Moderne numerische Methoden der Festkörperphysik
Lanczos-ED für die AF Heisenberg-Kette

Programmcode gen_matrix_Lanczos

Bedienung des Programms

Konvergenz des Lanczos-Verfahrens

Grundzustandsenergien der AF Heisenberg-Kette

Extrapolation zum thermodynamischen Limes

Programmcode gen_matrix_Lanczos

```
#define PROGNAME "gen_matrix_Lanczos"
#define VERSION "0.2"
#define DATE  "20.06.2007"
#define AUTHOR "Nils Bluemer"

/* creates tridiagonal matrix (for determination of eigenvalues)
    code based on gen_matrix4ED
    Version 0.1: only power method implemented
    NEW (0.2): conf is only created once */

#include <time.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "nrutil.c"
```

```
void error(char error_text[])
/* standard error handler */
{
  fprintf(stderr,"\n %s run-time error\n", PROGNAME);
    fprintf(stderr,"--%s--\n",error_text);
    fprintf(stderr,"for general help use option -h\n");
    fprintf(stderr,"...now exiting to system...\n");
    exit(1);
}



long comp_mag(long i, long N)
{
  long mag;
  mag=0;
  while (i>0){
    mag+=i%2;
    i/=2;
  }
  return(2*mag-N);
```

```
}

void gen_conf(int *conf, long i, long N)
{
  long n;
  for(n=1;n<=N;n++){
    conf[n]=i%2;
    i/=2;
  }
}

void erase(double *v, long size)
{
  long i;
  for (i=0;i<size;i++)
    v[i]=0.0;
}

void print_ivector(int *v, long size)
{
```

```
  long i;
  for (i=0;i<size;i++)
    printf("%6d",v[i]);
  printf("\n");
}

void print_dvector(double *v, long size)
{
  long i;
  for (i=0;i<size;i++)
    printf("%6lf",v[i]);
  printf("\n");
}

long E_Ising(int *v, long N, int periodic)
{
  long n, E;
  E=0;
  for (n=1;n<=N-1;n++)
    E+=(2*v[n]-1)*(2*v[n+1]-1);          /* sign in v0.3 corrected */
  E+=periodic*(2*v[N]-1)*(2*v[1]-1); /* sign in v0.3 corrected */
```

```
  return(E);
}

void printhelp ()
{
printf("*****************************************\n");
printf("%s:  prints matrix for 1D Heisenberg model\n",PROGNAME);
printf("Version:  %s, %s by %s\n",VERSION,DATE,AUTHOR);
printf("options:  -N# number of spins\n");
printf("          -m# mz sector (0<=m<=N)\n");
printf("          -o open boundary conditions\n");
printf("          -F ferromagnetic model\n");
printf("          -h this help\n");
}

void randomize_vector (double *v, long size, int *flag, int
verbosity)
{
  long n;
  double normsq, norm;
```

```
  /* Seed the random number generator */
  srand48(time(0) + getpid());
  normsq=0.0;

  for (n=0;n<size;n++)
    if (flag[n]>0){
      v[n]=drand48()-0.5;
      normsq+=v[n]*v[n];
    }
  norm=sqrt(normsq);
  for (n=0;n<size;n++)
    if (flag[n]>0){
      v[n]=v[n]/norm;
      if (verbosity>1) printf ("v[%d]=%lf\n",n,v[n]);
    }
}

void copy_vector (double *vold, double *vnew, int *flag, long size)
{
  int n;
```

```
  for (n=0;n<size;n++)
    if (flag[n]>0)
      vnew[n]=vold[n];
}

void matrix_vector (double *v, double *Hv, long size, long N, int
*flag, int periodic, int sign, int verbosity, int *conf)
{
  long i,n,base;

  if (verbosity>1)
    printf("N=%d\n",N);

  for (i=0;i<size;i++)
    if (flag[i]>0){
      gen_conf(conf,i,N);

      Hv[i]=E_Ising(conf,N,periodic)*v[i];   /* diagonal part of
hamiltonian */
      if (verbosity>1)
```

```
printf("v[%2d]= %lf\t",i,v[i]);
    }
  if (verbosity>1)
    printf("\n");

  for (i=0;i<size;i++)
    if (flag[i]>0){
      gen_conf(conf,i,N);

      base=1;
      for (n=1;n<=N-1;n++){
if (conf[n]!=conf[n+1])   /* spin flip possible */
   Hv[i + base*(2*conf[n]-1)]+=2*v[i];
base*=2;
      }
      if ((periodic>0)&&(conf[N]!=conf[1]))
Hv[i - (base-1)*(2*conf[n]-1)]+=2*v[i];
    }
}
```

```
int main (int argc, char *argv[])
{
  char c;
  long i, it, n, s, size, maxit, N;
  int mag, verbosity, periodic, sign;
  int *conf, *flag;
  double *vold, *vnew, *w, *alpha, *beta;


  verbosity=1;
  periodic=1;
  sign=1;
  maxit=100;

  while (--argc > 0 && (*++argv)[0] == '-')
    while (c= *++argv[0])
    switch (c) {
    case 'N':
       sscanf(++argv[0],"%d\n",&N);
              break;
    case 'm':
```

```
        sscanf(++argv[0],"%d\n",&mag);
              break;
    case 'o':
      periodic=0;
      break;
    case 'F':
      sign=-1;
      break;
    case 'v':
        sscanf(++argv[0],"%d\n",&verbosity);
              break;
    case 'h':
      printhelp();
      exit(0);
/*        default:   */
/*          error("No valid choice");   */
    }

  size=1;
  for (n=1;n<=N;n++)
```

```
  size*=2;

flag=ivector(0,size-1);
vnew=dvector(0,size-1);
vold=dvector(0,size-1);
w=dvector(0,size-1);

alpha=dvector(1,maxit);
beta=dvector(1,maxit);
conf=ivector(1,N);

/*****************************************/

for (i=0;i<size;i++)          /* mask n with wrong magnetization */
  if (comp_mag(i,N)==mag)
    flag[i]=1;
  else
    flag[i]=0;



randomize_vector(vnew,size,flag, verbosity);
```

```
  erase(vold,size);
  beta[1]=1.0;

  it=0;

/*    for (j=1;j<=10;j++){ */
  do {
    it++;
    matrix_vector(vnew,w,size,N,flag,periodic,sign,verbosity,conf);
    alpha[it]=0.0;
    for (i=0;i<size;i++)
      if (flag[i]>0){
w[i]-=beta[it]*vold[i];
alpha[it]+=w[i]*vnew[i];
      }
    beta[it+1]=0.0;
    for (i=0;i<size;i++)
      if (flag[i]>0){
w[i]=w[i]-alpha[it]*vnew[i];
beta[it+1]+=w[i]*w[i];
```

```
      }
    beta[it+1]=sqrt(beta[it+1]);
    for (i=0;i<size;i++)
      if (flag[i]>0){
vold[i]=vnew[i];
vnew[i]=w[i]/beta[it+1];
      }
/*   } while ((it<maxit)); */
  } while ((it<maxit)&&(fabs(beta[it+1])>0.00000001));

  printf("%d\n%lf  0.0\n",it,alpha[1]);
  for (s=2;s<=it;s++)
    printf("%lf %lf\n",alpha[s],beta[s]);

  /*****************************************/

  /*  free_ivector(flag,0,size-1);
  free_dvector(vnew,0,size-1);
  free_dvector(vold,0,size-1);
  free_dvector(w,0,size-1);
```

