

Instructions for R-JULES

Catherine Luke and Tim Jupp

June 27, 2011

This document provides instructions for running JULES and its adjoint through a newly created R-interface. By following these instructions you should be able to compile the code, run the model, plot output and optimise parameters. The model itself is known here as R-JULES. You can modify the code for R-JULES, recompile and then observe changes. The adjoint version of the model is known here as ADJULES. For licensing reasons, recalculation of the adjoint following code changes can only be done by us here at Exeter. However, we are able to distribute adjoint code after we have created it and we include the current version here. Thus, you can compile and run the *current* version ADJULES but you will not be able to propagate any code changes you make through to ADJULES.

We include in this distribution a precompiled version of the shared object file `adjules.so`. This was compiled on a 32-bit Windows Vista laptop. Thus, Windows users should be able to skip the compilation steps below for ADJULES. Linux/Mac users, and those wishing to use R-JULES, will have to create the shared object files afresh

1 Retrieving R-JULES

The R-JULES bundle is available online, including the files and data required to run the sites discussed at the recent JULES Science Meeting. To obtain the code:

1. Open a web browser and navigate to <http://empslocal.ex.ac.uk/people/staff/tej202/rjules/>. Download 'rjules.tar.gz' by right-clicking on the link and saving to your computer or homepage.
2. Unpack the download by executing `gunzip rjules.tar.gz` and then `tar -xvf rjules.tar`.

You are now set up with all the source code, files and data required for running R-JULES.

2 Compiling R-JULES

In order to run R-JULES you must compile the relevant code. You can edit and recompile the standard JULES code locally as you wish.

1. Compile the shared object file `rjules.so` by executing (something like) `make rjules COMPILER=gfortran`. Note that you may wish to reset the `COMPILER` option as appropriate for your local machine. You should now be ready to run R-JULES.

3 Running R-JULES

R-JULES is run from within the statistical package R, which can be downloaded for free from the web.

1. Begin a session in R by executing R at the command prompt.
2. Choose a location for your R-JULES run by executing `source("setup.R")`. By following the on-screen instructions you can select the site you wish to use, and, if appropriate, observation data and calibration variables. For a standard (non-adjoint) run you should answer `n` to the question `Would you like to optimise....`
3. To run and initialise R-JULES execute `source("rjules.R")`.
4. To run R-JULES execute `rjules()`.

You have now run R-JULES and generated output for the site.

4 Reading and plotting output

To investigate your results you must read output into R.

1. Execute `source("readoutput.R")` within your R session.
2. You can also read in driving data, by executing `source("readdrive.R")`. Output (and, if chosen, driving data) has been read into R and the available variables for plotting are displayed on screen.

A number of functions for plotting R-JULES output, and comparing to certain observables, has been written for use with the system. The following list describes the basic functionality of each one

plotvar This function allows you to plot a variable (or more than one variable) from the output profiles displayed on screen. You can select variables (**ivar**), profile (**profile**), plotting style (**type**) and range of points to plot (**imin** and **imax**).

Example: `plotvar(ivar=14:17,profile=1,type="p",imin=1,imax=1000)`.

plotcompvar This function allows you to plot one variable against another from the output profiles displayed on screen. You can select variables (**ivar**) and profile (**profile**).

Example: `plotcompvar(ivar=15:16,profile=1)`.

plotcorr This function allow you to compare error correlations in time series of R-JULES output with respect to observations. You can select time series to compare (**its1** and **its2**).

Example: `plotcorr(its1=2,its2=3)`.

plotts This function allows you to plot time series of R-JULES output along with time series of observed data (where available). You can select observed variable (**its**) and range of points to plot (**imin** and **imax**).

Example: `plotts(its=1,imin=1,imax=1000)`.

plottserr This function allows you to plot a time series of errors in R-JULES output with respect to observed data (where available). You can select observed variable (**its**) and range of points to plot (**imin** and **imax**).

Example: `plottserr(its=1,imin=1,imax=1000)`.

plotcompare This function allows you to compare R-JULES output to observed data (where available). You can choose the time series or observable (**its**).

Example: `plotcompare(its=2)`.

