ECMM723: Modelling the Weather and Climate

Coursework 2: Numerical Methods and Sub-grid Processes

Qun Liu (Student No: 670016014)
ql260@exeter.ac.uk
College of Enginerring, Mathematics and Physical Sciences

March, 2018

1. Numerical methods

(a) Write a MATLAB code to integrate the linear advection equation until $t = 0.5$ using a centred-in-time, centred-in-space (CTCS) discretization. Perform the integration for both sets of initial conditions. Take $u = 2.0$, use $N = 51$ equally spaced gridpoints, and take $\Delta t = 0.005$.

Solution:

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

$$x_j = j\Delta x,\ \ j = 0, 1, 2, ..,$$

$$t^{(n)} = n\Delta t, n = 0, 1, 2, ...$$

Using centred-in-time and centred-in-space discretization, we could get

$$\frac{\phi_j^{(n+1)} - \phi_j^{(n-1)}}{2\Delta t} + u\,\frac{\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}}{2\Delta x} = 0$$

$$\phi_j^{(n+1)} = \phi_j^{(n-1)} - \frac{u\Delta t}{\Delta x}\left(\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}\right)$$

This is three-time-level method, we should know ϕ values at t_{n-1}, t_n and t_{n+1}. To start the simulation, values of ϕ are needed at times t_0 and t_1. However, only $\phi(x, t_0)$ is available. So we use FTCS scheme to obtain $\phi^{(1)} = \phi(x, t_1)$, that is

$$\text{FTCS}:\quad \phi_j^{(1)} = \phi_j^{(0)} - \frac{u\Delta t}{2\Delta x}\left(\phi_{j+1}^{(0)} - \phi_{j-1}^{(0)}\right) \quad (n = 0),$$

and we use CTCS scheme when $n \geq 1$, that is

$$\phi_j^{(n+1)} = \phi_j^{(n-1)} - \frac{u\Delta t}{\Delta x}\left(\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}\right) \quad (n = 1, 2, ...).$$

The result of CTCS scheme is shown in the Figure 1.

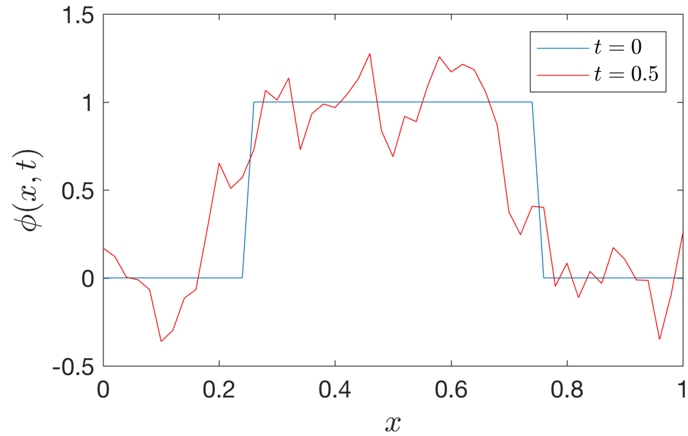

Figure 1: Numerical calculation of linear advection equation with CTCS scheme.

(b) Discretise the advection equation using a semi-Lagrangian scheme, and cubic-Lagrange interpolation. The departure points can be evaluated using a simple linear formula, i.e. a displacement of $u\Delta t$ upstream. Demonstrate how this formulation improves on the CTCS scheme when increasing the time step above the CFL limit.

Solution: Semi-Lagrangian schemes discretize the advective form of the advection equation

$$\frac{\mathrm{D}\,\phi}{\mathrm{D}\,t} = 0,$$

giving

$$\frac{\phi_j^{(n+1)} - \phi_{jD}^{(n)}}{\Delta t} = 0,$$

$$\phi_j^{(n+1)} = \phi_{jD}^{(n)},$$

where $\phi_{jD}^{(n)}$ is the $\phi^{(n)}$ value at $x_j - u\Delta t$, which could be get by cubic Lagrange interpolation, that is

