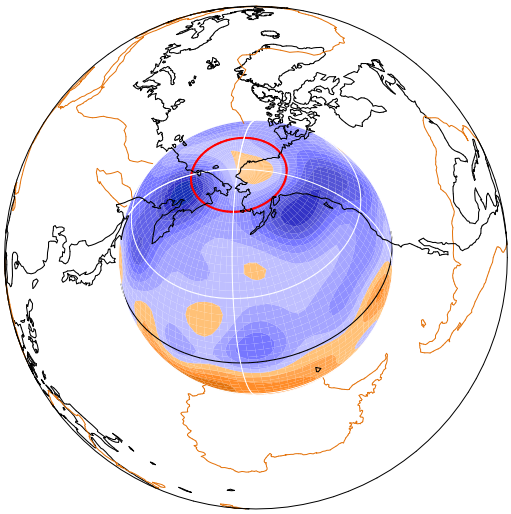
LEOPACK



linons2

LINear **ONS**et of Thermal Convection **2**

Steven J. Gibbons, Oslo

Original document: November 21^{st}, 2001. Updated: October 15^{th}, 2022.

1 linons2

LINear **ONS**et of Thermal Convection **2**

This program is almost identical to `linons1`, differing only in the way we begin the iteration to find the critical Rayleigh number. In `linons1`, a single initial guess, $c_h^{(1)}$, is supplied and the program itself chooses the second guess, $c_h^{(2)}$, as described in the notes.

In `linons2`, the user specifies both $c_h^{(1)}$ and $c_h^{(2)}$ in the input file via the variables `CH1` and `CH2`. The critical Rayleigh number **must** lie between these two values, otherwise the iteration process aborts in failiure. It will ofcourse return growth rates for the c_h values investigated which will help in choosing better initial values.

The equation defining the advection of heat is (see [GR87])

$$\frac{\partial T}{\partial t} + \boldsymbol{u}.\nabla T = \kappa \nabla^2 T + \frac{q}{C_p \rho} \tag{1}$$

where $\boldsymbol{u}$ is the fluid flow, T the temperature, κ the thermal diffusivity ($\mathrm{m^2 s^{-1}}$), q the rate of local heating ($\mathrm{Jm^{-3}s^{-1}}$), C_p the specific heat capacity ($\mathrm{Jkg^{-1}K^{-1}}$) and ρ the density ($\mathrm{kgm^{-3}}$).

The convection codes assume that the temperature, T, is expressed as follows:-

$$T(r,t,\theta,\phi) = T_0(r) + T_1(r,t,\theta,\phi) \tag{2}$$

The steady, basic state temperature distribution, $T_0(r)$, is given the form

$$T_0(r) = -\frac{1}{2} b_1 r^2 + \frac{b_2}{r} + b_3 \tag{3}$$

where b_1, b_2 and b_3 are constants. Its purpose is to define the temperature profile for the sphere or spherical shell, incorporating any internal heating sources. It satisfies

$$\nabla T_0 = -\left(b_1 r + b_2 r^{-2}\right) \boldsymbol{e}_r, \tag{4}$$

where $\boldsymbol{e}_r$ is the unit vector in the radial direction, and

$$\nabla^2 T_0 = -3 b_1. \tag{5}$$

If we substitute the definition (2) into Equation (1) and apply (4) and (5) we derive

$$\frac{\partial T_1}{\partial t} = \kappa \nabla^2 T_1 - 3\kappa b_1 + \frac{q}{C_p \rho} + -\boldsymbol{u}.\nabla T_1 \tag{6}$$

It is now clear that the constant b_1 defines the sources of internal heating with

$$b_1 = \frac{q}{3 C_p \rho \kappa}. \tag{7}$$

If there are no internal heating sources, then $q = 0$ and hence $b_1 = 0$. The constant b_2 is chosen appropriately for systems which have a simple temperature gradient from the inner to the outer boundary.

For numerical simplicity, it is best to solve for temperature functions with homogeneous boundary conditions. We therefore decompose T_1, the perturbation from the basic state temperature,

$$T_1(r,t,\theta,\phi) = \Theta(r,t,\theta,\phi) + \varepsilon T_{\rm a}(r,\theta,\phi). \tag{8}$$

Θ is the function which is solved for in all of the calculations. $T_{\rm a}$ is an additional temperature which is imposed if, for example, an inhomogeneous heat-flux at the outer boundary is required.

