1 Advection equations with FD

Reading

- *Spiegelman* (2004), chap. 5
- *Press et al.* (1993), sec. 19.1

1.1 The diffusion-advection (energy) equation for temperature in convection

So far, we mainly focused on the diffusion equation in a non-moving domain. This is maybe relevant for the case of a dike intrusion or for a lithosphere which remains undeformed. However, more often, we want to consider problems where material moves during the time period under consideration and takes temperature anomalies with it. An example is a plume rising through a convecting mantle. The plume is hot and hence its density is low compared to the colder mantle around it. The hot material rises with a velocity that depends on the density anomaly and viscosity (see Stokes velocity, sec. **??**). If the numerical grid remains fixed in the background, the hot temperatures should be moved to different grid points at each time step (see Figure 1 for an illustration of this effect).

More generally speaking, mantle convection is an example of a system where heat is transported by diffusion (temperature changes without moving mass, particularly important in the boundary layers) and advection (temperature changes by material transport, dominant in the interior the domain). How strongly these two effects are partitioned is indicated globally by the Rayleigh number, and locally by the Peclet number (sec. **??**).

Mathematically, the temperature equation gets an additional term for advection in a Eulerian (fixed grid) system, and the partial time derivative, $\partial/\partial t$, is replaced by the total derivative

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla, \tag{1}$$

where this is equation is for an operator, that applies to a quantity, such as temperature.

In 1-D and in the absence of heat sources, the diffusion-advection equation becomes (sec. **??**)

$$\rho c_p \left(\frac{\partial T}{\partial t} + v_x \frac{\partial T}{\partial x} \right) = \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) \tag{2}$$

or in 2-D

$$\rho c_p \left(\frac{\partial T}{\partial t} + v_x \frac{\partial T}{\partial x} + v_z \frac{\partial T}{\partial z} \right) = \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) \tag{3}$$

where v_x, v_z are velocities in x-, respectively z-direction. If k is constant, the general equation can be written as

$$\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T = \kappa \nabla^2 T. \tag{4}$$

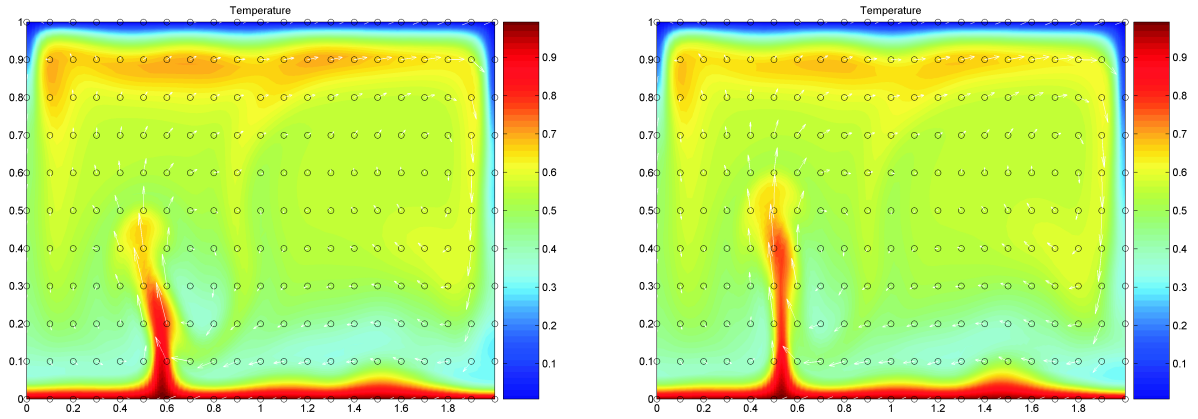

Figure 1: Snapshots of a bottom heated thermal convection model with a Rayleigh-number of 5×10^5 and constant viscosity (no internal heating). Temperature is advected through a fixed (Eulerian) grid (circles) with a velocity (arrows) that is computed with a Stokes solver.

Heat sources would lead to additional terms on the right hand side. Since temperature variations lead to buoyancy forces, the energy equation is coupled with the Stokes (conservation of momentum) equation from which velocities $\boldsymbol{v}$ can be computed to close the system needed for a convection algorithm.