$$\begin{aligned}\widehat{\phi_{jD}^{(n)}}(x) &= -\tfrac{1}{6}\beta(1-\beta)(2-\beta)\phi_{k-1}^{(n)} + \tfrac{1}{2}(1+\beta)(1-\beta)(2-\beta)\phi_k^{(n)} \\ &\quad + \tfrac{1}{2}(1+\beta)\beta(2-\beta)\phi_{k+1}^{(n)} - \tfrac{1}{6}(1+\beta)\beta(1-\beta)\phi_{k+2}^{(n)}\end{aligned}$$

where $\beta = (x - x_k)/\left(x_{k+1} - x_k\right) = (x - x_k)/\Delta x$, and k satisfies that $x_j - u\Delta t$ lies between x_k and x_{k+1}. If we define

$$\mu \equiv \frac{u\Delta t}{\Delta x},$$

and

$$p = [\mu] = \text{ Integral part of } \mu,$$

then

$$k = j - (p+1).$$

When using the semi-Lagrange scheme to integrate the linear advection equation from $t = 0$ to $t = 0.5$, the results are shown in Figure 2.

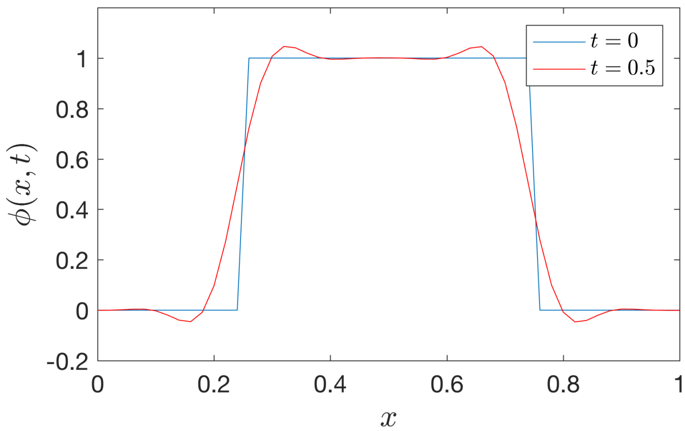

Figure 2: Numerical calculation of linear advection equation with semi-Lagrange scheme.

The CFL condition requires that

$$\mu = \frac{u\Delta t}{\Delta x} \le 1,$$

here $u = 2, \Delta x = \frac{1}{N-1} = \frac{1}{50} = 0.02$. So if we use CTCS scheme to discretize the advection equation, the Δt must satisfy

$$\Delta t < \frac{\Delta x}{u} = \frac{0.02}{2} = 0.01.$$

In 1(a), the CTCS scheme use $\Delta t = 0.005$, which satisfies the CFL condition.

For the semi-Lagrange scheme, if we choose $\Delta t > 0.01$, we could get different final state of ϕ (Figure 3), which are all stable and Δt is no longer limited by CFL condition.

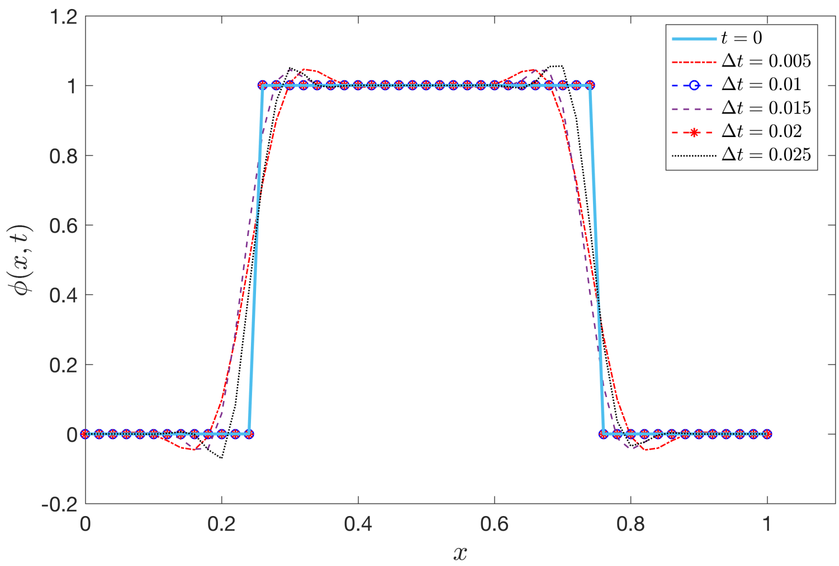

Figure 3: Change the Δt in semi-Lagrange scheme.

