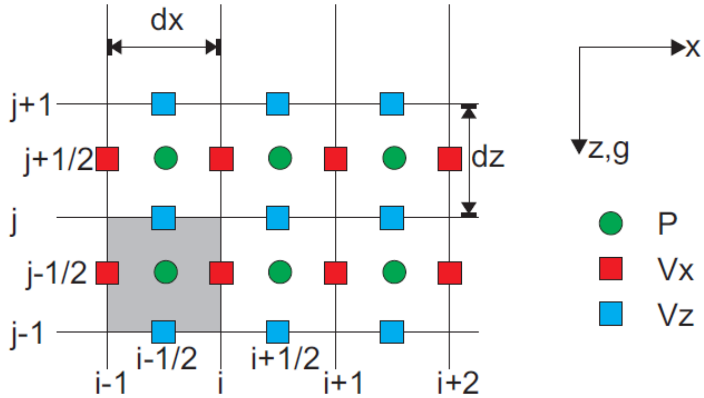

Figure 1: Staggered grid definition. Properties such as viscosity and density inside a control volume (gray) are assumed to be constant. Moreover, a constant grid spacing in x and z-direction is assumed.

1 2D Stokes equations on a staggered grid using primitive variables

1.1 Introduction

The basis of basically all mantle convection and lithospheric dynamics codes are the so-called Stokes equations for slowly moving viscous fluids. These equations describe the balance between buoyancy forces (*e.g.* due to temperature variations in the fluid) and viscous drag (sec. **??**). Here, we will describe the governing equations. There are several ways to solve those equations, and the goal of this project is to use a staggered finite difference approach in primitive variables.

For this, we solve the governing equations for $\underline{v} = \{v_x; v_z\}$ (velocities) and p (pressure). Staggered finite differences means that the different unknowns v_x, v_z, p are defined at physically different grid points. The main challenges of this project are, 1), having several variables instead of only one (*e.g.* temperature), and, 2), to do the bookkeeping for the present case that the variables are at different grid points. (While the governing equations are different, those computational challenges are similar to those arising in the staggered grid, finite difference approach for wave propagation discussed in sec. **??**.)

1.2 Governing equations

It is assumed that the rheology is incompressible and that the rheology is Newtonian viscous, *i.e.* $\underline{\underline{\sigma}} = 2\mu\underline{\underline{\dot{\varepsilon}}}$ with μ no function of $\underline{\underline{\dot{\varepsilon}}}$, where $\underline{\underline{\sigma}}$ is the stress tensor, μ viscosity, and $\underline{\underline{\dot{\varepsilon}}}$ strain-rate tensor. In this case, the governing equations in 2D (x and z) are (see sec. **??**):

$$
\begin{aligned}
\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} &= 0 && (1)\\
\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xz}}{\partial z} &= 0 && (2)\\
\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{zz}}{\partial z} - \rho g &= 0 && (3)\\
\sigma_{xx} &= -p + 2\mu \frac{\partial v_x}{\partial x} && (4)\\
\sigma_{zz} &= -p + 2\mu \frac{\partial v_z}{\partial z} && (5)\\
\sigma_{xz} &= \mu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right), && (6)
\end{aligned}
$$

where ρ is density and $\underline{g} = \{0, g\}$ the gravitational acceleration. The density is where these continuum and force balance equations (eqs. 1 to 3) couple to the the energy equation, *e.g.* the diffusion and advection of temperature for mantle convection, discussed in the previous sections.

It has been suggested that a particularly nice way to solve these equations is to use a staggered grid (more about this later) and to keep as variables v_x, v_z and p (*Gerya and Yuen*, 2003; *Gerya*, 2009).[1] Since there are three variables, we need three equations. Substituting eqs. (4)-(6) into eq. (2) and eq. (3) leads to:

$$
\begin{aligned}
\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} &= \frac{p}{\gamma} && (7)\\
-\frac{\partial P}{\partial x} + 2\frac{\partial}{\partial x}\left(\mu \frac{\partial v_x}{\partial x}\right) + \frac{\partial}{\partial z}\left(\mu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right)\right) &= 0 && (8)\\
-\frac{\partial P}{\partial z} + 2\frac{\partial}{\partial z}\left(\mu \frac{\partial v_z}{\partial z}\right) + \frac{\partial}{\partial x}\left(\mu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right)\right) &= \rho g && (9)
\end{aligned}
$$

Note that we added the term $\frac{P}{\gamma}$ to the incompressibility equations. This is a "trick" called the penalty method, which ensures that the system of equations does not become ill-posed. For this to work, γ should be sufficiently large ($\sim 10^4$ or so), so that the condition of incompressibility (conservation of mass, eq. 1) is approximately satisfied.

