Tutorial 1 & 2: getting the code to run, thermodynamics, expectation values, flow diagrams, ground-state energy

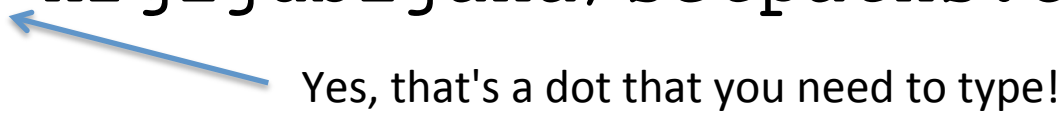
Rok Žitko
Institute Jožef Stefan
Ljubljana, Slovenia

Material for these tutorials

- Available from: http://nrgljubljana.ijs.si/tutorial.tar.gz
- The powerpoint slides will be posted at: http://nrgljubljana.ijs.si/slides-sissa/

Running on SISSA terminals

1. `wget nrgljubljana.ijs.si/nrgljubljana-sissa.tar.gz`
2. `wget nrgljubljana.ijs.si/tutorial-sissa.tar.gz`
3. `tar zxf nrgljubljana-sissa.tar.gz`
4. `tar zxf tutorial-sissa.tar.gz`
5. `. nrgljubljana/setpaths.ch`



That's it!

Alternatively, get the precompiled binaries

- **http://nrgljubljana.ijs.si/**

Binary distribution

There is now an experimental precompiled binary distribution of NRG Ljubljana. It is still being tested.

Linux x86-64 version (tested to work on RHEL 6.3 and derivatives)

Linux x86-64 version (tested to work on RHEL 5.6 and derivatives)

Requirements for compiling the code

- **Boost C++ library** (http://www.boost.org/): ublas numerics, serialization, MPI, containers
- **BLAS, LAPACK routines** (from ATLAS, Intel MKL, AMD Core Math Library, or from Apple framework Accelerate…)
- Optional: GNU Scientific Library (GSL), GNU multiple precision library (GMP)
- Standard stuff: perl, python, gnuplot, C++ compiler, etc.

NOTE: these are all available as packages for most Linux distributions and for Mac via MacPorts. You do not need to recompile them from source!

Mac and Xcode (1/2)

Use the instructions from the MacPorts project:

- http://guide.macports.org/chunked/installing.xcode.html
- http://www.macports.org/install.php

"Always **make sure to install the latest available version of Xcode** for your Mac OS X release; using outdated versions of Xcode may cause port install failures. Also note that Xcode is not updated via Mac OS X's Software Update utility on OS versions prior to 10.6, and is updated via the **Mac App Store** on 10.7."

Mac and Xcode (2/2)

- Start Xcode
- Go to Xcode/Preferences
- Go to the tab "Download"
- Install the "Command line tools"
- The command line tools are required by MacPorts

MacPorts

1. Install Xcode and Xcode command line tools.
2. Install MacPorts, http://www.macports.org/install.php

3. Then run:

```
sudo port install boost
sudo port install atlas
sudo port install gsl
sudo port install gmp
```

You can use framework Accelerate instead.

RHEL 6.3 (or CentOS, Fedora, etc.)

As superuser (root) run the following in the shell:

```
yum groupinstall 'Development Tools'
yum install boost boost-devel
yum install atlas atlas-devel
yum install gsl gsl-devel
yum install gmp gmp-devel
yum install gcc-gfortran
yum install gnuplot
or yum install gnuplot44
```

If the compiler complains about not finding fortran compiler.

Try to get gnuplot version 4.4 or above.

Tested on Scientific Linux 6.3, on June 5th 2013

Handling problems with library linking (common problem, unfortunately...)

Problems with BLAS routines from ATLAS? Try:

```
export LDFLAGS="-L/usr/lib64/atlas"
```

Problems with linking boost routines? Try:

```
./configure –with-tools –prefix=$HOME
–with-boost-serialization=boost_serialization
```

Sometimes linking fails because the order of the arguments to the linker is not what the linker expected. In such cases a simple (but inelegant) workaround is to copy & paste the offending command and repeat the library calls at the end of the line (-lgsl –lgslcblas, when linking gsl fails).

... or consult local computer experts!

Extra step on Mac computers

- If compiling from source:

From nrgljubljana-2.3.20/ directory, do the following:

```
cd mac
sudo cp math /usr/local/bin
```

(This is necessary for `nrginit` to be able to call Mathematica. On Linux, the Mathematica installer does this for you.)

- If using binary distribution: the script `math` is already in nrgljubljana/bin. You don't need to do anything.

Compiling boost from source (if everything else fails)

- Get boost (www.boost.org)

```
tar zxvf boost_1_53_0.tar.gz
cd boost_1_53_0
./bootstrap.sh --prefix=$HOME/boost
./b2 install
```

- Then add `--with-boost=$HOME/boost` as an option to configure

Obtaining the code

- **http://nrgljubljana.ijs.si/**

Licence and obtaining the code

I advocate open access to knowledge in science. The **"Ljubljana NRG"** framework is thus licenced under the General public licence (GPL). You are entitled to run the program, for any desired purpose, study how the program works and modify it, redistribute copies and improve the program. You are encouraged (but not required) to advertise "NRG Ljubljana". If the framework is used in producing published scientific results, you might, for example, acknowledge its use in the ensuing paper/poster/presentation.

Download the current version (2.3.20) of "NRG Ljubljana"

The package should work unmodified on any modern Linux distribution and, with some tweaking, on any Unix or Unix-like operating system with a good standards-compliant C++ compiler. (Mac OS X is fine.) The following libraries are required to compile the C++ part of the NRG code:

- LAPACK and BLAS linear algebra libraries - tested with (free) Atlas, and (commercial) Intel's Math Kernel Library (MKL).
- boost C++ libraries, in particular the uBLAS library.

In addition, Wolfram Research Mathematica must be installed for running the Mathematica part of the NRG code. Mathematica is only required for the initialization of the problem (basis construction, diagonalisation of the initial Hamiltonian, transformations of the operator matrices, etc.) which is relatively fast. When "NRG Ljubljana" is used on a **cluster**, it is therefore sufficient to have Mathematica installed on a single computer (for example on the cluster **host** computer), while the numerically demanding (C++) part of the program can be ran on the cluster **nodes**.

Installation

- Install the required libraries.
- Uncompress NRG Ljubljana:
  `tar zxvf nrgljubljana-2.3.20.tar.gz`
- `cd nrgljubljana-2.3.20`
- Configure:
  `./configure --with-tools --prefix=$HOME`
- Compile:
  `make`
- Install:
  `make install`

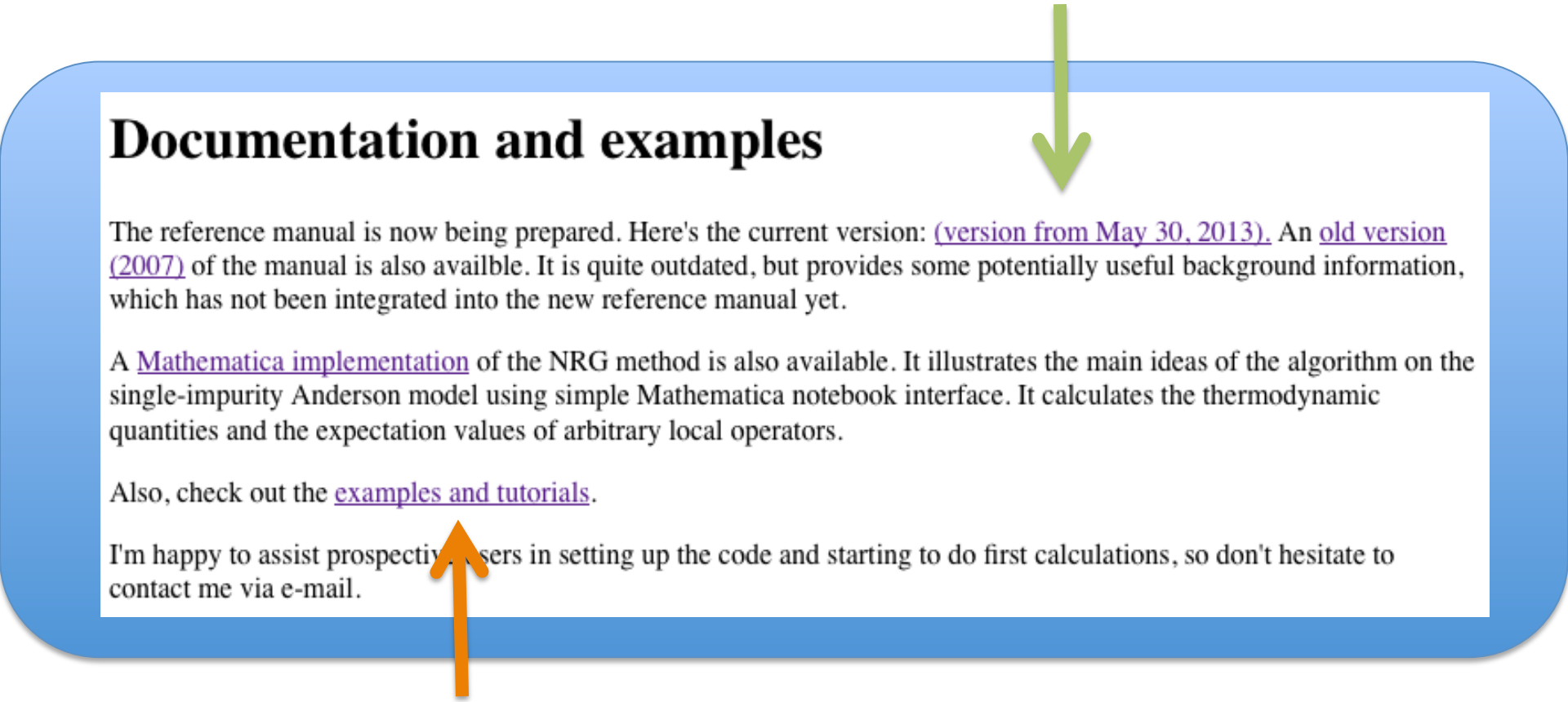
Post installation step

- If you used `--prefix=$HOME`, the executables are in `$HOME/bin`. Make sure this directory is in the `$PATH`. If it is not, you can, for example, add the following line to your **`.bachrc`** (or `.bash_profile`, `.profile`, or similar):

```
export PATH=~/bin:"${PATH}"
```

(You'll need to exit the shell session and start a new one for this to start having an effect.)

NRG Ljubljana documentation



Documentation and examples

The reference manual is now being prepared. Here's the current version: (version from May 30, 2013). An old version (2007) of the manual is also availble. It is quite outdated, but provides some potentially useful background information, which has not been integrated into the new reference manual yet.

A Mathematica implementation of the NRG method is also available. It illustrates the main ideas of the algorithm on the single-impurity Anderson model using simple Mathematica notebook interface. It calculates the thermodynamic quantities and the expectation values of arbitrary local operators.

Also, check out the examples and tutorials.

I'm happy to assist prospecti[illegible]sers in setting up the code and starting to do first calculations, so don't hesitate to contact me via e-mail.

Content of tutorial.tar.gz

```
01_td
01_td_0
01_td_imp
02_td
02_td_0
02_td_imp
02_td_kondo
02_td_kondo_imp
03_expv_kondo_magnetization
03_expv_siam
04_flow_kondo
04_flow_kondo_QN
04_flow_siam
04_flow_siam_QN
05_spec
05_spec_z
06_splitting
07_cond
07_tp
08_cond_kondo
09_underscreened
09_underscreened_spec
10_overscreened_flow
10_overscreened_td
10_overscreened_td_0
11_anderson_holstein
11_anderson_holstein_0
12_self_energy_trick
15_andreev
16_josephson
20_dqd_side_coupled_td
20_dqd_side_coupled_td_0
21_dqd_parallel
22_dqd_serial
23_dqd_serial_Uinf
45_Hubbard_after
45_Hubbard_before
46_Hubbard_MIT_after
46_Hubbard_MIT_before
alt_disc_C
alt_disc_more_states
```

Each example directory contains the **input file** (called **param**), the numerical input to the NRG iteration code (called **data**), the full set of results, the scripts to recreate these results, as well as scripts for plotting the results (mostly using gnuplot).