2. Sub-grid processes

Holton, section 5.3 gives a mixed-layer model of the atmospheric boundary layer

$$\overline{u} = \overline{u_g} - \kappa_s|\overline{\boldsymbol{V}}|\overline{v}, \quad \overline{v} = \kappa_s|\overline{\boldsymbol{V}}|\overline{u}$$

where the turbulence magnitude is given by

$$\kappa_s = C_d/(fh)$$

(a) Explain the relationship between the above mixed-layer equations and the Reynolds averaged momentum equations, explaining carefully the meaning of the notation.

Solution: The momentum and continuity equation under the Boussinesq approximation are

$$\frac{Du}{Dt} = -\frac{1}{\rho_0}\frac{\partial p}{\partial x} + fv + F_{rx}, \tag{1}$$

$$\frac{Dv}{Dt} = -\frac{1}{\rho_0}\frac{\partial p}{\partial y} - fu + Fry, \tag{2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{3}$$

where

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}.$$

Combine it with (3), we could get

$$\begin{aligned}\frac{Du}{Dt} &= \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + u\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \\ &= \frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z}\end{aligned}. \tag{4}$$

Take the Reynolds average over the (4), we could get

$$\overline{\frac{Du}{Dt}} = \frac{\partial \overline{u}}{\partial t} + \frac{\partial}{\partial x}\left(\overline{u}\overline{u} + \overline{u'u'}\right) + \frac{\partial}{\partial y}\left(\overline{u}\overline{v} + \overline{u'v'}\right) + \frac{\partial}{\partial z}\left(\overline{u}\overline{w} + \overline{u'w'}\right),$$

which could be re-written as

$$\overline{\frac{Du}{Dt}} = \frac{\overline{D}\overline{u}}{Dt} + \frac{\partial}{\partial x}\left(\overline{u'u'}\right) + \frac{\partial}{\partial y}\left(\overline{u'v'}\right) + \frac{\partial}{\partial z}\left(\overline{u'w'}\right),$$

where $\frac{\overline{D}}{Dt}$ is defined as

$$\frac{\overline{D}}{Dt} = \frac{\partial}{\partial t} + \overline{u}\frac{\partial}{\partial x} + \overline{v}\frac{\partial}{\partial y} + \overline{w}\frac{\partial}{\partial z},$$

which is the rate of change following the mean motion.

Hence, if we take Reynolds average in both sides of (1) and (2), we could obtain

$$\frac{\overline{D}\overline{u}}{Dt} = -\frac{1}{\rho_0}\frac{\partial\overline{p}}{\partial x} + f\overline{v} - \left[\frac{\partial\overline{u'u'}}{\partial x} + \frac{\partial\overline{u'v'}}{\partial y} + \frac{\partial\overline{u'w'}}{\partial z}\right] + \overline{F}_{rx} \tag{5}$$

$$\frac{\overline{D}\overline{v}}{Dt} = -\frac{1}{\rho_0}\frac{\partial\overline{p}}{\partial y} - f\overline{u} - \left[\frac{\partial\overline{u'v'}}{\partial x} + \frac{\partial\overline{v'v'}}{\partial y} + \frac{\partial\overline{v'w'}}{\partial z}\right] + \overline{F}_{ry} \tag{6}$$

If we neglect the molecular viscosity and horizontal turbulent momentum flux divergence terms, (5) and (6) then become

$$\frac{\overline{D}\overline{u}}{Dt} = -\frac{1}{\rho_0}\frac{\partial\overline{p}}{\partial x} + f\overline{v} - \frac{\partial\overline{u'w'}}{\partial z} \tag{7}$$

$$\frac{\overline{D}\overline{v}}{Dt} = -\frac{1}{\rho_0}\frac{\partial\overline{p}}{\partial y} - f\overline{u} - \frac{\partial v'w'}{\partial z} \tag{8}$$

