Crank-Nicolson Scheme for the 1D Heat Equation
in a Nutshell gerry@pdx.edu

ME 448/548
Winter 2021

1. Combine finite difference approximations for $\partial u/\partial t$ at $x = x_i$

$$\left.\frac{\partial u}{\partial t}\right|_{t_k,x_i} = \frac{u_i^k - u_i^{k-1}}{\Delta t} + \mathcal{O}(\Delta t) \tag{1}$$

and the average of the $\partial^2 u/\partial x^2$ operators at time t_k and at t_{k+1}

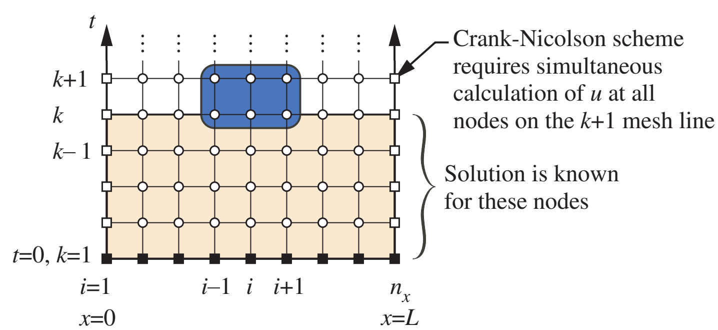
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{t_k,x_i} = \frac{\alpha}{2}\left[\frac{u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}}{\Delta x^2}\right] + \frac{\alpha}{2}\left[\frac{u_{i-1}^{k} - 2u_i^{k} + u_{i+1}^{k}}{\Delta x^2}\right] + \mathcal{O}(\Delta x^2). \tag{2}$$

to get a system of equations having the same structure as the BTCS method

$$-\frac{\alpha}{2\Delta x^2}u_{i-1}^{k+1} + \left(\frac{1}{\Delta t} + \frac{\alpha}{\Delta x^2}\right)u_i^{k+1} - \frac{\alpha}{2\Delta x^2}u_{i+1}^{k+1} =$$
$$\frac{\alpha}{2\Delta x^2}u_{i-1}^{k} + \left(\frac{1}{\Delta t} - \frac{\alpha}{\Delta x^2}\right)u_i^{k} + \frac{\alpha}{2\Delta x^2}u_{i+1}^{k} \tag{3}$$

Equation (3) is the computational formula for the Crank-Nicolson scheme. It is an *implicit* scheme because all u^{k+1} values are coupled and must be updated simultaneously.

2. Computational Molecule



3. Stability:

The Crank-Nicolson method is *unconditionally stable* for the heat equation.

The benefit of stability comes at a cost of increased complexity of solving a linear system of equations at each time step. The Crank-Nicolson scheme is not significantly more costly to implement than the BTCS Scheme

4. The Crank-Nicolson scheme has a truncation error that is $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$

5. For the one-dimensional heat equation, the linear system of equations for the Crank-Nicolson scheme can be organized into a tridiagonal matrix that looks just like the tridiagonal matrix for the BTCS scheme.

$$\begin{bmatrix} a_1 & b_1 & 0 & 0 & 0 & 0 \\ c_2 & a_2 & b_2 & 0 & 0 & 0 \\ 0 & c_3 & a_3 & b_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & c_{n_x-1} & a_{n_x-1} & b_{n_x-1} \\ 0 & 0 & 0 & 0 & c_{n_x} & a_{n_x} \end{bmatrix} \begin{bmatrix} u_1^{k+1} \\ u_2^{k+1} \\ u_3^{k+1} \\ \vdots \\ u_{n_x-1}^{k+1} \\ u_{n_x}^{k+1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n_x-1} \\ d_{n_x} \end{bmatrix} \tag{4}$$

The coefficients of the interior nodes ($i = 2, 3, \ldots, N-1$) are

$$\begin{aligned} a_i &= 1/\Delta t + \alpha/\Delta x^2 = 1/\Delta t - (b_i + c_i), \\ b_i &= c_i = -\alpha/(2\Delta x^2), \\ d_i &= -c_i u_{i-1}^k + (1/\Delta t + b_i + c_i)\, u_i^k - b_i u_{i+1}^k. \end{aligned}$$

As with the BTCS scheme, this system of equations is efficiently solved with a form of LU factorization. The LU factors need to be computed only once before the first time step.

6. MATLAB implementation: code from `demoCN`

```
% --- Coefficients of the tridiagonal system
b = (-alfa/2/dx^2)*ones(nx,1);   %  Super diagonal: coefficients of u(i+1)
c = b;                           %  Subdiagonal:    coefficients of u(i-1)
a = (1/dt)*ones(nx,1) - (b+c);   %  Main Diagonal:  coefficients of u(i)
at = (1/dt + b + c);             %  Coefficient of u_i^k on RHS
a(1) = 1;     b(1) = 0;          %  Fix coefficients of boundary nodes
a(end) = 1;   c(end) = 0;
[e,f] = tridiagLU(a,b,c);        %  Save LU factorization

% --- Assign IC and save BC values in ub.  IC creates u vector
x = linspace(0,L,nx)';   u = sin(pi*x/L);   ub = [0 0];

% --- Loop over time steps
for k=2:nt
  % --- Update RHS for all equations, including those on boundary
  d =  - [0;       c(2:end-1).*u(1:end-2); 0] ...
       + [ub(1);  at(2:end-1).*u(2:end-1); ub(2)] ...
       - [0;       b(2:end-1).*u(3:end);   0];
  u = tridiagLUsolve(e,f,b,d);                   %  Solve the system
end
```

A more general implementation is in `heatCN`.

