Numerical Simulation of Snow Transport Above Alpine Terrain

Author: Marc D. Ryser (*) *marcdaniel.ryser@epfl.ch*
Supervisor: Dr. M. Lehning (**) *lehning@slf.ch*

(*) EPFL, Swiss Federal Institute of Technology, Lausanne (CH), Department of Physics
(**) SLF, Swiss Federal Institute for Snow and Avalanche Research, Davos (CH)

Autumn 2004

Abstract

Wind transport of snow has a major impact on the avalanche hazard as well as on the ecology and hydrology in mountainous environments. An important feature of future avalanche forecast models is therefore the numerical simulation of snow-transport by turbulent wind fields above alpine terrain. One possible approach to this problem is the modeling of the snow-air mixture as a real fluid and consequently the description of its mass balance by means of an unsteady partial differential equation (PDE) of the advection-diffusion type. The aim of this project was to find an apt numerical method for solving the arising PDE. For certain reasons we have decided to concentrate on the large class of finite element methods (FEM), more precisely on Galerkin methods. The fact of the matter is that the problem to solve is highly advection-dominated and consequently a standard Galerkin (or Bubnov-Galerkin) approach performs rather unsatisfactorily as will be shown. Among the vast variety of stabilized versions of the Galerkin method we finally selected the SUPG (Streamline-Upwind/Petrov-Galerkin) scheme together with a second order Crank-Nicolson scheme for the time-discretization. Initially, we wrote a test version of this method in MATLAB and obtained, for instance in the case of a regular mesh, stable results even under advection-dominated conditions. Encouraged by these tests we continued by implementing basically the same method in C/C++. To accomplish this we had to modify and adapt it slightly because the code was finally to be embedded in a large model that simulates among other things the evolution of the snow cover. Besides the test simulations presented in this paper, we haven't obtained any conclusive results yet.

Contents

1 Introduction

Wind transport of snow has a major impact on the avalanche hazard as well as on the ecology and hydrology in mountainous environments. Topographic modification of the wind field causes the increased accumulation of snow in leeward slopes, which may result in spontaneous or human-caused avalanches. Basically, we distinguish two different kinds of snow transport in high alpine regions: saltation (hopping of particles above the surface following ballistic trajectories) and suspension (particles are picked up by turbulent eddies and are transported over large distances without contact to the ground).
Since the snow transport above ground (Suspension) has a large impact on the redistribution of snow and thus the evolution of the snow cover, it is inalienable for future forecast models to be able to simulate this transport process. A possible approach to a numerical solution is the description of the snow-air mixture as a continous fluid and therefore its mass transport as a fluid flow. These assumptions allow us to rewrite the mass balance for snow in suspension in form of an unsteady partial differential equation (PDE) of the advection-diffusion-type (together with appropriate boundary conditions):

$$\frac{\partial c}{\partial t} + \boldsymbol{b} \cdot \nabla c - \nabla \cdot (K \nabla c) = f, \tag{1}$$

where c is the concentration, $\boldsymbol{b}$ is the mean wind velocity, K is the diffusion matrix and f represents sink and source terms (note that all variables are functions of time and space). The wind field above alpine terrain is actually very complicated and far from being free of turbulence. Therefore we decompose it into the mean wind field $\boldsymbol{b}$ and into irregular swirls of motion (eddies) that are superimposed on $\boldsymbol{b}$. According to the mixing length theory [1] we take account of these eddies in terms of the diagonal diffusion matrix K. The f term may, for example, describe sublimation processes in the lower atmosphere due to the meteorological phenomenon called Föhn.
Neither the validity of the modeling of snow as a real fluid nor the quest for appropriate boundary conditions are the main aspects of this paper. The goal of this present study, focusing on the mathematical aspects of the problem, is to find an appropriate numercial method to solve the PDE (1) with, for instance, inhomogenous Dirichlet Boundary conditions on a well-chosen domain Ω above alpine terrain. The final problem (including the initial conditions) reads therefore:

$$\begin{cases} \frac{\partial c}{\partial t} + \boldsymbol{b} \cdot \nabla c - \nabla \cdot (K \nabla c) = f & \text{in } \Omega, \\ c(\boldsymbol{x}, t) = \chi(\boldsymbol{x}) & \text{on } \partial\Omega, \\ c(\boldsymbol{x}, 0) = \psi(\boldsymbol{x}) & \text{in } \Omega. \end{cases} \tag{2}$$

Facing the vast variety of numercial methods that are known and used these days, one has

to make a restrictive selection from the very beginning on. Since the numercial model that should finally arise from the following investigations and tests will be embedded in a large program that supplies in a preceeding step the wind field $\boldsymbol{b}$ on a well-defined mesh, we decided to use this very mesh as a finite element triangulation of the domain Ω.
After the investigation of general mathematical issues and the discussion of several possible finite element methods in one space dimension (sections 2, 3, 4), we shall restrict ourselves in section 5 to a more detailed analysis of the SUPG method in 3D, a method that seems to be apt to cope with the instability problems due to advection-domination of equation (2). In order to judge the chosen SUPG method upon its stability and convergence we had initially implemented a 3D test-version in MATLAB and then rewritten a slightly modified version of the method in C/C++ (section 6). Finally we shall discuss its performance on a regularly triangulated domain in section 7.

2 Standard Galerkin Method

Nowadays the variety of different methods for approximating partial differential equations (PDE) is enormous and first of all, we had to decide which class of methods could be appropriate for solving the unsteady advection-diffusion problem (2). Motivated by several authors like Quarteroni and Valli [4] and Johnson [3] we decided to focus our investigation on a finite element approach, more precisely on a Galerkin method. Since the problem of interest is clearly of an advection-dominated nature due to large wind velocities in alpine regions and since the closure of Ω by alpine terrain on its bottom side causes a rather irregular triangulation of the domain, care needs to be taken in choosing an apt version of the Galerkin method.
In order to simplify the mathematical development we will discuss the advantages and disadvantages of different Galerkin methods in one space dimension together with homogenous Dirichlet boundary conditions and only switch to a 3D formalism in the discussion of the SUPG method in section 3.2.

