UNIVERSITY OF
GOTHENBURG

Master Thesis
45 credits

SICT - A Simple User Interface of a Complex Turbulence Model

Josefina Algotsson

supervised by
PhD. Torsten Linders

February 14, 2017

Abstract

A one equation turbulence model in one dimension has been implemented in a MATLAB graphical user interface. A Simple User Interface of a Complex Turbulence Model, SICT is presented. SICT models the turbulent boundary layer and allows a user with limited MATLAB experience and introductory knowledge about turbulence to handle a fairly sophisticated turbulence model.

Discretization of the k model equations was done using the finite volume method. The construction and validation of the k model is presented along with further validation of SICT against the logarithmic law and against laboratory experiments.

A close correspondence to the logarithmic law was found when modelling a homogenous fluid with constant velocity at the surface. Good accordance with experimental data was found when modelling a linear stratification with constant surface stress. Additionally, validation in a 2-layer stratification showed good results.

Contents

1 Introduction

The study of turbulent flows is crucial in understanding the transport of momentum, heat, and the soluble and suspended substances in the ocean. It's also crucial since turbulent flows occur in most fluid flows encountered in nature.

Turbulence is also found in engineering applications such as behind airplanes, around buildings, on the edges of car windows and in combustion-engines (Versteeg and Malalasekera, 2007). Turbulence in the ocean is found in the top layer of the water column, in the breaking of waves, in the boundary layer close to the bottom and more (Thorpe, 2007).

The application of the current study, however, is the turbulent flow encountered in the turbulent boundary layer in an oceanic flow. The numerical models to describe turbulent flows are numerous and span in complexity, accuracy and computational cost. A common model is the 1D two-equation model (Launder and Spalding, 1974). This paper aims to apply a one-equation turbulence model by Axell and Liungman (2001) to simulate turbulent kinetic energy and solve for the velocity profile as well as the turbulent diffusion coefficients. Furthermore, the model is implemented in SICT; A Simple User Interface of a Complex Turbulence Model, built for this report.

Since turbulence entails fluctuations it's close at hand but inaccurate to refer to turbulence as simply randomness (Kundu and Cohen, 2002). Turbulent flow conserves energy, mass and momentum, which is not necessarily bound to happen in a random time dependent flow field. On the contrary, Thorpe (2007) does exemplify the cascade process as a process associated with a structure which by nature is stochastic.

This paper summarizes theoretical aspects of turbulence, two experiments on entrainment and presents a new graphical user interface for a one-equation turbulence model. The model used in the new graphical user interface was proposed by Axell and Liungman (2001). The study of turbulence is complex, and an important step to reach a greater understanding is approachable banks of knowledge. Some of those I touch on in section 2. I introduce the early turbulence experiments by Reynolds (1883), and summarize some of the most important theoretical aspects of turbulence discussed by Kundu and Cohen (2002), Thorpe (2005) and Cushman-Rosin and Beckers (2011). In addition, I elaborate on the turbulent boundary layer.

In section 3 I cover the entrainment experiments by Kato and Phillips (1969) and Kantha et al. (1977) and the discussion of these by Price (1979). In section 4 I talk about Reynolds decomposition, the closure problem of turbulent modelling schemes and present three common turbulence models. In the same section I present the k-equation proposed by Axell and Liungman (2001) in closer detail. I present my discretisation of the transport and velocity equations in section 5. Section 6.1 shows how I built SICT in the MATLAB application GUIDE. Section 6.2 contains a step-by-step guide to operating SICT. Finally, I use the logarithmic law and the findings of Kato and Phillips (1969), Kantha et al. (1977) and Price (1979) from section 3 to validate my implementation of the k model in section 7.

2 Turbulence theory

Reynolds iconic experiment in 1883 constitutes a possible point of entry to the study of turbulence. The experiment involved a pipe flow in which a thin stream of dye was added in the centerline of the pipe flow. He initiated the experiment with allowing a stream of water pass through a pipe and "When the velocities where sufficiently low, the streak of colour extended in a beautiful straight line through the tube [...]"(Reynolds, 1883, p. 91). He then increased the velocity in stages and found that "[...] at some point in the tube [...] the colour band would all at once mix up with the surrounding water, and fill the rest of the tube with a mass of coloured water [...]"(Reynolds, 1883, p. 92). Reynolds found that this would occur when a ratio reaches above a certain value. That ratio is now called the Reynolds number

$$Re = \frac{Ud}{\nu}, \tag{1}$$

where d is the diameter of the tube, U is the mean velocity of the flow and ν is the kinematic viscosity of the fluid. The Reynolds number relates the inertia (U and d) to the friction forces (ν). This value, or interval rather, for which small perturbations have the potential to grow into instabilities and turbulence, has been re-evaluated and still a single critical value has not been determined. It is usually referred to as around 2000-3000 (Kundu and Cohen, 2002). For example, the Reynolds number in a tube, where the mean velocity of water is 10 cm/s, the diameter of the tube is 2 cm and the kinematic viscosity of water is 10^{-6} m^2/s gives a Reynolds number of 2000. A laminar flow in a tube, around a cylinder or over a flat plate can gradually transform from a smooth flow with small Reynolds numbers to a flow sometimes shedding eddies from the flow in a regular manner into a flow in which turbulence (perhaps a turbulent wake) is present (Thorpe, 2005), given that the inertia forces start to empower the friction forces.

The Reynolds number gives us a first taste of the length scale important in the study of turbulence. Since Reynolds' experiments and rich contribution to the oceanographic field of science further understanding (although still incomplete) has been reached. We will return to the concept of length scale and now focus on a more in-depth description of turbulence.

Due to its irregular nature, there is no clear definition of turbulent flow. Perhaps the most important and characteristic trait of a turbulent flow is the dissipation of kinetic energy (Thorpe, 2007). The turbulent flow also works in an interaction with eddies and diffusion. Eddies initially stir the fluid creating stronger gradients, which increases dissipation and diffusion resulting in mixing of the fluid. Despite its irregular nature there are some predictions we can make, if we first study some of its traits. Traditionally, turbulence is said to have certain characteristics (Kundu and Cohen, 2002; Thorpe, 2005; Cushman-Rosin and Beckers, 2011). The paper at hand puts emphasis on processes not influenced by the coriolis force.

Irregularity The nature of a turbulent flow is stochastic, irregular and chaotic. The flow consists of large eddies, or vortices, which have the size-scale of the flow geometry.

Despite the chaotic nature of turbulent flow it can be mathematically described by the Navier-Stokes equations. The irregularities in a turbulent flow is often referred to as fluctuations. The fluctuations are the variations around a mean value in a property. These fluctuations make it practically impossible to resolve all the turbulent scales whilst trying to predict a turbulent flow i.e when modelling turbulent flow. However, it is seldom interesting or relevant to resolve all the turbulent scales in most common geophysical applications, which usually have a larger timescale than the turbulent fluctuations. The Reynolds decomposition (explained later in this section) is useful in dealing with this issue but also creates the Reynolds stresses, which in turn calls for a closure solution to the turbulent scheme.

Diffusivity Turbulence causes exchange of momentum. Due to the exchange of momentum the diffusivity, or spreading rate, increases as turbulence increases.

Large Reynolds numbers Non linearity and high Reynolds numbers are characteristic for turbulent flow. Where velocities are high in relation to molecular viscosity, turbulence is bound to happen. This means that small perturbations have the potential to grow and cause larger disturbances in the flow.

Three-dimensional Due to the vortex stretching and tilting characteristic of turbulent flow, the turbulent flow is three-dimensional.

Dissipation Energy in the mean flow is transferred to large eddies. The vortices divide into smaller eddies of ever decreasing size. For eddies of smaller and smaller scales the viscous stresses have an increasing impact, eventually dissipating (destroying) the eddies, transforming the turbulent kinetic energy (T.K.E) into internal energy, i.e. heat.

Reynolds averaged Navier Sokes equations

Let us consider a turbulent shear flow to study momentum exchange. This is simply a system, where turbulence is present and there is a mean positive velocity gradient in the z-direction with water flowing in the x-direction. As turbulent eddies are present in this flow, small packages of water with higher x-momentum will be brought down into areas with lower momentum. And small packages of water will leave an area of lower x-momentum into areas of higher x-momentum. This is called momentum exchange which results in smoothing of the velocity gradient; i.e. deceleration of faster moving fluid and acceleration of slower moving fluid. This process results in additional shear stresses called Reynolds stresses. It can be regarded as an additional stress on the mean flow, by the turbulent fluctuations, or simply the momentum exchange through turbulent fluctuations (Kundu and Cohen, 2002).

Turbulence is fully described by the Navier Stokes equations. However, the computational resources to resolve the scales necessary when modelling turbulent fluctuations are immense. By decomposing the instantaneous velocity described by the Navier Stokes equations into a mean and fluctuating component, and then averaging the Navier Stokes equations, a set of equations can be used to at least describe the mean state of flow properties, whilst still accounting for turbulence.

The Reynolds stresses described in the beginning of this section also appear mathematically when performing the Reynolds decomposition and time averaging of the Navier Stokes equations. The principal stages of Reynolds decomposition and averaging the Navier Stokes equations are presented below.

The Reynolds decomposition starts with considering the Navier-Stokes equation and the continuity equation. In a one-dimensional case the instantaneous Navier-Stokes equations read

$$\frac{\partial \tilde{u}}{\partial t} + \tilde{u}\frac{\partial \tilde{u}}{\partial x} + \tilde{v}\frac{\partial \tilde{u}}{\partial y} + \tilde{w}\frac{\partial \tilde{u}}{\partial z} = -\frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x} + \frac{\partial}{\partial x}\left(\nu\frac{\partial \tilde{u}}{\partial x}\right), \tag{2}$$

where the first term on the left hand side is the time dependent (transient) term, second to fourth terms are nonlinear convection terms, the first term on the left hand side is the pressure term, and the last term is divergence of stress or diffusion of momentum. The continuity equation reads

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

The instantaneous velocity $\tilde{u}$ is then decomposed into a mean velocity U and a fluctuating u'. The same is done for pressure; $\tilde{p} = P + p$. The Navier-Stokes equations with the decomposed variables are then time averaged. Using the rules which govern the averaging of fluctuating properties $\tilde{u} = U + u'$ and $\tilde{v} = V + v'$;

$$\begin{aligned}
\overline{u'} &= \overline{v'} = 0 \\
\overline{U} &= U \\
\overline{\frac{\partial \tilde{u}}{\partial s}} &= \frac{\partial U}{\partial s} \\
\overline{\tilde{u}\tilde{v}} &= UV + \overline{u'v'} \\
\overline{\tilde{u}V} &= UV \\
\overline{u'V} &= 0
\end{aligned} \tag{4}$$

gives the Reynolds Averaged Navier Stokes, RANS, equation:

$$\frac{\partial U}{\partial t} + \nabla(U\mathbf{U}) + \nabla(u'\mathbf{u}') = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\nabla^2 U \tag{5}$$

The Reynolds stresses (third term on the left hand side) are the fluctuating velocity components multiplied. These are, unlike the other terms, new and arose from the Reynolds averaging. They are unknown and treated as an additional stress on the mean velocities.

So how do we handle this unknown stress? The closure problem is solved with turbulence modelling which is handled in section 4. In the following two sections, attention is turned to T.K.E, dissipation, $\mathcal{E}$ of this energy, and the energy cascade. The transport equations for T.K.E and for dissipation namely the k-equation and the $\mathcal{E}$-equation are described in section 4.

2.1 Energy cascade

T.K.E is denoted k and has unit $[m^2/s^2]$ or energy per unit mass. The dissipation of T.K.E is denoted by $\mathcal{E}$ and has unit $[m^2/s^3]$ or energy per unit mass and unit time. The smaller the eddies, the bigger the relative impact from friction forces is on the eddies. However, the theory of the cascade process is not perfect; not all energy is passed directly from larger to smaller eddies, but dissipated also before reaching the smallest scales. About 90 % of the energy feeded into the large eddies from the mean flow is dissipated at the smallest scales. A free turbulent flow is unaffected by viscous forces. The same is not true for a wall bounded shear flow.

2.2 The turbulent boundary layer

The turbulent flow we now turn to is the turbulent boundary layer over a flat surface. As water flows over a surface, a seabed for example, it exerts a stress on the surface, and the surface in turn exerts a stress on the water. The stress exerted on the fluid creates a shear, which in turn is what sustains the turbulence together with the energy extraction from the mean flow by the large eddies. The production of T.K.E from the shear, and the dissipation of that energy through viscous forces are the two major factors that rule the steady state balance of turbulent energy.

The large scales (the size of the largest eddies) are of the order of the flow geometry, for example, the thickness of the boundary layer. These large scales extract energy from the mean flow.

The velocity of the flow closer to the seabed goes towards zero. This is where the viscous forces overpower the kinetic energy in the turning eddies. At the hard boundary the velocity is zero; there is a no-slip condition. Figure 1 shows the logarithmic velocity profile, typical for a turbulent boundary layer.

In the turbulent boundary layer close to a wall, both Reynolds stresses and viscous stress are important but in different regions with respect to the distance from the wall. Far away from the wall, the stress exerted on the fluid is dominated by Reynolds stress. Close to the wall, in an extremely thin layer, viscous stress dominates. These regions in the turbulent boundary layer are called the outer and inner region, respectively.

Below, two scaling laws applicable in these regions are presented briefly. Subsequently, the logarithmic law is deduced.

In the inner layer, the mean velocity profile, U, is independent of the free stream velocity,

U_∞, and of the boundary layer thickness. It depends on the parameters relevant near the wall. Therefore,

$$U = U(\rho, \tau_0, \nu, y) \tag{6}$$

where ρ is the density of the fluid, τ_0 is the shear stress at the wall, ν is the kinematic viscosity of the fluid and y is the distance from the wall. To rid the system of the unit mass we introduce the friction velocity,

$$u_* = \sqrt{\frac{\tau_0}{\rho}}. \tag{7}$$

These four variables (U, ν, y, u_*) create the law of the wall,

$$\boxed{\frac{U}{u_*} = f\left(\frac{yu_*}{\nu}\right) = f(y^+).} \tag{8}$$

This accounts for an extremely thin layer, where the stress can be considered constant and equal to the wall shear stress τ_0.

