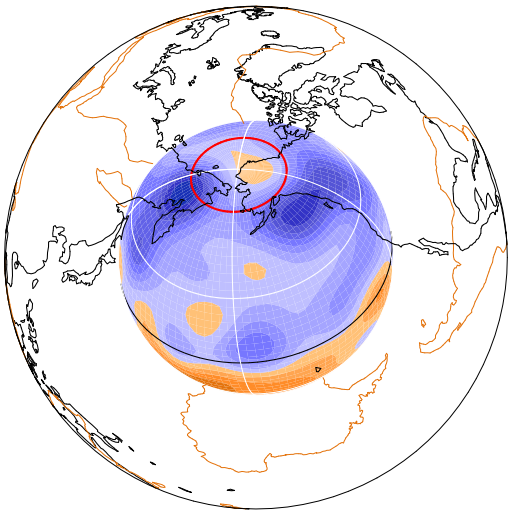
LEOPACK



sbrlinons1

Solid **B**ody **R**otation **LIN**ear **ONS**et of Thermal Convection **1**

Steven J. Gibbons, Oslo

Original document: November 21^{st}, 2001. Updated: October 24^{th}, 2022.

1 sbrlinons1

Solid **B**ody **R**otation **LIN**ear **ONS**et of Thermal Convection **1**

Solves iteratively for a critical Rayleigh number, $c_h^{\rm c}$ in the onset of thermal convection problem

$$c_a\sigma\Theta \;=\; c_d\nabla^2\Theta + b_1 u_r r + b_2\frac{u_r}{r^2} \tag{1}$$

and

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

which is fully described for the program `linons1`. This program differs in that it solves for the flow and temperature distribution in a drifting frame of reference (r',θ',ϕ') such that

$$r'=r,\qquad \theta'=\theta,\qquad \phi'=\phi-c_{\rm s}t. \tag{3}$$

This is achieved by constantly imposing a solid body rotation,

$$c_{\rm s}\boldsymbol{u}_{\rm s} = c_{\rm s}r\sin\theta\boldsymbol{e}_\phi, \tag{4}$$

where $\boldsymbol{e}_\phi$ is the unit vector in the ϕ direction, upon the solution and modifying the Coriolis force term accordingly to take account of the changed frame of reference.

The stand-alone source code version of the program is compiled by typing

```
make sbrlinons1
```

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

```
sbrlinons1 < inputfile
```

The inputs file must have the following format.

```
* input file for sbrlinons1
example_aOUTPUT                          : ROOT
0.666666666   1.666666666  2   1    45 : RI, RO, IVELBC, ITHEBC, LU
 800.0  4  20    12   0.0001  1       : DRSV  NEV  NCV  MXATT CTOL  IA
*-------------------------------------------------------------------------
*     NR  ISP  LH  SYM  M  IOF  REYSBR
* CA    CB1   CB2   CD    CE   CG    CH1    CH2     CI
*-------------------------------------------------------------------------
      50   2   26   1   4   0    0.0
 0.01   1.0   0.0   1.0   1.0  1000.0 1900.0  1950.0  1.0
      50   2   26   1   6   0    0.0
 0.1    1.0   0.0   1.0   1.0  1000.0 4200.0  4600.0  1.0
      50   2   26   1   6   0    0.0
 1.0    1.0   0.0   1.0   1.0  1000.0 13000.0 13500.0  1.0
      50   2   26   1   6   0    0.0
 10.0   1.0   0.0   1.0   1.0  1000.0 20000.0 20500.0  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 `sbrlinons1` with the above input file will create the files `example_aOUTPUT.res` and `example_aOUTPUT.log` - along with any outputs of the eigenfunctions which are required.

- `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 $\lambda_{\rm r}^{\rm s}$

- `NEV`. The number of eigenvalues requested.

- `NCV`. The length of the Arnoldi factorisation. See [LSY98]).

- `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.

- `IA`. Flag for setting number of θ points for Gaussian quadrature. Can take either one of the two values:
  `IA = 1` sets $N_\theta = L_{\max} + 2$.
  `IA = 2` sets $N_\theta = 3L_{\max}/2$.

The following (uncommented) lines come in pairs, with two consecutive lines giving the parameters for a single run. The first one of each of these pairs must contain the inputs

- `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.
- `REYSBR`. The speed of the drifting frame of reference imposed by adding the solid body rotation. `REYSBR` is the value c_s as given in Equation (4). Note that adding a positive solid body rotation to the flow means that we are treating the flow in a drifting frame of reference moving with drift-rate, $-c_\mathrm{s}$.

The second of these lines contains the numbers

- `CA`. Scaling parameter c_a in Equation (1).
- `CB1`. Scaling parameter b_1 in Equation (1).
- `CB2`. Scaling parameter b_2 in Equation (1).

- `CD`. Scaling parameter c_d in Equation (1).
- `CE`. Scaling parameter c_e in Equation (2).
- `CG`. Scaling parameter c_g in Equation (2).
- `CH1`. First estimate for critical scaling parameter c_h in Equation (2).
- `CH2`. Second estimate for critical scaling parameter c_h in Equation (2). Note that one of `CH1` and `CH2` must be below the critical value and the other must be above, otherwise the iteration process aborts.
- `CI`. Scaling parameter c_i in Equation (2).

1.1 Subprograms required for sbrlinons1

SUBS subroutines