If we denote the radial component of the velocity u_r, then applying Equation (8) to Equation (6) gives us the heat equation as applied in all of the programs:

$$c_a \frac{\partial \Theta}{\partial t} = c_d \nabla^2(\Theta + \varepsilon T_{\rm a}) + b_1 u_r r + b_2 \frac{u_r}{r^2} - c_c \boldsymbol{u}.\nabla(\Theta + \varepsilon T_{\rm a}) \tag{9}$$

The constants c_a, b_1, b_2, c_c and c_d are arbitrarily named, with no physical meaning attatched to them. Their use simply allows for any scaling to be applied to the equations. In the codes, c_a is stored in the double precision variable `CA`; and similarly with b_1 (`CB1`), b_2 (`CB2`), c_c (`CC`) and c_d (`CD`).

In the Boussinesq approximation, all density variations except those with respect to the buoyancy force are considered to be negligible, and following the analysis of [GR87], the momentum equation is written

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}.\nabla\boldsymbol{u} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -\nabla\tilde{\omega} + \frac{\delta\rho}{\rho_0}\boldsymbol{g} + \frac{\boldsymbol{J} \times \boldsymbol{B}}{\rho_0} + \nu\nabla^2\boldsymbol{u}, \tag{10}$$

where $\boldsymbol{J}$ and $\boldsymbol{B}$ are respectively the electric current and magnetic field. The scalar function $\tilde{\omega}$ combines the pressure, p, and the centrifugal force such that

$$\tilde{\omega} = \frac{p}{\rho} - \frac{1}{2}|\boldsymbol{\Omega} \times \boldsymbol{r}|^2,$$

and can be removed from the problem by taking the curl of Equation (10). The density variation $\delta\rho$ is expressed in terms of the thermal expansivity, α (K^{-1}), and T, the temperature perturbation from a well mixed state ($\rho = \rho_0$), to give

$$\frac{\delta\rho}{\rho_0} = -\alpha T. \tag{11}$$

The acceleration due to gravity, $\boldsymbol{g}$ is written in terms of the radial vector $\boldsymbol{r}$ as

$$\boldsymbol{g} = -\gamma\boldsymbol{r}, \tag{12}$$

for a constant γ, (s^{-2}). The linear dependence of $\boldsymbol{g}$ on r is a good approximation for the core (see for example [DA81] or [And89]), but would not be appropriate

for the mantle. ν is the viscosity ($\mathrm{m^2 s^{-1}}$) and $\boldsymbol{\Omega} = \Omega \boldsymbol{k}$ is the rotation vector, in terms of the unit vector, $\boldsymbol{k}$. The electric current is related to the magnetic field by

$$\nabla \times \boldsymbol{B} = \mu \boldsymbol{J} \tag{13}$$

where μ is the magnetic permeability and assumed to equal μ_0, the magnetic permeability of free space everywhere.

In order to eliminate the pressure gradient from the momentum equation, we take the curl of Equation (10) and apply equations (11) and (12). If we denote the vorticity (the curl of $\boldsymbol{u}$) by $\boldsymbol{\omega}$, then our vorticity equation becomes

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} &= -\nabla \times (\boldsymbol{u}.\nabla \boldsymbol{u}) - 2\Omega \nabla \times (\boldsymbol{k} \times \boldsymbol{u}) \\ &\quad + \alpha\gamma \nabla \times (T\boldsymbol{r}) + \frac{1}{\rho\mu_0} \nabla \times [(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}] + \nu \nabla^2 \boldsymbol{\omega}. \end{aligned} \tag{14}$$

It is assumed here that the kinematic viscosity, ν, is not a function of space. The basic state temperature, T_0, is a function of radius alone and therefore cannot contribute to the buoyancy term in the vorticity equation. Giving arbitrarily defined names to the scalings which multiply the terms in our equation (14), we write the curl of the momentum equation

$$\begin{aligned} c_e \frac{\partial \boldsymbol{\omega}}{\partial t} &= -c_f \nabla \times (\boldsymbol{u}.\nabla \boldsymbol{u}) - c_g \nabla \times (\boldsymbol{k} \times \boldsymbol{u}) \\ &\quad + c_h \nabla \times [(\Theta + \varepsilon T_{\mathrm{a}})\boldsymbol{r}] + c_j \nabla \times [(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}] + c_i \nabla^2 \boldsymbol{\omega}. \end{aligned} \tag{15}$$

