DJLES: Dubreil-Jacotin-Long Equation Solver

Feb 8, 2019
Michael Dunphy

1 Introduction

DJLES is a MATLAB/Octave package that finds a mode-one solution to the DJL equation. Stastna and Lamb (2002) describe the problem setup, weakly nonlinear theory, fully nonlinear theory, and iterative solution procedure. DJLES was inspired by `soliw`, a DJL solving C package resulting from the work of Stastna and Lamb (2002).

DJLES differs from `soliw` by being MATLAB based, which avoids the need to compile C code and manage the resulting raw binary files. When used with MATLAB, it also takes advantage of the built in parallelisation of FFTs which accelerates solution on multi-core computers. Lastly, it easily employs the resolution refinement approach of Dunphy et al. (2011), a feature not available in `soliw`.

If you find this code useful, please cite Dunphy et al. (2011) as it will encourage the author to maintain/improve DJLES. Bug reports, fixes/improvements and additional case files are welcome; please contact mdunphy@uwaterloo.ca for inclusion in a future version.

DJLES has benefited from contributions from Marek Stastna, Kevin Lamb, Chris Subich, Derek Steinmoeller, Jorge Magalhães and José da Silva.

The remainder of this document is a user guide for the code.

2 Quickstart

Download the code and run one of the case files in either MATLAB or Octave. The case files take less than one minute to run on a recent desktop computer, and may need 2-3 minutes on an older system. Upon completion the code will produce a plot of the wave.

3 Usage

Typical usage is to set problem parameters, then call `djles_refine_solution` to solve the DJL equation, which returns the isopycnal displacement field η and wave speed c in MATLAB variables `eta` and `c`, respectively. The required problem parameters are:

- L and H – Domain length and depth (m)
- N_x and N_z – Number of grid points in x and z
- $\bar{\rho}(z)$ and $\frac{\mathrm{d}}{\mathrm{d}z}\bar{\rho}(z)$ – Background density profile and first derivative
  - If using full density, also provide reference density parameter `rho0`.
  - If using non-dimensional density, DJLES will assume `rho0=1`.
- A – Target available potential energy (per unit in y)
  - If using full density, units are kg m/s^2.
  - If using non-dimensional density, units are m^4/s^2.
- $U_{bg}(z)$, $\frac{\mathrm{d}}{\mathrm{d}z}U_{bg}(z)$, and $\frac{\mathrm{d}^2}{\mathrm{d}z^2}U_{bg}(z)$ – Background velocity profile (m/s) and first two derivatives

If a background velocity is not desired, set it to zero (see test cases). Once the solution is complete, `djles_diagnostics` computes the associated velocity fields, vorticity, etc., `djles_pressure` computes the pressure, and `djles_plot` produces a simple plot of these fields.

During the first call to `djles_refine_solution`, we use weakly nonlinear theory to find an initial guess for the solution, which is then iterated to convergence. Subsequent calls to `djles_refine_solution` use the previous fully nonlinear solution as the initial guess. This allows for the strategy of successively adjusting parameters (resolution, wave APE, etc) to ease/accelerate finding a solution. The various test cases demonstrate several solution scenarios.

3.1 Tuning knobs

There are a few tuning knobs. Set them in the case file to override the default.

- `min_iterate` specifies the minimum number of iterations that the solver will take, default is 10.
- `max_iterate` specifies the maximum number of iterations that the solver will take, default is 2000. If you use a very small `epsilon` or `relax` you may need to increase this.
- `NL` is the number of Legendre points to use for the Gauss quadrature used for the APE integral. Values below 5 are probably too small, values in 15–25 are quite good, and values higher than 25 are likely unnecessary. Default is 20.
- `relax` is the underrelaxation factor, $0 <$ `relax` $<= 1$. The value is the fraction of the new iteration, η^{k*} to retain after performing an iteration, that is, $\eta^k = \eta^{k-1} +$ `relax`$(\eta^{k*} - \eta^{k-1})$. The default of 0.5 works well in most cases, and a value of 1 disables underrelaxation. Using a small value can help stabilise the solver at the expense of needing more iterations.
- `g` is the gravitational acceleration constant, default value is 9.81 m/s^2.
- `verbose` controls the verbosity of status updates. A value of 0 disables printing, a value ≥ 1 enables printing updates after each iteration, and a value ≥ 2 enables printing of timing data upon completion. Default is 1.
- `epsilon` controls the stop condition, which is the relative difference between successive iterations. Default 1×10^{-4}. Smaller values will require more solver iterations.

3.2 Miscellaneous

A successful solve will have a relative error of roughly `epsilon`/`relax` decimal places in η. Using a small value of `relax` should be accompanied with a reduction in `epsilon`.