```
fopen.f esnaas.f zcpaas.f nphpf.f gauwts.f schnla.f
sbrvmr.f vthmsr.f cindsw.f vecop.f svfdcf.f iocrbo.f
hmfwt.f svfwt.f xarrwt.f svprnt.f fclose.f fnamer.f
ldgnmf.f gfdcfd.f avmato.f sbrrfc.f vmeps.f evalas.f
evecex.f radvlf.f matop.f amlp.f  amlc.f  amccfo.f
amta.f  amcl.f  amhst.f iv0gto.f ivgt0o.f iv0cvo.f
ivcv0o.f bmrcop.f amsdea.f dvecz.f asvta.f asvcl.f
vesr.f  asvcpl.f amdlt.f amlica.f amccft.f corcoo.f
amhsar.f invgtt.f shveco.f vfdp.f  forsso.f innlca.f
invcvt.f vfcp.f  vf2qso.f asvdr.f matind.f vfcor.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, NHOMAX, ISOMAX
 PARAMETER ( NRMAX = 50, LHMAX = 62, NHMAX = 100, NBN = 3,
1            NTHMAX = 64, NPHMAX = 128, KLMAX = (NBN+1)*NHMAX-1,
2            NBNDMX = 3*KLMAX+1, NDCS = 4, NDRVM = 4,
3            ISVMAX = NRMAX*NHMAX, NHOMAX = 1,
4            ISOMAX = NRMAX*NHOMAX )
 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 SBRLINONS1

If the filename stem "root" was specified in the input file, the files `root.res` and `root.log`. The `.log` file contains information on the growth rate eigenvalues

```
-------------------------------------------
Iteration    1:   4 eigenvalues converged.
-------------------------------------------
Eval  1 (      -0.6003633,    -122.7856575) Res =        0.0000000
Eval  2 (      -0.6003633,     122.7856575) Res =        0.0000000
Eval  3 (     -46.4718232,     -58.3372256) Res =        0.0000009
Eval  4 (     -46.4718232,      58.3372256) Res =        0.0000009
 Iter.    1 R=   1.90000000D+03: ( -6.00363331D-01, -1.22785657D+02)
```

with a final line summarising the current estimate of c_h^c and the real and imaginary parts of the eigenvalue with the greatest real part. The `.res` file gives one line for each successful run with the numbers `M`, `LH`, `NR`, `CG`, `CH_crit`, `sigma_r`, `sigma_i` and `REYSBR`, in the format `(I4,I4,I4,1PD16.8,1PD16.8,1PD14.6,1PD16.8,1PD16.8)`.

1.4 Sample runs of sbrlinons1

The directory

`$LEOPACK_DIR/SAMPLERUNS/SBRLINONS1`

contains example input files and model output. Do not under any circumstances edit these files, as these examples should serve as a control for the correct working of the code. After compiling the program, copy the `.input` files to another directory, run the code and confirm that the output agrees with that in the directory.

