Industrial Training Report

Margarita Smolentseva
UPC, Master Program in Computational Mechanics

Introduction

The current report represents results of the following tasks:

- Develop a code to invert the quasi-geostrophic potential vorticity equation for the ocean given the potential vorticity, vertical stratification and the upper boundary condition
- Validate the code with analytical solutions of oceanographic interest
- Test the code under realistic conditions

The quasi-geostrophic potential vorticity equation is a kind of Navier-Stokes equation. This equation is hard to solve in analytical way. For this reason numerical methods were used for this task. Solution of this equation represents three-dimensional stream function which allows to get the velocity speed of the ocean.

In 3 dimensional space the quasi-geostrophic potential vorticity equation takes the view:

$$\nabla^2\psi + \frac{\partial}{\partial z}\left(\frac{{f_0}^2}{N^2}\frac{\partial\psi}{\partial z}\right) = q$$

With boundary conditions

$$\frac{\partial\psi}{\partial z}|_{z=0} = \frac{b_s}{f_0}$$

$$\frac{\partial\psi}{\partial z}|_{z=-H} = 0$$

Where

$\psi = \psi(x, y, z)$ - stream function;

$b_s = b_s(x, y)$ - surface buoyancy;

$q = q(x, y, z)$- potential vorticity;

$N = N(z)$ - Brunt- Väisälä frequency stratification parameter;

$f_0 = 2\Omega \sin\Phi$ – Coriolis parameter where $\Omega = 7.2921 \cdot 10^{-5}$, Φ – latitude. In case of Mediterranean sea in Barcelona surroundings latitude is considered as 41.39.

Discretization and system solution

In order to discretize the problem finite elements and finite differences methods were considered. Finite elements method allows to get solution near bounds with high accuracy. However, this method requires a lot of resources. Since for the current problem high accuracy on the bounds is not important, finite differences method was chosen for discretization of the problem because it works faster and requires less computational resources.

Let us rewrite the problem in the following way:

$$\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{{f_0}^2}{N^2}\frac{\partial^2\psi}{\partial z^2}+\frac{\partial}{\partial z}\left(\frac{{f_0}^2}{N^2}\right)\frac{\partial\psi}{\partial z}=q$$

Discretizing the problem as

$$\frac{\partial\psi}{\partial z}=\frac{\psi_{ijk+1}-\psi_{ijk-1}}{2\Delta z}$$

$$\frac{\partial^2\psi}{\partial z^2}=\frac{\psi_{ijk-1}-2\psi_{ijk}+\psi_{ijk+1}}{\Delta z^2}$$

$$\frac{\partial^2\psi}{\partial x^2}=\frac{\psi_{i-1jk}-2\psi_{ijk}+\psi_{i+1jk}}{\Delta x^2}$$

$$\frac{\partial^2\psi}{\partial y^2}=\frac{\psi_{ij-1k}-2\psi_{ijk}+\psi_{ij+1k}}{\Delta y^2}$$

For grid $N_x\times N_y\times N_z$ with steps $\Delta x,\Delta y,\Delta z$ in x,y,z directions relatively where $i\in[1,N_x],j\in[1,N_y],k\in[1,N_z]$.

Neumann boundary conditions in vertical direction take the view:

$$\frac{\psi_{ij2}-\psi_{ij0}}{2\Delta z}=\frac{b_s(x_i,y_j)}{f_0}$$

$$\frac{\psi_{ijN_z+1}-\psi_{ijN_z-1}}{2\Delta z}=0$$

In directions x and y periodic boundary conditions are used that is:

$$\psi_{0jk}=\psi_{N_xjk};\ \psi_{N_x+1jk}=\psi_{1jk};\ \psi_{i0k}=\psi_{iN_yk};\ \psi_{iN_y+1k}=\psi_{i1k}$$

Discretized problem takes the view:

$$\frac{\psi_{i-1jk}+\psi_{i+1jk}}{\Delta x^2}+\frac{\psi_{ij-1k}+\psi_{ij+1k}}{\Delta y^2}+e\psi_{ijk}+\psi_{ijk-1}\left(\frac{a}{\Delta z^2}+\frac{b}{2\Delta z}\right)+\psi_{ijk+1}\left(\frac{a}{\Delta z^2}-\frac{b}{2\Delta z}\right)=q_{ijk}$$

