Numerical Solutions to PDEs

Lecture Notes #6
Stability of Lax-Wendroff and Crank-Nicolson; Boundary Conditions

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2018

Outline

San Diego State University

Last Time

We introduced the concept of **order of accuracy**, which essentially is the measure of how fast a finite difference scheme **converges** to the solution of the PDE, as we refine the grid.

The order of accuracy is identified (using Taylor expansions) as

$$\mathcal{O}\left(k^p + h^q\right) \equiv \text{order-}(p,q), \text{ or with } k = \Lambda(h), \ \mathcal{O}\left(h^\rho\right) \equiv \text{order-}\rho$$

Two new schemes were introduced: The Lax-Wendroff (explicit) and Crank-Nicolson (implicit) schemes, both are order-(2,2).

We introduced quite a bit of technology: the concept of *symbols* ("fingerprints") of the schemes and PDEs, and congruence to zero modulo a symbol, so that in the end the analysis comes down to a mechanical Taylor expansion and identification of terms.

San Diego State University

Stability of the Lax-Wendroff Scheme

We apply the Lax-Wendroff scheme

$$v_m^{n+1} = v_m^n - \frac{a\lambda}{2}\left(v_{m+1}^n - v_{m-1}^n\right) + \frac{a^2\lambda^2}{2}\left(v_{m+1}^n - 2v_m^n + v_{m-1}^n\right),$$

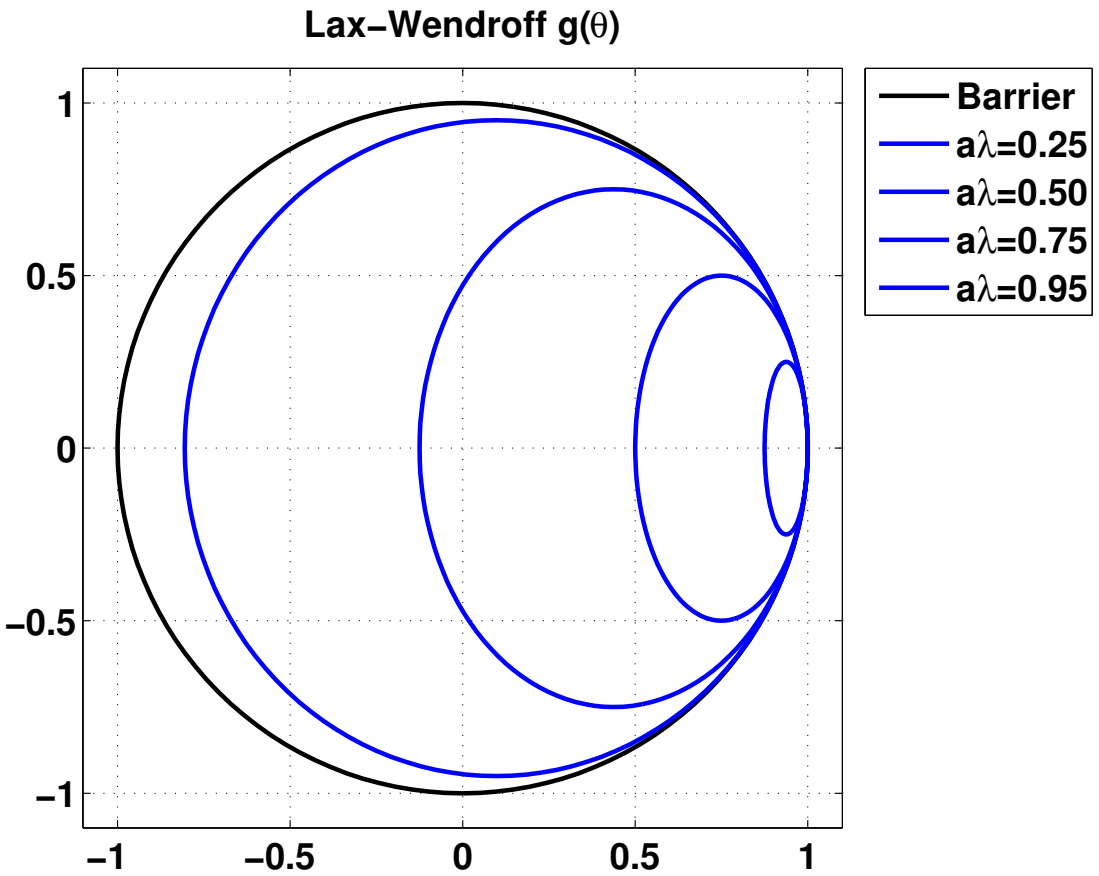
to the one-way wave equation, with right-hand-side $f = 0$ in order to identify the amplification factor.