Mantle convection codes typically deal with advection of a temperature field assuming that there is significant diffusion at the same time, $\kappa > 0$, and will produce non-physical artifacts in cases that are advection-dominated. One example would be if a chemical composition C is to be treated akin to T,

$$\frac{\partial C}{\partial t} + \boldsymbol{v} \cdot \nabla C = \kappa_c \nabla^2 C. \tag{5}$$

Chemical diffusivities are for mantle purposes zero, $\kappa_c \approx 0$, and special tricks are required to use field methods to solve

$$\frac{\partial C}{\partial t} + \boldsymbol{v} \cdot \nabla C = 0 \tag{6}$$

(*e.g.* *Lenardic and Kaula*, 1993), as discussed below. Often, one therefore uses tracer-based, or "particle methods" where C is assigned to virtual particles that are then advected with an ODE approach (to be solved with, *e.g.*, Runge Kutta, sec. **??**)

$$\frac{dC}{dt} = 0 \quad \text{and} \quad \frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{v} \tag{7}$$

where $\boldsymbol{x}_i$ is the location of the i-th tracer moving through the fluid. We will return to a hybrid approach (the semi-Lagrangian scheme) below, but see, *e.g.*, *Tackley and King* (2003) for a recent discussion of different tracer approaches. A related method is based

on marker chains (*e.g. van Keken et al.*, 1997), this works well if we are mainly interested in tracking a single interface between different materials with C_1 and C_2. For the latter problem, "level set" methods are also promising (*e.g. Suckale et al.*, 2010; *Samuel and Evonuk*, 2010).

1.2 Advection (transport equations)

We will return to the combined ("combo") solution of both diffusion and advection below, but for now focus on the advection part. In the absence of diffusion (*i.e.* $k, \kappa = 0$), the 1-D equations are

$$\frac{\partial T}{\partial t} + v_x \frac{\partial T}{\partial x} = 0 \tag{8}$$

and

$$\frac{\partial T}{\partial t} + v_x \frac{\partial T}{\partial x} + v_z \frac{\partial T}{\partial z} = 0. \tag{9}$$

We will now evaluate some options on how to solve these equations with a finite difference scheme on a fixed grid. Even though the equations appear simple, it is quite tricky to solve them accurately, more so than for the diffusion problem. This is particularly the case if there are large gradients in the quantity that is to be advected. If not done carefully, one can easily end up with strong numerical artifacts such as wiggles (oscillatory artifacts) and numerical diffusion (artificial smoothing of the solution).

1.2.1 FTCS method

In 1-D, the simplest way to discretize eq. (8) is by employing a central difference scheme in space, and go forward in time (another example of a forward-time, central space, FTCS, scheme):

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = -v_{x,i} \frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x}, \tag{10}$$

where $v_{x,i}$ is the v_x velocity at location i.

Exercise 1 We will consider a 1-D problem, with constant v_x velocity in which an exponential pulse of temperature is getting advected along the x axis (see Figure 2 and `exercise_1_ftcs.m`).

- Program the FTCS method in the code of Figure 2 and watch what happens.
- Change the sign of the velocity.
- Change the time step and grid spacing and compute the non-dimensional parameter $|v_x|\Delta t/\Delta x$.
- When do unstable results occur? Put differently, can you find a Δt small enough to avoid blow-up?

```
%
% FTCS advection schem
%
clear all

nx      =   201;
W       =   40;     % width of domain
Vel     =   -4;      % velocity
sigma   =   1;
Ampl    =   2;
nt      =   500;    % number of timesteps
dt      =   1e-2;   % timestep
dx      =   W/(nx-1);
x       =   0:dx:W;
% Initial Gaussian T-profile
xc      =   20;
T       =   Ampl*exp(-(x-xc).^2/sigma^2);
% Velocity
Vx      =   ones(1,nx)*Vel;
abs(Vel)*dt/dx
cfac = dt/(2*dx);
% Central finite difference discretization
for itime=1:nt
    % central fin. diff
    for ix=2:nx-1
      Tnew(ix) = ???
    end
    % BCs
    Tnew(1)     =   ???
    Tnew(nx)    =   ???
    % Update Solution & time incrememement
    T           =   Tnew;
    time        =   itime*dt;
    % Analytical solution for this case
    T_anal      =   ???
    figure(1),clf, plot(x,T,x,T_anal), ...
legend('Numerical','Analytical')
    xlabel('x')
    ylabel('temperature')
    drawnow
end
```

Figure 2: MATLAB script to be used with FTCS exercise 1.