Where

$e = e(z) = -2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{a}{\Delta z^2}\right); a = a(z) = \frac{{f_0}^2}{N(z)^2}; b = b(z) = \frac{\partial}{\partial z}\left(\frac{{f_0}^2}{N(z)^2}\right) = \frac{1}{2}\frac{{f_0}^2}{N(z)^{\frac{2}{3}}}\frac{\partial N}{\partial z};$

$$i \in [2, N_x - 1], j \in [2, N_y - 1], k \in [2, N_z - 1]$$

In matrix form, the problem has the following view:

$$X\psi = q$$

Where

$\psi = \left(\psi_{111}\ \psi_{112} \dots\ \psi_{11N_z}\psi_{121}\psi_{122} \dots\ \psi_{1N_y1} \dots \psi_{N_xN_yN_z}\right)$ – vector of n length where $n = N_xN_yN_z$;

$q = \left(q_{111} + s_{11}\ \ q_{112} \dots\ q_{11N_z}\ \ q_{121} + s_{12}\ \ q_{122} \dots\ q_{1N_y1} + s_{1N_y} \dots\ \ q_{N_xN_yN_z}\right)$ – vector of n length;

$$s_{ij} = \frac{2\Delta z b_s(x_i, y_j)}{f_0}\left(\frac{a(\Delta z)}{\Delta z^2} + \frac{b(\Delta z)}{2\Delta z}\right);\ x_i = i\Delta x;\ y_j = j\Delta y;\ i \in [1, N_x], j \in [1, N_y]$$

The matrix X is n by n matrix defined in the following way (I is identity matrix):

$$X = \begin{pmatrix} Y & \frac{1}{\Delta x^2}I & 0 & \cdots & 0 & \frac{1}{\Delta x^2}I \\ \frac{1}{\Delta x^2}I & Y & \frac{1}{\Delta x^2}I & \cdots & 0 & 0 \\ 0 & \frac{1}{\Delta x^2}I & Y & \cdots & \frac{1}{\Delta x^2}I & 0 \\ & & \cdots & & & \\ 0 & 0 & 0 & \cdots & Y & \frac{1}{\Delta x^2}I \\ \frac{1}{\Delta x^2}I & 0 & 0 & \cdots & \frac{1}{\Delta x^2}I & Y \end{pmatrix}_{N_xN_yN_z \times N_xN_yN_z}$$

$$Y = \begin{pmatrix} Z & \frac{1}{\Delta y^2}I & 0 & \cdots & 0 & \frac{1}{\Delta y^2}I \\ \frac{1}{\Delta y^2}I & Z & \frac{1}{\Delta y^2}I & \cdots & 0 & 0 \\ 0 & \frac{1}{\Delta y^2}I & Z & \cdots & \frac{1}{\Delta y^2}I & 0 \\ & & \cdots & & & \\ 0 & 0 & 0 & \cdots & Z & \frac{1}{\Delta y^2}I \\ \frac{1}{\Delta y^2}I & 0 & 0 & \cdots & \frac{1}{\Delta y^2}I & Z \end{pmatrix}_{N_yN_z \times N_yN_z}$$

$$Z = \begin{pmatrix} l & c_1 & 0 & \cdots & 0 & 0 \\ d_2 & l & c_2 & \cdots & 0 & 0 \\ 0 & d_3 & l & \cdots & c_{N_z-2} & 0 \\ & & \cdots & & & \\ 0 & 0 & 0 & \cdots & l & c_{N_z-1} \\ 0 & 0 & 0 & \cdots & d_{N_z} & l \end{pmatrix}_{N_z \times N_z}$$

$$l = -2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right);\ c_i = \frac{a(i\Delta z)}{\Delta z^2} - \frac{b(i\Delta z)}{2\Delta z};\ d_i = \frac{a(i\Delta z)}{\Delta z^2} + \frac{b(i\Delta z)}{2\Delta z};\ i \in [2, N_z - 1]$$

$$c_1 = \frac{2a(\Delta z)}{\Delta z^2};\ d_{N_z} = \frac{2a(N_z\Delta z)}{\Delta z^2}$$

