HW1 Advection-Diffusion Equations

June Yang

October 2019

1 Advection-Diffusion Equation in 1D

Consider numerically solving the advection-diffusion equation

$$u_t + au_x = (d(x)u_x)_x,$$

on the periodic domain $0 \le x \le 1$, for $a = 1$ and initial condition (remember not to use the specific analytical form of this in your solution, rather, treat it as some general smooth periodic function given to you by a *problem-specific input*)

$$u(x, 0) = [\sin(\pi x)]^{100},$$

on a uniform grid.

Here we focus on spatial discretization, and do not discretize time but rather study the consistency and stability of the semi-discrete approximation

$$\boldsymbol{w}'(t) = A\boldsymbol{w}(t) + \boldsymbol{g}(t), \tag{1}$$

where A depends on the choice of the spatial discretization.

Find some way to solve (1) with sufficient temporal accuracy. Ideally, we want to solve it "exactly", i.e., to numerical roundoff, so try that *instead* of using Matlab's ODE solvers as the lecture notes suggest. Explain what you did and indicate which norm(s) you used.

Hint: Use the Discrete Fourier Transform (DFT) to diagonalize the system of ODEs (1) and then solve it numerically using the FFT algorithm.

1.1 Pure advection

Consider pure advection, $d = 0$ and:

1. Compare the numerical solution $\boldsymbol{w}(t = 1)$ to the exact solution $u(x, 1)$ for the first-order upwind scheme, the second-order centered scheme, and the third-order upwind biased scheme, for several smartly-chosen grid resolutions. Comment on your observations, in particular their relation to artificial dissipation and dispersion as discussed in class.

The first-order upwind scheme is a second order accurate approximation for $u_t + au_x = \frac{1}{2}ahu_{xx}$. The term u_{xx} leads to diffusion. This phenomenon can also be explained by the eigenvalues of A. To find the eigenvalues of A, set $\frac{a}{h}(e^{2\pi ik(j-1)h} - e^{2\pi ikjh}) = \lambda_k e^{2\pi ikjh}$.

$$\begin{aligned}\lambda_k &= \frac{a}{h}(e^{-2\pi ikh} - 1) \\ &= \frac{a}{h}(\cos(2\pi kh) - 1) - i\frac{a}{h}\sin(2\pi kh), \quad k = 1, 2, \ldots, m\end{aligned}$$

$\mathrm{Re}(\lambda_k) < 0$ for all $k \neq m$. The damping factor $e^{t\mathrm{Re}(\lambda_k)}$ diminishes the amplitude of each wave.

The second-order centered scheme is a fourth order accurate approximation for the modified equation $u_t + au_x = -\frac{1}{6}ah^2u_{xxx}$. We can find the solution $\tilde{u}$ of this modified equation for each wave number k and its initial profile $\tilde{u}(x, 0) = \phi(x, 0) = e^{2\pi ikx}$. Set $\tilde{u}(x, t) = \phi(x)\gamma(t)$.

$$\begin{aligned}\gamma'\phi + a\gamma\phi' &= -\frac{1}{6}ah^2\gamma\phi''' \\ \frac{\gamma'}{\gamma} &= -\frac{1}{6}ah^2(2\pi ik)^3 - a(2\pi ik) \\ \gamma &= e^{-a(2\pi ik)(1-\frac{2}{3}(\pi kh)^2)t} \\ \tilde{u} &= e^{2\pi ik(x-a_kt)}, \quad a_k = a(1 - \frac{2}{3}(\pi kh)^2)\end{aligned}$$

Each Fourier mode moves with a different speed that depends on its wave number. This results in dispersive behaviors.

The numerical third-order upwind biased solution is closer to the solution of the advection-diffusion equation $u_t + au_x = -\frac{1}{12}ah^3u_{xxxx}$. The term u_{xxxx} is a dissipation term. However, unlike the first-order upwind scheme, this scheme is not non-negativity preserving.