As you can see from the exercise, the FTCS method does not work so well . . . In fact, it is a nice example of a scheme that looks logical on paper, but looks can be deceiving. The FTCS method is *unconditionally unstable*, blows up for any Δt, as can be shown by *von Neumann stability analysis* (*cf.* chap 5 of *Spiegelman*, 2004). The instability is related to the fact that this scheme produces negative diffusion, which is numerically unstable.

1.2.2 Lax method

The Lax approach consists of replacing the T_i^n in the time-derivative of eq. (10) with $(T_{i+1}^n + T_{i-1}^n)/2$. The resulting equation is

$$\frac{T_i^{n+1} - (T_{i+1}^n + T_{i-1}^n)/2}{\Delta t} = -v_{x,i}\frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x} \tag{11}$$

Exercise 2

- Program the Lax method by modifying the script of the last exercise.
- Try different velocities and Δt settings and compute the *Courant number*, α, which is

given by the following equation:

$$\alpha = \frac{v_x \Delta t}{\Delta x} \tag{12}$$

- Is the numerical scheme stable for all Courant numbers?
- What is the physical meaning of α? What happens for $\alpha = 1$ and why?
- *Bonus question*: Implement a generalized Galerkin-Lax-Wendroff method using the following equation:

$$\left[M_x - \frac{\alpha^2}{(\Delta x)^2} \frac{\partial^2}{\partial x^2} \right] (T_i^{n+1} - T_i^n) + \alpha \Delta x \frac{\partial}{\partial x} T_i^n - \frac{\alpha^2 (\Delta x)^2}{2} \frac{\partial}{\partial x} T_i^n = 0 \tag{13}$$

where $M_x = \{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\}$ and spatial derivatives are discretized using second order central differences:

$$\begin{aligned} &\frac{1}{6} \left(1 - c^2 (\Delta x)^2\right) (T_{i+1}^{n+1} - T_{i-1}^{n+1}) + \frac{2}{3} \left(1 + c^2 (\Delta x)^2\right) T_i^{n+1} \\ &\quad = \left[\frac{1}{6} - \frac{\alpha}{2} + \frac{\alpha^2 (\Delta x)^2}{3}\right] T_{i+1}^n + \frac{2}{3} \left(1 - \alpha^2 (\Delta x)^2\right) T_i^n + \left[\frac{1}{6} - \frac{\alpha}{2} + \frac{\alpha^2 (\Delta x)^2}{3}\right] T_{i-1}^n \end{aligned} \tag{14}$$

This formulation gives us much better accuracy ($O(\Delta t^2, (\Delta x)^2)$ by using a higher order discretization in both time and space. But what is its stability range in terms of Courant number? Notice the difference in terms of artificial diffusion, and oscillations with respect to the simple Lax method.

As you saw from exercise 2, the Lax method does not blow up, but does have a lot of numerical diffusion for $\alpha \neq 1$ (which is hard to attain for realistic problems, as v will vary in space and time). In fact, the Lax criterion stabilized the discretized advection equation by adding some artificial diffusion. So, it's an improvement but it's far from perfect, since you may now lose the plumes of Figure 1 around mid-mantle purely due to numerical diffusion. As for the case of the implicit versus explicit solution of the diffusion equation, you see that there are trade-offs between stability and accuracy. There is no free lunch, and numerical modeling is also a bit of an art.

The stability requirement

$$\alpha = \frac{|V| \Delta t}{\Delta x} \leq 1 \tag{15}$$

is called the *Courant criterion* (Figure 3).

