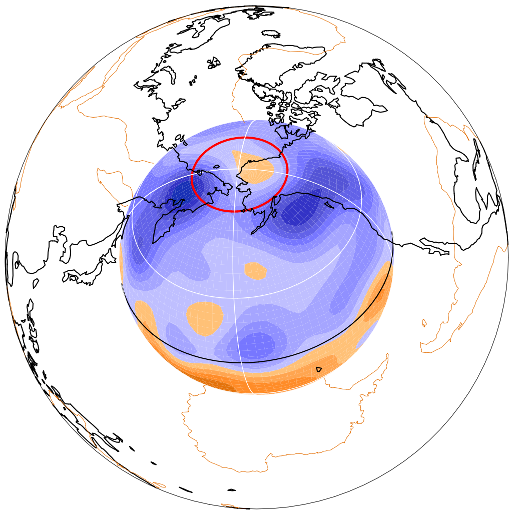
LEOPACK



krddmcmrnif

Kumar **R**oberts **D**ynamo (**D**,**M**) **C**ritical **M**agnetic
Reynolds **N**umber **I**terative **F**ind

Steven J. Gibbons, Oslo

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

1 krddmcmrnif

Kumar **R**oberts **D**ynamo (**D**,**M**) **C**ritical **M**agnetic **R**eynolds **N**umber **I**terative **F**ind

This performs essentially the same task as `krcmrnif`: finds a critical magnetic Reynolds number for the Kumar Roberts velocity based flow.

We concentrate on a class of flows developed from the velocity of Kumar Roberts, [KR75]. The flow is defined,

$$\boldsymbol{u} = \epsilon_0 \boldsymbol{t}_1^0 + \epsilon_1 \boldsymbol{s}_2^0 + \epsilon_2 \boldsymbol{s}_2^{2c} + \epsilon_3 \boldsymbol{s}_2^{2s}, \tag{1}$$

where $\boldsymbol{t}_l^m$, $\boldsymbol{s}_l^m$ are toroidal and poloidal vector spherical harmonics:

$$\boldsymbol{t}_l^{m\{{}^{c}_{s}\}} = \nabla \times [t_l^m(r) P_l^m(\cos\theta) \left\{ \begin{array}{c} \cos \\ \sin \end{array} \right\} (m\phi) \boldsymbol{e}_r], \tag{2}$$

$$\boldsymbol{s}_l^{m\{{}^{c}_{s}\}} = \nabla \times \nabla \times [s_l^m(r) P_l^m(\cos\theta) \left\{ \begin{array}{c} \cos \\ \sin \end{array} \right\} (m\phi) \boldsymbol{e}_r], \tag{3}$$

(r, θ, ϕ) are spherical coordinates, P_l^m are Schmidt-normalised associated Legendre functions, and $\boldsymbol{e}_r$ is the unit radial vector. The scalar functions were chosen to give a $\boldsymbol{u}$ that is differentiable at the origin, and to be zero with zero stress on the outer boundary:

$$\begin{aligned} t_1^0(r) &= r^2(1-r^2), \\ s_2^0(r) &= r^6(1-r^2)^3, \\ s_2^{2s}(r) &= r^4(1-r^2)^2 \cos(p\pi r), \\ s_2^{2c}(r) &= r^4(1-r^2)^2 \sin(p\pi r). \end{aligned} \tag{4}$$

Instead of using the coefficients ϵ_0, ϵ_1, ϵ_2 and ϵ_3, as defined by [KR75], [GBGL00] set $\epsilon_2 = \epsilon_3$ and $p = 3$ and defined the flow in terms of two parameters, D and M. These parameters are such that the fraction of kinetic energy in the meridian circulation ($\boldsymbol{s}_2^0$) is given by $|M|$, where

$$M = \mathrm{sgn}(\epsilon_1)\beta\epsilon_1^2, \tag{5}$$

and the fraction of energy in the differential rotation ($\boldsymbol{t}_1^0$) is given by $|D|$, where

$$D = \mathrm{sgn}(\epsilon_0)\alpha\epsilon_0^2. \tag{6}$$

Here, the scalars α, β, γ and δ are integrals of the radial functions in (4) such that

$$\int \boldsymbol{u}^2 dV = \alpha\epsilon_0^2 + \beta\epsilon_1^2 + \gamma\epsilon_2^2 + \delta\epsilon_3^2 = 1 \tag{7}$$

The fraction of kinetic energy which is in the non-axisymmetric convective rolls is given by

$$C = 1 - |M| - |D|. \tag{8}$$

The only difference is that whereas `krcmrnif` uses the standard ϵ scalings of [KR75], `krddmcmrnif` uses the (D, M) parametrisation of the flow as used by [GBGL00].

