FTCS Solution to the Heat Equation

ME 448/548 Notes

Gerald Recktenwald
Portland State University
Department of Mechanical Engineering
gerry@pdx.edu

Overview

1. Use the forward finite difference approximation to $\partial u/\partial t$.

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{k+1} - u_i^k}{\Delta t}$$

2. Use the central difference approximation to $\partial^2 u/\partial x^2$ at time t_k.

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} = \frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{\Delta x^2}$$

3. Solve for u_i^{k+1}. The computational formula is *explicit*: each value of u_i^{k+1} can be updated independently.
4. FTCS is easy to implement, but is only conditionally stable
5. Truncation errors are $\mathcal{O}\big((\Delta x)^2\big)$ and $\mathcal{O}(\Delta t)$.

Finite Difference Operators

Choose the *forward difference* to evaluate the time derivative at $t = t_k$.

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

Approximate the spatial derivative with the central difference operator and take all nodal values at time t_k.

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

FTCS Approximation to the Heat Equation

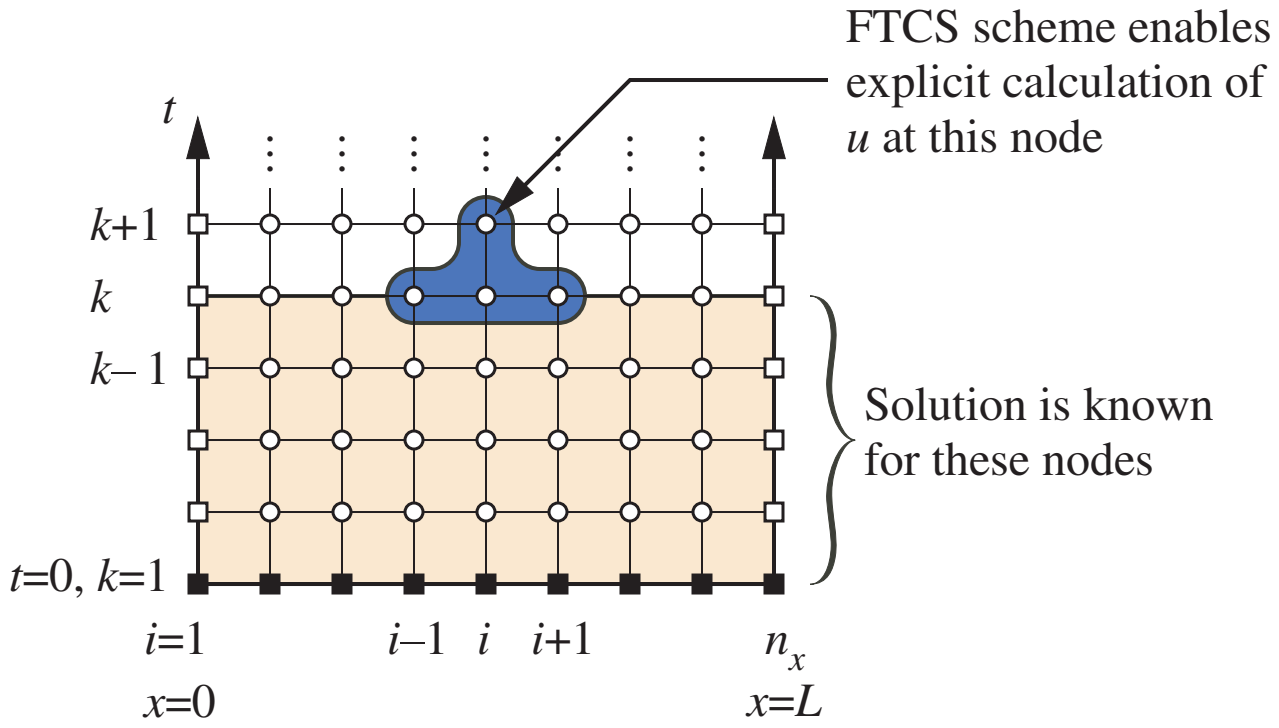
Substitute Equation (1) and Equation (2) into the heat equation