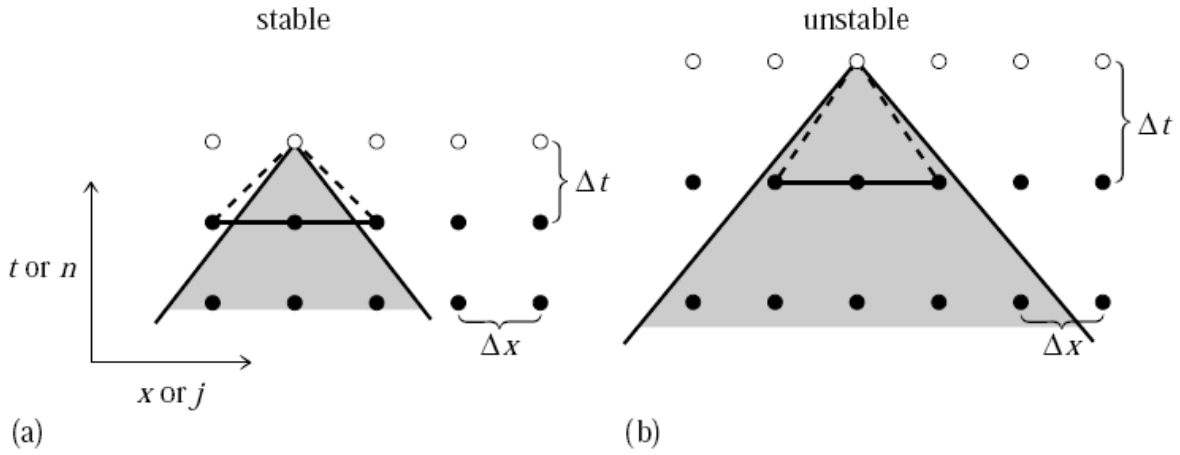

Figure 19.1.3. Courant condition for stability of a differencing scheme. The solution of a hyperbolic problem at a point depends on information within some domain of dependency to the past, shown here shaded. The differencing scheme (19.1.15) has its own domain of dependency determined by the choice of points on one time slice (shown as connected solid dots) whose values are used in determining a new point (shown connected by dashed lines). A differencing scheme is Courant stable if the differencing domain of dependency is larger than that of the PDEs, as in (a), and unstable if the relationship is the reverse, as in (b). For more complicated differencing schemes, the domain of dependency might not be determined simply by the outermost points.

Figure 3: Illustration of the Courant criterion (from *Press et al.*, 1993, chap 19.1).

1.2.3 Streamline upwind scheme

A popular scheme is the so-called (streamline) upwind approach (Figure 4a). Here, the spatial finite difference scheme depends on the sign of the velocity:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = -v_{x,i} \begin{cases} \frac{T_i^n - T_{i-1}^n}{\Delta x}, \text{ if } v_{x,i} > 0 \\ \frac{T_{i+1}^n - T_i^n}{\Delta x}, \text{ if } v_{x,i} < 0 \end{cases} \tag{16}$$

Note that we have replaced central with forward or backward derivatives, depending on the flow direction. The idea is that the flux into the local cell at x_i will only depend on the gradient of temperature in the direction "upstream", *i.e.* where the inflowing velocity is coming from.

Exercise 3

- Program the upwind scheme method.
- Try different velocity distributions (not just constant) and compute the Courant numbers α.
- Is the numerical scheme stable for all Courant numbers?

The upwind scheme also suffers from numerical diffusion, and it is only first order accurate in space. For some applications, particularly if there's also diffusion, it might just be good enough because the simple trick of doing FD forward or backward is closer to the underlying physics of transport than, say, FTCS. There are some mantle convection codes that use streamline upwind schemes.

So far, we employed explicit discretizations. You're probably wondering whether implicit discretizations will save us again this time. Bad news: they are not well-suited for this type of problem (try it and see). Implicit schemes behave like parabolic partial differential equations (*e.g.* the diffusion equation) in that a perturbation at node (j, n) will affect the solution at all nodes at time level $n + 1$. With hyperbolic PDEs like the advection equation or the wave equation, disturbances travel at a finite speed (the speed of the material displacement) and will not affect all nodes at time level $n + 1$. So we have to come up with something else.

1.2.4 Modified Crank-Nicolson

One approach to solving the advection equation is the previously introduced Crank-Nicolson semi-implicit scheme. Here we modify it slightly by introducing a general mass operator $M_x = \{\delta, 1 - 2\,\delta, \delta\}$.

$$M_x \left[\frac{T_i^{n+1} - T_i^n}{\Delta t} + \frac{v}{2} \frac{(T_{i+1}^n - T_{i-1}^n) + (T_{i+1}^{n+1} - T_{i-1}^{n+1})}{2\Delta t} \right] = 0 \tag{17}$$

Setting the mass operator to $\delta = 0$ gives us the previously seen Crank-Nicolson semi-implicit finite difference discretization, while setting $\delta = \frac{1}{6}$ gives us the finite element formulation. Below is eq. (17) written out with $\delta = \frac{1}{6}$.

$$\begin{aligned} \left[\frac{1}{6} - \frac{1}{4}\left(v\frac{\Delta t}{\Delta x}\right)\right] T_{i-1}^{n+1} - \left(1 - \frac{1}{3}\right) T_i^{n+1} + \left[\frac{1}{6} + \frac{1}{4}\left(v\frac{\Delta t}{\Delta x}\right)\right] T_{i+1}^{n+1} \\ = \left[\frac{1}{6} + \frac{1}{4}\left(v\frac{\Delta t}{\Delta x}\right)\right] T_{i-1}^{n} + \left(1 - \frac{1}{3}\right) T_i^{n} + \left[\frac{1}{6} - \frac{1}{4}\left(v\frac{\Delta t}{\Delta x}\right)\right] T_{i+1}^{n} \end{aligned} \tag{18}$$