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

```
make krddmcmrnif
```

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

```
krddmcmrnif < inputfile
```

The inputs file must have the following format.

```
* input file for krddmcmrnif
*
example_aOUTPUT                             : ROOT
0.0    1.0     10.0   4   20   7   0.001   : RI, RO, DRSV, NEV, NCV, NOITM, DTOL
*----------------------------------------------------------------------
* NR LH ITRI ISF IOF ISP NBNO     DD          DM      RM1     RM2
*----------------------------------------------------------------------
  50 10  0    1   0   1   4    0.98349148  0.00016327  51.466667  51.866667
  50 10  0    1   0   1   4    0.98349148  0.00016327  51.45      51.86
  50 12  0    1   0   1   4    0.98349148  0.00016327  51.45      51.86
*----------------------------------------------------------------------
```

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

- `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 $\lambda_{\mathrm{r}}^{\mathrm{s}}$
- `NEV`. The number of eigenvalues requested.
- `NCV`. The length of the Arnoldi factorisation. See [LSY98]) for details.

- `NOITM`: The maximum number of iterations (values for R_m tried) permitted in order to find R_m^{c}.
- `DTOL`: Stopping criterion. $|\sigma_{\mathrm{r}}|$ must be less than `DTOL` for convergence to have occured.

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.
- `ITRI`. Determines whether or not triangular truncation is used. The normal truncation of spherical harmonics is to include only terms with l up to including `LH`. This option is selected by `ITRI = 0`. If `ITRI = 1` then only radial functions with $(l + m)$ up to including `LH` are selected.
- `ISF`. Symmetry selection flag. This essential selects the seed field. Due to the symmetry of the flow there are four distinct symmetries of magnetic field which decouple. Each is determined by a poloidal magnetic seed field with a single spherical harmonic with degree l_{Seed} and order m_{Seed}. The options are:
  `ISF = 1`: $\rightarrow$ axial dipole. $l_{\mathrm{Seed}} = 1,\ m_{\mathrm{Seed}} = 0$.
  `ISF = 2`: $\rightarrow$ equatorial quadrupole. $l_{\mathrm{Seed}} = 2,\ m_{\mathrm{Seed}} = 0$.
  `ISF = 3`: $\rightarrow$ axial dipole. $l_{\mathrm{Seed}} = 1,\ m_{\mathrm{Seed}} = 1$.
  `ISF = 4`: $\rightarrow$ equatorial quadrupole. $l_{\mathrm{Seed}} = 2,\ m_{\mathrm{Seed}} = 1$.
  See ([Sar94]) and ([GZ93]) for details on symmetry.
- `IOF`. Output file flag. Options are:
  `IOF = 0`: $\rightarrow$ no output of eigenvectors.
  `IOF = 1`: $\rightarrow$ only output eigenfunctions corresponding to the eigenvalue with the largest real part.
  `IOF = 2`: $\rightarrow$ output all eigenfunctions.
  Note that for `krcmrnif`, 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.

- `DD`: Parameter D.
- `DM`: Parameter M.
- `RM1`: Sub-critical value of R_m.
- `RM2`: Super-critical value of R_m.

1.1 Subprograms required for krddmcmrnif

SUBS subroutines

```
fopen.f krvhmf.f esnaas.f zcpaas.f svkrvf.f kdthsr.f
svfdcf.f ontppf.f gauwts.f schnla.f dm2e.f  kdcmrn.f
hmfwt.f xarrwt.f svfwt.f fclose.f fnamer.f radvlf.f
cntric.f ldgnmf.f gfdcfd.f vcpcc.f matop.f rv0mfa.f
amlp.f  mfseps.f evecex.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 dgemm.f dtrsm.f dgemv.f dswap.f dcopy.f
dger.f  dscal.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