In order to solve this system of equations iterative successive over relaxation method (SOR) was chosen. This method is easy to program but it can take a lot of resources for big set of equations. The current problem is large, however, the matrix consists of zeros principally which saves time and computing resources. For the current problem the method takes the following form. On every step it is necessary to compute residual:

$$\xi_{ijk} = \frac{1}{\Delta x^2}\left(\psi_{i-1jk} + \psi_{i+1jk}\right) + \frac{1}{\Delta y^2}\left(\psi_{ij-1k} + \psi_{ij+1k}\right) + e\psi_{ijk} + \left(\frac{a}{\Delta z^2} + \frac{b}{2\Delta z}\right)\psi_{ijk-1} + \left(\frac{a}{\Delta z^2} - \frac{b}{2\Delta z}\right)\psi_{ijk+1} - q_{ijk}$$

$$\psi_{ijk}^{new} = \psi_{ijk}^{old} - \omega\frac{\xi_{ijk}}{e}$$

Here ω is overrelaxation parameter. The method converges only if $0 < \omega < 2$. This is necessary condition but not insufficient. Also, the matrix should be symmetric and positive-definite. In case if $\omega = 1$, the method is Gauss-Seidel method. When $1 < \omega < 2$, we can talk about over relaxation. It is important to define overrelaxation parameter in a proper way because it influences on speed of method convergence. Sometimes, the error can grow dramatically before convergence set in. In order to avoid it, Chebyshev acceleration was used for ω definition:

$$\omega^{(0)} = 1$$

$$\omega^{\left(\frac{1}{2}\right)} = \frac{1}{2}\left(1 - \frac{\rho_{\text{Jacobi}}^2}{2}\right)$$

$$\omega^{\left(n+\frac{1}{2}\right)} = \frac{1}{2}\left(1 - \frac{\rho_{\text{Jacobi}}^2}{4}\right)\ \text{n} = \frac{1}{2}, 1, \ldots$$

$$\omega^{(\infty)} \to \omega_{optimal}$$

The Chebyshev acceleration allows to decrease the error with every iteration. Here ρ_{Jacobi} is the spectral radius of the Jacobi iteration and ρ_{Jacobi}^2 is the spectral radius of the Gauss-Seidel iteration which is defined as:

$$\rho_{\text{Jacobi}} = \frac{\Delta x \cos\frac{2\pi}{N_x^2} + \Delta y \cos\frac{2\pi}{N_y^2} + \Delta z \cos\frac{\pi}{N_z^2}}{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

As $N(z)$ is a function, in order to calculate the derivative $\frac{\partial N}{\partial z}$, the Newton method was used.

The whole algorithm was realized with Fortran 90, with use of NetCDF libraries for Fortran 90.

Validation

In order to validate the algorithm, the analytical solution was used. For $q = 0, N(z) = n_0 f_0$ the analytical solution takes the view:

$$\hat{\psi}(\vec{k}, z) = \frac{\hat{b_s}(\vec{k})}{n_0 f_0 k} \exp(n_0 k z)$$

Where $\widehat{\ }$ designates the Fourier transform, $\vec{x} = (x, y), \vec{k} = (k_x, k_y), k = \|\vec{k}\|$ is wave number.

The buoyancy is characterized by Gaussian distribution:

$$b_s = b_0 \exp\left(-\frac{x^2}{R^2}\right)$$

Then, after Fourier transform it takes the view:

$$\hat{b_s}(\vec{k}) = \pi b_0 R^2 \exp\left(-\frac{k^2 R^2}{4}\right)$$

Then the stream function then takes the view:

$$\hat{\psi}(\vec{k}, z) = \frac{\pi b_0 R^2}{n_0 f_0 k} \exp\left(-\frac{k^2 R^2}{4}\right) \exp(n_0 k z)$$

In order to check the solution obtained with the established algorithm, the fast Fourier transform was realized with Fortran 90. The data is represented in NetCDF format. The following experiments were received for grid with the size of 500 by 500 by 6, step 1 in all directions.