The finite element Crank-Nicolson advection scheme is stable for $\alpha \leq 1$ and provides an improvement over previous schemes in that it is accurate to $O(\Delta t, (\Delta x)^3)$. This allows us to reduce the number of grid points to reach the same accuracy as the other schemes presented, as long as Δt is kept small enough.

1.2.5 Staggered leapfrog

The explicit discretizations discussed so far were second order accurate in time, but only first order in space. We can also come up with a scheme that is second order in time and

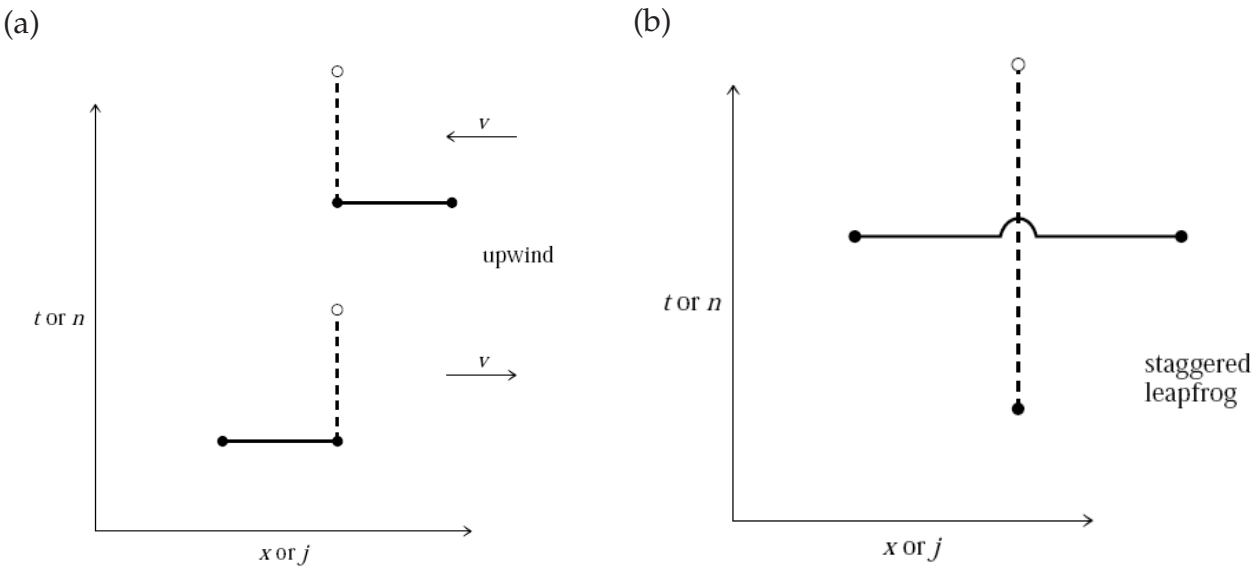

Figure 4: Illustration of the upwind (a) and leapfrog (b) schemes (from *Press et al.*, 1993, chap 19.1).

space

$$\frac{T_i^{n+1} - T_i^{n-1}}{2\Delta t} = -v_{x,i}\frac{T_{i+1}^n - T_{i-1}^n}{2\Delta x}, \tag{19}$$

called *staggered leapfrog* because of the way it's centered in shifted space-time (Figure 4b). The computational inconvenience in this scheme is that two time steps have to be stored, T^{n-1} and T^n.

Exercise 4

- Program the staggered leapfrog method (assume that at the first time step $T^{n-1} = T^n$).
- Try with different values of the Courant number α and compare the accuracy and stability of the different methods.
- Also, make the width of the Gaussian curve smaller.
- *Bonus:* Also program the two formulations of the Crank-Nicolson method with $\delta = 0$ and $\delta = \frac{1}{6}$.

The staggered leapfrog method works quite well regarding the amplitude and transport phase as long as α is close to one. If, however, $\alpha \ll 1$ and the length scale of the to-be-transported quantity is small compared to the number of grid points (*e.g.* we have a thin plume), numerical oscillations again occur (those are due to the lack of communication between cells, which can be remedied by artificial diffusion). The conditions where

leapfrog does not work well are typically the case in mantle convection simulations (*cf.* Figure 1). Onward ever, backward never.