Outside the boundary layer, the resulting approximation was then simply geostrophic balance., that is

$$-fv_g = -\frac{1}{\rho_0}\frac{\partial\overline{p}}{\partial x}, \tag{9}$$

$$fu_g = -\frac{1}{\rho_0}\frac{\partial\overline{p}}{\partial y}. \tag{10}$$

Put the (9) and (10) into (7) and (8) respectively, and ignore the inertial terms assuming that Rossby number is small, we could get

$$f\left(\overline{v} - \overline{v}_g\right) - \frac{\partial\overline{u'w'}}{\partial z} = 0 \tag{11}$$

$$-f\left(\overline{u} - \overline{u}_g\right) - \frac{\partial\overline{v'w'}}{\partial z} = 0 \tag{12}$$

In a well-mixed boundary layer, velocity and potential temperature profiles are constant with height and turbulent fluxes vary linearly with height. For simplicity, we assume that the turbulence vanishes at the top of the boundary layer (h), that is

$$\left(\overline{u'w'}\right)_h = 0, \quad \text{and} \quad \left(\overline{v'w'}\right)_h = 0.$$

The surface momentum flux can be represented as

$$\left(\overline{u'w'}\right)_s = -C_d|\overline{\mathbf{V}}|\overline{u}, \quad \text{and} \quad \left(\overline{v'w'}\right)_s = -C_d|\overline{\mathbf{V}}|\overline{v}$$

where C_d is a nondimensional drag coefficient, $|\overline{\mathbf{V}}| = \left(\overline{u}^2 + \overline{v}^2\right)^{1/2}$, and the subscript s denotes surface values. The approximate planetary boundary layer equations (11) and (12) can then be integrated from the surface to the top of the boundary layer at $z = h$ to give

$$f\left(\overline{v} - \overline{v}_g\right) = 0 - \left(\overline{u'w'}\right)_s/h = C_d|\overline{\mathbf{V}}|\overline{u}/h \tag{13}$$

$$-f\left(\overline{u} - \overline{u}_g\right) = 0 - \left(\overline{v'w'}\right)_s/h = C_d|\overline{\mathbf{V}}|\overline{v}/h \tag{14}$$

$$\Longrightarrow \overline{v} = \overline{v}_g + \kappa_s|\overline{\mathbf{V}}|\overline{u}, \quad \overline{u} = \overline{u}_g - \kappa_s|\overline{\mathbf{V}}|\overline{v}, \tag{15}$$

Without loss of generality we can choose axes such that $\overline{v}_g = 0$, then (13) and (14) can be rewritten as

$$\overline{v} = \kappa_s|\overline{\mathbf{V}}|\overline{u}, \quad \overline{u} = \overline{u}_g - \kappa_s|\overline{\mathbf{V}}|\overline{v}, \tag{16}$$

where

$$\kappa_S \equiv C_d/(fh).$$

(b) You are given the code *mixed_layer_wind1.m*. Modify the code to calculate the wind turning (the angle between the geostrophic and mixed layer winds) for a range of values of h. Use the values: $u_g = 10ms^{-1}, C_d = 2 \times 10^{-3}$ and $f = 10^{-4}s^{-1}$. Calculate and plot the wind turning for values of h in the range of 300m to 3000m.

Solution:

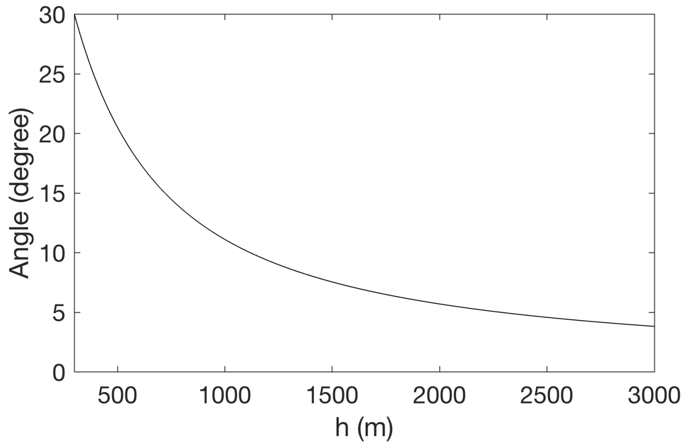

Figure 4: The angle between the geostrophic and mixed layer winds.

