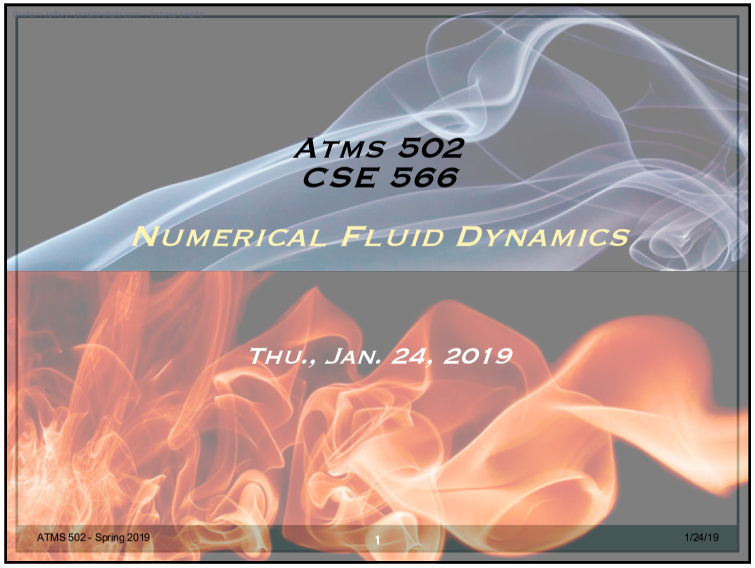


2

ATMS 502
CSE 566

Thursday,
24 January 2019

Class #4

Plan for Today

- 1) Review
  Predator-Prey
  Stampede-2 information
- 2) Resolution
  Terminology
- 3) NUMERICAL METHODS :
  Takacs paper (introduction)
  Truncation error
- 4) CODE/DATA:
  Program #2

ATMS 502 - Spring 2019 1/24/19

Review: Predator-Prey problem

3

$$\frac{dA}{dt} = \alpha A + \beta AB \qquad \frac{dB}{dt} = \gamma B + \delta AB$$

A *Taylor series* expansion in time, cut off after 1st derivative:

$$f(t+\Delta t) = f(t) + \Delta t \cdot f'(t) + \frac{\Delta t^2}{2!} f''(t) + \ldots \quad \text{so:}$$

NEGLECT!

$$f'(t) = \frac{df}{dt} \approx \left(\frac{f(t+\Delta t) - f(t)}{\Delta t} \right)$$

- This is a *finite difference approximation to the time derivative* df/dt.

ATMS 502 - Spring 2019 C.006: Finite differences; C.007: Truncated Taylor series 1/24/19

Stampede-2 – use summary

4

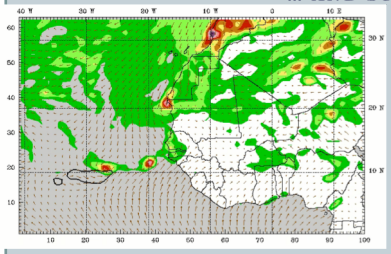
- Working in your home directory is fine.
  - But make occasional copies of your files to $WORK
  - Remember HOME is backed up; WORK is not.
- Future
  - Anticipate the day (by pgm 6) when you need to use $WORK
    - you go there by *cd $WORK* or by typing *cdw*
    - when that day comes, working in $WORK, you will need to regularly copy critical (source code, script) files back HOME.
- Consider ...
  - Mailing yourself or otherwise occasionally copy your code back to your PC.... it is good to have a copy if Stampede goes down.

ATMS 502 - Spring 2019 1/24/19

Resolution

5

... AND SCALE.



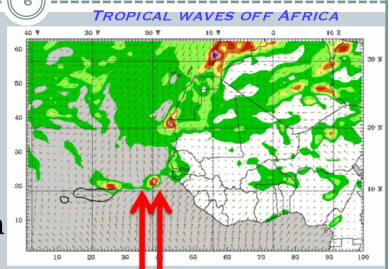
We'll start with this movie shown earlier.

The *domain* is large.
The *resolution* is coarse.
The *scale* of the phenomena is small, as seen here.

What does this all mean?