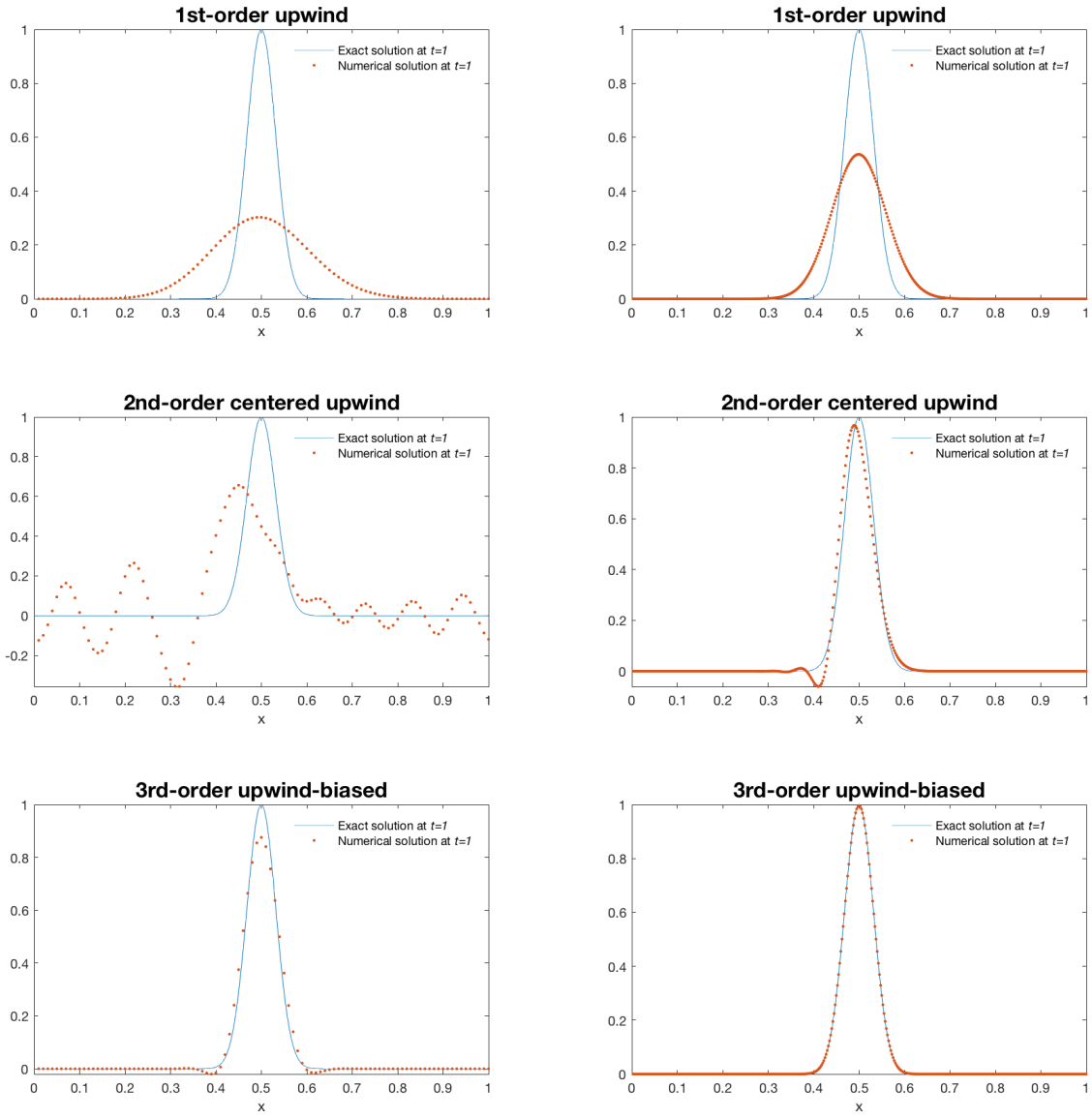


Figure 1: $m = 100$ (left) and $m = 400$ (right)

2. Compute the *relative* global error $\|\epsilon(t)\| = \|\boldsymbol{w}(t) - \boldsymbol{u}_h(t)\| \,/\, \|\boldsymbol{u}_h(t)\|$ at time $t = 1$ for different grid resolutions (think about how you choose them!), and estimate the spatial order of accuracy empirically.

See the output next page. Note upon halving h the relative global error for the first-order upwind scheme is estimated to be decreased by a factor of 2; however, this convergence behavior is not yet visible.

