Burgers' Equation

Michael Schmid

May 23, 2020

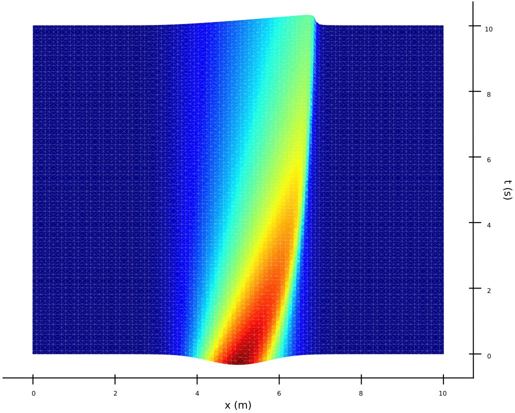
Nonlinear Convection aka. Burgers' Equation

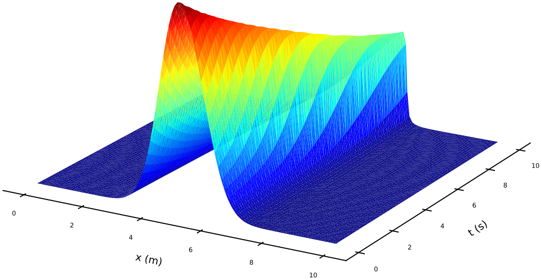
- Viscous Burgers' equation
  - $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$
- Inviscid Burgers' equation
  - $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$
- Appears in
  - Fluid dynamics
  - Nonlinear acoustics
  - Traffic flow

Nonlinear Convection aka. Burgers' Equation

- Viscous Burgers' equation
  - $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$
- Inviscid Burgers' equation
  - $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$
- Appears in
  - Fluid dynamics
  - Nonlinear acoustics
  - Traffic flow

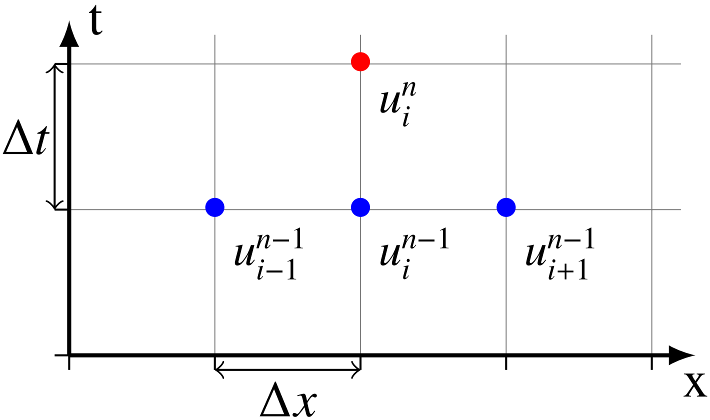
Nonlinear Convection aka. Burgers' Equation





Mathematical Notation

- Discretization
  - u_i^n is the point to be calculated
  - $i-1$ is a step back in space
  - $n-1$ is a step back in time
  - Δt is the step size in time
  - Δx is the step size in space



Explicit Euler Method

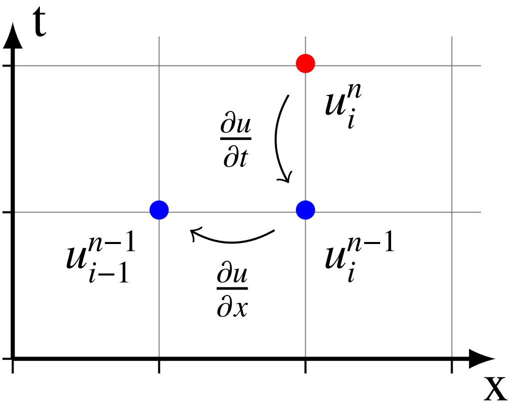
How to rewrtie our PDE?

