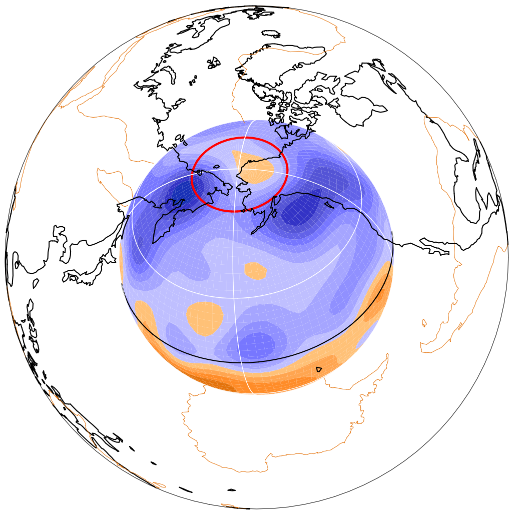
LEOPACK



o2ibtctsc2

Inhomogeneous **B**oundary **T**hermal **C**onvection **T**ime-**S**tep
Code **2**

Steven J. Gibbons, Oslo

Original document: November 21st, 2001. Updated: October 23rd, 2022.

1 o2ibtctsc2

Inhomogeneous **B**oundary **T**hermal **C**onvection **T**ime-**S**tep **C**ode **2**

This program time-steps a non-magnetic convection solution in a spherical shell subject to an imposed temperature function. This function is normally designed to impose a laterally varying temperature or heat-flux at the outer surface.

The equations to be integrated are

$$c_a \frac{\partial \Theta}{\partial t} = c_d \nabla^2 (\Theta + \varepsilon T_{\rm a}) + b_1 u_r r + b_2 \frac{u_r}{r^2} - c_c \boldsymbol{u}.\nabla(\Theta + \varepsilon T_{\rm a}) \tag{1}$$

and

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

Θ is the part of the temperature distribution which satisfies homogeneous boundary conditions, $\boldsymbol{u}$ is the velocity and $\boldsymbol{\omega}$ is the vorticity (with $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$).

The temperature function, $T_{\rm a}$, is present prior to execution in the standard form of a `.ints`, `.vecs` and `.xarr` file. The standard way of generating these files is using the program `itfvf` - although any file containing only temperature spherical harmonic radial functions is valid. The only condition is that the inner and outer boundaries on the temperature function coincide with those for the homogeneous boundary solution. Neither the number of radial grid nodes, nor their pattern of spacing, needs to match exactly as the temperature heterogeneity function is interpolated onto the mesh used by the main solution.

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

```
make o2ibtctsc2
```

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

```
o2ibtctsc2 < inputfile
```

The inputs file must have the following format.

```
* input file for o2ibtctsc2
*
example_aOUTPUT                           : Filename stem
example_a.ints                            : indices file for homog. soln.
example_a.vecs                            : vector file for homog. soln.
example_a.xarr                            : radial file for homog. soln.
example_aHEATFLUX.ints                    : boundary temp. harms
example_aHEATFLUX.vecs                    : boundary temp. vecs
example_aHEATFLUX.xarr                    : boundary temp. xarrs
  0.30                                    : SCAL
   -2            : NTHP (-1 --> LH+2, -2 --> 3*LH/2 )
```

```
 1.0   1.0   0.0           1.0   1.0       : CA  CB1  CB2   CC   CD
 1.66666666   1.66666666 200.0 1200.0  1.0 : CE   CF   CG   CH   CI
 0.5     0.0005    0.0000                  : CFAC    DELTAT   STIME
* ioutf is set to 0 for no output, 1 for file and 6 for screen
* nts is the total number of time-steps required
* ntsbb is the number of time-steps between backups
* ntsbs is the number of time-steps between snapshots
   0     10   1000  1000                : IOUTF  NTS NTSBB NTSBS
   6     10.0                           : ITMX   DTOL
   1     10    10                       : NTSBSE NTSBLE NTSBME
```

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 arguments are as follows

- `Filename stem`: First characters in output files to be generated by current run. Running `o2ibtctsc2` with the above input file will create the files `example_aOUTPUT.log`, `example_aOUTPUT.nrg`, along with all backup and snapshot outputs of the solution.

- `indices file for homog. soln.`: name of already existing indices file describing initial solution.

- `vector file for homog. soln.`: name of already existing vector file describing initial solution. Must contain the same number of radial functions as indicated in the `.ints` file.

- `radial file for homog. soln.`: name of already existing radial spacings file describing initial solution. Must contain the same number of radial grid nodes as indicated in the `.vecs` file.

- `boundary temp. harms`: name of already existing indices file describing initial solution.

- `boundary temp. vecs`: name of already existing vector file describing initial solution. Must contain the same number of radial functions as indicated in the `.ints` file.