3. Plot the evolution of the *relative* global error $\|\epsilon(t)\| = \|\boldsymbol{w}(t) - \boldsymbol{u}_h(t)\| \,/\, \|\boldsymbol{u}_h(t)\|$ over the time interval $0 \leq t \leq 10$ for the first-order upwind scheme and the second-order centered scheme, and compare to the theoretical estimate from class.

Since the eigenvalues in the second-order centered scheme only have complex part ($\lambda_k = -\frac{ia}{h}\sin(2\pi kh)$),

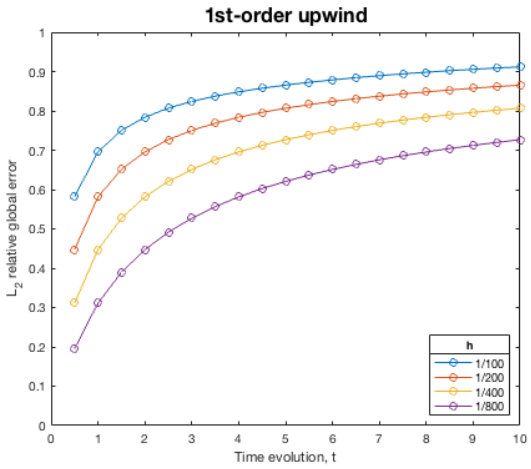
$$\left\|e^{tA}\right\| \leq Ke^{\omega t}, \quad 0 \leq t \leq T$$

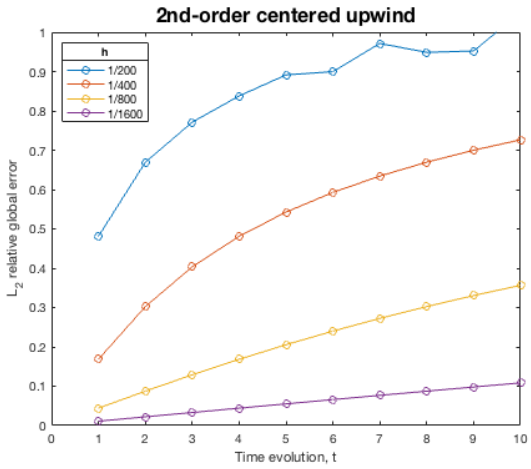
with $\omega = 0$ and a constant $K \geq 1$ independent of h.

By Theorem 4.1, Chapter 1 of the textbook, the global error

$$\|\boldsymbol{w}(t) - \boldsymbol{u}_h(t)\| \leq KC_0h^2 + KCh^2t, \quad 0 \leq t \leq T$$

with constants $C, C_0 > 0$. The error bound is a linear function of t. This linearity is reflected when h is small.





```
>> PureAdvection_order
Which method?
1st-order upwind
m     L1-error     L2-error     L_inf-error   ratio_1   ratio_2   ratio_3
100   1.03771911   0.69583963   0.69747162
200   0.81259487   0.58134691   0.59029442    1.28      1.20      1.18
400   0.58546442   0.44714581   0.46366795    1.39      1.30      1.27
800   0.38552688   0.31117514   0.33141461    1.52      1.44      1.40
1600   0.23252620   0.19555027   0.21384464    1.66      1.59      1.55
3200   0.13054283   0.11275920   0.12589086    1.78      1.73      1.70
>> PureAdvection_order
Which method?
2nd-order centered upwind
m     L1-error     L2-error     L_inf-error   ratio_1   ratio_2   ratio_3
100   1.78315752   0.83839130   0.55093899
200   0.72608944   0.48081104   0.42157974    2.46      1.74      1.31
400   0.19586177   0.16832552   0.17680611    3.71      2.86      2.38
800   0.04897896   0.04416571   0.04603273    4.00      3.81      3.84
1600   0.01226431   0.01108129   0.01125709    3.99      3.99      4.09
3200   0.00306668   0.00277106   0.00279191    4.00      4.00      4.03
>> PureAdvection_order
Which method?
3rd-order upwind-biased
m     L1-error     L2-error     L_inf-error   ratio_1   ratio_2   ratio_3
100   0.14824501   0.12022896   0.12422567
200   0.02622379   0.02318086   0.02588492    5.65      5.19      4.80
400   0.00352351   0.00319342   0.00368439    7.44      7.26      7.03
800   0.00044562   0.00040515   0.00047061    7.91      7.88      7.83
1600   0.00005579   0.00005075   0.00005902    7.99      7.98      7.97
3200   0.00000697   0.00000635   0.00000738    8.00      8.00      8.00
>>
```