The outer region is just like the inner region affected by the friction velocity and the distance from the wall, however, viscosity ν is not important. Instead, the boundary layer thickness δ enters the equation and

$$U = U(u_*, \delta, y). \tag{9}$$

The velocity distribution is described by the velocity defect law

$$\boxed{\frac{U - U_\infty}{u_*} = F\left(\frac{y}{\delta}\right) = F(\xi).} \tag{10}$$

The logarithmic law

The region which can be modelled in the present case is the overlap region between the outer and inner layer, where the logarithmic law rules. This logaritmic law can be deduced in at least two ways. The law of the wall, eq. (8), and the velocity defect law, eq. (10), can be consolidated by first deriving each law

$$\frac{dU}{dy} = \frac{u_*^2}{\nu}\frac{df}{dy^+} \tag{11}$$

and

$$\frac{dU}{dy} = \frac{u_*}{\delta}\frac{dF}{d\xi}. \tag{12}$$

By equating eq. (11) and (12) and multiplying by y/u_* the following is obtained:

$$y^+\frac{df}{dy^+} = \xi\frac{dF}{d\xi}. \tag{13}$$

Since the left side of equation (13) must be a function of y^+ and the right hand side must be a function of ξ, we make the assessment that both sides must be equal to a universal constant such as the experimental constant $1/\kappa$ where $\kappa \approx 0.4$ so,

$$y^+ \frac{df}{dy^+} = \xi \frac{dF}{d\xi} = \frac{1}{\kappa} \tag{14}$$

Lastly, integrating the expressions gives a logarithmic profile.

$$\begin{aligned} f(y^+) &= \frac{1}{\kappa} \ln(y^+) + A \\ f(\xi) &= \frac{1}{\kappa} \ln(\xi) + B \end{aligned} \tag{15}$$

The logarithmic law can also be deduced by dimensional arguments alone. In the logarithmic region, the relevant length scale, the distance to the wall, and the relevant velocity scale, u_*, are used to relate the velocity gradient by

$$\frac{dU}{dy} = \frac{u_*}{\kappa y} \tag{16}$$

Integrating this to find the velocity profile gives

$$U = \frac{u_*}{\kappa} \ln y + C \tag{17}$$

where C is a constant of integration. In natural systems, the wall surface at hand is not smooth however. In natural systems the roughness elements protrude through the viscous sublayer and the velocity profile is assumed to reach zero somewhere within the roughness elements. When introducing roughness to the wall surface, the velocity profile close to the wall will reach zero just above the roughness elements. Therefore, eq. (17) applies to $z > z_0$ if z_0 is the height of the roughness elements where the velocity reaches zero. When evaluating the integration constant C, we attain the logarithmic law:

$$U = \frac{u_*}{\kappa} \ln \frac{z}{z_0}. \tag{18}$$

3 Laboratory experiments

One way of studying the turbulent processes is to study the deepening of the mixed layer depth, the MLD, through turbulent entrainment. This entrainment is not easily modelled and needs to be parameterized. Laboratory experiments are often beneficial since it provides a simple, repeatable and mostly precise and cheap setup. Natural observations are often affected by diabatic heating and advection, which contort entrainment. Apart from the gainly aspects of laboratory experiments in entrainment, it is inevitable that the outcome

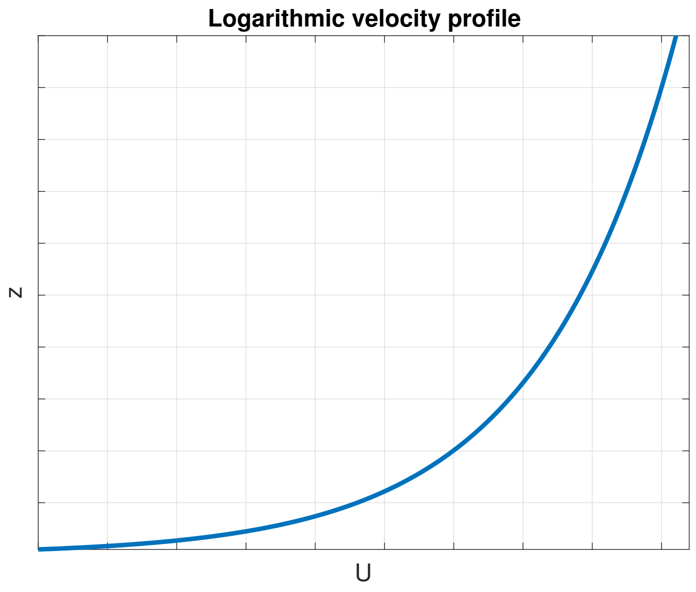


Figure 1: Logarithmic velocity profile typical for the turbulent boundary layer.

could describe a strict case despite the idea or process at hand being a general one (Price, 1979). Early entrainment lab experiments involved stirring the surface layer in a tank of stratified water by help of paddles or a grid. These experiments did not involve a velocity shear or a mean flow.

In this section, I present some important stages in the development of entrainment scalings. I start with the entrainments experiments by Kato and Phillips (1969) and the very similar but still different experiment by Kantha et al. (1977) (hereafter KP and KPA). Then, I study the summarizing article by Price (1979) which analyzes the entrainment experiments by KP and KPA in search for a general scaling of entrainment as well as considering the effect of wall stress in these experiments. The results will be used in validation of SICT in section 7.

3.1 Kato and Philips

Kato and Phillips (1969) describe earlier experiments on entrainment, such as Rouse and Dodu (1955) and Ellison and Turner (1959), as useful, but hard to interpret as they where conducted using mechanical agitation of the upper layer. However, turbulence in the ocean is usually created in different ways. Furthermore, Kato and Phillips (1969) argue that the scale of generated turbulence isn't easily palpable.

Instead, an experiment with a known surface stress was conducted and the depth of the mixed layer was measured. An annular (cylindical) non-rotating tank with a rotating screen exerting stress on the water surface was set up. The tank had a depth of 28.0 cm and an outer and inner diameter of 152.4 and 106.7 cm respectively making the width of the channel 22.8 cm. The tank was filled to a depth of 23 cm with density stratified fluid by filling the tank from the bottom through a ring of copper tubing with several holes. Increasingly salty water was let in to construct 20 layers of water of increasing density. One of the layers was dyed to assure that the mixing caused by the injection of new water and by the removal of the copper tubing was small enough to neglect. Molecular diffusivity was left to even out the stratification for 2.5 hours.

There were three different density gradients in the experiment ranging from 1000 to 1042, 1000 to 1084 and 1000 to 1092 kg/m^3. The water in the tank was illuminated from the bottom through a narrow window and observations of the fluid could be made through a curved window set in place in the outer wall of the tank. To control the depth of the mixed layer, Fluorescein dye was added into the water around the annulus. In addition, a mixture of benzene and carbon tetrachloride with different densities were added in drops. The rotating screen was set in motion exerting a stress on the water surface. As the mixed layer deepened, the drops would be present in the dyed water either at the surface or floating in the interface of undyed and dyed water indicating that the dye was in fact sufficient to show the mixed layer depth. The drops below the dyed water would sit on their respective equilibrium level.

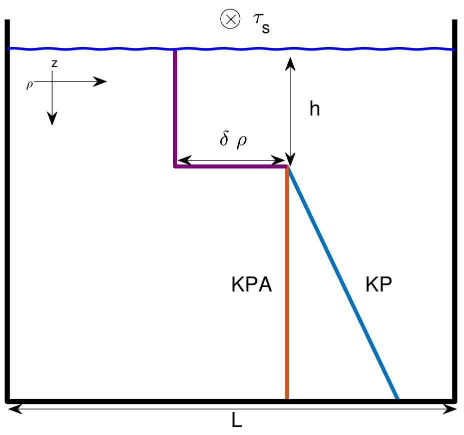
In the early stages of the entrainment experiment, KP describe the occurrence of evenly spaced eddies within the first few seconds of the experiments. Their axis laid cross channel and the spacing was of double the depth of the layer. The eddies persisted for about 1 to 2 seconds entraining uncoloured fluid, which would then diffuse through the upper layer. If the stratification was stronger, the eddies would be smaller and more closely spaced.

As the layer depth increased, the Reynolds number increased. The same goes for the increase of the velocity U of the screen. The measurements, taken 20 seconds and forward from the start of the experiment until the time at which the mixed layer had reached a depth of about 55 % of the width of the channel, were used in the study to find an entrainment rate. This was to avoid the uncertainty of the initial surface stress and to ensure that the measurements were as little as possible affected by wall stress. They reasoned that the ratio, E, of the entrainment velocity, u_e, to the friction velocity, u_*, should depend on three variables; the friction velocity, the depth, D, of the mixed layer and the fractional change in buoyancy $g\delta\rho/\rho_0$ at the interface. The variables were expressed in the dimensionless overall Richardson number originally defined by Ellison and Turner (1959):

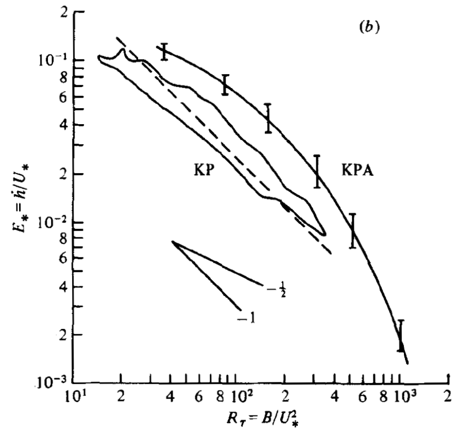
$$Ri_0 = \frac{g\delta\rho D}{\rho_0 u_*^2}, \tag{19}$$

where ρ_0 is the maximum density, and where the density jump in the interface is

$$\delta\rho = \frac{1}{2}\left(\frac{\partial\rho}{\partial z}\right)_0 D \tag{20}$$



(a) Cross section of annular tank in the KP and KPA stratification setup, adapted from Price (1979).



(b) The KP and KPA results. KP data shown in the cloud. Dashed line is the $E_* = 2.5R_\tau^{-1}$ fit to the KP data. Figure adapted from Price (1979).

where $(\delta\rho/\delta z)_0$ is the initial stratification. Insertion of equation (20) into equation (19) gives

$$Ri_0 = \frac{g(\delta\rho/\delta z)_0 D^2}{2\rho_0 u_*^2}. \tag{21}$$

Which by dimensional ground gives

$$E = \frac{u_e}{u_*} = f(Ri_0) \tag{22}$$

Kato and Phillips (1969) plotted the data with this function. The experimental data seemed to follow this relationship closely with only little scatter and deviation from eq. (22).

The experiments by Ellison and Turner (1959) used the mechanical agitation frequency and the mixed layer depth to express a velocity to implement in the Richardson number, and found that the entrainment velocity is then proportional to Ri^{-1}. Kato and Phillips (1969) later adopt these results reasoning that "[...] one would expect the functional forms to be the same provided the same processes were involved in each case." (Kato and Phillips, 1969, p. 652). This would enable the ratio of the entrainment coefficient, E to be expressed as

$$E = \frac{u_e}{u_*} = 2.5(Ri_0)^{-1} = 2.5\frac{\rho_0 u_*^2}{g\delta\rho D}. \tag{23}$$

where there is an uncertainty to the numerical constant of about 30%.

3.2 Kantha, Philips, Aszhad

The experiments by Kantha et al. (1977) adapted large parts of the KP experiments. The same annular tank of " [...] somewhat poorer condition [...]" was used (Kantha et al., 1977, p. 756). The initial form of stratification is the most notable difference between the experiments, as a 2-layer stratification was used in this case, with density difference $\delta\rho$ between the two layers. The tank was filled in a similar way capping denser water with a approximately 5.4 cm thick layer of fresh water. The denser water containing common salt or calcium chloride, the latter being used in order to attain a higher Ri without having to increase the depth of the initial surface layer or decrease the friction velocity.

In this experimental setup $\delta\rho D$ is constant due to the conservation of mass which means the overall Richardson number is also constant. Thus, the entrainment velocity is also constant. This will provide one measurement in the relation between friction velocity and entrainment velocity for each experimental run. Or, in other words, several experiments had to be run in order to establish a valuable dataset on the relation $u_e/u_* = f(Ri)$.

For low Richardson numbers, convolutions in the density interface was more prominent, and so a mean depth was taken from a couple of time steps in order to dial down the variation. However, large scatter remained compared to scatter in higer Richardson number experiments. Richardson numbers in the experiments ranged from 30 to 1100 where the Richardson number is

$$Ri_0 = \frac{g\delta\rho_i D_i}{\rho u_*^2}. \tag{24}$$

and where D_i is the depth of the mixed layer in the beginning of the experimental runs, and $\delta\rho_i$ is the initial density difference between the two layers.

KPA summarizes their findings with stating that the entrainment rate depends on two variables, namely

$$E = u_e/u_* = f(Ri, D/W) \tag{25}$$

where W is the width of the channel. Although KPA make a respectable effort towards dealing with the influence of the walls, neither this or the KP experiments resolve the issue. Furthermore, the KPA experiments fail to find a reasonable theory to explain the $E = E(Ri)$ relationship.

3.3 Price

Price et al. (1978) suggested that a proper scaling velocity in the experiments was the mean velocity and not friction velocity as discussed in KP and KPA.

Price (1979) seeks to deal with the influence of the walls in the aforementioned experiments by KP and KPA and, furthermore, to find a general form for the entrainment rate more universally applicable. The conservation law of mean momentum and buoyancy coupled with the KP and KPA observations was used to compute the entrainment law $E(R_v)$:

$$\frac{1}{V}\frac{dh}{dt} = E(R_v), \tag{26}$$

where h is the depth of the mixed layer, V the mean velocity in the mixed layer and $R_v = B/V^2$, where B is the total mixed layer buoyancy. The entrainment law found successfully collapsed the KP and KPA data. To find this, Price (1979) computed the mean velocity in the KP and KPA experiments and used a model momentum conservation equation to compute a function E(R$_v$) for the KP and KPA experiments. Although Price (1979) finds accordancy between these findings as well as the Ellison and Turner (1959) and Price et al. (1978) results, Price (1979) works the issue further to determine the effect of side wall drag and to consider the form of the stratification in close detail.

He uses E(R$_v$) to predict E$_*$ (the entrainment ratio dependent on the friction velocity). In conclusion, Price (1979) finds an entrainment law applicable to a wider range of circumstances and effectively does that with consideration of the influence of walls. Price finds that the entrainment law with zero wall stress is

$$E_* = \frac{\partial h}{\partial t}\frac{1}{u_*} = nR_v^{1/2}R_\tau^{-1/2} \tag{27}$$

where R_v is 0.6 for a steady forcing and n is either 0.5 or 1 depending on if the stratification is linear or two-layer configured. R_τ is a bulk Richardson number.

Inserting $R_\tau = B/u_*^2$ and $B = 0.5N^2h^2$ into eq. (27) and simplifying gives us the depth of the mixed layer in the case of a linear stratification

$$h_{KP} = 1.047u_*(t/N)^{1/2} \tag{28}$$

where N is the buoyancy frequency and t is the time. For a 2-layer stratification Price (1979) finds that the deepening of the mixed layer depth is

$$h'_{KPA} = u_*^2\sqrt{R_v/B} \tag{29}$$

where h'_{KPA} indicates the time derivative of the mixed layer depth, R_v is 0.6, and B is

$$B = gh_0\delta\rho_0/\rho \tag{30}$$

where g is the gravitational acceleration, h_0 is the thickness of the initial mixed surface layer, $\delta\rho_0$ is the density difference between the two layers and ρ is the density of the bottom layer.

4 Modelling turbulence

The vast range of turbulence applications calls for methods to model the effect of turbulence. There are three main ways of simulating turbulent flows (Versteeg and Malalasekera, 2007; Kundu and Cohen, 2002):

Models for RANS equations Reynolds decomposition and time averaging of the Navier-Stokes equations gives an insight to the effect of turbulence on the mean flow and the mean flow properties. Due to the interactions of fluctuating velocities in different directions, extra terms, the Renolds stresses, appear and must be modelled. A common model is the $k - \mathcal{E}$ model. This type of modelling gives satisfactory accuracy, with an acceptable demand of computer resources.

LES (Large eddy simulation) Amounts to separating large from small eddies prior to computing the Navier-Stokes equations. Gives a more sophisticated solution to the CFD problems, but has a higher demand on computer resources.

DNS (Direct numerical simulation) DNS has the power to simulate the mean flow as well as the velocity fluctuations, by using a time and spatial scale sufficiently fine to resolve the scales of dissipation. This technique demands immense computational resources making it unfitting for industrial use.

When modelling turbulence with RANS it is assumed that the Reynolds stresses act like a viscous stress on the mean flow. According to the viscosity laws of Newton, the rate of deformation of the fluid is proportional to the viscous stresses. This gives

$$\tau_{xy} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \tag{31}$$

where μ is the molecular viscosity. In addition to this, the mean rate of deformation is assumed to be proportional to Reynolds stresses (Boussinesq, 1877) which gives

$$\tau_{xy} = -\rho \overline{u'v'} = \nu_t \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) - \frac{2}{3} \rho k \delta_{xy} \tag{32}$$

where k is the turbulent kinetic energy per unit mass, and δ_{xy} is the Kronecker delta which is 1 if $i = j$ and 0 if $i \neq j$. (Versteeg and Malalasekera, 2007). Before going into turbulence closure, let us sum up what system of equations are relevant to the present study:

I assume a 1D case where all variables are homogenous in horizontal extent. This means advection as well as the pressure gradient, can be neglected from equation (5) . So the U equation reads

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial z}\left[\nu_{tot}\frac{\partial U}{\partial z}\right] \tag{33}$$

where ν_{tot} is the sum of the eddy viscosity, ν_t and the laminar viscosity ν. The transport equations for scalars such as salinity and temperature without source terms are

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial z}\left[\nu'_{tot}\frac{\partial S}{\partial z}\right] \tag{34}$$

and

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left[\nu'_{tot}\frac{\partial T}{\partial z}\right] \tag{35}$$

respectively, where ν'_{tot} is the total eddy diffusivity; the sum of the eddy diffusivity ν'_t and the laminar diffusivities ν_H or ν_S respectively. The eddy viscosity ν_t and the eddy diffusivity ν'_t are the key to turbulence closure, and handled in the following section.

4.1 Turbulence Closure

Before going into my implementation of the one-equation k model, a general description of a mixing length model, a $k - \mathcal{E}$ model and the k model will be described as closure to the turbulent scheme.