As per our recipe we set $v_m^n \rightsquigarrow g^n\, e^{im\theta}$, where $\theta = h\xi$, and get

$$\begin{aligned} g(\theta) &= 1 - \frac{a\lambda}{2}\left(e^{i\theta} - e^{-i\theta}\right) + \frac{a^2\lambda^2}{2}\left(e^{i\theta} - 2 + e^{-i\theta}\right) \\ &= 1 - ia\lambda\sin(\theta) - a^2\lambda^2(1-\cos(\theta)) \\ &= 1 - 2a^2\lambda^2\sin^2\left(\frac{\theta}{2}\right) - ia\lambda\sin(\theta). \end{aligned}$$

$$|g(\theta)|^2 = \left[1 - 2a^2\lambda^2\sin^2\left(\frac{\theta}{2}\right)\right]^2 + [a\lambda\sin(\theta)]^2$$

Stability of the Lax-Wendroff Scheme



Stability of the Lax-Wendroff Scheme

$$
\begin{aligned}
|g(\theta)|^2 &= \left[1 - 2a^2\lambda^2 \sin^2\left(\frac{\theta}{2}\right)\right]^2 + [a\lambda \sin(\theta)]^2 \\
&= \left[1 - 2a^2\lambda^2 \sin^2\left(\frac{\theta}{2}\right)\right]^2 + \left[2a\lambda \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\right]^2 \\
&= 1 - 4a^2\lambda^2 \sin^2\left(\frac{\theta}{2}\right) + 4a^4\lambda^4 \sin^4\left(\frac{\theta}{2}\right) \\
&\quad +4a^2\lambda^2 \sin^2\left(\frac{\theta}{2}\right)\cos^2\left(\frac{\theta}{2}\right) \\
&= 1 - 4a^2\lambda^2 \sin^2\left(\frac{\theta}{2}\right)\left[1 - \cos^2\left(\frac{\theta}{2}\right)\right] + 4a^4\lambda^4 \sin^4\left(\frac{\theta}{2}\right) \\
&= 1 - 4a^2\lambda^2 \sin^4\left(\frac{\theta}{2}\right) + 4a^4\lambda^4 \sin^4\left(\frac{\theta}{2}\right) \\
&= 1 - 4a^2\lambda^2 \left[1 - a^2\lambda^2\right] \sin^4\left(\frac{\theta}{2}\right)
\end{aligned}
$$

SAN DIEGO STATE UNIVERSITY

Stability of the Lax-Wendroff Scheme

$$|g(\theta)|^2 = 1 - \underbrace{4a^2\lambda^2\left[1 - a^2\lambda^2\right]\sin^4\left(\frac{\theta}{2}\right)}_{\gamma(\theta)}$$

In order for $|g(\theta)| \leq 1$, we must have that $0 \leq \gamma(\theta) \leq 2$. This gives us the condition $|a\lambda| \leq 1$. $\square$

At this point we know

Lax-Wendroff	
Mode	Explicit
Order of Accuracy	(2,2)
Stability Criterion	$\lvert a\lambda\rvert \leq 1$ (CFL)

Stability of the Crank-Nicolson Scheme

After the excitement of verifying the stability for the Lax-Wendroff scheme, we now attack the Crank-Nicolson scheme

$$\frac{v_m^{n+1} - v_m^n}{k} + a\frac{v_{m+1}^{n+1} - v_{m-1}^{n+1} + v_{m+1}^n - v_{m-1}^n}{4h} = 0,$$

with the same toolbox.

