现代数学概论【科学计算】

Lecture 5 - Numerical Methods for PDEs

Xian-Liang Hu

School of Mathematical Sciences, Zhejiang University, CHINA.

https://courses.zju.edu.cn/course/30542/content#/

Syllubus: Numerical Methods for PDEs

13. Finite Difference Methods(FDMs) for Parabolic PDEs

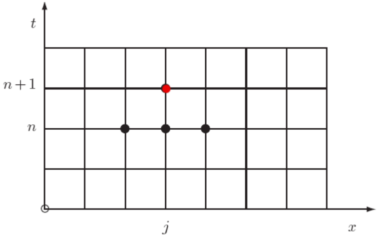
14. Finite Volumn Methods(FVMs) for Hyperbolic PDEs

15. Numerical Methods for Classical (Steady State) PDEs

Finite Difference for 1D Problem

To find a funciton $u(x,t)$, such that

$$\begin{cases} \dfrac{\partial u}{\partial t} = \nu \dfrac{\partial^2 u}{\partial x^2}, & x \in (0,1), t > 0 \\ u(x,0) = f(x), & x \in [0,1] \\ u(0,t) = a(t), u(1,t) = b(t), & t \geq 0 \end{cases}$$



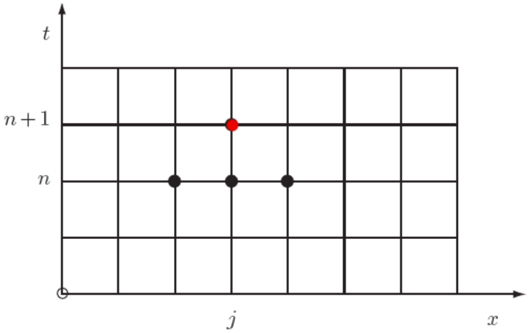
where $f(0) = a(0)$ and $f(1) = b(1)$.

- Discretize the interval $[0,1]$ with $x_j = jh$ and the time step with $t_n = n\tau$

$$\frac{\partial u}{\partial t}(x_j,t_n) \approx \frac{u(x_j,t_{n+1}) - u(x_j,t_n)}{\tau} := \frac{\mathrm{U}_j^{n+1} - \mathrm{U}_j^n}{\tau}$$

$$\frac{\partial^2 u}{\partial x^2}(x_j,t_n) \approx \frac{u(x_{j+1},t_n) - 2u(x_j,t_n) + u(x_{j-1},t_n)}{(h)^2} := \frac{\mathrm{U}_{j+1}^n - 2\mathrm{U}_j^n + \mathrm{U}_{j-1}^n}{(\mathrm{h})^2}$$

Explicit Scheme for 1D parabolic equation



At (x_j, t_n), the 1D parabolic equation yields $\left(\mu = \nu\tau/(h)^2\right)$

$$U_j^{n+1} = U_j^n + \mu(U_{j+1}^n - 2U_j^n + U_{j-1}^n), \tag{1}$$

It is preferred as an **Explicit Scheme**.

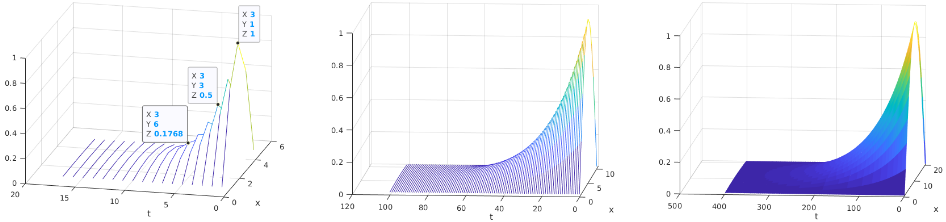
Algorithm & Implementation

Initialize $\nu, f, [a,b]$ and N, T, τ;
Calculate $h = (b-a)/N$;
Let $x_j = j * h, \forall j = 0, 1, \cdots, N$;
u = zeros(N+1,T+1) ;
for $n = 1, 2, \cdots, T$ **do**
  $u(0,n) = a(n\tau); u(N,n) = b(n\tau)$;
  for $j = 1, 2, \cdots, N-1$ **do**
    $U(j, n+1) = \mu U(j+1, n) + (1-2\mu)U(j,n) + \mu U(j-1, n)$;
  end
end

```matlab
N = 21;  % Solve U_t = \nu U_{xx}
a = 0; b = 1;   nu = 1.0;   T = 0.5;
h = (b-a)/(N-1); x = linspace(a,b,N);
tau  = 0.5*h*h/nu;   mu = nu*tau/h/h;
NT = ceil(T/tau); uh = zeros(N,NT+1);
uh(:,1)  = sin(pi*x);  % u_0 = sin(\pi x);
for  n = 1:NT
  for  j  = 2:N-1
     uh(j,n+1) = (1-2*mu)*uh(j,n) + ...
                 mu*(uh(j-1,n) + uh(j+1,n));
  end
end
waterfall(uh');  xlabel('x');  ylabel('t');
```

Example (Analytic Solution $u(x,t) = \exp(-\pi^2 t)\sin(\pi x)$)

- physical parameters: $\nu = 1, u(x,0) = \sin(\pi x), u(0,t) = 0, u(1,t) = 0$
- computational parameters: $T = 1, N = 5, 11, 21, \cdots$
- solution plotted with "waterfall(uh)":



Consistency(相容性):Does it do the right thing?

Let $L = \frac{\partial}{\partial t} - \nu \frac{\partial^2}{\partial x^2}, (\nu > 0)$ be the operator and $U_j^{n+1} = L_h U_j^n$ be the finite difference scheme, where L_h dependent on τ and h. It is defined that the finite difference scheme is consistent with the original differential equation, if

$$T(x_j, t_n) = \left(L_h u(x_j, t_n) - u(x_j, t_{n+1})\right) \to 0, \qquad \tau, h \to 0.$$

Its truncation Error(截断误差) is

$$\begin{aligned} e(x,t) &= \frac{u(x, t+\tau) - u(x,t)}{\tau} - \nu \frac{(u(x+h,t) - 2u(x,t) + u(x-h,t))}{h^2} \\ &= (u_t(x,t) + \frac{\tau}{2} u_{tt}(x,t) + \ldots) - \nu(u_{xx} + \frac{h^2}{12} u_{xxxx} + \ldots) \\ &\approx \frac{\tau}{2} u_{tt}(x,t) - \frac{\nu h^2}{12} u_{xxxx} \end{aligned}$$

Convergence(收敛性): Is $U_j^n \to u(x_j, t_n)$?