Mixing length model

The mixing length model uses no extra transport equations for closure, but an algebraic expression to calculate turbulent viscosity. The model is built on the assumption that the turbulent viscosity can be taken as the product of a turbulent length scale (ℓ) and a turbulent velocity scale v. The length scale ℓ is taken as the characteristic length scale of the largest eddies since these are the eddies with the most kinetic energy, and which interact with the mean flow. Dimensional analysis gives

$$\nu_t = Cv\ell \tag{36}$$

where C is a dimensionless constant. The mixing length model also utilizes the assumption that there is a strong connection between the mean flow and the character of the largest eddies by linking the mean velocity gradient and the velocity scale of the eddies by

$$v = c\ell\left|\frac{\partial U}{\partial z}\right| \tag{37}$$

where c is a dimensionless constant. Using eq. (36) and eq. (37) gives

$$\nu_t = \ell_m^2\left|\frac{\partial U}{\partial z}\right| \tag{38}$$

where ℓ_m is the new length scale absorbing constants C and c. In some cases, such as in free turbulent flow and flat plate boundary layer or pipe flow, this is sufficient. Depending on the flow the length scale is modified to account for the specific case. This model is well established and gives nice predictions for thin boundary layers (Versteeg and Malalasekera, 2007).

$k - \mathcal{E}$ model

This two-equation model handles processes that have effect on turbulent kinetic energy and sets up transport equations for turbulent kinetic energy k and for dissipation of turbulent kinetic energy $\mathcal{E}$. Although this model is widely used, the exact $\mathcal{E}$ equation has many unmeasurable and unknown terms. It is also computationally expensive. The standard $k - \mathcal{E}$ model was set up by Launder and Spalding (1974) and uses the transport equations for k and $\mathcal{E}$. Axell and Liungman (2001) expresses the k-equation as

$$\boxed{\frac{\partial k}{\partial t} = \frac{\partial}{\partial z}\left(\frac{\nu_t}{\sigma_k}\frac{\partial k}{\partial z}\right) + P_s + P_b - \mathcal{E}.} \tag{39}$$

where P_s is the production of T.K.E through shear and P_b is the production or destruction of T.K.E through buoyancy. The $\mathcal{E}$ equation reads

$$\frac{\partial \mathcal{E}}{\partial t} = \frac{\partial}{\partial z}\left(\frac{\nu_t}{\sigma_{\mathcal{E}}}\frac{\partial \mathcal{E}}{\partial z}\right) + \frac{\mathcal{E}}{k}(c_{\mathcal{E}1}P_s + c_{\mathcal{E}3} - c_{\mathcal{E}2}\mathcal{E}) \tag{40}$$

where $c_{\mathcal{E}1}, c_{\mathcal{E}2}$ and $c_{\mathcal{E}3}$ are empirical constants and $\sigma_{\mathcal{E}}$ is the Schmidth number for $\mathcal{E}$ (Axell and Liungman, 2001). In the $k - \mathcal{E}$ model the k and $\mathcal{E}$ terms are used to determine the eddy viscosity and through dimension analysis we obtain:

$$\nu_t = \rho C_\mu \frac{k^2}{\mathcal{E}} \tag{41}$$

where C_μ is dimensionless constant (Versteeg and Malalasekera, 2007).

k model

Axell and Liungman (2001) put forward two compelling reasons to depart from the most common two-equation model for a simpler yet satisfying solution of the one-equation k model. The first reason is the lower demand that the model puts on computational resources.

The second and perhaps stronger argument is that, although the modelled equation for k is built on physically solid ground, the modelled equation for the transport of dissipation is not and does involve some rather complicated terms such as dissipation of dissipation.

One could argue for a solution similar to the one Mellor (1982) puts forward, where length scale is found by using the transport equation for turbulence intensity and $2kl$.

However, the computational demand withstands. The k model simulates the turbulent kinetic energy of equation (39) where the production through shear is

$$P_s = \nu_{tot} \left[\left(\frac{\partial u}{\partial z} \right)^2 + \left(\frac{\partial v}{\partial z} \right)^2 \right], \tag{42}$$

and the production through buoyancy is

$$P_b = -\nu'_{tot} N^2. \tag{43}$$

N is the buoyancy frequency,

$$N^2 = -\frac{g}{\rho_0} \frac{\partial \rho}{\partial z}. \tag{44}$$

where g is the gravitational acceleration, ρ_0 is a reference density and ρ is the density of the fluid. The turbulent viscosity ν_t and the turbulent diffusivity ν'_t are described by

$$\nu_t = c_\mu k^{1/2} l \tag{45}$$

and

$$\nu'_t = c'_\mu k^{1/2} l \tag{46}$$

respectively. c_μ and c'_μ are stability functions described in Appendix B. The k model uses an algebraic expression for the turbulent length scale, l, to close the turbulent scheme. The length scale is related to dissipation by

$$l = (c_\mu^0)^3 \frac{k^{3/2}}{\mathcal{E}} \tag{47}$$

where c_μ^0 is a proportionality constant required in order for the logarithmic law to be fulfilled (Burchard and Baumert, 1995).

Axell and Liungman (2001) presents the assumption that l should be determined locally by a geometric length scale and a buoyancy length scale in each instant. The connection between the length scale l and turbulence quantities will be through k in the buoyancy length scale. The geometric length scale is

$$l_g = \kappa(z + z0) \tag{48}$$

where κ is the von Karman constant with value 0.4, z is the distance from the bottom, and z_0 is the roughness length. The buoyancy length scale is

$$l_b = c_b \frac{k^{1/2}}{N} \tag{49}$$

where $c_b = 0.35$ is a model constant of proportionality. However, this expression only holds up away from walls and for strong, stable stratification. The expression

$$\frac{1}{l^2} = \frac{1}{l_g^2} + \frac{N^2}{c_b^2 k} \tag{50}$$

is desirable on the ground that it gives the law of the wall, eq. (8) in case of neutral conditions, if $N^2 = 0$. Therefore, it will limit the size of the largest eddies close to the wall. Also, if stratification is stable, if $N^2 > 0$, then the length scale l will go towards eq. (49). The result is a dampening of T.K.E by limiting the size of the largest eddies which have to work against the stratification and against gravity. The opposite happens in an unstable situation, if $N^2 < 0$, where l will be large and T.K.E will be promoted. If there is a strong stable stratification, if $N^2 < -c_b k/l_g^2$, eq. (50) will yield a negative l which is unrealistic. Multiplying eq. (50) by $l^2 l_g^2$ on both sides solves this problem and

$$l^2 = l_g^2 - \frac{l^2 l_g^2 N^2}{c_b^2 k} \tag{51}$$

where l on the right hand side will be taken as eq. (47) composed of values from the old time step in SICT. In summary, in a stable stratification, $N^2 > 0$, eq. (50) is used. In an unstable stratification, $N^2 < 0$, eq. (51) is used.

Axell and Liungman (2001) verify the k model using five techniques; Using the logarithmic law, using the Monin-Obukhov similarity, using simulation of idealized entrainment laboratory experiments, and simulations of two cases of historical observations. Three of these are summarized below.

Validating with the logarithmic law, Axell and Liungman (2001) test if the proposed k-equation eq. (39) in a non buoyant shear flow fulfills the logarithmic law. A non rotating steady state situation without pressure terms or influence from the surface boundary is applied to the x-momentum equation. Then, a region of constant stress close to the boundary is assumed as well as constant T.K.E in the region. Axell and Liungman (2001) find that the equations at hand reduce to the logarithmic law

$$U = \frac{u_*}{\kappa} \ln\left(\frac{z + z_0}{z_0}\right) \tag{52}$$

where z_0 is the roughness parameter of the wall (see appendix B).

The Monin-Obukhov similarity is the empirical relations between the gradients of temperature and velocity to the Monin-Obukhov length. Axell and Liungman (2001) look at an analytical solution to the model equations by assuming steady state, negligible turbulent transport of k, an infinitely deep fluid, and a constant momentum flux near the boundary. This was to examine how the proposed k-equation along with the model constants perform when simulating a Monin-Obukhov similarity relations. A fair agreement with the empirical data was found, however, the velocity profile is somewhat underestimated. Identical results were produced when using numerical simulation with the full k-equation, which strengthens the fact that the results were in fair agreement with empirical data. Furthermore, similar results were attained when using the same stability functions with the $k - \mathcal{E}$ model.

When comparing the model's performance with a simulation of free convection in a laboratory experiment, Axell and Liungman (2001) use the experimental results by Deardorff et al. (1969) where a vessel of water with a thermal stratification was heated from the bottom. The temperature profile was measured continuously. Both the k and $k - \mathcal{E}$ models were used to simulate the laboratory case, and both showed reasonable agreement with the data. The largest difference was that whilst the experimental measurements showed neutral or stable stratification, the simulations showed neutral or unstable stratification. The k model appeared also to indicate stronger mixing earlier than the $k - \mathcal{E}$ model.

4.2 Boundary conditions

The boundary condition for k at the bottom is no flux in accordance with the law of the wall. The north boundary condition for k without buoyancy flux at the boundary is parametrized as

$$k = \frac{u_*^2}{(c_\mu^0)^2} \tag{53}$$

Boundary conditions for T and S are hardcoded in SICT to account for no flux at both bottom and surface. The boundary condition for u, however, is open for the user of the UI to determine. The user can input a friction velocity (i.e determine a flux or stress) at the surface or set a fixed value of U.

5 Discretization

There are four different equations to account for when running the turbulence model. First off, we want to simulate velocity U, which is then used to model T.K.E, k. The scheme is closed with an algebraic expression for the length scale. Furthermore, temperature, T and salinity, S, are also simulated using the turbulent diffusivity yielded in modelling k. We find the transport equations for properties U, k, S and T in equations (33), (39), (34) and (35) respectively. Aside from two of them using a turbulent viscosity, instead of a turbulent diffusivity, and in addition to having some different source terms, the four equations are structured similarly. The discretisation (or parametrization) of these transport equations can therefore be described by discretizing a general property σ and can then modified according to each unique transport equation.

The finite volume method of discretizing the tranport equations starts with defining the control volume. Figure 2 shows the definition of the control volume, P, with grid size Δz. Node P is neighboured by node N to the north and node S to the south. The interfaces between the nodes are called the faces, n and s respectively.

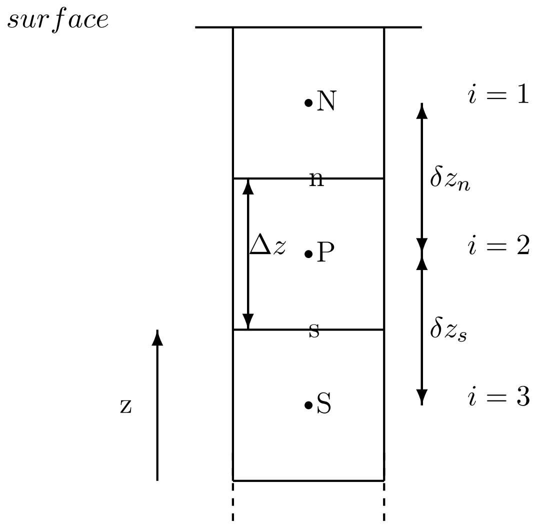


Figure 2: Illustration of the control volume with node P and neighbouring nodes N and S. The nodes are placed exactly half way between the faces. The faces are denoted n (for the north face of P) and s (for the south face of P). The matrix notation of nodes is also visible to the right of the control volume.

The general transport equation of property σ reads

$$\boxed{\frac{\partial \sigma}{\partial t} = \frac{\partial}{\partial z}\left(K\frac{\partial \sigma}{\partial z}\right) + S} \tag{54}$$

where K is the diffusion coefficient or viscosity and S is the source term, not salinity. According to the finite volume method we integrate this equation over its control volume P from face s to face n. Since the transport is time dependent, values not only change in space but also in time. We have to integrate with respect to time. We integrate from time t to $t + \Delta t$. Altogether, this constitutes a two-dimensional integration for a one-dimensional time dependent problem.

5.1 Integration of transient term

On the left hand side (hereafter LHS) of eq. (54), we start with the temporal integral and continue with the spatial integral which reads

$$\int_s^n \int_{t^0}^{t^0+\Delta t} \frac{\partial \sigma}{\partial t} dt dz = \int_s^n (\sigma^1 - \sigma^0) dz = \Delta z(\sigma_P^1 - \sigma_P^0) \tag{55}$$

where index ^{1} indicates the value at the new time step, $t = t^0 + \Delta t$, and index ^{0} indicates the value at the old time step, $t = t^0$. The last operation of eq. (55) tranformed the LHS of eq. (54) into a fully algebraic expression. The integration and discretisation for the LHS of our σ-equation is finished. We now turn to the RHS of eq. (54).

5.2 Integration of spatial terms

On the right hand side (hereafter RHS) of eq. (54) we start with the spatial integration without source terms:

$$\int_{t^0}^{t^0+\Delta t} \int_s^n \frac{\partial}{\partial z}\left(K\frac{\partial \sigma}{\partial z}\right) dzdt = \int_{t^0}^{t^0+\Delta t} \left[\left(K\frac{\partial \sigma}{\partial z}\right)_n - \left(K\frac{\partial \sigma}{\partial z}\right)_s\right] dt \tag{56}$$

where index n indicates the value in the north face and index s indicates the value in the south face. We discretise these gradients using central differencing, so eq. (56) becomes

$$\int_{t^0}^{t^0+\Delta t} \left[\frac{K_n}{\delta z_n}(\sigma_N - \sigma_P) - \frac{K_s}{\delta z_s}(\sigma_P - \sigma_S)\right] dt.$$

In the time integral we need different strategies if we want a fully explicit, fully implicit or a Crank Nicholson scheme. For the fully implicit scheme, we take all values from the new time step so that the right hand side of eq. (54) is

$$RHS = \frac{\Delta t K_n}{\delta z_n}(\sigma_N^1 - \sigma_P^1) - \frac{\Delta t K_s}{\delta z_s}(\sigma_P^1 - \sigma_S^1)$$

We have established the discretised version of the LHS and the RHS of eq. (54) without source terms:

$$\Delta z(\sigma_P^1 - \sigma_P^0) = \frac{\Delta t K_n}{\delta z_n}(\sigma_N^1 - \sigma_P^1) - \frac{\Delta t K_s}{\delta z_s}(\sigma_P^1 - \sigma_S^1). \tag{57}$$

However, the source terms on the LHS of eq. (54) must be taken care of. We discretise the source term according to

$$\int_{t^0}^{t^0+\Delta t} \int_s^n S \quad dzdt = \overline{S}\Delta z \Delta t = S_U + S_P \sigma_P^1. \tag{58}$$

where negative terms from the source terms will go into S_P and subsequently on the LHS eliminating negative coefficients. The fully discretised general equation is

$$\boxed{\Delta z(\sigma_P^1 - \sigma_P^0) = \frac{\Delta t K_n}{\delta z_n}(\sigma_N^1 - \sigma_P^1) - \frac{\Delta t K_s}{\delta z_s}(\sigma_P^1 - \sigma_S^1) + S_U + S_P \sigma_P^1.} \tag{59}$$

We will return to this stage when handling the Neuman boundary condition.