There are two vector quantities in the momentum equation, the velocity $\boldsymbol{u}$ and the magnetic field $\boldsymbol{B}$. $\boldsymbol{B}$ must always satisfy the solenoidal condition

$$\nabla.\boldsymbol{B} = 0 \tag{16}$$

and similarly, for a Boussinesq fluid, $\boldsymbol{u}$ must satisfy

$$\nabla.\boldsymbol{u} = 0. \tag{17}$$

We can therefore express both velocity and magnetic field in poloidal/toroidal decompositions

$$\boldsymbol{B} = \nabla \times \nabla \times \left[\quad {}^{P}B(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right] \quad + \quad \nabla \times \left[\quad {}^{T}B(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right] \tag{18}$$

and

$$\boldsymbol{u} = \nabla \times \nabla \times \left[\quad {}^{P}v(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right] \quad + \quad \nabla \times \left[\quad {}^{T}v(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right]. \tag{19}$$

Note that these definitions are different from those of, for example Bullard and Gellman - [BG54], who use the unit radial vector, $\hat{\boldsymbol{r}}$, instead of $\boldsymbol{r}$.

In this proram, we completely ignore the magnetic field, $\boldsymbol{B}$.

At the onset of thermal instability, we neglect the non-linear and inhomogeneous terms in the equations (9) and (15) and impose an exponential time-dependence of the form

$$\frac{\partial \cdot}{\partial t} = \sigma \cdot \tag{20}$$

where the complex growth rate, σ, is written

$$\sigma = \sigma_{\rm r} + i\sigma_{\rm i}. \tag{21}$$

Our heat and momentum equations become

$$c_a \sigma \Theta \;=\; c_d \nabla^2 \Theta + b_1 u_r r + b_2 \frac{u_r}{r^2} \tag{22}$$

and

$$c_e \sigma \nabla \times \boldsymbol{u} \;=\; -c_g \nabla \times (\boldsymbol{k} \times \boldsymbol{u}) + c_h \nabla \times [\Theta \boldsymbol{r}] + c_i \nabla \times (\nabla^2 \boldsymbol{u})\,. \tag{23}$$

Equations (22) and (23) decouple in m which means that each azimuthal wavenumber can be considered separately without any loss of generality. The equations link terms with different l through the Coriolis force term $(\boldsymbol{k} \times \boldsymbol{u})$. `linons2` treats the parameters c_a, c_d, b_1, b_2, c_e, c_g and c_i as constants and iterates c_h (Rayleigh number) in an attempt to find a critical value, $c_h^{\rm c}$, such that $\sigma_{\rm r} = 0$. $c_h^{(1)}$ and $c_h^{(2)}$ must be on either side of $c_h^{\rm c}$ and are specified in the input file.

Of all the eigenvalues found by the Implicitly Restarted Arnoldi Method ([Arn51],[LSY98], [Sor92]), σ is chosen to be the eigenvalue corresponding to the largest real part, $\sigma_{\rm r}$. If $\sigma_{\rm r}^{(1)}$ (the largest real part on the first iteration) is negative, Once we have two values for c_h and σ, we can begin to iterate; the $j^{\rm th}$ iteration has $c_h = c_h^{(j)}$ and a largest growth rate real part $\sigma_{\rm r}^{(j)}$. The new estimate, $c_h^{(j)}$, is obtained from the previous two estimates by the recursion:

$$\begin{aligned}
\Delta c_h &= c_h^{(j-1)} - c_h^{(j-2)} \\
\Delta \sigma_{\rm r} &= \sigma_{\rm r}^{(j-1)} - \sigma_{\rm r}^{(j-2)} \\
m &= \frac{\Delta \sigma_{\rm r}}{\Delta c_h} \\
c &= \sigma_{\rm r}^{(j-1)} - m c_h^{(j-1)} \\
c_h^{(j)} &= -\frac{c}{m}.
\end{aligned} \tag{24}$$

The new growth rate, $\sigma_{\rm r}^{(j)}$, is found by solving the eigensystem (22, 23) with $c_h^{(j)}$ as the buoyancy multiplier. The iteration continues until either $|\sigma_{\rm r}^{(j)}|$ is less than a low positive number, specified by `CTOL` in the input file, or j exceeds the maximum allowed number of iterations, specified by `MXATT` in the input file.

