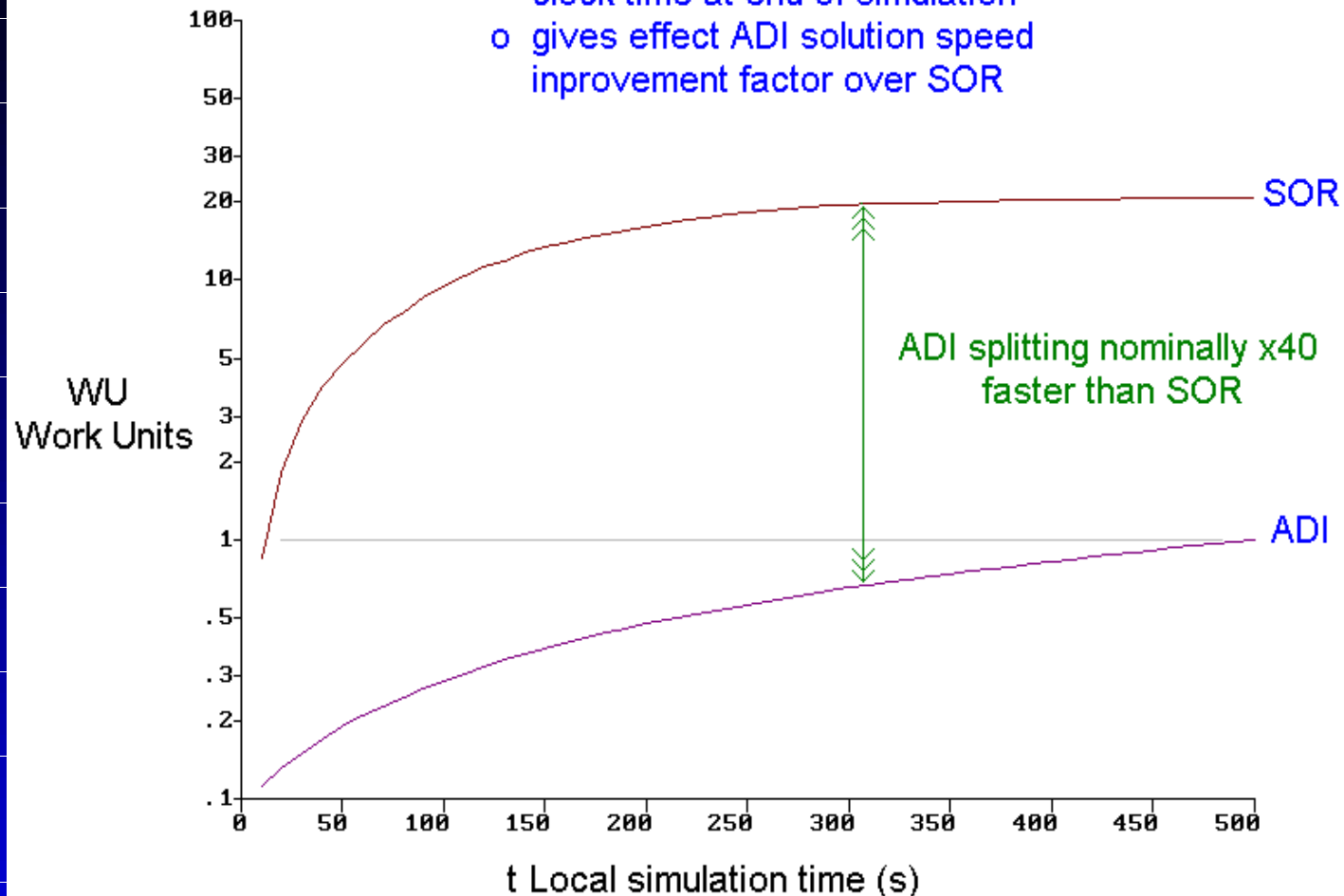
ADI-SPLITTING METHOD

Application Study - Plate With 2 Boundary Conditions

- **Solution Efficiency:** *ADI splitting proceeds as a very rapid rate compared to SOR ($\omega = 1$)*
 - SOR requires almost 100 iterations per each time step
 - ADI nominally x40 faster during transient solution
- SOR achieves a speedup as steady state is reached:
 - note that ADI cannot sense this pending steady condition
 - basic WU (work unit) is fixed by the code design and discretization
 - however ADI will generally still outpace SOR