To simplify, we collect the terms associated with σ_P^1 on the LHS and the rest on the RHS. We also divide by Δz on both sides and assume a uniform grid ($\Delta z = \delta z_s = \delta z_n$), which gives:

$$\left(1 + \frac{\Delta t K_n}{(\delta z)^2} + \frac{\Delta t K_s}{(\delta z)^2} - S_P\right)\sigma_P^1 = \frac{\Delta t K_n}{(\delta z)^2}\sigma_N^1 + \sigma_P^0 + \frac{\Delta t K_s}{(\delta z)^2}\sigma_S^1 + S_U. \tag{60}$$

In other words our finite volume formulation for the σ-equation can be expressed with

$$a_P\sigma_P^1 = a_N\sigma_N^1 + a_P^0\sigma_P^0 + a_S\sigma_S^1 + S_U \tag{61}$$

where

$$a_N = \frac{\Delta t K_n}{(\delta z)^2} \tag{62}$$

$$a_S = \frac{\Delta t K_s}{(\delta z)^2} \tag{63}$$

$$a_P^0 = 1 \tag{64}$$

and where

$$a_P = a_P^0 + a_N + a_S - S_P \tag{65}$$

5.3 Setting up an equation system

Equation (61) can form an equation system, if we separate the known (old time step) and the unknown (new time step) terms:

$$\boxed{-a_N\sigma_N^1 + a_P\sigma_P^1 - a_S\sigma_S^1 = a_P^0\sigma_P^0 + S_U} \tag{66}$$

where the property σ in the new time step resides on the LHS and the known value of property σ from the old time step resides on the RHS. Equation (66) can be written with matrix indexation:

$$\boxed{-A_i\sigma_{i-1}^{n+1} + B_i\sigma_i^{n+1} - C_i\sigma_{i+1}^{n+1} = F_i} \tag{67}$$

where

$$\begin{aligned} A_i &= a_N \\ C_i &= a_S \\ B_i &= a_P^0 + A_i + C_i - S_P \end{aligned} \tag{68}$$

and where

$$F_i = a_P^0\sigma_i^n + S_U. \tag{69}$$

As we move towards the programmatic implementation of the finite volume formulation we bring in an indexation used in matrices and programming in MATLAB. Imagine the values of e.g. σ_P being stored in a column vector, F (see equation system (70)). The north boundary node will be stored in the first element and have index $i = 1$ in the column vector, and the south boundary node will be stored in the last element and be indexed $i = I$. This is why σ_N will be indexed σ_{i-1} (one row above) etc. The equation system in eq. (67) can be set up as follows:

$$\begin{bmatrix} -A_2 & B_2 & -C_2 & \cdots & 0 \\ 0 & -A_3 & B_3 & -C_3 & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & -A_i & B_i & -C_i & 0 \\ \vdots & & \vdots & & \vdots \\ & -A_{I-2} & B_{I-2} & -C_{I-2} & 0 \\ 0 & \cdots & -A_{I-1} & B_{I-1} & -C_{I-1} \end{bmatrix} \begin{bmatrix} \sigma_1^{n+1} \\ \sigma_2^{n+1} \\ \vdots \\ \sigma_i^{n+1} \\ \vdots \\ \sigma_{I-1}^{n+1} \\ \sigma_I^{n+1} \end{bmatrix} = \begin{bmatrix} \sigma_2^n + S_U \\ \sigma_3^n + S_U \\ \vdots \\ \sigma_i^n + S_U \\ \vdots \\ \sigma_{I-2}^n + S_U \\ \sigma_{I-1}^n + S_U \end{bmatrix} \tag{70}$$

However, the equation system as it stands is unsolvable, since the number of unknown terms are larger than the number of equations. What we need is some boundary conditions enabling us to overlook two of the columns in equation system (70).

The boundary conditions can be treated in two ways; using a Dirichlet boundary condition to set a certain value of σ in node 1 and I (thereby converting an unknown term on the LHS into a known term on the RHS) or using a Neuman boundary condition to set a certain flux in the north face of node 2 and in the south face of node I-1, thereby eliminating and modifying our coefficients. Below I sort out how these boundary conditions are implemented in setting up the equation system to be solved in an iterative solver.

Dirichlet Boundary condition

Setting a Dirichlet boundary condition is essentially to set σ on the boundary (in the boundary node). For the north boundary (i=1), we set

$$\sigma_1^{n+1} = s_1^{n+1} \tag{71}$$

where s is a known value. Let's put this into equation (66) for the second node (i=2)

$$-a_N s_1^1 + a_P \sigma_2^1 - a_S \sigma_3^1 = a_P^0 \sigma_2^0 + S_U \tag{72}$$

yielding two unknown terms and one known term on the LHS. Let's move the known term to the RHS, since this term belongs to the forcing of the system.

$$a_P \sigma_2^1 - a_S \sigma_3^1 = a_P^0 \sigma_2^0 + S_U + a_N s_1^1 \tag{73}$$

Converting back to matrix notation in the second boundary node we yield

$$\boxed{B_2 \sigma_2^{n+1} - C_2 \sigma_3^{n+1} = F_2} \tag{74}$$

where

$$\boxed{F_2 = a_P^0\sigma_2^n + S_U + A_2 s_1^{n+1}.} \tag{75}$$

The same technique goes for the south boundary. At the south boundary we set:

$$\sigma_I^{n+1} = s_I^{n+1} \tag{76}$$

And for node $i = I - 1$, equation (66) becomes:

$$-a_N\sigma_{I-2}^1 + a_P\sigma_{I-1}^1 - a_S s_I^1 = a_P^0\sigma_{I-1}^0 + S_U \tag{77}$$

We put the known term on the RHS:

$$-a_N\sigma_{I-2}^1 + a_P\sigma_{I-1}^1 = a_P^0\sigma_{I-1}^0 + S_U + a_S s_I^1 \tag{78}$$

and for a Dirichlet boundary condition on the south boundary and converting back to matrix notation we yield

$$\boxed{-A_{I-1}\sigma_{I-2}^1 + B_{I-1}\sigma_{I-1}^1 = F_{I-1}} \tag{79}$$

where

$$\boxed{F_{I-1} = a_P^0\sigma_{I-1}^n + S_U + C_{I-1}s_I^{n+1}} \tag{80}$$

The equation system to be solved is

$$\begin{bmatrix} 0 & B_2 & -C_2 & \cdots & 0 \\ 0 & -A_3 & B_3 & -C_3 & \\ \vdots & & \vdots & & \vdots \\ & -A_i & B_i & -C_i & \\ \vdots & & \vdots & & \vdots \\ & -A_{I-2} & B_{I-2} & -C_{I-2} & 0 \\ 0 & \cdots & -A_{I-1} & B_{I-1} & 0 \end{bmatrix} \begin{bmatrix} \sigma_1^{n+1} \\ \sigma_2^{n+1} \\ \vdots \\ \sigma_i^{n+1} \\ \vdots \\ \sigma_{I-1}^{n+1} \\ \sigma_I^{n+1} \end{bmatrix} = \begin{bmatrix} \sigma_2^n + S_U + A_2 s_1^{n+1} \\ \sigma_3^n + S_U \\ \vdots \\ \sigma_i^n + S_U \\ \vdots \\ \sigma_{I-2}^n + S_U \\ \sigma_{I-1}^n + S_U + C_{I-1}s_I^{n+1} \end{bmatrix} \tag{81}$$

where B_2 and B_{I-1} are calculated with

$$B = a_P^0 + \frac{\Delta t K_n}{(\delta z)^2} + \frac{\Delta t K_s}{(\delta z)^2} - \frac{S_P}{\delta z} \tag{82}$$

Neuman boundary condition

In the Neuman boudary condition we choose a certain flux, q, at the boundary. Keep in mind that flux in a north face is calculated by

$$q_n = -\Gamma_n \frac{\sigma_N - \sigma_P}{\delta z} \tag{83}$$

and flux in a south face is calculated by

$$q_s = -\Gamma_s \frac{\sigma_P - \sigma_S}{\delta z} \tag{84}$$

Remember that the rate of change of a property in node P is actually equal to the spatial change of the fluxes over the boundaries to the cell (excluding effects from sources and sinks). In other words, if the flux into the cell from the north is larger than the flux out of the cell to the south, the change in σ is positive unless the source terms counteract that difference to the same degree. Excluding effects from sources, this can be described as.

$$\frac{\partial \sigma}{\partial t} = -\frac{\partial q}{\partial z} = -\frac{(q_n - q_s)}{\partial z}$$

Let's use eq. (59) and set the flux in the south boundary of cell $I-1$ to q_s.

$$\Delta z(\sigma^1_{I-1} - \sigma^0_{I-1}) = \frac{\Delta t K_n}{\delta z_n}(\sigma^1_{I-2} - \sigma^1_{I-1}) + \Delta t q_s + S_U + S_P \sigma^1_{I-1} \tag{85}$$

We divide by Δz on both sides and assume equidistant grid ($\Delta z = \delta z$):

$$\sigma^1_{I-1} - \sigma^0_{I-1} = \frac{\Delta t K_n}{(\delta z)^2}(\sigma^1_{I-2} - \sigma^1_{I-1}) + \frac{\Delta t}{\delta z} q_s + \frac{S_U}{\delta z} + \frac{S_P}{\delta z}\sigma^1_{I-1} \tag{86}$$

and collect terms so that the unknown terms are on the LHS, and the known terms are on the RHS:

$$-\frac{\Delta t K_n}{(\delta z)^2}\sigma^1_{I-2} + \left(1 + \frac{\Delta t K_n}{(\delta z)^2} + 0 - \frac{S_P}{\delta z}\right)\sigma^1_{I-1} - 0\sigma^1_I = \sigma^0_{I-1} + \frac{\Delta t}{\delta z} q_s + \frac{S_U}{\delta z}. \tag{87}$$

This can be rewritten with coefficients from eq. (68) into

$$-A_{I-1}\sigma^1_{I-2} + B_{I-1}\sigma^1_{I-1} - C_{I-1}\sigma^1_I = F_{I-1} \tag{88}$$

where

$$\boxed{C_{I-1} = 0} \tag{89}$$

and

$$\boxed{F_{I-1} = a^0_P \sigma^0_{I-1} + \frac{\Delta t}{\delta z} q_s + \frac{S_U}{\delta z}} \tag{90}$$

Analogously for a Neuman BC at the north wall we set the flux to q_n and eq. (59) for node i=2 is written

$$\Delta z(\sigma^1_2 - \sigma^0_2) = -\Delta t q_n - \frac{\Delta t K_s}{\delta z_s}(\sigma^1_2 - \sigma^1_3) + S_U + S_P \sigma^1_2 \tag{91}$$

We divide by Δz on both sides and assume equidistant grid $\Delta z = \delta z$:

$$\sigma_2^1 - \sigma_2^0 = -\frac{\Delta t}{\delta z} q_n - \frac{\Delta t K_s}{(\delta z)^2}(\sigma_2^1 - \sigma_3^1) + \frac{S_U}{\delta z} + \frac{S_P}{\delta z}\sigma_2^1 \tag{92}$$

and collect terms so that the unknown terms are on the LHS, and the known terms are on the RHS:

$$0\sigma_1^1 + \left(1 + 0 + \frac{\Delta t K_s}{(\delta z)^2} - \frac{S_P}{\delta z}\right)\sigma_2^1 - \frac{\Delta t K_s}{(\delta z)^2}\sigma_3^1 = \sigma_2^0 + \frac{S_U}{\delta z} - \frac{\Delta t}{\delta z} q_n. \tag{93}$$

This can be expressed with the coefficients from eq. (68);

$$-A_2\sigma_1^1 + B_2\sigma_2^1 - C_2\sigma_3^1 = F_2. \tag{94}$$

but where

$$\boxed{A_2 = 0} \tag{95}$$

and where

$$\boxed{F_2 = a_P^0 \sigma_2^0 - \frac{\Delta t}{\delta z} q_n + \frac{S_U}{\delta z}} \tag{96}$$

For clarity, in the Neuman case

$$\boxed{B_2 = a_P^0 + A_2 + C_2 - \frac{S_P}{\delta z}} \tag{97}$$

The equation system now looks like

$$\begin{bmatrix} A_2 & B_2 & -C_2 & \cdots & 0 \\ 0 & -A_3 & B_3 & -C_3 & \\ \vdots & & \vdots & & \vdots \\ & -A_i & B_i & -C_i & \\ \vdots & & \vdots & & \vdots \\ & -A_{I-2} & B_{I-2} & -C_{I-2} & 0 \\ 0 & \cdots & -A_{I-1} & B_{I-1} & -C_{I-1} \end{bmatrix} \begin{bmatrix} \sigma_1^{n+1} \\ \sigma_2^{n+1} \\ \vdots \\ \sigma_i^{n+1} \\ \vdots \\ \sigma_{I-1}^{n+1} \\ \sigma_I^{n+1} \end{bmatrix} = \begin{bmatrix} \sigma_2^n + \frac{S_U}{\delta z} - \frac{\Delta t}{\delta t} q_n \\ \sigma_3^n + \frac{S_U}{\delta z} \\ \vdots \\ \sigma_i^n + \frac{S_U}{\delta z} \\ \vdots \\ \sigma_{I-2}^n + \frac{S_U}{\delta z} \\ \sigma_{I-1}^n + \frac{S_U}{\delta z} + \frac{\Delta t}{\delta z} q_s \end{bmatrix} \tag{98}$$

where

$$\begin{aligned} A_2 &= 0 \\ B_2 &= a_P^0 + A_2 + C_2 - \frac{S_P}{\delta z} \end{aligned} \tag{99}$$

5.4 Explicit Neuman

To set explicit Neuman on the north boundary we use the expression in eq. (83) and solve for σ_N.

$$\sigma_N = -\frac{q_n \delta z}{\Gamma_n} + \sigma_P \tag{100}$$

For the south boundary the flux is described by

$$\sigma_S = \frac{q_s \delta z}{\Gamma_s} + \sigma_P. \tag{101}$$

5.5 Summary of setting boundary conditions

To summarize this extensive discretisation, let us recap the equation system and its manipulation by setting boundary conditions once more. The unsolvable equation system is

$$\begin{bmatrix} -A_2 & B_2 & -C_2 & \cdots & 0 \\ 0 & -A_3 & B_3 & -C_3 & \\ \vdots & & \vdots & & \vdots \\ & -A_i & B_i & -C_i & \\ \vdots & & \vdots & & \vdots \\ & -A_{I-2} & B_{I-2} & -C_{I-2} & 0 \\ 0 & \cdots & -A_{I-1} & B_{I-1} & -C_{I-1} \end{bmatrix} \begin{bmatrix} \sigma_1^{n+1} \\ \sigma_2^{n+1} \\ \vdots \\ \sigma_i^{n+1} \\ \vdots \\ \sigma_{I-1}^{n+1} \\ \sigma_I^{n+1} \end{bmatrix} = \begin{bmatrix} \sigma_2^n + S_U \\ \sigma_3^n + S_U \\ \vdots \\ \sigma_i^n + S_U \\ \vdots \\ \sigma_{I-2}^n + S_U \\ \sigma_{I-1}^n + S_U \end{bmatrix} \tag{102}$$

where

$$\begin{aligned} A_i &= \frac{\Delta t K_n}{(\delta z)^2} \\ C_i &= \frac{\Delta t K_s}{(\delta z)^2} \\ a_P^0 &= 1 \\ B_i &= a_P^0 + A_i + C_i - S_P \end{aligned} \tag{103}$$

To implement boundary conditions we do the following:

Dirichlet

On the north wall we set $\sigma_1^{n+1} = s_1^{n+1}$, and move this now known term to the RHS along with A_2. On the south wall we set $\sigma_I^{n+1} = s_I^{n+1}$, and move this now known term to the

RHS along with C_{I-1}. The new equation system to be solved is.

$$\begin{bmatrix} 0 & B_2 & -C_2 & \cdots & 0 \\ 0 & -A_3 & B_3 & -C_3 & \\ \vdots & & \vdots & & \vdots \\ & -A_i & B_i & -C_i & \\ \vdots & & \vdots & & \vdots \\ & -A_{I-2} & B_{I-2} & -C_{I-2} & 0 \\ 0 & \cdots & -A_{I-1} & B_{I-1} & 0 \end{bmatrix} \begin{bmatrix} \sigma_1^{n+1} \\ \sigma_2^{n+1} \\ \vdots \\ \sigma_i^{n+1} \\ \vdots \\ \sigma_{I-1}^{n+1} \\ \sigma_I^{n+1} \end{bmatrix} = \begin{bmatrix} \sigma_2^n + S_U + A_2 s_1^{n+1} \\ \sigma_2^n + S_U \\ \vdots \\ \sigma_i^n + S_U \\ \vdots \\ \sigma_{I-2}^n + S_U \\ \sigma_{I-1}^n + S_U + C_{I-1} s_I^{n+1} \end{bmatrix} \tag{104}$$

Neuman

On the north boundary we eliminate the occurence of A_2 by setting the flux from the north face. On the south boundary we eliminate the occurence of C_{I-1} by setting the flux on the south face, yielding the equation system

$$\begin{bmatrix} A_2 & B_2 & -C_2 & \cdots & 0 \\ 0 & -A_3 & B_3 & -C_3 & \\ \vdots & & \vdots & & \vdots \\ & -A_i & B_i & -C_i & \\ \vdots & & \vdots & & \vdots \\ & -A_{I-2} & B_{I-2} & -C_{I-2} & 0 \\ 0 & \cdots & -A_{I-1} & B_{I-1} & C_{I-1} \end{bmatrix} \begin{bmatrix} \sigma_1^{n+1} \\ \sigma_2^{n+1} \\ \vdots \\ \sigma_i^{n+1} \\ \vdots \\ \sigma_{I-1}^{n+1} \\ \sigma_I^{n+1} \end{bmatrix} = \begin{bmatrix} \sigma_2^n + S_U - \frac{\Delta t}{\delta t} q_n \\ \sigma_3^n + S_U \\ \vdots \\ \sigma_i^n + S_U \\ \vdots \\ \sigma_{I-2}^n + S_U \\ \sigma_{I-1}^n + S_U + C_{I-1} s_I^{n+1} \end{bmatrix} \tag{105}$$

where

$$A_2 = 0, \tag{106}$$

$$C_{I-1} = 0 \tag{107}$$

In both cases (Dirichlet or Neuman) B_2 and B_{I-1} are the sum of the neighbouring coefficients, and A_2 and C_{I-1} can both be set to 0 (as we can see by observing the topmost left and bottommost right corner of both equation systems (104) and (105)). However, A_2 and C_{I-1} have to be set to 0 after calculating B_2 or B_{I-1} and F_2 or F_{I-1} respectively when using the Dirichlet boundary condition otherwise they will be missing in the central coefficient (eq. (82)) and in the forcing on the RHS.