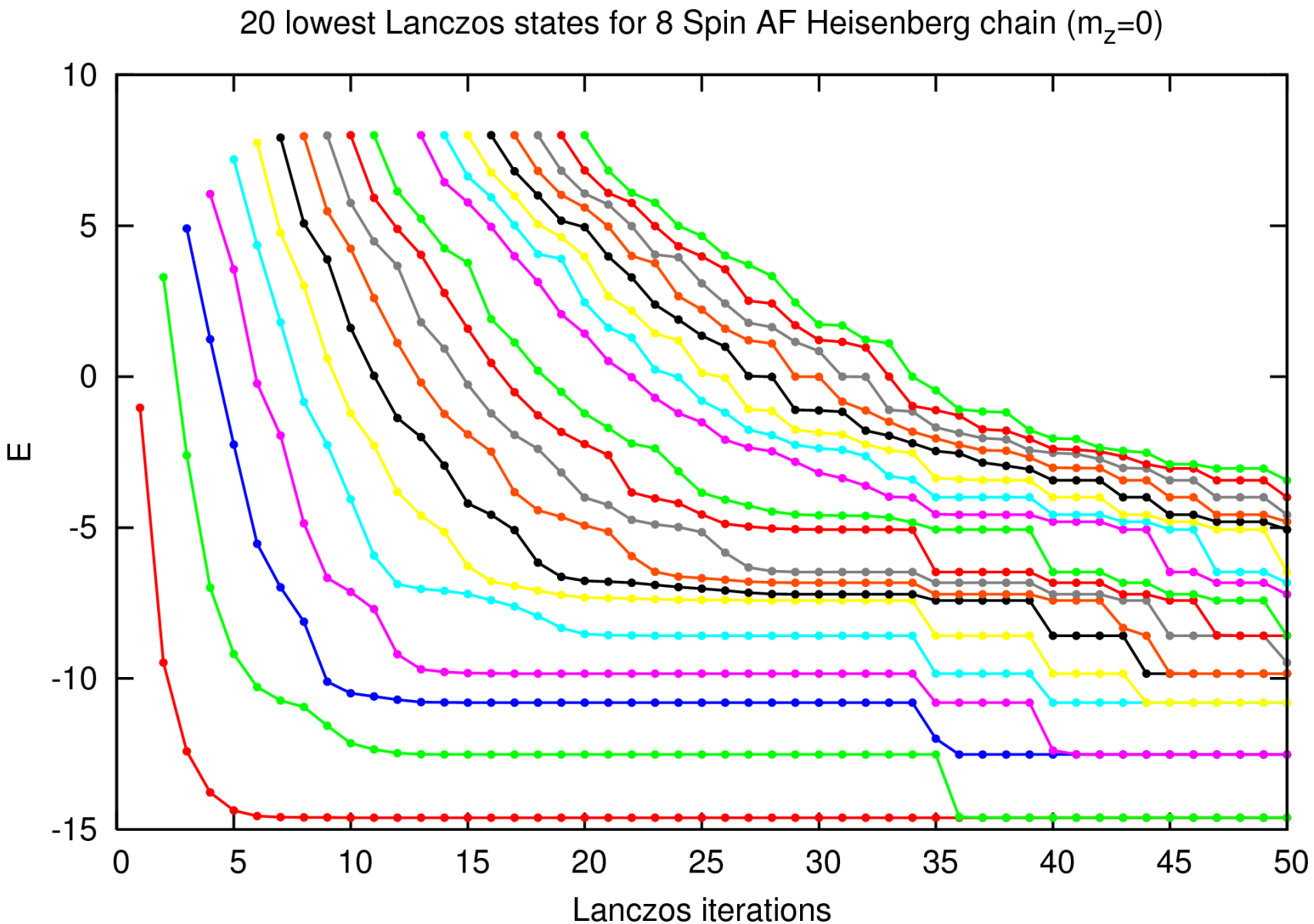
```
  free_dvector(alpha,1,maxit);
  free_dvector(beta,1,maxit);
  free_ivector(conf,1,N); */
}
```

