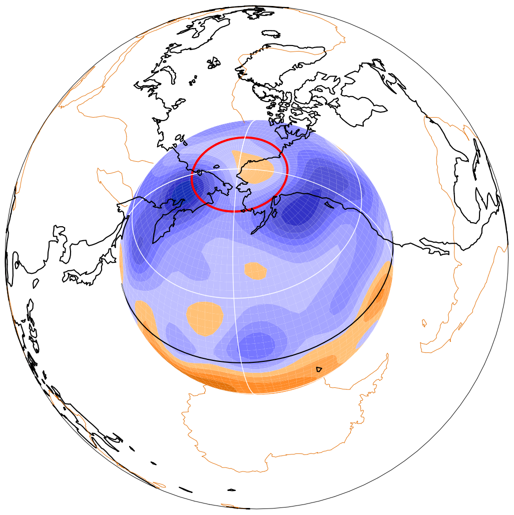
# LEOPACK



# krssgeps

**K**umar **R**oberts velocity **S**tationary **S**olution **G**eneralised **E**igenvalue **P**roblem **S**olve.

Steven J. Gibbons, Oslo

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

## 1 krssgeps

**K**umar **R**oberts velocity **S**tationary **S**olution **G**eneralised **E**igenvalue **P**roblem **S**olve.

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 $\sigma$ 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 $L$, $U$, $L^2/\eta$ and $\eta^2/L^3$ 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, $\nabla^2 \boldsymbol{B}$.

If we know that a steady flow, $\boldsymbol{u}$, produces a steady magnetic field then we can set $\partial \boldsymbol{B}/\partial t$ to zero *a priori* and get the relationship

$$\nabla \times (\boldsymbol{u} \times \boldsymbol{B}) = \lambda(-\nabla^2 \boldsymbol{B}). \tag{7}$$

$\lambda$ is an eigenvalue of the discretised problem such that the critical magnetic Reynolds number, $R_m^{\mathrm{c}}$, can be found from

$$R_m^{\mathrm{c}} = \frac{1}{\lambda}. \tag{8}$$

We may write the discretised form of Equation (7) as the linear algebraic equation

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{M}\boldsymbol{x}. \tag{9}$$

The vector $\boldsymbol{x}$ is a solution vector of the form containing both the poloidal and toroidal magnetic field radial functions. The matrix $\boldsymbol{A}$ is the discretisation of the advection operator, $\nabla \times (\boldsymbol{u} \times \cdot)$, and the matrix $\boldsymbol{M}$ is the discretisation of the diffusion operator, $-\nabla^2$. The generalised eigenvalue problem (7) is solved using the Implicitly Restarted Arnoldi Method, or IRAM, ([Arn51],[LSY98], [Sor92]). We want only a single eigenvalue and, possibly, the corresponding eigenvector. If we have a guess, $R_m^{\mathrm{g}}$, at the critical magnetic Reynolds number then our expected eigenvalue, $\lambda$, is

$$\lambda^{\mathrm{s}} = \frac{1}{R_m^{\mathrm{g}}}. \tag{10}$$

Subtracting $\lambda^{\mathrm{s}}\boldsymbol{M}\boldsymbol{x}$ from both sides of Equation (9) gives

$$\left(\boldsymbol{A} - \lambda^{\mathrm{s}}\boldsymbol{M}\right)\boldsymbol{x} = \left(\lambda - \lambda^{\mathrm{s}}\right)\boldsymbol{M}\boldsymbol{x}. \tag{11}$$

Rearranging to

$$\left(\boldsymbol{A} - \lambda^{\mathrm{s}}\boldsymbol{M}\right)^{-1}\boldsymbol{M}\boldsymbol{x} = \left(\lambda - \lambda^{\mathrm{s}}\right)^{-1}\boldsymbol{x} \tag{12}$$

gives us a new eigenproblem where the eigenvalue $(\lambda - \lambda^{\mathrm{s}})^{-1}$ will have a far higher magnitude than any others. ARPACK will locate this eigenvalue far more effectively than it would for the original formulation (7).

The program `krssgeps` finds such critical magnetic Reynolds number for the specific case of the Kumar Roberts velocity ([KR75]). This flow, $\boldsymbol{u}_{\mathrm{KR}}$, is defined as

$$\boldsymbol{u}_{\mathrm{KR}} = \nabla \times \nabla \times \left[\, {}^{P}_{\mathrm{KR}}v(r,t,\theta,\phi)\; \boldsymbol{r} \,\right] \; + \; \nabla \times \left[\, {}^{T}_{\mathrm{KR}}v(r,t,\theta,\phi)\; \boldsymbol{r} \,\right], \tag{13}$$

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

$${}^{T}_{\mathrm{KR}}v_1^{0c}(r) = \varepsilon_0 \left[ r(1-r^2) + \Lambda r \right], \tag{14}$$

$$ {}^{P}_{\mathrm{KR}}v_2^{0c}(r) = \varepsilon_1 r^5 (1 - r^2)^3, \tag{15} $$

$$ {}^{P}_{\mathrm{KR}}v_2^{2s}(r) = \varepsilon_2 r^3 (1 - r^2)^2 \cos(pr) \tag{16} $$

and

$$ {}^{P}_{\mathrm{KR}}v_2^{2c}(r) = \varepsilon_3 r^3 (1 - r^2)^2 \sin(pr). \tag{17} $$

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

The relative strengths of different components of the velocity are given by the parameters $\varepsilon_0$, $\varepsilon_1$, $\varepsilon_2$ and $\varepsilon_3$.