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

Drop truncation error terms to get

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \alpha \frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{\Delta x^2} \tag{4}$$

FTCS Computational Molecule



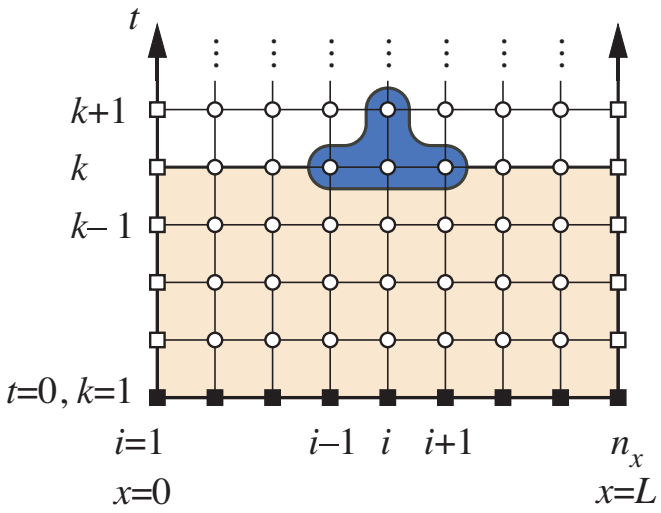
FTCS Approximation to the Heat Equation

Solve Equation (4) for u_i^{k+1}

$$\boxed{u_i^{k+1} = r u_{i+1}^k + (1-2r) u_i^k + r u_{i-1}^k} \qquad (5)$$

where $r = \alpha \Delta t / \Delta x^2$.

FTCS is an *explicit* scheme because it provides a simple formula to update u_i^{k+1} independently of the other nodal values at t_{k+1}.



`demoFTCS` **Code**

```
function errout = demoFTCS(nx,nt)

% ... Comments and processing of optional inputs skipped

% --- Assign physical and mesh parameters
alfa = 0.1;  L = 1;  tmax = 2;  % Diffusion coefficient, domain length and max time
dx = L/(nx-1);       dt = tmax/(nt-1);
r = alfa*dt/dx^2;    r2 = 1 - 2*r;

% --- Assign IC and BC.  u is initialized to a vector that includes BC
x = linspace(0,L,nx)';    u = sin(pi*x/L);

% --- Loop over time steps
for k=2:nt
  uold = u;    % prepare for next step
  for i=2:nx-1
    u(i) = r*uold(i-1) + r2*uold(i) + r*uold(i+1);
  end
end
```

Remember that the formula for updating u_i^{k+1} is

$$u_i^{k+1} = ru_{i+1}^k + (1-2r)u_i^k + ru_{i-1}^k$$

Alternative formulation to the FTCS Algorithm

Equation (5) can be expressed as a matrix multiplication.

