# Total energy and forces in PW

Nicola Manini
*ETSF and Dipartimento di Fisica, Università degli Studi di Milano, Via Celoria 16, 20133 Milano, Italy*
(Dated: May 13, 2011)

PW computes extremely accurate forces. Total energies suffer from some nonrandom noise of unknown nature. The "damp" minimization algorithm has a bug showing near minima.

These notes follow a discussion with Paolo Giannozzi (partly reported verbatim).

```
On 05/11/2011 08:46 PM, Paolo Giannozzi wrote:
>
> On May 9, 2011, at 16:33 , Nicola Manini wrote:
>
>> At Paolo's request, I attach the input file of a typical example
>> of problematic damp minimization
>
> I don't see any problem here: the system is already minimized.
> Really: forces are very small and energy is almost constant.
> of hidden and forgotten approximations, will conspire to make
> any minimization algorithm unusable. I don't know what the
> origin of the above "oscillations" in the energy is: maybe the
> discreteness of the FFT. In any event, I would say that there
> is nothing wrong in the damped dynamics (or in bfgs) itself:
> there is some numerical noise in forces and energy calculation,
> somewhere, God knows where, and whether it can be removed.
```

Those energy fluctuations do look small. I am aware of all these problems you are mentioning. Specifically, the lack of translational invariance makes the total energy depend on the specific place one puts atoms at. As one FFT grid is used for the calculation of energy and force, if forces are the derivatives of total energy, it just means that the minimum of a free minimization will make the molecule/crystal drift to some "absolute" location in space. Also a constrained minimization will lead to different molecular/crystal shapes depending on the absolute place where one puts the constraints. This may indeed be a problem for the physics one wishes to describe. Mathematically however if forces are consistent with the total adiabatic energy, they should lead to a consistent local minimum of energy (accounting for the specific translational non-invariance).

The nice thing is that indeed they often do! In examples similar to the one discussed above (same number & type of atoms) one obtains convergence to a force smaller than $2*10^{-6}$ Ry/$a_0$ and of all the 8 printed decimal digits of energy (and likely a few more), no noise. I attach an example (`test_ok.in test_ok.out`) of such a successful minimization. How is this possible?

```
> By the way: some years ago the energy was printed with 6
> decimals. Increasing the number of decimals to 8 was in
> my opinion a very bad idea: it gives the illusion that those
> last figures are "significant", but in fact they change with the
> phase of the moon.
```

I disagree. Indeed I prefer pw to the similar code `abinit` precisely because of its far better numerical stability. Such moon-phase fluctuation of small digits is more typical of `abinit` than of pw! In `abinit` one is happy to reach milliRy accuracy. In most pw calculations you can count on getting the microRy and often even the 2 following digits to be reliable & reproducible (at least if atoms are few enough that total energy does not exceed thousands of Ry, and in nonpathological situations!). By numerical stability I mean that a tiny atomic displacement produces a tiny perturbation to the output quantities, not a random jump. If you double the perturbation you double the change (unless that quantity is stationary, of course!).

So the question is: why in some minimization pw seems extremely accurate, while in others one gets those wierd fluctuations? Setting the phase of the moon aside, I think the problem can be attributed to one or more of:

- inaccuracy of the total energy
- inaccuracy of forces

| # | $x$(Å) | $f$(Ry/bohr) | $E$ | $x$(bohr) |
|---|---|---|---|---|
| 1 | 0.274245261 | 0.00093865 | -138.3052420377 | 0.5182484365317439 |
| 10 | 0.280245261 | 0.00037589 | -138.3052578645 | 0.5295867933290578 |
| 8 | 0.283645261 | 0.00005056 | -138.3052611465 | 0.5360118621808689 |
| 7 | 0.283945261 | 0.00002175 | -138.3052611768 | 0.5365787800207347 |
| 4 | 0.284045261 | 0.00001204 | -138.3052611796 | 0.5367677526340232 |
| 3 | 0.284145261 | 0.00000236 | -138.3052616147 | 0.5369567252473118 |
| 0 | 0.284245261 | -0.00000730 | -138.3052616101 | 0.5371456978606004 |
| 2 | 0.284345261 | -0.00001695 | -138.3052616020 | 0.5373346704738889 |
| 5 | 0.284445261 | -0.00002661 | -138.3052615901 | 0.5375236430871775 |
| 9 | 0.284845261 | -0.00006535 | -138.3052610701 | 0.5382795335403318 |
| 11 | 0.288245261 | -0.00039668 | -138.3052581930 | 0.5447046023921429 |
| 6 | 0.294245261 | -0.00099143 | -138.3052419963 | 0.5560429591894568 |

TABLE I: The computed values force and energy for a slightly bend $C_{12}H_2$ chain, for various lateral displacements of the two central carbon atoms. The index # in the first column labels the file c12-h2_side_d20dd#.in (and .out). $x$ is the x-coordinate of these 2 atoms. $f$ is the corresponding cartesian $F$ component as given by pw. $E$ is the total adiabatic energy as given by pw. The last column reports $x$ in atomic units, using the conversion factor $a_0 = 0.52917720859$Å taken from the pw source code.