Bedienung des Programms

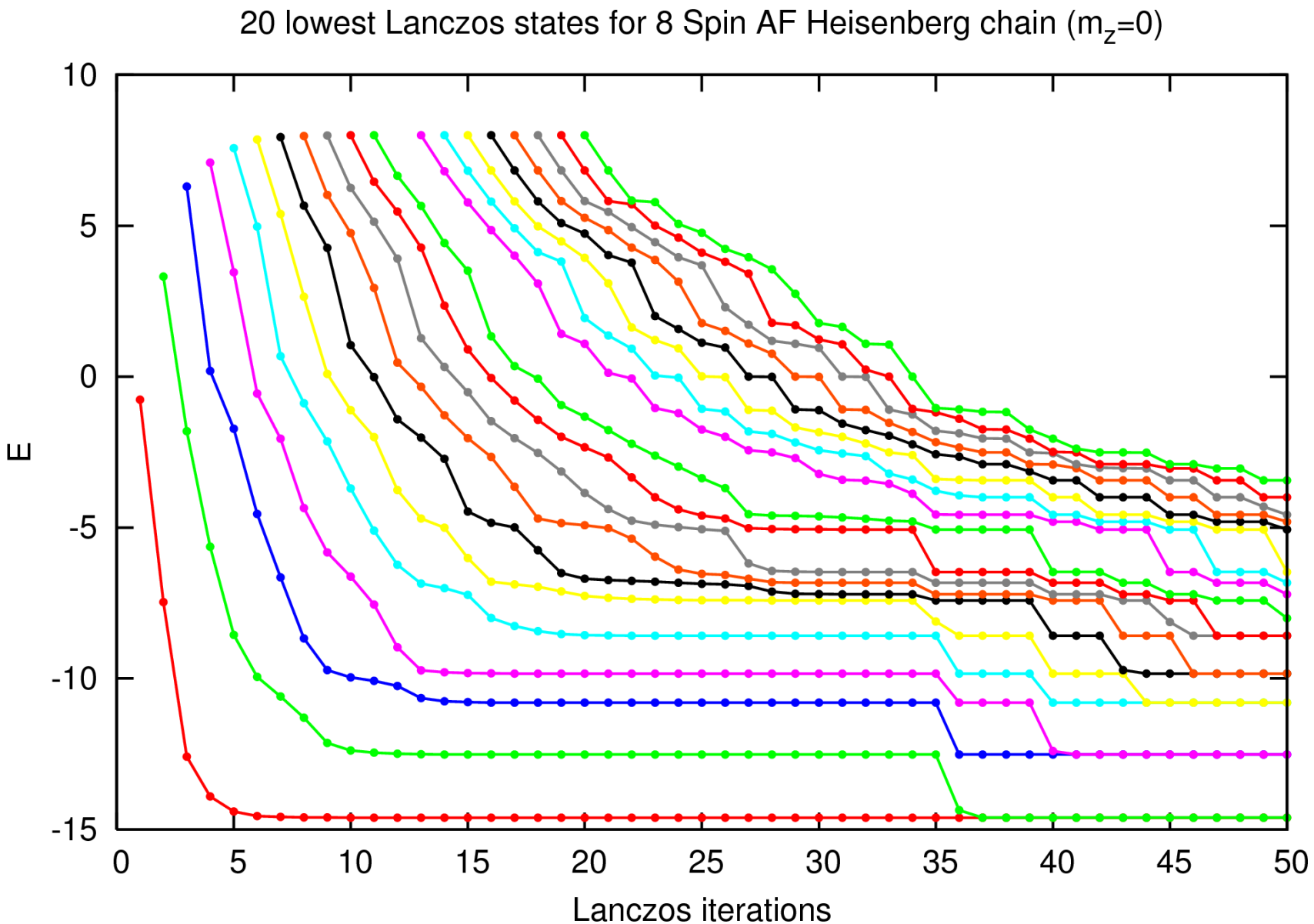
Bedienung analog zu vollem Matrix-Generator gen_matrix4ED; hier Ausgabe der Lanczos-Vektoren bei Verbosität 2 (-v2). Hilfeseite:

```
nils/NumMeth> ~/C/gen_matrix_Lanczos_v2 -h
*****************************************************
gen_matrix_Lanczos: prints matrix for 1D Heisenberg model
Version: 0.2, 20.06.2007 by Nils Bluemer
options: -N# number of spins
         -m# mz sector (0<=m<=N)
         -o open boundary conditions
         -F ferromagnetic model
         -h this help
```