$$u^{(k+1)} = Au^{(k)} \tag{6}$$

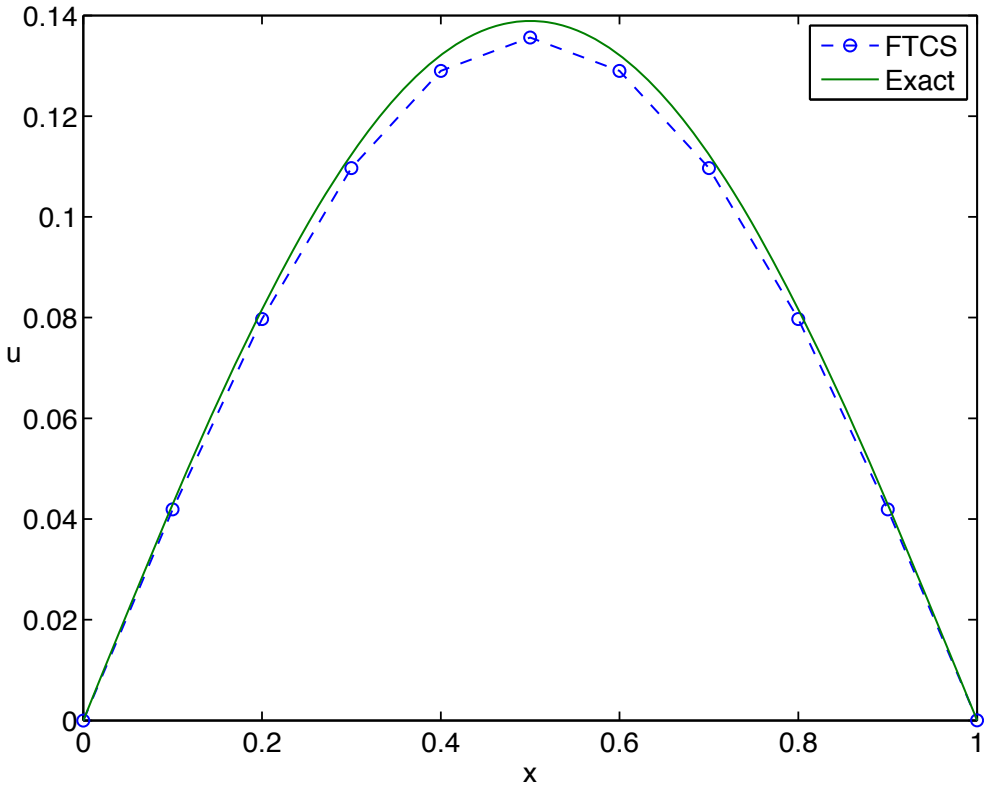
where $u^{(k+1)}$ is the vector of u values at time step $k+1$, $u^{(k)}$ is the vector of u values at time step k, and A is the *tridiagonal matrix*

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ r & (1-2r) & r & 0 & 0 & 0 \\ 0 & r & (1-2r) & r & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & r & (1-2r) & r \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

The first and last rows of A are set to enforce the Dirichlet boundary conditions at $x = 0$ and $x = L$.