Similarly, the Crank-Nicolson method works well for $v \leq 0.1$ and $\alpha \leq 0.1$, and eliminates the staggered problem. But what happens for $\alpha \geq 0.1$? What about the finite element formulation? What about computational time? Is Crank-Nicolson's increased accuracy worth the extra work? Is it well-suited for mantle convection problems?

1.2.6 MPDATA

This is a technique that is frequently applied in (older) mantle convection codes. The idea is based on *Smolarkiewicz* (1983) and represents an attempt to improve on the upwind scheme by adding some anti-diffusion, which requires iterative corrections. The results are pretty good, but MPDATA is somewhat more complicated to implement. Moreover we still have a restriction on the time step (given by the Courant criterion), for details see *Spiegelman* (2004).

1.2.7 Semi-Lagrangian approaches

What we want is a scheme that is stable, has only small numerical diffusion and is not limited by the Courant criterion. A contender is the semi-Lagrangian method, which is often used for climate modeling. The method is related to tracer-based advection by solving ODEs and has little to do with the finite difference schemes we discussed so far. Since this scheme could be the one that is most important for practical purposes we will go in more detail. It has few drawbacks, one being that it is not necessarily flux conserving.

Basic idea The basic idea of the semi-Lagrangian method is illustrated in Figure 5A and is given by the following, simplified scheme. Instead of allowing the numerical scheme to transport noise in from unknown regions, the semi-Lagrangian method uses transport by going back one (*e.g.* Euler) step.

For each point i at x_i and time t_n:

1. Assume that the future velocity $v_x(t_{n+1}, x_i)$ at x_i is known. Under the assumption that the velocity at the old time step is close to the future velocity ($v_x(t_{n+1}, x_i) \approx v_x(t_n, x_i)$) *and* that velocity does not vary spatially ($v_x(t_n, x_{i-1}) \approx v_x(t_n, x_i) \approx v_x(t_n, x_{i+1})$), we can compute the location $\boldsymbol{X}$ where the particle came from by $\boldsymbol{X} = x_i - \Delta t v_x(t_{n+1}, x_i)$.

2. Interpolate temperature from grid points $\{x_i\}$ to the location X at time t_n, $T(t_n, X)$. For example, use cubic interpolation (in MATLAB use the command `interp1(x,T, X, 'cubic')` for interpolation, where `x` is supposed to be the vector that holds the $\{x_i\}$).
   Note 1: Be careful with interpolation. For smooth functions, polynomial interpolation, say of cubic order, is often a good idea. However, at edges, or if the function is otherwise discontinuous, "ringing", *i.e.* large, wiggly excursions, can occur. Linear, or spline, interpolation may be preferred.

Note 2: Most of the Matlab interpolation functions will by default not extrapolate outside the

$$[min(x_i), max(x_i)]$$

range and return `NaN` (not a number). If extrapolation is desired, `'extrap'` needs to be set as an option when calling the `interp1` function.

3. Assume that $T(t_{n+1}, x_i) = T(t_n, X)$, *i.e.* temperature has been transported (along "characteristics") without any modification (*e.g.* due to diffusion).

This scheme assumes that no heat-sources were active during the advection of T from $T(t_n, X)$ to $T(t_{n+1}, x_i)$. If heat sources are present *and* are spatially variable, some extra care needs to be taken (*Spiegelman*, 2004, sec. 5.6.1).

Exercise 5

- Program the semi-Lagrangian advection scheme illustrated in Figure 5A. Is there a Courant criterion for stability?

Some improvements The algorithm described in Figure 5A illustrates the basic idea of the semi-Lagrangian scheme. However, it has two problems. First it assumes that velocity is spatially constant (which is clearly not the case in mantle convection simulations). Second, it assumes that velocity does not change between time n and $n+1$. We can overcome both problems by using a more accurate time stepping algorithm (see the ODE section). An example is an iterative mid-point scheme which works as follows (*cf.* Figure 5B):