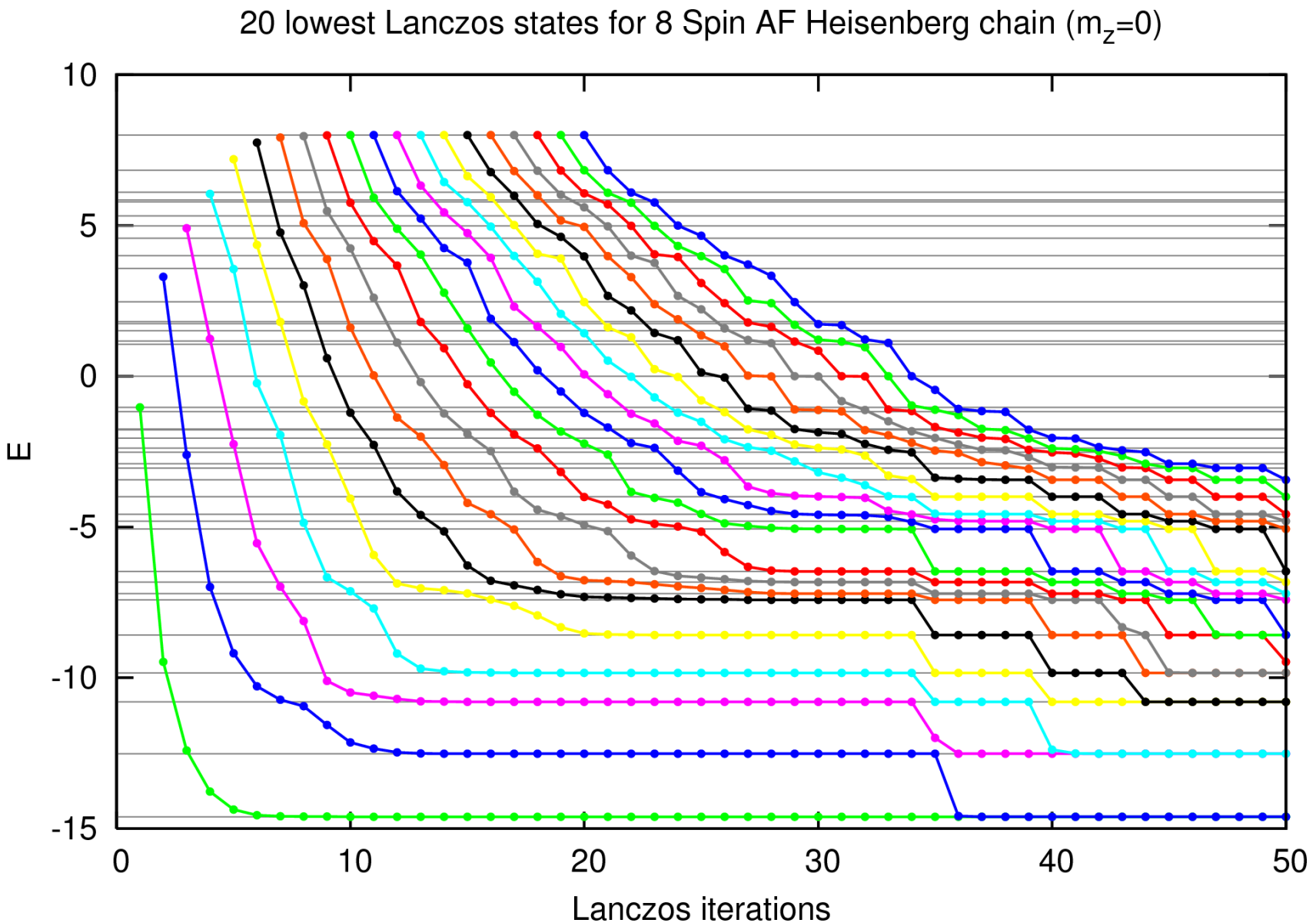
Konvergenz der Lanczos-Prozedur (8-Spin AF Heisenberg)

