

# text1r — calculate 1D radial texture

June 23, 2015

This library calculates  $^3\text{He}$ -B texture in a 1D cylindrically symmetric geometry. Code and ideas came from ROTA programs (by J.Kopu, S.Autti, ...). Interfaces for C, F, F90, Matlab, and Octave languages are available.

## Energy terms

Sum of following energies is minimized:

Gradient energy:

$$F_G = \lambda_{G1} \int_V \frac{\partial R_{\alpha i}}{\partial r_i} \frac{\partial R_{\alpha j}}{\partial r_j} + \lambda_{G2} \int_V \frac{\partial R_{\alpha j}}{\partial r_i} \frac{\partial R_{\alpha j}}{\partial r_i}$$

Magnetic energy in a uniform field  $H$  along  $z$  axis:

$$F_{DH} = -a \int_V (\mathbf{n} \cdot \mathbf{H})^2 = -aH^2 \int_V \sin^2 \beta_N$$

Spin-orbit energy for arbitrary distribution of precessing magnetization  $\Psi$ :

$$F = \frac{8}{5} \frac{\chi \Omega_B^2}{\gamma^2} \sin^2 \frac{\beta_L}{2} \sin^2 \frac{\beta_M}{2} = 15 \lambda_D \sin^2 \beta_M \sin^2 \frac{\beta_M}{2}, \quad \left( \sin^2 \frac{\beta_L}{2} = \frac{5}{8} \sin^2 \beta_N \right)$$

Superflow energy for arbitrary distribution of counterflow  $\mathbf{v}_s - \mathbf{v}_n$ :

$$F_{HV} = -\lambda_{HV} \int_V [\mathbf{H} \cdot \mathbf{R} \cdot (\mathbf{v}_s - \mathbf{v}_n)]^2$$

Vortex energy for arbitrary distribution of vortex density  $\Omega_v$  and polarization  $\mathbf{l}_v$ :

$$F = \frac{1}{5} \frac{\lambda}{\Omega} \int_V \Omega_v [\mathbf{H} \cdot \mathbf{R} \cdot \mathbf{l}_v]^2$$

Surface energy:

$$\begin{aligned} F_{SH} &= -d \int_S [\mathbf{H} \cdot \mathbf{R} \cdot \mathbf{s}]^2 d^2r \\ F_{SG} &= (2\lambda_{G2} + \lambda_{SG}) \int_S s_j R_{\alpha j} \frac{\partial R_{\alpha i}}{\partial r_i} d^2r \end{aligned}$$

### Note:

- $\lambda/\Omega$  is not set automatically because it is not known (only counterflow part is known which is important at high temperatures).
- Dipole energy is assumed to be constant,  $\theta = \cos^{-1}(-1/4)$ .
- Maximal number of points is hardcoded into the library. You can increase it by changing MAXN parameter in `make.inc` script and recompiling everything.
- $\alpha_N$  can be noisy for flat textures where  $\beta_N$  is near zero (large  $\xi_H$  limit)

# Usage

## General

To keep parameters and results we use a global data structure with following fields:

```
n      -- number of points
rr()   -- r grid [cm]
an()   -- azimuthal angle of the n vector [rad]
bn()   -- polar angle of the n vector [rad]
R      -- Cell radius [cm]
H      -- Magnetic field [G]

a      -- Textural dipole-field parameter a [erg/(cm3 G2)]
lg1    -- Textural parameter lambda_g1 [erg/cm]
lg2    -- Textural parameter lambda_g2 [erg/cm]
lhv    -- Textural parameter lambda_HV [erg/(cm3 G2) 1/(cm/s)2]
lsg    -- Textural parameter lambda_SG [???]
ld     -- Textural parameter lambda_D [erg/cm3]
lo     -- Textural parameter lambda/omega [s/rad]
d      -- Textural surface parameter d [erg/(cm2 G2)]

vr(), vz(), vf() -- velocity profile
lr(), lz(), lf() -- vortex polarization
w()     -- vortex dencity
bm()    -- tilting angle of precessing magnetization

energy -- final energy after minimization
db0    -- d(bn)/dr at r=0 after minimization
db1    -- d(bn)/dr at r=R after minimization
bmax   -- bn at r=R after minimization

bnd    -- use strict boundary condition instead of surface energies
abnd,bbnd -- boundary condition, an and bn at r=R
al()   -- azimuthal angle of the calculated l vector [rad]
bl()   -- polar angle of the calculated l vector [rad]

err    -- error code after minimization
```

Following functions are provided to calculate texture:

- **text1r\_init(ttc, p, nu0, r, n, itype)** – Initialize data structure. Textural parameters are set according to temperature **ttc** ( $T/T_c$ ) and pressure **p** (bar) using **libhe3** library.  $\lambda/\Omega$  parameter is set to zero (no theory). Magnetic field is set from **nu0** Larmor frequency. Vortex and counterflow distributions are set to zero. Initial distributions for  $\alpha_N$  and  $\beta_N$  are set according to **itype** parameter: 0 means usual flare-out texture, 1 means texture with 90-degree peak, 2 and more means larger rotation of **n** vector.
- **text1r\_set\_vortex\_cluster(dat, omega, omega\_v)** – Set counterflow and vortex profiles for central vortex cluster. Here **omega** is a rotation velocity of the container and **omega\_v** is rotation velocity of the cluster.
- **text1r\_set\_vortex\_uniform(dat, omega, omega\_v)** – Set counterflow and vortex profiles for uniform vortex cluster. Here **omega** is a rotation velocity of the container and **omega\_v** is rotation velocity of the cluster.
- **text1r\_set\_vortex\_twisted(dat, omega, kr)** – Set counterflow and vortex profiles for twisted vortex cluster.

- **text1r\_minimize(msglev)** – Vary  $\alpha_N$  and  $\beta_N$  to find energy minimum. Parameter `msglev` is used to control verbosity level. To turn off all messages use -3.
- **text1r\_print(filename)** – Print all data to a file.

Additional `matlab` functions are provided to deal with magnon condensates (see below).

## Fortran

Examples of Fortran 77 and Fortran 90 programs can be found in `examples` folder.

Simple usage:

```
include '../text1r.fh'
call text1r_init(ttc, p, nu0, r, n, itype)
text_lo=5D0;
call text1r_set_vortex_cluster(omega, omega_v);
call text1r_minimize(msglev)
call text1r_print('result.dat')
```

Data is arranged as a common block `text1r_pars` with fields `text_n`, `text_rr`, `text_an` etc.

## C

Example of C program can be found in `examples` folder.

Simple usage:

```
#include "text1r.h"
...
text1r_init_(&ttc, &p, &nu0, &r, &n, &itype);
text1r_pars_.lo=5;
text1r_set_vortex_cluster_(&omega, &omega_v);
text1r_minimize_(&msglev);
text1r_print_(fname, strlen(fname));
```

You should include `text1r.h` header file. Data is arranged as a global structure `text1r_pars_` with fields `n`, `rr`, `an` etc. Usual way of calling Fortran functions from a C program is used.

## Matlab/Octave

MEX-files for Matlab and Octave can be found in `matlab` folder. Example of matlab script can be found in `examples` folder.

Simple usage:

```
dat = text1r_init(ttc, p, nu0, r, n, itype);
dat = text1r_set_vortex_cluster(dat, omega, omega_v);
dat.lo = 5;
dat = text1r_minimize(dat, msglev);
```

- additional `dat` parameter is used to keep all texture data. It is a matlab structure with fields `n`, `rr`, `an` etc.
- `n` and `itype` parameters in `text1r_init` can be omitted. Default values: `MAXN` and `0`.
- `msglev` parameter in `text1r_minimize` can be omitted. Default value is `-3`

## Additional matlab functions

- `[psi en] = text1r_wave(dat, states)`. Calculate magnon wavefunction in a textural potential for states (0,1,2...)
- `dat = text1r_qball_bm0(dat, bm0)`. Calculate self-consistent texture and precessing magnetization distribution for a given  $\beta_M(r=0)$  value (in radians).
- `[dat e0] = text1r_qball_e(dat, E)`. Same, but for a given  $\int(1 - \cos \beta_M)$

## Technical details

### Program structure

- **ebulk**, **egrad** and **esurf** subroutines calculate bulk, gradient and surface energy and its derivatives as a function of texture ( $\alpha$  and  $\beta$ ) and texture gradients ( $\partial\alpha/\partial r$ ,  $\partial\beta/\partial r$ ).
- **eint** subroutine calculates total energy as integral of bulk + gradient energy over volume plus integral of surface energy over surface and its derivatives as a function of texture and texture gradients at the whole grid.
- **mfunc** is a wrapper for **egrad**. Texture is represented as 1-d array suitable for minimization. Conversion is done by **x2text** and **text2x** subroutines.
- **minimize** subroutine does minimization.
- $n_x/(1 + n_z)$  and  $n_y/(1 + n_z)$  are used as minimization parameters.