If the solver fails to converge,

1. Ensure that your density profile $\rho(z)$ is stably stratified.
2. Run the solver with a smaller underrelaxation factor. A value of 0.1 or even 0.01 may suppress oscillations and stabilise the solver.
3. Ensure that your density profile (and first derivative) is smooth.
4. Ensure that the domain width L is wide enough to contain the wave. The boundary conditions for the problem are $\eta = 0$ on all boundaries; if your domain is too narrow it will be enforcing that condition at the wrong location.

3.3 Compatibility notes

- Tested with MATLAB R2013a (Linux 64-bit and Windows), R2012a (OSX 10.6.8).
- Tested with Octave v3.8.1, v4.0.1 and v4.4.1 (Linux 64-bit).
- The initial guess part uses `polyeig.m`, and Octave added it in v3.8.0, so it's unlikely to work with Octave older than v3.8.0.
- MATLAB R2014a prints warnings about deprecating `ppval`, you can suppress the warnings with `warning('off','MATLAB:interp1:ppGriddedInterpolant');`

3.4 Numerical Grid

The domain is the rectangular region $[0, L] \times [-H, 0]$ and is discretized into $N_x \times N_z$ grid boxes. The boundary conditions for the problem are zero, and we use the cell centres for the coordinates, that is,

$$x_i^c = \frac{\Delta x}{2} + i\Delta x, \quad i \in 0, 1, 2, \ldots, N_x - 1, \tag{1}$$

$$z_j^c = -H + \frac{\Delta z}{2} + j\Delta z, \quad j \in 0, 1, 2, \ldots, N_z - 1, \tag{2}$$

where $(\Delta x, \Delta z) = (L/N_x, H/N_z)$. The grid shifting routine `djles_shift_grid.m` shifts the data from the cell centres to cell edges,

$$x_i^e = i\Delta x, \quad i \in 0, 1, 2, \ldots, N_x \tag{3}$$

$$z_j^e = -H + j\Delta z, \quad j \in 0, 1, 2, \ldots, N_z. \tag{4}$$

The array sizes are $N_x + 1 \times N_z + 1$ on the cell edges grid.

4 Description of M-files

The code consists of a handful of functions, scripts, and test cases.

4.1 Scripts

The scripts are

```
Filename                  - Description
-------------------------------------------------------------------------------------
djles_common.m            - Sets default parameters and generates grid and wavenumbers.
djles_diagnostics.m       - Computes wave velocities, vorticity, etc from eta and c.
djles_initial_guess.m     - Used by djles_refine_solution.m to get an initial guess for
                            eta & c from WNL theory if an initial guess is not provided.
djles_plot.m              - Plots the solution and some diagnostics.
djles_pressure.m          - Computes the non-hydrostatic pressure and residuals in the
                            governing equations. Uses results from djles_diagnostics.m.
djles_refine_solution.m   - This is the main m-file that each case files calls. It
                            performs the iterative procedure to find a wave solution.
```

4.2 Helper Functions

The helper functions are

```
djles_change_resolution.m - Changes the resolution of the solution.
djles_compute_apedens.m   - Computes the available potential energy density.
djles_diffmatrix.m        - Generates a finite difference differentiation matrix for use
                            in the initial guess procedure.
djles_extend.m            - Extends the input data using specified symmetry.
djles_gradient.m          - Computes the gradient of input data.
djles_quadweights.m       - Returns the 1-dimensional interior grid quadrature weights.
djles_residual.m          - Computes the residual in the DJL equation.
djles_shift_grid.m        - Shifts the input data from cell centres to cell endpoints.
djles_sinequadrature.m    - Computes the interior grid sine quadrature weights for
                            area integrals.
djles_wavelength.m        - Computes an estimate of the wavelength.
```

4.3 Test cases

Included are eight test cases that demonstrate how to use DJLES. A brief description of each follows.

4.3.1 Small APE

Test case: `case_small_ape.m`

This test case demonstrates the parameter regime of no background current, a smooth pycnocline, and small wave APE. The initial guess obtained from weakly nonlinear theory will be "close" to the fully nonlinear DJL solution, so the solver readily converges. Once we find the low resolution wave (32×32), we increase the resolution to 512×256 and iterate to convergence. The low resolution solution is a very good initial guess for the high resolution problem, so the high resolution wave is also readily solved.

4.3.2 Large APE

Test case: `case_large_ape.m`

This test case demonstrates the parameter regime of no background current, a smooth pycnocline, and large wave APE. The initial guess from weakly nonlinear theory will be rather poor. The solution strategy we use is to find a wave with a small APE, and supply that wave as the initial guess to find a wave with a larger APE. Applying this several times yields the final wave with a large APE. In this case we start at one percent of the target APE and raise the APE in five steps. Lastly, we increase the resolution once we reach the target APE. The result is a broad flat crested wave.

4.3.3 Background current

Test case: `case_ubg.m`

We use the same successive solution strategy here by solving without a background current first, and then raise the background current strength in increments until we reach the full background current. We increase epsilon for the intermediate solutions to accelerate the process as we only need them to be of "initial guess" quality. Lastly, we reduce epsilon, and increase the resolution in two increments.