When solving for the (complex) growth rates, σ, it is (almost) always the one with the greatest real part which we are interested in. The eigenvalue problem is

solved using the implicitly restarted Arnoldi method and the ARPACK software. ARPACK is most effective at finding extremal eigenvalues, that is ones with the greatest magnitude or the greatest real or imaginary parts. If $\boldsymbol{x}$ represents our solution vector $(\boldsymbol{u}, \Theta)$ then our eigensystem (22, 23) can be written

$$\boldsymbol{A}\boldsymbol{x} = \sigma \boldsymbol{M}\boldsymbol{x} \tag{25}$$

where the matrix $\boldsymbol{A}$ contains all the terms multiplied by b_1, b_2, c_d, c_g, c_h and c_i - and the matrix $\boldsymbol{M}$ incorportates the multipliers c_a and $c_e\nabla\times$, for the temperature and velocity parts of the solution vector respectively. The matrix $\boldsymbol{A}$ is invertible[1] and is generated by the subroutine `AVMLTA`. $\boldsymbol{M}$ may be singular if, for example, the Prandtl number is infinite. In this case, one of the time-derivative coefficients, c_a or c_e, is zero; they can ofcourse never both be zero!

We apply a shift-invert scheme to effectively find the largest growth rates. ARPACK will return `nev` eigenvalues closest in the complex plane to a specified shift or *target*,

$$\lambda^{\mathrm{s}} = \lambda^{\mathrm{s}}_{\mathrm{r}} + i\lambda^{\mathrm{s}}_{\mathrm{i}}. \tag{26}$$

For reasons of conservation of memory (the matrix $\boldsymbol{A}$ can be quite enormous if a large number of spherical harmonic radial functions are required to represent the solution), real arithmetic is used and so we are restricted to real shifts (i.e. $\lambda^{\mathrm{s}}_{\mathrm{i}} = 0$). If we subtract $\lambda^{\mathrm{s}}\boldsymbol{M}\boldsymbol{x}$ from both sides of Equation (25) we find

$$\left(\boldsymbol{A} - \lambda^{\mathrm{s}}\boldsymbol{M}\right)\boldsymbol{x} = (\sigma - \lambda^{\mathrm{s}})\boldsymbol{M}\boldsymbol{x}. \tag{27}$$

This eigenvalue problem can be rearranged

$$\left(\boldsymbol{A} - \lambda^{\mathrm{s}}\boldsymbol{M}\right)^{-1}\boldsymbol{M}\boldsymbol{x} = (\sigma - \lambda^{\mathrm{s}})^{-1}\boldsymbol{x}. \tag{28}$$

The closer σ is to our imposed shift, λ^{s}, the greater the magnitude of this new eigenvalue, $(\sigma - \lambda^{\mathrm{s}})^{-1}$, and the more effective ARPACK will be in locating this value. It is worth experimenting with values of λ^{s}.

Generally, the best plan is to set $\lambda^{\mathrm{s}}_{\mathrm{r}}$ to a value slightly greater than the anticipated growth rate, and then our desired eigenvalue, σ, is likely to be that closest in the complex plane to $(\lambda^{\mathrm{s}}_{\mathrm{r}}, 0)$. The only problem is when the imaginary part of the desired eigenvalue, σ_{i}, is very large; ARPACK may simply not find this eigenvalue as there may be others, with smaller real parts, which lie closer in the complex plane to our shift. The imaginary part of the eigenvalue is related to the rate at which the convection rolls drift in longitude. The way to deal with this problem is then probably to use the programs `sbrlinons2` or `sbrlinonsd`. These

[1]Without careful treatment, $\boldsymbol{A}$ may be singular if there is no constraint upon the angular momentum of the solution: i.e. the boundaries are stress-free and our solution can be perturbed by a solid body rotation. Once an additional constraint is added for the specification of angular momentum, $\boldsymbol{A}$ is invertible.

programs treat the problem in a drifting frame of reference and the imaginary part of the growth rates will then be proportional to the drift rate of the rolls relative to the imposed frame of reference. $\sigma_{\rm i}$ can therefore be reduced by selecting an appropriate frame.

The code is compiled by typing

```
make linons2
```

within this directory. Once the executable is created, begin execution by typing

```
linons2 < inputfile
```

adjusting the path if necessary.