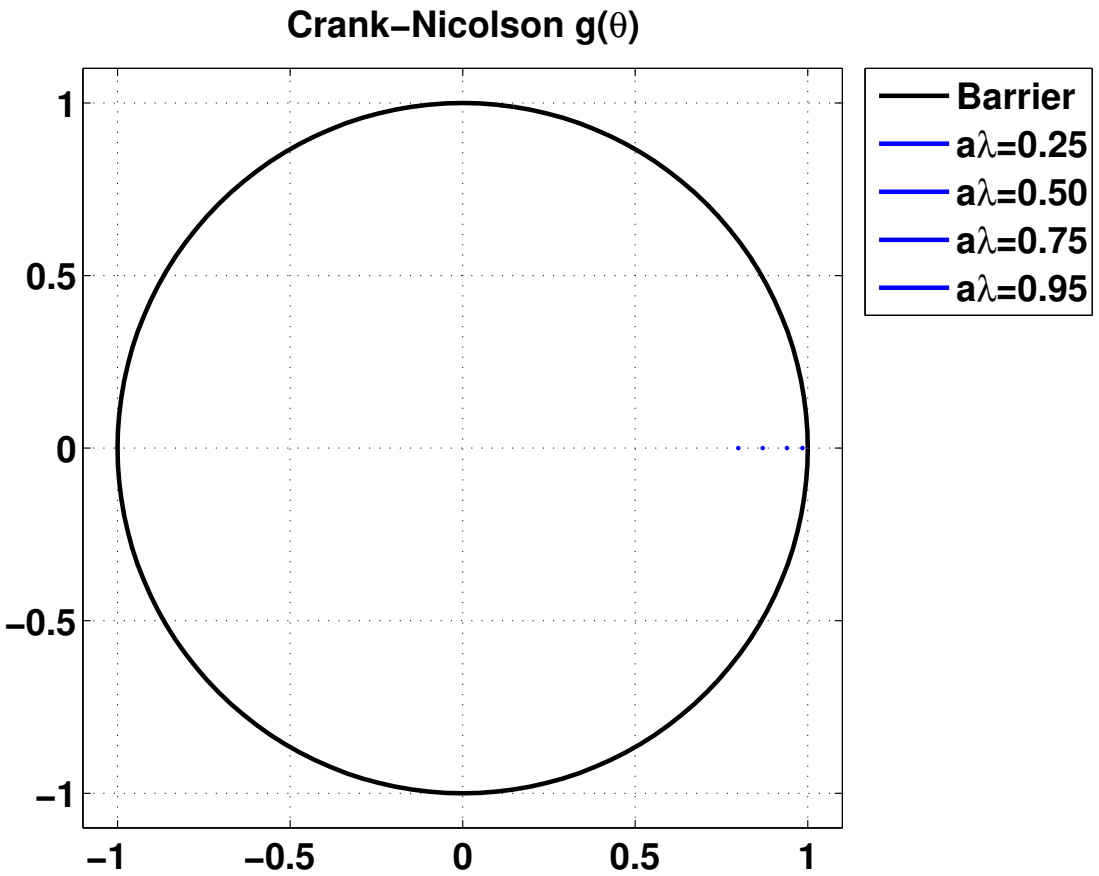
The usual $v_m^n \rightsquigarrow g^n\, e^{im\theta}$, gives us

$$\frac{g-1}{k} + a\frac{g(e^{i\theta} - e^{-i\theta}) + (e^{i\theta} - e^{-i\theta})}{4h} = 0,$$

so that

$$g - 1 + ia\lambda\frac{(g+1)\sin(\theta)}{2} = 0.$$

Stability of the Crank-Nicolson Scheme



Stability of the Crank-Nicolson Scheme

We write

$$g - 1 + ia\lambda \frac{(g+1)\sin(\theta)}{2} = 0,$$

as

$$g\left[1 + \frac{ia\lambda}{2}\sin(\theta)\right] - \left[1 - \frac{ia\lambda}{2}\sin(\theta)\right] = 0.$$