Terminology: Resolution and scale

6

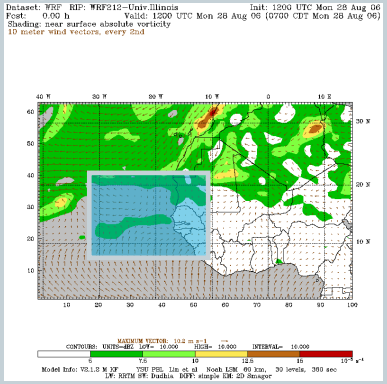


- *Plot*: simulated surface rotation (shaded)
- *Region*: west Africa
- *Grid size:* 100x63
- *Grid spacing*: 60 km
- *Domain:* 5940x3720 km
  - $(nx-1)*\Delta x = 5940$ km
- *Scale* of phenomena: ~300 km
- *Resolution*: coarse *(poor)* – is near 5 grid points wide.
  - **Scale** is a physical quantity (e.g. measured in km).
  - ***Resolution*** is always measured in terms of the grid spacing.

Resolution: Accuracy and cost

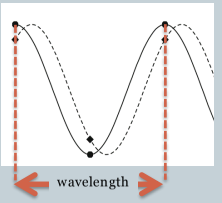
7

- Two runs – varying grid spacing Δx
  - Single 60-km grid
    - Quicker execution
    - Large Δx, large Δt
    - Lower accuracy
  - 2-grids: 20-km *nest*
    - Slower execution
    - Inner nested grid used small Δx, small Δt
    - Better accuracy (less truncation error)
    - Longer execution time



Wavenumber vs. wavelength

8



- Wavelength
  - dimensional
  - is a physical phenomena in, e.g., km
  - For a truly wavelike feature, it is the distance between successive wave peaks or troughs
- Wavenumber
  - dimensionless
  - is the *number* of wavelengths that appear in a computational domain
  - our Program 1 has a wavenumber-1 initial condition
  - 5 complete waves in a 1D simulation > wavenumber = 5

Resolution: extremes

9

- An infinitely long wave
  - … is a straight line.
- A well-resolved wave
  - … has 7-10 (or more) grid points over a wavelength
- A poorly-resolved wave
  - … has 3-5 grid points spanning a wavelength
- A 2Δx *("two delta-x")* wave
  - … is basically unresolved at all (see Durran figure), though we say 2Δx is the "minimum" wavelength we could describe.
  - … similarly, a 2Δt wave appears over the span of two time steps. We say 2Δt is the minimum period we could describe.

Poor resolution: Problems

10

- An under-resolved wave …
  - will have (or develop) problems with amplitude
  - will have (or develop) problems with *phase speed*
  - will have (or develop) problems with *group velocity*

  } 3 key properties we will often discuss
  - will probably not last very long
    - because numerical models tend to damp these out
  - its appearance may be *wrong* – it may not look like a coarse representation of a fine-scale feature!
- An unresolved wave …
  - less than 2Δx wavelength or 2Δt period
  - will be *aliased* to appear to have a longer wavelength or period
  - this is always bad!
    - *can* cause the simulation to fail, or spurious waves to form

2Δx waves: misrepresentation

11

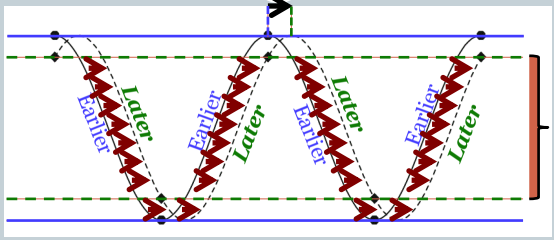


Fig. 3.4 Misrepresentation of a $2\Delta x$ wave translating to the right as a decaying standing wave when the wave is sampled at fixed grid points on a numerical mesh. The grid-point values are indicated by *dots* at the earlier time and *diamonds* at the later time

The basic problem is that there are only two possible configurations, differing by a phase angle of 180°, in which 2Δx waves can appear on a finite mesh"

- Durran, p. 104

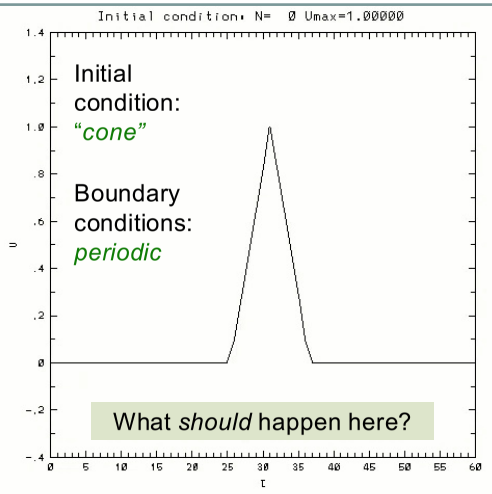
Numerical methods: Takacs (1985)

12

- ORDER OF (SPATIAL) ACCURACY
- DOMINANT TYPES OF ERRORS
- HIS METHOD (USED LATER)