2.1 Weak Formulation of the Steady Problem (1D)

The classical or strong form (S) of a steady advection-diffusion equation in one space dimension with homogenous Dirichlet conditions reads as follows:

$$\begin{cases} b\frac{dc}{dx} = d\frac{d^2c}{dx^2} + f & \text{in [a,b]} \ \equiv I \\ \\ c(a) = 0, & c(b) = 0, \end{cases} \tag{3}$$

where c is the concentration, b is the advection coefficient, d is the diffusion coefficient and f is a sink or source term. Let's introduce now a space of test functions V, for instance a space of linear functions on I:

$$V = \{v \mid v \in C^0(I) \text{ and v' is p.c. on I}, v(a) = v(b) = 0\}. \tag{4}$$

If we multiply (3) by v∈V and integrate over the interval I we may rewrite the equation by using the scalar product notation $\int_a^b dx f(x)g(x) \equiv (f,g)$ and $\frac{dc}{dx} \equiv c'$ as:

$$\begin{cases} b(c',v) = -d(c',v') + (f,v) \quad \text{in I} \\ \\ c(a) = 0, \qquad c(b) = 0. \end{cases} \tag{5}$$

Note that we have integrated the first term on the right hand side by parts and used the fact that v(a)=v(b)=0. Obviously, every solution c of (5) is also an element of V and therefore we can reformulate the strong problem (S) in its weak or variational form (W):

$$\text{Find } c \in V \text{ such that } b(c',v) + d(c',v') = (f,v),\ \forall\ v \in V. \tag{6}$$

On the other hand, one can easily verify that a solution of (6) also solves (3) if c" is sufficiently regular. As for the (identical) trial and test spaces V we shall from now on use the Hilbert space

$$H_0^1 = \{v \mid v, v' \in L_2(I),\ v(a) = v(b) = 0\}. \tag{7}$$

Let us now derive the standard Galerkin method from the weak formulation (6).

2.2 Bubnov-Galerkin Method

Let's choose a finite dimensional subspace $V_h \subset H_0^1$ and introduce a basis $\{\phi_i(x)\}_{i=1}^n$ of V_h. The parameter $h > 0$ represents the mesh size of the finite elements and we assume that $\forall\ v \in V,\ \inf_{v_h \in V_h} \|v - v_h\| \to 0,$ as $h \to 0$. The corresponding finite dimensional weak formulation of (6) is therefore (W_h):

$$\text{Find } c_h \in V_h \text{ such that } b(c_h',v_h) + d(c_h',v_h') = (f,v_h)\ \forall\ v_h \in V_h. \tag{8}$$

Under certain regularity conditions the Laax-Milgram lemma (see for example [4]) ensures existence of a unique solution for (8) and its convergence to the exact solution c for $h \to 0$. Since the solution of (8) can be expanded on the chosen basis, i.e. $c_h = \sum_{i=1}^n \gamma_i \phi_i(x)$, we may rewrite (8) as: find γ_i, $i = 1, \dots, n$ such that:

$$b\sum_{i=1}^n \gamma_i(\phi_i'(x), v_h) + d\sum_{i=1}^n \gamma_i(\phi_i'(x), v_h') = (f, v_h),\ \forall\ v_h \in V_h \tag{9}$$

Figure 1: Triangular basis functions (hat functions) on [0,10]: ϕ_0, ϕ_5 and ϕ_{10}

and this is by basic linear algebra equivalent to: find γ_i, $i = 1, \ldots, n$ such that:

$$b \sum_{i=1}^{n} \gamma_i(\phi_i'(x), \phi_j(x)) + d \sum_{i=1}^{n} \gamma_i(\phi_i'(x), \phi_j'(x)) = (f, \phi_j(x)), \ \forall\ j = 1, \ldots, n. \tag{10}$$

The last equation (10) may be rewritten in matrix form as:

$$bBc + dKc = \Gamma, \tag{11}$$

where $B_{ij} = (\phi_j', \phi_i)$, $K_{ij} = K_{ji} = (\phi_j', \phi_i')$ and $\Gamma_j = (f, \phi_j)$. Reducing the original strong problem to solving a linear system of equations like (11) is referred to as the Galerkin method (Bubnov-Galerkin method). The matrix K is usually called the stiffness matrix.

So far, the finite dimensional subspace V_h hasn't been specified explicitely. In one space dimension, one usually divides the interval I=[a,b] into k+1 subintervals K_i with the nodes $a = x_0,\ x_1, \ldots,\ x_{k+1} = b$ and introduces the subspace

$$V_h = \{\text{ v is continous on I, } v\mid_{K_i} \text{ is linear, } \ \forall\ i = 1, \ldots, n,\ v(a) = v(b) = 0\}. \tag{12}$$

As a basis one may choose the triangular hat functions (see figure 1).

Before analysing some results obtained by the Bubnov-Galerkin method it is important to explain how to cope with inhomogenous Dirichlet boundary conditions, i.e. we want the function c(x) of equation (5) to satisfy the inhomogenous boundary conditions $c(a) = \nu$ and $c(b) = \eta$. In this case our test-space is simply

$$H^1 = \{v \mid v, v' \in L^2(I)\} \tag{13}$$

and its finite dimensional subspace is

$$V_h^* = \{\text{v is continous on I, } v\mid_{K_i} \text{ is linear, } \ \forall\ i = 1, \ldots, n,\ v(a) = \nu,\ v(b) = \eta\}, \tag{14}$$

where $K_i \equiv [x_i, x_{i+1}]$. We obtain a basis of V_h^* by simply adding the two basis vectors ϕ_0 and ϕ_{n+1} to the basis $\{\phi_i\}_{i=1}^n$ of V_h (see figure 1). The problem reads now:

$$\text{Find } c_h \in V_h^* \text{ such that } b(c_h', v_h) + d(c_h', v_h') = (f, v_h)\ \forall\ v_h \in V_h. \tag{15}$$

Figure 2: Stability problems with Bubnov-Galerkin for f=2, b=2 and d=0.4

Since we can write $c_h \in V_h^*$ as:

$$c_h = \Big(\sum_{i=1}^{n} \gamma_i \phi_i\Big) + \Big(\gamma_0 \phi_0 + \gamma_{n+1}\phi_{n+1}\Big) := z_h + \xi_h, \text{ where } z_h \in V_h,\ \xi_h \in V_h^*,$$

the inhomogenous problem (15) can finally be reformulated with respect to V_h instead of V_h^*:

$$\text{Find } z_h \in V_h \text{ such that } b(z_h', v_h) + d(z_h', v_h') = (f, v_h) - b(\xi_h', v_h) + d(\xi_h', v_h'),\ \forall\, v_h \in V_h. \quad (16)$$

2.3 Stability Issues

Unfortunately, the Bubnov-Galerkin method shows a behaviour of instability while solving problems that are advection-dominated. Let's consider the steady advection-diffusion equation in 1D:

$$\begin{cases} -dc''(x) + bc'(x) = f & \text{in } [0,10] \\ c(0) = 0, \qquad c(10) = 1. \end{cases} \quad (17)$$

In the following numerical example we set shall d=0.4 and b=2, expressing the fact that the problem is advection-dominated. In figure 2 we can observe that the numerical approximation by Bubnov-Galerkin to this problem shows an oscillatory behaviour for h=1 and h=0.5. A more detailed ananlysis (cf [4] and [3]) shows that these oscillations occur for $h > d$ and can be eliminated by choosing $h < d$, a result that is verified again in figure 2 for h=0.2. One possibility to avoid these tedious oscillations is obviously to decrease h in function of the diffusion coefficient d. However, this is a unconvienent solution since often the mesh size h can't be decreased arbitrarily (as it is the case for a mesh generated in advance). More successful strategies will be discussed in the following section.