(c) You are given the code *mixed_layer_wind2.m* which prescribes the geostrophic wind (u_g, v_g) at each horizontal point. Calculate the mixed-layer wind $(\bar{u}, \bar{v})$ at each horizontal point using the algorithm in *mixed_layer_wind1.m*. Hint: you may want to incorporate the mixed-layer algorithm as a function. Plot the mixed-layer wind as a vector-wind plot (as done for the geostrophic wind). Assuming a constant boundary-layer depth of $1000m$, calculate the Ekman pumping for each horizontal point and plot the results.

Solution: The mixed-layer wind is shown in Figure 5 (b).

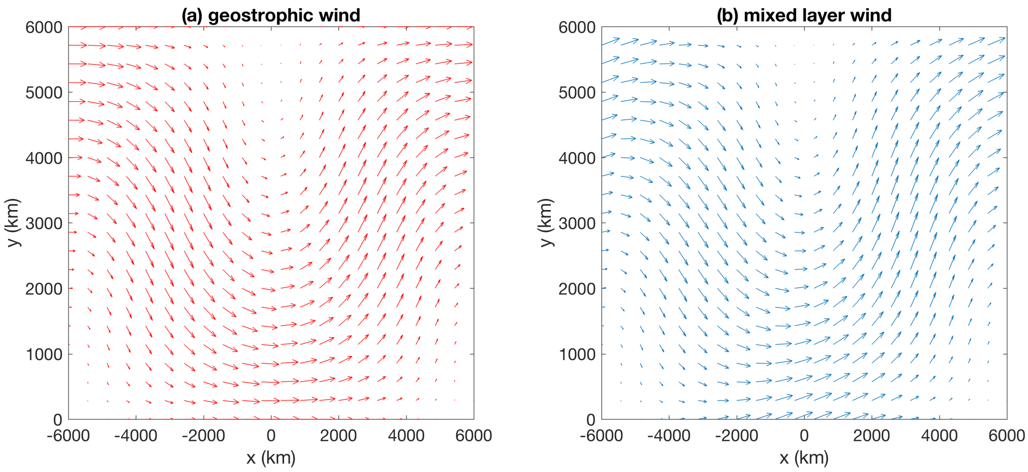

Figure 5: Comparison of geostrophic and mixed layer winds.

Eckman pumping could be represented by the vertical velocity

$$w_h = -h \left(\frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial y} \right),$$

which is shown in Figure 6.

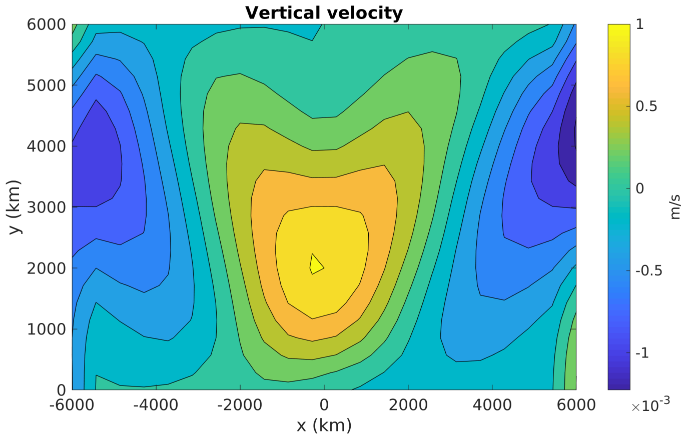

Figure 6: Eckman pumping (vertical velocity).

Matlab Code

Problem 1

- ic.m

```
function val=ic(x);
%
% Initial condition for diffusion problem
if (0.25 <= x) & (x <= 0.75)
  val=1.0;
else
  val=0.0;
end;
```

- next.m

```
function next_num = next(i, m_next, len)
% Input:
%    i is the current position
%    m_next is the next m th position
%    len is the number of list
% Output:
%    next_num is the index of the item m_next of i in the list

    next_num = mod(i + m_next - 1, len) + 1;

end
```

- prev.m

```
function prev_num = prev(i, m_prev, len)
% Input:
%    i is the current position
%    m_prev is the previous m th position
%    len is the number of list
% Output:
%    prev_num is the index of the item m_prev of i in the list

    prev_num = mod(i-m_prev+len-1, len) + 1;

end
```