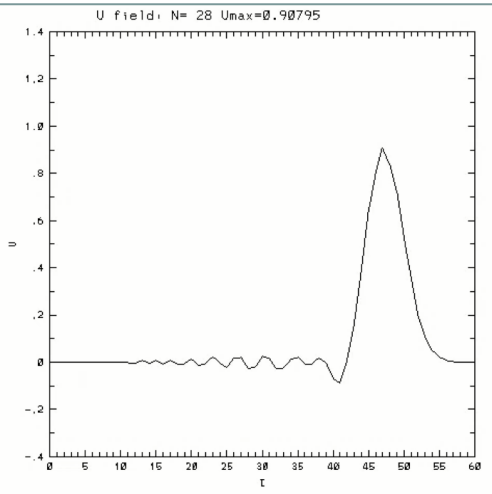
References:
- C001 (Lax-Wendroff)
- C006 (Finite differences)
- C007 (Taylor series)
- C052 (Advection)

4 Numerical methods … the first is Lax-Wendroff.



Four Numerical methods … the first is Lax-Wendroff.

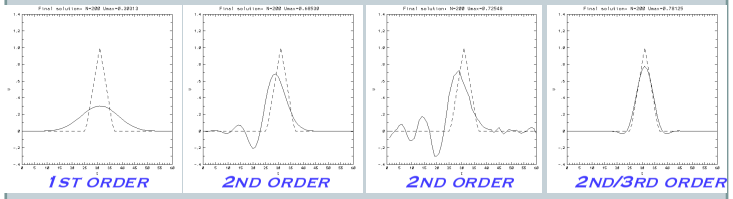
- What types of errors do you see?
- How would you quantify or categorize these errors?



Takacs (1985)

15

- Considerations:
  - Odd-order schemes generally dissipative
    - *amplitude errors*
  - Even-order schemes generally dispersive
    - *phase errors*



1ST ORDER	2ND ORDER	2ND ORDER	2ND/3RD ORDER
UPSTREAM	LAX-WENDROFF	LEAPFROG	TAKACS

Numerical methods: Truncation error

16

OBJECTIVES:

- *QUANTIFY THE DISCRETIZATION ERROR;*
- *DETERMINE IF A SCHEME IS CONSISTENT;*
- *LEARN TO ANTICIPATE ERROR CHARACTERISTICS.*

Truncation error

17

- There is a discretization or *truncation* error for any finite difference scheme.
  - We want to quantify this error.
  - Substitute Taylor series into the scheme.

- Example #1: *diffusive* process, $q_t=Kq_{xx}$
- Approximate with the following:

$$\frac{q_j^{n+1}-q_j^n}{\Delta t} = K\frac{\left(q_{j+1}^n - 2q_j^n + q_{j-1}^n\right)}{\Delta x^2}$$

- Remember our notation,
  - subscript *j* represents grid (space) index: j=1,...,nx
  - superscript *n* represents time level; *n+1* is next step

Truncation error: Derivation

18

$$\frac{q_j^{n+1}-q_j^n}{\Delta t} = K\frac{\left(q_{j+1}^n - 2q_j^n + q_{j-1}^n\right)}{\Delta x^2}$$

Truncation error, and consistency

19

- We rearrange the result to have the *original* PDE on the left; everything on the right side is the *truncation error*.

$$q_t - Kq_{xx} = \left[\frac{-\Delta t}{2!}q_{tt} + 2K\frac{\Delta x^2}{4!}q_{xxxx} + (\text{higher terms})\right]$$

- The right side [] *is not* zero.
- Truncation error: order ($\Delta t, \Delta x^2$)
- Definition: A scheme is consistent if the truncation error approaches zero in limit as $\Delta t \to 0$ and $\Delta x \to 0$.

Review: Truncation error

20

- Truncation error:
  - Tells us if the scheme reduces to the PDE as Δt, Δx (,Δy,Δz..)$\to 0$.
  - Tells us the discretization error, and accuracy
- But:
  - It *does not* tell us anything about what choices to make for Δt, Δx, etc
  - It *does not* tell us whether we will get a reasonable solution with finite Δt, Δx

Truncation error – Example 2

21

- Example: advection process, $q_t = -cq_x$
- Approximate with the following:

$$\frac{q_j^{n+1} - q_j^n}{\Delta t} = -c\frac{\left(q_{j+1}^n - q_{j-1}^n\right)}{2\Delta x}$$

  - This method is a *forward time, centered-space* approximation to the one-way wave equation for ***q***
- Truncation error analysis gives us the following:

$$\frac{\partial q}{\partial t} + c\frac{\partial q}{\partial x} = -\frac{\Delta t}{2}\frac{\partial^2 q}{\partial t^2} - \frac{c\Delta x^2}{3!}\frac{\partial^3 q}{\partial x^3} + \ldots$$