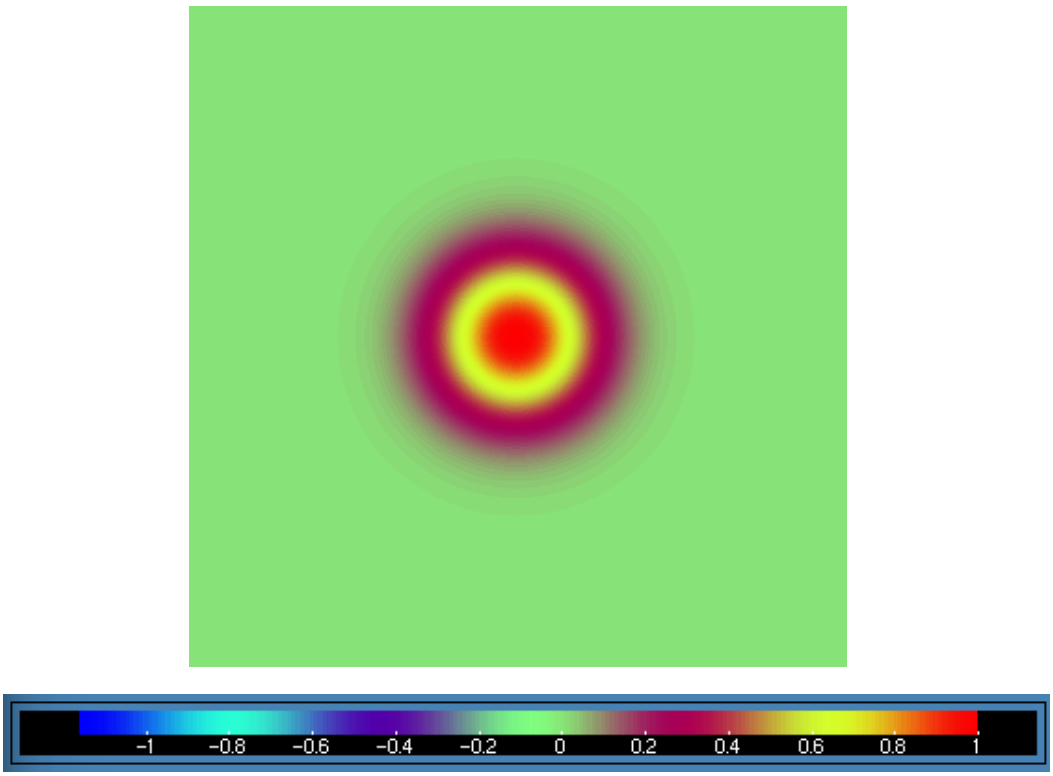

Fig. 1 Analytical solution

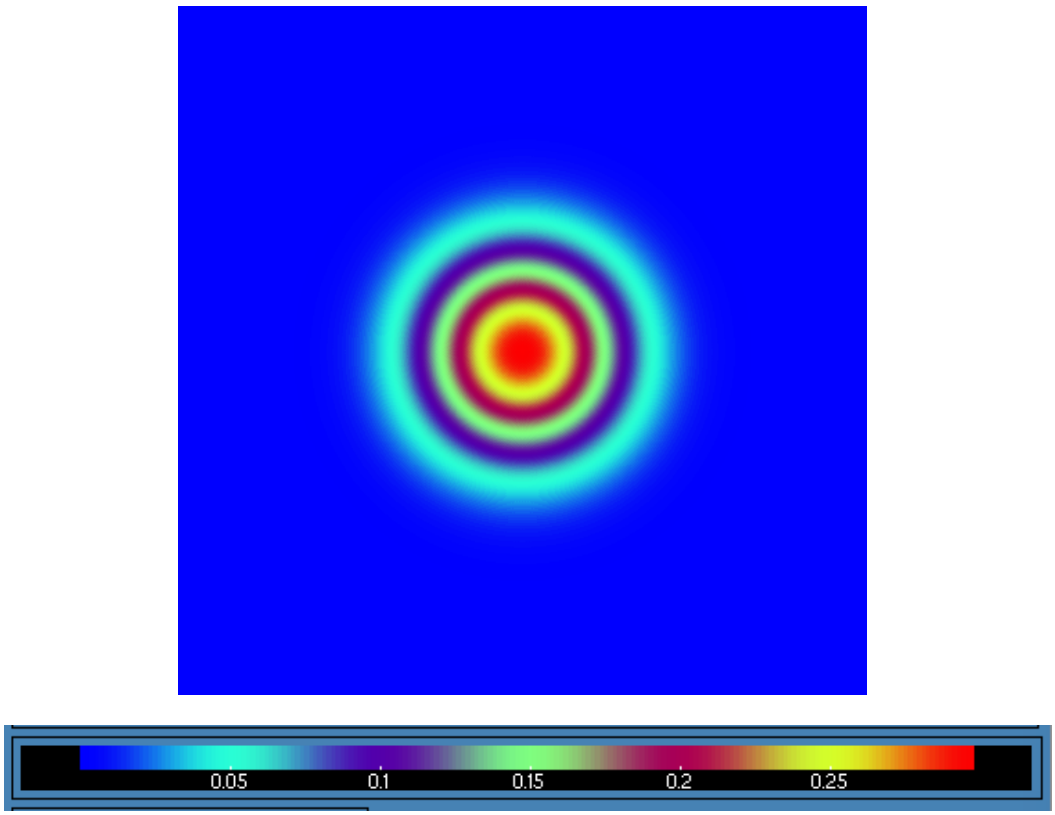

Fig. 2. Result of the numerical computations for analytical buoyancy

As it can be seen from the figures 1 and 2, the numerical solution is close to analytical one and represents the Gaussian integral curvature what was expected.

Test with real data

The following results were obtained while using real data of surface buoyancy and potential vorticity q equal to zero.