4.3.4 Sharp pycnocline

Test case: `case_sharp_pycnocline.m`

Obtaining a solution with a sharp pycnocline can be difficult for the solver to find directly, and the successive solution strategy works here as well. We begin with a low-resolution wide-pycnocline that is readily solved and begin successively reducing the pycnocline thickness. As we sharpen the pycnocline, we also increase the resolution such that the pycnocline is not under-resolved.

4.3.5 Synthetic data

Test case: `case_synthetic_data.m`

This case matches the parameters of the background current case, except that we use synthetic mooring data as the background profiles. We sample the analytic functions at 25 evenly spaced points and construct continuous profiles using linear interpolating polynomials. Second order differentiation provides the gradients, which are linearly interpolated in the same fashion. The `linear` and `pchip` interpolation methods preserve monotonocity which is important to ensure the interpolated density profile is stable. The `spline` method does not preserve monotonicity so be careful if you use it. As in `case_ubg.m`, we raise the background velocity in four successive steps, followed by three resolution refinements.

4.3.6 Lake Erie data

Test case: `case_lakeerie.m` & `case_lakeerie.txt`

Here we use time-averaged temperature profile data collected from Lake Erie (K. Lamb 2014, pers. comm.). The text file contains two columns, depth (m) and temperature (°C), and we convert the temperature to density with a linear equation of state. There is no surface value so we extrapolate to find a value for $z = 0$. We construct the density and density gradient in the same fashion as in the synthetic data case, and use no background current. We solve the wave at low resolution and then raise the resolution in two steps.

4.3.7 Pineda et al. (2015)

Test case: `case_pineda.m` & `case_pineda_cast1.txt`

This case is intended as a typical shallow continental shelf case where we reproduce the DJL solutions shown in Pineda et al. (2015). The text file contains two columns, depth (m) and density (kg/m^3), and is an approximate reproduction (courtesy Jorge Magalhães and José da Silva) of the smoothed curve of Fig 9a. We use the full density and set the reference density `rho0` as the maximum value, and construct the density and first derivative functions as in the synthetic data case.

The first part of the script finds the wave showcased in Figure 11. After raising the target APE in a few steps at low resolution and large epsilon, we apply several steps of resolution refinement with epsilon reductions. The relative residual of the final wave is order 4×10^{-7}, and matches Pineda et al. (2015) to three significant figures in wave height (-14.1176 m vs. -14.1 m) and two significant figures in phase speed (0.585978 m/s vs. 0.585 m/s).

The second part of the script reproduces the solid curves of Figure 10. We sweep over a range of target APE values and record the wave height, phase speed, and velocity and pressure one metre above the bottom. Each wave is reused as the initial guess for the next APE value as the sweep progresses.

Lastly, we note there is a difference in amplitude between the pressure reported in Pineda et al. (2015) and that computed by DJLES (Figures 10c and 11c). We compute residuals by substituting the wave fields into the Boussinesq equations moving with the wave (see bottom part of `djles_pressure.m`), and find that the residuals are appropriately small, which indicates that the pressure computed by DJLES is correct.

4.3.8 Amazon River mouth data

Test case: `case_amazon.m` & `case_amazon.txt`

This case is intended to find the maximum wave amplitude for internal solitary waves on the continental shelf near the mouth of the Amazon River. The included text file contains three columns: depth (m), density (kg/m^3) and velocity (m/s) projected along the wave direction of propagation (data courtesy of Jorge Magalhães and José da Silva).

The density profile has inversions, so we make ad-hoc adjustments to ensure that the first derivative is always negative (and thus $N^2(z)$ is always positive). Much like the other cases, we start with a small amplitude at low resolution, and incrementally raise the values until we reach a high resolution wave of 35.3 m amplitude. This wave is near its amplitude limit; further increases to the APE will expand the wave as a broad flat crested wave and a larger domain is needed to calculate such a wave.

References

Dunphy, M.; Subich, C., and Stastna, M. Spectral methods for internal waves: indistinguishable density profiles and double-humped solitary waves. *Nonlinear Processes in Geophysics*, 18:351–358, June 2011. doi: 10.5194/npg-18-351-2011. URL https://doi.org/10.5194/npg-18-351-2011.

Pineda, Jesús; Starczak, Victoria; da Silva, José C. B.; Helfrich, Karl; Thompson, Michael, and Wiley, David. Whales and waves: Humpback whale foraging response and the shoaling of internal waves at Stellwagen Bank. *Journal of Geophysical Research: Oceans*, 120(4):2555–2570, 2015. ISSN 2169-9291. doi: 10.1002/2014JC010564. URL https://doi.org/10.1002/2014JC010564.

Stastna, M. and Lamb, K. G. Large fully nonlinear internal solitary waves: The effect of background current. *Physics of Fluids*, 14:2987–2999, September 2002. doi: 10.1063/1.1496510. URL https://doi.org/10.1063/1.1496510.