The inputs file must have the following format.

```
* input file for linons2
example_aOUTPUT                        : ROOT
0.666666666   1.666666666  2   1    45 : RI, RO, IVELBC, ITHEBC, LU
  10.0  8  20    12   0.000001         : DRSV  NEV  NCV  MXATT CTOL
*-------------------------------------------------------------------------
*NR  ISP  LH  SYM  M  IOF   CA   CB1   CB2   CD   CE   CG   CH1   CH2    CI
*-------------------------------------------------------------------------
 40   1   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1600.0 1800.0 1.0
 50   1   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1400.0 1900.0 1.0
 40   2   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1800.0 1600.0 1.0
 50   2   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1600.0 1720.0 1.0
```

Any line in the input file beginning with an asterisk, `*`, is ignored by the program and can thus be used to enter comments and notes.

The following arguments are common to all of the runs carried out by the execution:

- `ROOT`: First characters in output files to be generated by current run. Running `linons2` with the above input file will create the files `example_aOUTPUT.bc`, `example_aOUTPUT.lh`, `example_aOUTPUT.log`, `example_aOUTPUT.ca`, `example_aOUTPUT.m`, `example_aOUTPUT.cb1`, `example_aOUTPUT.nr`, `example_aOUTPUT.cb2`, `example_aOUTPUT.res`, `example_aOUTPUT.cd`, `example_aOUTPUT.ri`, `example_aOUTPUT.ce`, `example_aOUTPUT.ro`, `example_aOUTPUT.cg`, `example_aOUTPUT.ch`, `example_aOUTPUT.ci`, `example_aOUTPUT.isp` and `example_aOUTPUT.isym` - along with any outputs of the eigenfunctions which are required. The critical "Rayleigh number" (c_h^c) is a function of so many different parameters that I found it difficult to decide how to arrange the output. My first solution was to write all parameters into separate files as demonstrated here: this is not a good solution as it just clutters up the directory with useless files. I recommend commenting out all lines which write files which are not of use. Best probably is to add output lines tailored to your needs.

- `RI`: Radius of the inner boundary, $r_{\rm i}$.
- `RO`: Radius of the outer boundary, $r_{\rm o}$.
- `IVELBC`: Boundary condition for the velocity.
  `IVELBC = 1` $\rightarrow$ rigid boundaries.
  `IVELBC = 2` $\rightarrow$ stress-free boundaries.
- `ITHEBC`: Boundary condition for the temperature.
  `ITHEBC = 1` $\rightarrow$ fixed temperature at both $r = r_{\rm i}$ and $r = r_{\rm o}$.
  `ITHEBC = 2` $\rightarrow$ fixed temperature at $r = r_{\rm i}$ and fixed heat-flux at $r = r_{\rm o}$.
  `ITHEBC = 3` $\rightarrow$ fixed heat-flux at $r = r_{\rm i}$ and fixed temperature at $r = r_{\rm o}$.
- `LU`: Output flag. Set to zero to suppress growth rate information and set to `45` to output growth rate information.
- `DRSV`. The real shift - this is $\lambda_{\rm r}^{\rm s}$ in Equation (26).
- `NEV`. The number of eigenvalues requested.
- `NCV`. The length of the Arnoldi factorisation. See [LSY98]) for details. The source code to the ARPACK routine `DNAUPD`, used in this program states:

```
Number of columns of the matrix V. NCV must satisfy the two
inequalities 2 <= NCV-NEV and NCV <= N.
This will indicate how many Arnoldi vectors are generated
at each iteration.  After the startup phase in which NEV
Arnoldi vectors are generated, the algorithm generates
approximately NCV-NEV Arnoldi vectors at each subsequent update
iteration. Most of the cost in generating each Arnoldi vector is
in the matrix-vector operation OP*x.
NOTE: 2 <= NCV-NEV in order that complex conjugate pairs of Ritz
values are kept together.
At present there is no a-priori analysis to guide the selection
of NCV relative to NEV.  The only formal requrement is
that NCV > NEV + 2.  However, it is recommended that
NCV .ge. 2*NEV+1.  If many problems of the same type are
to be solved, one should experiment with increasing
NCV while keeping NEV fixed for a given test problem.  This will
usually decrease the required number of OP*x operations but it
also increases the work and storage required to maintain the orthogonal
basis vectors.  The optimal "cross-over" with respect to CPU time
is problem dependent and must be determined empirically.
```

- `MXATT`. The maximum number, $j_{\max}$, of modifications to c_h before a growth rate σ is found converged such that $\sigma_{\rm r}^{(j)}$ has a magnitude less than `CTOL`. Setting `MXATT = 1` is a way of simply obtaining growth rates. The iteration will have to give up after one attempt but will write the complex growth rates to the file `root.log`.