- prob1_a_linear_advection_CTCS.m

```matlab
% Linear advection with CTCS scheme

clear; clc; close all

figure;

u = 2.0; % u velocity
t = 0.5; % time to run
nx = 51; % no. of grid points
dt = 0.005; % time step
dx = 1.0/(nx-1); % grid length
x = (0:1:nx-1).*dx;
c = (u*dt)/dx;

phi = zeros(nx,1);  % phi at time n
phip = zeros(nx,1); % phi at time n+1
phim = zeros(nx,1); % phi at time n-1

% Set initial condition
for j = 1:nx
    phi(j) = ic(x(j));
end
% Plot the initial phi
h1 = plot(x, phi, '-');

%% Get the values at first time step with FTCS method
% Loop over grid points
for j = 1:nx
    jm = prev(j, 1, nx);     % index of item previous to j
    jp = next(j, 1, nx);     % index of item next to j
    phip(j) = phi(j) - c/2.0*(phi(jp)- phi(jm));
end % j

phim = phi; % t = 0
phi = phip; % t = 1*dt

%% Loop over steps with CTCS scheme
nstep = t/dt;   % total number time step approximately
for istep = 2:nstep
  % Loop over grid points
  for j=1:nx
    jm = prev(j, 1, nx); % index of item previous to j
    jp = next(j, 1, nx); % index of item next to j
    phip(j) = phim(j) - c*(phi(jp)- phi(jm)); % phi at time t = (n+1)*dt
  end % j
  phim = phi; % t = (n-1)*dt
  phi = phip; % t = n*dt
end % istep

%% Plot the phi at time t
hold on
h2 = plot(x, phi,'r');

% Add legends
legend([h1,h2], {'$$t=0$$','$$t=0.5$$'},'Interpreter','latex')
xlim([0,1])
set(gca, 'FontSize', 14)
% Add x and y label
```

```
xlabel('$$x$$', 'Interpreter','latex', 'FontSize', 18);
ylabel('$$\phi (x,t)$$', 'Interpreter','latex', 'FontSize', 18);

% Print the figure
set(gcf, 'PaperUnits', 'Inches', 'PaperPosition', [0, 0, 6, 6*0.618]);
print(gcf,'linear_advection_CTCS.png','-dpng','-r300');
```

- prob1_b1_linear_advection_semi_Lagrange.m

```
% Linear advection with semi-Lagrange scheme

clear; clc; close all;

u = 2.0; % u velocity
t = 0.5; % total time to simulate
nx = 51; % no. of grid points
dt = 0.005; % time step
L = 1.0;
dx = L/(nx-1); % grid length
x = (0:L:nx-1).*dx;

phi = zeros(nx,1);  % phi at n time step
phip = zeros(nx,1); % phi at next step

% Set initial condition
for j = 1:nx
    phi(j) = ic(x(j));
end
% Plot the initial phi
h1 =plot(x, phi, '-');

%% Loop over steps with semi-Lagrange method and cubic Lagrange interpolation
nstep = t/dt;
for istep = 1:nstep
  % Loop over grid points
  for j=1:nx
    % locate the new position of the fluid parcel
    x0 = mod(x(j)-u*dt, L);
    % find the parcel indexes that will be used in cubic Lagrange interpolation
    p = floor(u*dt/dx);
    jl = prev(j, p+1, nx);  % j-1
    jr = next(jl, 1, nx);   % j+1
    jll = prev(jl, 1, nx);  % j-2
    jrr = next(jl, 2, nx);  % j+2
    % Use cubic Lagrange interpolation to get the phi at next step
    phip(j) = cubic_lagrange_interp( x0, x(jl), x(jr),...
        phi(jll), phi(jl), phi(jr), phi(jrr) );
  end % j
  phi = phip;   % new phi at istep
end % istep

%% Plot the phi at time t
hold on
h2 = plot(x, phi,'r');
% Add legend
legend([h1,h2], {'$$t=0$$','$$t=0.5$$'},'Interpreter','latex')
xlim([0,1])
set(gca, 'FontSize', 14)
% Add x and y labels
```

```
xlabel('$$x$$', 'Interpreter','latex', 'FontSize', 18);
ylabel('$$\phi (x,t)$$', 'Interpreter','latex', 'FontSize', 18);
% Print the figure
set(gcf, 'PaperUnits', 'Inches', 'PaperPosition', [0, 0, 6, 6*0.618]);
print(gcf,'linear_advection_semi_Lagrange.png','-dpng','-r300');
```