3 Stabilization Methods

3.1 Artificial Diffusion and Petrov-Galerkin scheme

A well-known method that avoids instability due to advection-domination is the so-called Petrov-Galerkin method. The main idea of the Petrov-Galerkin approach is to take a test space that is different from the trial space, i.e. we may take the usual trial space for a homogenous problem:

$$V_h = \{\text{v is continous on I, } v\,|_{K_i} \text{ is linear},\ \forall\, i = 1, \dots, n,\ v(a) = v(b) = 0\}, \quad (18)$$

Figure 3: Stability of Petrov-Galerkin for f=2, b=2, d=0.4

but a modified version of the test space:

$$W_n = \text{span}\{\psi_1, \psi_2, \dots, \psi_n\}, \tag{19}$$

where

$$\psi_j(x) = \phi_j(x) + \sigma(h^{-1}x - j). \tag{20}$$

The functions $\phi_j(x)$ are still the basis functions (hat functions) and $\sigma(x)$ is given by:

$$\sigma(x) = \begin{cases} -3x(1+x) & , \quad -1 \leq x \leq 0 \\ -3x(1-x) & , \quad 0 \leq x \leq 1 \\ 0 & , \quad |x| \geq 1 \end{cases} \tag{21}$$

The corresponding finite dimensional problem reads then

$$\text{Find } c_h \in V_h \text{ such that } b(c_h', v_h) + d(c_h', v_h') = (f, v_h) \ \forall \ v_h \in W_h. \tag{22}$$

Direct calculations prove true that the oscillations are indeed eliminated, i.e. even for rather large spacings like h=1 we obtain a non-oscillatory solution (see figure 3). Although this Petrov-Galerkin method is stable in comparison to the standard Galerkin method, we can observe that the convergence is worse; later on in this section we shall see why.

A more intuitive comprehension of the scheme (22) can be achieved by considering the finite difference scheme (where we suppose that $b > 0$):

$$\begin{cases} -d\frac{c_{j+1}-2c_j+c_{j-1}}{h^2} + b\frac{c_j-c_{j-1}}{h} = 0 \quad , j = 1, \dots .n \\ c_0 = 0 \\ c_{n+1} = 1 \end{cases} \tag{23}$$

As a matter of fact, (23) is equivalent to the Petrov-Galerkin problem (22) (cf [4]). Due to the positive coefficient b, the transport occurs from the left to the right and therefore one is motivated to write an upwind finite difference scheme as (23) to solve the original problem. However, this scheme is obviously only of O(h) which implies a loss in accuracy and a diminished convergence as mentioned before. By noticing that the transport term of (23) can be written as

$$\frac{u_j - u_{j-1}}{h} = \frac{u_{j+1} - u_{j-1}}{2h} - \frac{h}{2}\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2}, \tag{24}$$

we may generalize the upwind scheme to higher dimensional problems by generally introducing an artificial diffusion term corresponding to the second term on the right hand side of (24). But also here, we shall not forget that the artificial diffusion term leads to a first order accurate scheme and therefore it is more convenient to introduce stabilizing terms that don't affect the accuracy imposed by the chosen finite element subspace. Two good examples for strongly consistent methods are the Bubble functions (see [4]) and the SUPG method that shall be discussed in (3.2).

3.2 Numercial Approximation by SUPG

Since, at the end of the day, we shall decide to implement a SUPG scheme we would like to introduce this particular method directly in three space dimensions. The steady problem in 3D reads :

$$\begin{cases} \boldsymbol{b} \cdot \nabla c - \nabla \cdot (K \nabla c) = f & \text{in } \Omega \\ c(\boldsymbol{x}) = g(\boldsymbol{x}) \qquad \text{on } \partial\Omega \end{cases} \tag{25}$$

The respective weak formulation of (25) is (with, for instance, $g(\boldsymbol{x}) \equiv 0$ and with $(\cdot,\cdot)$ the L^2 scalar product):

$$\text{find } c \in H_0^1(\Omega), \text{ such that } (\boldsymbol{b} \cdot \nabla c, v) + (K \nabla c, \nabla v) = (f, v), \ \forall \ v \in H_0^1, \tag{26}$$

where the diffusion term has been integrated by parts according to the divergence theorem. With the notation $a(v,w) = (\boldsymbol{b} \cdot \nabla v, w) + (K \nabla v, \nabla w)$, where a is obviously a bilinear form, we may rewrite (26) as follows:

$$\text{find } c \in H_0^1(\Omega), \text{ such that } a(c,v) = (f,v), \ \forall \ v \in H_0^1. \tag{27}$$

Instead of simply choosing a finite element subspace V_h of H_0^1 and formulating the standard Galerkin method, we first introduce a splitting of the differential operator L into $L = L_S + L_{SS}$ such that:

$$L_{SS} c = \boldsymbol{b} \cdot \nabla c \qquad \text{and} \qquad L_S = -\nabla \cdot (K \nabla c). \tag{28}$$

Define:

$$\Lambda_h^{(\rho)}(z,v) := \sum_{K \in T_h} \delta_K \Big(Lz, \frac{h_K}{|\boldsymbol{b}|}(L_{SS} + \rho L_S) v \Big)_K \text{ and} \tag{29}$$

$$\Theta_h^{(\rho)}(v) = \sum_{K \in T_h} \delta_K \Big(f, \frac{h_K}{|\boldsymbol{b}|}(L_{SS} + \rho L_S) v \Big)_K, \tag{30}$$

where T_h is a triangulation of the domain (see section 5.3), ρ and δ_K are parameters and $(\cdot,\cdot)_K$ is the scalar product on the element K. Now, we introduce the stabilized approximation in terms of the finite element subspace V_h:

$$\text{Find } c_h \in V_h \text{ such that } a(c_h, v_h) + \Lambda_h^{(\rho)}(c_h, v_h) = (f, v_h) + \Theta_h^{(\rho)}(v_h),\ \forall\, v_h \in V_h. \tag{31}$$

Since for all choices of ρ and δ_K the equation (31) is verified for the exact solution, this method is said to be strongly consistent (in contrary to the artificial diffusion scheme which decreases the global order of convergence). This is a very desirable property because the stabilization does not decrease the order of convergence imposed by the finite element space. From now on we will confine our attention to the special case $\rho = 0$, the so-called SUPG (Streamline-Upwind/Petrov-Galerkin) method. The choice of δ_K will be discussed in section 5.2.