WU:

- o Normalized to 1 for ADI wall clock time at end of simulation
- o gives effect ADI solution speed improvement factor over SOR



Comparison of work units to achieve each second of simulation time, SOR and ADI-Splitting

ADI-SPLITTING METHOD

Banding Effect

- ADI-splitting neglects any contribution in the orthongonal directions on a sweep pass:
 - X-sweep equation neglects heat flow terms $\delta Q(U, \mathbf{G}_{jj}, i, \mathbf{e}_y, \mathbf{e}_z)$
 - the effects of this can be observed by displaying intermediate \mathbf{U} , \mathbf{V} states in animation
- Each sweep propagates the respective boundary information inward
 - basically each sweep is an exact solution to a 1D problem
 - both solutions are an exact solution but in their respective sweep direction
 - these exact solutions conflict with one another
 - leads to a ***banding effect***

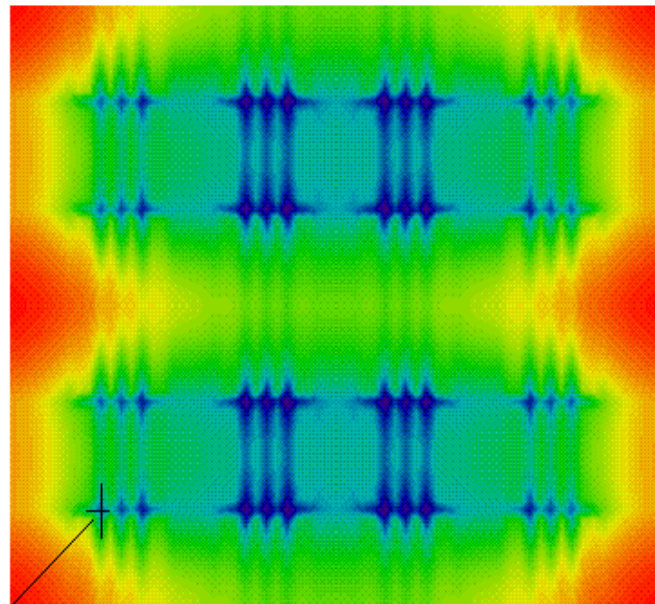
● **Banding Effect:** *Caused by lack of orthogonal heat flow terms in each sweep equation*

- exacerbated by large time steps
- exacerbated by large variations in conductance

➔ *While banding can be mitigated by using smaller time steps, generally the reduction in Δt is so extreme as to make ADI-Splitting no more efficient than SOR*

➔ *The solution to banding is hybrid ADI methods such as ADI-Brian*

Analysis of a copper backed circuit board with coupling through discrete pins to the cold boundary condition



ADI Splitting
Note the banding effect caused by splitting the equations in each direction

Banding effect introduced in ADI-Splitting solution where cold boundary condition couples through a low conductance interface to a copper surface plate

ANIMATION DISPLAY

ADI Splitting Solution 5 second Time Step Showing Sub-States

Note the banding effect is reduced as indicated by the smaller oscillation between the ADI sub states U and V. But by using a smaller time step, the ADI solution efficiency has been decreased.

ANIMATION DISPLAY

ADI Splitting Solution 2 second Time Step Showing Sub-States

Brian Method Setup - Mitigates Banding

- Derive 3 orthogonal sweep equations:
 - solution advances at time steps of $\frac{1}{2}\Delta t$ time step
 - note the orthogonal heat flow components
 - note the larger number of terms
 - will require more calculations

$$(X\text{-Sweep}) \quad (\rho \delta V C_p)_i \frac{U_i - T_i^{(n-1)}}{\frac{\Delta t}{2}} = \delta Q(\mathbf{U}, \mathbf{G}_{ji}, i, \mathbf{e}_x) + \delta Q(\mathbf{T}^{(n-1)}, \mathbf{G}_{ji}, i, \mathbf{e}_y) + \delta Q(\mathbf{T}^{(n-1)}, \mathbf{G}_{ji}, i, \mathbf{e}_z) + \dot{S}_i$$