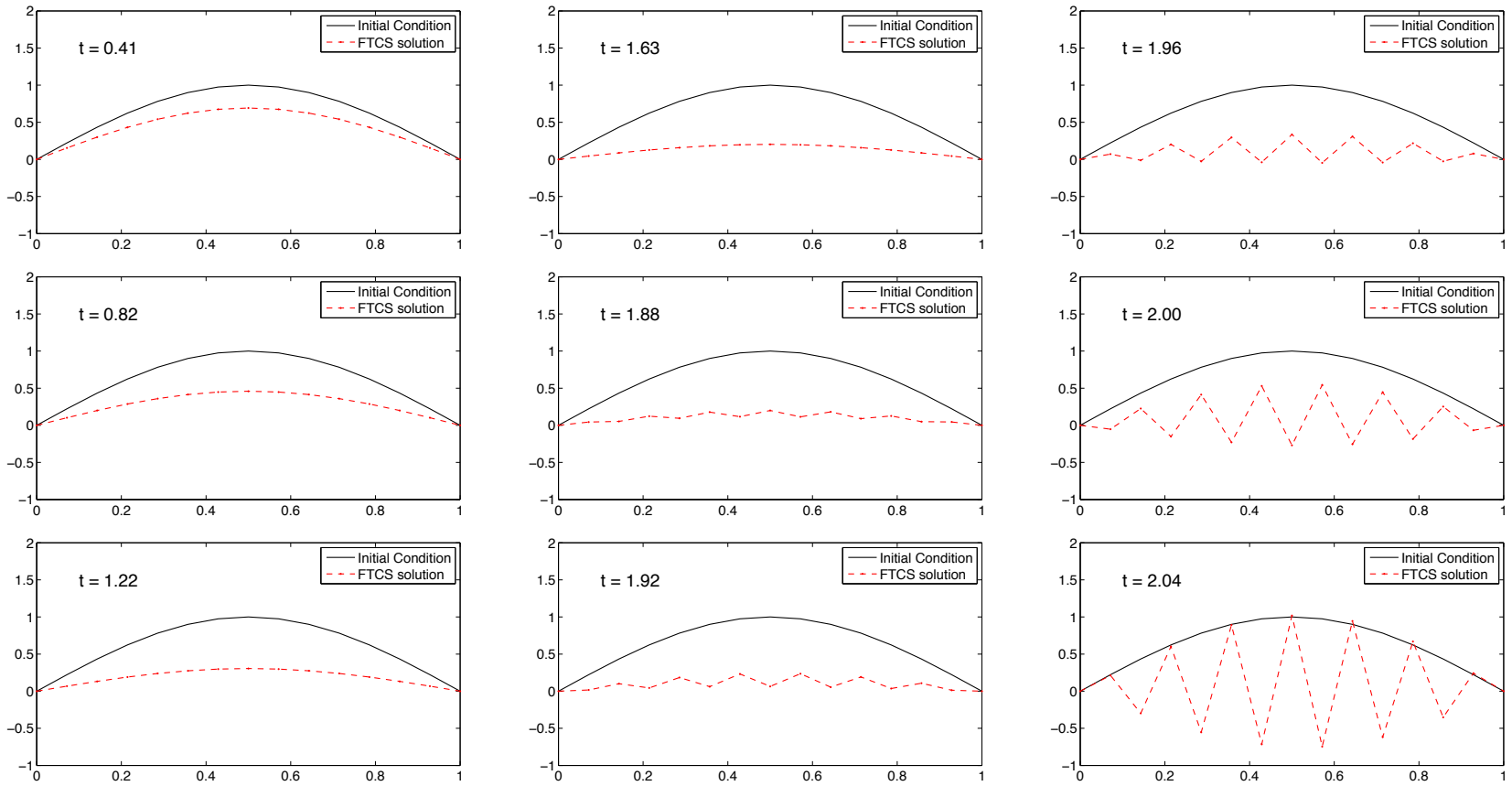
Run the `demoFTCS` code

```
>> demoFTCS
Error in FTCS solution = 0.002221
```



Unstable FTCS Solution

`movieFTCS(15):`



Conditionally Stable FTCS Scheme

Observations:

- The FTCS solution appears to be stable at first
- Toward the end of the simulation time, oscillations grow exponentially
- Instability is not caused by truncation error
- Instability is fed by round-off errors, but not directly caused by round-off

We'll use a simplified form of Fourier Stability Analysis

- Suppose that the initial condition is a small sine wave
- The correct solution is a decay of the sine wave
- Under what condition does the solution grow instead of decay?

Stability or instability is an intrinsic property of the scheme

FTCS is *conditionally stable* for solutions to the heat equation.

Stability Analysis

Suppose the initial condition looks like this

$$u_0(x) = \sigma \cos\left(\frac{\pi x}{\Delta x}\right) = \sigma(-1)^{i-1}$$

where σ is a suitably small value

The solution at time step $k = 2$ is

$$\begin{aligned} u_i^{(2)} &= r\big(u_0(x_{i+1}) + u_0(x_{i-1})\big) + (1-2r)u_0(x_i) \\ &= r\sigma\big((-1)^i + (-1)^{i-2}\big) + (1-2r)\sigma(-1)^{i-1} \\ &= -r\sigma\big((-1)^{i+1} + (-1)^{i-1}\big) + (1-2r)\sigma(-1)^{i-1} \end{aligned}$$

Stability Analysis

Since $(-1)^{i+1} = (-1)^{i-1}$ for any i, the solution at $k = 2$ can be further simplified

$$\begin{aligned} u_i^{(2)} &= -2r\sigma(-1)^{i-1} + (1-2r)\sigma(-1)^{i-1} \\ &= (1-4r)\sigma(-1)^{i-1} \\ &= (1-4r)u_0(x_i) \end{aligned}$$