6 The graphical user interface

In this section, I first present the MATLAB application GUIDE in which I built SICT. Then, I explain what functions are featured in SICT and how to operate them.

6.1 GUIDE

A user interface is anything a user and a computer can use to mutually exchange information. It can be the keypad, screen, speaker and mouse of a computer. A Graphical User Interface (hereafter GUI) is a user interface with graphical objects such as windows, buttons, menus, text boxes and other components.

The GUI I created, the SICT, was built using the Graphical User Interface Development Environment, the GUIDE application in MATLAB 2016b, shown in figure (3). The GUIDE application allows a user to set up a GUI by dragging and dropping GUI components onto a figure. This generates a figure file. There are 9 different components available to drag and drop into the figure. These components are actually 9 different styles (checkbox, edit, list box, pop-up menu, push button, radio button, slider, text and toggle button) of the same type of UI objects *uicontrol* (italic to show type of object). In addition to these *uicontrol* objects, there are other UI objects such as *uipanel* and *uibuttongroup* which the user can select and add to the GUI. The option to add axis object is also available in the drag and drop function of the GUIDE application. All aforementioned objects are children of MATLAB figure objects (Hanselman and Littlefield, 2005).

Running the figure file generates an m-file containing the minimum code to generate the UI figure with all components added. As the user adds components to the UI, additional callback and creation functions can be coupled with the components making them interactive.

Each object has a number of properties, some of which, such as the size and position, can be changed easily in the GUIDE application. An important property for the appearance of the GUI is the "Units" property of the objects. The units can be either of absolute or normalized character. The absolute units such as "centimeters", "pixels", "inches" and "points" make sure objects stay at a locked distance relative to the figure bottom and left edge regardless of resizing. Furthermore, the MathWorks webpage (MathWorks, 2017) recommends using the "points" or "characters" units for a cross platform GUI, as it will conserves the layout of the GUI across different platforms. Using the "normalized" units the objects will resize with the figure and have them stay at the same relative distance from each other. I tested setting the units of my objects to either points or normalized and found, despite the recommendations, that the normalized units worked best even across platforms.

One of the most important and by me frequently utilized property is the "Tag" property which is the handle used to attain information about the object. In most occations, the information inputted by the user into the settings modules are collected through the "Tag" property in one external function *getinputs.m* rather than creating separate callbacks for each uicontrol. The only uicontrols except the push buttons which have callbacks are the three pop-up menues, where the units change if the velocity boundary conditions are changed, or the text "Top layer thicknes [m]" changes into "Calculated N2" ore vice versa, if the stratification is changed by the user.

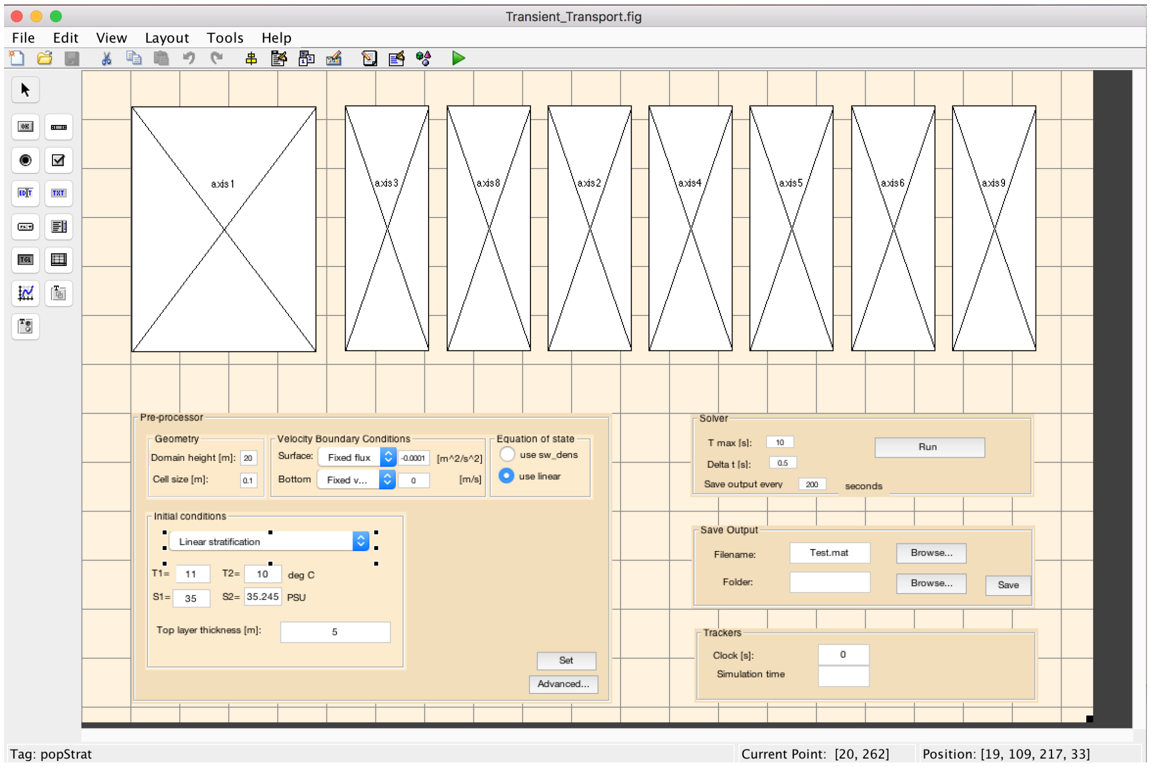
6.2 SICT

SICT is shown in figure 4 and figure 5. SICT includes plotting windows to display calculated variables, pop-up menus, radio buttons and text boxes to change settings of the simulation and buttons to run and save simulations. It also contains some text boxes displaying simulation parameters such as simulated time and time spent simulating which is practical for the user to keep track of the marching of the simulation.

The 6 numbered solid boxes in figures 4 and 5 show the stages in which the user needs to operate the model. The user should use these 6 stages in numerical order to operate SICT. In stages 1, 3 and 5 there are several settings modules the user can use in any order within each stage. Dashed boxes and arrows show the areas affected by each settings module. The following list is a short guide to the stages 1-6 in fig 4 and 5 of operating the SICT. A more extensive instruction is presented in Appendix C.

1. Set up domain, boundary conditions and initial conditions.

    - **Set up geometry of your domain** (red box, fig. 4) by changing the inputted values in boxes "Domain height [m]" and "Grid size [m]". When pressing the "Set" button in stage 2, this will affect the scale of the z-axis in your domain (dashed red ellipse, fig 4).
    - **Set boundary condition for mean velocity, U** (blue box, fig. 4) by either choosing "Fixed flux" or "Fixed value" and then entering a friction velocity or a fixed velocity in the text box. This will effect the value of the velocity (dashed blue circles, fig 4), as well as update the units of the boundary condition (to the right in the solid blue box).
    - Make settings for stratification (orange boxes, fig. 4). These will have effect on the form of the stratification (dashed orange ellipse, fig 4).
        - **Choose whether a linear equation of state or the MATLAB-function sw_dens is used to calculate density** (top right orange box, fig. 4) by choosing one of two radio buttons.
        - **Choose stratification form** (bottom orange box, fig. 4) by choosing either "Linear temperature and salinity" or "2-layer stratification" in the pop-up menue.



Pre-processor

Geometry
Domain height [m]: 20
Cell size [m]: 0.1

Velocity Boundary Conditions
Surface: Fixed flux -0.0001 [m^2/s^2]
Bottom Fixed v... 0 [m/s]

Equation of state
use sw_dens
[✓ use linear]

Initial conditions
Linear stratification
T1= 11 T2= 10 deg C
S1= 35 S2= 35.245 PSU
Top layer thickness [m]: 5

Solver
T max [s]: 10
Delta t [s]: 0.5
Save output every 200 seconds

Save Output
Filename: Test.mat
Folder:

Trackers
Clock [s]: 0
Simulation time

Tag: popStrat
Current Point: [20, 262]
Position: [19, 109, 217, 33]

Figure 3: Screen capture of the GUIDE application in MATLAB 2016b. Left panel shows available components which the user can drag into the GUI figure template. The beige area shows the layout of the SICT figure.

  - **Make changes to the absolute stratification** (bottom orange box, fig. 4) by changing the values of T1, T2, S1 and S2. **If the 2-layer stratification is chosen, change the top layer thickness** by editing the "Top layer thickness [m]:" box.
- **If desirable, make changes to constants** of the equation of state, change the roughness lengths at the surface or the bottom, or change the value of gravitational acceleration (green box, fig. 4) by clicking the "Advanced..." button to open an external window. Then click the "Save & Close" button.

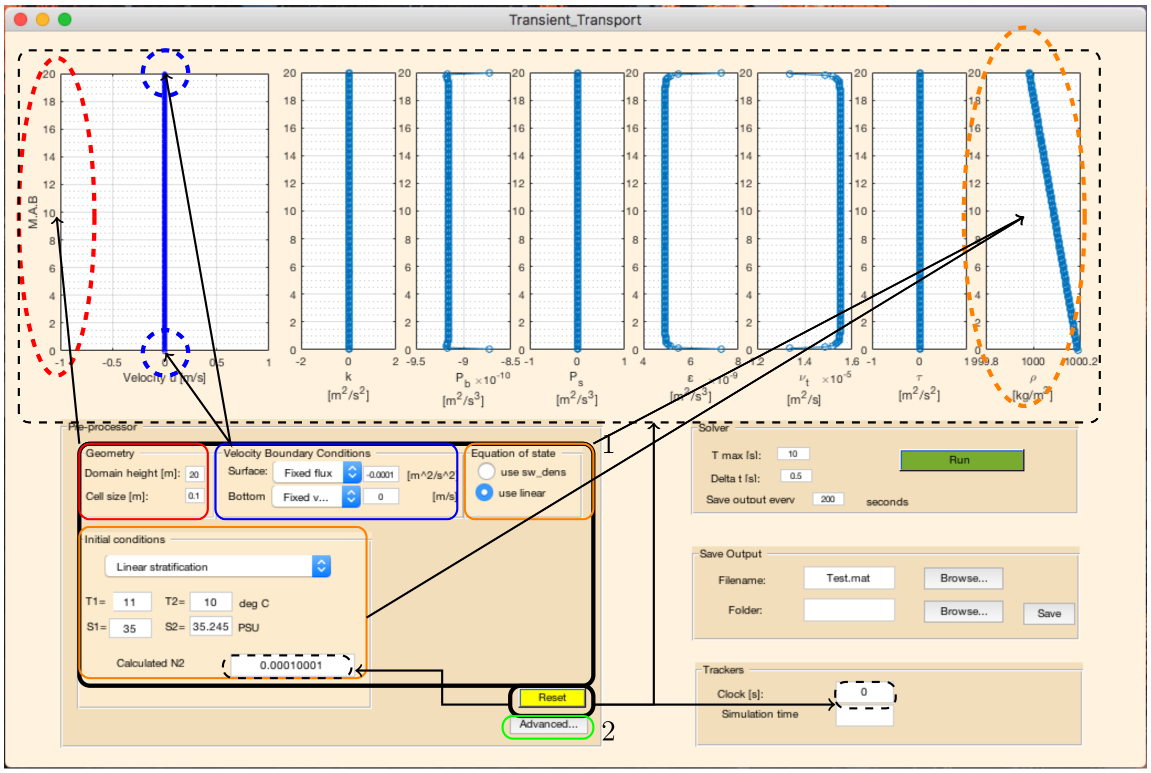
2. **Apply your settings** by clicking the "Set"-button. This will plot initial velocity and density, and other terms in the plotting windows (large dashed black box in fig 4). SICT also calculates and displays the buoyancy frequency squared if the linear stratification is chosen (dashed black box inside bottom orange box). The clock is also reset by pressing the "Set" button (dashed black box to the left).

3. Make marching settings
   - **Set simulation time** (cyan box, fig. 5) by editing the "T max [s]:" box.
   - **Set time step** (magenta box, fig. 5) by editing the "Delta t [s]:" box.
   - **Set save interval** (violet box, fig. 5) by editing the "Save output every" box. This setting applies from the so far simulated time to the end of the forthcomming simulation. That is, the save interval applies for the additional T max time.

4. Run your simulation by clicking the "Run" button. SICT will update the plotting windows when the inputted T max time has been simulated. This time will be added to the clock. It is possible to continue the simulation by simply clicking the "Run" button again. SICT will continue the simulation with the additional inputted T max time. This simulated time will also be added to the clock (bottom right Tracker panel). In between clicking "Run", it is possible to change both T max, delta t and save interval, enabling the user to adjust the temporal resolution in certain stages of the simulation.

5. Set up output file and folder
   - **Choose filename** (teal box, fig. 5) by either editing the "Filename:" box or pressing the "Browse..." button to choose a pre-existing filename
   - **Choose output folder** (lime green box, fig. 5) by either editing the "Folder:" box or pressing the "Browse..." button to choose a folder.

6. Save output by pressing the "Save" button. The outputted variables include time series for temperature, T, salinity, S and velocity, u. The output also contains the boundary conditions, domain grid, and setting for the last used delta t.



Geometry
Domain height [m]: 20
Cell size [m]: 0.1

Velocity Boundary Conditions
Surface: Fixed flux -0.0001 [m^2/s^2]
Bottom Fixed v... 0 [m/s]

Equation of state
( ) use sw_dens
(●) use linear

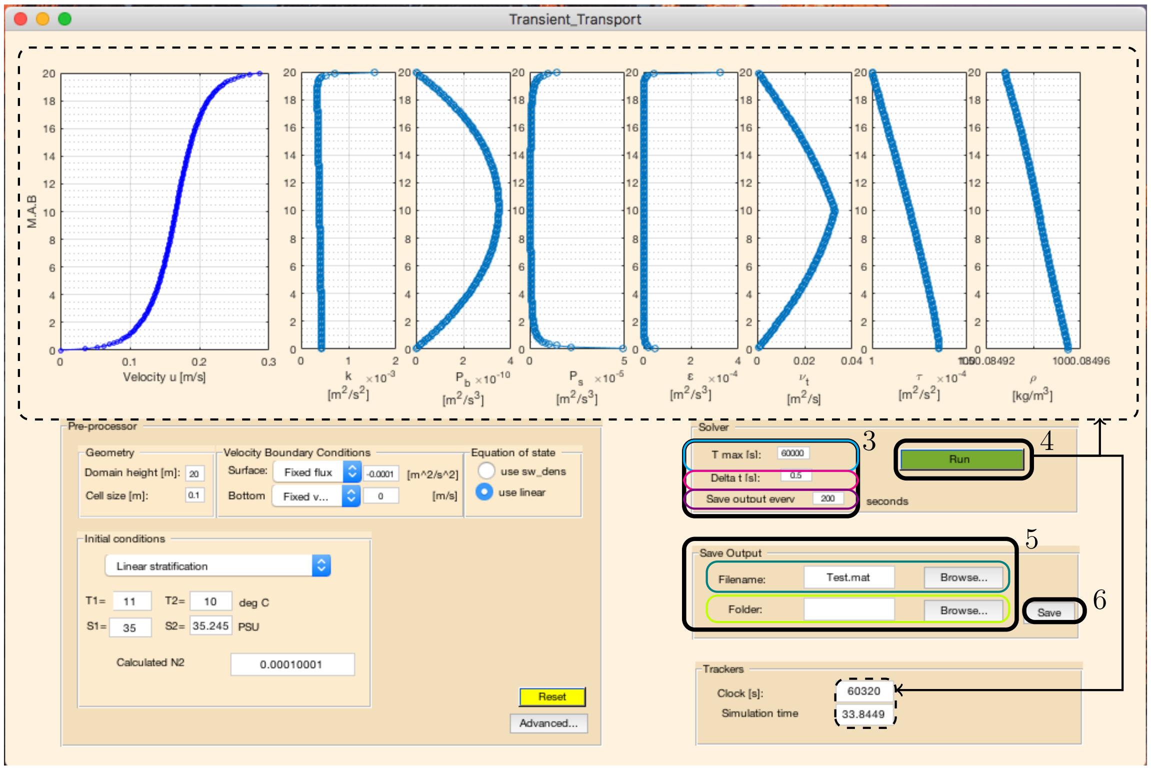
Initial conditions
Linear stratification
T1= 11 T2= 10 deg C
S1= 35 S2= 35.245 PSU
Calculated N2 0.00010001

T max [s]: 10
Delta t [s]: 0.5
Save output every 200 seconds

Save Output
Filename: Test.mat
Folder:

Trackers
Clock [s]: 0
Simulation time

Figure 4: SICT in a stage where geometry of the domain and boundary conditions have been set. Solid black boxes show stages which the user should use in numerical order to manage SICT. Solid coloured boxes show settings modules which the user can modify to produce the requested scenario. Dashed boxes show affected areas changing in accordance with the boundary conditions and the geometry of the domain.