Using fixed initial and boundary values and $\mu = \tau/(h)^2$, and let $\tau \to 0, h \to 0$. If on any given position $(x^*, t^*) \in (0,1) \times (0, T)$,

$$U_j^n \to u(x_j, t_n), \forall x_j \to x^*, t_n \to t^*.$$

- **Approximation Error:** $e_j = U_j^n - u(x_j, t_n)$
- Finite difference scheme - $T(x,t)$ yields

$$e_{j+1} = (1 - 2\mu)e_j^n + \mu e_{j+1}^n + \mu e_{j-1}^n - T_j^n \tau,$$

  which yield $E^n \leq \frac{1}{2}\tau\left(M_{tt} + \frac{1}{6\mu}M_{xxxx}\right)$ if $E^n = \max\{|e_j|, j = 0, 1, \cdots, n\}$ and M_{tt} and M_{xxxx} be the upper limit for u_{tt} and u_{xxxx} respectively.
- The explicit scheme (1) be convergent if $\mu := \frac{\tau}{h^2} \leq \frac{1}{2}$.

Error Analysis via Fourier Mode

$$U_j^n = (\lambda)^n e^{ik(jh)}$$

as the solution of the finite differentce scheme (1) , it yields

$$\begin{aligned}\lambda &:= \lambda(k) = 1 + \mu(e^{ikh} - 2 + e^{-ikh}) \\ &= 1 - 2\mu(1 - cos(kh)) \\ &= 1 - 4\mu sin^2 \frac{1}{2} kh\end{aligned}$$

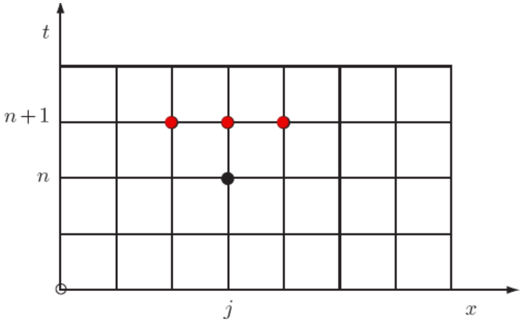
- Since $U_j^{n+1} = \lambda U_j^n$, λ is referred as **amplification factor**
- At frequency $k = m\pi$处, $\mu > \frac{1}{2}$ makes $\lambda > 1$: divergent
- **stable**: there exist a K independent of k, which makes

$$|[\lambda(k)]^n| \leq K, \qquad \forall k, n\tau \leq T$$

Implicit schemes(隐格式)

The stability condition $\mu = \frac{\tau}{h^2} \leq \frac{1}{2}$ is too strong: it yields too small time-step length $\tau \leq \frac{1}{2}h^2$ if the grid space $h \to 0$. Let us consider another scheme,

$$U_j^{n+1} = U_j^n + \mu(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}) \tag{2}$$



The implicit scheme yields

$$-\mu U_{j-1}^{n+1} + (1+2\mu)U_j^{n+1} - \mu U_{j+1}^{n+1} = U_j^n, \qquad \forall j = 1, 2, \ldots, (N-1).$$

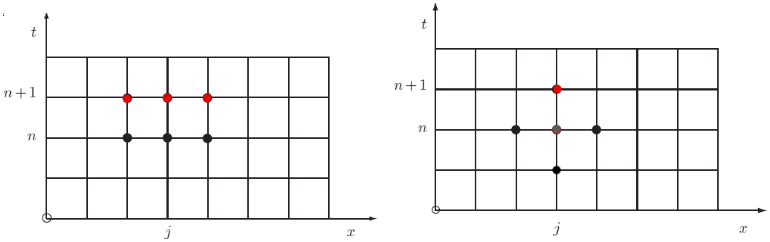
U_0^{n+1} and U_N^{n+1} are known with the boundary condition.

- **Thomas algorithm** is most efficient for tri-diagonal system
- using Fourier mode $U_j^n = (\lambda)^n e^{ik(jh)}$ yields

$$\lambda = \frac{1}{1 + 4\mu \sin^2 \frac{1}{2}kh} < 1,$$

  which says the implicit scheme is **unconditionally stable**
- However, the truncation error is same with the explicit one.

Classical Implicit Schemes



- Crand-Nickson(Left, $\lambda < -1$): $\mu(1-2\theta) > \frac{1}{2}$
- Leap Frog(Right): $\lambda^2 + 8\lambda\mu\sin^2\frac{1}{2}kh - 1 = 0$

Example (Please write it into Homework 5)

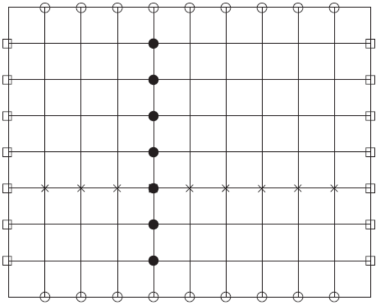
- To implement any implicit scheme, compared with the explicit one in (1).

Multivariate Problem

Find a function $u(x, y, t)$, such that

$$u_t(x, y, t) = b(u_{xx}(x, y, t) + u_{yy}(x, y, t)), \qquad (b > 0).$$

with proper initial value $u(x, y, 0)$ and Dirichlet boundary condition on $\Omega = [0, X] \times [0, Y]$, which is discretized with equal-space grids:



Explicit V.S. Implicit

Take time step $\triangle t$, grid space $\triangle x$ and $\triangle y$

$$U^n_{r,s} \approx u(x_r, y_s, t_n), \qquad \forall r = 0, \cdots, Nx, s = 0, \cdots, Ny.$$

- Explicit scheme

$$\frac{U^{n+1}_{r,s} - U^n_{r,s}}{\triangle t} = b\left[\frac{U^n_{r+1,s} - 2U^n_{r,s} + U^n_{r-1,s}}{(\triangle x)^2} - \frac{U^n_{r,s+1} - 2U^n_{r,s} + U^n_{r,s-1}}{(\triangle y)^2}\right]$$

- Implicit scheme(Relax with **Jacobi** and **Gauss Sediel** solver)

$$\frac{U^{n+1}_{r,s} - U^n_{r,s}}{\triangle t} = b\left[\frac{U^{n+1}_{r+1,s} - 2U^{n+1}_{r,s} + U^{n+1}_{r-1,s}}{(\triangle x)^2} - \frac{U^{n+1}_{r,s+1} - 2U^{n+1}_{r,s} + U^{n+1}_{r,s-1}}{(\triangle y)^2}\right]$$

Alternative Direction Interaction(ADI) - 交替方向(隐)

Two dimensional Crank-Nicolson scheme

$$(1 - \frac{1}{2}\mu_x\delta_x^2 - \frac{1}{2}\mu_y\delta_y^2)U^{n+1} = (1 + \frac{1}{2}\mu_x\delta_x^2 + \frac{1}{2}\mu_y\delta_y^2)U^n$$

with a slight modification

$$(1 - \frac{1}{2}\mu_x\delta_x^2)(1 - \frac{1}{2}\mu_y\delta_y^2)U^{n+1} = (1 + \frac{1}{2}\mu_x\delta_x^2)(1 + \frac{1}{2}\mu_y\delta_y^2)U^n$$