Finally,

$$g(\theta) = \frac{1 - \frac{ia\lambda}{2}\sin(\theta)}{1 + \frac{ia\lambda}{2}\sin(\theta)}, \quad |\mathbf{g}(\theta)|^2 = \frac{\mathbf{1} + \frac{\mathbf{a^2\lambda^2}}{\mathbf{4}}\sin^2(\theta)}{\mathbf{1} + \frac{\mathbf{a^2\lambda^2}}{\mathbf{4}}\sin^2(\theta)} = \mathbf{1}.$$

Hence,

Property: Unconditional Stability of Crank-Nicolson

The Crank-Nicolson scheme is stable for any value of $a\lambda$, we say that it is **unconditionally stable**.

Summary: Lax-Wendroff vs. Crank-Nicolson

	Lax-Wendroff	**Crank-Nicolson**
Mode	Explicit	Implicit
Order of Accuracy	(2,2)	(2,2)
Stability Criterion	$\lvert a\lambda\rvert \leq 1$ (CFL)	Unconditionally Stable

The Lax-Wendroff scheme is easier to propagate (since it is explicit), but if the speed a is large, the stability criterion may impose a severe time-step restriction, recall $k = h/|a|$.

The fact that the Crank-Nicolson scheme is unconditionally stable makes it (and variants) extremely useful; the only down-side is that for each time-step we must solve a linear system $A\bar{\mathbf{v}}^{n+1} = \bar{\mathbf{b}}_n(\bar{\mathbf{v}}^n)$.

Difference Notation and the Difference Calculus

We introduce the following notation

$$\underbrace{\delta_+ v_m = \frac{v_{m+1} - v_m}{h}}_{\textbf{Forward Difference}}, \quad \underbrace{\delta_- v_m = \frac{v_m - v_{m-1}}{h}}_{\textbf{Backward Difference}}$$

$$\underbrace{\delta_0 v_m = \frac{1}{2}(\delta_+ + \delta_-)v_m = \frac{v_{m+1} - v_{m-1}}{2h}}_{\textbf{Central Difference}}.$$

Further, we can define the second difference operator $\delta^2 = \delta_+\delta_- \equiv \frac{\delta_+ - \delta_-}{h}$:

$$\delta^2 v_m = \frac{v_{m+1} - 2v_m + v_{m-1}}{h^2}.$$

We can define the corresponding time-differences δ_{t+}, δ_{t-}, δ_{t0}, and δ_t^2...

What's The Use??? — Deriving Higher-Order Approximations

Consider the Taylor expansion of the central difference operator

$$\delta_0 u = \frac{du}{dx} + \frac{h^2}{6}\frac{d^3u}{dx^3} + \mathcal{O}\left(h^4\right) = \left[1 + \frac{h^2}{6}\delta^2\right]\frac{du}{dx} + \mathcal{O}\left(h^4\right)$$

where, in the second equality we have used

$$\delta^2 u = \frac{d^2u}{dx^2} + \mathcal{O}\left(h^2\right).$$

Now, formally (symbolically)

$$\frac{du}{dx} = \left[1 + \frac{h^2}{6}\delta^2\right]^{-1}\delta_0 u + \mathcal{O}\left(h^4\right).$$

The inverse operator $[\circ]^{-1}$ is (almost) always eliminated by operating on both sides with the operator $[\circ]$ itself...

Example: A Fourth Order Approximation to $u_x = f$

Applying what we have developed to the equation

$$\frac{du}{dx} = f,$$

we get

$$\begin{aligned}
\left[1 + \frac{h^2}{6}\delta^2\right]^{-1} \delta_0 v_m &= f_m && (1)\\
\delta_0 v_m &= \left[1 + \frac{h^2}{6}\delta^2\right] f_m && (2)\\
\frac{v_{m+1} - v_{m-1}}{2h} &= f_m + \frac{1}{6}\left[f_{m+1} - 2f_m + f_{m-1}\right] && (3)\\
&= \frac{1}{6}\left[f_{m+1} + 4f_m + f_{m-1}\right].
\end{aligned}$$

If/When the right-hand-side is simply f_m, we only have a second order approximation...