For each point i

1. Use the velocity $v_x(t_{n+1}, x_i)$ to compute the location X' at time $t_{n+1/2}$ (*i.e.* take half a time step backward in time).
2. Find the velocity at the location X' at half time step $t_{n+1/2}$. Assume that the velocity at the half time step can be computed as

$$v_x(t_{n+1/2}, x_i) = \frac{v_x(t_{n+1}, x_i) + v_x(t_n, x_i)}{2}. \tag{20}$$

   Use linear interpolation for the spatial interpolation of velocity $v_x(t_{n+1/2}, X')$.
3. Go back to point 1, but use the velocity $v_x(t_{n+1/2}, X')$ instead of $v_x(t_{n+1}, x_i)$ to move the point $x_i(t_{n+1})$ backward in time. Repeat this process a number of times (*e.g.* 5 times). This gives a fairly accurate centered velocity.
4. Compute the location X at t_n with the centered velocity $X = x_i - \Delta t v_x(t_{n+1/2}, X')$.
5. Use cubic interpolation to find the temperature at point X as before.

Other ODE-motivated methods such as 4th order Runge Kutta are also possible (but take a bit more work). Note that the various velocity interpolation and iteration schemes add overhead that is, however, typically made up for by not needing to obey the Courant criterion.

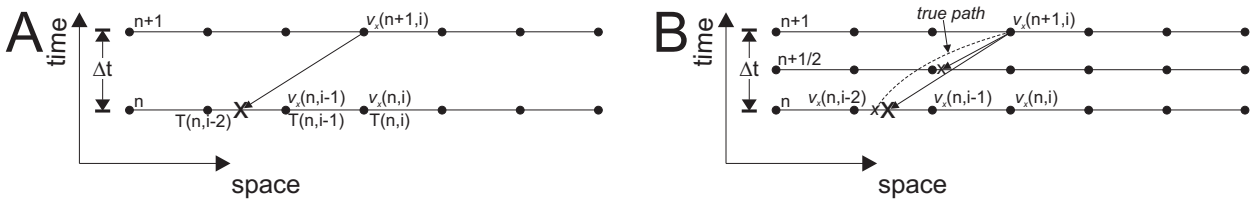

Figure 5: Basics of the semi-Lagrangian method. See text for explanation.

Exercise 6

- Program the semi-Lagrangian advection scheme with the centered midpoint method as illustrated in Figure 5B (*cf. Spiegelman*, 2004, p. 67).

  Some care has to be taken if point $\boldsymbol{X}$ is outside of the computational domain, since MATLAB will return `NaN` for the velocity (or temperature) of this point. If no extrapolation is desired, use the velocity $v_x(t_{n+1}, x_i)$ in this case. A pseudo-code is given by

```
if isnan(Velocity)
Velocity = Vx(i)
end
```

1.2.8 2D advection example

The semi-Lagrangian method is likely a good, general advection algorithm (except in the case of pseudo spectral methods), so this is the one we will implement in 2D.

Assume that velocity is given by

$$v_x(x,z) = z \tag{21}$$
$$v_z(x,z) = -x \tag{22}$$

Moreover, assume that the initial temperature distribution is Gaussian and given by

$$T(x,z) = 2\exp\left(\frac{(x+0.25)^2+z^2}{0.1^2}\right) \tag{23}$$

with $x \in [-0.5, 0.5]$, $z \in [-0.5, 0.5]$.

Exercise 7

- Program advection in 2D using the semi-Lagrangian advection scheme with the centered midpoint method.