At last, split into two steps as

$$\begin{aligned}
(1 - \frac{1}{2}\mu_x\delta_x^2)U^{n+\frac{1}{2}} &= (1 + \frac{1}{2}\mu_y\delta_y^2)U^n \\
(1 - \frac{1}{2}\mu_y\delta_y^2)U^{n+1} &= (1 + \frac{1}{2}\mu_x\delta_x^2)U^{n+\frac{1}{2}}
\end{aligned}$$

Extension: Operator Splitting

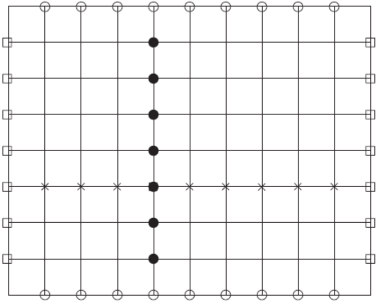
Concept of ADI

Step 1:

$$(1 - \frac{1}{2}\mu_x\delta_x^2)U^{n+\frac{1}{2}} = (1 + \frac{1}{2}\mu_y\delta_y^2)U^n$$

Step 2:

$$(1 - \frac{1}{2}\mu_y\delta_y^2)U^{n+1} = (1 + \frac{1}{2}\mu_x\delta_x^2)U^{n+\frac{1}{2}}$$



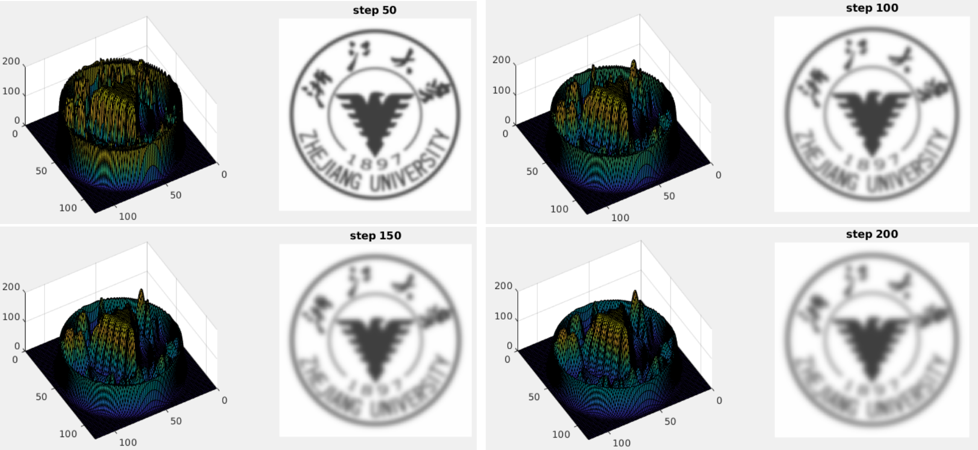
Reference:

- Peaceman D.W. and Rachford H.H. Jr. The numerical solution of parabolic and elliptic differential equations. J. Soc. Indust. Appl. Math. 3, 28-41. 1955.

Application of 2D heat equation: Image Denoising

```matlab
I = rgb2gray(imread('zju600.jpeg'));
[m,n] = size(I);   uh = double(I);
nu = 5.0;   dt = 1.0/(2*nu);
for k = 1:150
    for i = 2:m-1
        for j = 2:n-1
            uh(i,j) = dt*((uh(i+1,j) - 2*uh(i,j) + uh(i-1,j)) ...
                + (uh(i,j+1) - 2*uh(i,j) + uh(i,j-1))) + uh(i,j);
        end
    end
    subplot(1,2,1); surf(256-uh(1:4:end, 1:4:end)); view(150,60);
    subplot(1,2,2); I = uint8(uh); imshow(I);
    title(['step ',num2str(k)]); pause(0.01);
end
```

Smooth Effects at Different Time Steps(50, 100, 150, 200)



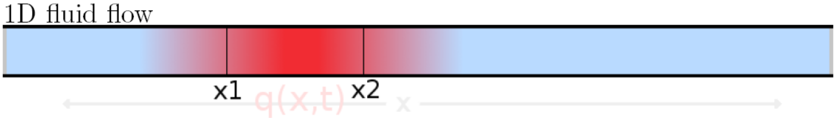
13. Finite Difference Methods(FDMs) for Parabolic PDEs

14. Finite Volumn Methods(FVMs) for Hyperbolic PDEs

15. Numerical Methods for Classical (Steady State) PDEs

Convection Problem

Consider 1D fluid flow,



$$\int_{x_1}^{x_2} q(x,t)dx = \text{mass of tracer between } x_1 \text{ and } x_2.$$

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = F_1(t) - F_2(t),$$

where F_i is the flux of mass from right to left at x_i.

Conservative(Integral) Formulation

For general autonomous flux $F = f(q)$, one can have

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = f\big(q(x_1,t)\big) - f\big(q(x_2,t)\big).$$

If f is sufficiently smooth, apply the Newton-Leibniz formula to RHS:

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = -\int_{x_1}^{x_2} \frac{\partial}{\partial x} f\big(q(x,t)\big)dx,$$

which leads to

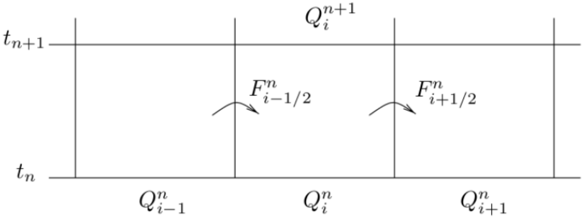
$$\int_{x_1}^{x_2}\left[\frac{\partial}{\partial t}q(x,t) + \frac{\partial}{\partial x}f\big(q(x,t)\big)\right]dx = 0. \tag{3}$$

Finite "Volume"

Denote cells $C_i = \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$ and mean values on cells

$$Q_i^n \approx \frac{1}{|C_i|}\int_{C_i} q(x, t_n)dx.$$

FVM update Q_i^{n+1} based on the fluxes F^n between the cells



Finite Volume Scheme

Considering the integral form/Conservation Law

$$\frac{d}{dt}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} q(x,t)dx = f(q(x_{i-\frac{1}{2}},t)) - f(q(x_{i+\frac{1}{2}},t)).$$

By integrating on $[t_n, t_{n+1}]$ and divided by $\triangle x$, it yields

$$\begin{aligned}\frac{1}{\triangle x}\int_{C_i} q(x,t_{n+1})dx &= \frac{1}{\triangle x}\int_{C_i} q(x,t_n)dx \\ &- \frac{1}{\triangle x}\left[\int_{t_n}^{t_{n+1}} f\big(q(x_{i+\frac{1}{2}},t)\big)dt - \int_{t_n}^{t_{n+1}} f\big(q(x_{i-\frac{1}{2}},t)\big)dt\right].\end{aligned}$$

Utilizing the definition of Q and F, it could be written as

$$Q_i^{n+1} = Q_i^n - \frac{\triangle t}{\triangle x}\left(F^n_{i+\frac{1}{2}} - F^n_{i-\frac{1}{2}}\right), \tag{4}$$

where $F_{i-\frac{1}{2}} \approx \frac{1}{\triangle t}\int_{t_n}^{t_{n+1}} f(q(x_{i-1/2},t))dt$ is the so-called "Flux".

