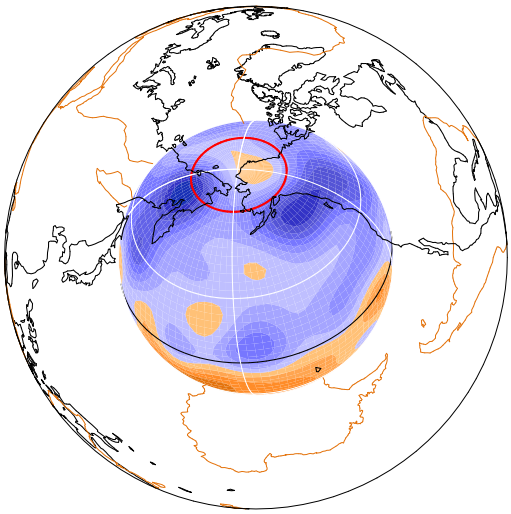
LEOPACK



djiepgrf

Dudley and **J**ames **I**nstability **E**igenvalue **P**roblem **G**rowth **R**ate **F**ind

Steven J. Gibbons, Oslo

Original document: November 21^{st}, 2001. Updated: October 30^{th}, 2022.

1 djiepgrf

Dudley and **J**ames **I**nstability **E**igenvalue **P**roblem **G**rowth **R**ate **F**ind

The majority of these notes are directly plagerised from [Sar94].

The equation describing the evolution of a magnetic field, $\boldsymbol{B}$, in a conducting fluid with velocity $\boldsymbol{u}$ is derived from the pre-Maxwell equations

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \ , \quad \nabla \times \boldsymbol{B} = \mu \boldsymbol{J} \quad \text{and} \quad \nabla . \boldsymbol{B} = 0 \ , \tag{1}$$

and Ohm's law

$$\boldsymbol{J} = \sigma(\boldsymbol{E} + \boldsymbol{u} \times \boldsymbol{B}), \tag{2}$$

where $\boldsymbol{E}$ is the electric field and σ the electrical conductivity. The pre-Maxwell forms are used since the displacement current, $\partial \boldsymbol{E}/\partial t$, will be negligible for the relatively slow variations appropriate for the Earth. Assuming the electrical conductivity to be a constant, taking the curl of Equation (2) and applying the relations of (1) together with the vector identity

$$\nabla \times (\nabla \times \boldsymbol{V}) = \nabla(\nabla . \boldsymbol{V}) - \nabla^2 \boldsymbol{V},$$

gives the induction equation

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + \frac{1}{\mu_0 \sigma} \nabla^2 \boldsymbol{B}. \tag{3}$$

The generalised form of the induction equation, as used by the programs, is

$$c_k \frac{\partial \boldsymbol{B}}{\partial t} = c_m \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + c_l \nabla^2 \boldsymbol{B}. \tag{4}$$

We can write Equation (3) in the form

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + \eta \nabla^2 \boldsymbol{B}, \tag{5}$$

where $\eta = 1/(\mu_0 \sigma)$ is the magnetic diffusivity. If L, U, L^2/η and η^2/L^3 are respectively scales for length, velocity, time and the magnetic field strength (we are employing the diffusive time-scale) then we can write Equation (5) in the non-dimensional form

$$\frac{\partial \boldsymbol{B}}{\partial t} = R_m \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + \nabla^2 \boldsymbol{B}. \tag{6}$$

The magnetic Reynolds number, $R_m = (UL/\eta) = \mu_0 UL\sigma$, gives the ratio of the magnitude of the advective term, $\nabla \times (\boldsymbol{u} \times \boldsymbol{B})$, to the diffusive term, $\nabla^2 \boldsymbol{B}$.