1.3 Exercise

a) Discretize eqs. (7)-(9) on a staggered grid as shown on Figure 1.

b) A MATLAB subroutine is shown on Figure 2. The subroutine sets up the grid, the node numbering and discretizes the incompressibility equations.

[1]For a comparison of different finite difference approaches, see *Deubelbeiss and Kaus* (2008), for example.

```
% Solve the 2D Stokes equations on a staggered grid, using the Vx,Vz,P formulation.
clear
% Material properties
%               phase #1  phase #2
mu_vec      =   [1      1   ];
rho_vec     =   [1      2   ];
% Input parameters
Nx          =   20;
Nz          =   .9*Nx;
W           =   1;
H           =   1;
g           =   1;
% Setup the interface
x_int       =   0:.01:W;
z_int       =   cos(x_int*2*pi/W)*1e-2 - 0.5;
% Setup the grids---------------------------------------------------------
dz          =   H/(Nz-1);
dx          =   W/(Nx-1);
[X2d,Z2d]   =   meshgrid(0:dx:W,-H:dz:0);
XVx         =   [X2d(2:end,:) + X2d(1:end-1,:)]/2; % Vx
ZVx         =   [Z2d(2:end,:) + Z2d(1:end-1,:)]/2;
XVz         =   [X2d(:,2:end) + X2d(:,1:end-1)]/2; % Vz
ZVz         =   [Z2d(:,2:end) + Z2d(:,1:end-1)]/2;
XP          =   [X2d(2:end,2:end) + X2d(1:end-1,1:end-1)]/2; % p
ZP          =   [Z2d(2:end,2:end) + Z2d(1:end-1,1:end-1)]/2;
% Compute material properties from interface, properties are computed in the center of a control volume
Rho              =   ones(Nz-1,Nx-1)*rho_vec(2);
Mu               =   ones(Nz-1,Nx-1)*mu_vec(2);
z_int_intp       =   interp1(x_int,z_int,XP(1,:));
for ix = 1:length(z_int_intp)
    ind          =   find(ZVz(:,1)<z_int_intp(ix));
    Rho(ind(1:end-1),ix) = mu_vec(1);
    Mu(ind(1:end-1),ix)  = rho_vec(1);
    fac          =  (z_int_intp(ix) - ZVz(ind(end),1))/dz;
    Rho(ind(end),ix)  =  fac*rho_vec(1) + (1-fac)*rho_vec(2);
    Mu(ind(end),ix)   =  fac*mu_vec( 2) + (1-fac)*mu_vec( 2);
end
% Setup numbering scheme-------------------------------------------------
Number_Phase      =   zeros(Nz + Nz-1, Nx + Nx-1);  % Create the general numbering scheme
Number_ind        =   zeros(Nz + Nz-1, Nx + Nx-1);  % Create the general numbering scheme
Number_Vx         =   zeros(Nz-1,Nx  );Number_Vz         =   zeros(Nz  ,Nx-1);
Number_P          =   zeros(Nz-1,Nx-1);
for ix=1:2:Nx+Nx-1, for iz=2:2:Nz+Nz-1, Number_Phase(iz,ix) = 1; end; end % Vx equations
for ix=2:2:Nx+Nx-1, for iz=1:2:Nz+Nz-1, Number_Phase(iz,ix) = 2; end; end % Vz equations
for ix=2:2:Nx+Nx-1, for iz=2:2:Nz+Nz-1, Number_Phase(iz,ix) = 3; end; end % P  equations
num               =     1;
for ix=1:size(Number_Phase,2)
    for iz=1:size(Number_Phase,1)
        if Number_Phase(iz,ix)~=0
            Number_ind(iz,ix)   =   num;
            num                 =   num+1;
        end
    end
end
num_eqns         =   num-1;
ind_Vx           =   find(Number_Phase==1);  Number_Vx(find(Number_Vx==0)) = Number_ind(ind_Vx);
ind_Vz           =   find(Number_Phase==2);  Number_Vz(find(Number_Vz==0)) = Number_ind(ind_Vz);
ind_P            =   find(Number_Phase==3);  Number_P (find(Number_P ==0)) = Number_ind(ind_P );
% Setup the stiffness matrix
A                      =   sparse(num_eqns,num_eqns);
Rhs_vec                =   zeros(num_eqns,1);
% Setup the incompressibility equations------------------------------------
ind_list = [];ind_val  = [];
[ind_list,ind_val] = Add_coeffs(ind_list,ind_val, Number_Vx(:,2:end  ),  ( 1/dx));%dVx/dx
[ind_list,ind_val] = Add_coeffs(ind_list,ind_val, Number_Vx(:,1:end-1),  (-1/dx));
[ind_list,ind_val] = Add_coeffs(ind_list,ind_val, Number_Vz(2:end,:  ),  ( 1/dz));%dVz/dz
[ind_list,ind_val] = Add_coeffs(ind_list,ind_val, Number_Vz(1:end-1,:),  (-1/dz));
% Add local equations to global matrix
for i=1:size(ind_list,2)
    A = A + sparse([1:size(ind_list,1)].',ind_list(:,i),ind_val(:,i),num_eqns,num_eqns);
end
num_incomp = length(ind_list);
% Perform testing of the system of equation, setup some given matrixes
mu  = mu_vec(1);
Vx                        =         cos(XVx).*sin(ZVx);
Vz                        =        -sin(XVz).*cos(ZVz);
P                         =    2*mu*sin(XP ).*sin(ZP );
C                         =   zeros(num_eqns,1);
C(Number_Vx(:))           =   Vx(:);
C(Number_Vz(:))           =   Vz(:);
C(Number_P(:))            =   P(:);
Rhs                       =   A*C;
% Check whether the compressibility equations are implemented correctly
max(abs(Rhs(1:num_incomp)))
```