Numerical flux(数值流通量)

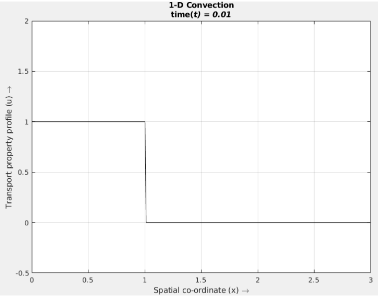
For a hyperbolic problem, information propagates at a finite speed. So it is reasonable to assume that we can obtain $F^n_{i-1/2}$ using only the values Q^n_{i-1} and Q^n_i:

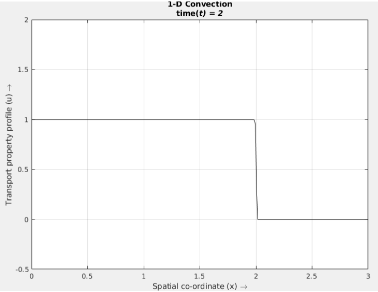
$$F^n_{i-1/2} = \mathcal{F}(Q^n_{i-1}, Q^n_i)$$

where $\mathcal{F}$ is some *numerical flux function*. Then our numerical method becomes

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{\Delta x}\left[\mathcal{F}(Q^n_i, Q^n_{i+1}) - \mathcal{F}(Q^n_{i-1}, Q^n_i)\right].$$

Example (Convection Equation(convection_fvm.m))





Try finite element schemes(MacCormack(defult, explicit, second order)):

1. Beam-Warming with artificial viscosity(implicit, second order)
2. Lax-Friedrichs(explicit, first order)
3. Lax-Wendroff(explicit, second order)

Convergence(FVM的收敛性)

We say that the numerical solution for a hyperbolic equation is convergent in the meaning of $\triangle x \to 0$ and $\triangle t \to 0$, it requires

- The method be *consistent*, which promises the local truncation error goes to 0 as $\triangle t \to 0$
- The method be *stable*, which means any small error in each timestep is under control(will not grow too fast)

Consistency(相容性)

Denote the numerical method as $A^{n+1} = \mathcal{N}(Q^n)$ and the exact value as q^n and q^{n+1}. Then the local truncation error is defined as

$$\tau = \frac{\mathcal{N}(q^n) - q^{n+1}}{\triangle t}$$

We say that the method is *consistent* if τ vanished as $\triangle t \to 0$ for all smooth $q(x,t)$ satisfying the differential equation. It is usually stratightforward when Taylor expasions are used.

Stability(稳定性)

Courant-Friedrichs-Levy condition: the numerical domain of dependence contains the true domain of dependence domain of the PDE, at least in the limit as $\triangle t, \triangle x \to 0$

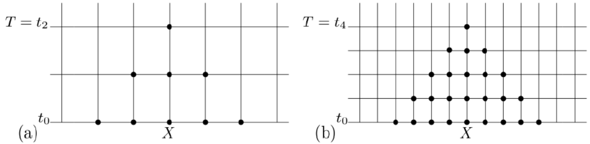


Fig. 4.3. (a) Numerical domain of dependence of a grid point when using a three-point explicit finite difference method, with mesh spacing Δx^a. (b) On a finer grid with mesh spacing $\Delta x^b = \frac{1}{2}\Delta x^a$.

For a hyperbolic system with characterestic wave speeds λ^p,

$$\frac{\triangle x}{\triangle t} \geq \max_p |\lambda^p|, \qquad p = 1, \cdots, m.$$

- necessary but not sufficient !

Flux(通量) Function

To do the calculation,

$$Q_i^{n+1} = Q_i^n - \frac{\triangle t}{\triangle x}\left(F_{i+\frac{1}{2}}^n - F_{i-\frac{1}{2}}^n\right),$$

the key step is to compute the numerical flux term

- unstable: $\mathcal{F}(Q_{i-1}^n, Q_{i+1}^n) = \frac{1}{2}\left[f(Q_{i-1}^n) + f(Q_i^n)\right]$
- stable: looking into the direction from which the flow come from(upwind), for e.g. $q_t + \lambda q_x = 0$ with $\lambda > 0$, yields

$$Q_i^{n+1} = Q_i^n - \lambda\frac{\triangle t}{\triangle x}\left(Q_i^n - Q_{i-1}^n\right) \tag{5}$$

Roe Scheme

Recall the numerical method for Conservation Law

$$Q_i^{n+1} = Q_i^n - \frac{\triangle t}{\triangle x}\Big(\mathcal{F}(Q_i^n, Q_{i+1}^n) - \mathcal{F}(Q_{i-1}^n, Q_i^n)\Big),$$

A linearized choice of the numerical flux based on the Godunov's method for the nonlinear problems. Define $|A| = R|\Sigma|R^{-1}$, where $|\Sigma| = diag(|\lambda^p|)$, then we can derive the Roe's flux as

$$F_{i-\frac{1}{2}}^n = \frac{1}{2}\Big(f(Q_{i-1}) + f(Q_i)\Big) - \frac{1}{2}|A|\Big(Q_{i-1} + Q_i\Big)$$

Remark:In this sense, R is properly chosen, such that A is a good enough approximation to nonlinear functional $\mathcal{F}$.

High Order : Godunov Scheme

The following *REA algorithm* was proposed by Godunov (1959):

1. **Reconstruct** a piecewise polynomial function $\tilde{q}^n(x, t_n)$ from the cell averages Q_i^n. In the simplest case, $\tilde{q}^n(x, t_n)$ is piecewise constant on each grid cell:

$$\tilde{q}^n(x, t_n) = Q_i^n, \qquad \text{for all } x \in C_i.$$

2. **Evolve** the hyperbolic equation with this initial data to obtain $\tilde{q}^n(x, t_{n+1})$.
3. **Average** this function over each grid cell to obtain new cell averages

$$Q_i^{n+1} = \frac{1}{\Delta x} \int_{C_i} \tilde{q}^n(x, t_{n+1})\, dx.$$

Remark: Evolve step (2) requires solving the Riemann problem.