The pattern is

$$\begin{aligned} u_i^{(2)} &= (1-4r)u_0(x_i) \\ u_i^{(3)} &= (1-4r)u_i^{(2)}(x_i) = (1-4r)^2 u_0(x_i) \\ u_i^{(4)} &= (1-4r)u_i^{(3)}(x_i) = (1-4r)^3 u_0(x_i) \end{aligned}$$

where $u_i^{(2)}$ is the numerical solution at x_i and $k = 2$;
and where $(1-4r)^2$ is the square of $(1-4r)$

Stability Analysis

The general pattern is

$$u_i^k = (1 - 4r)^{k-1} u_0(x_i).$$

where $(1 - 4r)^{k-1}$ is $(1 - 4r)$ raised to the $k - 1$ power.

Therefore, the solution grows when $|1 - 4r| > 1$.

Therefore, stability requires $1 - 4r < 1$ and $-(1 - 4r) < 1$.

$$1 - 4r < 1 \Longrightarrow -4r < 2 \Longrightarrow r > -\frac{1}{2} \quad \text{true for any } r > 0.$$

The second case is

$$-(1 - 4r) < 1 \Longrightarrow 4r < 2 \Longrightarrow r < \frac{1}{2}$$

Stability Analysis

The quick and dirty stability analysis shows that the FTCS scheme is stable only if

$$r = \frac{\alpha \Delta t}{\Delta x^2} < \frac{1}{2}. \tag{8}$$

Recall: α is a parameter of the physical problem.

We must choose Δt and Δx so that Equation (8) is satisfied.

Working with the FTCS Stability Criterion

To increase accuracy, we want to decrease both Δx and Δt.

For a given Δx, the stability limit for FTCS imposes an upper limit on Δt

$$\Delta t < \frac{\Delta x^2}{2\alpha} \tag{9}$$

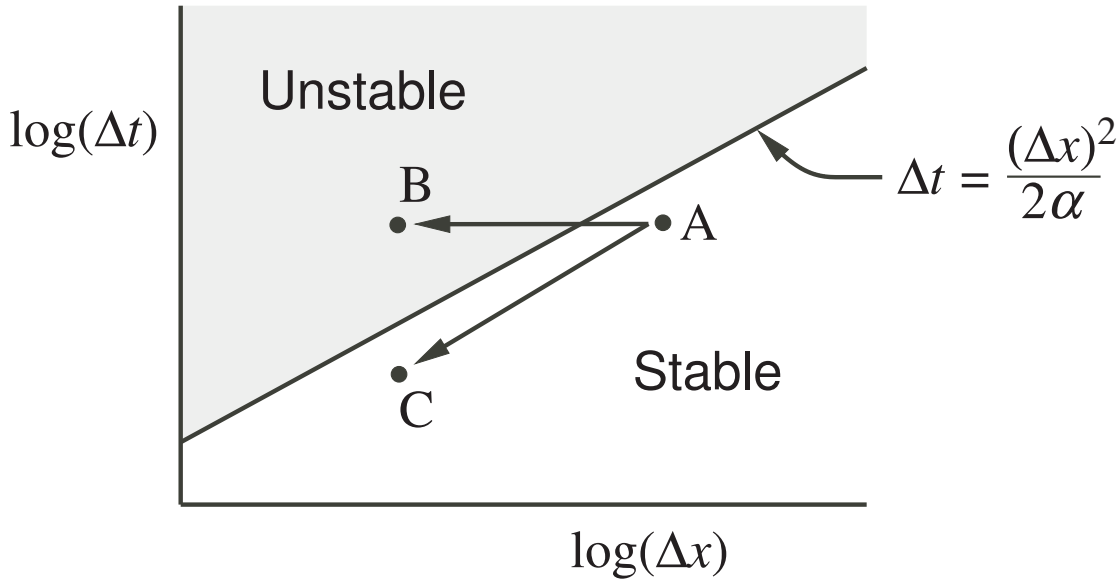
Choosing Δt and Δx so that $r < \frac{1}{2}$ does not guarantee an accurate numerical solution.

$r < \frac{1}{2}$ only guarantees that the FTCS solution will not blow up.