$$(Y\text{-Sweep}) \quad (\rho \delta V C_p)_i \frac{V_i - U_i}{\frac{\Delta t}{2}} = \delta Q(\mathbf{U}, \mathbf{G}_{ji}, i, \mathbf{e}_x) + \delta Q(\mathbf{V}, \mathbf{G}_{ji}, i, \mathbf{e}_y) + \delta Q(\mathbf{T}^{(n-1)}, \mathbf{G}_{ji}, i, \mathbf{e}_z) + \dot{S}_i$$

$$(Z\text{-Sweep}) \quad (\rho \delta V C_p)_i \frac{T_i^{(n)} - V_i}{\frac{\Delta t}{2}} = \delta Q(\mathbf{U}, \mathbf{G}_{ji}, i, \mathbf{e}_x) + \delta Q(\mathbf{V}, \mathbf{G}_{ji}, i, \mathbf{e}_y) + \delta Q(\mathbf{T}^{(n)}, \mathbf{G}_{ji}, i, \mathbf{e}_z) + \dot{S}_i$$

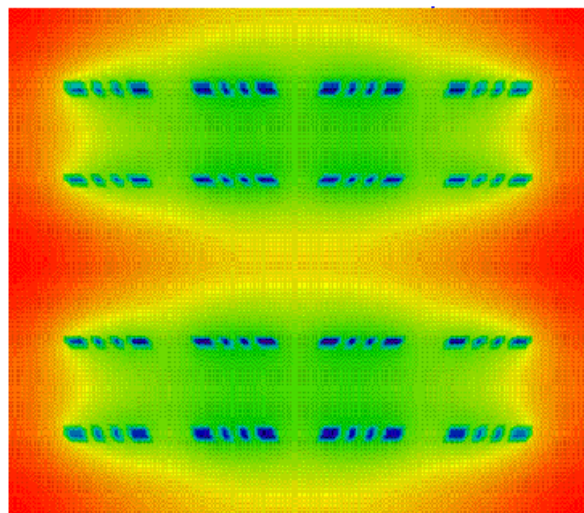
ADI-BRIAN METHOD

Coefficient Extraction

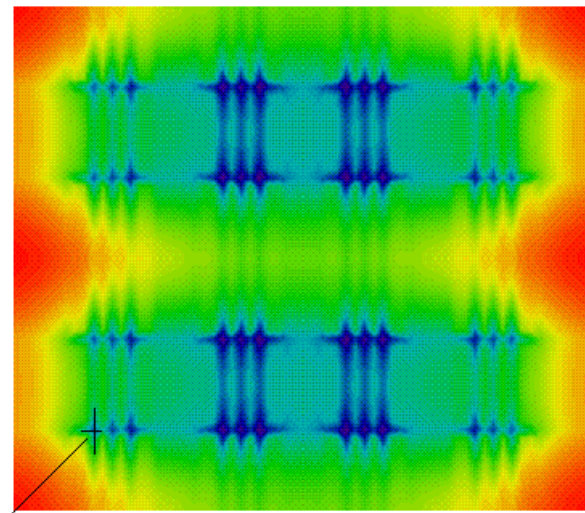
sweep direction	p_i	q_i	r_i	b_i
x	$-G_{i-1,x,i}$	$\frac{(\rho \delta V C_p)_i}{\Delta t/2} + (G_{i-1,x,i} + G_{i+1,x,i})$	$-G_{i+1,x,i}$	$\dot{S}_i + \frac{(\rho \delta V C_p)_i}{\Delta t/2} T_i^{(n-1)} + \delta Q(\mathbf{T}^{(n-1)}, \mathbf{G}_{j\bar{j}}, i, e_y) + \delta Q(\mathbf{T}^{(n-1)}, \mathbf{G}_{j\bar{j}}, i, e_z)$
y	$-G_{i-1,y,i}$	$\frac{(\rho \delta V C_p)_i}{\Delta t/2} + (G_{i-1,y,i} + G_{i+1,y,i})$	$-G_{i+1,y,i}$	$\dot{S}_i + \frac{(\rho \delta V C_p)_i}{\Delta t/2} U_i^{(n-\frac{1}{2})} + \delta Q(\mathbf{U}, \mathbf{G}_{j\bar{j}}, i, e_x) + \delta Q(\mathbf{T}^{(n-1)}, \mathbf{G}_{j\bar{j}}, i, e_z)$
z	$-G_{i-1,z,i}$	$\frac{(\rho \delta V C_p)_i}{\Delta t/2} + (G_{i-1,z,i} + G_{i+1,z,i})$	$-G_{i+1,z,i}$	$\dot{S}_i + \frac{(\rho \delta V C_p)_i}{\Delta t/2} V_i^{(n-\frac{1}{2})} + \delta Q(\mathbf{U}, \mathbf{G}_{j\bar{j}}, i, e_x) + \delta Q(\mathbf{V}, \mathbf{G}_{j\bar{j}}, i, e_y)$