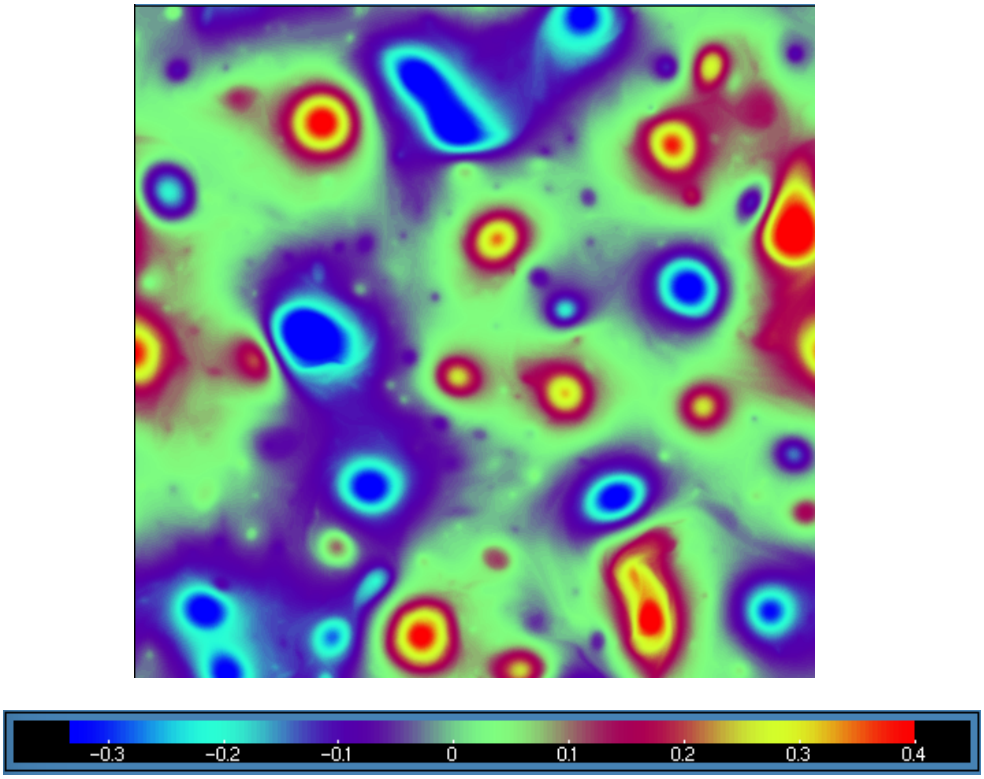

Fig. 3. Exact solution for stream function derived from buoyancy

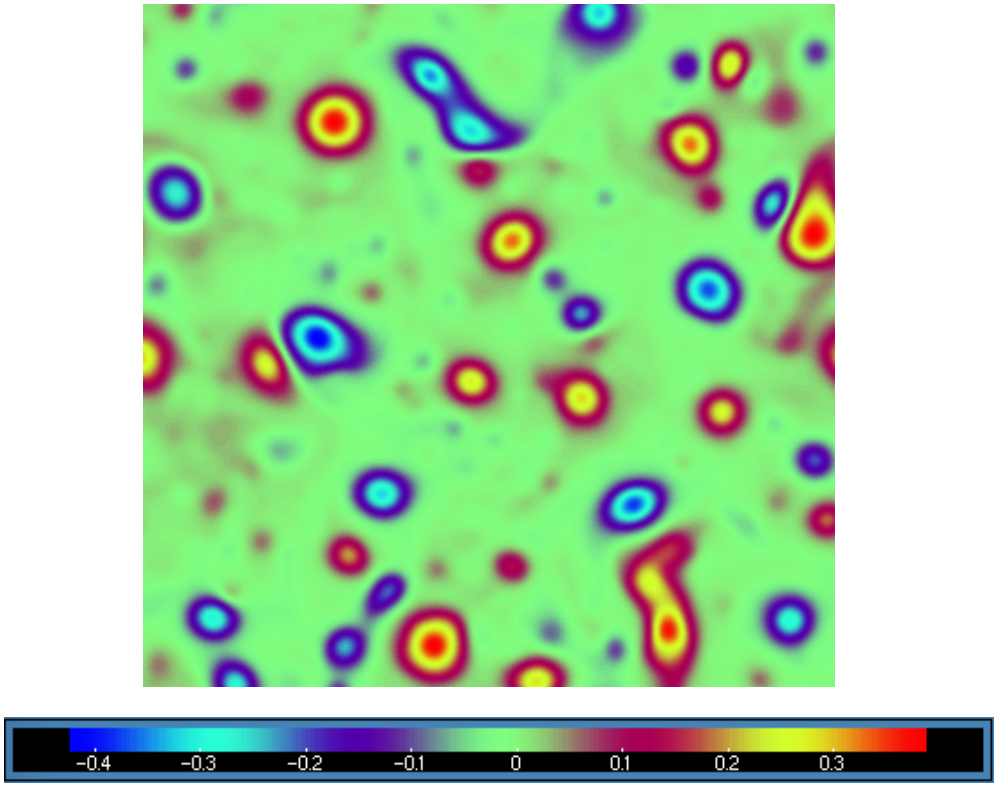

Fig. 4. Stream function derived for buoyancy using numerical computations

The root-mean-square error (RMSE) for the solution is 0.0619702 which was computed as

$$RMSE = \sqrt{\frac{\sum_{i,j,k}\left(\psi_{ijk}^{num} - \psi_{ijk}^{exact}\right)^2}{n}}$$

where $i \in [1, N_x], i \in [1, N_y], i \in [1, N_z], n$ - the total number of points.

As it can be seen from the figures 3 and 4 and the value of root-mean-square-error, the numerical solution has a good accuracy and quite close to the exact solution.

Conclusion

The quasi-geostrophic potential vorticity equation for the ocean given the potential vorticity was discretized by finite differences method and the obtained system of equation was solved with use of successive overrelaxation method. The resulted algorithm was tested with analytical solution and real data and provided solutions with fine accuracy. In the work NetCDF format was used for representation of results and work with appropriate NetCDF libraries for Fortran 90 was studied. The basis of geophysical physics dynamics was studied.