Pre-processor

Geometry
Domain height [m]: 20
Cell size [m]: 0.1

Velocity Boundary Conditions
Surface: Fixed flux -0.0001 [m^2/s^2]
Bottom Fixed v... 0 [m/s]

Equation of state
( ) use sw_dens
(●) use linear

Initial conditions
Linear stratification
T1= 11 T2= 10 deg C
S1= 35 S2= 35.245 PSU
Calculated N2 0.00010001

Solver
T max [s]: 60000
Delta t [s]: 0.5
Save output every 200 seconds

Save Output
Filename: Test.mat
Folder:

Trackers
Clock [s]: 60320
Simulation time 33.8449

Figure 5: SICT in a stage where the model has been run for a certain amount of time. Solid coloured boxes show stages which the user should go through in chronological order to manage SICT. Dashed boxes show affected areas changing in accordance with pressing the Run button. Box 3 shows the marching settings module handling simulation time, size of the time step and the saving interval. Box 4 shows the button for running the simulation. Box 5 shows the settings module for outputting model data. Box 6 shows the button for saving output to the selected file and folder.

7 Validation

7.1 Validation against the logarithmic velocity profile

SICT was run with a domain height of 10 meters, a grid size of 0.1 meters and a fixed velocity of 1 m/s in the surface. Stratification was removed by setting T1=T2=10 °C and S1=S2=35 psu. The simulation was run for 1 hour (3600 seconds) to ensure the system had reached steady state. The logarithmic profile deducted through dimensional analysis by Axell and Liungman (2001) in eq. (52) is plotted along with the outputted velocity profile in figure 6. This is a very close fit which shows the ability of SICT to reproduce the theoretical logarithmic law when forced to a steady state. Furthermore, at the end of the simulation, the eddy viscosity (not shown here) showed a linear profile consistent with the logarithmic law.

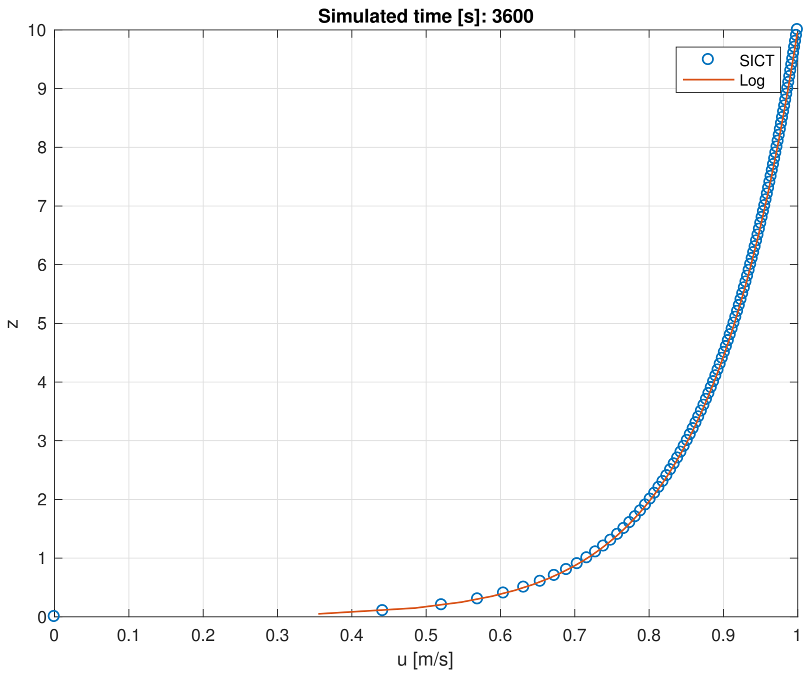


Figure 6: Model output velocity profile (blue) and the logarithmic velocity profile of eq. (52) (red).

7.2 Validation against laboratory experiments

In this section I compare my SICT output with the experimental results of KP and KPA as presented by Price (see section 3). Two tests were conducted based on the mixed layer depth of the SICT output. A constant friction velocity u_* at the surface was set to a fluid at rest in both cases.

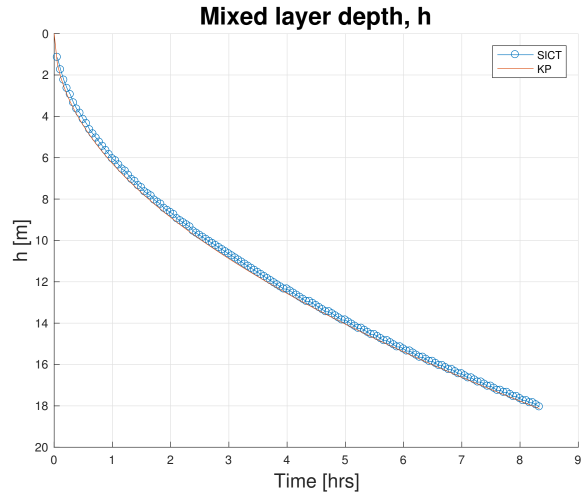
In the first test the fluid was stratified linearly. This test is similar to the one conducted by Saetra et al. (2008). In the second test there was a two-layer stratification. The criteria for the MLD in the model,h$_{SICT}$,was set in both tests to the depth where the kinetic energy, k, deceeds 10^{-6} m^{2}s^{-2}.

Kato and Philips (Linear stratification)

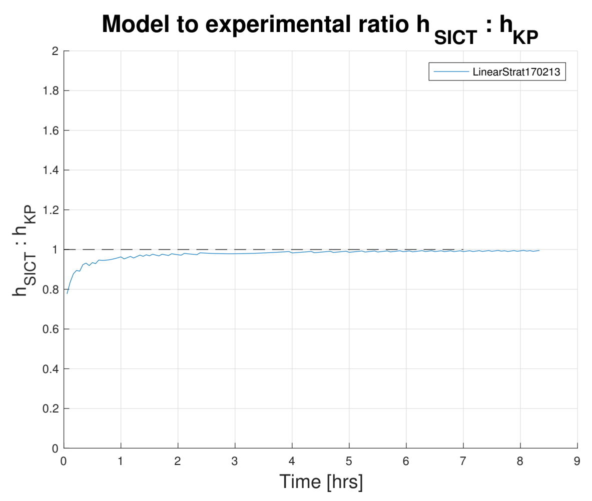
The SICT was set up with a linear stratification with a mean buoyancy frequency squared of N^2 =10^{-4} s^{-2}, (T1=11, T2=10, S1=35, S2=35.245). A linear equation of state was chosen to calculate density. The boundary condition for the velocity was set to a fixed flux, u_*, at the surface of -10^{-4} m^2/s^2 and to the fixed value 0 at the bottom. The domain size was 20 meters with a grid size of 0.1 m. Tmax was set to 8.3 hours ($3{\cdot}10^4$ seconds). Every 200 seconds of simulated time was saved in an output file. The simulation took 18 seconds to complete.

The outputted results from SICT is plotted and compared with the experimental results of KP in figure 7. In figure 7a h_{SICT} and h_{KP} are plotted together showing a very close fit with only small deviations which go smaller still with time. There is a slight underestimation (<1 m) of the mixed layer depth in SICT. The close fit is in accordance with the same type of validation by Saetra et al. (2008). In figure 7b the ratio between h$_{SICT}$ and h$_{KP}$ is plotted over time and this clearly shows that the ratio turns smaller and compliance between model and experimental result grows better over time. However, whilst the current validation slightly underestimates the mixed layer depth, Saetra et al. (2008) overestimates the mixed layer depth compared to the experimental results.

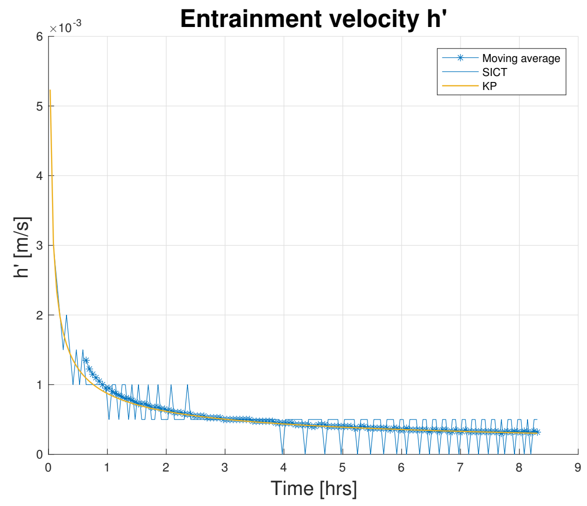
A time derivative of h_{KP} was calculated from the SICT output data and plotted together with the time derivative of eq. (28) in figure 7c. Due to the insufficient resolution of the grid, the derivative of h$_{SICT}$ is fluctuating. Also, as entrainment slows down at times the time derivative is 0. A moving average of the SICT time derivative was therefore calculated with a span of 67 minutes (20 output points) to get a sense of the overall entrainment rate. The moving average is also plotted in figure 7c. The ratio of the SICT time derivatives (both regular and moving average) to time derivative of h$_{KP}$ are shown in fig 7d.



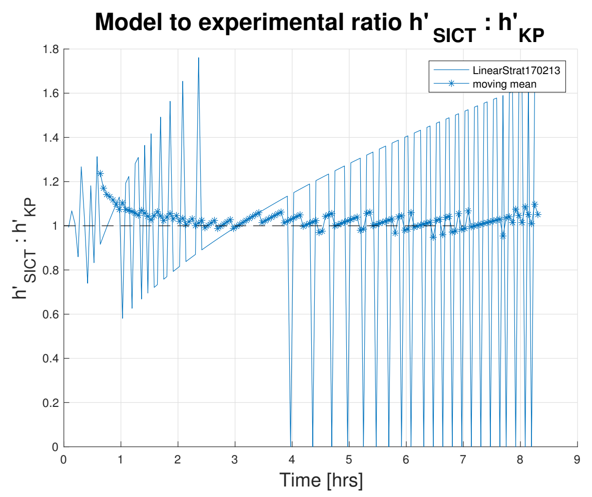
(a) Mixed layer depth h_{SICT} from SICT (blue line) and h_{KP} (red line) plotted over time.



(b) The ratio of modelled to experimental mixed layer depths h_{SICT}/h_{KP} plotted over time.



(c) Time derivative h'_{SICT} (solid blue line), moving average of h'_{SICT} (blue line with stars) and h'_{KP} (solid red line) plotted over time



(d) Ratio h'_{SICT} to h'_{KP} (solid blue line) and ratio moving average of h'_{SICT} to h'_{KP}

Figure 7: Comparison of the SICT mixed layer depth in a linear stratification with the experimental results of Kato and Phillips (1969).

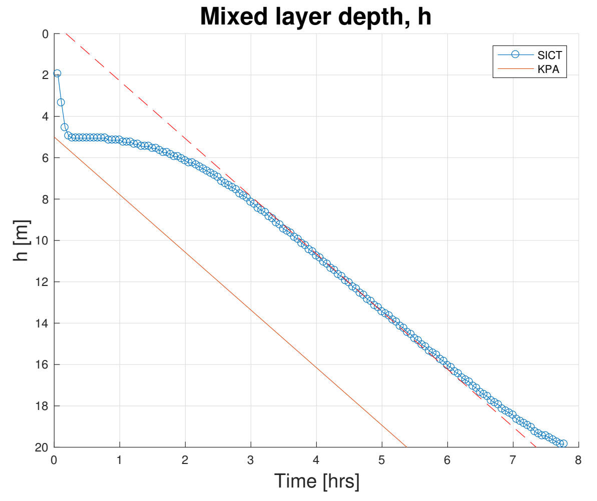
Kantha, Philips, Azhad (2-layer stratification)

The SICT was set up with a 2-layer stratification with a density difference of 0.2 kg/m^3 (T1=11, T2=10, S1=35, S2=35.245). A linear equation of state was chosen to calculate density. The boundary condition was set to a fixed flux at the surface of u_*^2= -10^{-4} m^2/s^2 and to the fixed value 0 at the bottom. The domain size was 20 meters with a grid size of 0.1 m. Tmax was set to 8.3 hours (3·10^4 seconds). The top layer thickness was 5 meters. The simulation took 17 seconds.

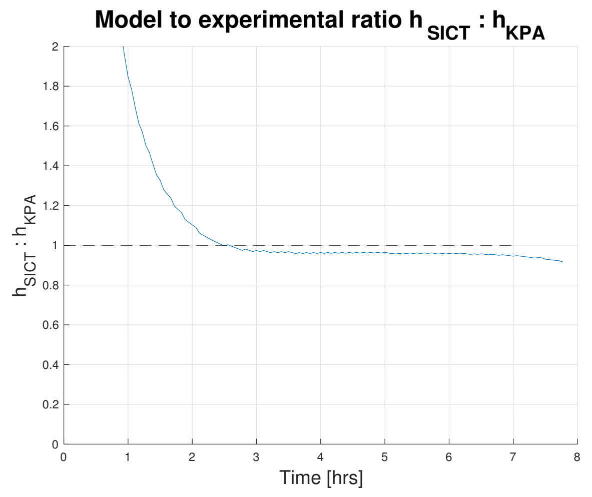
h_{SICT} is plotted together with h_{KPA} (eq. (29) multiplied with the time step 200 seconds) in figure 8a. A dashed line with the same slope as the function for h_{KPA} is plotted next to h_{SICT} for easier visual comparison. There is an evident spin up of the surface layer of SICT of about 2.5 hours. However, the rate at which SICT entrains fluid does go toward the rate anticipated by Price (1979). The end tail of h_{SICT} departs slightly from the linear form. This is due to the effect of the bottom. As h_{SICT} approaches 20 meters turbulent entrainment is dampened by the vicinity of the wall. An additional test (not shown here) where the domain height was increased showed the same diverging effect but delayed in time.

The ratio between h_{SICT} and h_{KPA} is shown in figure 8b. Following the spin up, the ratio shows a close fit. However the ratio grows slightly larger over time from 4.5 hours. This is due to the effect of the bottom.