Example: Another Possibility — 4th Order Scheme

We can go a slightly different route:

$$\delta_0 u = \frac{du}{dx} + \frac{h^2}{6}\frac{d^3u}{dx^3} + \mathcal{O}\left(h^4\right) = \frac{du}{dx} + \frac{h^2}{6}\delta^2\delta_0 u + \mathcal{O}\left(h^4\right),$$

so that

$$\left[1 - \frac{h^2}{6}\delta^2\right]\delta_0 u = \frac{du}{dx} + \mathcal{O}\left(h^4\right).$$

From which we get the fourth order scheme

$$\frac{-v_{m+2} + 8v_{m+1} - 8v_{m-1} + v_{m-2}}{12h} = f_m.$$

Clearly, this notation may come in handy...

Boundary Conditions — Physical

We have seen that when we solve IVPs in finite physical domains, we need **physical boundary conditions** at the boundaries where characteristics enter the domain.

— This corresponds to a physical process, such as keeping a temperature constant (or varying in time), regulating the flow of water through the turbines in a dam, the absorption of sound in the ceiling and walls of your subterranean media room...

Clearly, a numerical scheme must accurately capture these physical boundary conditions.

San Diego State University

Boundary Conditions — (Additional) Numerical

Further, many numerical schemes also require additional boundary conditions, called **numerical boundary conditions** in order for the solution to be well defined (unique).

Numerical boundary conditions often arise for reasons similar to the ones that impose the need for additional numerical initial conditions (for multi-step schemes) and/or to make a finite computational domain "act" infinite.

Dealing with boundary conditions, physical and/or numerical is many times the most difficult part of simulating a PDE.

Boundary Conditions

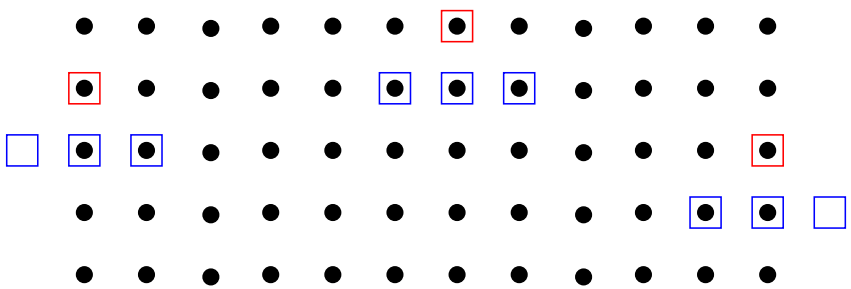
For our discussion we use the Lax-Wendroff scheme

$$v_m^{n+1} = v_m^n - \frac{a\lambda}{2}\left(v_{m+1}^n - v_{m-1}^n\right) + \frac{a^2\lambda^2}{2}\left(v_{m+1}^n - 2v_m^n + v_{m-1}^n\right),$$