Total Variation Diminishing(TVD) Scheme

Recall the numerical method for Conservation Law

$$Q_i^{n+1} = Q_i^n - \frac{\triangle t}{\triangle x}\left[\mathcal{F}(Q_i^n, Q_{i+1}^n) - \mathcal{F}(Q_i^n, Q_{i+1}^n)\right],$$

where $\mathcal{F}(Q_i^n, Q_{i+1}^n) \approx F_{i+\frac{1}{2}}^n = h(Q_{i+\frac{1}{2}}^-, Q_{i+\frac{1}{2}}^+)$.

TVD: It is required that the numerical flux function $h(\cdot,\cdot)$ is monotone(Lipschitz continuous, monotone, $h(a,a)=a$)

Example (Write it in Homework)

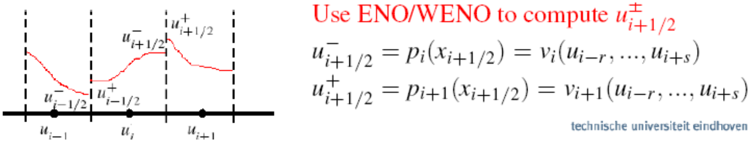
Following convection_fvm.m, please implement the following TVD scheme

$$h(a,b) = 0.5\big(f(a) + f(b) - \alpha(b-a)\big), \quad \text{with } \alpha = \max_u |f'(u)|.$$

Please describe the formulation and show main part of the code.

Weighted Essentially Non-Oscillatory (WENO)

The main concept of (W)ENO is



where $\{u_i\}_{i=0}^n$ are the given **cell average** of a function $q(x)$.

1. Construct polynomials $p_i(x)$ of degree $k-1$, for each cell C_i, such that it is a k-th order accurate approximation to the function $q(x)$, which means

$$p_i(x) = q(x) + \mathcal{O}(\triangle^k) \qquad \forall x \in C_i, i = 0, 1, \cdots, N$$

2. Evaluate u at each cell interface($u^-_{i+1/2}$ and $u^+_{i+1/2}$)

High Order Method - Spectral methods (谱方法)

Lloyd N. Trefethen:

Spectral methods are one of the "big three" technologies for the numerical solution of PDEs, which came into their own roughly in successive decades:

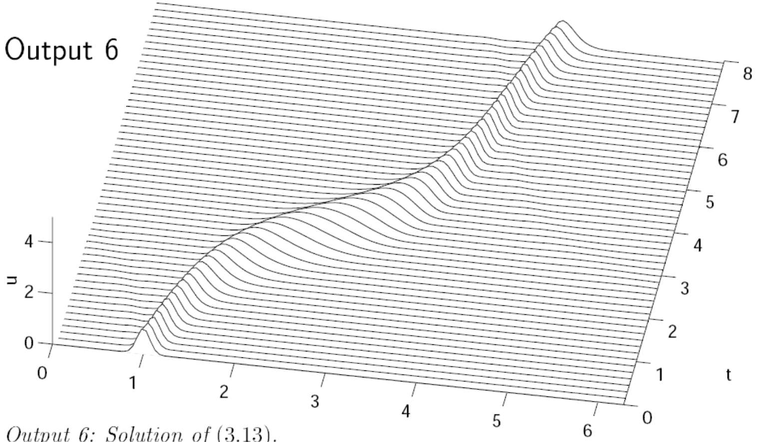
- 1950s: Finite Difference Methods
- 1960s: Finite Element Methods
- 1970s: Spectral Methods

Spectral Solver: $u_t + c(x)u_x = 0$


```
% p6.m - variable coefficient wave equation

% Grid, variable coefficient, and initial data:
  N = 128; h = 2*pi/N; x = h*(1:N); t = 0; dt = h/4;
  c = .2 + sin(x-1).^2;
  v = exp(-100*(x-1).^2); vold = exp(-100*(x-.2*dt-1).^2);

% Time-stepping by leap frog formula:
  tmax = 8; tplot = .15; clf, drawnow
  plotgap = round(tplot/dt); dt = tplot/plotgap;
  nplots = round(tmax/tplot);
  data = [v; zeros(nplots,N)]; tdata = t;
  for i = 1:nplots
    for n = 1:plotgap
      t = t+dt;
      v_hat = fft(v);
      w_hat = 1i*[0:N/2-1 0 -N/2+1:-1] .* v_hat;
      w = real(ifft(w_hat));
      vnew = vold - 2*dt*c.*w; vold = v; v = vnew;
    end
    data(i+1,:) = v; tdata = [tdata; t];
  end
  waterfall(x,tdata,data), view(10,70), colormap([0 0 0])
  axis([0 2*pi 0 tmax 0 5]), ylabel t, zlabel u, grid off
```



Output 6: Solution of (3.13).

The code and figures are from: Trefethen, spectral method in matlab.

Illustration on 2D Conservation LAW(Vector-Valued)

Example (Shallow Water Equation)

$$\frac{\partial}{\partial t}\begin{bmatrix} h \\ hu \\ hv \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} hu \\ hu^2 + gh^2/2 \\ huv \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} hv \\ huv \\ hv^2 + gh^2/2 \end{bmatrix} = 0$$

is known as the shallow water equations in conservative form.

Consider the IV and BV conditions,

$$\begin{aligned} \Omega : (x,y) \in [-1,1]^2, &\quad t \in [0,3] \\ U(x,y,0) = [2,0,0]^\top, &\quad \text{for } (x,y) \in [-1/2,1/2]^2, \\ U(x,y,0) = [1,0,0]^\top, &\quad \text{otherwise.} \end{aligned}$$

Here, $U = [h\ hu\ hv]^\top$.

- MATLAB code: shallow_water_fvm.m and shallow_water_fdm.m.

finite volume method with numerical flux function of Lax-Friedrichs type:

$$V_i^{n+1} = V_i^n - \frac{\Delta t}{\Delta x}\left(F^{*n}_{i+\frac{1}{2}} - F^{*n}_{i-\frac{1}{2}}\right),$$

with

$$F^*_{i+\frac{1}{2}} = \frac{1}{2}[F(V_i) + F(V_{i+1})] - \frac{1}{2}|\lambda_{i+\frac{1}{2}}|_{max}(V_{i+1} - V_i),$$

where $|\lambda_{i+\frac{1}{2}}|_{max}$ is the largest eigenvalue in absolute value of the Jacobian matrix of the hyperbolic system at interface $i + \frac{1}{2}$ (in this case, $|\lambda_x|_{max} = |u| + \sqrt{gh}$, or $|\lambda_y|_{max} = |v| + \sqrt{gh}$). For calculating $|\lambda_{i+\frac{1}{2}}|_{max}$, the averages of u (or v) and h are used.