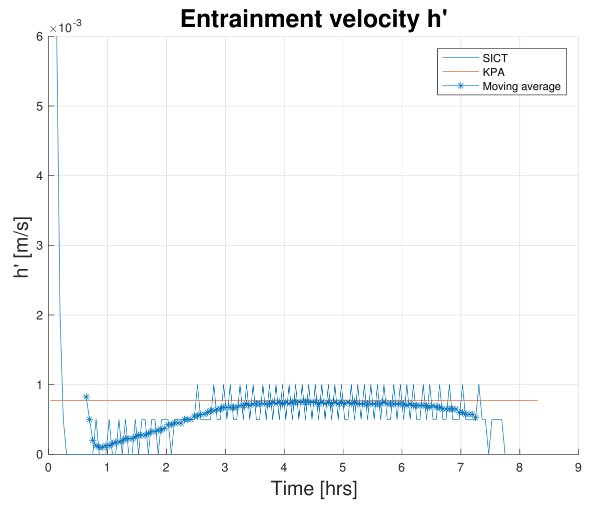
The time derivative, h'_{SICT} of the SICT mixed layer depth was calculated by dividing the difference in h_{SICT} between time steps by the size of the time step, 200 seconds. h'_{SICT} is compared to eq. (29) in figure 8c. The derivative, h$'_{SICT}$, was shaky for the same reason as in the previous KP case, so a moving mean was applied to this data series as well, also shown in figure 8c. The ratio between the derivatives (model to experimental and model moving average to experimental) is shown in fig 8d.



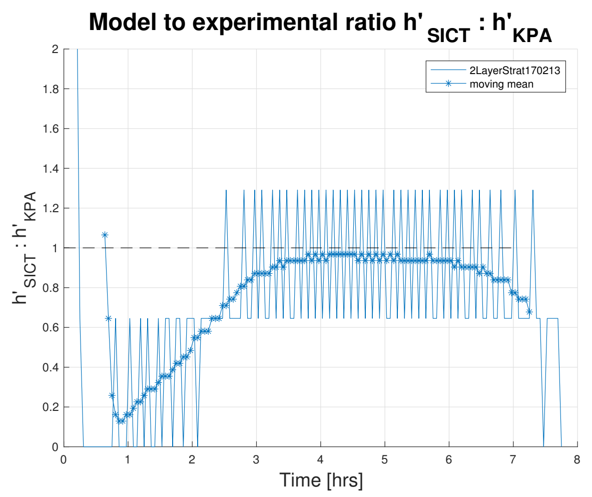
(a) Mixed layer depth h_m from SICT (blue line) and h_{KPA} (red line) plotted over time.



(b) The ratio of modelled to experimental mixed layer depths h_m/h_{KPA} plotted over time.



(c) Time derivative h'_m (solid blue line), moving average of h'_m (blue line with stars) and h'_{KPA} (solid red line) plotted over time



(d) Ratio h'_{SICT} to h'_{KPA} (solid blue line) and ratio sliding average of h'_{SICT} to h'_{KPA} (blue line with stars)

Figure 8: Comparison of the SICT output mixed layer depth with the results of Kantha et al. (1977).

8 Summary

The close correspondance of the k model to both the KP and KPA cases are partially due to the fact that the k model has been calibrated against these experiments. However, the over all compliance with both theory and reality were thoroughly tested by Axell and Liungman (2001) and showed good results even in comparison with oceanic datasets. SICT has undergone validation against the KP and KPA experiments, and shown accordance in this as well as in the steady state case of the logarithmic profile. I conclude that the k model is practical, and relatively true to theory and reality, especially with regards to the modest computational resources it demands. SICT showed promising results and offers an easy to use interface for practically anyone interested in turbulence. Further validation of SICT against oceanic datasets such as conducted by Axell and Liungman (2001) and Axell and Liungman (2001) would be interesting, however in this case time was a restricting factor. SICT provides users with an interface for the k model as proposed by Axell and Liungman (2001). The simplifications of the model include no buoyancy flux and no heat flux at the boundaries. SICT models the turbulent boundary layer near a wall. With SICT a user can easily

- Determine the size of the model domain
- Determine the resolution of the model domain
- Modify the boundary conditions for velocity U
- Modify initial temperature gradient
- Modify initial salinity gradient
- Initiate the simulation with either a one layer or two layer stratification
- Either use a linear equation of state or the MATLAB function sw_dens.m to calculate density
- Save output to a MATLAB workspace

Moreover, the user can by modifying the Advanced settings

- Change the thermal expansion coefficient of water in the linear equation of state
- Change the saline contraction coefficient of water in the linear equation of state
- Change the reference salinity and reference temperature in the linear equation of state
- Change the surface and bottom roughness lengths used in the model
- Change the gravitational acceleration of the model

The SICT code is available at https://github.com/josefinaalgotsson/SICT and is licensed under the GNU General Public License v.3.

Acknowledgements

I'd like to thank my supervisor Torsten Linders, PhD Physical Oceanography, University of Gothenburg, Department of Marine Sciences for being the ever challenging sounding board I've appreciated the past three years. I'd also like to thank my husband Jonas for his supporting spirit, neck massages and lunch boxes. They've been golden.

References

Axell, L. and Liungman, O. (2001). A one-equation turbulence model for geophysical applications: Comparison with data and the k-e model. *Environmental Fluid Mechanics*, 1:71–106.

Boussinesq, J. (1877). Essai sur la théorie des eaux courantes. *Mémoires présentés par divers Savants*, 23:1–680.

Burchard, H. and Baumert, H. (1995). On the performance of a mixed-layer model based on the k-$\mathcal{E}$ turbulence closure. *J. Geophys. Res*, 100:8523–8540.

Cushman-Rosin, B. and Beckers, J. (2011). *Introduction to Geophysical Fluid Dynamics Physical and Numerical Aspects*, volume 101 of *International geopgysics series*. Academic Press, 225 Wyman street, Waltham, MA 02451, USA, second edition.

Deardorff, J., Willis, G., and Lilly, D. (1969). Laboratory investigation of non-steady penetrative convection. *J. Fluid Mech.*, 35:7–31.

Ellison, T. and Turner, J. (1959). Turbulent entrainment in stratified flows. *J. Fluid. Mech*, (6):423–448.

Hanselman, D. and Littlefield, B. (2005). *Mastering Matlab 7*. Pearson Education International, Upper Saddle River , New Jersey 07458.

Kantha, L. H., Phillips, O. M., and Azad, R. S. (1977). On turbulent entrainment at a stable density interface. *Journal of Fluid Mechanics*, 79:753–768.

Kato, H. and Phillips, O. M. (1969). On the penetration of a turbulent layer into stratified fluid. *Journal of Fluid Mechanics,*, 37(4):643–665.

Kundu, P. and Cohen, I. (2002). *Fluid Mechanics*. Academic Press, California, USA, second edition.

Launder, B. E. and Spalding, D. (1974). The numerical computation of turbulent flows. *Comput. Methods Appl. Mech. Eng.*, 3:269–289.

MathWorks (12/2/2017). Lay out a UI programmatically. https://se.mathworks.com /help/matlab/creating_guis/layout-a-gui-programmatically.html.

Mellor, G. (1982). Development of a turbulence closure model for geophysical fluid problems. *Reviews of geophysics and space physics*, 20(4):851–875.

Price, J. (1979). On the scaling of stress-driven entrainments. *Journal of Fluid Mechanics*, (90):509–529.

Price, J., Mooers, C., and van Leer, J. (1978). Observation and simulation of storm driven mixed-layer deepening. *J. Phys. Ocean*, 8:582–599.

Reynolds, O. (1883). An experimental investigation of the circumstances which determine wheter the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Proceedings of the Royal Society of London*, (35):84–99.

Rouse, H. and Dodu, J. (1955). Turbulent diffusion across a density discontinuity. *Houille Blanche*, (10):522–532.

Saetra, O., Linders, T., and Debernard, J. (2008). Can polar lows lead to a warming of the ocean surface? *Tellus*, (60A):141–153.

Thorpe, S. (2005). *The Turbulent Ocean.* Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 1 edition.

Thorpe, S. (2007). *An introduction to Ocean Turbulence.* Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 1 edition.

Versteeg, H. and Malalasekera, W. (2007). *An introduction to Computational Fluid Dynamics.* Pearson Prentice Hall, Edinburgh Gate, Harlow, Essex CM20 2JE, second edition.

A Complete discretisation formulation for the transport equations

A.1 U-equation

The U-equation in eq. (33), have no source terms, and consequently the discretisation formulation is

$$a_P u_P^1 = a_N u_N^1 + a_P^0 u_P^0 + a_S u_S^1 + S_U \tag{108}$$

where

$$\begin{aligned} S_U &= 0 \\ S_P &= 0 \end{aligned} \tag{109}$$

$$a_N = \frac{\Delta t \nu_{totn}}{(\delta z)^2} \tag{110}$$

$$a_S = \frac{\Delta t \nu_{tots}}{(\delta z)^2} \tag{111}$$

$$a_P^0 = 1 \tag{112}$$

$$a_P = a_P^0 + a_N + a_S - S_P \tag{113}$$

A.2 T-equation and S-equation

Modelling salinity or temperature is identical to the modelling of U except for using the turbulent diffusivity ν'_{tot} instead of turbulent viscosity ν_{tot}. So the transport equations for temperature and salinity eq. (35) and eq. (34) can both be described with the general discretisation formulation

$$a_P \sigma_P^1 = a_N \sigma_N^1 + a_P^0 \sigma_P^0 + a_S \sigma_S^1 + S_U \tag{114}$$

where σ is either temperature or salinity and

$$\begin{aligned} S_U &= 0 \\ S_P &= 0 \end{aligned} \tag{115}$$

$$a_N = \frac{\Delta t \nu'_{totn}}{(\delta z)^2} \tag{116}$$

$$a_S = \frac{\Delta t \nu'_{tots}}{(\delta z)^2} \tag{117}$$

$$a_P^0 = 1 \tag{118}$$

$$a_P = a_P^0 + a_N + a_S - S_P. \tag{119}$$

A.3 k-equation

The discretisation of the k-equation is a bit more complex. To start with the discretisation formulation

$$a_P k_P^1 = a_N k_N^1 + a_P^0 k_P^0 + a_S k_S^1 + S_U \tag{120}$$

looks the same, and the neighbouring terms look the same as for velocity U, using ν_{tot}

$$a_N = \frac{\Delta t \nu_{totn}}{(\delta z)^2} \tag{121}$$

$$a_S = \frac{\Delta t \nu_{tots}}{(\delta z)^2} \tag{122}$$

The source term S contain the production and dissipation of T.K.E, k;

$$S = P_s + Pb - \mathcal{E} \tag{123}$$

will be discretised as follows

$$\int_{t^0}^{t^0+\Delta t} \int_s^n S \quad dzdt = \overline{S}\Delta z \Delta t = S_U + S_P k_P^1. \tag{124}$$

where negative terms go into S_P since that term will go on the LHS, and positive terms go into S_U. So the integration of the source terms gives

$$\overline{S}\Delta z\Delta t = \nu_t \left(\frac{\partial u}{\partial z}\right)^2 \Delta z \Delta t - \nu'_{tot} N^2 - (C_\mu^0)^3 \frac{k^{3/2}}{\ell}\Delta z \Delta t. \tag{125}$$

Now we break out a k_P^1 from the second and third term (the negative terms) of this equation, model "the rest" of the k as k_P^0. Identification of terms give

$$\begin{aligned} S_U &= \nu_t \left(\frac{\partial u}{\partial z}\right)^2 \Delta z \Delta t \\ S_P &= \left(-\frac{c'_\mu N^2}{(k^0)^{1/2}} l - (C_\mu^0)^3 \frac{(k_P^0)^{1/2}}{\ell}\right) \Delta z \Delta t \end{aligned} \tag{126}$$

However, the second term on the left hand side of eq. (125) will be positive if stratification is unstable ($N^2 < 0$). Identification of terms in that case yields

$$
\begin{aligned}
S_U &= \left[\nu_t \left(\frac{\partial u}{\partial z}\right)^2 + \nu'_{tot} N^2\right] \Delta z \Delta t \\
S_P &= \left(-(C_\mu^0)^3 \frac{(k_P^0)^{1/2}}{\ell}\right) \Delta z \Delta t
\end{aligned}
\tag{127}
$$

Remember that the source terms (see eq. (59)) are also subject to a division operation so our finite volume formulation of the sources are

$$
\begin{aligned}
S_U &= \nu_t \left(\frac{\partial u}{\partial z}\right)^2 \Delta t \\
S_P &= \left(-\frac{c'_\mu N^2}{(k^0)^{1/2}} l - (C_\mu^0)^3 \frac{(k_P^0)^{1/2}}{\ell}\right) \Delta t
\end{aligned}
\tag{128}
$$

if the stratification is stable, and

$$
\begin{aligned}
S_U &= \left[\nu_t \left(\frac{\partial u}{\partial z}\right)^2 + \nu'_{tot} N^2\right] \Delta z \Delta t \\
S_P &= \left(-(C_\mu^0)^3 \frac{(k_P^0)^{1/2}}{\ell}\right) \Delta z \Delta t
\end{aligned}
\tag{129}
$$

if stratification is unstable.

B Model stability functions and constants

The stability function for ν_t is

$$c_\mu = \frac{0.556 + 0.108R_t}{1 + 0.308R_t + 0.00837R_t^2} \tag{130}$$

where R_t is the turbulent Richardson number

$$R_t = \frac{k^2 N^2}{\mathcal{E}}. \tag{131}$$

The stability function for ν_t' is

$$c_\mu' = \frac{0.556}{1 + 0.277R_t}. \tag{132}$$

Unless the user chooses the MATLAB function sw_dens.m to calculate density, a linear equation of state is used. The linear equation os state is described by Cushman-Rosin and Beckers (2011) and is reads as follows:

$$\rho = \rho_0(1 - \alpha(T - T_0) + \beta(S - S_0)) \tag{133}$$

where α is the coefficient of thermal expansion, β is the coefficient of saline contraction, ρ_0 is a reference density, S_0 is a reference salinity and T_0 is a reference temperature. The default settings of SICT is α=0.17$\cdot$10^{-4} C^{-}1, β=7.6$\cdot$10^{-4}, ρ_0=1000 kg/m^3, S_0=35 psu and T_0=10 psu.

C Detailed instruction to operate SICT

SICT is delivered in a folder (available at https://github.com/josefinaalgotsson/SICT) containing the principal code in function Transient_Transport.m, and a series of complimentary functions. Save the folder on your computer and open MATLAB. Set your current directory to the folder you've downloaded containing the SICT code. Put the folder and including subfolders on the search path. Run the script runtransport.m. This will close all figures including SICT figures, clear all variables including global ones, and set the default figure window style to "normal", in case the setting was "docked". This makes sure old runs of SICT doesn't affect the creation of new global variables, and opens a new SICT figure floating instead of docked. Lastly, it runs the Transient_Transport function opening the SICT figure.

1. Set up domain, boundary conditions and initial conditions.

    - **Set up geometry of your domain** (red box, fig. 4) by changing the inputted values in boxes "Domain height [m]" and "Grid size [m]". When pressing the "Set" button in stage 2, this will affect the scale of the z-axis in your domain (dashed red ellipse, fig 4). If the domain size is not evenly divisible with the grid size, an error message will appear in the command window.
    - **Set boundary conditions for mean velocity, U** (blue box, fig. 4) by choosing either "Fixed flux" or "Fixed value" in both the "Surface:" and "Bottom:" popup menue. This will affect the near surface and near bottom value of the velocity (dashed blue circles, fig 4). As well as update the units of the boundary conditions (to the right in the solid blue box).
    - Make settings for stratification (orange boxes, fig. 4). These will have effect on the form of the stratification (dashed orange ellipse, fig 4).
        - **Choose whether a linear equation of state or the MATLAB-function sw_dens is used to calculate density** (top right orange box, fig. 4) by choosing one of two radiobuttons.
        - **Choose stratification form** (bottom orange box, fig. 4) by choosing either "Linear stratification" or "2-layer stratification" in the pop-up menue.
        - **Make changes to the absolute stratification** (bottom orange box, fig. 4) by changing the values of T1, T2, S1 and S2. **If the 2-layer stratification is chosen, change the top layer thickness** by editing the "Top layer thickness [m]:" box. A bug in SICT updates the calculated buoyancy frequency squared improperly immediately after switching back to linear stratification in the popup menue. It is updated correctly when pressing the "Set" button.
    - **If desirable, make changes to constants** of the equation of state, change the roughness lengths at the surface or the bottom, or change the value of grav-

itational acceleration (green box, fig. 4) by clicking the "Advanced..." button to open an external window. Then click the "Save & Close" button. The button doesn't actually close the new figure, it only makes it invisible. Closing the figure by actually pressing the top corner will close it completely and make it impossible to open the figure again. Use the "Reboot" button and make your settings again.