- `CTOL`. See `MXATT` above.

All of the lines which follow describe a single run and contain the following variables:

- `NR`. Number of radial grid nodes.

- `ISP`. Radial grid node spacings flag.
  `isp = 1` forces evenly spaced grid nodes from `ESNAAS` and `isp = 2` forces Chebyshev zero spaced nodes from `ZCPAAS`.

- `LH`. Highest spherical harmonic degree, l requested.

- `SYM`. Equatorial symmetry flag.
  `ISYM = 1` $\rightarrow$ equatorially symmetric modes.
  `ISYM = 2` $\rightarrow$ equatorially anti-symmetric modes.

- `M`. Wavenumber, m.

- `IOF` Output flag.
  `IOF = 1` $\rightarrow$ solution is output in the standard format, i.e. `.ints`, `.vecs` and `.xarr` files. Specifically, for run `17`, we would have the files `example_aOUTPUT.run017.ints`, `example_aOUTPUT.run017.vecr` (real part of eigenvector), `example_aOUTPUT.run017.veci` (imaginary part of eigenvector) and `example_aOUTPUT.run017.xarr`. If the iteration has failed to converge to a zero real part growth rate, the eigenvectors corresponding to the highest σ on the last iteration are output. The solution vector output in `root.runXXX.veci` is always the vector stored after `root.runXXX.vecr` by ARPACK. They are a complex pair when the eigenvalue is complex. Otherwise (if we have a real eigenvalue, for example at zero Rayleigh number), `root.runXXX.vecr` contains the corresponding eigenvector and `root.runXXX.veci` is simply the next (unrelated) eigenvector.
  `IOF = 2` $\rightarrow$ solution is output in a radial function display format only.

- `CA`. Scaling parameter c_a in Equation (22).

- `CB1`. Scaling parameter b_1 in Equation (22).

- `CB2`. Scaling parameter b_2 in Equation (22).

- `CD`. Scaling parameter c_d in Equation (22).
- `CE`. Scaling parameter c_e in Equation (23).
- `CG`. Scaling parameter c_g in Equation (23).
- `CH1`. Lower bound for parameter c_h in Equation (23).
- `CH2`. Upper bound for parameter c_h in Equation (23).
- `CI`. Scaling parameter c_i in Equation (23).

1.1 Subprograms required for linons2

SUBS subroutines

```
fopen.f esnaas.f zcpaas.f ontppf.f gauwts.f schnla.f
vthmsr.f cindsw.f svfdcf.f locrnf.f hmfwt.f svfwt.f
xarrwt.f svprnt.f fclose.f fnamer.f ldgnmf.f gfdcfd.f
avmlta.f sbrrfc.f vmeps.f evalas.f evecex.f radvlf.f
matop.f amlp.f  amlc.f  amccfa.f amta.f  amcl.f
amhst.f vecop.f bmrcop.f amsdea.f dvecz.f asvta.f
asvcl.f vesr.f  asvcpl.f amdlt.f amlica.f amccft.f
corcof.f amhsar.f asvdr.f matind.f shvect.f vfcor.f
vf2qst.f cubeop.f fftrlv.f powtwo.f
```

SUBS double precision function

```
pmm.f   pmm1.f  plm.f   dpmm.f  dpmm1.f dplm.f
emmult.f dl.f    sqrll1.f
```

SUBS integer function

```
indfun.f indshc.f
```

BLAS double precision function

```
dnrm2.f ddot.f  dasum.f
```

BLAS integer function

```
idamax.f
```

BLAS subroutines

```
daxpy.f dgemm.f dtrsm.f dgemv.f dswap.f dcopy.f
dger.f  dscal.f dtrmm.f dtbsv.f drot.f  dtrmv.f
```

ARPACK subroutines

```
dnaupd.f dneupd.f dnaup2.f dvout.f ivout.f second.f
dstatn.f dmout.f dgetv0.f dnaitr.f dnconv.f dneigh.f
dngets.f dnapps.f dlaqrb.f dsortc.f
```