1.4.1 Example a

We attempt to reproduce the work of [ZB87]: specifically the results in Table 1 of this paper. They define a temperature distribution

$$T_s = T_0 - \beta \frac{r^2}{2} \tag{5}$$

and gravity given by

$$\boldsymbol{g} = -\gamma \boldsymbol{r} \tag{6}$$

Length, time and temperature are scaled by $d = r_\mathrm{o}(1-\eta)$, d^2/ν and $\beta d^2 \nu/\kappa$ respectively, where ν is the kinematic viscosity, κ the thermal diffusivity and η the ratio $r_\mathrm{i}/r_\mathrm{o}$ which is here set to 2/5. They proceed to give their equations as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \tau\boldsymbol{k}\times\boldsymbol{u} = -\nabla\pi + R\boldsymbol{r}\Theta + \nabla^2\boldsymbol{u}, \tag{7}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{8}$$

and

$$P\left(\frac{\partial}{\partial t} + \boldsymbol{u}\cdot\nabla\right)\Theta = \boldsymbol{u}\cdot\boldsymbol{r} + \nabla^2\Theta \tag{9}$$

where the Rayleigh number, R, the Taylor number, $T = \tau^2$, and the Prandtl number, P, are given by

$$R = \frac{\alpha\gamma\beta d^6}{\nu\kappa}, \quad T^{1/2} = \frac{2\Omega d^2}{\nu}, \quad \text{and } P = \frac{\nu}{\kappa} \tag{10}$$

Performing a simple comparison between these equations and the scalings in equations (1) and (2) allows us to apply the coefficients as provided in the file `example_a.input`. As we see from this input file, the drift-rate of the moving frame of reference is set to zero.

Table (1) gives the results from `sbrlinons1` at two different numerical resolutions, together with the best (i.e. highest numerical resolution) result of [ZB87] for each of the four examples considered in Table 1 of this paper. The convergence of the results from `sbrlinons1` appears very good and appears to indicate that the results of [ZB87] would converge towards them, had the computing facilities of the time allowed. The only result which seems suspicious is the $P = 0.01, m = 4$ value. We shall investigate this further in the next example.

P (m)	0.01 (4)	0.1 (6)	1.0 (6)	10.0 (6)
Best result	$R_c = 1935.1$	$R_c = 4426.5$	$R_c = 13182$	$R_c = 20204$
[ZB87]	$c = -113.4$	$c = 17.14$	$c = 5.698$	$c = 0.6772$
`LH = 24`	$R_c = 1949.0$	$R_c = 4431.4$	$R_c = 13202$	$R_c = 20256$
`NR = 40`	$\|c\| = 30.656$	$\|c\| = 17.116$	$\|c\| = 5.717$	$\|c\| = 0.6771$
`LH = 26`	$R_c = 1949.5$	$R_c = 4432.1$	$R_c = 13202$	$R_c = 20256$
`NR = 50`	$\|c\| = 30.656$	$\|c\| = 17.116$	$\|c\| = 5.717$	$\|c\| = 0.6771$

Table 1: Critical Rayleigh numbers and drift rates, comparing values obtained by the program sbrlinons1 and those obtained by [ZB87]. Note that the drift-rates obtained by sbrlinons1 are simple the magnitude of the imaginary part, σ_i, divided by the wavenumber, m, and as such are undetermined to a factor of -1.

1.4.2 Example b

We see in Table (1) that the four cases $P = 0.01$, $P = 0.1$, $P = 1.0$ and $P = 10.0$ have solutions with drift-rates with magnitudes 30.656, 17.116, 5.717 and 0.6771 respectively. Given that the numerical resolution appears to be adequate, we shall use only the lower of the two resolutions used in example a). The direction of drift could be determined by examining the eigenvectors, but here we shall try by imposing solid body rotations of $|c|$ (and so measuring in a frame of reference with drift-rate $-|c|$) and $-|c|$ (therefore measuring in a frame of reference with drift-rate $|c|$).

Note that the critical Rayleigh number, R^c, must be EXACTLY the same, regardless of the drifting frame, and so we can refine our estimates `CH1` and `CH2` based on the results of the previous run.

The first run demanded by the input file `example_b.input` failed because the first two estimates for the critical Rayleigh number both returned eigenvalues with positive real parts. Now, in the previous section, the critical Rayleigh number for the $m = 4$, $P = 0.01$ case was found to be `1949`. However, for exactly the same numerical resolution we now find that a Rayleigh number of `1945` gives a positive growth rate. This means that there is another mode which is more unstable than that located by the previos run which the ARPACK software was simply not able to find. This is because only 4 eigenvalues were requested and there were 2 pairs of eigenvalues closer in the complex plane to our real shift (which is quite large: `DRSV = 1000.0`) than those corresponding to the most unstable solution. The first figure we obtained in the above example is therefore wrong.