- `boundary temp. xarrs`: name of already existing radial spacings file describing initial solution. Must contain the same number of radial grid nodes as indicated in the `.vecs` file. The first and last grid node radii must correspond to the first and last grid node radii of the solution vector: otherwise any spacing will do.

- `SCAL`: Multiplier for the inhomogeneous temperature function.

- `NTHP`: number of θ points for Gaussian quadrature in spherical transforms. Must be greater than `LH` - the highest spherical harmonic degree in the expansions. Is limited by the parameter `NTHMAX`. There are two "special" values of `NTHP` which may be entered and are infact recommended. `NTHP = -1` sets $N_\theta = L_{\max} + 2$ and `NTHP = -2` sets $N_\theta = 3L_{\max}/2$. This higher number of θ points eliminates aliassing effects in the transforms.
- `CA`: Scaling parameter c_a in Equation (1).
- `CB1`: Scaling parameter b_1 in Equation (1).
- `CB2`: Scaling parameter b_2 in Equation (1).
- `CC`: Scaling parameter c_c in Equation (1).
- `CD`: Scaling parameter c_d in Equation (1).
- `CE`: Scaling parameter c_e in Equation (2)
- `CF`: Scaling parameter c_f in Equation (2)
- `CG`: Scaling parameter c_g in Equation (2)
- `CH`: Scaling parameter c_h in Equation (2)
- `CI`: Scaling parameter c_i in Equation (2)
- `CFAC`: Determines how explicit or implicit the time-stepping scheme is. c must be strictly greater than zero and strictly less than 1. The time derivative of a function f, $\partial_t f$, over a time-step is a linear combination of $\partial_t f$ at the current time-step (i.e. $\partial_t f^i$) and $\partial_t f$ at the next time-step (i.e. $\partial_t f^{i+1}$) such that

  $$\partial_t f = c\partial_t f^i \ + \ (1-c)\partial_t f^{i+1}$$

  If $c = 0.5$, the weighting is equal and we have a Crank-Nicolson scheme. $c < 0.5$ puts greater weight on the value at the next time-step and makes the scheme more implicit (usually more stable, and found to be necessary for many calculations). $c = 0.4$ has been used often. For circumstances where it is stable, $c = 0.5$ is the most accurate.
- `DELTAT`: fixed value of the time-step, Δt.
- `STIME`: starting time. Usually set to zero, but can be useful to set it to higher values for "smooth continuation" of runs which have expired.

- `IOUTF`: output writing flag. Set to 0 to stop all output. Set to 1 to write to a file and set to 6 to write to screen. The amount of output is enormous and so should always be switched off for serious calculations. The option is intended for the initialisation of new runs. If the time-stepping procedure fails, it is a good idea to switch output on. This may reveal whether the predictor-corrector scheme is not converging (time-step too large?) or whether some other fault has occured.
- `NTS`: the total number of time-steps to be taken.
- `NTSBB`: number of time-steps between each output of a "backup solution vector".
- `NTSBS`: number of time-steps between each labelled solution vector output.
- `ITMX`. The maximum number of iterations allowed in the predictor-corrector time-stepping scheme. `ITMX` is the highest permitted value of j.
- `DTOL` required norm between successive iterations in the predictor-corrector scheme. A large value for `DTOL` essentially says *accept the first iteration of the corrector*. It may be worth setting this to a very small value for a trial run in order to find, by trial and error, how small a time-step is required such that the solution converges to the desired precision with a single iteration.
- `NTSBSE` number of time-steps between each evaluation of kinetic energy.
- `NTSBLE` number of time-steps between each breakdown of magnetic and kinetic energy in terms of the spherical harmonic degree, l.
- `NTSBME` number of time-steps between each breakdown of magnetic and kinetic energy in terms of the spherical harmonic order, m.

1.1 Subprograms required for o2ibtctsc2

SUBS subroutines

```
fopen.f shkeer.f hmfrd.f svfrd.f xarrrd.f svfdcf.f
fdcmbd.f iiasce.f mc2scv.f pvccf.f nphpf.f gauwts.f
schnla.f pvtsmf.f tvtsmf.f tmtsmf.f avbmbr.f vobmbr.f
asvcpl.f svfwt.f pvvcpl.f vecop.f nsvhst.f nsvbta.f
rsdv2c.f sf2vgc.f sf2sdc.f xsvsdc.f rsdv2d.f sf2vgd.f
sf2sdd.f xsvsdd.f nmcxse.f fclose.f bihfrd.f itvaaf.f
fnamer.f asvdr.f ldgnmf.f gfdcfd.f matop.f amlica.f
amsdea.f amdlt.f vobmar.f nmlica.f dvecz.f fftrlv.f
svrint.f matind.f bmrcop.f powtwo.f
```