LAPACK subroutines

```
dgetrf.f dgetri.f dgbtrf.f dgetf2.f dlaswp.f xerbla.f
dtrtri.f dgbtf2.f dgbtrs.f dlahqr.f dgeqr2.f dlacpy.f
dlaset.f dorm2r.f dtrevc.f dtrsen.f dtrti2.f dlabad.f
dlanv2.f dlarfg.f dlarf.f dlaln2.f dlacon.f dtrexc.f
dtrsyl.f dlarnv.f dlascl.f dlartg.f dlassq.f dladiv.f
dlaexc.f dlasy2.f dlaruv.f dlarfx.f
```

LAPACK double precision function

```
dlapy2.f dlamch.f dlanhs.f dlange.f
```

LAPACK integer function

```
ilaenv.f
```

LAPACK logical function

```
lsame.f
```

1.2 Run-time limitations

Several parameters are set at the outset which limit the physical size of the problem.

```
 INTEGER NRMAX, LHMAX, NHMAX, NPHMAX, NTHMAX, KLMAX,
1        NBNDMX, NDCS, NDRVM, ISVMAX, NPMAX, LHLH2M, NCFM,
2        NBN, NCVM, NRUNM
 PARAMETER ( NRMAX = 150, LHMAX = 62, NHMAX = 100, NBN = 3,
1            NTHMAX = 64, NPHMAX = 128, KLMAX = (NBN+1)*NHMAX-1,
2            NBNDMX = 3*KLMAX+1, NDCS = 3, NDRVM = 4,
3            ISVMAX = NRMAX*NHMAX )
 PARAMETER ( NPMAX = (LHMAX+1)*(LHMAX+2)/2,
1            LHLH2M = LHMAX*(LHMAX+2), NCFM = 2*NBN + 1,
2            NCVM = 24, NRUNM = 100 )
```

If the values are insufficient, then change them and recompile. (Note that `NDRVM` and `NDCS` are not size dependent and should not be changed.)

- NRMAX is the maximum permitted number of radial grid nodes.
- LHMAX is the highest permitted spherical harmonic degree, l.
- NHMAX is the maximum permitted number of radial functions (total: all types).
- NBN is the number of bounding nodes on either side of the finite difference stencil. NBN = 3 is recommended value.
- NTHMAX is the maximum permitted number of grid nodes in θ for Gaussian quadrature in the spherical transforms.
- NPHMAX is the maximum permitted number of grid nodes in ϕ for the Fast Fourier Transforms.
- NCVM is the maximum value of NCV.
- NRUNM is the maximum number of independent runs permitted.

1.3 Outputs from LINONS2

If the filename stem "root" was specified in the input file, the files root.bc, root.lh, root.log, root.ca, root.m, root.cb1, root.nr, root.cb2, root.res, root.cd, root.ri, root.ce, root.ro, root.cg, root.ch, root.ci, root.isp and root.isym will all be created. These files just contain the values of various parameters for each run. My advice is to comment out the lines which write these files.

The most important file is root.log which records the values of all of the eigenvalues found. For example, with NEV = 8 for the problem in Section (1.4.1) we get, for the first iteration of the first run,

```
-------------------------------------------
Iteration    1:   8 eigenvalues converged.
-------------------------------------------
Eval  1 (      -1.4185733,      -1.8011187) Res =        0.0000000
Eval  2 (      -1.4185733,       1.8011187) Res =        0.0000000
Eval  3 (     -24.8369785,      -0.9971629) Res =        0.0000000
Eval  4 (     -24.8369785,       0.9971629) Res =        0.0000000
Eval  5 (     -42.8117246,      -0.0852464) Res =        0.0000002
Eval  6 (     -42.8117246,       0.0852464) Res =        0.0000002
Eval  7 (     -49.4945686,      -1.4313750) Res =        0.0000009
Eval  8 (     -49.4945686,       1.4313750) Res =        0.0000009
 Iter.    1 R=   1.60000000D+03: ( -1.41857327D+00, -1.80111865D+00)
```