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

Euler forwards

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \frac{u_i^{n-1} - u_{i-1}^{n-1}}{\Delta x} = 0$$

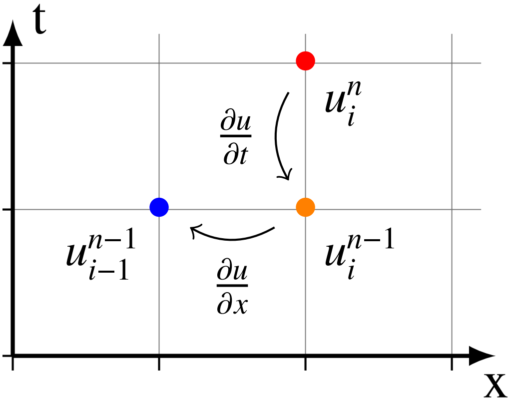
$$u_i^n = \frac{\Delta x \, u_i^{n-1}}{-\Delta t \, u_{i-1}^{n-1} + \Delta t \, u_i^{n-1} + \Delta x}$$



Euler forwards

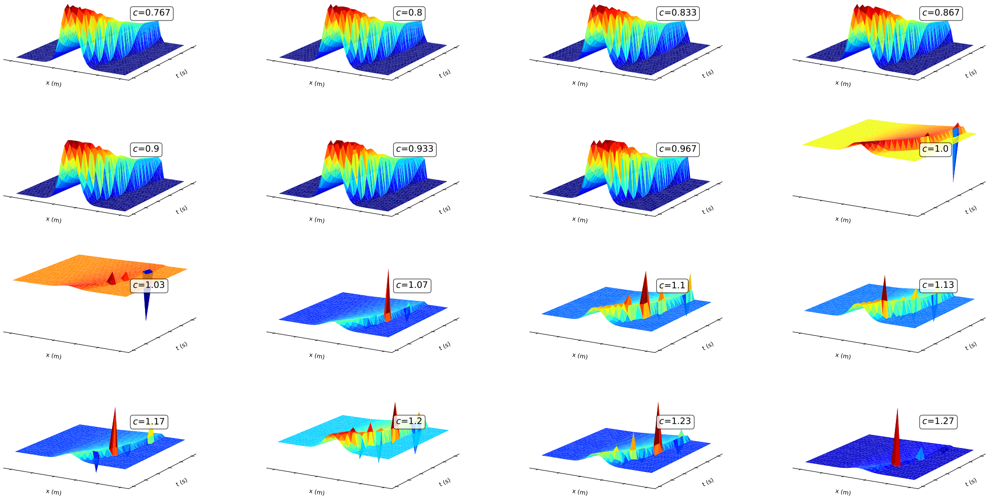
$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1} \frac{u_i^{n-1} - u_{i-1}^{n-1}}{\Delta x} = 0$$

$$u_i^n = \frac{u_i^{n-1} \left(\Delta t \, u_{i-1}^{n-1} - \Delta t \, u_i^{n-1} + \Delta x\right)}{\Delta x}$$



convergence condition by Courant–Friedrichs–Lewy

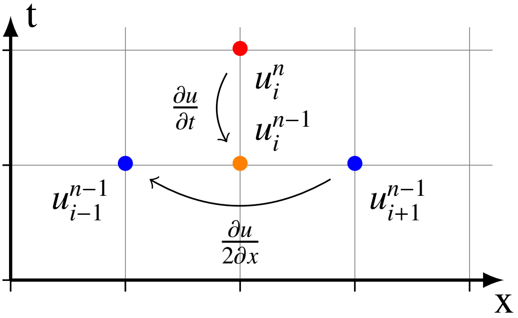
$$c = \frac{u\,\Delta t}{\Delta x} < 1$$


c=0.767
c=0.8
c=0.833
c=0.867
c=0.9
c=0.933
c=0.967
c=1.0
c=1.03
c=1.07
c=1.1
c=1.13
c=1.17
c=1.2
c=1.23
c=1.27
x (m)
t (s)

Euler forwards

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1}\,\frac{u_{i+1}^{n-1} - u_{i-1}^{n-1}}{2\,\Delta x} = 0$$

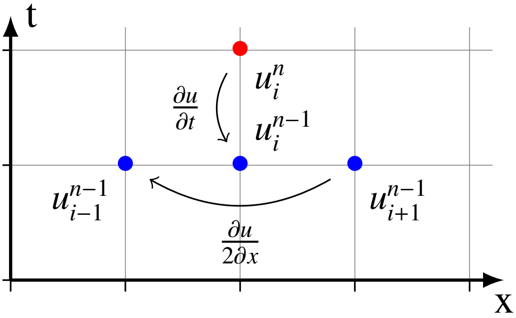
$$u_i^n = \frac{u_i^{n-1}\left(-\Delta t\, u_{i+1}^{n-1} + \Delta t\, u_{i-1}^{n-1} + 2\Delta x\right)}{2\Delta x}$$