- prob1_b2_linear_advection_semi_Lagrange_different_dt.m

```
% Plot linear advection with different time steps
clear; clc; close all;

figure;

u = 2.0; % u velocity
t = 0.5; % the total time to run
nx = 51; % no. of grid points
L = 1.0; % the length of x

dtlist = 0.005:0.005:0.025; % A list of time steps

% marks for plots with different time steps
marks = {'r-.','b--o','--','r--*','k:'};
h2 = {};
labels = {};

for i = 1:length(dtlist)
    dx = L/(nx-1); % grid length
    x = (0:L:nx-1).*dx;
    phi = zeros(nx,1);
    phip = zeros(nx,1);

    % Set initial condition
    for j = 1:nx
        phi(j) = ic(x(j));
    end
    % Plot inital state
    h1 = plot(x, phi, '-', 'LineWidth', 2);
    hold on

    % Loop over steps with semi-Lagrange method and cubic Lagrange interpolation
    dt = dtlist(i);
    nstep = t/dt;
    for istep = 1:nstep
      % Loop over grid points
      for j=1:nx
        % locate the new position of the fluid parcel
        x0 = mod(x(j)-u*dt, L);
        % find the parcel indexes that will be used in cubic Lagrange interpolation
        p = floor(u*dt/dx);
        jl = prev(j, p+1, nx);
        jr = next(jl, 1, nx);
        jll = prev(jl, 1, nx);
        jrr = next(jl, 2, nx);
        % Use cubic Lagrange interpolation to get the phi at next step
        phip(j) = cubic_lagrange_interp( x0, x(jl), x(jr),...
            phi(jll), phi(jl), phi(jr), phi(jrr) );
      end % j
      phi = phip;
    end % istep
```

```
    % plot the phi for different time steps
    h2{i} = plot(x, phi, marks{i}, 'LineWidth', 1);
    labels{i} = ['$$\Delta t=', num2str(dt), '$$'];
end
% Add legends
legend([h1, h2{:}], {'$$t=0$$', labels{:}},'Interpreter','latex', 'Location','best')
xlim([0,1.1])
set(gca, 'FontSize', 14)
% Add x and y labels
xlabel('$$x$$', 'Interpreter','latex', 'FontSize', 18);
ylabel('$$\phi (x,t)$$', 'Interpreter','latex', 'FontSize', 18);

% Print the figure
set(gcf, 'PaperUnits', 'Inches', 'PaperPosition', [0, 0, 9, 9*0.618]);
print(gcf,'linear_advection_semi_Lagrange_dt_inc.png','-dpng','-r300');
```

Problem 2

- prob2_b_mixed_layer_wind_turning_angle.m

```
% Wind turning in the mixed-layer

clc; clear; close all

ug = 10;         % Geostrophic u wind is fixed at 10m/s.
vg = 0;          % Geostrophic v wind is fixed at 0.
Cd = 2.0e-3;     % Nondimensional drag coefficient
f = 1.0e-4;      % Coriolis parameter

h = 300:3000;    % The depths of mixed layer
N = length(h);

V = zeros(N,2); % Vectors for components u, v
Vb= zeros(N,1); % Magnitude of mixed layer wind
angles = zeros(N,1); % The anlges between geostrophic and mixed layer winds

for j = 1:N
    kappa = Cd/f/h(j);  % Coefficient for mixed layer s/m
    u = ug;             % Initialize u, v to geostrophic values
    v = vg;
    V(j,:) = [u v];     % Vector wind
    Vb(j) = abs(V(j,1) + 1i*V(j,2));  % Wind magnitude
    % Iterate to find boundary layer solution
    % Stop if wind magnitude from this step is close enough to that from last step
    Vb_old = Vb(j);     % Original wind magnitude
    err = 1;
    while err > 1e-6
        v = vg + kappa*Vb(j)*u;
        u = ug - kappa*Vb(j)*v;
        Vb(j) = abs(u + 1i*v) ;      % New estimate for Vb, use complex number
        V(j,:) = [u v];              % New estimate for V
        err = abs(Vb(j)-Vb_old);     % Diffeerence between old and new wind magnitude
        Vb_old = Vb(j);
    end
    angles(j) = atand(v/u)-atand(vg/ug); % The angels between winds (in degree)
```

```
end

plot(h, angles, 'k') % Plot the angles against depths
xlabel('h (m)');
ylabel('Angle (degree)');
xlim([min(h), max(h)])
set(gca, 'FontSize', 14)

% Print the figures
set(gcf, 'PaperUnits', 'Inches', 'PaperPosition', [0, 0, 6, 6*0.618]);
print(gcf, 'Angles_change_with_depth_of_mixed_layer.png', '-dpng', '-r300');
```