The time-step is chosen dynamically in every step according to

$$\Delta t = \frac{c}{2}\min\left(\min\left(\frac{\Delta x}{|\lambda_x|_{max}}\right), \min\left(\frac{\Delta y}{|\lambda_y|_{max}}\right)\right).$$

The CFL safety constant c is chosen to be smaller than 1 for this nonlinear system in order to avoid oscillations (for example, $c = 0.8$).

13. Finite Difference Methods(FDMs) for Parabolic PDEs

14. Finite Volumn Methods(FVMs) for Hyperbolic PDEs

15. Numerical Methods for Classical (Steady State) PDEs

Finite Difference for 2D Case

Denote $u_{i,j}$ as the approximation of u at grid point (i,j), then

$$\frac{\partial^2 u}{\partial x^2} = \frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{h_x^2} + O(h_x^2)$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{h_y^2} + O(h_y^2).$$

Let $h_x = h_y = h$, then we get 5-point Stencil:

$$-\triangle u \approx \frac{1}{h^2}\begin{pmatrix} & -u_{i,j+1} & \\ -u_{i-1,j} & 4u_{i,j} & -u_{i+1,j} \\ & -u_{i,j-1} & \end{pmatrix} \tag{6}$$

```
function A = spLaplaceKron(n)
I = speye(n,n);
E = sparse(2:n,1:n-1, 1,n,n);
D = 2*I - (E + E');
A = kron(D,I) + kron(I,D);
return
```

Alternative implementation

```
function [A, idx_bnd, idx_inner] = spLaplacian(a,b,n)
h = abs(b - a)/n;  h2 = 1/(h*h);
nequation = (n+1)*(n+1);  nunknown = (n-1)*(n-1);
%% find out the boundary
idx_bnd = zeros(4*n,1);  idx_bnd(1:n) = 1:n;
idx_bnd(n+1:2*n) = n+1:n+1:n*(n+1);
idx_bnd(2*n+1 : 3*n) = (n+1)*(n+1):-1:n*(n+1)+2;
idx_bnd(3*n+1 : 4*n) = n*(n+1)+1: -(n+1) : n+2;
% and the inner boundary
idx_inner = setdiff((1:(n+1)*(n+1))', idx_bnd);
%% This is for the non-zero entries of the sparse matrix
ii = zeros(nunknown, 5);  jj = ii;  nnz = ii;
ii(:,1) = idx_inner;  jj(:,1) = idx_inner;  nnz(:,1) = 4*h2; % diagnal one
ii(:,2) = idx_inner;  jj(:,2) = idx_inner + 1;  nnz(:,2) = -h2; % east
ii(:,3) = idx_inner;  jj(:,3) = idx_inner - 1;  nnz(:,3) = -h2; % west
ii(:,4) = idx_inner;  jj(:,4) = idx_inner - (n+1);  nnz(:,4) = -h2; % south
ii(:,5) = idx_inner;  jj(:,5) = idx_inner + (n+1);  nnz(:,5) = -h2; % north
A = sparse(ii,jj,nnz, nequation, nequation) + ...
        sparse(idx_bnd, idx_bnd, 1, nequation, nequation);
end
```

Example 1: Poisson Equation based Boundary Value Problem

Let us consider

$$-\triangle u = f, \quad (x,y) \in [0,1]^2, \tag{7}$$

with the analytic solution being

$$u(x,y) = \sin(\pi x)\sin(\pi y), \tag{8}$$

It is straightfoward that $f = 2\pi^2 u(x,y)$.

MATLAB code:

```
a = 0; b = 1;  n = 10; % try 10,20,40,...,1280
func_u = @(X,Y) sin(pi*X).*sin(pi*Y);
func_rhs = @(u) 2*pi*pi*u;
h = abs(b-a)/n;  t = a + (0:n)*h;
neqn = (n+1)*(n+1); u_old = zeros(neqn,1);
[xx,yy] = meshgrid(t);
X = reshape(xx', neqn, 1);
Y = reshape(yy', neqn, 1);
[A, idx_bnd, idx_inner] = spLaplacian(a,b,n);
uu = func_u(X,Y);  b = func_rhs(uu);
b(idx_bnd) = func_u(X(idx_bnd), Y(idx_bnd));
    uh = A\b;   %% Solve Linear System
mesh(xx', yy', reshape(abs(uh-uu), n+1, n+1));
```

Numerical Convergence/Resolution Study

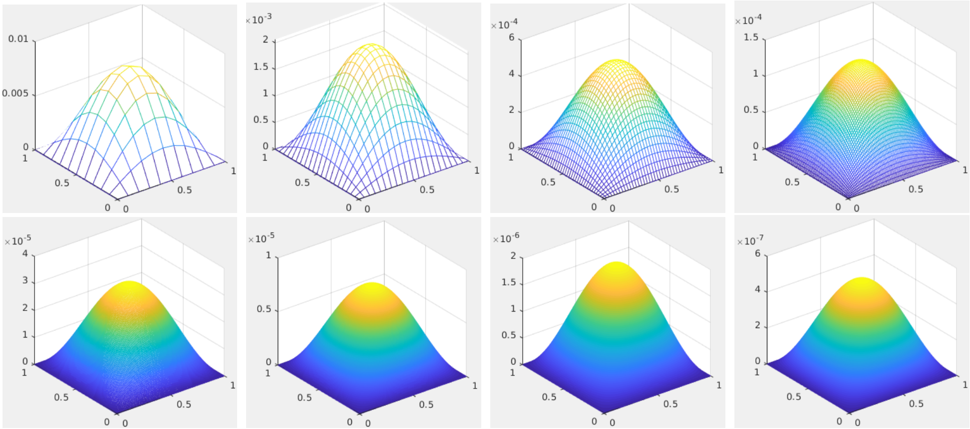


Figure: (Top) $N = 10, 20, 40, 80$; (Bottom) $N = 160, 320, 640, 1280$.

Example 2: Nonlinear Consideration

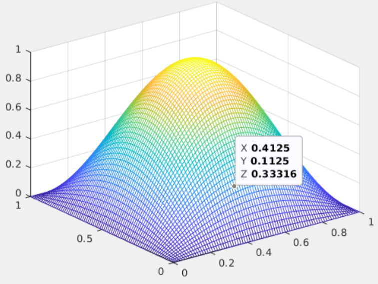
Consider a semi-linear steady state/elliptic equation:

$$-\triangle u + f(\mathrm{x}, u) = g, \quad \mathrm{x} \in \Omega \subset \mathrm{R}^2, \tag{9}$$

with the nonlinear term being

$$f(\mathrm{x}, u) = u^3. \tag{10}$$

The right hand side g followed as soon as u given.