Organization of the directories

```
[zitko@atos 05_spec]$ ls -al
total 184
drwxr-xr-x  2 zitko users  4096 Jun  5 14:49 .
drwxr-xr-x 37 zitko users  4096 Jun  5 14:49 ..
-rwxr-xr-x  1 zitko users    25 May 31 22:47 1_run
-rwxr-xr-x  1 zitko users   246 Jun  1 15:36 2a_plot
-rwxr-xr-x  1 zitko users   271 Jun  1 20:42 2b_plot_log
-rwxr-xr-x  1 zitko users   317 Jun  1 20:42 2c_plot_log_friedel
-rw-r--r--  1 zitko users 35580 Jun  1 20:42 A-rescaled.dat
-rw-r--r--  1 zitko users  1197 Jun  2 15:06 custom
-rw-r--r--  1 zitko users  3805 Jun  2 15:05 data
-rw-r--r--  1 zitko users 44983 Jun  2 15:07 log
-rw-r--r--  1 zitko users 10010 Jun  2 15:07 log2
-rw-r--r--  1 zitko users  8858 Jun  2 15:05 mmalog
-rw-r--r--  1 zitko users   248 May 31 22:52 param
-rwxr-xr-x  1 zitko users   208 Jun  1 15:39 rescale
-rw-r--r--  1 zitko users 27976 Jun  2 15:07 spec_FDM_dens_A_d-A_d.dat
-rw-r--r--  1 zitko users  8254 Jun  2 15:06 td
```

Single-Impurity Anderson Model

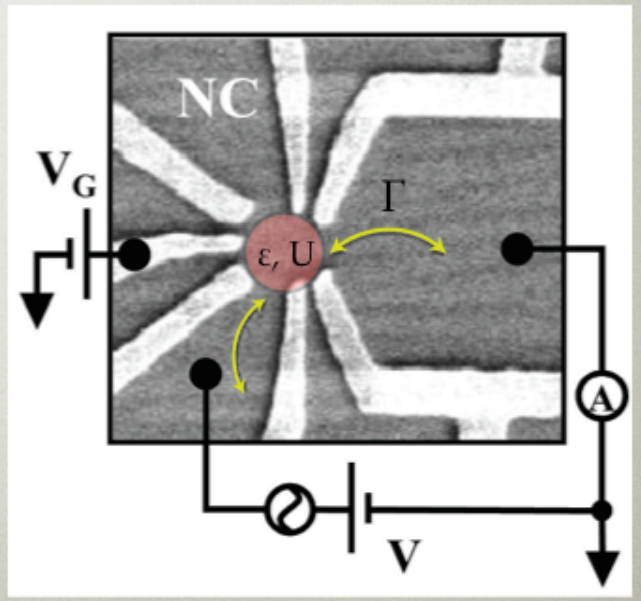
$$H = H_{\text{imp}} + H_{\text{band}} + H_{\text{hyb}}$$

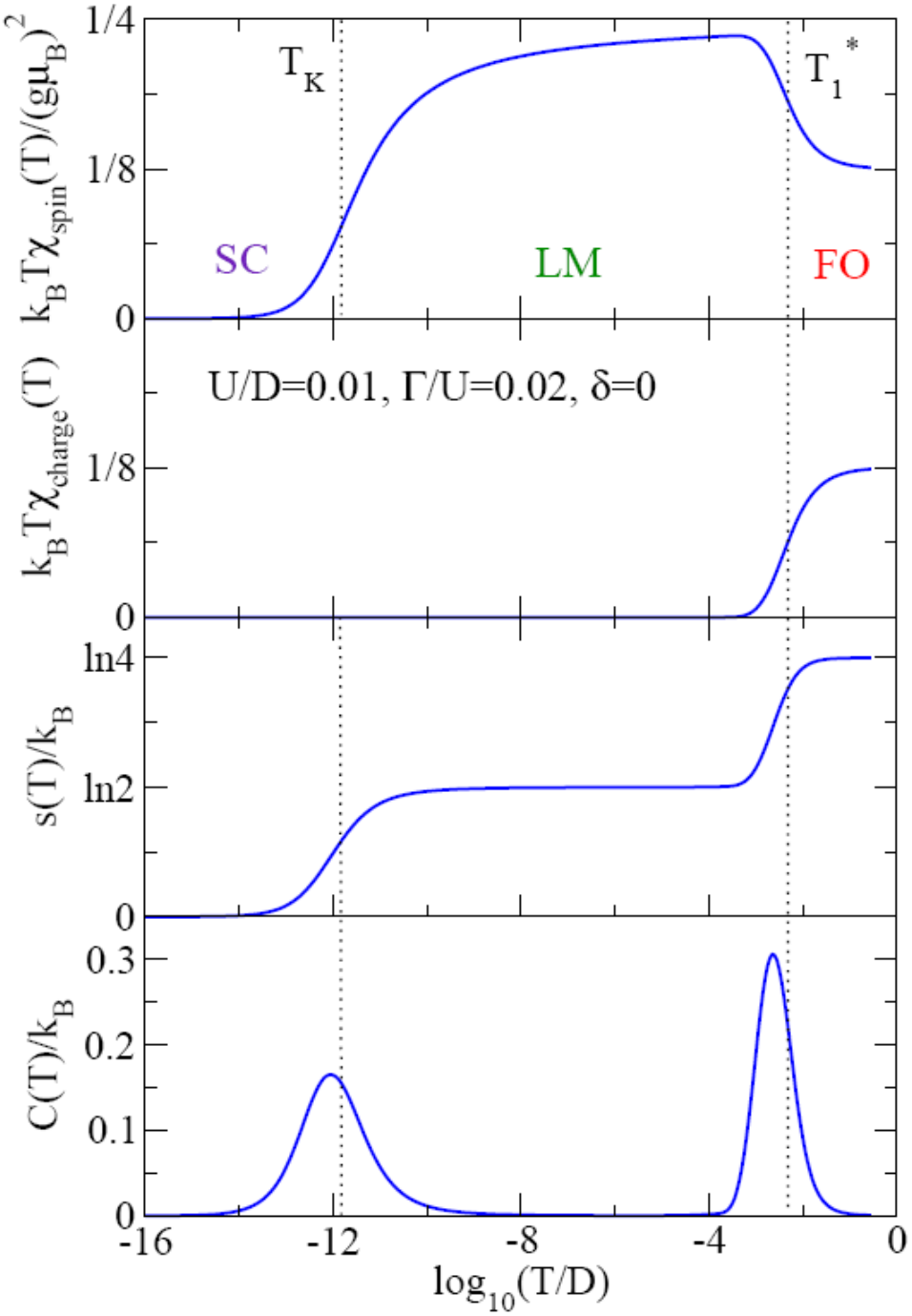
$$H_{\text{imp}} = \sum_{\sigma} \epsilon n_\sigma + U n_\uparrow n_\downarrow \qquad n_\sigma = d_\sigma^\dagger d_\sigma$$

$$H_{\text{band}} = \sum_{\mathbf{k},\sigma} \epsilon_k \, c_{\mathbf{k},\sigma}^\dagger c_{\mathbf{k},\sigma}$$

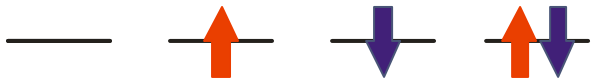
$$H_{\text{hyb}} = \sum_{k,\sigma} \left(V_k \, c_{k,\sigma}^\dagger d_\sigma + \text{H.c.} \right)$$

$$\Delta(\omega) = \sum_k \frac{|V_k|^2}{\omega - \epsilon_k} \approx i\Gamma$$



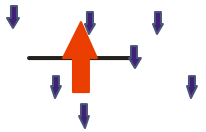


FO: free orbital



LM: local moment

SC: strong coupling



Krishnamurthy, Wilkins, Wilson, PRB **21**, 1003 (1980)

Example 1: thermodynamics of the SIAM

- Go to **01_td/**
- Optionally run the script **1_run**. It takes approx. 2-3 minutes to recalculate the results.

```
[zitko@atos 01_td]$ cat 1_run
#!/bin/sh
nrginit
nrgrun
report td td
```

- Use the scripts **2a_plot-entropy**, etc., to produce the graphs.

The input file **param**

01_td/param

```
[param]
symtype=QS
discretization=Y
Lambda=2
Tmin=1e-10
keepenergy=10
keep=5000

model=SIAM
U=0.01
Gamma=0.0006
delta=0
```

symmetry: charge conservation Q, spin invariance S

discretization scheme: Y, C, Z

Λ

lowest temperature considered, controls the chain length

energy cutoff for the truncation of NRG states

maximum number of states kept

predefined model

model parameters

NRG initialization **data**

```
# Input file for NRG Ljubljana, Rok Zitko, rok.zitko@ijs.si, 2005-2012
# symtype    QS
# $Id: initial.m,v 1.1 2012/05/15 09:58:19 rokzitko Exp rokzitko $
# Using sneg version        1.233
# Model:      SIAM   Variant:              Channels:    1
# U=   0.01    Gamma=     0.0006         Delta=        0        t=      0.
#
# Gamma^(1/2)=    0.019928474108038798
# Lambda=   2.        BAND=        flat
# DISCRETIZATION=         Y        z=      1.
# ops=
#!7
# Number of channels, impurities, subspaces:
1       1       6
# SCALE       1.0606601717798214
# Energies (GS energy subtracted, multiplied by 1/SCALE):
-2      1
1
0.040008362507503385
-1      2
2
0.01871532881885043    0.05658735098824603
```

information about the code (for reproducibility!)

model & parameters

eigenvalues

Output: thermodynamics, **td**

```
# $Id: nrg.cc,v 1.31 2012/10/26 10:18:22 rokzitko Exp rokzitko $ QS
# disk=Y band=flat Lambda=2 z=1
# keep=5000 keepenergy=10 Nmax=66 channels=1 dots=1 betabar=1
#           T       <Sz^2>          <Q>        <Q^2>         <E>        <E^2>             C             F             S
      1.06066     0.250273            0     0.998733    0.0372928   0.00174929   0.000358537      -2.73512       2.77241
         0.75     0.342811 -2.38971e-17      1.36791     0.969774      1.43799      0.497526      -2.92227       3.89204
      0.53033     0.414585  8.91184e-17      1.65363      1.15726      2.52562       1.18637      -3.65375       4.81102
        0.375     0.455022   3.7147e-16      1.81343      1.87943      5.61993       2.08768      -3.53284       5.41226
     0.265165     0.471207 -9.43152e-17      1.87542      1.80704      6.10112       2.83573      -3.89572       5.70276
       0.1875     0.475324  -9.8978e-16        1.888      2.16353      7.87683       3.19596      -3.62524       5.78877
     0.132583     0.476624 -2.83157e-15      1.88772      1.91076      6.91681       3.26581      -3.89142       5.80218
      0.09375      0.47758 -5.60621e-15      1.88382      2.18488      8.04907       3.27538      -3.61793       5.80281
    0.0662913     0.478891 -7.23267e-15       1.8782      1.94711       7.0622       3.27096      -3.85433       5.80144
     0.046875     0.480536 -7.50049e-15      1.86956      2.20899      8.15927       3.27963      -3.58985       5.79884
    0.0331456     0.482907 -3.86627e-15      1.85772      1.99794      7.26981       3.27802      -3.79613       5.79407
    0.0234375     0.486071  8.91873e-15      1.84081      2.25118      8.36007       3.29226      -3.53571       5.78689
    0.0165728     0.490519   2.7768e-14      1.81792      2.06758      7.57761       3.30273      -3.70759       5.77517
    0.0117188     0.496412   5.0081e-14      1.78636      2.31348      8.68439       3.33222      -3.44287       5.75634
   0.00828641     0.504261  7.34783e-14      1.74507       2.1487      7.98583       3.36893       -3.5773         5.726
   0.00585938     0.514037  9.40466e-14      1.69217      2.37433      9.07244       3.43501      -3.30375       5.67808
    0.0041432     0.525532  1.06593e-13      1.62974      2.19753      8.33671       3.50756      -3.40808       5.60561
   0.00292969     0.537363  1.05941e-13      1.56276      2.36443       9.1721       3.58156      -3.14198       5.50641
    0.0020716     0.547672  8.82013e-14      1.50188       2.1434      8.19096       3.59678      -3.24663       5.39004
   0.00146484     0.554371  6.35151e-14      1.45639      2.25527      8.62897       3.54274      -3.02525       5.28052
    0.0010358     0.557172  3.90226e-14      1.42979      2.04731      7.62564       3.43417      -3.15395       5.20126
  0.000732422     0.557003  2.35007e-14      1.41647      2.17528      8.08395       3.35209      -2.98178       5.15707
    0.0005179     0.555377  1.01226e-14      1.41031      2.02307      7.39646       3.30365      -3.11228       5.13535
  0.000366211      0.55296  2.08221e-15      1.40669      2.15468      7.93408       3.29142      -2.96906       5.12375
   0.00025895     0.550169 -1.63703e-15      1.40468      2.03516      7.42282       3.28095      -3.08073       5.11589
  0.000183105      0.54694  1.71493e-15      1.40301      2.14669      7.88892       3.28064       -2.9629        5.1096
  0.000129475     0.543412  9.45343e-15      1.40215       2.0571      7.50807       3.27643      -3.04708       5.10417
  9.15527e-05     0.539392  1.66439e-14      1.40124      2.14047       7.8606       3.27899      -2.95829       5.09875
  6.47376e-05     0.534972  2.28651e-14      1.40089       2.0861      7.63009       3.27826      -3.00722       5.09333
```