Working with the FTCS Stability Criterion

When reducing Δx and Δt to improve accuracy, follow a path like A$\rightarrow$C not A$\rightarrow$B.

Note that the axes have logarithmic scales.



Measuring the FTCS Truncation Error

The error reported by `demoFTCS` is defined as

$$E(n_x, n_t) = \frac{1}{\sqrt{n_x}} \| u_i^k - u(x_i, t_k) \|_2 \tag{10}$$

The factor of $1/\sqrt{n_x}$ converts $\| u_i^k - u(x_i, t_k) \|_2$ to an average error.

Measuring the FTCS Truncation Error

Designate the local error at $x = x_i$ and $t = t_k$ as

$$e_i^k = u_i^k - u(x_i, t_k). \tag{11}$$

Define $\bar{e}_k$ as an *RMS average error per node* at time step t_k

$$\bar{e}^k \equiv \left[\frac{1}{n_x} \sum_{i=1}^{n_x} (e_i^k)^2 \right]^{1/2} \tag{12}$$

Algebraic substitution shows that $E(n_x, n_t) = \bar{e}^k$.

If the solution to the heat equation is smooth, then $E(n_x, n_t) = \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$

Measuring the truncation error

Systematically reduce Δx and Δt to determine whether truncation error prediction is realized by the code.

We know the exact solution, so we use Equation(10).

This is a test of whether the code correctly implements FTCS.

Recall that for FTCS

$$E(n_x, n_t) = \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$$

but stability requires $\Delta t = C\Delta x^2$, where C is a constant.

Therefore, a stable solution to FTCS should demonstrate

$$E(n_x, n_t) = \mathcal{O}(\Delta x^2)$$

Measuring the truncation error

Suppose we don't know the exponent of the truncation error for our code.

In other words, instead of

$$E(n_x, n_t) = \mathcal{O}(\Delta x^2)$$

we have

$$E(n_x, n_t) = \mathcal{O}(\Delta x^p)$$

where p is uknown

Since $\Delta x = \dfrac{L}{n_x - 1}$ we have

$$E(n_x, n_t) = \mathcal{O}(\Delta x^p) = \mathcal{O}\left(\frac{L^p}{(n_x - 1)^p}\right)$$

Measuring the truncation error

Simplify the preceding expression

$$
\begin{aligned}
E(n_x, n_t) &= \mathcal{O}(\Delta x^p) \\
&= \mathcal{O}\left(\frac{L^p}{(n_x - 1)^p}\right) \\
&\sim \mathcal{O}\left(\frac{1}{(n_x - 1)^p}\right) \quad \text{ignore multiplicative constants} \\
&\sim \mathcal{O}\left(\frac{1}{n_x^p}\right) \quad \text{leading powers dominate}
\end{aligned}
$$

Therefore, we expect

$$
E(n_x, n_t) = \mathcal{O}\left(\frac{1}{n_x^p}\right)
$$

Measuring the truncation error

Measure $E(n_x, n_t)$ on two difference meshes, $n_{x,1}$ and $n_{x,2}$.

With

$$E(n_x, n_t) = \mathcal{O}\left(\frac{1}{n_x^p}\right)$$

we form the ratio

$$\frac{E(n_{x,2}, n_t)}{E(n_{x,1}, n_t)} = \frac{n_{x,1}^p}{n_{x,2}^p} = \left(\frac{n_{x,1}}{n_{x,2}}\right)^p$$

Solve for p

$$p = \frac{\log\left(E(n_{x,2}, n_t)/E(n_{x,1}, n_t)\right)}{\log\left(n_{x,1}/n_{x,2}\right)}.$$