The results from the first iterations of the first run are as follows:

- $c_s = 30.656$

```
-----------------------------------------------
Iteration    1:   4 eigenvalues converged.
-----------------------------------------------
```

```
Eval  1 (       0.0170820,    -331.0361451) Res =        0.0000001
Eval  2 (       0.0170820,     331.0361451) Res =        0.0000001
Eval  3 (      -0.0486312,    -245.2621327) Res =        0.0000005
Eval  4 (      -0.0486312,     245.2621327) Res =        0.0000005
 Iter.    1 R=   1.94500000D+03: (  1.70819705D-02, -3.31036145D+02)
```

- $c_s = -30.656$

```
---------------------------------------------
Iteration    1:   4 eigenvalues converged.
---------------------------------------------
Eval  1 (      -0.0487084,      -0.0140159) Res =        0.0000000
Eval  2 (      -0.0487084,       0.0140159) Res =        0.0000000
Eval  3 (     -46.2973364,     -64.2399268) Res =        0.0000000
Eval  4 (     -46.2973364,      64.2399268) Res =        0.0000000
 Iter.    1 R=   1.94500000D+03: ( -4.87083607D-02, -1.40159032D-02)
```

Only by imposing an eastward solid body rotation, and therefore considering a westward drifting frame of reference, did we manage to find this eigenvalue with positive growth rate. We conclude that the solution we seek is drifting westward. The imaginary parts of the corresponding eigenvalues are ± 331.0361451. As our wavenumber, m, is 4, this means that the solution indicated by this eigenvalue is drifting with a rate ± 82.759 relative to our new drifting frame of reference. This, in turn, means that the solution is drifting with rate either $c = -30.656 + 82.759 = 52.103$ or $c = -30.656 - 82.759 = -113.415$ relative to our fixed frame of reference.

$c = 52.103$ has to be considered unlikely as this is an eastward drifting solution! This would have been picked up by our applying a westward (negative) solid body rotation, c_s. We then conclude that the desired solution drifts with a rate $c = -113.415$ relative to our fixed frame of reference. This looks suspiciously like the result that [ZB87] give! Put this to the test by running the `sbrlinons1` with the input file `example_c.input` (which has `REYSBR = 113.415`).

The runs from the $P = 0.1$ and $P = 1.0$ problems return the following lines in the `.res` file:

```
6  24  40  1.00000000D+03  4.43140480D+03  7.477742D-06 -2.05393319D+02  1.71160000D+01
6  24  40  1.00000000D+03  4.43140479D+03  7.604961D-06 -1.31974439D-03 -1.71160000D+01
6  24  40  1.00000000D+03  1.32023578D+04  3.387941D-05 -6.86024577D+01  5.71700000D+00
6  24  40  1.00000000D+03  1.32023578D+04  3.387932D-05 -1.54228396D-03 -5.71
```

In both cases, the number in column 7 (the imaginary part of the eigenvalue) is far smaller for the negative `REYSBR` cases. This means that the solution is closer to being stationary in a westward drifting frame of reference than the eastward drifting frame of reference. We deduce that our drift-rates, c, for the solutions are therefore positive. This is in agreement with [ZB87].

Finally we note that no line of the `.res` file was written for run number 8. It simply ran out of iterations, and examining the `.log` file indicates that it would ultimately have converged had it been allowed to run for long enough. This is a pathological case for the iteration process. It obtains one limit from the wrong

mode (which has obviously had a greater growth rate at one Rayleigh number than the mode which is ultimately realised) and the iteration process therefore proceeds very slowly.

This section has highlighted some of the problems with iterating to critical Rayleigh numbers. The program `sbrlinonsd` is almost identical apart from that it changes the strength of the applied solid body rotation at every iteration, in order to try to find the frame of reference in which the solution is stationary.

References

[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.

[ZB87] K. Zhang and F. H. Busse. On the onset of convection in rotating spherical shells. *Geophys. Astrophys. Fluid Dyn.*, 39:119–147, 1987.