temperature — magnetization — heat capacity — entropy

Plotting with gnuplot

01_td/2a_plot-entropy

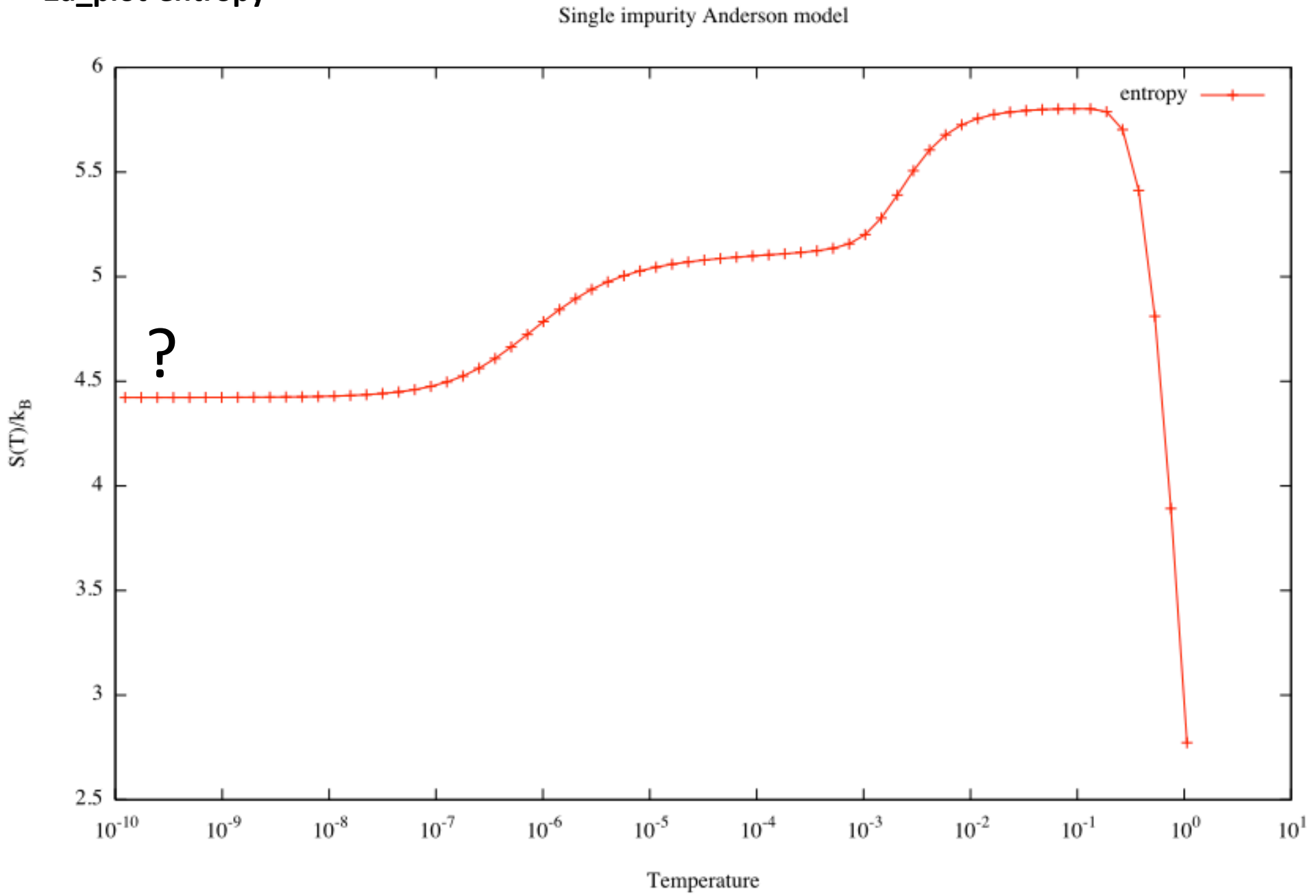
```
#!/bin/sh
gnuplot --persist <<EOF
set termoption enh
set title "Single impurity Anderson model"
set logscale x
set format x '10^{%L}'
set xlabel 'Temperature'
set ylabel 'S(T)/k_B'
plot 'td-S.dat' with lp title 'entropy'
```

plot window persists after gnuplot exits

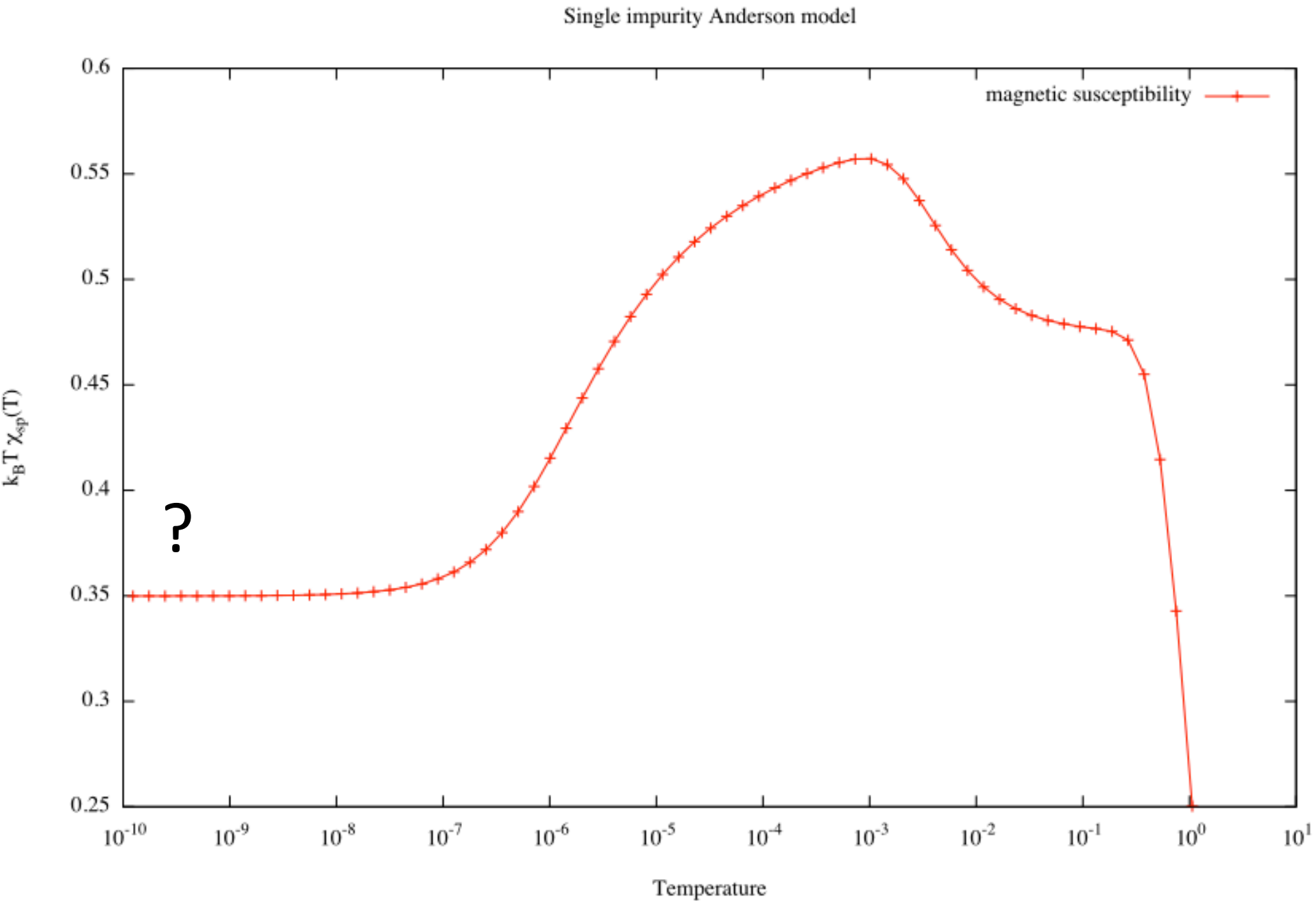
support for Greek characters, subscripts,etc.

10^x notation for powers

2a_plot-entropy



2a_plot-magnetic_susceptibility



What should we be computing?

- Impurity contribution to a thermodynamic quantity *o*

$$\langle o \rangle_{\mathrm{imp}} = \langle o \rangle - \langle o \rangle_0$$

System with impurity

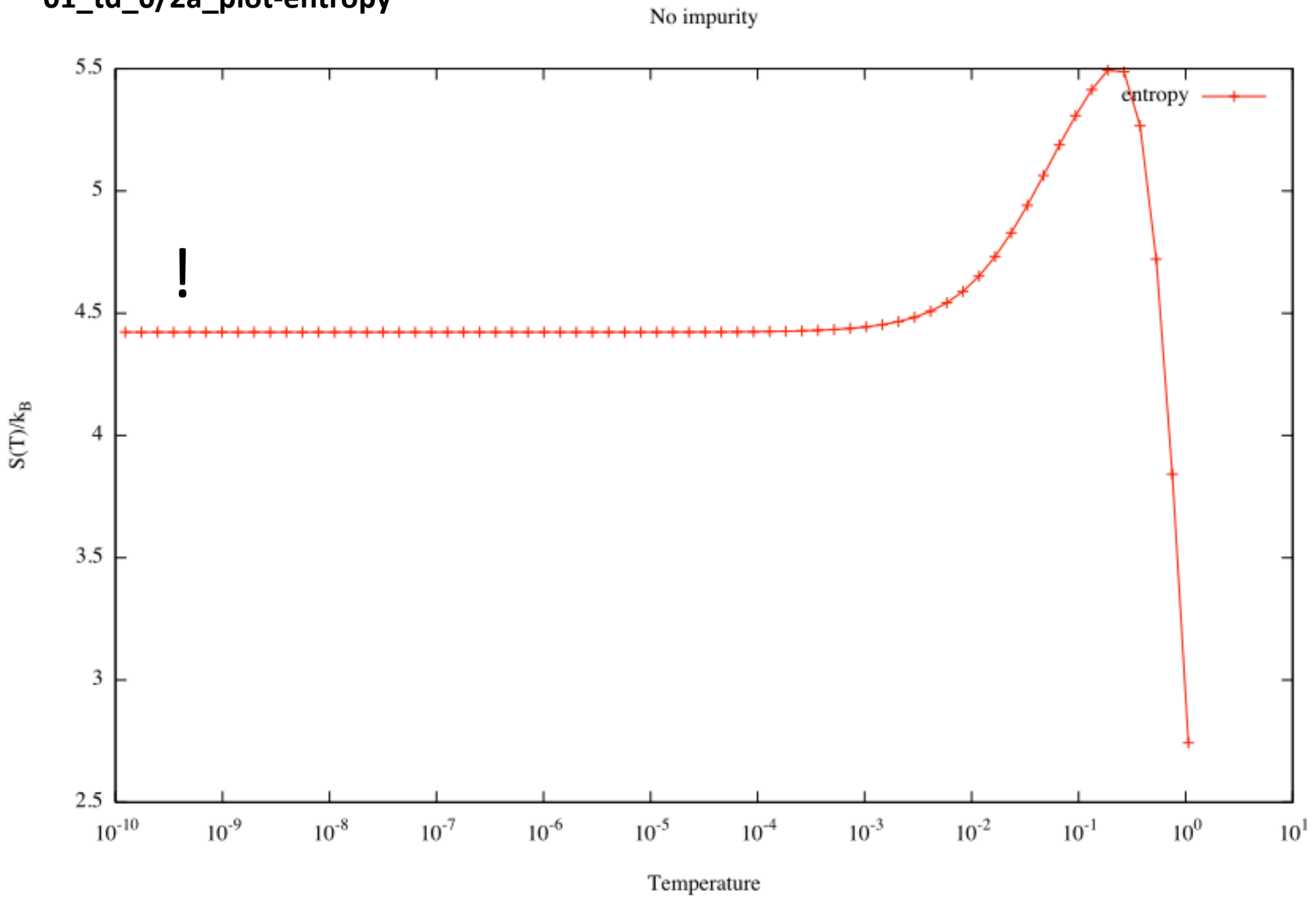
Clean system

$$\langle O \rangle = \langle O \rangle_0 + Nc\langle o \rangle_{\mathrm{imp}} + \mathcal{O}(Nc^2)$$