We give the magnetic field, $\boldsymbol{B}$, an exponential time-dependence such that

$$\frac{\partial \boldsymbol{B}}{\partial t} = \sigma \boldsymbol{B}. \tag{7}$$

The growth rate, $\sigma = \sigma_{\rm r} + i\sigma_{\rm i}$, is in general complex. Substituting (7) into the induction equation (6) gives an eigenvalue problem of the form

$$\boldsymbol{A}\boldsymbol{x} = \sigma \boldsymbol{x}. \tag{8}$$

The vector $\boldsymbol{x}$ is a solution vector containing both the poloidal and toroidal magnetic field radial functions. The matrix $\boldsymbol{A}$ is the discretisation of the operator, $\nabla \times (\boldsymbol{u} \times \cdot) + \nabla^2 \cdot$. The eigenvalue problem (8) is solved using the Implicitly Restarted Arnoldi Method, or IRAM, ([Arn51],[LSY98], [Sor92]). The eigenvalues we are interested in are those with the largest real parts as these will control the stability. We apply a real shift, $\lambda^{\rm s}$, to the system; subtracting $\lambda^{\rm s}\boldsymbol{x}$ from both sides of Equation (8) gives

$$\left(\boldsymbol{A} - \lambda^{\rm s}\boldsymbol{I}\right)\boldsymbol{x} = \left(\sigma - \lambda^{\rm s}\right)\boldsymbol{x}. \tag{9}$$

Rearranging to

$$\left(\boldsymbol{A} - \lambda^{\rm s}\boldsymbol{I}\right)^{-1}\boldsymbol{x} = \left(\sigma - \lambda^{\rm s}\right)^{-1}\boldsymbol{x} \tag{10}$$

gives us a new eigenproblem where the eigenvalue $(\sigma - \lambda^{\rm s})^{-1}$ will have a far higher magnitude than any others. ARPACK will locate this eigenvalue far more effectively than it would for the original formulation (8).

The program `djiepgrf` returns growth rate eigenvalues, σ, as a function of R_m for a given parametrisation of the Dudley James velocity, $\boldsymbol{u}_{\rm DJ}$, [DJ89]. $\boldsymbol{u}_{\rm DJ}$ is defined as

$$\boldsymbol{u}_{\rm DJ} = \nabla \times \nabla \times \left[\, {}^{P}_{\rm DJ}v(r,t,\theta,\phi)\ \boldsymbol{r} \ \right] \ + \ \nabla \times \left[\, {}^{T}_{\rm DJ}v(r,t,\theta,\phi)\ \boldsymbol{r} \ \right], \tag{11}$$

where the only non-zero radial functions for the velocity are

$${}^{T}_{\rm DJ}v_1^{0c}(r) = \sin(\pi r) \tag{12}$$

and

$${}^{P}_{\rm DJ}v_2^{0c}(r) = \varepsilon r \sin(\pi r). \tag{13}$$

The geometry is a sphere with outer boundary at $r = 1$: this means $r_{\rm i} = 0$ and $r_{\rm o} = 1$.

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

```
make djiepgrf
```

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

```
djiepgrf < inputfile
```

The inputs file must have the following format.

```
* input file for djiepgrf
*
example_aOUTPUT                    : ROOT
0.0    1.0     40.0   4   20      : RI, RO, DRSV, NEV, NCV
*---------------------------------------------------------------
*  NR  LH  SFT  SFL  SFM  IOF  ISP  NBNO EPS   LAMBDA RM
*---------------------------------------------------------------
   50  16   4    1    1    0    1    3   0.13  0.0    94.0
   50  16   4    1    1    0    1    3   0.13  0.0    95.0
   50  18   4    1    1    0    1    3   0.13  0.0    94.0
   50  18   4    1    1    0    1    3   0.13  0.0    95.0
  100  18   4    1    1    0    1    3   0.13  0.0    94.0
  100  18   4    1    1    0    1    3   0.13  0.0    95.0
```

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.
- `RI`. Inner boundary radius: essentially always set to zero.
- `RO`. Outer boundary radius: essentially always set to 1.0.
- `DRSV`. The real shift - this is $\lambda_{\mathrm{r}}^{\mathrm{s}}$ in Equation (10).
- `NEV`. The number of eigenvalues requested.
- `NCV`. The length of the Arnoldi factorisation. See [LSY98]) for details.

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

- `NR`. Number of radial grid nodes.
- `LH`. Highest spherical harmonic degree, l requested.
- `SFT`: Type for the seed field harmonic. This can be either `4` for a poloidal magnetic field of `5` for a toroidal magnetic field. `SFL`: Degree, l, for the seed field harmonic. `SFM`: Order, m, for the seed field harmonic. (This selects a $\cos m\phi$ harmonic but, by the selection rules for the velocity, the $\sin m\phi$ harmonic will also be chosen and hence this is no restriction.