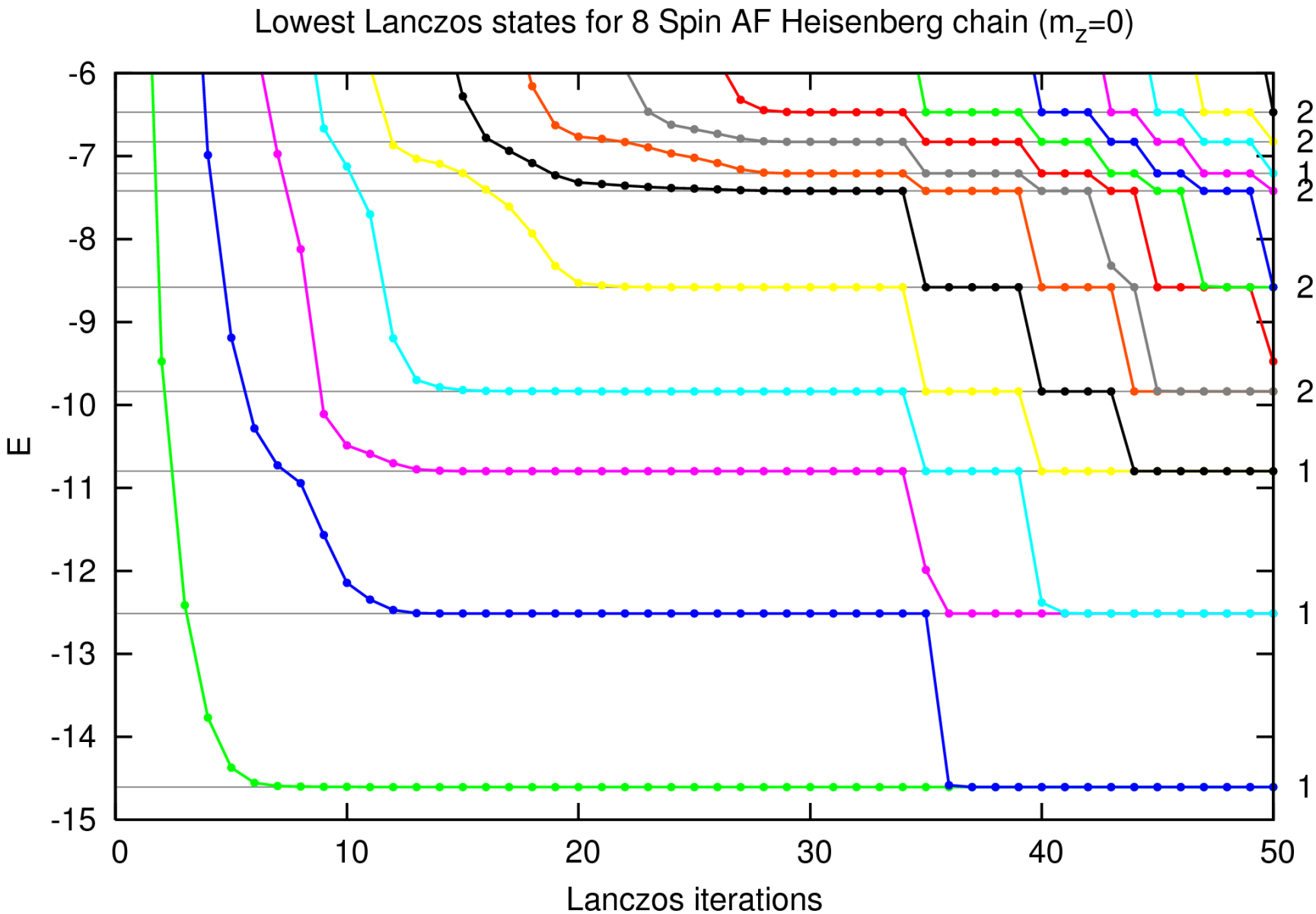


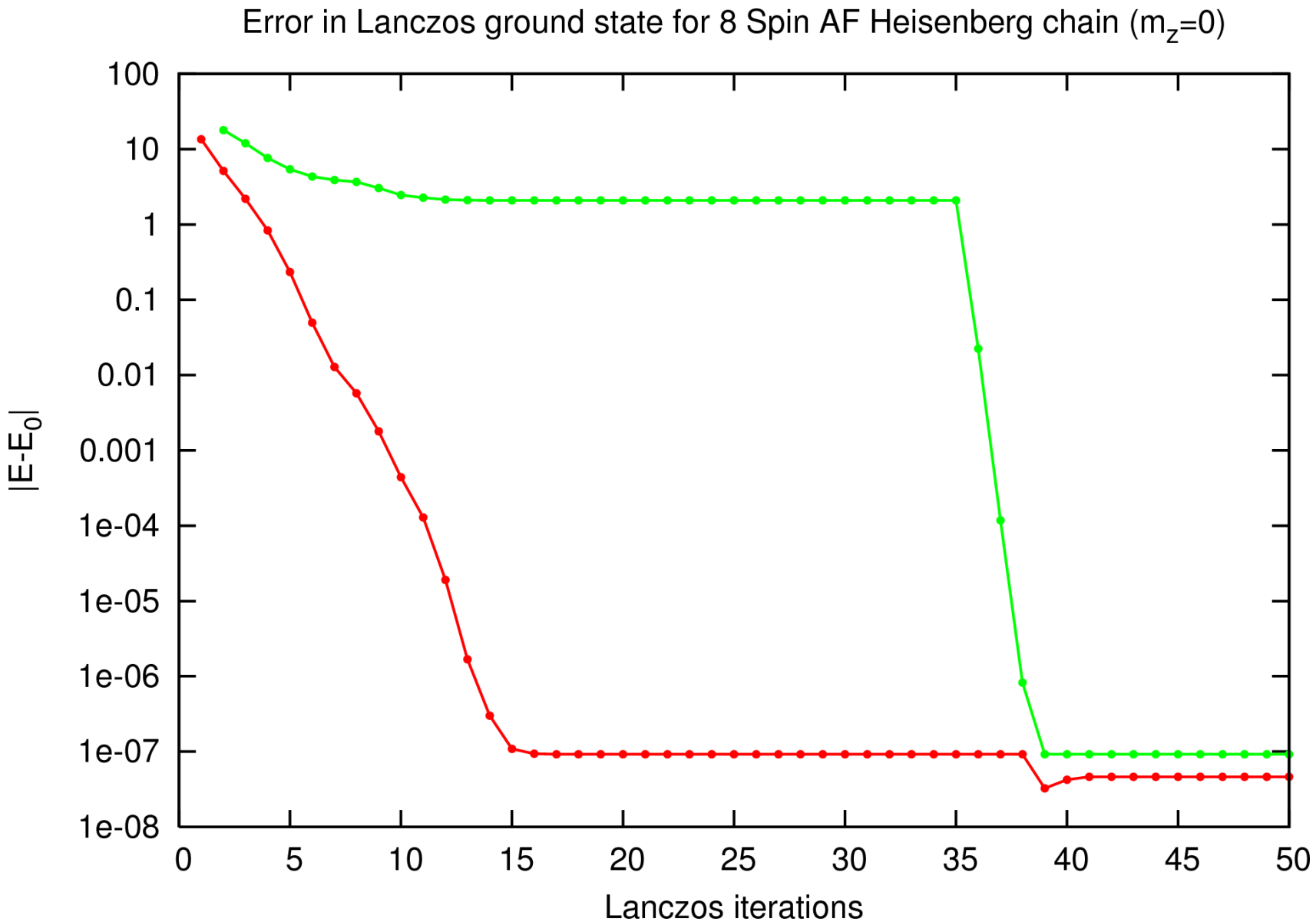
Initialisierung mit anderem (Zufalls-)Startvektor:

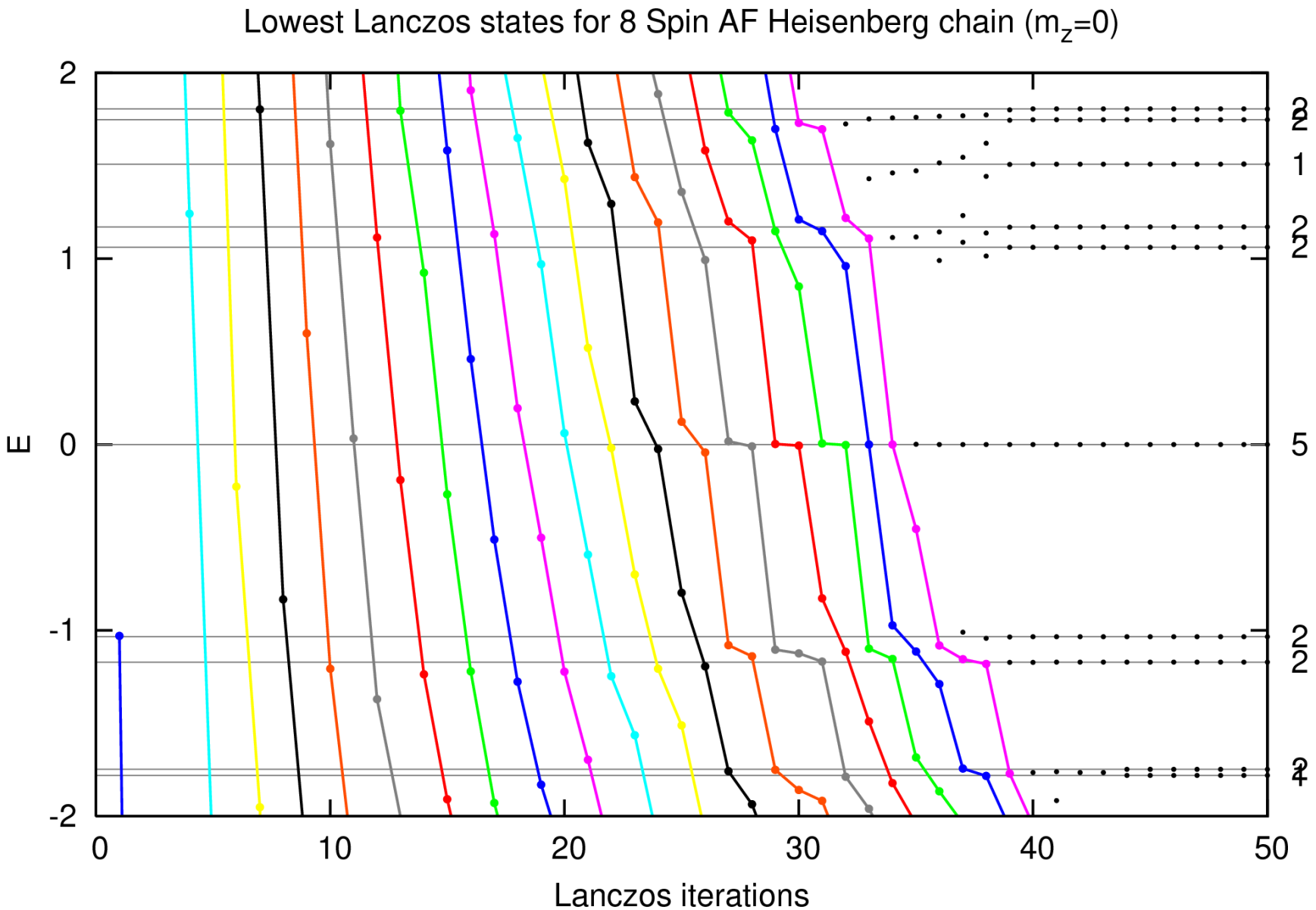


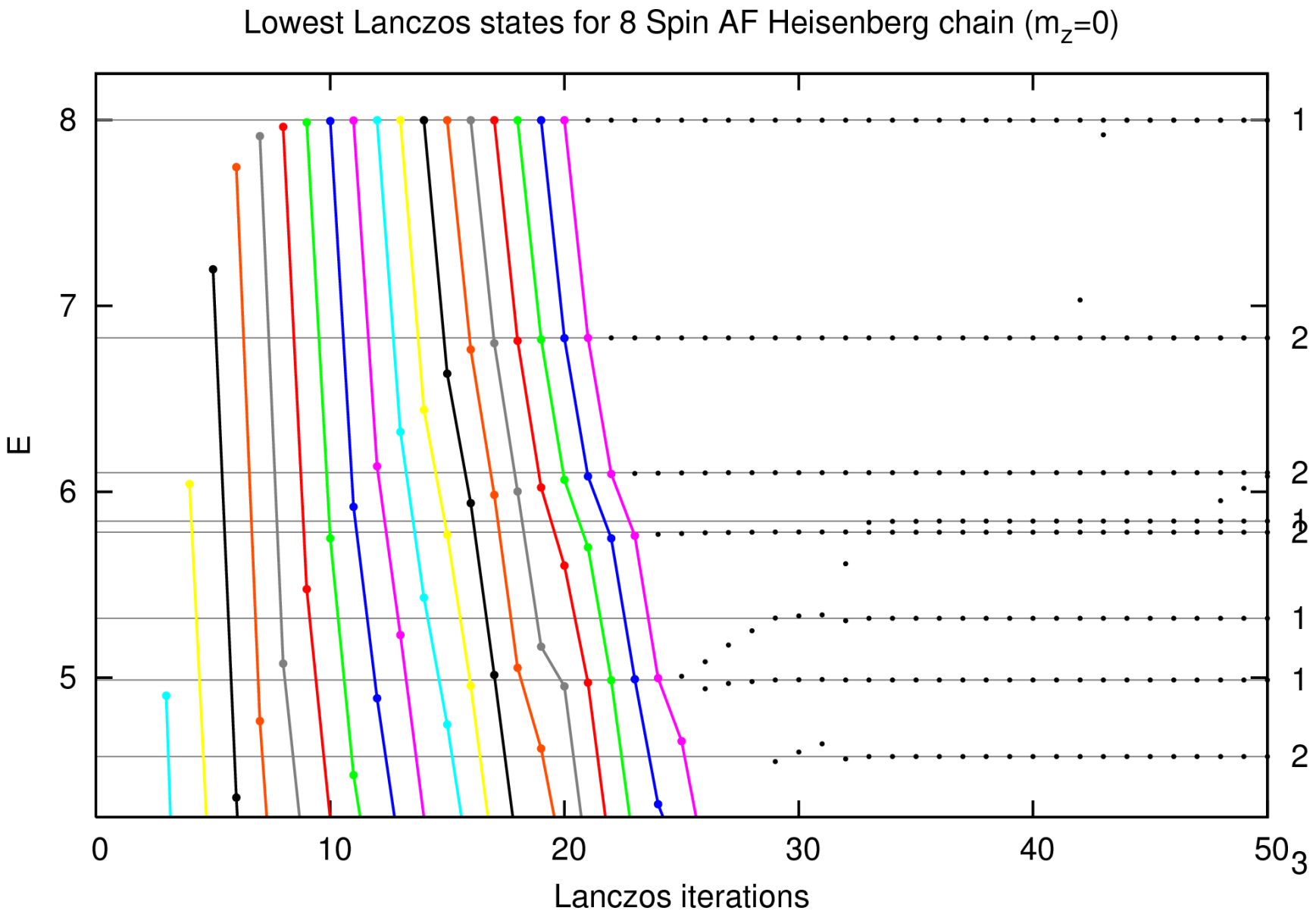
Vergleich mit exakten Eigenwerten




Lowest Lanczos states for 8 Spin AF Heisenberg chain (m_z=0)
E
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
0
10
20
30
40
50
Lanczos iterations
2
2
1
2
2
2
1
1
1


Error in Lanczos ground state for 8 Spin AF Heisenberg chain (m_z=0)
100
10
1
0.1
0.01
0.001