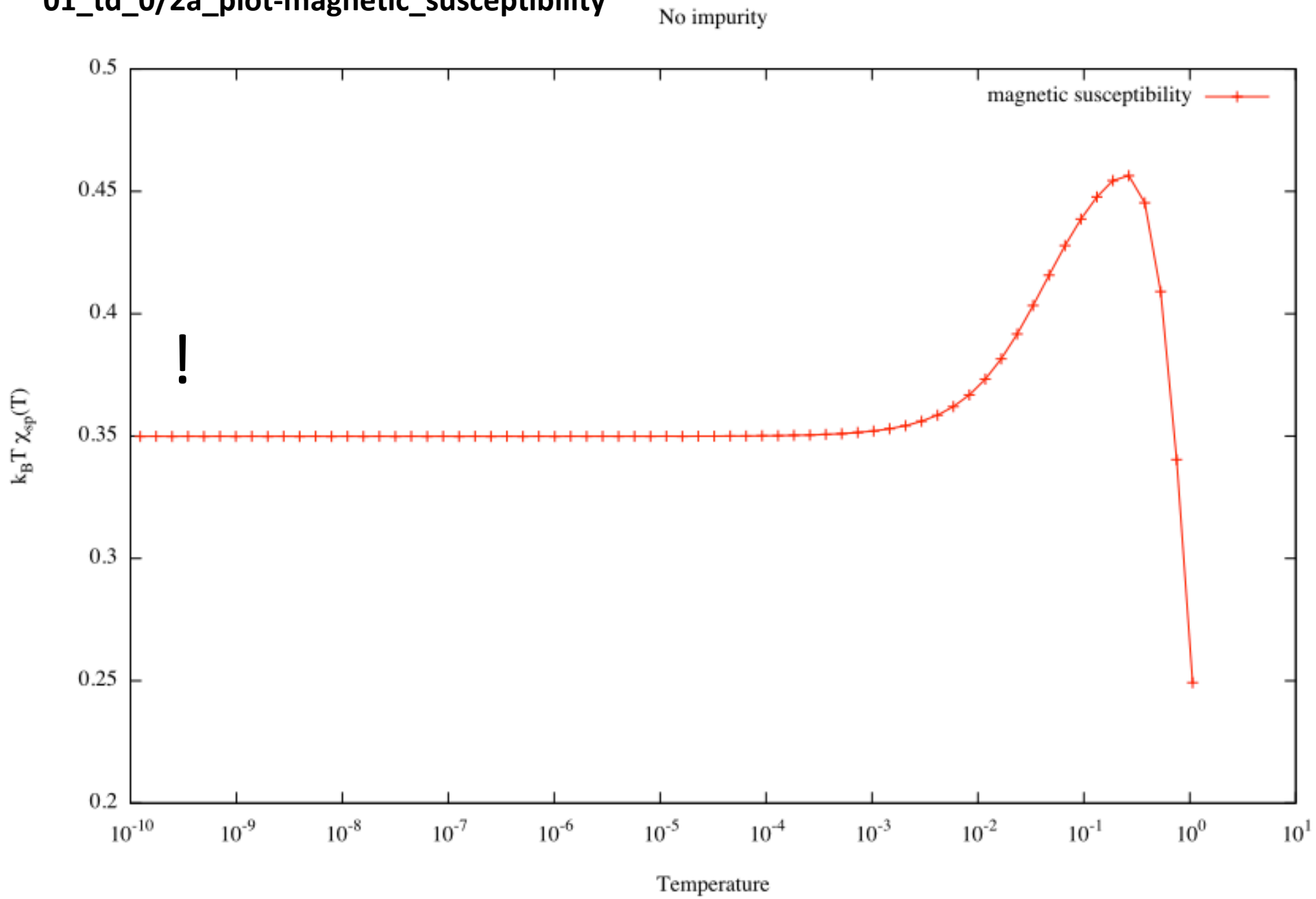
The reference system: Wilson chain without any impurities

- Calculation for the impurity problem: **01_td**
- Reference calculation: **01_td_0**
- Taking the difference: **01_td_imp**

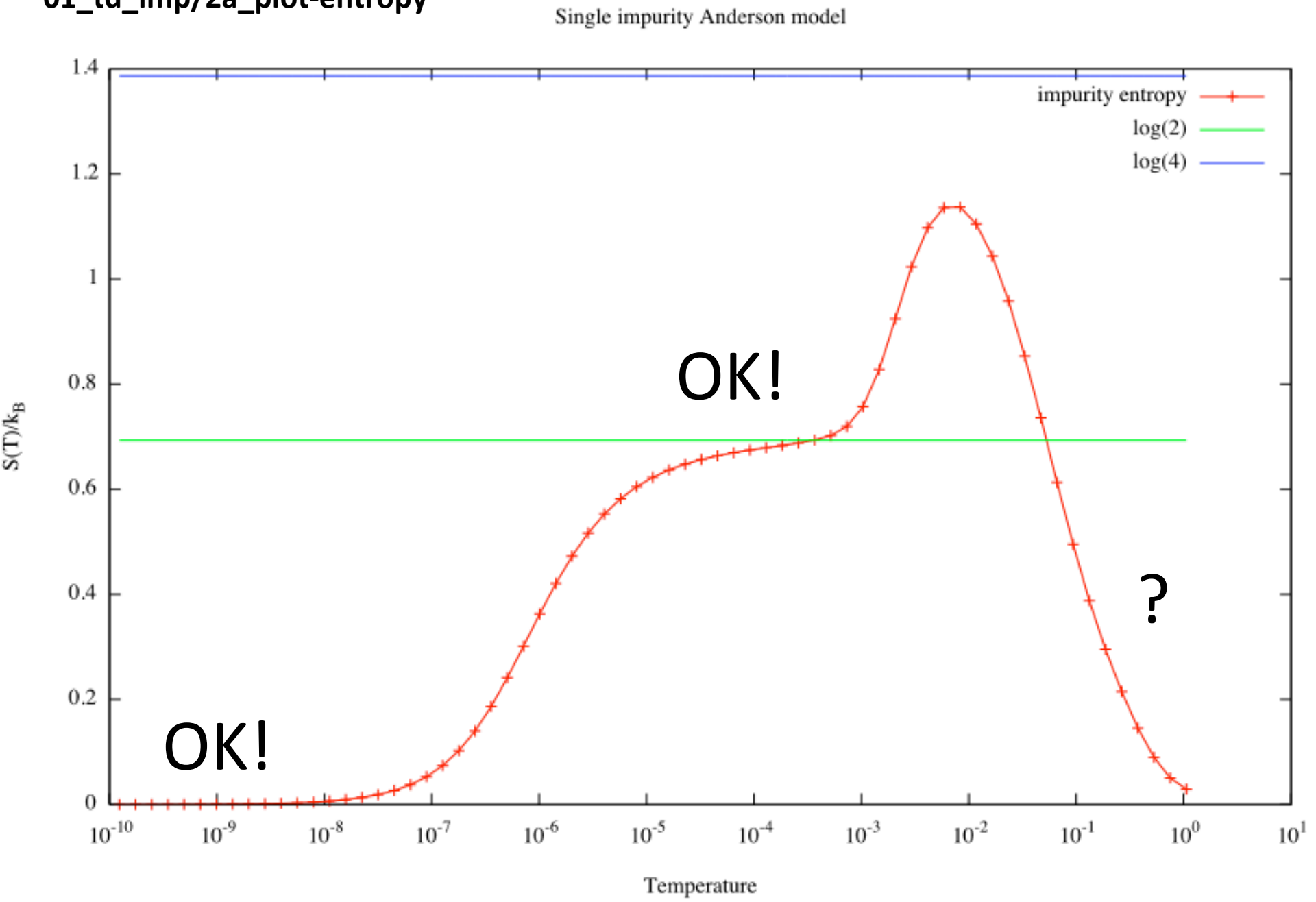
01_td_0/2a_plot-entropy



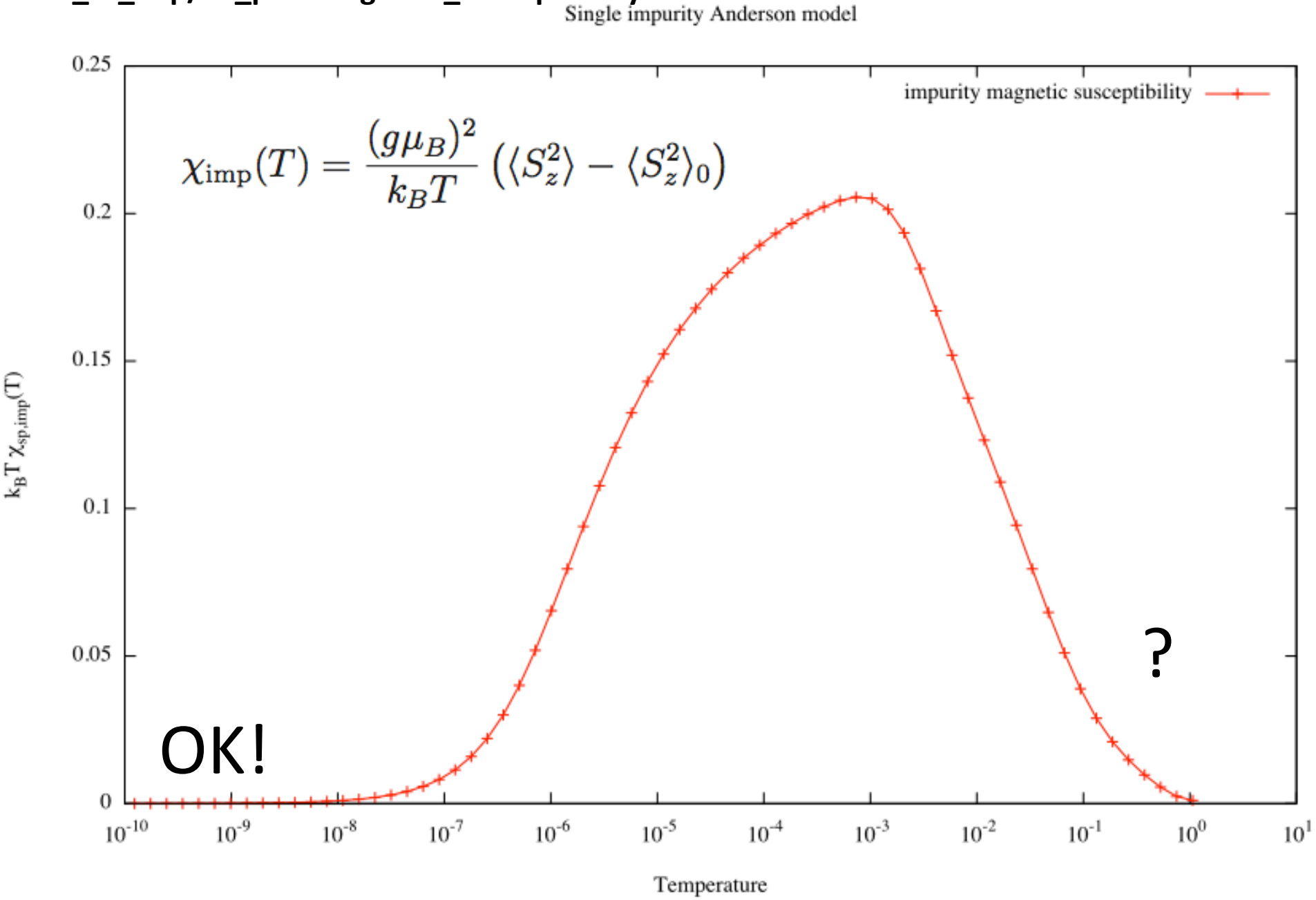
01_td_0/2a_plot-magnetic_susceptibility



01_td_imp/2a_plot-entropy



01_td_imp/2a_plot-magnetic_susceptibility



Z-averaging helps to recover the expected high-temperature asymptotics

02_td/1_zloop

```
#!/usr/bin/env looper
#AUTOLOOP: nrginit ; nrgrun
#OVERWRITE
```

looper is a script bundled with NRG Ljubljana

```
[param]
symtype=QS
discretization=Z
@$z = 1/4; $z <= 1; $z += 1/4
z=$z
Lambda=2
Tmin=1e-10
keepenergy=10
keep=5000

model=SIAM
U=0.01
Gamma=0.0006
delta=0
```

This is essentially a *for* loop using perl syntax!

Postprocessing tools

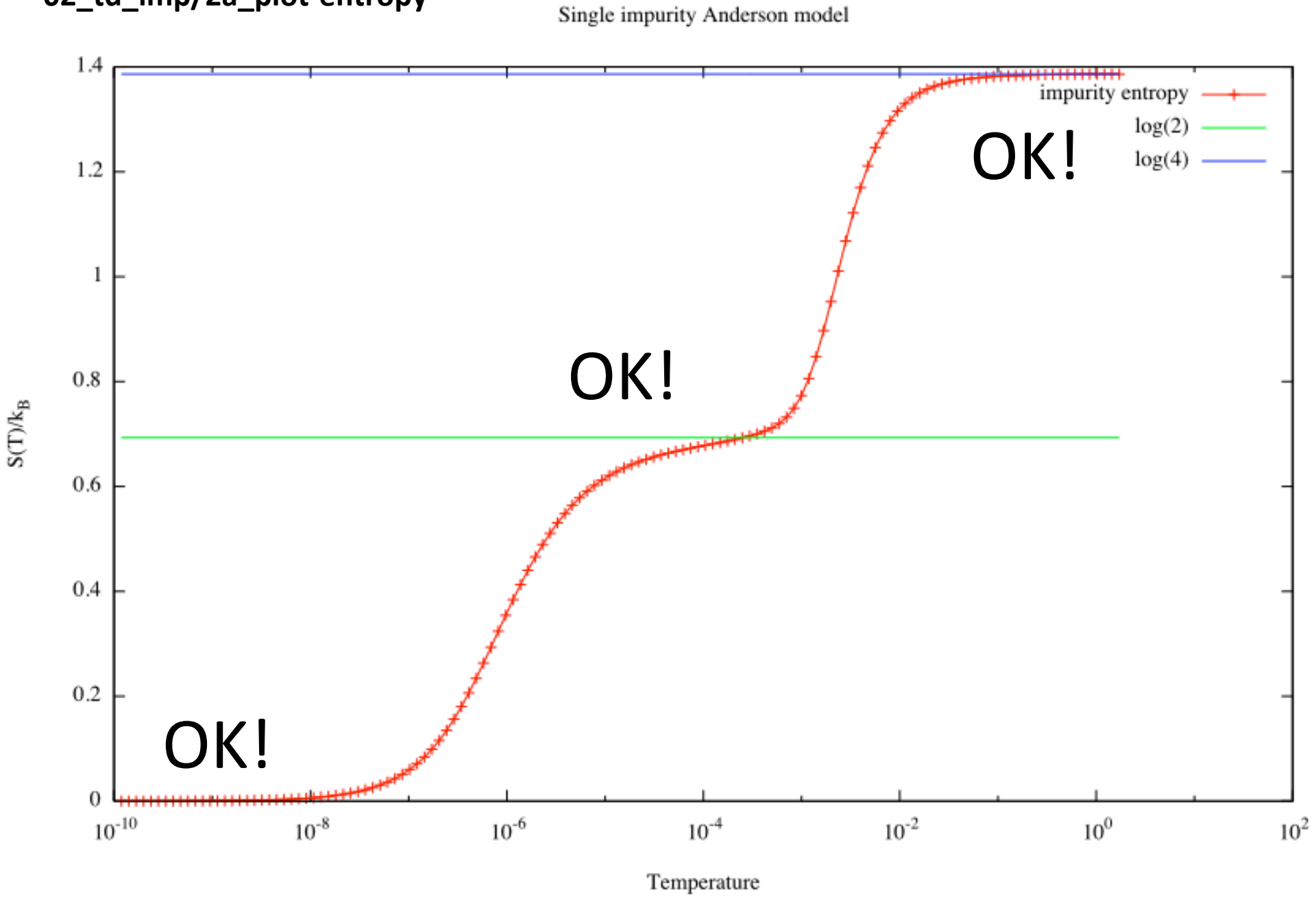
02_td/2_proc

```
#!/bin/sh

# Gather therodynamics results -> td.dat
gathertd_nosrt

# Average over z: td.dat -> td-avg.dat
tdavg -c
```