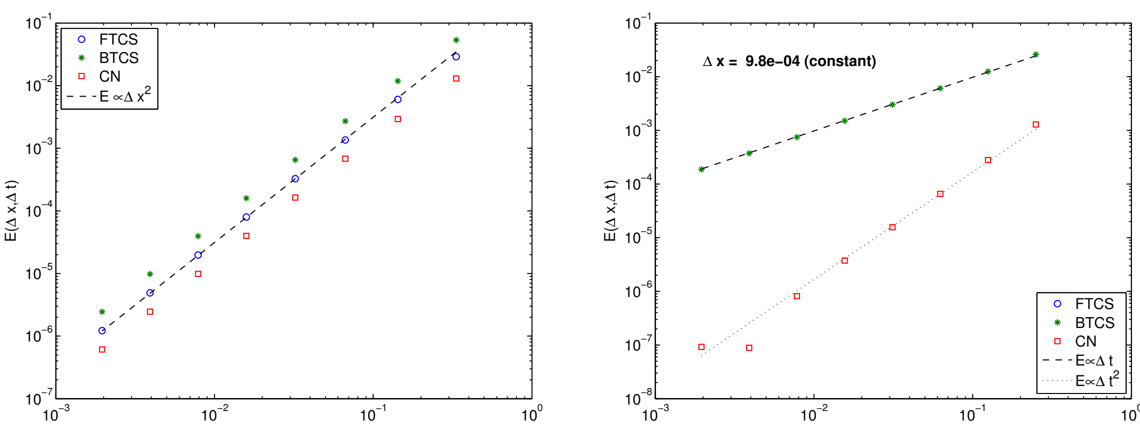
7. A comparison of FTCS, BTCS and Crank-Nicolson shows that all three have the same spatial truncation error. FTCS and BTCS have the same temporal truncation error. Crank-Nicolson has superior temporal truncation error.

	Truncation Errors	
Scheme	Spatial	Temporal
FTCS	Δx^2	Δt
BTCS	Δx^2	Δt
C-N	Δx^2	Δt^2

8. The `compHeatSchemes` function shows that our MATLAB implementation of all three schemes demonstrate the correct behavior of truncation error.

```
>> compHeatSchemes

Reduce both dx and dt within the FTCS stability limit

                     -------------  Errors  ------------
      nx       nt        FTCS        BTCS          CN
       4        5    2.903e-02   5.346e-02    1.304e-02
       8       21    6.028e-03   1.186e-02    2.929e-03
      16       92    1.356e-03   2.716e-03    6.804e-04
      32      386    3.262e-04   6.522e-04    1.630e-04
      64     1589    7.972e-05   1.594e-04    3.984e-05
     128     6453    1.970e-05   3.939e-05    9.847e-06
     256    26012    4.895e-06   9.790e-06    2.448e-06
     512   104452    1.220e-06   2.440e-06    6.101e-07

Reduce dt while holding dx = 9.775171e-04 (L=1.0, nx=1024) constant

                     -------------  Errors  ------------
      nx       nt        FTCS        BTCS          CN
    1024        8          NaN   2.601e-02    1.291e-03
    1024       16          NaN   1.246e-02    2.798e-04
    1024       32          NaN   6.102e-03    6.534e-05
    1024       64          NaN   3.020e-03    1.570e-05
    1024      128          NaN   1.502e-03    3.749e-06
    1024      256          NaN   7.492e-04    8.154e-07
    1024      512          NaN   3.742e-04    8.868e-08
    1024     1024          NaN   1.871e-04    9.218e-08
```

In the plot of truncation error versus Δt (right hand plot), there is an irregularity at $\Delta t \sim 3.9 \times 10^{-3}$. At that level of Δt, and for the chosen Δx, which is held constant, the truncation error due to Δx is no longer negligible. Further reductions in Δt alone will not reduce the total truncation error.



9. Truncation error is additive.

The truncation error for the finite-difference schemes that we have explored in this class so far are of the form

$$\bar{e} = \mathcal{O}(\Delta t^p) + \mathcal{O}(\Delta x^q) \tag{5}$$

where p and q are integers. For example, for the Crank-Nicolson scheme, $p = q = 2$.

The plot below demonstrates the effect of additive truncation errors. The horizontal axis is the time step size, Δt. The curves are for different spatial step sizes, Δx.

If Δt is reduced while Δx is held constant, the measured error is reduced until the point that the temporal truncation error is less than the spatial truncation error.

The results in the plot show that we need to be cognizant of all sources of truncation error. Usually, reducing Δt and Δx will help. However, there are situations where one source of truncation error dominates and that dominant source limits improvements in other factors you can control.

In a complex CFD simulation, the role of truncation errors may be hard to isolate. There are many modeling choices that can affect the result. That said, the fundamental effect of truncation error is always present.