Figure 2: MATLAB script `Staggered_Stokes.m` that sets up numbering, matrix $\underline{\underline{A}}$ and that solves the incompressibility equations.

```
function [ind_list,ind_val] = Add_coeffs(ind_list,ind_val,ind_add,val_add)% Add coefficients to an array
if (length(val_add(:))==1)
   val_add  =   ones(size(ind_add))*val_add;
end
ind_list    =   [ind_list, ind_add(:)];ind_val     =   [ind_val , val_add(:)];
```

Figure 3: MATLAB script `Add_coeffs.m`, used by `Staggered_Stokes.m`.

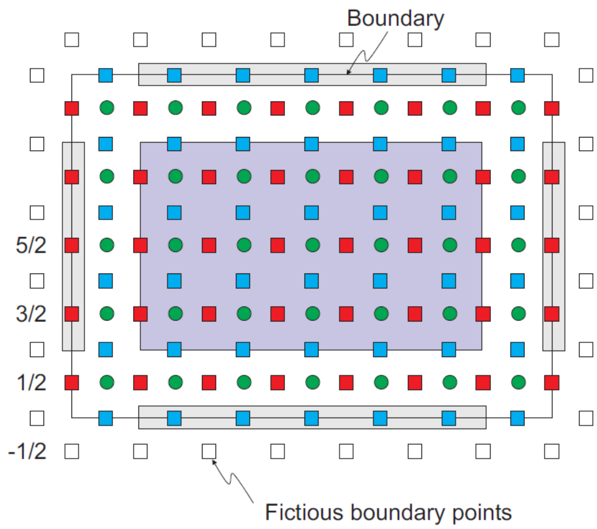

Figure 4: Staggered grid definition with the boundary points. Within the purple domain, the finite difference scheme for center points can be applied. At the boundaries, we have to apply a special finite difference scheme which employ fictious boundary nodes.

Add the discretization of the force balance equations (including the effects of gravity) into the equation matrix $\underline{\underline{A}}$. Assume that the viscosity is constant and $\mu = 1$ in a first step, but density is variable.

An example is given in how to verify that the incompressibility equation is incorporated correctly. This is done by assuming a given (sinusoidal) function for, let's say, v_x (*e.g.* $v_x = \cos(\omega x)\cos(\omega z)$). From the incompressibility equation (eq. 1) a solution for v_z than follows. By setting those solutions in the $\underline{c}$ vector, we can compute $\underline{\underline{A}}\,\underline{c}$ and verify that $\underline{rhs}$ for those equations is indeed zero.

c) Add free-slip boundary conditions on all sides (which means $v_z = 0, \sigma_{xz} = 0$ on the lower and upper boundaries and $\sigma_{xz} = 0, v_x = 0$ on the side boundaries). Use fictious boundary points to incorporate the σ_{xz} boundary conditions.

d) Assume a model domain $x = [0;1], \quad z = [0;1]$, and assume that the density below $z = 0.1\cos(2\pi x) + 0.5$ is 1, whereas the density above it is 2. Compute the velocity and pressure, and plot the velocity vectors.

e) Write the code for the case of variable viscosity (which is relevant for the Earth since rock properties are a strong function of temperature).

Bibliography

Deubelbeiss, Y., and B. J. P. Kaus (2008), Comparison of Eulerian and Lagrangian numerical techniques for the Stokes equations in the presence of strongly varying viscosity, *Phys. Earth Planet. Inter.*, *171*, 92–111.

Gerya, T. (2009), *Introduction to Numerical Geodynamic Modelling*, Cambridge University Press, Cambridge UK.

Gerya, T. V., and D. Yuen (2003), Characteristics-based marker-in-cell method with conservative finite-differences schemes for modeling geological flows with strongly variable transport properties, *Phys. Earth Planet. Inter.*, *140*, 293–318.