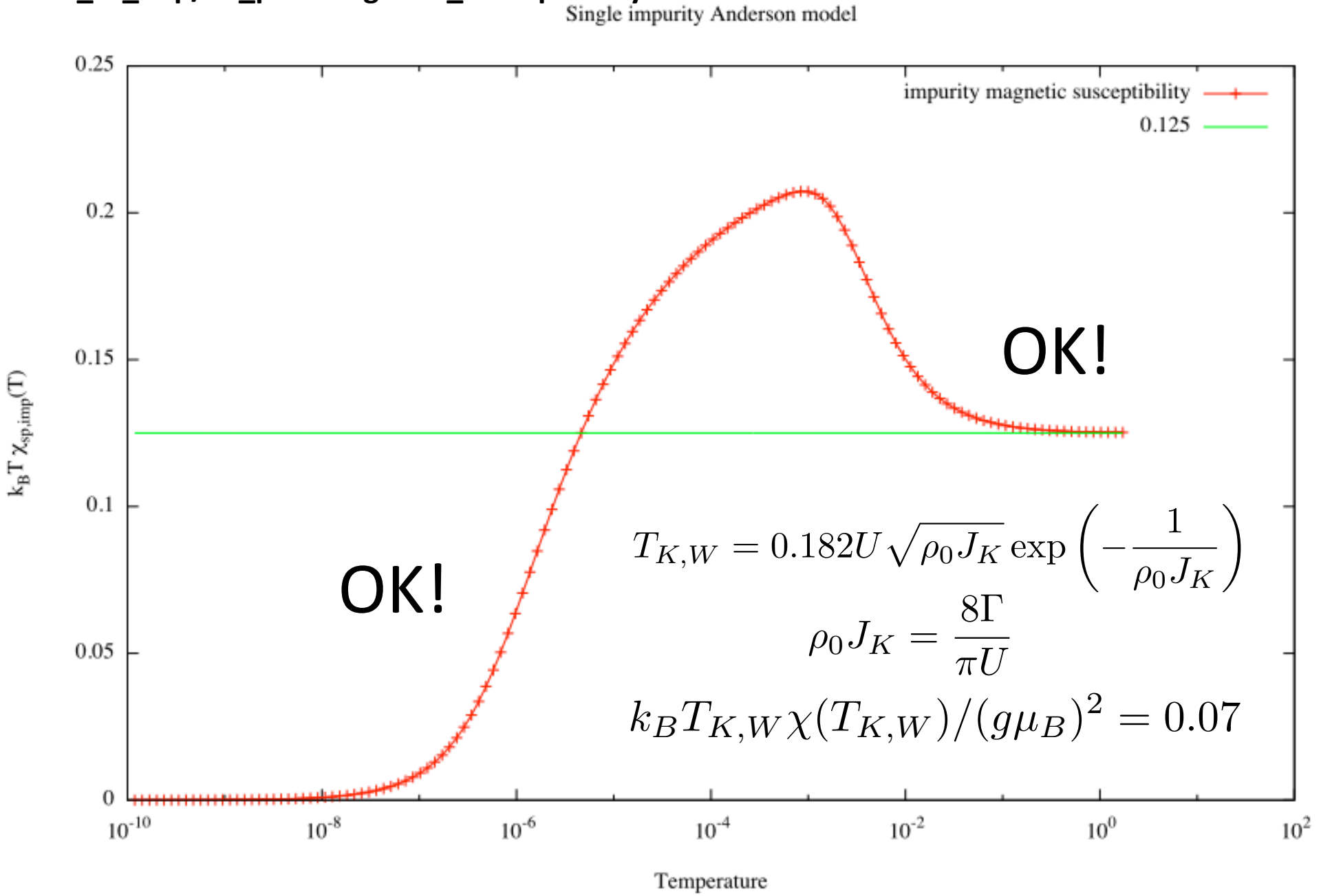
02_td_imp/1_proc

```
#!/bin/sh

# Copy the results and reference results
cp ../02_td/td.dat td.dat
cp ../02_td_0/td.dat td-ref.dat

# Average over z: td.dat & td-ref.dat -> td-avg.dat
tdavg -c

# Split according to the columns
report td-avg.dat td
```

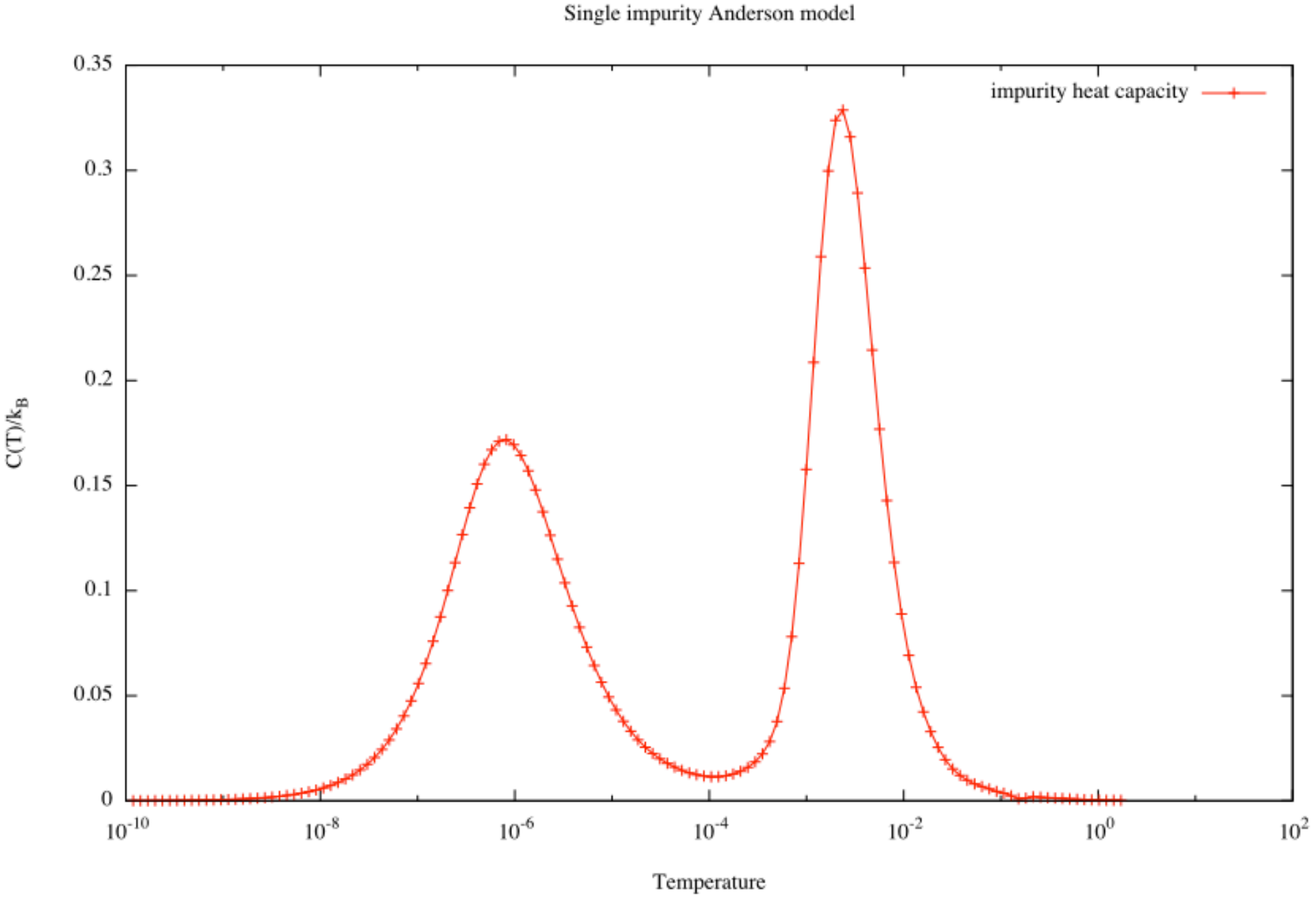
02_td_imp/2a_plot-entropy



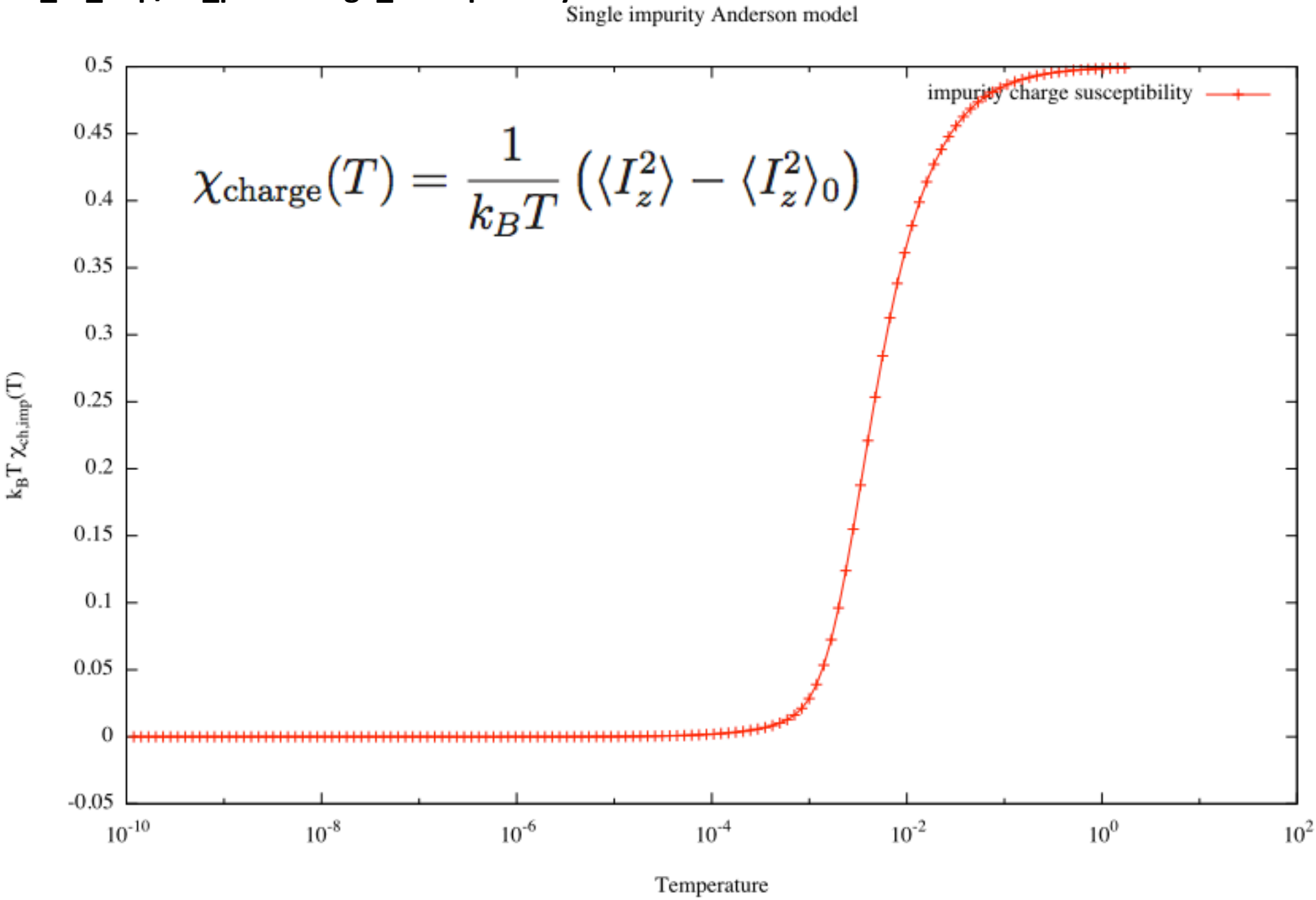
02_td_imp/2a_plot-magnetic_susceptibility