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

```
make sakrssgeps
```

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

```
krssgeps < inputfile
```

The inputs file must have the following format.

```
* input file for krssgeps
filename_stem                        : ROOT
0.0    1.0     10                    : RI, RO, NCV
*------------------------------------------------------------
*  NR  LH  ITRI  ISF  IOF  ISP  NBNO    E0   E1   E2   E3  PPAR  RMGUESS
*------------------------------------------------------------
  40    8   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  40   10   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  40   12   0     1    1    1    4    1.00  0.03  0.04  0.04  3   3840.0
  40   14   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  60    8   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  60   10   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  60   12   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  60   14   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
```

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 `krssgeps` with the above input file will create the files `filename_stem.log`, `filename_stem.run003.ints`, `filename_stem.run003.xarr` and `filename_stem.run003.e001.vecs`. Note that the `e001` part of the filename stem for the eigensolution vector file results from it being the first eigenvector found - since only one is asked for.

- `RI`. Inner boundary radius: essentially always set to zero.
- `RO`. Outer boundary radius: essentially always set to 1.0.
- `NCV`. Length of the Arnoldi factorisation. Must be greater than 2. See ([LSY98]) for details.

All of the 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 (equations 14 to 17) 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 $l_{\rm Seed}$ and order $m_{\rm Seed}$. The options are:
  `ISF = 1`: $\rightarrow$ axial dipole. $l_{\rm Seed} = 1,\ m_{\rm Seed} = 0$.
  `ISF = 2`: $\rightarrow$ equatorial quadrupole. $l_{\rm Seed} = 2,\ m_{\rm Seed} = 0$.
  `ISF = 3`: $\rightarrow$ axial dipole. $l_{\rm Seed} = 1,\ m_{\rm Seed} = 1$.
  `ISF = 4`: $\rightarrow$ equatorial quadrupole. $l_{\rm Seed} = 2,\ m_{\rm Seed} = 1$.
  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 **`krssgeps`**, 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.

- `E0`. Parameter $\varepsilon_0$ in Equation (14).
- `E1`. Parameter $\varepsilon_1$ in Equation (15).
- `E2`. Parameter $\varepsilon_2$ in Equation (16).
- `E3`. Parameter $\varepsilon_3$ in Equation (17).
- `PPAR`. $p/\pi$ where $p$ is defined in Equations (16) and (17). `PPAR = 3.0` $\rightarrow$ $p = 3\pi$.
- `RMGUESS`. A suggested value for the critical magnetic Reynolds number, $R_m$. This just applies the appropriate shift, $\lambda^{\rm s}$. It is better to underestimate the $R_m^{\rm c}$ value than overestimate it - if it is too high, you may pick up zero growth rate $R_m$ for higher modes.

## 1.1 Subprograms required for krssgeps

### SUBS subroutines

```
fopen.f krvhmf.f esnaas.f zcpaas.f svkrvf.f kdthsr.f
svfdcf.f ontppf.f gauwts.f schnla.f matop.f vcpcc.f
rv0mfa.f mfgeps.f hmfwt.f xarrwt.f evecex.f svfwt.f
fclose.f fnamer.f radvlf.f cntric.f ldgnmf.f gfdcfd.f
shvect.f vfcp.f  vf2qst.f invmft.f vicexr.f innlca.f
amlp.f  amsdea.f vecop.f asvlp.f asvcpl.f cubeop.f
dvecz.f fftrlv.f asvdr.f matind.f amdlt.f amlica.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 dl.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 = 100, LHMAX = 14, MAXNVI = 10000, NRUNM = 50,
1            LHLH2M = LHMAX*(LHMAX+2), NBN = 2, NCVM = 25,
2            NHMAX = LHLH2M/2, NPHMAX = 32, 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, 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.) Tables (1) and (2) show the difference between results with `NBN = 2` and `NBN = 3`.

- `NPHMAX` is the maximum permitted number of grid nodes in $\phi$ 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. (See section 1.3.)

## 1.3 Adapting krssgeps to other flows

`krssgeps` will only calculate solutions for the Kumar Roberts velocity definition. However, the core of the code will easily cope with an arbitrarily defined flow, provided that it is defined in the standard way: stored in the array `VEC0`, with the integer arrays `IN0`, `MT0`, `ML0`, `MM0` and `MP0` respectively holding the information described by the arrays `INARR`, `MHT`, `MHL`, `MHM` and `MHP`.

## 1.4 Outputs from KRSSGEPS

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 line of the input file, the following information is given in the file `root.log`:

```
ri:    0.0000000D+00 ro:    1.0000000D+00 nr:    40 isp:  1 nbn0:  4
lh:   8 itri:  0 Field sym: Axial_dipole
e0:    1.0000000D+00 e1:    3.0000000D-02 e2:    4.0000000D-02
e3:    4.0000000D-02 pp:    3.0000000D+00 rm:    3.8400000D+03
   1 eigenvalues converged.
eval:   1 (   2.5905233D-04,   0.0000000D+00) res:    6.7359754D-14
Rmc =    3.8602239D+03
```

As you can see, most of it is simply echoing the information specified in the appropriate part of the input file. It also gives the eigenvalue, along with the direct residual and the critical $R_m$ value.

## 1.5 Sample runs of krssgeps

The directory

`$LEOPACK_DIR/SAMPLERUNS/KRSSGEPS`

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.

### 1.5.1 Example a

We wish to reproduce the results for the original Kumar Roberts dynamo. This means that the parameters in equations (14) through to (17) are defined by

$$\begin{aligned}\varepsilon_0 &= 1.00\\ \varepsilon_1 &= 0.03\\ \varepsilon_2 &= 0.04\\ \varepsilon_3 &= 0.04.\end{aligned} \tag{18}$$

The input file

```
* input file for krssgeps
example_aOUTPUT                  : ROOT
0.0    1.0     10                : RI, RO, NCV
*-------------------------------------------------------------
*  NR  LH  ITRI  ISF  IOF  ISP  NBNO    E0   E1   E2   E3  PPAR  RMGUESS
*-------------------------------------------------------------
  50    8   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  50   10   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
  50   12   0     1    1    1    4    1.00  0.03  0.04  0.04  3   3840.0
  50   14   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 100    8   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 100   10   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 100   12   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 100   14   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 150    8   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 150   10   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 150   12   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
 150   14   0     1    0    1    4    1.00  0.03  0.04  0.04  3   3840.0
```

seeks to reproduce the results as displayed in Table (3.3), page 101 of ([Sar94]). Typing

```
grep Rmc example_aOUTPUT.log
```

gives the following output

| $L$ / $N$ | 50 | | 100 | | 150 | |
|---|---|---|---|---|---|---|
| 8 | 3877.97 | (3833.28) | 3897.01 | (3882.74) | 3899.64 | (3892.76) |
| 10 | 3878.21 | (3832.86) | 3897.15 | (3882.68) | 3899.76 | (3892.80) |
| 12 | 3878.22 | (3832.81) | 3897.17 | (3882.69) | 3899.79 | (3892.82) |
| 14 | 3878.21 | (3832.80) | 3897.16 | (3882.67) | 3899.77 | (3892.81) |

Table 1: $R_m^{\mathrm{c}}$ values with varying resolutions $L$ and $N$ for the axial dipole solution of the Kumar Roberts velocity. Values are given by the code krssgeps with NBN = 2 and those in parentheses from ([Sar94]). Using Richardson extrapolation, Sarson gives the result, converged to 6 significant figures as 3901.11.

```
Rmc =    3.8779657D+03
Rmc =    3.8782076D+03
Rmc =    3.8782242D+03
Rmc =    3.8782087D+03
Rmc =    3.8970094D+03
Rmc =    3.8971467D+03
Rmc =    3.8971724D+03
Rmc =    3.8971560D+03
Rmc =    3.8996365D+03
Rmc =    3.8997623D+03
Rmc =    3.8997897D+03
Rmc =    3.8997731D+03
```

which, by comparing with the lines in our input file, allows us to tabulate the results in the same format as Graeme Sarson (see Table 1).

### 1.5.2 Example b

We temporarily edit the source code to change the default `NBN = 2` to the more accurate but more expensive `NBN = 3`. We recomplie the code and repeat using the file `example_b.input` what are essentially exactly the same runs as demanded by `example_a.input`. We see from Figure (2) that the results are somewhat closer to the Richardson extrapolated figure quoted from ([Sar94]).

| $L$ / $N$ | 50 | | 100 | | 150 | |
|---|---|---|---|---|---|---|
| 8 | 3893.61 | (3833.28) | 3899.45 | (3882.74) | 3900.52 | (3892.76) |
| 10 | 3893.72 | (3832.86) | 3899.58 | (3882.68) | 3900.54 | (3892.80) |
| 12 | 3893.74 | (3832.81) | 3899.60 | (3882.69) | 3900.67 | (3892.82) |
| 14 | 3893.72 | (3832.80) | 3899.59 | (3882.67) | 3900.66 | (3892.81) |

Table 2: $R_m^{\mathrm{c}}$ values with varying resolutions $L$ and $N$ for the axial dipole solution of the Kumar Roberts velocity. Values are given by the code krssgeps with NBN $= 3$ and those in parentheses from ([Sar94]). Using Richardson extrapolation, Sarson gives the result, converged to 6 significant figures as 3901.11.

## 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.

[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.

[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.