Finite Difference Discretization - Newton's Iterative Scheme

$$\frac{1}{h^2}\begin{pmatrix} & -u_{i,j+1} & \\ -u_{i-1,j} & 4u_{i,j} & -u_{i+1,j} \\ & -u_{i,j-1} & \end{pmatrix} + f(x_i, y_j, u_{i,j}) = g_{i,j},$$

which is numerically denoted as

$$A\mathrm{u} + f(\mathrm{u}) = \mathrm{g}. \tag{11}$$

Let $F(\mathrm{u}) = A\mathrm{u} + f(\mathrm{u}) - \mathrm{g}$, then we have

$$\mathrm{u}^{new} = \mathrm{u}^{old} - F'(\mathrm{u}^{old})^{-1} F(\mathrm{u}^{old}) \tag{12}$$

where $F'(\mathrm{u}) = A + f'(\mathrm{u})$.

MATLAB Code(main_FD5Newton.m)

```matlab
a = 0; b = 1; n = 80;   % try 10, 20, 40, 80, 160, etc. ...
func_u = @(X,Y) sin(pi*X).*sin(pi*Y); % define the Nonlinear PDE
func_rhs = @(u) 2*pi*pi*u + u.*u.*u;
Newton_F  = @(u) u.*u.*u;   Newton_DF = @(u) 3*u.*u;
h = abs(b-a)/n;  t = a + (0:n)*h; [xx,yy] = meshgrid(t); % build mesh
neqn = (n+1)*(n+1);            u_old = zeros(neqn,1);
X = reshape(xx', neqn, 1);    Y = reshape(yy', neqn, 1);
[A, idx_bnd, idx_inner] = spLaplacian(a,b,n); % compute the general matrix
NMaxNewtonIter = 100; TolNewton = 1e-8; iter = 1; error = 100.0;
while iter <= NMaxNewtonIter && error >= TolNewton
    f  = Newton_F(u_old);    df = Newton_DF(u_old);  % Build Newton Step
    Mat = A + sparse(idx_inner, idx_inner, df(idx_inner), neqn,neqn);
    b = A*u_old + f - func_rhs(func_u(X,Y));
    b(idx_bnd) = func_u(X(idx_bnd), Y(idx_bnd));     % Dirichlet boundary
    u_new = u_old - Mat\b;   %% do the newton iteration
    error = norm(u_new - u_old, 2);     %% evaluate the errors
    fprintf('%d: %20.15f\n', iter, error);      %% print out om information
    u_old = u_new; iter = iter + 1;     %% prepare for the next iteration
end
mesh(xx', yy', reshape(u_new, n+1, n+1));  axis tight;
```

Do It Yourself

Example (Written in Homework)

1. 修改Poisson方程算例(Page 42,用不同u)，记录$h = 10, 20, 40, \cdots, 1280$ 等尺度时数值解的L_2或L_∞误差，请列出误差表格数据，并计算收敛阶
2. 运行半线性问题算例，计算Newton迭代法的收敛阶

其他可进行的深度探索(Optional)：

- 尝试将上述方法扩展至x-和y-方向的区间和步长不一致的情形：$h_x \neq h_y$
- 尝试变更方程编号的顺序（列优先或其他感兴趣的顺序）
- 尝试其他可能加速求解线性方程组的方法(如共轭梯度法、多重网格法等)

Incompressible Navier-Stokes equation

Let Ω is a bounded and connected open domain in R^2,

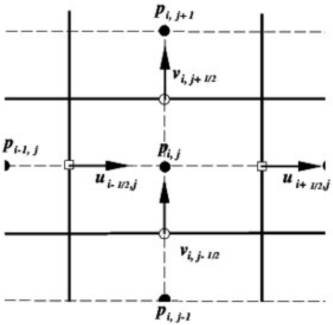
$$\begin{cases} -\nu\Delta \mathrm{u} + (\nabla \mathrm{u})\mathrm{u} + \nabla p = \mathrm{f} & in \quad \Omega, \\ \nabla \cdot \mathrm{u} = \mathrm{g} & in \quad \Omega, \\ \mathrm{u} = 0 & on \quad \partial\Omega, \end{cases} \tag{13}$$

where u is the velocity vector valued function, p is the pressure function.Then the Navier-Stokes equations in integral form reads:

$$\int_S \rho v \cdot n dS = 0$$

$$\int_\Omega \rho u_i d\Omega + \int_S \rho u_i v \cdot n \; dS = \int_S \tau_{ij} i_j \cdot n \; dS - \int_S p i_i \cdot n \; dS + \int_\Omega (\rho - \rho_0) g_i d\Omega$$

Finite Difference Discretization on Staggered Grid



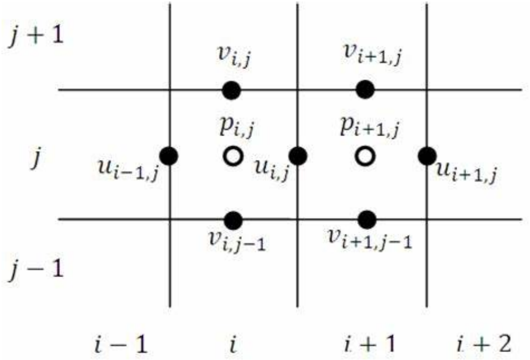
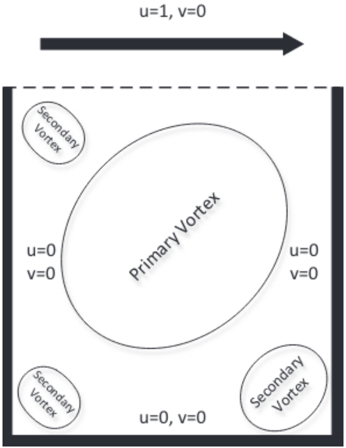
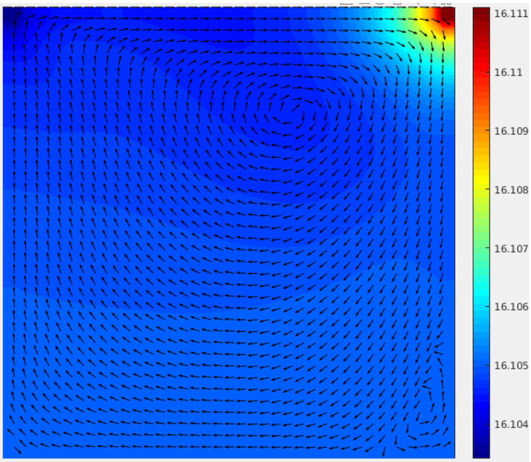


Figure: Staggered grid: illustration on one cell(left) and two cell(right). See, for e.g., `Cavity2017.pdf` for more details。

2D Lid driven Cavity Flow - www.cavityflow.com





- See cavity_stagger_grid.m and cavity_simple.m for a closer look.