02_td_imp/2a_plot-heat_capacity



02_td_imp/2a_plot-charge_susceptibility



Exercises

- Try increasing the parameter $\delta=\varepsilon+U/2$. What happens when $\delta>U/2$? **1a1**
- Try increasing the parameter Γ. What happens when $\Gamma>U/\pi$? **1a2**
- Decrease systematically keepenergy from 10 to small values. How does the quality of the results deteriorate? **1a3**
- Increase Λ (to 4, then to 8). Try to do the *z*-averaging with a different number of *z* values. **1a4**

- Change the symmetry type from QS to QSZ. How much slower is the calculation? **1a5**
- Try redoing the calculations with different discretization schemes (Z, C, Y). How much do the results differ?
- Verify the formula for T_K, by performing the calculation for a range of Γ and plotting $\ln T_K$ vs. Γ. **1a2**

Expectation values

03_expv_siam/1_zloop

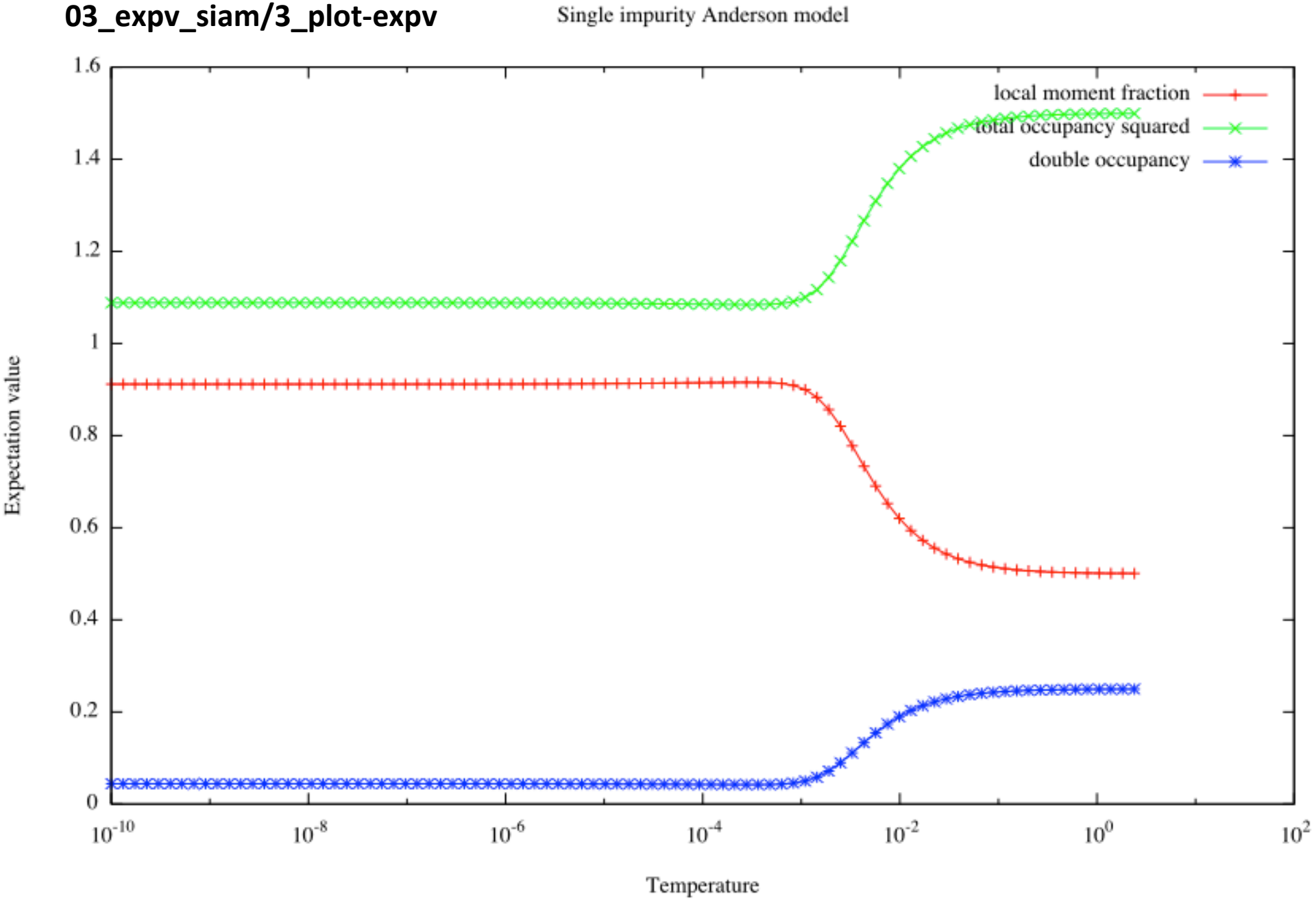
```
#!/usr/bin/env looper
#AUTOLOOP: nrginit ; nrgrun
#OVERWRITE

[param]
symtype=QS
discretization=Z
@$z = 1/4; $z <= 1; $z += 1/4
z=$z
# We may increase Lambda for this calculation
Lambda=3
Tmin=1e-10
keepenergy=10
keep=5000

model=SIAM
U=0.01
Gamma=0.0006
delta=0
```

```
ops=n_d n_d^2 n_d_ud flm hop0
```

Output: expectation values vs. T, **custom**

```
# $Id: nrg.cc,v 1.31 2012/10/26 10:18:22 rokzitko Exp rokzitko $ QS
# disk=Z band=flat Lambda=3 z=0.25
# keep=5000 keepenergy=10 Nmax=43 channels=1 dots=1 betabar=1
#           1             2             3             4             5             6
#           T           flm          hop0           n_d         n_d^2        n_d_ud
      2.39588      0.500522  -0.00815732             1       1.49948      0.249739
      1.38326      0.500904   -0.0139248             1        1.4991      0.249548
     0.798628      0.501565   -0.0234987             1       1.49844      0.249218
     0.461088      0.502712   -0.0380832             1       1.49729      0.248644
     0.266209      0.504689   -0.0574889             1       1.49531      0.247655
     0.153696      0.508127   -0.0790268             1       1.49187      0.245937
    0.0887365      0.514001    -0.100339             1         1.486         0.243
     0.051232      0.524112    -0.121695             1       1.47589      0.237944
    0.0295788      0.541309    -0.142702             1       1.45869      0.229346
    0.0170773      0.570167     -0.16338             1       1.42983      0.214916
   0.00985961      0.616671      -0.1828             1       1.38333      0.191665
   0.00569245      0.686505    -0.199825             1       1.31349      0.156747
   0.00328654       0.77532    -0.211715             1       1.22468       0.11234
   0.00189748      0.856776    -0.216771             1       1.14322     0.0716121
   0.00109551      0.901107    -0.216847             1       1.09889     0.0494463
  0.000632494       0.91364    -0.216375             1       1.08636       0.04318
  0.000365171      0.915606    -0.216489             1       1.08439     0.0421972
```

03_expv_siam/3_plot-expv



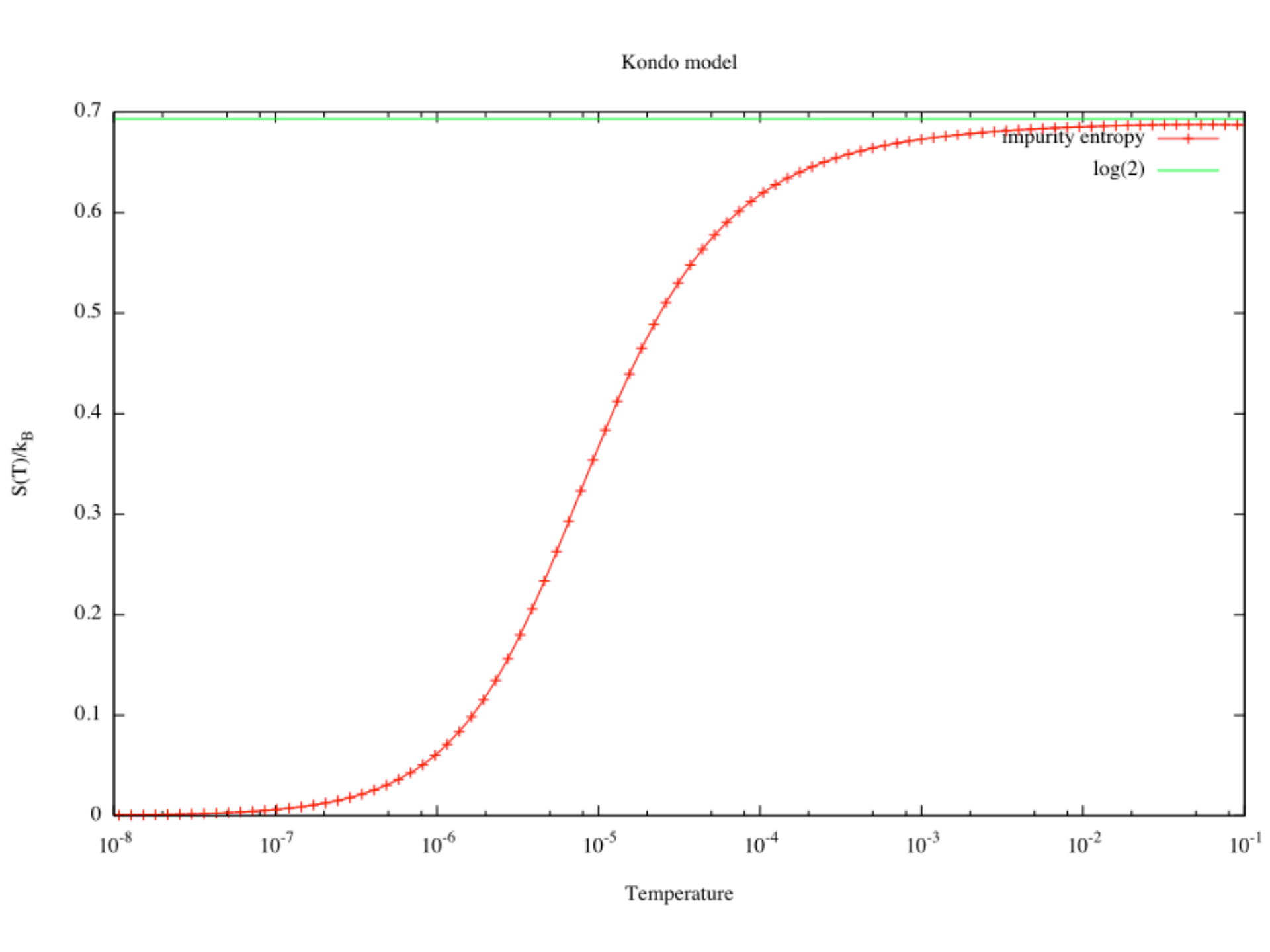
Kondo model

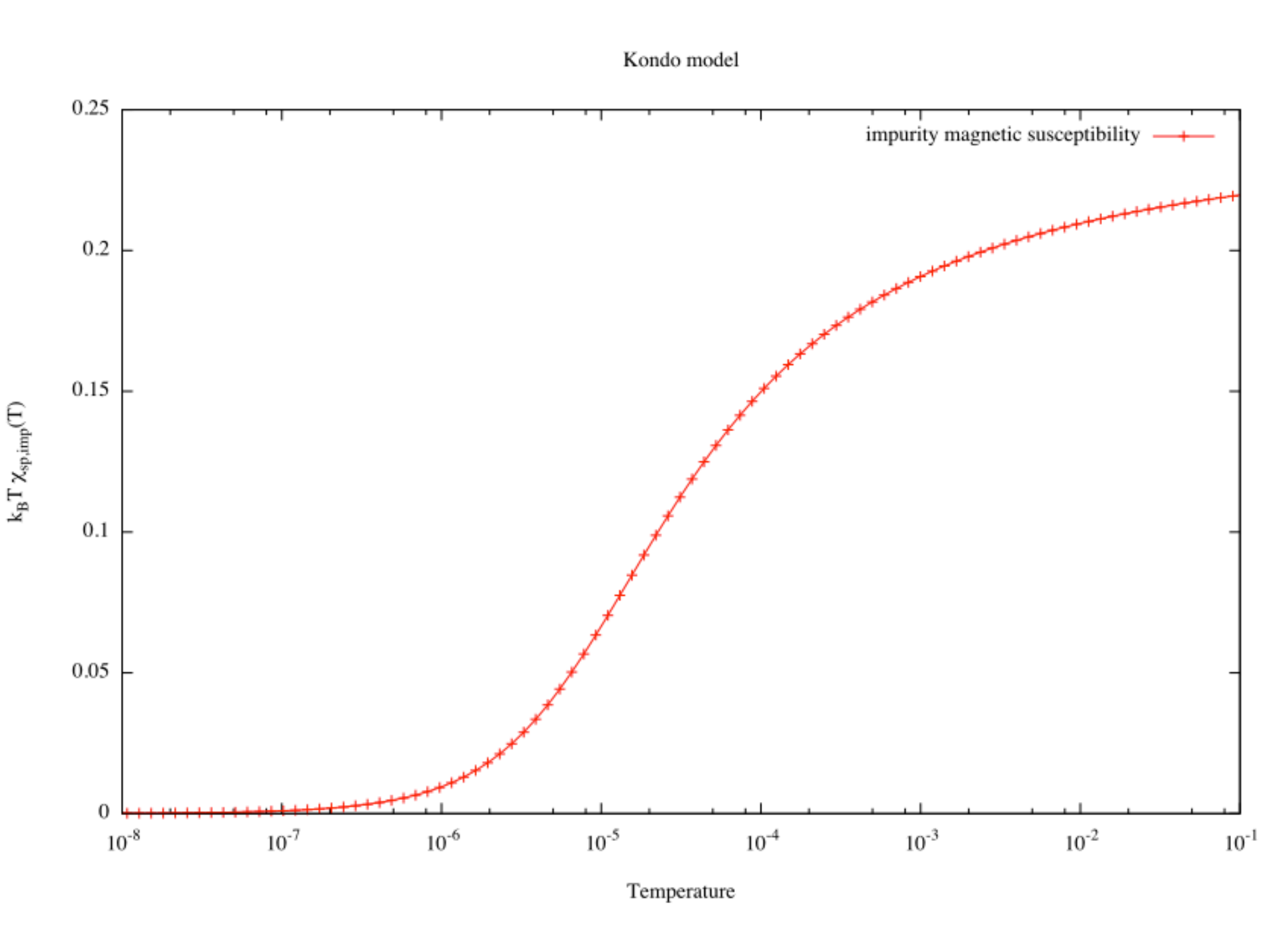
1_zloop

```
#!/usr/bin/env looper
#AUTOLOOP: nrginit ; nrgrun
#OVERWRITE

[extra]
spin=1/2
Jkondo=0.2

[param]
symtype=QS
discretization=Z
@$z = 1/4; $z <= 1; $z += 1/4
z=$z
Lambda=2
Tmin=1e-10
keepenergy=10
keep=5000

model=../kondo.m
```