1.2 Advection-Diffusion

For this part choose one or several "good" spatial discretizations and *write down* the scheme you used (I should not need to look in the code). For each scheme, explain what its advantages and disadvantages are, and what its theoretical (spatial) order of accuracy is.

I used the following spatial discretization scheme.

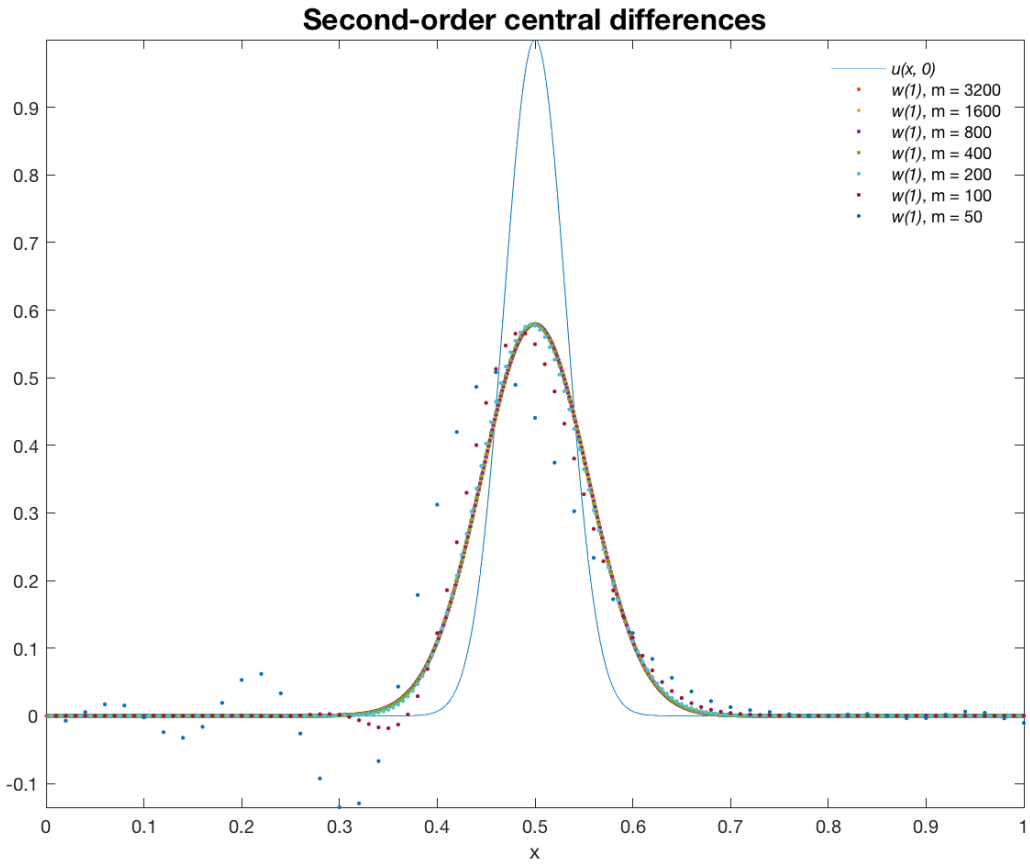
$$w_j^{'}(t) = (\frac{d}{h^2} + \frac{a}{2h})w_{j-1}(t) - \frac{2d}{h^2}w_j(t) + (\frac{d}{h^2} - \frac{a}{2h})w_{j+1}(t)$$

To be discussed... advantages and disadvantages of this scheme...

1.2.1 Constant Diffusion

Let's add a small amount of constant diffusion, $d = 0.001$.

1. Compare the numerical solution $\boldsymbol{w}(t = 1)$ to the exact solution $u(x, 1)$ (explain how you computed the "exact" solution to roundoff tolerance) for several resolutions and comment on your observations.