- prob2_c_mixed_layer_wind2.m

```
% MATLAB file:   mixed_layer_wind_2.m    (2/13/02)
% Display of mixed layer solution for u and v
% given a specified horizontal geopotential field.
clc
clear all
close all
Lx = 12.e6;  Ly = 6.e6;       % horizontal domain size
cor = 1.e-4;                  % Coriolis parameter
Nx = 22; Ny =22 ;             % number of grid points in each direction
xx=linspace(-Lx/2,Lx/2,Nx);   % Nx gridpoints in x
yy=linspace( 0,Ly,Ny);        % Ny gridpoints in y
[x,y]=meshgrid(xx,yy);        % Sets matrix for grid system in x and y
%  Define the function to be contoured
h = 1000;                     % boundary layer depth

k = pi/6.e6;                  % zonal wavenumber in units of 1/m
m=pi/6.e6;                    % meridional wavenumber
U0 = 5;                       % mean zonal velocity
A = -1.0e3;                   % constant value of streamfunction
%
phi =  9800 - (U0*cor)*y +A*cos(k*x).*sin(m*y);
ug = U0-A/cor*m*cos(k*x).*cos(m*y);
vg = -A/cor*k*sin(k*x).*sin(m*y);

%% initialize wind magnitude to geostrophic
figure (1)
%subplot(2,1,1);
cs =contour(x/1000,y/1000,phi);
axis([-6000 6000 0 6000])
clabel(cs), title('geopotential')
xlabel('x (km)'), ylabel('y (km)');

%%  velocity shown in arrows
figure(2)
subplot(121)
quiver(x/1000,y/1000,ug,vg,'r')
axis([-6000 6000 0  6000])
xlabel('x (km)'), ylabel('y (km)');
title('(a) geostrophic wind')
set(gca, 'fontsize', 14)

%% u v in mixed layer
subplot(122)
[ucomp, vcomp] = mixed_layer_velocity(ug, vg, h); % get wind speed in mixed layer
quiver(x/1000, y/1000, ucomp, vcomp) % vector plot for wind
```

```
axis([-6000 6000 0  6000])
xlabel('x (km)'), ylabel('y (km)');
set(gca, 'fontsize', 14)
title('(b) mixed layer wind')
% Print the figure
set(gcf, 'PaperUnits', 'Inches', 'PaperPosition', [0, 0, 14, 9*0.618]);
print(gcf,'wind_compare.png','-dpng','-r300');

%% Vertical Velocity
figure,
dux = gradient(ucomp, mean(diff(xx))); % du/dx
[~, dvy] = gradient(vcomp, mean(diff(xx)), mean(diff(yy))); % dv/dy
w = -h.*(dux+dvy);
% plot the contourf of w
contourf(xx/1000, yy/1000, w);
h = colorbar;
ylabel(h, 'm/s')
% cs_w = contour(xx/1000, yy/1000, w);
% clabel(cs_w)
axis([-6000 6000 0  6000])
xlabel('x (km)')
ylabel('y (km)')
set(gca, 'fontsize', 14)
title('Vertical velocity')

% Print the figure
set(gcf, 'PaperUnits', 'Inches', 'PaperPosition', [0, 0, 9, 9*0.618]);
print(gcf,'vertical_velocity.png','-dpng','-r300');
```