4 Time-Advancing

After having introduced several methods for a semi-discretization in space, we have to focus now on the time-dependant problem, i.e. the unsteady advection-diffusion equation:

$$\begin{cases} \frac{\partial c}{\partial t} + Lc = f, & \text{in } Q_T := \ ]0,T[\times\Omega \\ c = g(\boldsymbol{x}), & \text{on } \Sigma_T := \ ]0,T[\times\partial\Omega \\ c = c_0, & \text{on } \Omega,\ t=0 \end{cases} \tag{32}$$

where

$$Lc := \boldsymbol{b}\cdot\nabla c - \nabla\cdot(K\nabla c)$$

and all variables are functions of time and space. The case of inhomogenous Dirichlet boundary conditions is dealt similarly as in one dimension.

4.1 Time-Advancing by Finite Differences

There are several possibilities to solve the time-dependent problem (32), but a common approach consists in a semi-discretization in space by means of a previously discussed finite element method and a subsequent discretization of the arising scheme in time. In order to achieve the discretization in time of e.g. a Bubnov-Galerkin scheme:

$$\left(\frac{\partial c_h}{\partial t}, v_h\right) + a(c_h, v_h) = (f, v_h), \tag{33}$$

one may apply the following θ-scheme:

$$\begin{cases} \frac{1}{\Delta t}(c_h^{n+1} - c_h^n, v_h) + a(\theta c_h^{n+1} + (1-\theta)c_h^n, v_h) = (\theta f(t^{n+1}) + (1-\theta) f(t_n), v_h), \quad \forall\ v_h \in V_h \\ c_h^0 = c_{0,h}. \end{cases} \tag{34}$$

The special case $\theta = 0$ corresponds to the *Backward Euler* (BE) scheme and is a first order implicit scheme, whereas $\theta = \frac{1}{2}$ corresponds to a *Crank-Nicolson* (CN) scheme that is of $O(h^2)$. Another possibility is the *semi-implicit* scheme where we evaulate the principal part of the differential operator L at time t_{n+1} and the remaining parts at time t_n:

$$\begin{cases} \frac{1}{\Delta t}(c_h^{n+1} - c_h^n, v_h) + (K\nabla c_h^{n+1}, \nabla v_h) + (\boldsymbol{b} \cdot \nabla c_h^n, v_h) = (f(t_{n+1}), v_h)\ \forall\ v_h \in V_h \\ c_h^0 = c_{0,h}. \end{cases} \tag{35}$$

A third possibility is the so-called *algebraic splitting*: After the semi-discretization in space by e.g. a Bubnov-Galerkin scheme and the reformulation of the arising scheme in terms of matrices we obtain the following ODE (cf. 11):

$$\begin{cases} A\frac{d}{dt}\boldsymbol{c_h}(t) + P\boldsymbol{c_h}(t) = \boldsymbol{F}(t) \\ \boldsymbol{c_h}(0) = \boldsymbol{c}_{0,h}. \end{cases} \tag{36}$$

We split the matrix P into $P = P_1 + P_2$:

$$(P_1)_{ij} = (K\nabla\phi_j, \nabla\phi_i), \qquad (P_2)_{ij} = (\boldsymbol{b} \cdot \nabla\phi_j, \phi_i) \tag{37}$$

and solve the ODE in time according to the Peaceman-Rachford scheme with one intermediate step:

$$\begin{cases} A\frac{\boldsymbol{c}^{n+1/2}-\boldsymbol{c}^{n}}{\tau} + P_2\boldsymbol{c}^{n+1/2} + P_1\boldsymbol{c}^{n} = \boldsymbol{F}(t_{n+1/2}) \\ A\frac{\boldsymbol{c}^{n+1}-\boldsymbol{c}^{n+1/2}}{\tau} + P_2\boldsymbol{c}^{n+1/2} + P_1\boldsymbol{c}^{n+1} = \boldsymbol{F}(t_{n+1/2}) \\ \tau = \frac{\Delta t}{2}, \qquad \boldsymbol{c}^0 = \boldsymbol{c}_{0,h} \end{cases} \tag{38}$$

4.2 Discontinous Galerkin Method

A more sophisticated approach to the time-dependant problem is the discontinous Galerkin method which makes use of a finite element formulation in space and time. As we will see, the computational cost of this method is fairly high and therefore it is, although pretty efficient, not very attractive. However, we have implemented a discontinous Galerkin method combined with a SUPG-discretization in time and shall discuss it briefly in section 5.1.

The basic idea of the discontinous Galerkin method is to introduce the space of trial functions

$$W_{hk} = \{v : I \to V_h \ \mid \ v_{|I_m} \in \boldsymbol{P}_q(I_m),\ m = 1, \dots, M\}, \tag{39}$$

where

$$\boldsymbol{P}_q(I_m) = \{v : I_m \to V_h : v(t) = \sum_{i=0}^{q} v_i t^i,\ \text{with } v_i \in V_h\},$$

and where I is the global time interval, $I_m \equiv [t_m, t_{m+1}]$ are the finite elements in time and q is the degree of the polynomial that we shall set to 1. Therefore W_{hk} is the space of functions on I with values in V_h that on each time interval I_n vary as polynomials of degree at most q and that may be discontinous in time at the points t_m. For a more detailed investigation of the method we refer to [3] and [4], and we shall simply give the final scheme for the time interval I_m, with q=1 and k_m the corresponding length of the time interval:

$$\begin{cases} (U_0, v) + k_m a(U_0, v) + (U_1, v) + \frac{1}{2}k_m a(U_1, v) = (U_-^{m-1}, v) + \int_{I_m}(f(s), v)ds,\ \forall\ v \in V_h, \\ \frac{1}{2}k_m a(U_0, v) + \frac{1}{2}(U_1, v) + \frac{1}{3}k_m a(U_1, v) = \frac{1}{k_m}\int_{I_m}(s - t_{m-1})(f(s), v)ds,\ \forall\ v \in V_h, \end{cases} \tag{40}$$

where U_-^{m-1} is the left-side limit in time of U at t_{m-1}.

The final solution on the time interval will be, according to (39):

$$U(t) = U_0 + \frac{t - t_{m-1}}{k_m},\ t \in I_m,\ U_i \in V_h.$$

Figure 4: Time evolution by Petrov-Galerkin/Operator-Splitting with h=dt=0.1

Since we have to solve this system for U_0 and U_1, the amount of work is doubled in comparison to a simple time-stepping, but higher accuracy is achieved.