1e-04
1e-05
1e-06
1e-07
1e-08
$|E-E_0|$
0
5
10
15
20
25
30
35
40
45
50
Lanczos iterations


Lowest Lanczos states for 8 Spin AF Heisenberg chain (m_z=0)
E
Lanczos iterations
2
1
0
-1
-2
0
10
20
30
40
50
2
2
1
2
2
5
2
2
2
4

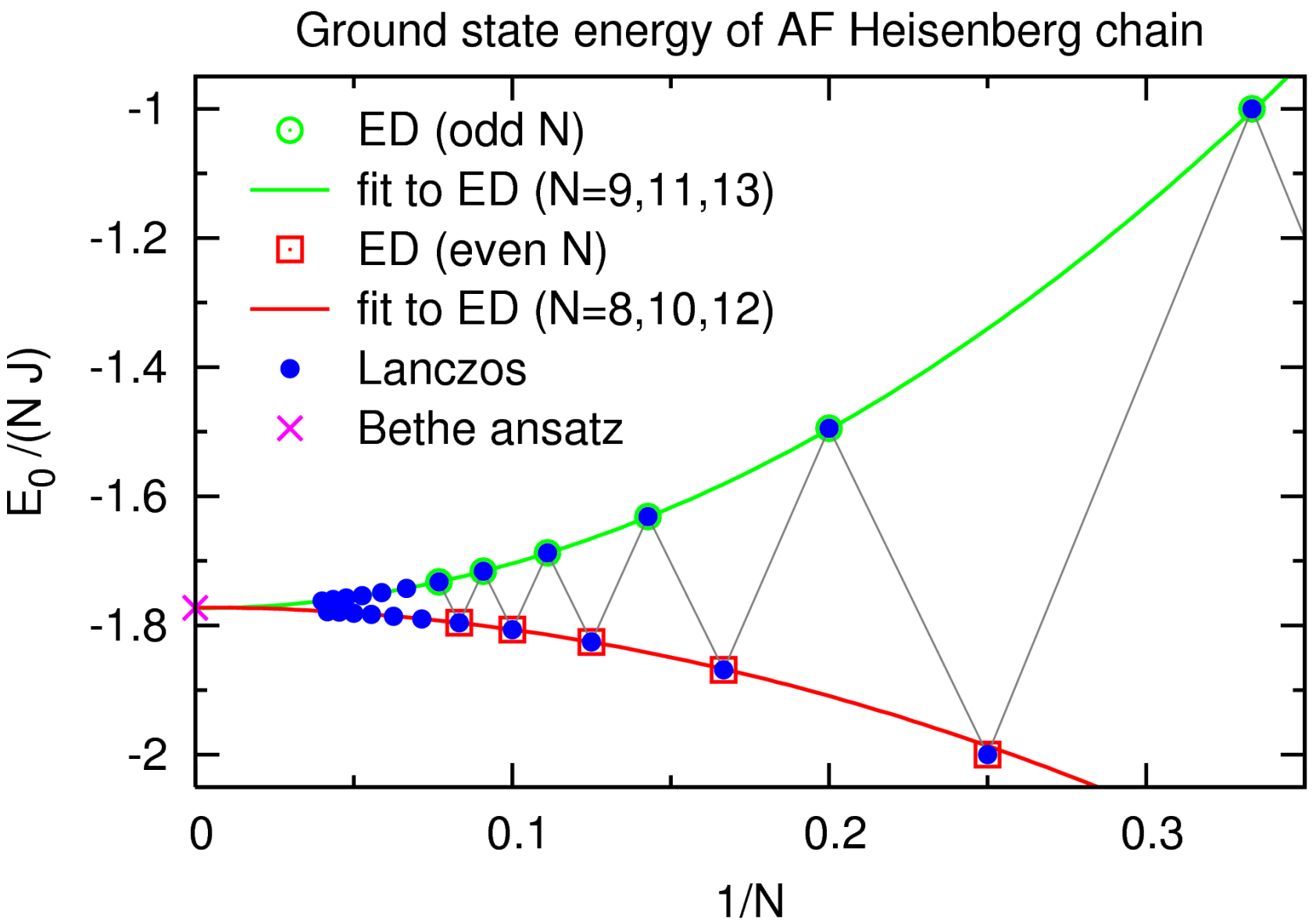

Lowest Lanczos states for 8 Spin AF Heisenberg chain (m_z=0)
E
Lanczos iterations
8
7
6
5
0
10
20
30
40
50
1
2
2
1
2
1
1
2
3