Computer Program 2

22

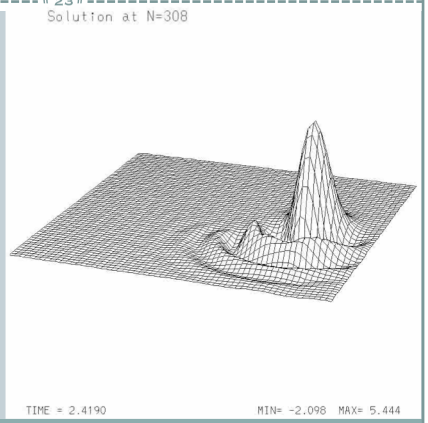
TWO-DIMENSIONAL ADVECTION

VELOCITY VARIES IN SPACE, NOT IN TIME
A LINEAR ADVECTION PROBLEM.

Program 2 - animation

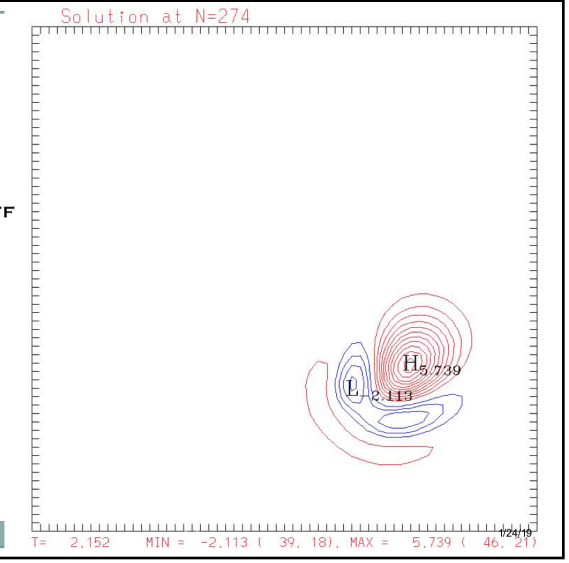
23

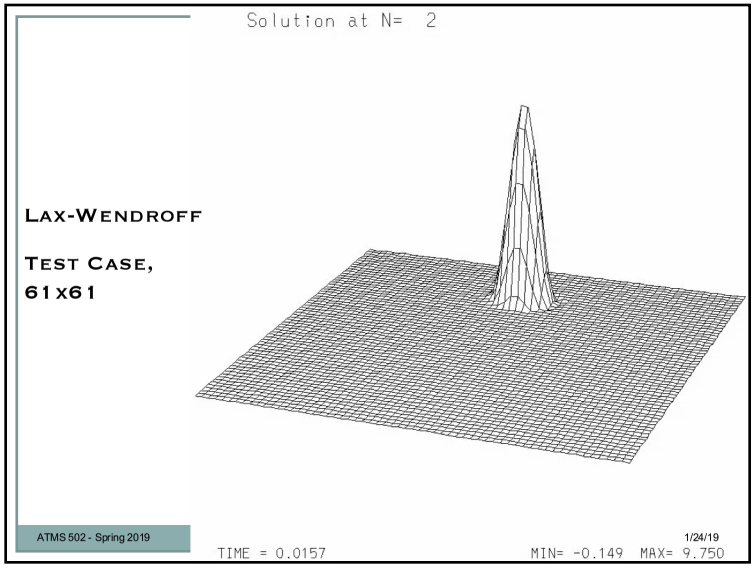
- Animation:
  - field: "s" (now 2-D)
  - shown: surface.
  - view: ~towards +Y
- Note
  - initial circular feature (a "cone" in 3-D) develops waves ...
  - waves *follow* the cone
  - develops <0 values.