2. Compute the *relative* global error $\|\epsilon(t)\| = \|\boldsymbol{w}(t) - \boldsymbol{u}_h(t)\| \,/\, \|\boldsymbol{u}_h(t)\|$ at time $t = 1$ for different grid resolutions, and estimate the spatial order of accuracy empirically.

See the output next page.

```
>> AdvectionDiffusion
m     L1-error     L2-error     L_inf-error  ratio_1   ratio_2   ratio_3
3200   0.00000000   0.00000000   0.00000000
1600   0.00044762   0.00040576   0.00040911
800   0.00223813   0.00202882   0.00204898    5.00      5.00      5.01
400   0.00939740   0.00852048   0.00865726    4.20      4.20      4.23
200   0.03803250   0.03442360   0.03581548    4.05      4.04      4.14
100   0.15179820   0.13399748   0.14162688    3.99      3.89      3.95
50   0.59204369   0.41556297   0.37988326    3.90      3.10      2.68
>>
```

Code

```
% HW1: Advection-Diffusion Equations
% Advection-Diffusion Equation in 1D
% u_t + a * u_x = d * u_xx

% Spatial discretization only - no time discretization
% Study the consistency and stability of the semi-discrete approximation
% w'(t) = Aw(t) + g(t) ... (1)

% Use the Discrete Fourier Transform (DFT) to diagonalize the system of
% ODEs (1) and then solve it numerically using the FFT algorithm.


% 1.1 Pure advection (d = 0)
% i) Plot: numerical solution w(t = 1) vs exact solution u(x, 1)

a = 1;

m = 400;
h = 1 / m;

f = @(x) (sin(pi * x)) .^ 100;
fplot(f, [0 1]);
hold on

x_grid = h : h : 1;
w_init = f(x_grid);

method = '3rd-order upwind-biased';

switch method

    case '1st-order upwind'

        A = eye(m);
        subdiag_A = ones(m - 1, 1);
        A = diag(-subdiag_A, -1) + A;
        A(1, m) = -1;
        A = - a / h * A;

    case '2nd-order centered upwind'

        subdiag_A = ones(m - 1, 1);
        A = diag(-subdiag_A, -1) + diag(subdiag_A, 1);
        A(1, m) = -1;
        A(m, 1) = 1;
        A = - 2 * a / h * A;

    case '3rd-order upwind-biased'

        A = - 1 / 2 * eye(m);
        subdiag_A = ones(m - 1, 1);
        A = diag(subdiag_A, -1) + diag(- 1 / 3 * subdiag_A, 1) + A;
        subdiag_A = ones(m - 2, 1);
        A = diag(- 1 / 6 * subdiag_A, -2) + A;
        A(1, m) = 1;
        A(1, m - 1) = - 1 / 6;
        A(2, m) = - 1 / 6;
        A(m, 1) = - 1 / 3;
```

```
        A = a / h * A;

end

S = dftmtx(m);
S_inv = (1 / m) * conj(S);
D = S_inv * A * S;

% w = S * diag(exp(diag(D))) * S_inv * w_init(:);
w = fft(diag(exp(diag(D)))) * ifft(w_init(:)); % w(t = 1)

plot(h : h : 1, real(w), '.')
xlabel('x')
legend('Exact solution at \itt=1', 'Numerical solution at \itt=1', 'Location',↙
'northeast');
legend('boxoff')
title(method, 'fontsize', 16)
```

```
% 1.1 Pure advection (d = 0)
% ii) Compute the relative global error at time t = 1 for different grid
% resolutions

function [L1err, L2err, L_inferr] = PureAdvection_err(m, method)

    a = 1;

    h = 1 / m;

    f = @(x) (sin(pi * x)) .^ 100;

    x_grid = h : h : 1;
    w_init = f(x_grid);

    switch method

        case '1st-order upwind'

            A = eye(m);
            subdiag_A = ones(m - 1, 1);
            A = diag(-subdiag_A, -1) + A;
            A(1, m) = -1;