kondo.m

```
def1ch[0];

SPIN = ToExpression @ param["spin", "extra"];

Module[{sx, sy, sz, ox, oy, oz, ss},
 sx = spinketbraX[SPIN];
 sy = spinketbraY[SPIN];
 sz = spinketbraZ[SPIN];

 ox = nc[ sx, spinx[ f[0] ] ];
 oy = nc[ sy, spiny[ f[0] ] ];
 oz = nc[ sz, spinz[ f[0] ] ];

 ss = Expand[ox + oy + oz];
 Hk = Jkondo ss;
];

H = H0 + Hk;

MAKESPINKET = SPIN;
```


Kondo model
impurity entropy
log(2)
S(T)/k_B
Temperature
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10^-8
10^-7
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1


Kondo model
impurity magnetic susceptibility
0.25
0.2
0.15
0.1
0.05
0
$k_B T \chi_{sp,imp}(T)$
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
Temperature

Running parameter sweeps

- Example: magnetization of the impurity spin (Kondo model) in magnetic field <Sz>(B) at T=0.

```
#!/usr/bin/env perl

open (O, ">magnetization.dat") or die;

for ($b = 1e-8 ; $b <= 1e-1; $b *= 1.5) {
   system "m4 -DFIELD=$b param.m4 >param";
   system "nrginit ; nrgrun";
   $SZ = `extractcolumn custom SZ | tail -n 1`;
   $SZ *= -1; # flip sign
   print O "$b $SZ\n";
}
```

03_expv_kondo_magnetization/

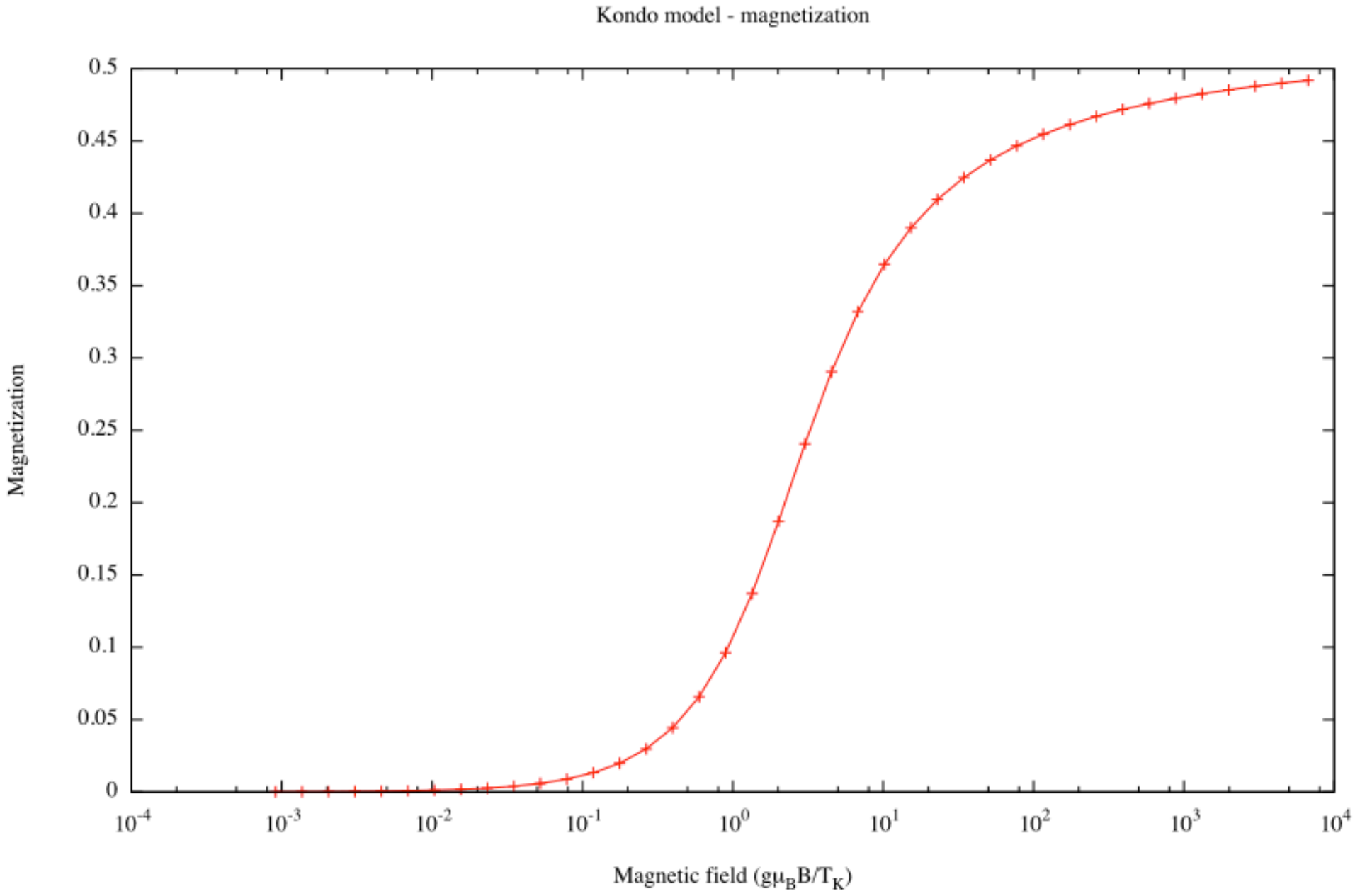
03_expv_kondo_magnetization/param.m4

template file, to be processed by the m4 macro processor

```
[extra]
spin=1/2
Jkondo=0.2
B=FIELD

[param]
symtype=QSZ
discretization=Z
Lambda=3
Tmin=1e-10
keepenergy=10
keep=5000

model=kondo.m

ops=SZ
```

In the presence of the magnetic field, only Sz is a good quantum number, i.e., SU(2) spin symmetry is reduced to a U(1) axial spin symmetry.

03_expv_kondo_magnetization/2_plot



Exercises

- For SIAM, plot n_d for a sweep of the parameter δ. Notice how the occupancy is pinned to 1 in a wide interval of δ. **1b1**
- For Kondo model, calculate the spin polarization SZ for a S=1 Kondo model for a range of magnetic field B going from negative to positive values. **1b2**

Renormalization group flow diagrams

```
[param]
symtype=QS
discretization=Z
Lambda=2
Tmin=1e-10
keepenergy=8
keep=5000

model=SIAM
U=0.01
Gamma=0.0006
delta=0

dumpannotated=100
```