  Use the MATLAB routine `interp2` for interpolation and employ linear interpolation for velocity and cubic interpolation for temperature. A MATLAB script that will get you started is shown on Figure 6 (`semi_lagrangian_2D_1.m`).

```
% semi_lagrangian_2D: 2D semi-lagrangian  with center midpoint time stepping method
%

clear all

W       =   40;     % width of domain
sigma   =   .1;
Ampl    =   2;
nt      =   500;    % number of timesteps
dt      =   5e-1;

% Initial grid and velocity
[x,z]   =   meshgrid(-0.5:.025:0.5,-0.5:.025:0.5);
nz      =   size(x,1);
nx      =   size(x,2);

% Initial gaussian T-profile
T       =   Ampl*exp(-((x+0.25).^2+z.^2)/sigma^2);

% Velocity
Vx      =   z;
Vz      =   -x;

for itime=1:nt

    Vx_n    = Vx;                   %   Velocity at time=n
    Vx_n1   = Vx;                   %   Velocity at time=n+1
%    Vx_n1_2 = ??;                   %   Velocity at time=n+1/2
%    Vz_n    = ??;                   %   Velocity at time=n
%    Vz_n1   = ??;                   %   Velocity at time=n+1
%    Vz_n1_2 = ??;     %   Velocity at time=n+1/2
    Tnew    =   zeros(size(T));
    for ix=2:nx-1
        for iz=2:nz-1

            Vx_cen = Vx(iz,ix);
            Vz_cen = Vz(iz,ix);
   %         for ??
%                X =?
%                Z = ?

               %linear interpolation of velocity
%                Vx_cen = interp2(x,z,?,?, ?, 'linear');
%                Vz_cen = interp2(x,z,?,?, ?, 'linear');

                if isnan(Vx_cen)
                    Vx_cen =   Vx(iz,ix);
                end
                if isnan(Vz_cen)
                    Vz_cen =   Vz(iz,ix);
                end


%            end
%            X = ?;
%            Z = ?;

            % Interpolate temperature on X
%            T_X = interp2(x,z,?,?,?, 'cubic');
            if isnan(T_X)
                T_X = T(iz,ix);
            end

            Tnew(iz,ix) = T_X;
        end

    end

    Tnew(1,:)   =   T(1,:);
    Tnew(nx,:)  =   T(nx,:);
    Tnew(:,1)   =   T(:,1);
    Tnew(:,nx)  =   T(:,nx);

    T           =   Tnew;
    time        =   itime*dt;


    figure(1),clf
    pcolor(x,z,T), shading interp, hold on, colorbar
    contour(x,z,T,[.1:.1:2],'k'),
    hold on, quiver(x,z,Vx,Vz,'w')
    axis equal, axis tight
    drawnow
    pause
end
```

Figure 6: MATLAB script to be used with exercise 7.

1.3 Advection and diffusion: operator splitting

In geodynamics, we often want to solve the coupled advection-diffusion equation, which is given by eq. (2) in 1-D and by eq. (3) in 2-D. We can solve this pretty easily by taking the equation apart and by computing the advection part separately from the diffusion part. This is called operator-splitting, and what is done in 1-D is, for example: First solve the advection equation

$$\frac{\tilde{T}^{n+1} - T^n}{\Delta t} + v_x \frac{\partial T}{\partial x} = 0 \tag{24}$$

for example by using a semi-Lagrangian advection scheme. Then solve the diffusion equation

$$\rho c_p \frac{\partial \tilde{T}}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial \tilde{T}}{\partial x}\right) + Q. \tag{25}$$

For this, we assumed that Q is spatially constant; if not, one should consider to slightly improve the advection scheme by introducing source terms. A good general method would be to combine Crank-Nicolson for diffusion with a semi-Lagrangian solver for advection (*Spiegelman*, 2004, sec. 7.2), but we will try something simpler first:

Exercise 8

- Program diffusion-advection in 2D using the semi-Lagrangian advection scheme coupled with an implicit 2D diffusion code (from last section's exercise). Base your code on the script of Figure 6.

Bibliography

Lenardic, A., and W. M. Kaula (1993), A numerical treatment of geodynamic viscous flow problems involving the advection of material interfaces, *J. Geophys. Res.*, *98*, 8243–8260.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1993), *Numerical Recipes in C: The Art of Scientific Computing*, 2 ed., Cambridge University Press, Cambridge.

Samuel, H., and M. Evonuk (2010), Modeling advection in geophysical flows with particle level sets, *Geochem., Geophys., Geosys., in press*, doi:10.1029/2010GC003081.

Smolarkiewicz, P. K. (1983), A simple positive definite advection scheme with small implicit diffusion, *Mon. Weather Rev.*, *111*, 479–486.

Spiegelman, M. (2004), *Myths and Methods in Modeling*, Columbia University Course Lecture Notes, available online at `http://www.ldeo.columbia.edu/~mspieg/mmm/course.pdf`, accessed 06/2006.

Suckale, J., J.-C. Nave, and B. H. Hager (2010), It takes three to tango 1: Simulating buoyancy-driven flow in the presence of large viscosity contrasts, *J. Geophys. Res.*, *in press*, doi:10.1029/2009JB006916.

Tackley, P. J., and S. D. King (2003), Testing the tracer ratio method for modeling active compositional fields in mantle convection simulations, *Geochem., Geophys., Geosys.*, *4*, doi:10.1029/2001GC000214.

van Keken, P. E., S. King, H. Schmeling, U. Christensen, D. Neumeister, and M.-P. Doin (1997), A comparison of methods for the modeling of thermochemical convection, *J. Geophys. Res.*, *102*, 22,477–22,495.