            A = - a / h * A;

        case '2nd-order centered upwind'

            subdiag_A = ones(m - 1, 1);
            A = diag(-subdiag_A, -1) + diag(subdiag_A, 1);
            A(1, m) = -1;
            A(m, 1) = 1;
            A = - 2 * a / h * A;

        case '3rd-order upwind-biased'

            A = - 1 / 2 * eye(m);
            subdiag_A = ones(m - 1, 1);
            A = diag(subdiag_A, -1) + diag(- 1 / 3 * subdiag_A, 1) + A;
            subdiag_A = ones(m - 2, 1);
            A = diag(- 1 / 6 * subdiag_A, -2) + A;
            A(1, m) = 1;
            A(1, m - 1) = - 1 / 6;
            A(2, m) = - 1 / 6;
            A(m, 1) = - 1 / 3;
            A = a / h * A;

    end

    S = dftmtx(m);
    S_inv = (1 / m) * conj(S);
    D = S_inv * A * S;

    w = fft(diag(exp(diag(D)))) * ifft(w_init(:)); % w(t = 1)

    L1err = sum(abs(real(w) - w_init(:))) * h / (sum(abs(w_init(:))) * h); % relative↙
global error
    L2err = sqrt(sum((real(w) - w_init(:)).^2) * h) / sqrt(sum((w_init(:)).^2) * h);
    L_inferr = max(abs(real(w) - w_init(:))) / max(abs(w_init(:)));
```

```
end
```

```
% 1.1 Pure advection (d = 0)
% ii) Estimate the spatial order of accuracy empirically

prompt = 'Which method?\n';
method = input(prompt, 's');

disp('m     L1-error     L2-error     L_inf-error   ratio_1   ratio_2   ratio_3');

L1err_pre = NaN;
L2err_pre = NaN;
Linferr_pre = NaN;

for i = 1 : 6

    m = 100 * (2 ^ (i - 1));

    [L1err, L2err, Linferr] = PureAdvection_err(m, method);

    if i == 1

        row = [m L1err L2err Linferr];
        fprintf('%d   %0.8f   %0.8f   %0.8f\n', row);

    else

        ratio_1 = L1err_pre / L1err;
        ratio_2 = L2err_pre / L2err;
        ratio_3 = Linferr_pre / Linferr;

        row = [m L1err L2err Linferr ratio_1 ratio_2 ratio_3];
        fprintf('%d   %0.8f   %0.8f   %0.8f    %0.2f      %0.2f      %0.2f\n', row);

    end

    L1err_pre = L1err;
    L2err_pre = L2err;
    Linferr_pre = Linferr;

end
```

```
% 1.1 Pure advection (d = 0)
% iii) Plot the evolution of the relative global error over the time interval
% (0, 10]

a = 1;

for i = 1 : 4

    m = 100 * 2 ^ (i - 1);
    h = 1 / m;

    f = @(x) (sin(pi * x)) .^ 100;

    x_grid = h : h : 1;
    w_init = f(x_grid); % u(t = 0, 1, ...)
    u_half = f(x_grid - 0.5); % u(t = 0.5, 1.5, ...)

    method = '1st-order upwind';

    switch method

        case '1st-order upwind'

            A = eye(m);
            subdiag_A = ones(m - 1, 1);
            A = diag(-subdiag_A, -1) + A;
            A(1, m) = -1;
            A = - a / h * A;

        case '2nd-order centered upwind'

            subdiag_A = ones(m - 1, 1);
            A = diag(-subdiag_A, -1) + diag(subdiag_A, 1);
            A(1, m) = -1;
            A(m, 1) = 1;
            A = - 2 * a / h * A;

        case '3rd-order upwind-biased'

            A = - 1 / 2 * eye(m);
            subdiag_A = ones(m - 1, 1);
            A = diag(subdiag_A, -1) + diag(- 1 / 3 * subdiag_A, 1) + A;
            subdiag_A = ones(m - 2, 1);
            A = diag(- 1 / 6 * subdiag_A, -2) + A;
            A(1, m) = 1;
            A(1, m - 1) = - 1 / 6;