You can also execute, for example, `whatvars(profile=1)` to recall the variables available for plotting in profile 1. `storerun(1)` will store the time series and output from your run as the first item in a list of run output. `chooserun(1)` reassigns this output as the current run. You may like to compare the results from two different runs:

comprunvar This function allows you to plot a variable (or more than one variable) from the output profiles displayed on screen and compare it to the same numbered variables from a different run. You can select variables (**ivar**), profile (**profile**), stored run for comparison (**run**) and range of points to plot (**imin** and **imax**). Please note that you must have stored a run in order to use this function.

Example: `plotvar(ivar=14:17,profile=1,run=1,imin=1,imax=1000)`.

5 Performing a parameter optimisation with ADJULES

ADJULES uses the analytical derivative of JULES to efficiently search parameter space for the best values for a subset of JULES parameters to match model output to observations. We provide in the bundle a file `adjules.so` which is precompiled for 32-bit Windows machines. For other systems you will have to execute `make clean` and then (something like) `make adjules COMPILER=gfortran` to create a new shared object file `adjules.so` appropriate to your system.

1. Within an R session, execute `source("setup.R")` and choose to optimise over a set of timeseries.
2. Initialise the adjoint by executing `source("adjules.R")`.
3. You have already chosen the observables against which you would like to optimise in section 3. Type `tsuse` to recall these.
4. To find out which parameters you will be optimising type `z`.
5. To optimise, execute `source("findopt.R")`. This will take some time, as each iteration requires a call to ADJULES.
6. The final output shows you the cost after optimisation, along with initial and final parameter values.

6 Plotting optimised results

A number of functions for plotting optimised output, and comparing to observables, has been written for use with the system. The following list describes the basic functionality of each one.

plotadts This function allows you to plot time series of R-JULES output for initial parameter values and optimised parameter values along with time series of observed data (where available). You can select observed variable (**its**) and range of points to plot (**imin** and **imax**).

Example: `plotadts(its=1,imin=1,imax=1000)`.

plotttserr This function allows you to plot a time series of errors for initial parameter values and optimised parameter values in R-JULES output with respect to observed data (where available). You can select observed variable (**its**) and range of points to plot (**imin** and **imax**).

Example: `plotadtserr(its=1,imin=1,imax=1000)`.

ploterr This function allows you to plot the change in errors in a time series after optimisation. You can select a time series (**its**).

Example: `ploterr(its=1)`.

plotslice This function allows you to plot the cost function for a slice through parameter space, varying one parameter. You can select a location in z-space (**zz**), a parameter to vary (**zw**), range over which to vary the parameter (**zlo** and **zhi**) and the number of points to evaluate at (**n**).

Example: `plotslice(zz=z, zw=1, zlo=0.1, zhi=1)`.

plotadcorr This function allow you to plot error correlations in time series of R-JULES output with respect to observations for initial parameter values and optimised parameter values. You can select time series to compare (**its1** and **its2**).

Example: `plotadcorr(its1=2,its2=3)`.

7 Editing code

- If you would like to change JULES code to try to improve model results, you can edit the code in a text editor (suggest typing `gedit *path to .F90 file* &`). You must then recompile following instructions in section 2.
- To change the JULES control file, find the appropriate location `.jin` file in `'jules.verify_data/control/ver2.2'` and edit by typing `gedit *control*.jin`

&. When you have saved your changes you must set up the location again by executing `source("setup.R")`.

- Similarly, you can edit the code used to choose parameters for optimisation by typing `gedit Rinit1.R &`.

8 Site-specific options

Note that certain sites (specifically BOREAS and snow) do not have associated flux data to compare to JULES output. However, the Atqasuk site has extra observations that can be plotted.

1. Read in Atqasuk soil temperature and soil moisture observations by executing `source(readatq.R)`.
2. The functions `atq_temp` and `atq_moist` add lines to existing plots of JULES output:

`atq_temp` This function will add lines for all available soil temperature observations in a chosen layer. You can choose the layer you are interested in.

Example: `atq_temp(layer=3)`.

`atq_moist` This function will add lines for all soil moisture observations in a chosen layer. You can choose the layer you are interested in.

Example: `atq_moist(layer=3)`.

Please note that you can only use these functions to add lines to an existing plot. They will not generate a new plot.

9 Contact details

The R-JULES system is a work in progress. Please do not hesitate to contact us if you would like to suggest additions or changes, or require help:

- Tim Jupp: `T.E.Jupp@exeter.ac.uk`.
- Catherine Luke: `C.M.Luke@exeter.ac.uk`.