5 Choosing SUPG and Crank-Nicolson

5.1 Comparison of the different methods in 1D

The actual aim of implementing the whole variety of different time and space discretizations in 1D presented in sections 3 and 4 was to choose the best method for the problem of interest, i.e. the three dimensional problem (2). For reasons we have already discussed in section 2.3 the standard Galerkin scheme is not a very fruitful method for advection-dominated problems and we definitely had to find a stabilized version of the Galerkin scheme.
Let's focus now on the three different time stepping approaoches presented in section 4.1, i.e. the $\theta-$*scheme* (34), the *semi-implicit method* (35) and the *algebraic splitting* (38). The two $\theta-$schemes of interest, i.e. the Backward Euler scheme ($\theta = 0$) and the Crank-Nicolson scheme ($\theta = \frac{1}{2}$) are both unconditionally stable (a very desirable property), but since the CN method is of order $O(h^2)$ we shall prefer it to the BE scheme, only of $O(h)$. For solely diffusion-dominated problems all three methods perform well, as we can see for example in figure 4 which presents the solution to (3) by a Petrov-Galerkin/Operator-Splitting method after 0.6, 6 and 60 seconds respectively. However, applied to advection-dominated problems, the performance of the semi-implicit scheme and the operator-splitting method are very poor as we can see in figures 5 and 6. These approximations are unstable and show huge oscillations whereas the approximations obtained by Petrov-Galerkin in space and Crank-Nicolson in time are stable and converge to the exact solution with decreasing mesh-size h as we can observe in figure 7. Note that if we modify the semi-implicit scheme (35) by evaluating the transport term instead of the diffusion term at time t_{n+1}, we obtain very unstable results already for low advection-domination.
Finally the Petrov-Galerkin scheme (22) is only first order accurate and therefore we decided to focus on the SUPG approach. The last decision was to be taken between a time-stepping by the Crank-Nicolson scheme and the discontinous Galerkin method. A direct comparison between those two methods (i.e. SUPG-CN and SUPG-DG) in figure 8 reveals a somewhat strange behaviour of the SUPG-DG method: we actually loose accuracy by decreasing the time-step dt from 0.5s to 0.1s, and with dt=0.1s SUPG-CN is still more precise than SUPG-DG (note that all graphs represent the concentration after 500s and can therefore be considered to be in the steady state. Motivated by these results and the substantial computational cost arising in the SUPG-DG method, we finally decided to work with SUPG-CN.

Figure 5: Instability of Petrov-Galerkin/Operator-Splitting for advection-dominated case after 60 seconds with d=0.4, b=8, h=dt=0.1

Figure 6: Instability of Petrov-Galerkin/Semi-Implicit scheme after 60 seconds with d=0.4, b=8, h=dt=0.1

5.2 The Complete Discretization by SUPG-CN

After having introduced the strongly consistent SUPG method for the steady advection-diffusion problem in section 3.2 we will now work out the numerical scheme that enables us to find an approximation to the unsteady problem (32).
Starting off the strong form of the unsteady problem, we obtain, after having multiplied both sides by a test function $v \in V_h$ and having integrated over the domain Ω, the following finite dimensional weak formulation:

$$\text{find } c_h \in V_h, \text{ such that } \frac{d}{dt}(c_h(t), v_h) + a(c_h(t), v_h) = (f(t), v_h), \ \forall\ v \in V_h, \tag{41}$$

where $c_h(0) = c_{0,h}$. According to section 3 we write the stabilized semi-discrete scheme:

$$\begin{cases} \frac{d}{dt}(c_h(t), v_h) + a(c_h(t), v_h) + \sum_{K \in T_h} \delta_K \Big(\frac{\partial c_h}{\partial t}(t) + Lc_h(t), \frac{h_K}{|\boldsymbol{b}|}(L_{SS} v_h) \Big)_K \\ = (f(t), v_h) + \sum_{K \in T_h} \delta_K \Big(f(t), \frac{h_K}{|\boldsymbol{b}|}(L_{SS} v_h) \Big)_K, \ \forall\ v \in V_h, \\ c_h(0) = c_{0,h}, \end{cases} \tag{42}$$

for $t \in (0, T)$. And finally, using notation (29) and (30), we can write the full discretization with a Crank-Nicolson scheme in time:

$$\begin{cases} \frac{1}{\Delta t}(c_h^{n+1} - c_h^n, v_h) + \frac{1}{2} a\Big((c_h^{n+1} + c_h^n), v_h\Big) \\ + \sum_{K \in T_h} \delta_K \Big(\frac{1}{\Delta t}(c_h^{n+1} - c_h^n) + \frac{1}{2}(\boldsymbol{b} \cdot \nabla - \nabla \cdot (K \nabla))(c_h^{n+1} - c_h^n), \frac{h_K}{|\boldsymbol{b}|}(L_{SS} v_h) \Big)_K \\ = (f(t), v_h) + \sum_{K \in T_h} \delta_K \Big(f(t), \frac{h_K}{|\boldsymbol{b}|}(L_{SS} v_h) \Big)_K, \ \forall\ v \in V_h, \\ c_h(0) = c_{0,h}. \end{cases} \tag{43}$$

Figure 7: Petrov-Galerkin with Crank-Nicolson: Convergence with increasing mesh-size h

Figure 8: Good performance of SUPG-CN; SUPG-DG shows a weird behaviour for decreasing dt. Results obtained after 500s and with b=8, d=0.4

Note that we have split the time interval $[0, T]$ into M+1 subintervals $[t_i, t_{i+1}]$, $\forall\ i = 0, \dots, M$.
The value δ_K is another important parameter of the SUPG method and in what concerns its choice, we would like to refer to a statement by Brezzi, Franca, Hughes and Russo (1997): *'The general strategy to find it may be described as follows: first you guess the form of it, then you perform an error analysis to confirm that the order of the parameter yields optimal or quasi-optimal estimates. If this is not the case, guess again, until you get the best possible estimate. Then perform numerical experiments for your choices and try to cover a wide range of problems of interest. If all works then you have a method.'* Inspired by several authors we decided to take for instance the following choice of δ_K:

$$\begin{cases} \delta_K = \frac{h_K^2}{12\epsilon}, & \text{if } Pe < 1, \\ \\ \delta_K = \frac{h_K}{2|\boldsymbol{b}|}, & \text{if } Pe \geq 1, \end{cases} \tag{44}$$

and Pe is the local Peclet number $\frac{|\boldsymbol{b}|h_K}{6\epsilon}$ and $\epsilon = Tr(K)$.