```
dgetrf.f dgetri.f dgbtrf.f dgbtrs.f dgetf2.f dlaswp.f
xerbla.f dtrtri.f dgbtf2.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, NBNOMX, NRUNM, NCFMOM, NVMAX, IVELMX, NDCS0,
3        MAXNVI, NCVM
 PARAMETER ( NRMAX = 100, LHMAX = 20, MAXNVI = 30000, NRUNM = 50,
1            LHLH2M = LHMAX*(LHMAX+2), NBN = 2, NCVM = 25,
2            NHMAX = LHLH2M/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 = 4,
3              IVELMX = NVMAX*NRMAX, NDCSO = 1,
4              NPMAX = (LHMAX+1)*(LHMAX+2)/2 )
```

If the values are insufficient, then change them and recompile. (Note that `NDRVM` and `NDCSO` 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 = 4` is quite adequate for the Kumar and Roberts flow (which has exactly 4 spherical harmonics). This will need to be changed if this program is to be modified to deal with other flows.

1.3 Outputs from KRDDMCMRNIF

If the filename stem "root" was specified in the input file, the files `root.log` and `root.res` 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`:

```
 Number of vector interactions =   2016
----------------------------------------------
Iteration    1:   4 eigenvalues converged.
Rm value =           51.4666670
----------------------------------------------
Eval  1 (      -0.0355436,       0.0000000) Res =        0.0000000
Eval  2 (     -35.6623743,       0.0000000) Res =        0.0000000
Eval  3 (     -37.6145508,     -42.3467569) Res =        0.0000000
Eval  4 (     -37.6145508,      42.3467569) Res =        0.0000000
----------------------------------------------
Iteration    2:   4 eigenvalues converged.
Rm value =           51.8666670
----------------------------------------------
Eval  1 (       0.0230214,       0.0000000) Res =        0.0000000
Eval  2 (     -35.6658094,       0.0000000) Res =        0.0000000
Eval  3 (     -37.6436482,     -42.6605030) Res =        0.0000000
Eval  4 (     -37.6436482,      42.6605030) Res =        0.0000000
----------------------------------------------
Iteration    3:   4 eigenvalues converged.
Rm value =           51.7094304
----------------------------------------------
Eval  1 (       0.0000016,       0.0000000) Res =        0.0000000
Eval  2 (     -35.6644089,       0.0000000) Res =        0.0000000
Eval  3 (     -37.6321748,     -42.5371073) Res =        0.0000000
Eval  4 (     -37.6321748,      42.5371073) Res =        0.0000000
RM=    5.1709430D+01 Real=    1.5642527D-06 Imag=    0.0000000D+00
ri:    0.0000000D+00 ro:    1.0000000D+00 nr:    50 isp:  1 nbn0:  4
lh:  10 itri:  0 Field sym: Axial_dipole
e0:    7.4999974D+01 e1:    2.2500067D+00 e2:    3.0000000D+00
e3:    3.0000000D+00 pp:    3.0000000D+00 rmc:    5.1709430D+01
osc:    0.0000000D+00
```

This simply returns a summary of eigenvalues, their direct residuals and parameter specifications (in terms of the ϵ parameters, rather than D and M).

The file `root.res` produces one line of summary for each line of the input file:

```
 D=   9.8349148D-01 M=   1.6327000D-04 R=   5.1706213D+01 O=   0.0000000D+00
```

1.4 Sample runs of krddmcmrnif

The directory

`$LEOPACK_DIR/SAMPLERUNS/KRDDMCMRNIF`

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

`example_a.input` basically reproduces the Kumar Roberts flow results obtained by the programs `krssgeps` and `krcmrnif` with notes explaining the different scaling.

References

[GBGL00] D. Gubbins, C. N. Barber, S. Gibbons, and J. J. Love. Kinematic dynamo action in a sphere: I effects of differential rotation and meridional circulation on solutions with axial dipole symmetry. *Proc. R. Soc. Lond. A*, 456:1333–1353, 2000.

[GZ93] D. Gubbins and K. Zhang. Symmetry properties of the dynamo equations for paleomagnetism and geomagnetism. *Phys. Earth Planet. Inter.*, 75:225–241, 1993.

[KR75] S. Kumar and P. H. Roberts. A three-dimensional kinematic dynamo. *Proc. Roy. Soc. Lond. A*, 344:235–258, 1975.

[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.