applied to the equation

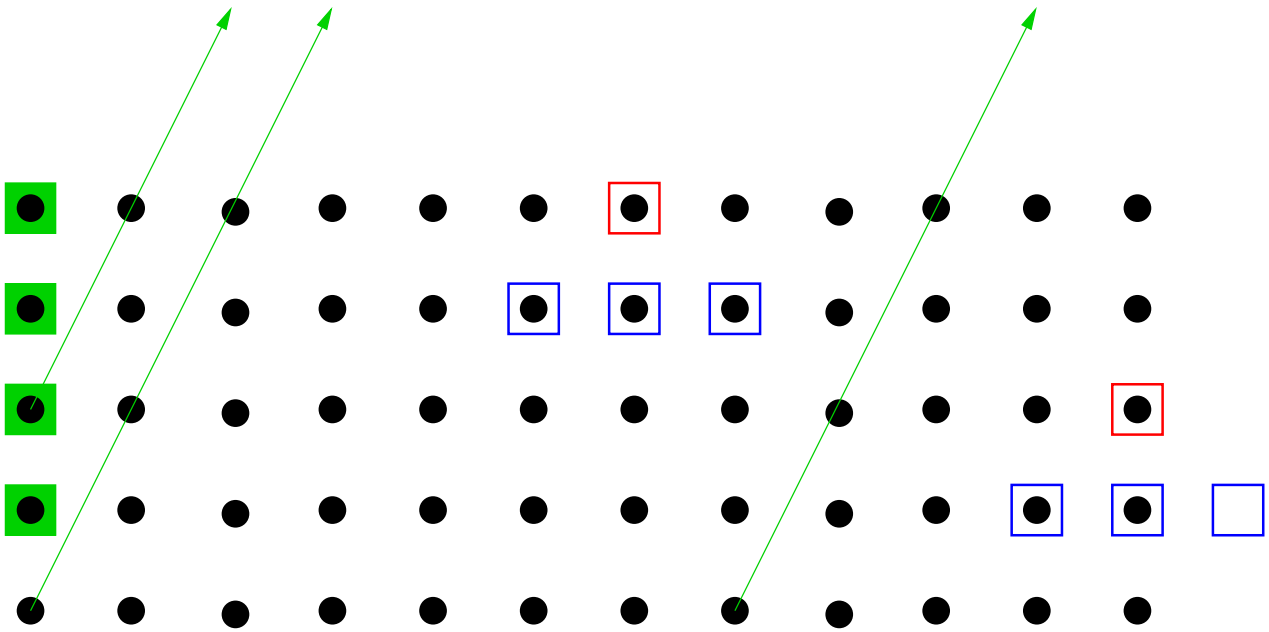
$$u_t + au_x = 0, \quad 0 \le x \le 1, \ t \ge 0.$$

But, we run into problems at the boundaries, some points are "missing:"



Boundary Conditions

Let's assume that $k = h$, *i.e.* $\lambda = 1$, and $a = 0.5 > 0$, then the characteristics come in from the left, and we must have [due to well-posedness] a physical boundary condition there:



We still need do so **something** at the right boundary, x_M...

Examples of Numerical Boundary Conditions

Here are some possibilities:

$$v_M^{n+1} = v_{M-1}^{n+1} \quad (1)$$

$$v_M^{n+1} = 2v_{M-1}^{n+1} - v_{M-2}^{n+1} \quad (2)$$

$$v_M^{n+1} = v_{M-1}^{n} \quad (3)$$

$$v_M^{n+1} = 2v_{M-1}^{n} - v_{M-2}^{n-1} \quad (4)$$

Formulas (1) and (2) are simple extrapolations of interior grid points to the boundary. Formulas (3) and (4) are referred to as **quasi-characteristic extrapolation**, since the extrapolation uses points "near" the characteristics. Usually, but not always, it is better to use **one-sided** difference formulas at the boundaries, *i.e.*

$$v_M^{n+1} = v_M^n - a\lambda(v_M^n - v_{M-1}^n). \quad (5)$$

Problems that Can Occur — Accuracy & Stability

The decision on what to do at the boundary may seem like a small one... We're only talking about what to do at **one** point, and quite possibly we may have thousands or billions of interior points...

However, **any error we introduce at the boundary will eventually affect the entire solution:**

If we have a 4th order scheme (in the interior), but use a sloppy 1st order one-sided difference at the boundary, then the overall scheme is only 1st order.

In addition, **the boundary condition will affect the stability of the scheme!**

We will re-visit these issues again (and again), in more detail; for now, we ponder the table on the next slide.

Stability Impact of Boundary Conditions

Scheme	Boundary Condition (slide 20) (1)	(2)	(3)	(4)
Leapfrog	**unstable**	**unstable**	stable†	stable†
Crank-Nicolson	stable‡	stable‡	$a\lambda < 2$	$a\lambda < 2$

† conditionally stable; ‡ unconditionally stable.

The effect of incorrect boundary conditions are usually oscillations in the solution. These oscillations may be observed **away** from the boundary, which makes it hard to correctly diagnose the cause of the problem.