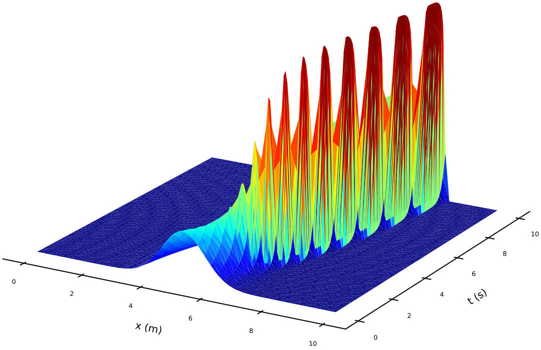


Euler forwards

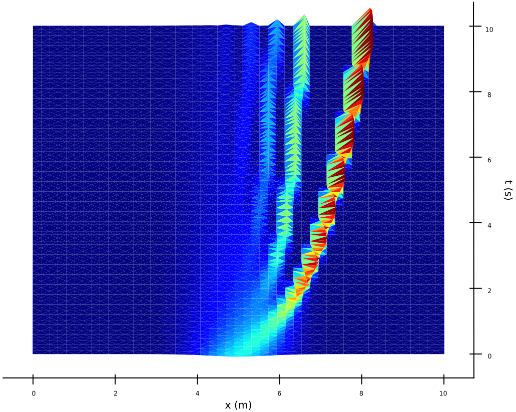
$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \, \frac{u_{i+1}^{n-1} - u_{i-1}^{n-1}}{2\,\Delta x} = 0$$

$$u_i^n = \frac{2\Delta x \, u_i^{n-1}}{\Delta t \, u_{i+1}^{n-1} - \Delta t \, u_{i-1}^{n-1} + 2\Delta x}$$





0
2
4
6
8
10
x (m)
0
2
4
6
8
10
t (s)

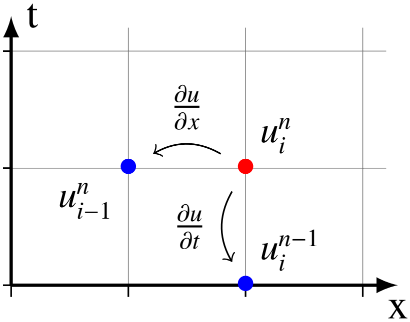


0
2
4
6
8
10
x (m)
0
2
4
6
8
10
t (s)

Euler backwards

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \frac{u_i^n - u_{i-1}^n}{\Delta x} = 0$$

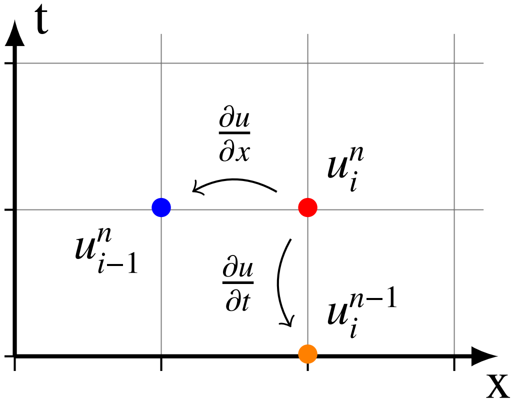
$$u_i^n = \begin{bmatrix} \dfrac{\Delta t\, u_{i-1}^n - \Delta x - \sqrt{\Delta t^2 \left(u_{i-1}^n\right)^2 + 4\Delta t\, \Delta x\, u_i^{n-1} - 2\Delta t\, \Delta x\, u_{i-1}^n + \Delta x^2}}{2\Delta t} \\[2ex] \dfrac{\Delta t\, u_{i-1}^n - \Delta x + \sqrt{\Delta t^2 \left(u_{i-1}^n\right)^2 + 4\Delta t\, \Delta x\, u_i^{n-1} - 2\Delta t\, \Delta x\, u_{i-1}^n + \Delta x^2}}{2\Delta t} \end{bmatrix}$$