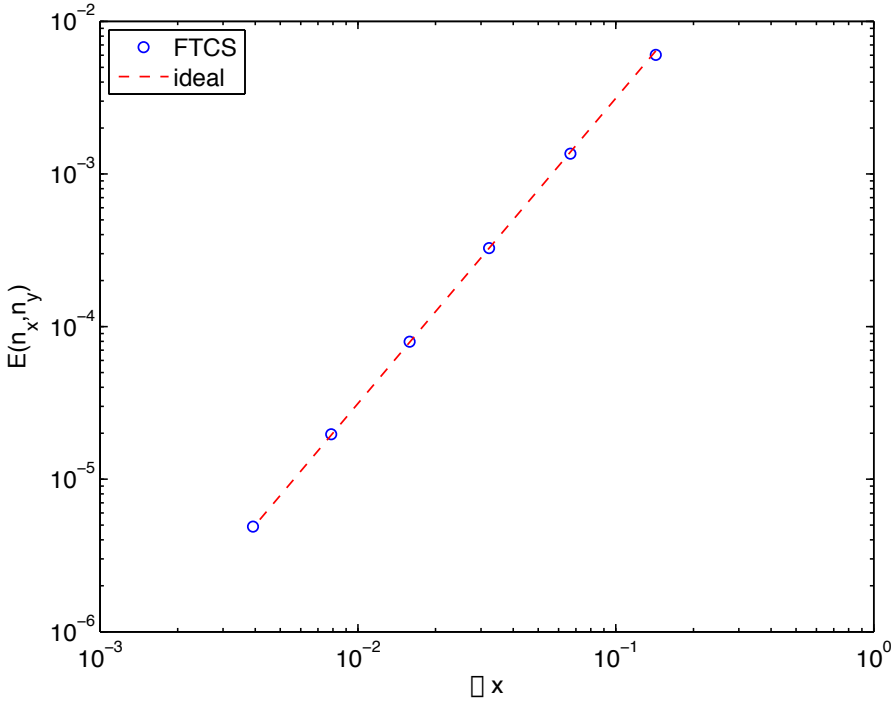
convFTCS **Code**

```
function convFTCS
% convFTCS  Convergence of FTCS on a series of finer spatial meshes

% --- Set constants to be consistent with demoFTCS
alfa = 0.1;  L = 1;  tmax=2;   rsafe = 0.49999;  %  stable r<0.5

% --- Specify nx and compute nt consistent with stability limit
nx = [8 16 32 64 128 256];    dx = L./(nx-1);
nt = ceil( 1 + alfa*tmax*((nx-1).^2)/(rsafe*L^2) );

% --- Loop over mesh sizes, store error and compute order of scheme
fprintf('\n    nx     nt     error     E(j)/E(j-1)    p\n');
er = NaN;  p = 0;
for j=1:length(nx);
  e(j) = demoFTCS(nx(j),nt(j));
  if j>1
    er = e(j)/e(j-1);  p = log(er)/log(nx(j-1)/nx(j));
  end
  fprintf(' %5d  %5d  %11.3e  %8.4f  %8.4f\n',nx(j),nt(j),e(j),er,p);
end

% -- plotting code skipped ...
```

Running the `convFTCS` Code

```
>> convFTCS

    nx      nt      error      E(j)/E(j-1)     p
     8      21     6.028e-03        NaN     0.0000
    16      92     1.356e-03     0.2249     2.1524
    32     386     3.262e-04     0.2406     2.0553
    64    1589     7.972e-05     0.2444     2.0329
   128    6453     1.970e-05     0.2471     2.0170
   256   26012     4.895e-06     0.2485     2.0085
```

The last column of text output shows that `demoFTCS` has the right truncation error behavior.



Summary for the FTCS Scheme

- FTCS is easy to implement. The update formula is

$$u_i^{k+1} = ru_{i+1}^k + (1-2r)u_i^k + ru_{i-1}^k$$

- The FTCS scheme is conditionally stable when

$$r = \frac{\alpha \Delta t}{\Delta x^2} < 1/2$$

  In two spatial dimensions with $\Delta y = \Delta x$, the stability condition is $r < 1/4$. In 3D with $\Delta y = \Delta z = \Delta x$, the stability condition is $r < 1/8$.
- There are *much better* schemes for solving the heat equation.
- FTCS is a toy used to introduce the numerical solution of PDEs