5.3 Finite Element Approximation and Shape Functions

We suppose that our domain of interest, $\Omega \in \mathbb{R}^3$, can be decomposed in the following way:

$$\Omega = \bigcup_{K \in T_h} K, \tag{45}$$

where each K is a polyhedron and T_h is called a triangulation of Ω. Admissible triangulations are such that no vertex of an element touches the edges of its adjacent elements, i.e. each element shares the position of its vertices with its adjacent elements. In 3D, one often uses either tetrahedrons or hexahedrons for the triangulation; we shall confine myself to a hexahedral triangulation, i.e. a triangulation by (irregular) cubes. We assume furthermore that for each element K there exists an invertible map F such that $K = F(\widehat{K})$, where $\widehat{K}$ is a reference hexahedron. Once the triangulation is determined, we can choose a finite dimensional subspace $V_h \subset H_0^1$, in general a space of piecewise polynomials, i.e. $\forall K \in T_h,\ P_K := \{v_{h|K} | v_h \in V_h\}$ consists of algebraic polynomials. Basically we will choose V_h in such a way that its basis can be expressed in (p,q,r) space (the space of the reference element) by the following 8 trilinear functions defined on the reference element:

$$\begin{cases} \widehat{\phi}_1(p,q,r) = pqr, \\ \widehat{\phi}_3(p,q,r) = p(1-q)r, \\ \widehat{\phi}_4(p,q,r) = (1-p)qr, \\ \widehat{\phi}_5(p,q,r) = (1-p)(1-q)r, \\ \widehat{\phi}_6(p,q,r) = (1-p)q(1-r), \\ \widehat{\phi}_7(p,q,r) = p(1-q)(1-r), \\ \widehat{\phi}_8(p,q,r) = (1-p)(1-q)(1-r). \end{cases} \tag{46}$$

Note that the term shape function refers to the restricition of a basis function to a specified element; the functions (46) are therefore called shape functions.

5.4 Boundary Conditions

The appropriate choice of boundary conditions is an essential part of the whole modeling process and special care needs to be taken. For the following discussion it is important to note that we will work with a wind field which is supposed to be constant throughout the interval of integration [0,T] and that we will stop the iteration as soon as the fluid flow becomes quasi-steady.
The bottom side of Ω: Since there is a thin layer of snow transport by saltation just above the terrain (hopping of the particles), the snow concentration on the lower boundary of Ω is mainly influenced by this saltation concentration. Therefore we take inhomogenous Dirichlet boundary conditions that are determined by the saltation concentration at the beginning of the time interval. On the lateral and the upper boundary of Ω we apply an infinity-limit, i.e. we assume that the precipitation inflow and outflow on these boundaries are not influenced by the perturbation caused by steep topography and turbulent wind fields within the domain. Thus the apt boundary conditions seem to be of an inhomogenous Dirichlet type and are determined by the precipitation concentration in the air at the beginning of the time-advancing. A more detailed discussion of the process taking place on the lower boundary of the domain can be found in section 6.4.

6 Implementational Issues

First of all we have to specify several features of our discretization model: we suppose that our domain Ω can be decomposed into an union of hexahedrals according to (45) and that we enumerate the nodes of the space discretization from 1 to N and the nodes of each element from 1 to 8. We assume that there are in total E elements and that we enumerate with a well-chosen enumeration scheme. Furthermore we split the time interval $[0,T]$ into M+1 uniform intervals of length Δt. Now we express c_h^m in terms of the finite dimensional basis on V_h:

$$c_h^m = \sum_{i=1}^{N} \gamma_i^m \phi_i(\boldsymbol{x}). \tag{47}$$

Then we plug this sum into the fully descretized scheme (43) and we immediately obtain the system of equations:

$$\begin{cases} \Big(\frac{1}{\Delta t}B + \frac{1}{2}C + \frac{1}{2}A_1 + \frac{1}{\Delta t}A_2\Big)c_h^{n+1} = Bf + A_3 f + \Big(\frac{1}{\Delta t}B - \frac{1}{2}C - \frac{1}{2}A_1 + \frac{1}{\Delta t}A_2\big)c_h^n, \\ \\ c_h^0 = c_{0,h}, \end{cases} \tag{48}$$

where

$$\begin{cases} B_{ij} = (\phi_j, \phi_i), \\ \\ C_{ij} = a(\phi_j, \phi_i), \\ \\ (A_1)_{ij} = \frac{1}{2}\sum_{K \in T_K} \delta_K \Big(\boldsymbol{b} \cdot \nabla\phi_j, \frac{h_K}{|\boldsymbol{b}|}(\boldsymbol{b} \cdot \nabla\phi_i)\Big)_K, \\ \\ (A_2)_{ij} = \frac{1}{2}\sum_{K \in T_K} \delta_K \Big(\phi_j, \frac{h_K}{|\boldsymbol{b}|}(\boldsymbol{b} \cdot \nabla\phi_i)\Big)_K, \\ \\ (A_3)_{ij} = \frac{1}{2}\sum_{K \in T_K} \delta_K \Big(f, \frac{h_K}{|\boldsymbol{b}|}(\boldsymbol{b} \cdot \nabla\phi_i)\Big)_K. \end{cases} \tag{49}$$

Thus we have reduced the finite dimensional weak formulation of the unsteady PDE to solving a system of linear equations (48) for each time step. At this point we are able to reveal a major advantage of the finite element formulation with basis functions of compact support: in point of fact, for all matrices of (49), the matrix elements M_{ij} are zero except for the case that the nodes i and j are nearest neighbours. This implies that the matrices (49) are sparse matrices. In order to calculate these matrix elements we refer to the concept of element matrices. Since every element has 8 nodes, the element matrices have a size of 8×8 and adopt the following generic form (here for the element matrix of B in the element K):

$$\begin{pmatrix} (\phi_1, \phi_1)_K & (\phi_2, \phi_1)_K & \cdots & \cdots \\ (\phi_1, \phi_2)_K & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & (\phi_8, \phi_8)_K \end{pmatrix}, \tag{50}$$

where the numbers refer to the enumeration of the nodes within the element and $(\cdot,\cdot)_K$ is the scalar product on the element K. After having calculated the element matrices for every single element we have to assemble these element matrices into the final matrices that occur in (48). For this purpose we introduce the two assembling matrices Z and T: Z is a $3 \times M$ matrix which contains the x, y, and z coordinates of every single node in the domain and T is a matrix of size $8 \times E$ and stores for all 8 nodes of each element the global node number.