Usually, if you suspect that an unstable numerical boundary condition is causing instability, the easiest way to pinpoint the problem is to change the boundary condition and observe the solution.

We will return to the analysis of boundary conditions, but we need more (mathematical) tools before we do so.

Propagating Crank-Nicolson: Solving Tridiagonal Systems

We now return to the issue of propagating the Crank-Nicolson scheme

$$\frac{v_m^{n+1} - v_m^n}{k} + a\frac{v_{m+1}^{n+1} - v_{m-1}^{n+1} + v_{m+1}^n - v_{m-1}^n}{4h} = 0.$$

We rewrite this so we have all the unknown terms to the left, and the known terms to the right

$$-\left[\frac{a}{4h}\right] v_{m-1}^{n+1} + \left[\frac{1}{k}\right] v_m^{n+1} + \left[\frac{a}{4h}\right] v_{m+1}^{n+1} = \underbrace{\frac{a}{4h} v_{m-1}^n + \frac{1}{k} v_m^n - \frac{a}{4h} v_{m+1}^n}_{b_m^n},$$

where if the x-grid is given by $x_0, x_1, \ldots, x_M$, the index m runs from 1 to $(M-1)$, and we apply appropriate boundary conditions at x_0 and x_M.

Propagating Crank-Nicolson: Solving Tridiagonal Systems

This gives rise to a tri-diagonal system

$$\begin{bmatrix} 1 & 0 & & & \\ -\alpha & \beta & \alpha & & \\ & \ddots & \ddots & \ddots & \\ & & -\alpha & \beta & \alpha \\ & & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{M-1} \\ v_M \end{bmatrix}^{n+1} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{M-1} \\ 0 \end{bmatrix}^{n}$$

where $\alpha = a/4h$, $\beta = 1/k$; $b_0^n = \varphi(t_{n+1})$ is the specification of the physical boundary condition at (t_{n+1}, x_0); the last row $-v_{M-1} + v_M = 0$ corresponds to (BC-1 on slide 20); and b_1^n through b_{M-1}^n are computed according to the previous slide.

San Diego State University

Propagating Crank-Nicolson: Solving Tridiagonal Systems

A tri-diagonal system like this can be solved on $\mathcal{O}(M)$ operations, which should be compared with the more general requirement $\mathcal{O}(M^3)$ for a full matrix. The **Thomas Algorithm** is discussed in **Math 541**, and is also presented in Strikwerda.

A matlab implementation (without error checking, for brevity) is presented on the next slide.

In order for the algorithm to work, the tridiagonal matrix T **"must be in compact form: the sub-diagonal elements in the first column, the diagonal in the second column, and the super-diagonal in the third column. [...] Note that T(1,1) and T(n,3) are never accessed, i.e. the sub-diagonal entries start on the second row, and the super-diagonal elements end on the (n-1)st row."**

Thomas Algorithm for Tridiagonal Systems

Thomas Algorithm for Tridiagonal Systems [matlab]

```
function [x] = trisolve(T,b)
[n,m] = size(T);
work = zeros(n,1);
work(1) = T(1,2);
x(1,:)  = b(1,:);
% Forward sweep.
for i=2:n
    beta = T(i,1)/work(i-1);
    x(i,:)  = b(i,:)  - beta*x(i-1,:);
    work(i) = T(i,2) - beta*T(i-1,3);
end
x(n,:)  = x(n,:)/work(n);
% Backward sweep.
for i=n-1:-1:1
    x(i,:)  = (x(i,:)  - T(i,3)*x(i+1,:))  / work(i);
end
```

Homework #2 — Due 2/23/2018, 12:00pm

Strikwerda-2.1.4 — Theoretical

Strikwerda-2.1.5 — Theoretical

Strikwerda-2.2.1 — Theoretical

Strikwerda-2.2.4 — Theoretical

Strikwerda-3.2.1 — Theoretical

Strikwerda-3.2.3 — Theoretical

Strikwerda-3.4.1 — Numerical

SAN DIEGO STATE UNIVERSITY