The numbers in brackets are the real and imaginary parts of the eigenvalues and the residual is the arithmetic norm of $(\boldsymbol{A}\boldsymbol{x} - \sigma\boldsymbol{M}\boldsymbol{x})$. The last line summarises the eigenvalue with highest growth rate real part. Indeed, typing

```
grep ' Iter. ' example_aOUTPUT.log
```

reveals the progression towards our converged state:

```
 Iter.    1 R=   1.60000000D+03: ( -1.41857327D+00, -1.80111865D+00)
 Iter.    2 R=   1.61600000D+03: ( -1.18262874D+00, -1.81558784D+00)
 Iter.    3 R=   1.69619707D+03: (  1.41629053D-04, -1.88740428D+00)
 Iter.    4 R=   1.69618747D+03: ( -1.54813762D-08, -1.88739575D+00)
```

The file `root.res` provides a brief summary of results for runs with c_h^c, σ_r and σ_i: one line for each run undertaken.

1.4 Sample runs of linons2

The directory

```
LEOPACK_linons2
```

contains an example input file.

1.4.1 Example a

Scaling length, time and temperature by $d = r_\mathrm{o} - r_\mathrm{i}$, d^2/κ and d^2/b_1 respectively, gives us the non-dimensional forms of equations (22) and (23),

$$\frac{\partial\Theta}{\partial t} + \boldsymbol{u}.\boldsymbol{r} = \nabla^2\Theta \tag{29}$$

and

$$P_r^{-1}\nabla\times\frac{\partial\boldsymbol{u}}{\partial t} + E^{-1}\nabla\times(\boldsymbol{k}\times\boldsymbol{u}) = R\nabla\times\Theta\boldsymbol{r} + \nabla\times\nabla^2\boldsymbol{u}. \tag{30}$$

The three controlling parameters are given by

$$P_r = \frac{\nu}{\kappa}\ , \quad R = \frac{\alpha b_1\gamma d^6}{\nu\kappa}\ , \quad \text{and} \quad E = \frac{\nu}{2\Omega d^2}. \tag{31}$$

Our first example is for the onset of thermal convection for fixed temperature at inner and outer boundaries, $r_\mathrm{i}/r_\mathrm{o} = 0.4$ and stress-free boundaries. We know that the critical Rayleigh number is approximately 1600 and the critical m is 4. With the following input file, we seek critical Rayleigh numbers for infinite Prandtl number, $E = 0.01$ and $m = 4$. We will restrict our calculations to `LH = 14` for the time being.

```
* input file for linons2
example_aOUTPUT                          : ROOT
0.666666666   1.666666666  2   1    45 : RI, RO, IVELBC, ITHEBC, LU
  10.0  8  20    12   0.000001          : DRSV  NEV  NCV  MXATT CTOL
*-------------------------------------------------------------------------
*NR  ISP  LH  SYM  M  IOF   CA   CB1   CB2   CD   CE   CG   CH1   CH2    CI
*-------------------------------------------------------------------------
 40   1   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1600.0 1800.0 1.0
 50   1   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1400.0 1900.0 1.0
 40   2   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1800.0 1600.0 1.0
 50   2   14   1   4   0 1.0   1.0   0.0  1.0  0.0  100.0 1600.0 1720.0 1.0
```

We obtain the `.res` file

```
1.69618747D+03 -2.02669597D-08 -1.88739575D+00
1.69653823D+03 -2.47717928D-07 -1.88577621D+00
1.69690645D+03 -2.11945412D-08 -1.88371916D+00
1.69688936D+03 -8.12494960D-10 -1.88360784D+00
```

References

[And89] D. L. Anderson. *Theory of the Earth*. Blackwell Scientific Press, Oxford., 1989.

[Arn51] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. J. Appl. Math.*, 9:17–29, 1951.

[BG54] E. C. Bullard and H. Gellman. Homogeneous dynamos and terrestrial magnetism. *Phil. Trans. R. Soc. Lond. A*, 247:213–278, 1954.

[DA81] A. M. Dziewonski and D. L. Anderson. Preliminary reference earth model. *Phys. Earth Planet. Inter.*, 25:297–356, 1981.

[GR87] D. Gubbins and P. H. Roberts. Magnetohydrodynamics of the earth's core. In J. A. Jacobs, editor, *Geomagnetism Volume* II, pages 1–183. Academic Press, 1987.

[LSY98] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *Arpack users guide: Solution of large scale eigenvalue problems by implicitly restarted Arnoldi methods*. SIAM, 1998.

[Sor92] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Analysis and Applications*, 13:357–385, 1992.