- In the ADI-Brian implementation, the tri-diagonal vectors are evaluated much the same way as ADI-Splitting
 - however, the source vector b_n is more costly to compute
 - vector b_n must be updated at each time step
 - this computational cost is offset by the ability to apply larger Δt

Analysis of a copper backed circuit board with coupling through discrete pins to the cold boundary condition



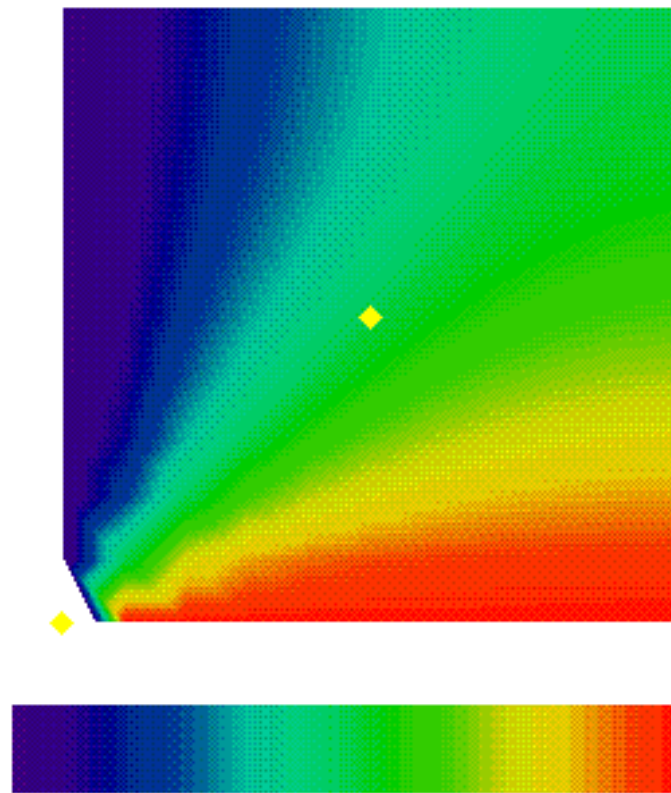
ADI Brian



ADI Splitting

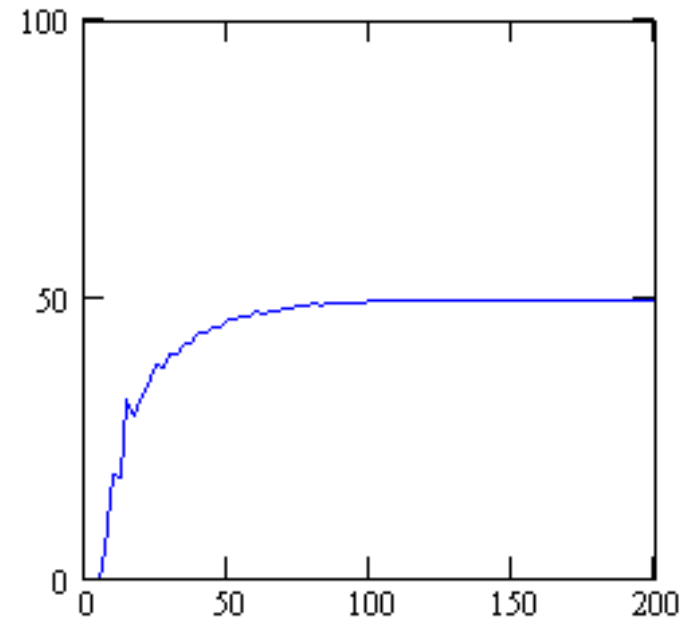
Note the banding effect caused by splitting the equations in each direction

Banding can only be suppressed by using small time step in the splitting scheme. Thus, while splitting is more computationally efficient, the banding effect requires more operations to simulate the process to steady state. For this reason, the Brian method (or other) is recommended for the ADI solution kernel.