Euler backwards

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1}\,\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0$$

$$u_i^n = \frac{u_i^{n-1}\,\left(\Delta t\, u_{i-1}^n + \Delta x\right)}{\Delta t\, u_i^{n-1} + \Delta x}$$

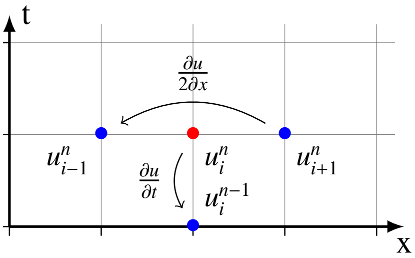


Implicit Euler method

Euler backwards

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1}\,\frac{u_{i+1}^n - u_{i-1}^n}{2\,\Delta x} = 0$$

$$-u_i^{n-1}\,dt\,u_{i-1}^n + 2\,dx\,u_i^n + u_i^{n-1}\,dt\,u_{i+1}^n = 2\,dx\,u_i^{n-1}$$



Implicit Euler method

$$-u_i^{n-1}\,dt\,u_{i-1}^n + 2\,dx\,u_i^n + u_i^{n-1}\,dt\,u_{i+1}^n = 2\,dx\,u_i^{n-1}$$

$$\begin{bmatrix} dx - u_1^n\,dt & u_1^n\,dt & 0 & & 0 \\ -u_2^n\,dt & 2\,dx & u_2^n\,dt & & 0 \\ 0 & -u_3^n\,dt & 2\,dx & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & u_{M-1}^n\,dt \\ 0 & 0 & & -u_M^n\,dt & dx + u_1^M\,dt \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_M^{n+1} \end{bmatrix} = \begin{bmatrix} dx\,u_1^n \\ 2\,dx\,u_2^n \\ 2\,dx\,u_3^n \\ \vdots \\ dx\,u_M^n \end{bmatrix}$$

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

Algorithm 1 Tridiagonal matrix algorithm (Thomas algorithm)

function THOMAS($\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$) ▹ Vectors

  $\hat{c}_1 \leftarrow \dfrac{c_1}{b_1}$

  $\hat{d}_1 \leftarrow \dfrac{d_1}{b_1}$

  for $i = 2, 3, \ldots, n-1$ **do** ▹ Forward sweep

    $\hat{c}_i \leftarrow \dfrac{c_i}{b_i - a_i \hat{c}_{i-1}}$

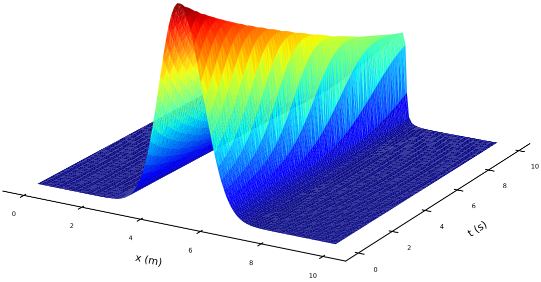
    $\hat{d}_i \leftarrow \dfrac{d_i - a_i \hat{d}_{i-1}}{b_i - a_i \hat{c}_{i-1}}$

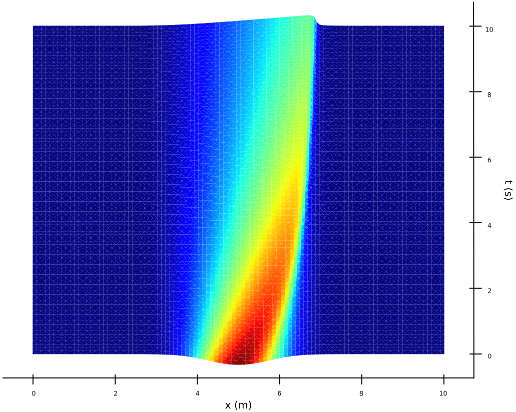
  $x_n \leftarrow \hat{d}_n$

  for $i = n-1, n-2, \ldots, 1$ **do** ▹ Backwards substitution

    $x_i \leftarrow \hat{d}_i - \hat{c}_i \, x_{i+1}$

  return $\mathbf{x}$


0
2
4
6
8
10
x (m)
0
2
4
6
8
10
t (s)


0
2
4
6
8
10
x (m)
0
2
4
6
8
10
t (s)

Convergence Condition

Stable!

References

https://github.com/barbagroup/CFDPython