SUBS double precision function

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

SUBS integer function

```
indfun.f
```

BLAS double precision function

```
dnrm2.f
```

BLAS integer function

```
idamax.f
```

BLAS subroutines

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

LAPACK subroutines

```
dgbtrs.f dgbtrf.f xerbla.f dgetrf.f dgetri.f dgbtf2.f
dlaswp.f dgetf2.f dtrtri.f dtrti2.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, NH1MAX, NH2MAX, NH3MAX, NHMAX,
1                 NIVMAX, NDCS, NCFM, NDRVM, NBNM, NIV1MX,
2                 NIV2MX, NIV3MX, NCMXX, NRMAXC, NIVCMX,
3                 NHCMAX, NNDM
 PARAMETER      ( NRMAX = 50, NH1MAX = 1000, NH2MAX = 1000,
1                 NH3MAX = 1000, NCMXX = 13, NRMAXC = 100,
2                 NHMAX = NH1MAX+NH2MAX+NH3MAX, NHCMAX = 300,
3                 NIVMAX = NHMAX*NRMAX, NIVCMX = NHCMAX*NRMAXC )
```

```
 PARAMETER      ( NIV1MX = NH1MAX*NRMAX,
1                 NIV2MX = NH2MAX*NRMAX,
2                 NIV3MX = NH3MAX*NRMAX )
 PARAMETER      ( NDCS = 4, NBNM = 3, NCFM = 2*NBNM+1,
1                 NDRVM = 4, NNDM = 6 )
 INTEGER          LHMAX, NTHMAX, NPHMAX, NPMAX
 PARAMETER      ( LHMAX = 52, NTHMAX = 100, NPHMAX = 100,
1                 NPMAX = (LHMAX+1)*(LHMAX+2)/2 )
```

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

- `NRMAX` is the maximum permitted number of radial grid nodes.
- `NH1MAX` is the maximum permitted number of poloidal velocity spherical harmonic radial functions.
- `NH2MAX` is the maximum permitted number of toroidal velocity spherical harmonic radial functions.
- `NH3MAX` is the maximum permitted number of temperature spherical harmonic radial functions.
- `NRMAXC` is the maximum permitted number of radial grid nodes for the inhomogeneous temperature function.
- `NHCMAX` is the maximum permitted number of spherical harmonic radial functions for the inhomogeneous temperature function.
- `NBNM` is the number of upper (and lower) diagonals in the banded matrix. `NBNM = 3` is recommended value.
- `NNDM` is the number of nodes used for interpolating radial functions (subroutine `ITVAAF`). This can be quite large without any harm done as the calculation is only performed once at the beginning: interpolating the inhomogeneous temperature onto the same grid as the solution vector.
- `LHMAX` is the highest permitted spherical harmonic degree, l.
- `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.

1.3 Outputs from O2IBTCTSC2

In addition to the snapshots of solution vectors, the number and names of which being determined by the integer numbers `NTS`, `NTSBB` and `NTSBS`, there are two files output: `root.nrg` and `root.log`.

`root.nrg` has a line added to it every `NTSBFE` time-steps. This line contains, in the format `(1PD16.7,1PD16.7,1PD16.7,1PD16.7)`, the four numbers `DTIME`, `DTOTKE`, `DEATOT` and `DTORKE`.

`DTIME` is the time elapsed, `DTOTKE` is the volume integral of the total kinetic energy, `DEATOT` is the volume integral of the kinetic energy contained in the equatorially anti-symmetric components only, and `DTORKE` is the volume integral of the kinetic energy contained in the toroidal components only.

`root.log` reports back on the physical size of the problem, the allocation of finite difference schemes and echoes back the physical parameters read in. Then, every `NTSBLE` time-steps, a breakdown of kinetic energy is done as a function of spherical harmonic degree, l. For every l, such that the kinetic energy stored in spherical harmonic radial functions with degree l is not negligible (i.e. less than a specified parameter `DLOW`), a line is written to `root.log` in the format `('L= ',I5,1PD16.7,1PD16.7,1PD16.7,1PD16.7)` with the numbers `L`, `DTIME`, `TOTAL`, `EA` and `TOR`. `L` is the spherical harmonic degree, l, `DTIME` the time elapsed and `TOTAL`, `EA` and `TOR` are the volume integrals of the total kinetic energy, the equatorially anti-symmetric kinetic energy and the toroidal kinetic energy respectively. For example, the lines

```
L=    31   6.0000000D-05   8.1171900D-03   0.0000000D+00   8.1171900D-03
L=    32   6.0000000D-05   2.5628502D-03   0.0000000D+00   0.0000000D+00
```

indicate that at $t = 0.00006$, the volume integral of the kinetic energy contributed by velocity harmonics with $l = 31$ was 0.00812, of which none was stored in poloidal or equatorially anti-symmetric terms. The volume integral of the kinetic energy contributed by velocity harmonics with $l = 32$ was 0.00256, of which none was stored in toroidal or equatorially anti-symmetric terms. This is a consequence of an equatorial symmetry restriction on the velocity.