            A(2, m) = - 1 / 6;
            A(m, 1) = - 1 / 3;
            A = a / h * A;

    end

    S = dftmtx(m);
    S_inv = (1 / m) * conj(S);
    D = S_inv * A * S;

    norm_2 = NaN(20, 1);
```

```
    for k = 1 : 20

        t = k / 2;
        w = fft(diag(exp(diag(D * t)))) * ifft(w_init(:)); % w(t)

        if mod(k, 2) == 1
            norm_2(k) = sqrt(sum((real(w) - u_half(:)).^2) * h) / sqrt(sum(u_half(:).^2)↙
* h);
        else
            norm_2(k) = sqrt(sum((real(w) - w_init(:)).^2) * h) / sqrt(sum(w_init(:).^2)↙
* h);
        end

    end

    plot(0.5 : 0.5 : 10, norm_2, '-o')
    hold on

end

axis([0 10 0 1])
xlabel('Time evolution, t')
ylabel('{L_2} relative global error')

lgd = legend({'1/100', '1/200', '1/400', '1/800'}, 'Location', 'southeast');
title(lgd, 'h')

title(method, 'fontsize', 16)

hold off
```

```
% 1.2 Advection-Diffusion (d = 0.001)
% i) Plot the numerical solution w(t = 1) for several resolutions
% ii) Compute the relative global error at time t = 1 for different grid
% resolutions and estimate the spatial order of accuracy empirically.

a = 1;
d = 0.001;

f = @(x) (sin(pi * x)) .^ 100;
fplot(f, [0 1]);
hold on

disp('m     L1-error     L2-error     L_inf-error  ratio_1   ratio_2   ratio_3');

K = 7;
W = NaN(50 * 2 ^ (K - 1), 1); % to store the "exact" solution
L1err_pre = NaN;
L2err_pre = NaN;
Linferr_pre = NaN;

for i = K : -1 : 1

    m = 50 * 2 ^ (i - 1);
    h = 1 / m;

    x_grid = h : h : 1;
    w_init = f(x_grid);

    A = - (2 * d) / (h ^ 2) * eye(m);
    subdiag_A = ones(m - 1, 1);
    sub_const_m = d / (h ^ 2) + a / (2 * h);
    sub_const_p = d / (h ^ 2) - a / (2 * h);
    A = sub_const_m * diag(subdiag_A, -1) + A + sub_const_p * diag(subdiag_A, 1);
    A(1, m) = sub_const_m;
    A(m, 1) = sub_const_p;

    S = dftmtx(m);
    S_inv = (1 / m) * conj(S);
    D = S_inv * A * S;

    % w = S * diag(exp(diag(D))) * S_inv * w_init(:);
    w = fft(diag(exp(diag(D)))) * ifft(w_init(:)); % w(t = 1)

    if i == K
        W = real(w); % "exact" solution
    end

    u = W(2 ^ (K - i) : 2 ^ (K - i) : 50 * 2 ^ (K - 1));
    L1err = sum(abs(real(w) - u)) * h / (sum(abs(u)) * h);
    L2err = sqrt(sum((real(w) - u).^2) * h) / sqrt(sum(u.^2) * h);
    Linferr = max(abs(real(w) - u)) / max(abs(u));

    if (i == K) || (i == K - 1)

        row = [m L1err L2err Linferr];
        fprintf('%d   %0.8f   %0.8f   %0.8f\n', row);

    else
```

```
        ratio_1 = L1err / L1err_pre;
        ratio_2 = L2err / L2err_pre;
        ratio_3 = Linferr / Linferr_pre;

        row = [m L1err L2err Linferr ratio_1 ratio_2 ratio_3];
        fprintf('%d   %0.8f   %0.8f   %0.8f    %0.2f      %0.2f      %0.2f\n', row);

    end

    L1err_pre = L1err;
    L2err_pre = L2err;
    Linferr_pre = Linferr;

    plot(h : h : 1, real(w), '.')
    hold on

end

xlabel('x')
legend('\itu(x, 0)', '{\itw(1)}, m = 3200', '{\itw(1)}, m = 1600', '{\itw(1)}, m = 800',↙
'{\itw(1)}, m = 400', '{\itw(1)}, m = 200', '{\itw(1)}, m = 100', '{\itw(1)}, m = 50',↙
'Location', 'northeast');
legend('boxoff')
title('Second-order central differences', 'fontsize', 16)
hold off
```