6.1 Numerical Integration

In order to be able to start the time-stepping we need to calculate the matrices (49) and thus the element matrices. As mentioned above we proceed element by element, calculate the corresponding element matrices and assemble them into the final matrices. As indicated by the generic form of these matrices (50), this involves the calculation of integrals over the respective element K in Ω, which we will refer to as the (x,y,z) space. Since our domain is composed by irregular hexahedrons we need to switch by means of an invertible transformation to a reference element which has its 8 vertices at $(1,-1,-1)$, $(1,1,-1)$, $(-1,1,-1)$, $(-1,-1,-1)$, $(1,-1,1)$, $(1,1,1)$, $(-1,1,1)$, $(-1,-1,1)$ in (p,q,r) space. We have chosen a so called isoparametric transformation, i.e. a transformation

$$\begin{cases} T_K : \widehat{K} \to K, (p,q,r) \mapsto (x,y,z), \\ F(p,q,r) = \Big(\sum_{i=1}^8 x_i \widehat{\phi}_i(p,q,r), \sum_{i=1}^8 y_i \widehat{\phi}_i(p,q,r), \sum_{i=1}^8 z_i \widehat{\phi}_i(p,q,r) \Big), \end{cases} \tag{51}$$

where $\widehat{\phi}_i(p,q,r)$ are the basis functions and (x_i, y_i, z_i) are the coordinates of the ith node of K in (x,y,z) space. The relation between $\phi(x,y,z)$ and $\widehat{\phi}(p,q,r)$ is due to linearity of the coordinate transformation:

$$\phi(x,y,z) = \widehat{\phi}\Big(F_1^{-1}(x,y,z), F_2^{-1}(x,y,z), F_3^{-1}(x,y,z)\Big) \equiv \widehat{\phi}(p,q,r). \tag{52}$$

The name isoparametric point transformation is induced by the fact that the coordinates transform in the same manner as the interpolated function c_h:

$$c_{h|K} = \sum_{i=1}^{8} \gamma_i \phi_i(p,q,r). \tag{53}$$

According to the theorem of multiple integration, we can therefore evaluate the following integral on the reference element:

$$\int_K dxdydz\ \phi_j(x,y,z) = \int_K dxdydz\ \widehat{\phi}_j(p,q,r) = \int_{\widehat{K}} dpdqdr\ \widehat{\phi}_j(p,q,r)\ \Big|\frac{\partial(x,y,z)}{\partial(p,q,r)}\Big|. \tag{54}$$

Together with the equality

$$\nabla\phi_j = (J^{-1})^T\nabla\widehat{\phi}_j = \frac{1}{detJ}J_0\nabla\widehat{\phi}_j, \tag{55}$$

where J_0 is the transpose of the adjoint of J, we can finally rewrite the three types of integrals that we will need to calculate in order to obtain the element matrices:

$$\begin{cases} \int_K dxdydz\ \nabla\phi_j\nabla\phi_i = \int_{\widehat{K}} dpdqdr\ (J_0\nabla\widehat{\phi}_j)(J_0\nabla\widehat{\phi}_i)\frac{1}{|detJ|} \\ \int_K dxdydz\ \nabla\phi_j\phi_i = \int_{\widehat{K}} dpdqdr\ (J_0\nabla\widehat{\phi}_j)\widehat{\phi}_i\frac{detJ}{|detJ|} \\ \int_K dxdydz\ \phi_j\phi_i = \int_{\widehat{K}} dpdqdr\ \widehat{\phi}_i\widehat{\phi}_j|detJ|. \end{cases} \tag{56}$$

Note that the Jacobian of the transformation is easily calculated by equation (51), e.g.

$$\frac{\partial x}{\partial p} = \sum_{i=1}^{8} x_i\frac{\partial\widehat{\phi}_i}{\partial p}(p,q,r).$$

Once we are left with integrals over the reference element (56), we can easily calculate these numerically by first order Gaussian quadrature for triple integrals (cf [5]).

6.2 Sparse Matrices

The size of the matrices in (49) depends obviously on the total number of nodes N in the domain and is precisely $(N \times N)$. For instance, already for a very rough triangulation of say $63 \times 63 \times 12$ nodes we will end up with handling matrices of size (47628×47628). If we assume a storage use of 8 byte per entry of the datatype double we need roughly 18 GB to store one single matrix. Here it is crucial that our matrices are sparse, i.e. the major part of the entries is actually zero. Inspired by a sparse matrix formulation of [6] we have written an algorithm that places the entries of the element matrices into vectors which represent the sparse formulation of the final matrices (49). The multiplication of a vector by a matrix or its transpose have been implemented according to an algorithm of [6].

6.3 Solution of the Linear Algebraic Equations

The system of linear equations that remains to be solved at each time step is another crucial point in what concerns the computational costs of the problem. Since the matrices (49) are generally not symmetric we decided to use the Bi-ConjugateGradient method, a special derivative of the standard Conjugate Gradient method (which is good for symmetric and positive definite matrices). This method is an iterative method which could be accelerated by introcucing a preconditioner. For further informations, we refer to [4].

6.4 Modeling and Calculation of Snow Deposition end Erosion

Basically, we modelise the process of snow accumulation and erosion on the snow cover as follows: if a snow particle in suspension (i.e. at a pos $\boldsymbol{x} \in \Omega$) is moved by wind or diffusion towards the lower boundary (i.e. the snow surface) it enters the saltation layer, the zone of particles which are hopping along ballistic trajectories on the surface instead of being in suspension. We assume that for each particule entering the saltation layer, another particle (or the particle itself) is capted by the bulk of the snow cover. The capted particule is for instance bounded by the bulk and is therefore a contribution to the growing snow layer. However, if there is a windfield whose projection on the normal vector of the surface is positive it can erode snow from the bulk, adopted that the wind speed is above a certain threshold value. The assumption that the number of particles in the saltation layer is constant is a good justification for the (constant) inhomogenous Dirichlet condition (see section 5.4).
Obviously, there are two contributions to the snow accumulation: one contribution due to the wind field and the other one due to the diffusion matrix K. Since we have Dirichlet conditions on the lower boundary it is pointless to calculate the inflow due to the windfield on the very bottom of the domain. In order to cope with this problem we have introduced an artificial layer situated inbetween the snow surface nodes and the first layer of nodes in the interior of the domain. The height of this artificial layer can be adjusted, but it should not be set too low since the resulting flat elements on the bottom may cause stability problems. Optimal position is roughly the height of the saltation layer since such a choice is coherent with the modeling described above.
Next we need the windfield at the artificial nodes. This is a very delicate point since the accumulation is mainly determined by this wind field. According to the theory of real fluid flow, one would suppose the wind flow to become parallel to the topography with decreasing distance to the surface. However, this is pointless since the inflow/outflow would be zero. On the other hand, the wind field component parallel to the surface decays logarithmically (cf [1]) and the wind speed at a height $\hat{h}$ between the height h and the ground reads:

$$v(\hat{h}) = \frac{ln(\frac{\hat{h}}{Z0})}{ln(\frac{h}{Z0})}, \tag{57}$$

where Z0 is the roughness length. For instance, we apply a compromise, i.e. the wind vector at an artificial node points in the same direction as the wind vector at the point above and its norm is reduced by the factor of equation (57):

$$\hat{\boldsymbol{b}} = \frac{ln(\frac{\hat{h}}{Z0})}{ln(\frac{h}{Z0})} \frac{\boldsymbol{b}}{|\boldsymbol{b}|}. \tag{58}$$