Every `NTSBME` time-steps, a breakdown of kinetic energy is done as a function of spherical harmonic order, m. The lines output are entirely analogous to the `L` spectrum. The first character is `M`, to indicate wavenumber. The next four numbers are once again time, total kinetic energy integral, equatorially anti-symmetric kinetic energy integral and toroidal kinetic energy integral.

1.4 Sample runs of o2ibtctsc2

The directory

```
$LEOPACK_DIR/SAMPLERUNS/O2IBTCTSC2
```

contains example input files, initial solution vectors 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

In the [GGZ07], we wish to calculate non-linear solutions to the equations

$$P_r^{-1}\left(\frac{\partial}{\partial t}+\boldsymbol{v}.\nabla\right)\boldsymbol{v}+E^{-1}\boldsymbol{k}\times\boldsymbol{v}=-\nabla\tilde{p}+RT\boldsymbol{r}+\nabla^2\boldsymbol{v} \tag{3}$$

and

$$\frac{\partial T}{\partial t}+\boldsymbol{v}.\nabla T=\nabla^2 T+\boldsymbol{v}.\boldsymbol{r}. \tag{4}$$

The problem is now controlled by the Prandtl number, P_r, the Rayleigh number, R, and the Ekman number, E, given respectively by

$$P_r=\frac{\nu}{\kappa},\ R=\frac{\alpha\beta\gamma d^6}{\nu\kappa},\ \text{and}\ E=\frac{\nu}{2\Omega d^2}. \tag{5}$$

The aspect ratio $\eta = r_{\rm i}/r_{\rm o}$ is set to 0.4. The temperature, T, satisfies the boundary conditions

$$T(r,\theta,\phi)\Big|_{r_{\rm i}}=0 \tag{6}$$

and

$$\frac{\partial T(r,\theta,\phi)}{\partial r}\Big|_{r_{\rm o}}=-\mathcal{A}Y_2^{2c}(\theta,\phi) \tag{7}$$

where $\mathcal{A}$ is a scale for the outward heat-flux.

We impose these boundary conditions by decomposing the temperature, T, such that

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

The temperature function, $T_{\rm a}$, is output in standard format using the program `itfvf`. Firstly, in the file `itfvf.coeffs` - we specify the Y_2^{2c} form for the heat-flux:

```
OB   2   2   1        1.0
```

Then we specify an input file for `itfvf` as follows:

```
example_aHEATFLUX                           : Output filename stem
itfvf.coeffs                                : File containing coeff.s
 60  2   4   0.66666666   1.66666666 2      : nr isp ifrmf ri ro ithebc
 0.0     1.0                                : epsi  epso
```

This creates the three files

```
example_aHEATFLUX.ints
example_aHEATFLUX.vecs
example_aHEATFLUX.xarr
```

which specify a temperature function which disappears at the inner boundary and, at the outer boundary, has a radial derivative equal to the Y_2^{2c} spherical harmonic.

By comparing equations (3) and (4) with equations (2) and (1), and applying the parameters $E = 5 \times 10^{-3}$, $R = 1200$ and $P_r = 0.6$, we get the scalings as given in the `o2ibtctsc2` input file, `example_a.input`.

We want to look at the case $\varepsilon = 0.3$ and so set `SCAL = 0.3`. **IMPORTANT !**. Since `SCAL` is positive, this means that the temperature gradient at the surface is positive in the shape specified. At $\phi = 0$, the temperature gradient is positive meaning that the outward heat-flux is below average at this point. Set `SCAL` to be negative if it is the outward heat-flux you wish to impose. In the case Y_2^{2c} ofcourse, it is not really of any consequence given the simple $\cos 2\phi$ longitude dependency: a rotation of $\pi/2$ will give you exactly the same pattern with the reverse sign. However, if you are studying patterns with a certain longitudinal asymmetry - such as the seismic shear-wave variations, then the sign of the parameter `SCAL` is crucial.

Note that the inhomogeneous temperature function, T_{a}, has different radial spacings (`example_aHEATFLUX.xarr`) to the solution vector (`example_a.xarr`). As an exercise, when reproducing these results, alter the settings in the file `itfvf.input` and see what effect, if any, is had on the output.

References

[GGZ07] S. J. Gibbons, D. Gubbins, and K. Zhang. Convection in rotating spherical fluid shells with inhomogeneous heat flux at the outer boundary. *Geophysical & Astrophysical Fluid Dynamics*, 101(5-6):347–370, 2007.