Numerical errors in solving rate matrix

XXX XXX

February 15, 2016

Abstract

XXX XXX

1 LTE

Here we consider a single point in the homogeneous atmosphere, the temperature is 5000 K and total gas pressure is $10^{-1}\,\mathrm{dyn\,cm^{-1}}$. We consider 2 level + continuum hydrogen atom, where levels correspond to the second and third levels of hydrogen. The relative difference between solving rate equations and computing "analytical" LTE populations is of the order of 10^{-9}.

We want to compute response of populations with respect to perturbation of 1 K in Temperature. Let us compare 5 approaches and try to decipher what is going on from numerical point of view:

1. We use finite differences. Perturb, solve for populations, perturb, solve for populations, subtract and divide by the step ($\Delta T = 1\,\mathrm{K}$).

2. Do the same, but subtract populations "manually", i.e. print, read into numpy, subtract, and see what is going on.

3. Export the appropriate rate matrices and then solve for populations in python, subtract.

4. After solving for populations, we compute the response of the collisional rates and solve for the population responses, "on-the-spot", using LU decomposition.

5. We "pack" the populations and rate derivatives in the matrix for the whole atmosphere (as we would do for NLTE responses). This way we get the block diagonal matrix which we solve using LU decomposition to get population derivatives for all levels at all points in the atmosphere, simultaneously.

Approach 1 results in the following responses:

3.832990967405721e-01 1.277248255751262e-02 8.376627229684353e+06

For the approaches 2 and 3, we first show all the rate equations:
Rate matrix and the right hand side for $T + \Delta T/2$:

-6.069824894881946E+03 5.075663002170855E+02 1.991794151327135E-04 0.000000000000000E+00
1.425732423872086E+01 -4.052259836767817E+05 3.739278507386271E-04 0.000000000000000E+00
1.000000000000000E+00 1.000000000000000E+00 1.000000000000000E+00 2.488868629353189E+09

Populations resulting from the code:
8.186373537905747e+01 2.299517951039844e+00 2.488868545189936e+09

Rate matrix and the right hand side for $T - \Delta T/2$:

-6.039117330371747E+03 5.059097629240367E+02 1.979095454520364E-04 0.000000000000000E+00
1.419833901992229E+01 -4.035291506541349E+05 3.715438712490932E-04 0.000000000000000E+00
1.000000000000000E+00 1.000000000000000E+00 1.000000000000000E+00 2.480492001727433E+09

Populations resulting from the code:
8.148043628231690e+01 2.286745468482332e+00 2.480491917960251e+09

Manually subtracting these two populations results in:
3.832990967405721e-01 1.277248255751218e-02 8.376627229685307e+06

which shows slight differences with what we get using finite differences, already! What is weird is that for the first response we get identical results. Responds for second and third level differ in last 2 and 3 digits, respectively, but taking into account that numbers are in double precision and appropriate exponents, we can explain that. However, why is the first response identical?

The next step is to solve rate matrix outside of our code (i.e. using numpy linear algebra package, which is supposed to be based on lapack, and works in double precision). Then be subtract obtained populations. We get:
3.832990967405721e-01 1.277248255751173e-02 8.376627229684353e+06

Which is, again, slightly different.But again the response for the first level is identical(!). The differences in second two responses are now one order of magnitude higher which can be explained with additional operations in the solution of the linear system.
Results for approaches 3 and 4 are:
3.832993111681434e-01 1.277248754582499e-02 8.376627229683989e+06

and:

3.832993111681434e-01 1.277248754582499e-02 8.376627229683989e+06
These are identical. Now, once again, all 5 solutions beneath each other, for the comparison:

3.832990967405721e-01 1.277248255751262e-02 8.376627229684353e+06
3.832990967405721e-01 1.277248255751218e-02 8.376627229685307e+06
3.832990967405721e-01 1.277248255751173e-02 8.376627229684353e+06
3.832993111681434e-01 1.277248754582499e-02 8.376627229683989e+06
3.832993111681434e-01 1.277248754582499e-02 8.376627229683989e+06

Perhaps computing svd for the rate matrix would be beneficial. Even for such a

simple problem, span of singular values is rather high, condition number is over 10^5!:
4.052263021440419e+05
6.069802348995461e+03
1.000000020240077e+00