References

Alemayehu Shiferaw, and Ramesh Chand Mittal, 2011, *An Efficient Direct Method to Solve the Three Dimensional Poisson's Equation*, American Journal of Computational Mechanics, pp. 285-293 (http://www.SciRP.org/journal/ajcm)

Benoit Cushman-Roisin, and Jean-Marie Beckers, 2009, *Introduction to Geophysical Fluid Dynamics. Physical and numerical aspects* (New York, USA: Academic Press)

Isaac M. Held, Raymond T. Pierrehumbert, Stephen T. Garner, and Kyle L. Swanson, 1995, *Surphace quasi-geostrophic dynamics, Journal of Fluid Mechanics*, Volume 282, pp. 1-20 (Cambridge, U.K.: Cambridge University Press)

La Casce, J. H., 2012, *Surface Quasigeostrophic Solutions and Baroclinic Modes with Exponential Stratification*, Journal of Physical Oceanography, Volume 42, pp. 569-580

Rindin E.A. 2003, *Methods of solution of mathematical physics problems* (Taganrog, Russia: TRTU Press)

Wlliam H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannary, 2001, *Numerical Recipes in Fortran 77. The art of scientific computing*, second edition (Cambridge, U.K.: Cambridge University Press)

Margarita Smolentseva

14.09.2016