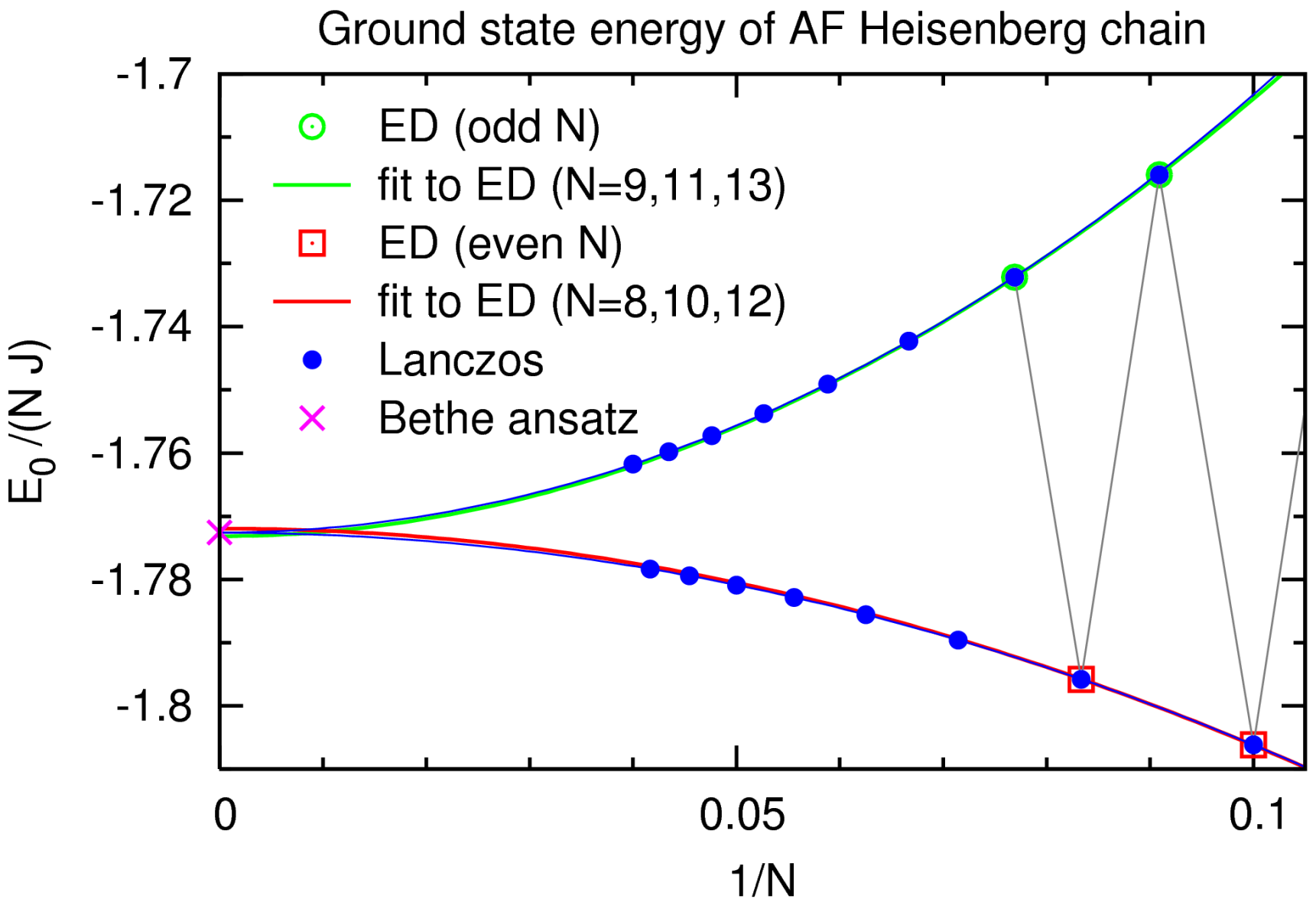
Berechnung der Grundzustandsenergie

Ergebnisse (AF Heisenberg, periodische Randbedingungen):

# N	E	E/N	14	-25.05419842	-1.78958560
2	-6.00000013	-3.00000006	15	-26.13467083	-1.74231139
3	-3.00000027	-1.00000009	16	-28.56918547	-1.78557409
4	-7.99999936	-1.99999984	17	-29.73408089	-1.74906358
5	-7.47213642	-1.49442728	18	-32.09099668	-1.78283315
6	-11.21110254	-1.86851709	19	-33.32195393	-1.75378705
7	-11.42071722	-1.63153103	20	-35.61754605	-1.78087730
8	-14.60437393	-1.82554674	21	-36.90161193	-1.75721962
9	-15.18919956	-1.68768884	22	-39.14752279	-1.77943285
10	-18.06178620	-1.80617862	23	-40.47521438	-1.75979193
11	-18.87574562	-1.71597687	24	-42.68005795	-1.77833575
12	-21.54956379	-1.79579698	25	-44.04422642	-1.76176906
13	-22.51833756	-1.73217981			

Extrapolation zum thermodynamischen Limes




Ground state energy of AF Heisenberg chain
ED (odd N)
fit to ED (N=9,11,13)
ED (even N)
fit to ED (N=8,10,12)
Lanczos
Bethe ansatz
-1.7
-1.72
-1.74
-1.76
-1.78
-1.8
E0/(N J)
0
0.05
0.1
1/N