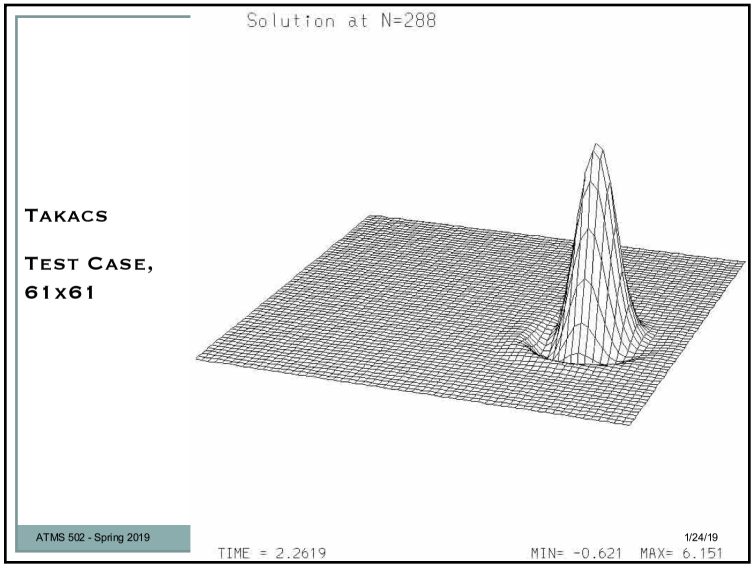


LAX-WENDROFF

TEST CASE,
61x61



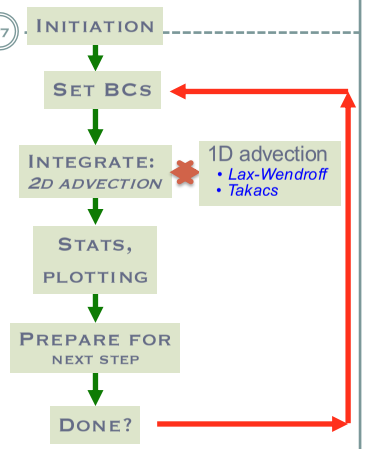




Code structure, program 2

27

- Program 2 structure
  - Is similar to program 1
  - But we step up to 2-D
    - 1-D method used in *X* and *Y*
  - We introduce non-constant flow – varies in (x,y), not time
    - Two flow components: u, v
  - Use two numerical methods
    - Two *1-D* methods! Lax is one
  - Derive stats for final solution
    - Equations from *Takacs* paper



ATMS 502 - Spring 2019 1/24/19

Program 2 - recommendations

28

- Coding programs is like test taking …
  - There are distinct advantages to having a plan.
- 1) Make the necessary arrays 2D.
- 2) Code & evaluate the *initial conditions (ICs)*.
  - Create scalar field + U, V velocity components
  - Be careful with dimensions & physical locations
  - Plot it - plotting+code examples online *~tg457444/502/Pgm2*
- 3) Set the boundary conditions *(BCs)*
  - Alter your BC routine for two dimensions.
- 4) Now continue with 2D advection.

ATMS 502 - Spring 2019 1/24/19

Program 2 - *advect1d()*

29

- **Advect1d** routine: *does transport*
  - *Averages* velocity to s1d() location. *It* expects -
    - 1-D velocity array with nx+1 elements, s1d array w/ghost points

Program 2 - *advection()*

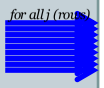
30

- **Advection** routine: *data management*
  - I set up three 1-D arrays in my advection routine –
    - s1d(0:nx+1), u1d(nx+1), v1d(ny+1)
  - Advection routine is now data management; calls *advect1d*
- **Advect1d** routine: *does transport*
  - *Averages* velocity to s1d() location. *It* expects -
    - 1-D velocity array with nx+1 elements, s1d array w/ghost points

Program 2 - details

31

- **Advection** routine: *data management*
  - I set up three 1-D arrays in my advection routine –
    - s1d(0:nx+1), u1d(nx+1), v1d(ny+1)
  - Advection routine is now data management; calls *advect1d*
  - When copying 1-D rows/columns of *S*, copy ghost points, too!
  - When advecting rows (X) …
    - copy s1(i,j) > s1d, U(i,j) > u1d; pass s1d, u1d to *advect1d*.

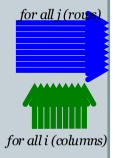


- **Advect1d** routine: *does transport*
  - *Averages* velocity to s1d() location. *It* expects -
    - 1-D velocity array with nx+1 elements, s1d array w/ghost points

Program 2 - details

32

- **Advection** routine: *data management*
  - I set up three 1-D arrays in my advection routine –
    - s1d(0:nx+1), u1d(nx+1), v1d(ny+1)
  - Advection routine is now data management; calls *advect1d*
  - When copying 1-D rows/columns of *S*, copy ghost points, too!
  - When advecting rows (X) …
    - copy s1(i,j) > s1d, U(i,j) > u1d; pass s1d, u1d to *advect1d*.
  - When advecting columns (Y) …
    - copy s1(i,j) > s1d, V(i,j) > v1d; pass s1d, v1d to *advect1d*.



- **Advect1d** routine: *does transport*
  - *Averages* velocity to s1d() location. *It* expects -
    - 1-D velocity array with nx+1 elements, s1d array w/ghost points