- `IOF`. Output file flag. Options are:
  `IOF = 0`: → no output of eigenvectors.
  `IOF = 1`: → only output eigenfunctions corresponding to the eigenvalue with the largest real part.
  `IOF = 2`: → output all eigenfunctions.
  Note that for `djiepgrf`, options `IOF = 1` and `IOF = 2` only ever produce a single eigenvector.

- `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`

- `NBNO`. Number of bounding nodes for derivatives of velocity radial functions. Limited by the integer parameter `NBNOMX` in the source code. Whereas, for example, David Gubbins' code uses analytic expressions for the velocity radial functions and their derivatives, this code differentiates the functions numerically. The main reason for this approach was so that the subroutines used could handle general flows, e.g. steady flows locked by thermal boundary heating. However, it also has the advantage that (once thoroughly tested) any velocity may be applied without fear of making algebraic errors in the differentiation. The clear disadvantage is that it is not exact, although `NBNO` may be increased arbitrarily with little detriment to the program, other than the time required to compute these derivatives. Experiment to see what effect this has. `NBNO` must be atleast 3.

- `EPS`. Parameter ε in Equation (13).

- `LAMBDA`. Always set to zero.

- `RM`: Value of the magnetic Reynolds number.

1.1 Subprograms required for djiepgrf

SUBS subroutines

```
fopen.f djvhmf.f esnaas.f zcpaas.f svdjvf.f kdthsr.f
svfdcf.f ontppf.f gauwts.f schnla.f matop.f vcpcc.f
rv0mfa.f amlp.f  mfseps.f hmfwt.f xarrwt.f evecex.f
svfwt.f fclose.f fnamer.f cntric.f ldgnmf.f gfdcfd.f
shvect.f vfcp.f  vf2qst.f invmft.f vicexr.f innlca.f
amdlt.f amlica.f amta.f  amsdea.f vecop.f asvta.f
asvcpl.f cubeop.f dvecz.f fftrlv.f asvdr.f matind.f
bmrcop.f powtwo.f
```

SUBS double precision function

```
emmult.f pmm.f   pmm1.f  plm.f   dpmm.f  dpmm1.f
dplm.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 dcopy.f dgemm.f dger.f  dscal.f dswap.f
dtrsm.f dgemv.f dtbsv.f dtrmm.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

```
dgbtrf.f dgbtrs.f dgetrf.f dgetri.f dgbtf2.f dlaswp.f
xerbla.f dlahqr.f dgeqr2.f dlacpy.f dlaset.f dorm2r.f
dtrevc.f dtrsen.f dgetf2.f dtrtri.f dlabad.f dlanv2.f
dlarfg.f dlarf.f dlaln2.f dlacon.f dtrexc.f dtrsyl.f
dtrti2.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, NBNOMX, NRUNM, NCFMOM, NVMAX, IVELMX, NDCS0,
3        MAXNVI, NCVM
 PARAMETER ( NRMAX = 200, LHMAX = 50, MAXNVI = 30000, NRUNM = 50,
1            LHLH2M = LHMAX*(LHMAX+2), NBN = 2, NCVM = 25,
2            NHMAX = LHMAX*2, NPHMAX = 64, NTHMAX = LHMAX + 2,
3            KLMAX = (NBN+1)*NHMAX-1, NBNDMX = 3*KLMAX+1 )
 PARAMETER ( NBNOMX = 5, NDCS = LHMAX + 1, NDRVM = 2,
1            NCFM = 2*NBN + 1, NCFMOM = 2*NBNOMX + 1,
2            ISVMAX = NRMAX*NHMAX, NVMAX = 2,
3            IVELMX = NVMAX*NRMAX, NDCS0 = 1,
4            NPMAX = (LHMAX+1)*(LHMAX+2)/2 )
```

If the values are insufficient, then change them and recompile. (Note that `NDRVM` and `NDCS0` 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.
- `MAXNVI` is the maximum number of spherical harmonic interactions. These are pre-calculated and stored in an array. There is no real way of knowing how many of these will be required without simply working them out. This number is therefore rather trial and error based: it clearly increases with `LH`. (This is probably quite an inefficient way of calculating non-linear interactions. Anybody is welcome to think up new ways. However, since the majority of time in this program is at the linear algebra stage, this aspect of the code never seemed worth examining.)
- `NRUNM` is the maximum number of independent runs permitted.
- `NBN` is the number of bounding nodes on either side of the finite difference stencil. `NBN = 2` is recommended value. Since the matrix describing the advective term couples distinct spherical harmonic radial functions (i.e. we are using the **IFORMF = 3** option, the bandwidth of the matrix soon becomes very large as the number of radial functions increases. `NBN = 3` therefore is a far higher computational cost than `NBN = 2`. (`NBN = 3` is necessary for the vorticity equation, for example in `linons1`, because of the fourth derivatives for the poloidal velocity. However, for the induction equation, `NBN = 2` gives fourth order accuracy.)

- `NPHMAX` is the maximum permitted number of grid nodes in ϕ for the Fast Fourier Transforms.
- `NBNOMX` is the upper limit for `NBNO`.
- `NVMAX` is the maximum number of velocity spherical harmonic radial functions. `NVMAX = 2` is quite adequate for the Dudley and James flow (which has exactly 2 spherical harmonics). This will need to be changed if this program is to be modified to deal with other flows.

1.3 Outputs from DJIEPGRF

If the filename stem "root" was specified in the input file, the file `root.log` will be generated, along with any output solution vectors.

For each iteration of each run demanded from the input file, the following information is given in the file `root.log`:

```
ri:    0.0000000D+00 ro:    1.0000000D+00 nr:   100 isp:  1 nbn0:  3
lh:  18 sft:   4 sfl:   1 sfm:   1
eps:    1.3000000D-01 La:    0.0000000D+00 rm:    9.4000000D+01
   4 eigenvalues converged.
eval:   1 (  -6.0806673D-02,  -1.8137069D+01) res:    7.9819322D-14
eval:   2 (  -6.0806673D-02,   1.8137069D+01) res:    7.9819322D-14
eval:   3 (  -5.8972648D+01,  -4.6746802D+01) res:    5.7720976D-10
eval:   4 (  -5.8972648D+01,   4.6746802D+01) res:    5.7720976D-10
ri:    0.0000000D+00 ro:    1.0000000D+00 nr:   100 isp:  1 nbn0:  3
lh:  18 sft:   4 sfl:   1 sfm:   1
eps:    1.3000000D-01 La:    0.0000000D+00 rm:    9.5000000D+01
   4 eigenvalues converged.
eval:   1 (   5.2330877D-02,  -1.8253576D+01) res:    6.2493177D-14
eval:   2 (   5.2330877D-02,   1.8253576D+01) res:    6.2493177D-14
eval:   3 (  -5.9118135D+01,  -4.7150592D+01) res:    1.9780605D-10
eval:   4 (  -5.9118135D+01,   4.7150592D+01) res:    1.9780605D-10
```

This simply returns the eigenvalues found along with their direct residuals and the various settings.

1.4 Sample runs of djiepgrf

The directory

`$LEOPACK_DIR/SAMPLERUNS/DJIEPGRF`

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

Table (5.1) on page 161 of [Sar94] gives critical magnetic Reynolds numbers for the Dudley and James flow with $\varepsilon = 0.13$. At the critical R_m value, the growth rate has zero imaginary part. Therefore, below the critical value, we should get only eigenvalues with negative real parts and, above the critical value, we should get an eigenvalue with a positive real part.

According to [Sar94], the critical R_m value appears to converge to 94.923, with an oscillation frequency of 18.210. The input file `example_a.input` includes R_m values above and below this value for different numerical resolutions.

References

[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.

[DJ89] M. L. Dudley and R. W. James. Time dependent dynamos with stationary flows. *Proc. R. Soc. Lond. A*, 425:407–429, 1989.

[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.

[Sar94] G. R. Sarson. *Kinematic Dynamo Calculations for Geomagnetism*. PhD thesis, University of Leeds, 1994.

[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.