2. **Apply your settings** by clicking the "Set"-button. This will plot initial velocity and density, and other terms in the plotting windows (large dashed black box in fig 4). SICT also calculates and displays the buoyancy frequency squared if the linear stratification is chosen (dashed black box inside bottom orange box). The clock is also reset by pressing the "Set" button (dashed black box to the left).

3. Make marching settings

   - **Set simulation time** (cyan box, fig. 5) by editing the "T max [s]:" box.
   - **Set time step** (magenta box, fig. 5) by editing the "Delta t [s]:" box.
   - **Set save interval** (violet box, fig. 5) by editing the "Save output every " box. This setting applies from the so far simulated time to the end of the forthcomming simulation. That is, the save interval applies for the additional T max time. It is possible to run the simulation a certain amount of time (Tmax) further, then changing the time step and save interval to run the model an additional Tmax time further and so on. It is also possible, but not recommended to change other settings of SICT during simulation.

4. Run your simulation by clicking the "Run" button. SICT will update the plotting windows when the inputted T max time has been simulated. This time will be added to the clock. It is possible to continue the simulation by simply clicking the "Run" button again. SICT will continue the simulation with the additional inputted T max time. This simulated time will also be added to the clock (Bottom right Tracker panel). In between clicking "Run" it is possible to change both T max, delta t and save interval, enabling the user to adjust the temporal resolution in certain stages of the simulation.

5. Set up output file and folder

   - **Choose filename** (teal box, fig. 5) by either editing the "Filename:" box or pressing the "Browse..." button to choose a pre-existing filename
   - **Choose output folder** (lime green box, fig. 5) by either editing the "Folder:" box or pressing the "Browse..." button to choose a folder.

6. Save output by pressing the "Save" button. The outputted variables include time series for temperature, T, salinity, S and velocity, u as well as other time series,

model constants and settings. The output also contains the boundary conditions, domain grid, and setting for the last used delta t.

C.1 SICT output

The outputted variables saved when using the Save button are listed below. Matrices contain some variables at saved time steps, where the values at one time step are inputted in a column, where the first row corresponds to the surface and the last row corresponds to the bottom of the domain. These columns are stacked side by side where every next column represents the next saved time step.

Table 1: Outputted MATLAB variables from SICT. Boundary conditions (bcNk, bcNS, bcNT...) are defined as a 1 if the boundary condition is Dirichlet and a 2 if the boundary condition is Neuman.

Variable name	
alfa	Thermal expansion coefficient
bcNk	Boundary condition for k at surface boundary
bcNS	Boundary condition for S at surface boundary
bcNT	Boundary condition for T at surface boundary
bcNu	Boundary condition for U at surface boundary
bcSk	Boundary condition for k at bottom boundary
bcSS	Boundary condition for S at bottom boundary
bcST	Boundary condition for T at bottom boundary
bcSu	Boundary condition for u at bottom boundary
beta	Saline contraction coefficient
boundValNk	Boundary condition value for k at surface boundary
boundValNS	Boundary condition value for S at bottom boundary
boundValNT	Boundary condition value for T at surface boundary
boundValNu	Boundary condition value for U at surface boundary
boundValSk	Boundary condition value for at bottom boundary
boundValSS	Boundary condition value for S at bottom boundary
boundValST	Boundary condition value for T at bottom boundary
boundValSu	Boundary condition value for U at bottom boundary
breakDep	depth at which density jump is situated in case of 2-layer stratification
cb	Model constant
Cmu0	Stability function
deltat	Size of time step
deltaz	Size of grid cells
eosChoice	Choice for equation of state
Epssave	Matrix containing dissipation at saved time steps, with depth in rows and time in columns.
faceZ	Column vector containing positions of faces between nodes
filename	Filename of file
g	Gravitational acceleration
H	Depth of domain
kar	von Karmans constant

Table 2: Outputted MATLAB variables from SICT.

Variable name	
ksave	Matrix containing k at saved time steps, with depth in rows and time in columns.
momFluxsave	Matrix containing momentum flux at saved time steps with depth in rows and time in columns
nodeZ	Column vector containing positions of nodes
nutPsave	Matrix containing total turbulent viscosity at saved time steps with depth in rows and time in columns
Pbsave	Matrix containing production of k from buoyancy at saved time steps, with depth in rows and time in columns.
Pssave	Matrix containing production of k from shear at saved time steps, with depth in rows and time in columns.
rho0	Reference density
S0	Reference salinity
Ssave	Matrix containing S at saved time steps, with depth in rows and time in columns.
stratChoice	Choice for stratification
T0	Reference temperature
tsave	Row vector containing t at saved time steps
Tsave	Matrix containing T at saved time steps, with depth in rows and time in columns.
usave	Matrix containing U at saved time steps, with depth in rows and time in columns.
z0b	Roughness length at the bottom
z0s	Roughness length at the surface

D Code

The most important part of my SICT code are presented in this section. The complete code is available at https://github.com/josefinaalgotsson/SICT.

```
function run_Callback(hObject, eventdata, handles)
% hObject    handle to run (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global reset
% Geometry of grid
global deltaz lg l nodeZ lg2 faceZ lb
% Marching
global t deltat tmax iteratedtime  iterCounter tend
global ksave Tsave Ssave  usave tsave m  si Pbsave Pssave Epssave nutPsave
global momFluxsave
% Initial- and boundary conditions
global boundValSu bcSu
global boundValNu bcNu
global boundValNk bcNk
global boundValSk bcSk
global boundValST bcST
global boundValNT bcNT
global boundValSS bcSS
global boundValNS bcNS
% Modeled/transport equations
global F_iu u
global F_ik k
global T S
% Source terms
global Ps Eps Pb
% Model constants and stability functions
global  Cmu Cmu2 cb  momFlux Cmu0
% Stratification and equation os state
global alfa beta T0 S0 eosChoice rho rho0
% Miscellaneous
global N2 nutTot nut2Tot nutP nu nup  g Rt drhodz

%Measure time it takes to reach tmax
tic;
% To reset plots
reset=0;
% Get inputs from UI
getinputs

% Reset the iterated time
iteratedtime=0;
% While next timestep won't make iterated time exceed tmax
while iteratedtime+deltat<=tmax
```

```
% Counter for number of iterations
iterCounter=iterCounter+1;
% Update timevector
timetracker
% --------------------- Length scales --------------------------------
% Density gradient
drhodz = -diff(rho)/deltaz;
% Buoyancy frequency squared
N2=-g/rho0*drhodz;
% Byoyancy length scale squared
lb2=cb.^2*k./N2;
% Limiting lb to the size of domain
% (Length scale can not be larger than domain)
lb2=min(lb2, max(abs(nodeZ)));
% Buoyancy length scale with sign
lb = sign(lb2).*sqrt(abs(lb2));
% Total length scale, neutral
l = lg;
% Total length scale, neutral and squared
l2 = l.^2;                        % Mixing length, total, neutral, squared...
    .
% Total length scale, stable and squared
l(lb2>0) = sqrt(1./(1./lg2(lb>0) + 1./lb2(lb>0))); % Mixing length, ...
    total, stable, squared
l(lb2<0) = sqrt(lg2(lb2<0) - l2(lb2<0).*lg2(lb2<0)./lb2(lb2<0));
% --------------------------------------------------------------------
% -------------------- Stability functions ---------------------------
% Calculate dissipation Epsilon
Eps=Cmu0.^3.*k.^(3/2)./l;
% Calculate Turbulent Richardson number
Rt=k.^2.*N2./(Eps.^2);
%Stability functions
Cmu=(Cmu0+0.108*Rt)./(1+0.308.*Rt+0.00837.*Rt.^2);
Cmu2=Cmu0./(1+0.277.*Rt);
% --------------------------------------------------------------------
% ------------ Turbulent viscosity and tirbulent diffusivity ---------
% Calculate turbulent viscosity nu_t
nut=Cmu.*sqrt(k).*l;
% Calculate turbulent diffusivity nu_t^'
nut2=Cmu2.*sqrt(k).*l;
% Total turbulent viscosity
nutTot=nut+nu;
% Total turbulent diffusivity
nut2Tot=nut2+nup;
% Total turbulent viscosity in nodes
nutP=interp1(faceZ,nutTot,nodeZ(2:end-1));
% --------------------------------------------------------------------
% -------------------- Source terms ----------------------------------
% Velocity shear du/dz in faces
dudz2=(diff(u)/deltaz).^2;
```

```
% Production, P_s, of turbulent kinetic energy from shear
Ps=nutTot.*dudz2;
% Production, Pb, of turbulent kinetic energy from buoyancy
Pb=nut2Tot.*N2;
% Calculate momentum flux
momFlux=nutTot.*-diff(u)./deltaz;
%----------------------------------------------------------------------
% Mask for positive Pb
pmask=Pb>0;
% Mask for negative Pb
nmask=Pb<0;

%---------------------Discretization coefficients----------------------
% Common coefficient
a_P0=1;
%-----------------U----------------------------------------------------
A_iu=deltat.*nutTot(1:end-1)./(deltaz.^2);
C_iu=deltat*nutTot(2:end)./(deltaz.^2);
B_iu=a_P0+A_iu+C_iu;
F_iu=u(2:end-1);
% Dirichlet boundary condition, north
if bcNu==1
    F_iu(1)=F_iu(1)+A_iu(1)*boundValNu;
end
A_iu(1)=0;
% Neuman boundary condition, north
if bcNu==2
    B_iu(1)=a_P0+A_iu(1)+C_iu(1);
    F_iu(1,1)=F_iu(1,1)-boundValNu.*deltat./deltaz;
end
% Dirichlet boundary condition, south
if bcSu==1
    F_iu(end)=F_iu(end)+C_iu(end)*boundValSu;
end
C_iu(end)=0;
% Neuman boundary condition, south
if bcSu==2
    B_iu(end)=a_P0+A_iu(end)+C_iu(end);
    F_iu(end,1)=F_iu(end,1)+boundValSu.*deltat./deltaz;
end
% Calculate new timestep
u(2:end-1,1)=tridiag(B_iu,-A_iu,-C_iu,F_iu);
% Explicit Neuman boundary condition north
if bcNu==2
    u(1)=u(2)-boundValNu.*deltaz./nutTot(1);
end
% Explicit Neuman boundary condition south
if bcSu==2
    u(end)=u(end-1)+boundValSu.*deltaz./nutTot(end);
end
```

```

%----------------------------------------------------------------------
%-------------k-------------------------------------------------------
if bcNu==2
    ustar2=boundValNu;
elseif bcNu==1
    ustar2=momFlux(1);
end
boundValNk=Cmu0^(-2)*ustar2;
A_ik=deltat.*nutP(1:end-1)./deltaz^2;
C_ik=deltat.*nutP(2:end)./deltaz.^2;
B_ik=a_P0+A_ik+C_ik+deltat.*Eps(2:end-1)./...
        k(2:end-1)+deltat.*Pb(2:end-1).*pmask(2:end-1)./k(2:end-1);
F_ik=k(2:end-1)+Ps(2:end-1).*deltat+deltat.*Pb(2:end-1).*nmask(2:end-1);
% Dirichlet boundary condition, north
if bcNk==1
    F_ik(1)=F_ik(1)+A_ik(1)*boundValNk;
    %F_ik(1)=boundValNk;
end
% Neuman boundary condition, north
if bcNk==2
    B_ik(1)=B_ik(1)-A_ik(1);
    F_ik(1,1)=F_ik(1,1)-boundValNk.*deltat./deltaz;
end
A_ik(1)=0;
% Dirichlet boundary condition, south
if bcSk==1
    F_ik(end)=F_ik(end)+C_ik(end)*boundValSk;
end
% Neuman boundary condition, south
if bcSk==2
    B_ik(end)=B_ik(end)-C_ik(end);
    F_ik(end,1)=F_ik(end,1)+boundValSk.*deltat./deltaz;
end
C_ik(end)=0;
% Calculate new timestep
k(2:end-1,1)=tridiag(B_ik,-A_ik,-C_ik,F_ik);
% Explicit Neuman boundary condition north
if bcNk==2
    k(1)=k(2)-boundValNk.*deltaz./nutTot(1);
end
% Explicit Neuman boundary condition south
if bcSk==2
    k(end)=k(end-1)+boundValSk.*deltaz./nutTot(end);
end
% Correct negative k if necessary
k=max(k,1e-20);
%----------------------------------------------------------------------
%----------------Temp---------------------------------------------------
A_iT=deltat.*nut2Tot(1:end-1)./(deltaz.^2);
```

```
C_iT=deltat*nut2Tot(2:end)./(deltaz.^2);
B_iT=a_P0+A_iT+C_iT ;
F_iT=T(2:end-1);
% Dirichlet boundary condition, north
if bcNT==1
    F_iT(1)=F_iT(1)+A_iT(1)*boundValNT;
end
A_iT(1)=0;
% Neuman boundary condition, north
if bcNT==2
    B_iT(1)=a_P0+A_iT(1)+C_iT(1);
    F_iT(1,1)=F_iT(1,1)-boundValNT.*deltat./deltaz;
end
% Dirichlet boundary condition, south
if bcST==1
    F_iT(end)=F_iT(end)+C_iT(end)*boundValST;
end
C_iT(end)=0;
% Neuman boundary condition, south
if bcST==2
    B_iT(end)=a_P0+A_iT(end)+C_iT(end);
    F_iT(end,1)=F_iT(end,1)+boundValST.*deltat./deltaz;
end
% Calculate new timestep
T(2:end-1,1)=tridiag(B_iT,-A_iT,-C_iT,F_iT);
% Explicit Neuman boundary condition north
if bcNT==2
    T(1)=T(2)-boundValNT.*deltaz./nut2Tot(1);
end
% Explicit Neuman boundary condition south
if bcST==2
    T(end)=T(end-1)+boundValST.*deltaz./nut2Tot(end);
end

%----------------------------------------------------------------------
%-----------------Salinity...
    ---------------------------------------------------
A_iS=deltat.*nut2Tot(1:end-1)./(deltaz.^2);
C_iS=deltat*nut2Tot(2:end)./(deltaz.^2);
B_iS=a_P0+A_iS+C_iS ;
F_iS=S(2:end-1);
% Dirichlet boundary condition, north
if bcNS==1
    F_iS(1)=F_iS(1)+A_iS(1)*boundValNS;
end
A_iS(1)=0;
% Neuman boundary condition, north
if bcNS==2
    B_iS(1)=a_P0+A_iS(1)+C_iS(1);
    F_iS(1,1)=F_iS(1,1)-boundValNS.*deltat./deltaz;
    end
    % Dirichlet boundary condition, south
    if bcSS==1
        F_iS(end)=F_iS(end)+C_iS(end)*boundValSS;
    end
    C_iS(end)=0;
    % Neuman boundary condition, south
    if bcSS==2
        B_iS(end)=a_P0+A_iS(end)+C_iS(end);
        F_iS(end,1)=F_iS(end,1)+boundValSS.*deltat./deltaz;
    end
    % Calculate new timestep
    S(2:end-1,1)=tridiag(B_iS,-A_iS,-C_iS,F_iS);
    % Explicit Neuman boundary condition north
    if bcNS==2
        S(1)=S(2)-boundValNS.*deltaz./nut2Tot(1);
    end
    % Explicit Neuman boundary condition south
    if bcSS==2
        S(end)=S(end-1)+boundValSS.*deltaz./nut2Tot(end);
    end
    %---------------------------------------------------------------------
    % Calculate density
    % Use linear equation of state if chosen by user
    if strcmp(eosChoice,'buttonlinear')
        rho=rho0.*(1-alfa.*(T-T0)+beta.*(S-S0));
    % Use sw_dens if chosen by user
    elseif strcmp(eosChoice,'buttonswdens')
        rho=sw_dens(S,T,0);
    end
    % Save timestep if appropriate
    if t/si==round(t/si)
        m=m+1;
        % Velocity
        usave(:,m)=u;
        % Time
        tsave(m)=t;
        % Turbulent kinetic energy
        ksave(:,m)=k;
        % Temperature
        Tsave(:,m)=T;
        % Salinity
        Ssave(:,m)=S;
        % Buoyancy production
        Pbsave(:,m)=Pb;
        % Shear production
        Pssave(:,m)=Ps;
        % Dissipation Epsilon
        Epssave(:,m)=Eps;
        % Turbulent viscosity
        nutPsave(:,m)=nutP;
        % Momentum flux
        momFluxsave(:,m)=momFlux;
    end
end

% Update plots in user interface
updateplots
tend=toc;
% Update time counter in user interface
showtrackers
```