Using this wind field we calculate the accumulation of snow at every step of the time integration procedure as follows:

$$c_{\boldsymbol{b}}^{acc}(t + \Delta t) = c_{\boldsymbol{b}}^{acc}(t) - \hat{\boldsymbol{b}} \cdot \boldsymbol{n}\ c(t)\Delta t, \tag{59}$$

where $\boldsymbol{n}$ is the normal vector of the surface at the respective node and the minus sign is due to the fact that $\boldsymbol{n}$ points upwards. A similar approach can be applied to the diffusion contribution, that is to say:

$$c_{K}^{acc}(t + \Delta t) = c_{K}^{acc}(t) + (K\nabla c) \cdot \boldsymbol{n}\ \Delta t. \tag{60}$$

7 Tests and Discussions

First we implemented the SUPG-CN method in MATLAB in order to survey the stability issues on regular meshes. Throughout the whole implementation the manyfold of subroutines (e.g. numerical integration, linerar system solver) have been tested consecutively to their accurate performance and won't be discussed in this section at all.
Although the analytic solution to the PDE (2) is in general far from being trivial (and in the case of non-uniform meshes even unknown), there are several situations where we can reduce the equation (2) to a simpler one. For example, if we choose the domain to be a cube of side length L, if we let the wind blow into the domain only in x-direction (i.e. $\boldsymbol{b} = (b, 0, 0)$), if we allow diffusion only in x-direction (i.e. $K = diag(d, 0, 0)$) and if we impose a constant concentration β on the whole boundary, the solution c_h^m on a line of nodes in x-direction should converge to the analytic solution of the following ODE for $h \to 0$, $\Delta t \to 0$ and m sufficiently large:

$$\begin{cases} bc' - dc'' = f, & x \in [0, L] \\ x(0) = x(L) = \beta \end{cases} \tag{61}$$

Note that the line of nodes in x-direction should be taken in the middle of the domain, since close to the boundary the concentration has to approach the imposed boundary conditions.
At this point we would like to present some of the most important test situations: For instance, the time evolution is not a problematic aspect since we know from simple error analysis that the Crank-Nicolson scheme is of $O(h^2)$. Therefore we confined ourselves to

Figure 9: Time evolution at time t=0.5s, 1s, 2.5s, 5s, 10s, 50s of the SUPG-CN method and the exact solution

Figure 10: Performance of SUPG-CN under advection-dominated conditions (b=10, d=1)

verifying that the numerical approximation converges in time to the steady state solution of the problem. In figure 9 we can see that this convergence is indeed verified for the case of β=1, $\boldsymbol{b} = (1, 0, 0)$ and $K = diag(1, 0, 0)$. Another important case is the solution under advection-dominated conditions in figure 10: the solution is still stable for a wind velocity of $\boldsymbol{b} = (10, 0, 0)$ and a diffusion matrix $K = diag(1, 0, 0)$. This is a very important result since stability for advection-dominated cases is the main issue of this study. A second advection-dominated test run for more complicated advection and diffusion values, i.e. $\boldsymbol{b} = (7.4, 4.23, 5.3)$, $\quad diag(K) = (0.7, 1.6, 0.6)$, produced satisfactory results as well (see figure 11).
Finally, we would like to stress the importance of boundary conditions and wind fields that are physically meaningful. As soon as we try to solve the equation for non-physical conditions, the method may produce unstable results: in figure 12 we can see that a negative diffusion coefficient leads to a complete failure of the method ($\boldsymbol{b} = (1, 1, 1)$ and $K = diag(1, 1, -1)$).

After the test implementation in MATLAB we rewrote and adapted the code to the final mesh and wind fields in the programming language C/C++. On the test mesh described above the code performs in the exactly same way as the MATLAB code and for instance we were unable to obtain conclusive results on the final mesh above steep topography: the tests we ran so far on highly turbulent wind fields converged at each time step but did not reach a steady state with increasing time. However, this behaviour may also be caused by wind fields and K matrices that are physically not meaningful enough and thereby let the method fail.

Figure 11: SUPG-CN for advection-dominated conditions; the concentration c (in z-direction) for a layer of nodes in the middle of the domain

Figure 12: SUPG-CN: Impact of a negative diffusion coefficient

8 Conclusion and Outlook

The tests performed on a regular mesh in section 7 are rather satisfactory; the main goal, the stability under advection-dominated conditions is clearly achieved.
But at this point it must be emphasized that these results are not at all representative for the mesh above alpine terrain with complicated wind fields: the elements of the final mesh above steep topography will be far from regular and in contrary rather flat. Thus it may be fruitful to look for differently shaped reference elements and more adequate parametrizations. Another important point is the modeling of the wind field, because it is a priori not divergence free in contradiction to our assumption in equation (1). Furthermore the treatment of turbulent eddies by means of a diffusion matrix is delicate too.
Another critical aspect is the choice of boundary conditions and it is overwhelmingly probable that the infinity-argument for the inhomogenous Dirichlet condition may not be sufficiently satisfied. Therefore it may be better to fix the inflow boundary but let the outflow boundary evolve freely. Additionally, future efforts may be focused on the amelioration of the calculation of snow accumulation and erosion.
Finally, the choice of the linear system solver has a high impact on computational costs because, once the matrices (49) are calculated, we are left with the time-stepping and thereby with the solution of the linear system (48).

In the meantime we ran first tests on the mesh with steep topography and on real wind data (data obtained experimentally in a series of measurements in the Swiss Alps). Depending on the wind field the solution of the PDE does not reach a steady state, but since this could be caused by wind data that may be physically incoherent, we cannot yet draw a final conclusion on the general performance of the SUPG-CN method. After all, there are two possibilities to continue with: either one may try to improve the method decribed in this paper or one may adopt a completely different approach to the initial problem. Such a different solution may consist in a Monte-Carlo simulation describing the trajectories of a representatative number of particles within the domain Ω. A major advantage would be that we could simulate the transport of snow in the saltation and the suspension layer in one process instead of applying two coupled simulations. Furthermore, such a method wouldn't reduce the ensemble of snow particles to a fluid and it is less susceptible to non-physical conditions that may arise in insufficient modeling of boundary conditions and wind fields.

References

[1] Stull R. B.: 1988, *An Introduction to Boundary Layer Meteorology*, Kluwer Academic Publishers, Dordrecht

[2] Doorschot J.: 2002, *Mass Transport of Drifting Snow in High Alpine Environments*, SLF, Davos

[3] Johnson C.: 1987, *Numercial solutions of partial differential equations by the finite element method*, Cambridge University Press, Cambridge

[4] Quarteroni A., Valli A.: 1997, *Numercial Approximation of Partial Differential Equations*, Springer

[5] Burden, Faires: 1998, *Numercial Analysis*, 7th edition

[6] Numerical Recipes in C (online version): `http://www.library.cornerll.edu/nr/bookcpdf.html`

[7] Wait R., Mitchell A.R.: 1985, *Finite Element Analysis and Applications*, John Wiley and Sons