12 steps to Navier-Stokes

This module has been proved in the classroom for four consecutive years. It has brought several dozen students to develop their own 2D Navier-Stokes finite-difference solver from scratch in just over a month (with two class meetings per week). The module consists of the following steps (links are to the individual IPython Notebooks):

Steps 1–4 are in one dimension:

(i) linear convection with a step-function initial condition (IC) and appropriate boundary conditions (BC);

with the same IC/BCs:

(ii) nonlinear convection, and

(iii) diffusion only;

with a saw-tooth IC and periodic BCs

(iv) Burgers' equation.

Steps 5–10 are in two dimensions:

(v) linear convection with square function IC and appropriate BCs;

(vi) nonlinear convection, with the same IC/BCs

(vii) diffusion only, with the same IC/BCs;

(viii) Burgers' equation;

(ix) Laplace equation, with zero IC and both Neumann and Dirichlet BCs;

(x) Poisson equation in 2D.

Steps 11–12 solve the Navier-Stokes equation in 2D:

(xi) cavity flow;

(xii) channel flow.

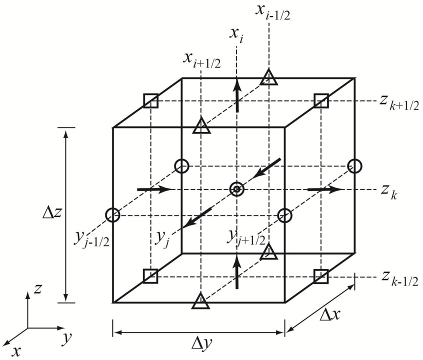
Students are instructed to follow these steps one by one, without skipping any! The most important step is #1, in fact. Everything builds from there.

Lorena A. Barba group:

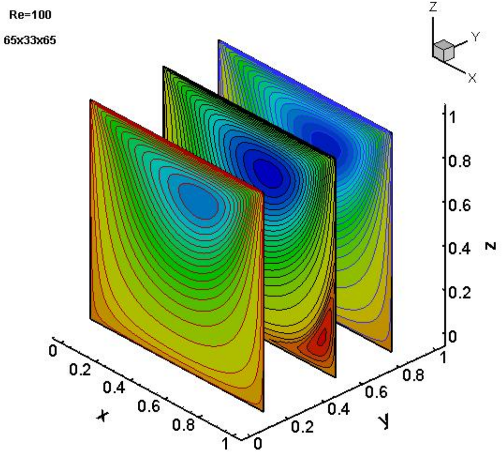
https://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/

https://github.com/barbagroup/CFDPython

Example (3D Driven Cavity Flow)

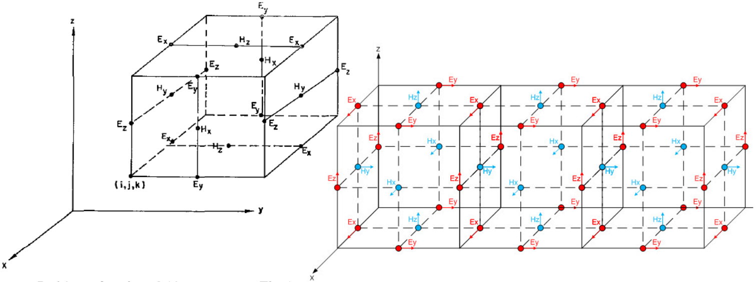


Velocity	↙ u	→ v	↑ w
Vorticity Vector potential	□ ω_x, ψ_x	△ ω_y, ψ_y	○ ω_z, ψ_z
Scalor potential	◎ ϕ		



Example (Maxwell's Equation and Yee Grid)

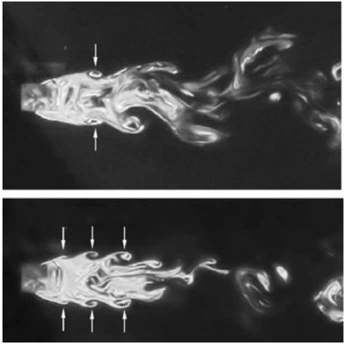
$$
\begin{aligned}
\nabla \cdot \mathrm{D} &= \rho_v, \\
\nabla \cdot \mathrm{B} &= 0, \\
\nabla \times \mathrm{E} &= -\frac{\partial \mathrm{B}}{\partial t}, \\
\nabla \times \mathrm{H} &= \mathrm{J} + \frac{\partial \mathrm{D}}{\partial t}.
\end{aligned}
$$



g. 1. Positions of various field components. The E-components are in the middle of the edges and the H-components are in the center of the faces.

Finite Difference Time Domain(FDTD) method: define different component of the Electric field $\mathrm{E} := (\mathrm{E}_x, \mathrm{E}_y, \mathrm{E}_z)$ and the magnetic field $\mathrm{H} := (H_x, H_y, H_z)$ at different surface of the so-called Yee **grid**.

Discussion: Flow Around Cylinder

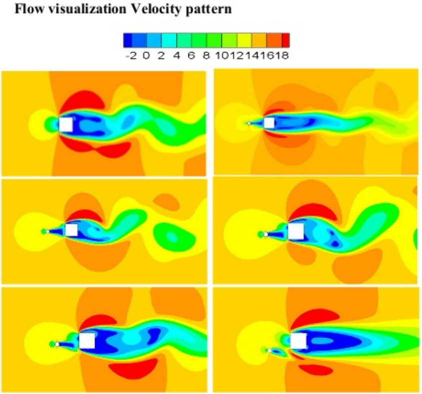


At L/D=1.9,d/D=0.267
Bare cylinder, $\alpha=1°$, $\alpha=4°$, $\alpha=6°$, $\alpha=7°$, $\alpha=19°$

In this figure as the staggered angle α is increasing the "shielding effect" is decreasing which results in the reduction of drag, due to decrease in the shielding effect, the back suction pressure decreases which is the main cause of drag reduction.

At $\alpha \geq 20$, the rod cylinder arrangement starts acting like a bare cylinder arrangement.

In wake independent region rod and cylinder gives the drag similar to that in bare cylinder case, which is 1.98 in the present computational work.



- `wavebridge.mpeg`

To do or Not to do ...

1. Why I need numerical solutions?
2. Is there any available software or open-source code?
3. Is the results I have obtain good enough?
4. Does a better algorithm exists?
5. Can I improve it?

Conclusion

13. Finite Difference Methods(FDMs) for Parabolic PDEs

14. Finite Volumn Methods(FVMs) for Hyperbolic PDEs

15. Numerical Methods for Classical (Steady State) PDEs