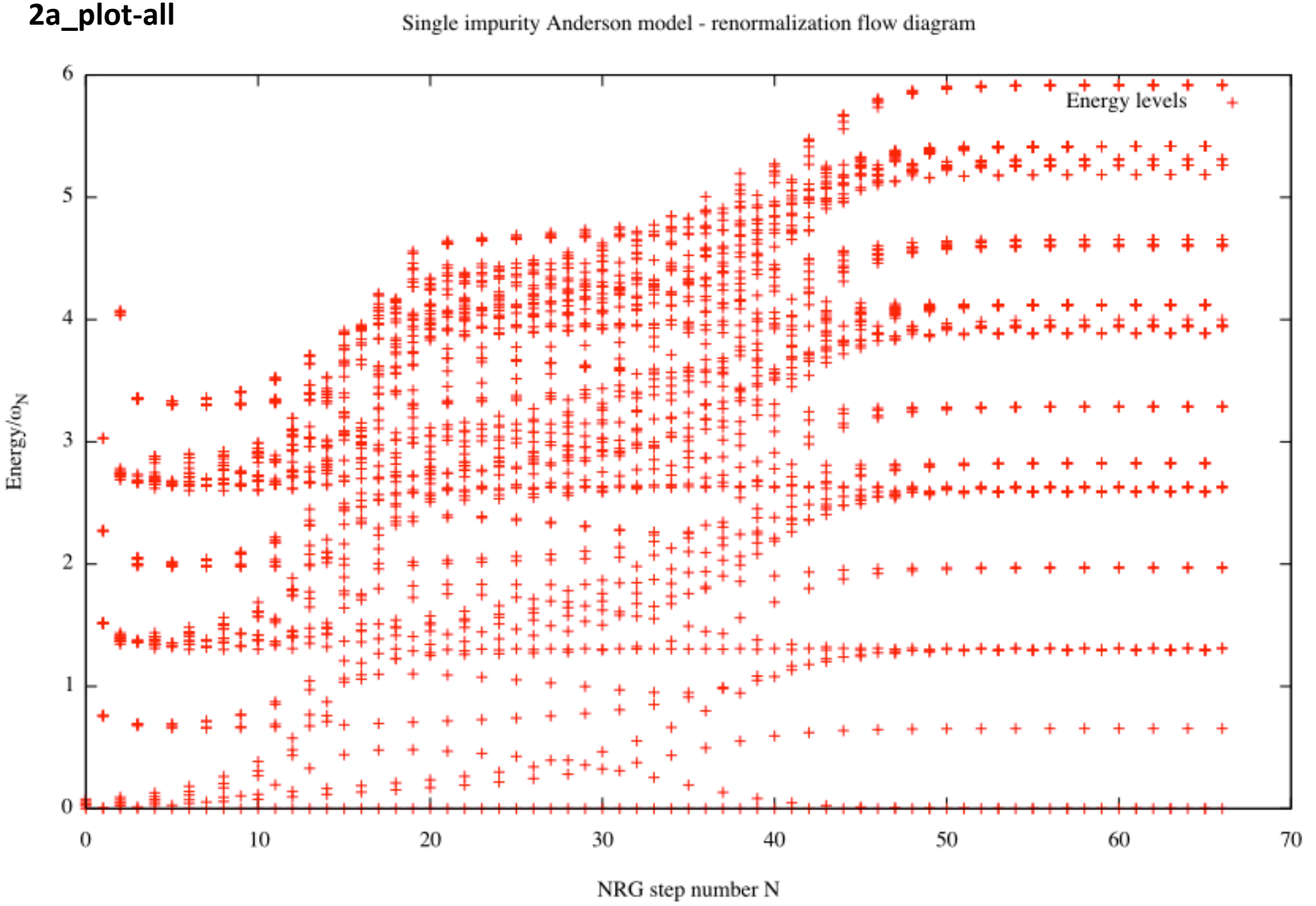
generates **annotated.dat**

04_flow_siam/

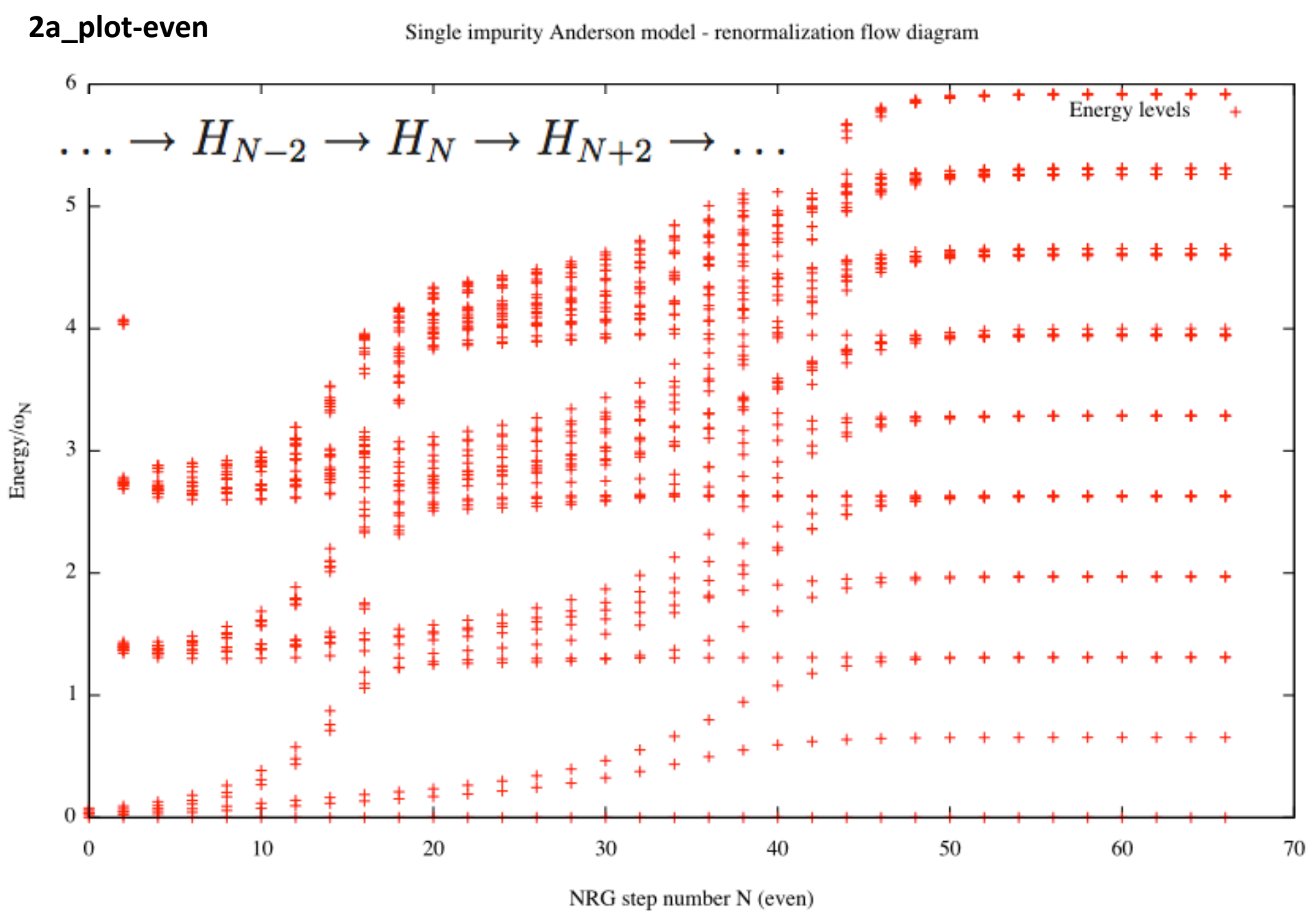
```
# $Id: nrg.cc,v 1.31 2012/10/26 10:18:22 rokzitko Exp rokzitko $ QS
# disk=Z band=flat Lambda=2 z=1
# keep=5000 keepenergy=8 Nmax=66 channels=1 dots=1 betabar=1
0 (0 1) [1]
0.019080451 (-1 2), (1 2) [4]
0.035944171 (0 3) [3]
0.040845462 (-2 1), (0 1), (2 1) [3]
0.057709182 (-1 2), (1 2) [4]
0.076789633 (0 1) [1]

0 (0 2) [2]
0.0069136957 (-1 1), (1 1) [2]
0.75640994 (-1 3), (1 3) [6]
0.7564277 (-1 1), (1 1) [2]
0.76331475 (0 2) [2]
0.76333251 (-2 2), (0 2), (2 2) [6]
1.5128199 (0 4) [4]
1.5128288 (-2 2), (0 2), (2 2) [6]
1.5128465 (0 2) [2]
1.5197247 (-1 1), (1 1) [2]
1.5197424 (-1 3), (1 3) [6]
1.5197513 (-1 1), (-3 1), (1 1), (3 1) [4]
2.2692387 (-1 3), (1 3) [6]
2.2692565 (-1 1), (1 1) [2]
2.2761435 (0 2) [2]
```

annotated.dat

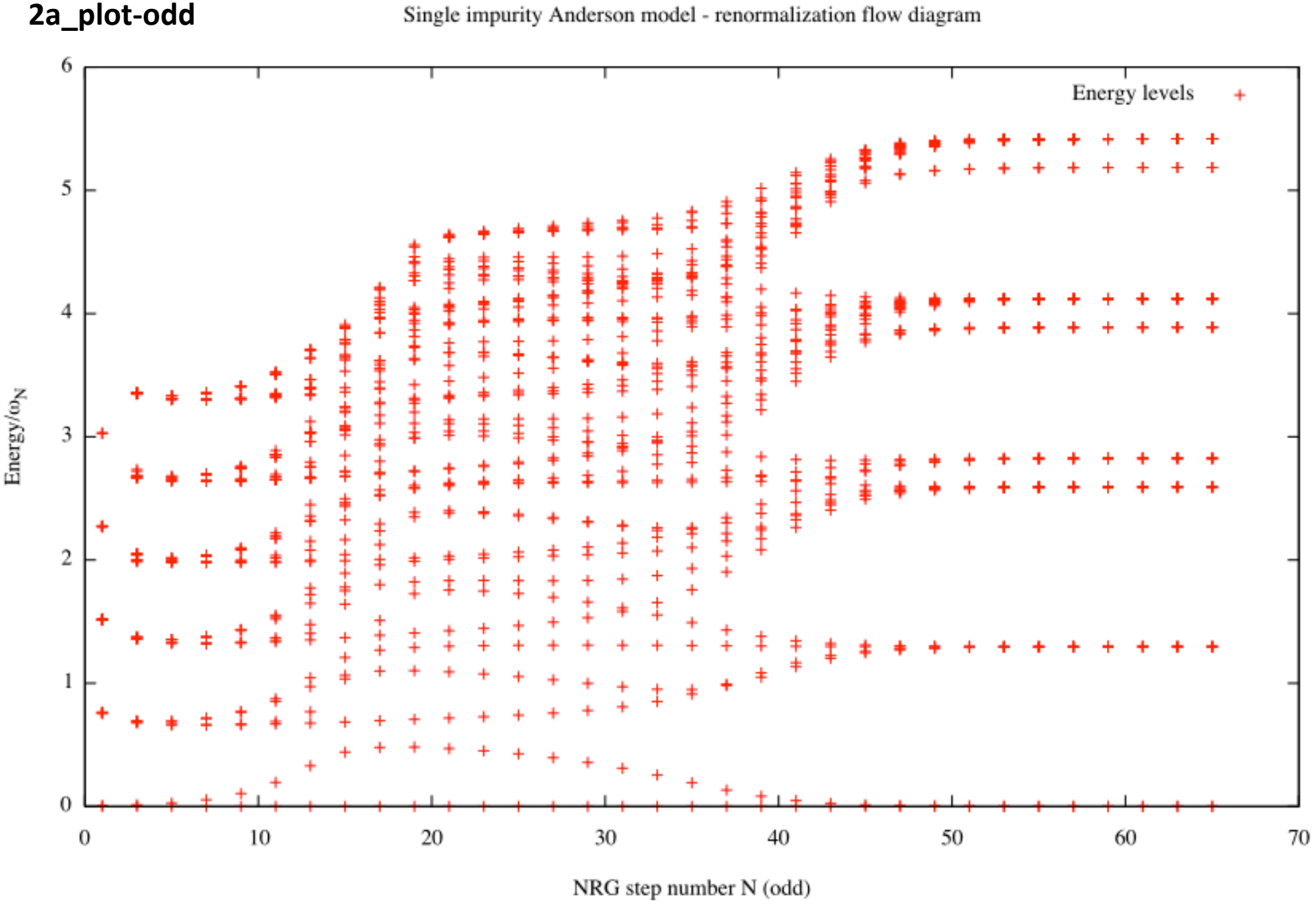
2a_plot-all


Single impurity Anderson model - renormalization flow diagram
Energy levels
Energy/ω_N
NRG step number N
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70

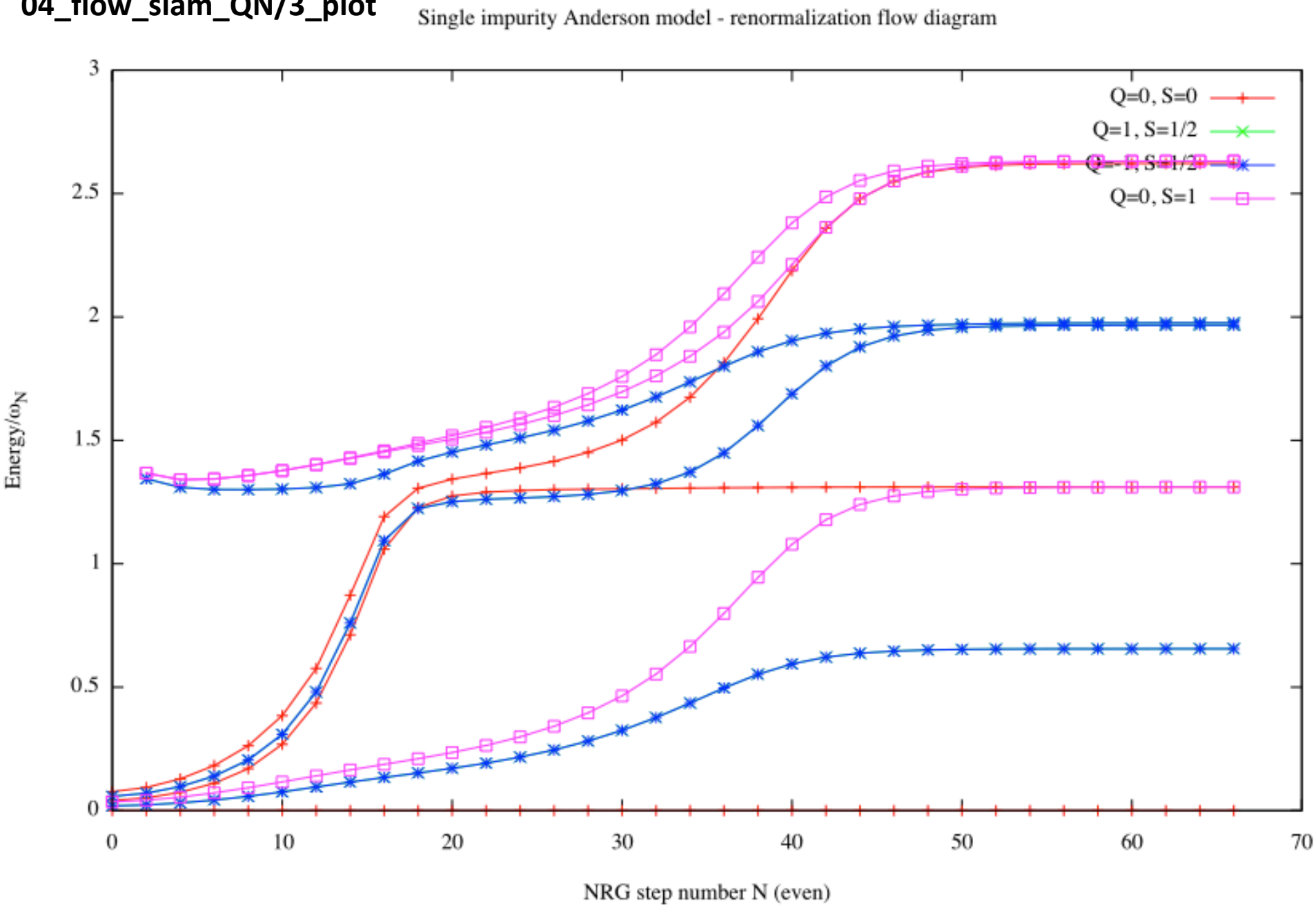
2a_plot-even


Single impurity Anderson model - renormalization flow diagram
$\ldots \rightarrow H_{N-2} \rightarrow H_N \rightarrow H_{N+2} \rightarrow \ldots$
Energy levels
Energy/ω_N
NRG step number N (even)
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70

2a_plot-odd


Single impurity Anderson model - renormalization flow diagram
Energy levels
Energy/ω_N
NRG step number N (odd)
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70

04_flow_siam_QN/3_plot



Exercises

- In SIAM, does the low-energy fixed point change if Γ is increased to large values? How is the behavior at intermediate energies affected?
- Increase δ to large values. How is the low-energy fixed point affected? Compare the flow diagrams for even and odd chain lengths.

1c1

Binding energy

$$E_{\mathrm{imp}} = E_{\mathrm{total}} - E_{\mathrm{total},0}$$

```
#!/usr/bin/env perl

my $Nz = 4;

sub getE
{
  my $dir = shift;
  my $sum = 0;
  for ($i = 1; $i <= $Nz; $i++) {
    $sum += `gettotalenergy $dir/$i/log`;
  }
  my $E = $sum/$Nz;
  return $E;
}

my $E1 = getE('../02_td_kondo');
my $E0 = getE('../02_td_0');

my $Eimp = $E1-$E0;

print "Impurity binding energy: $Eimp\n";
```

Kondo, ρJ=0.1

$$E_{\mathrm{imp}} = -0.0065436$$

$$T_{K,W} = 1.1 \times 10^{-5}$$

02_td_energy