$S_{\text{Min}} = 0$

$S_{\text{Max}} = 100$



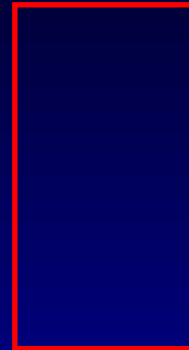
$T(t)$ temperature at indicated point

$$t = 405 \text{ s}^{-1} \cdot s$$

$$T_{i \text{ FRAME} - 1} = 49.996$$

Final result ADI-Brian showing elimination of banding effect

The ADI-Brian method reduces the difference between each of the sweeps. There is some initial oscillations at the start of the transient simulation as shown below.



ANIMATION DISPLAY

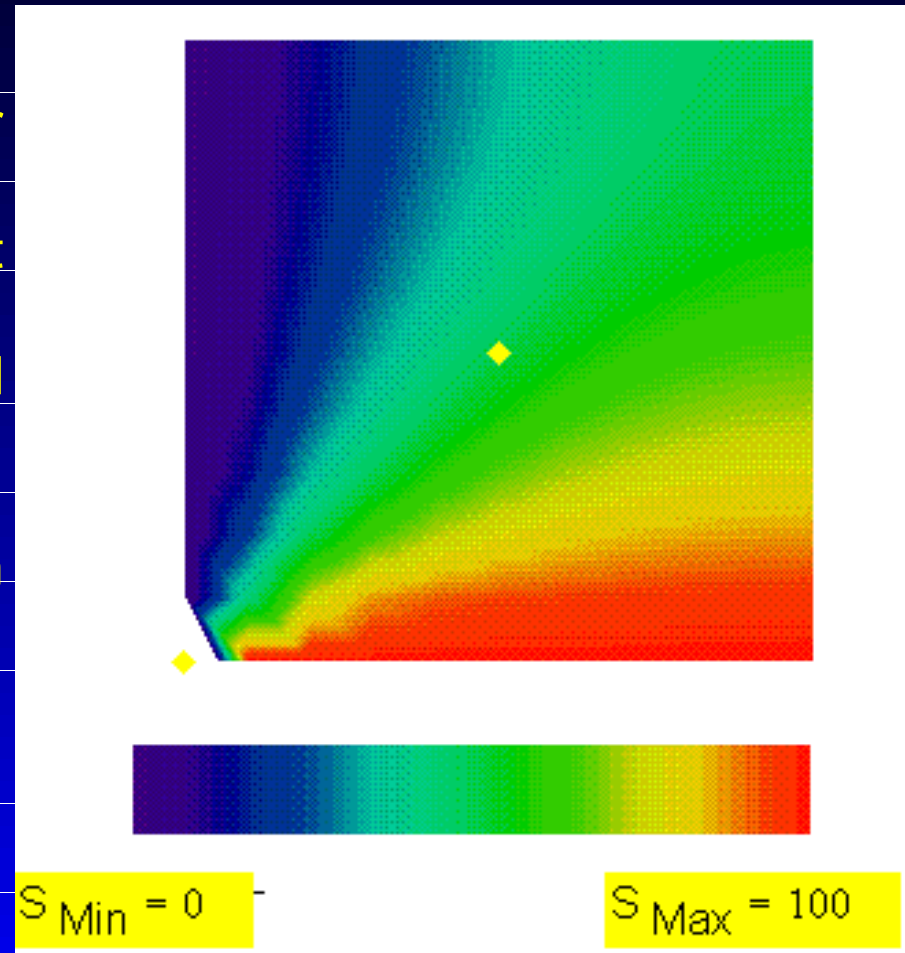
ADI Brian Solution 5 second Time Step Showing Sub-States

ADI-BRIAN METHOD

Applied as a Steady State Solver

- ADI methods are generally not associated with steady state solvers as they must progress through a transient
- However, with ADI-Brian, the solver can be pushed to a large time step
 - solve 2D plate problem with $\Delta t = 50$ seconds
 - progress through a distorted transient solution
- Achieve the steady state result with nominally 20 sweeps

➔ *With hybrid ADI methods, the steady state solution can be achieved in a quasi-exact inversion process*



ADI-Brian, $\Delta t = 50$ seconds, 20 sweep combinations

ANIMATION DISPLAY

ADI Brian Solution 50 second Time Step Showing Sub-States