- failure of the minimization algorithms.

The fact that some calculations turn out very good suggests that forces must be fine. But we can test that! We can compute very near fixed-geometry configurations (sub-pm displacements of one or a few atoms).

For each configuration we can compute energy and forces, and see how they compare. Whatever numerical noise will show. Moreover, we can check energy conservation: the work done by the forces acting on the displaced atoms needs to match the change of the total energy (changed in sign):

$$E(x) - E(x_0) = -\int_{x_0}^{x} F(x') \cdot dx' \,, \tag{1}$$

where $E$ is the total adiabatic energy, $x$ is a full geometrical configuration ($3N$ Cartesian coordinates of all positions of all atoms), $F$ is the vector of the $3N$ computed force components acting on all atoms (for that configuration $x$), and the dot product indicates a summation over all displacement components of all atoms. Of course in practice to compute this work one needs to add only the atoms and component which do move along the selected displacement path, since nonchanging components do not contribute to the summation.

I did this exercise for a carbyne $C_{12}H_2$ molecule in configurations near to the starting point of the minimization that prompted this discussion. I attach the input and output files to this report. Note that I set `conv_thr = 1.0d-11`, to make sure that a highly converged electron density is produced. Table I (same as file `check`) summarizes all data.

Figure 1 demonstrates that indeed pw computes the Hellmann-Feynman forces with extreme numerical accuracy. The fitted parabola goes smack through the forces and provides an essentially exact expression $f(x) = a + bx + cx^2$ for the force at any point in the explored interval, with the parameter values given in the figure caption. A linear fit would work equally well for the central points, but a significative curvature is visible over the full $\pm 1$ pm range.

Given the force $f(x)$, it is straightforward to compute the energy variation with Eq. (1), in terms of the function $\epsilon(x) = ax + bx^2/2 + cx^3/3$. We select point 0 in the table as the reference point $x_0$ for the energy differences. The summation implicit in the dot product yields a factor 2, since we selected equal displacements (and, by symmetry, forces) for the two central atoms. The results shown in Fig. 2 are remarkable: although the total energy follows roughly the smooth dependency given by energy conservation, it is clearly affected by significant numerical noise in the order of $0.5\,10^{-6}$ Ry.

This noise is clearly not random, but it adds jumps and steps to an otherwise smooth total energy. It might be due to some rounding approximation, possibly connected to summing large positive and negative numbers. This "stepping" behavior is however quite unfortunate, since it seems to be tricking the bfgs minimization algorithms into error. The fact that forces are immensely more stable indicates that probably the density and wavefunctions are fine. I recommend tracking the origin of this numerical noise to one term or another in the calculation of the total energy. In the present pw version (espresso-4.3), the good old breakdown of the total energy into Hartree, exchange, correlation etc. contributions does not print out explicitly any more (at least by default): I suggest that tracking the changes of these individual contributions for the 12 configurations considered in this work (plus, possibly, others)

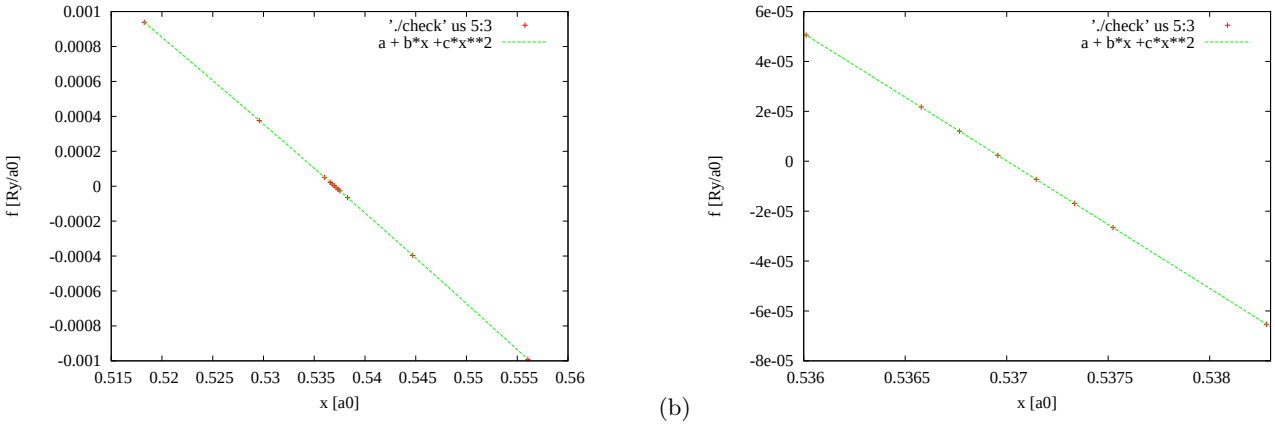


FIG. 1: Crosses: the x-force component computed by pw as a function of the x position of the two central atoms. Line: a parabolic best fit of the force data, with $a = 0.0119920663804434$, $b = 0.00639461504118769$, $c = -0.0534933675170383$. (a) the full $\pm 1$ pm range. (b) a zoom of the central region where the equally spaced points are separated by 10 fm.

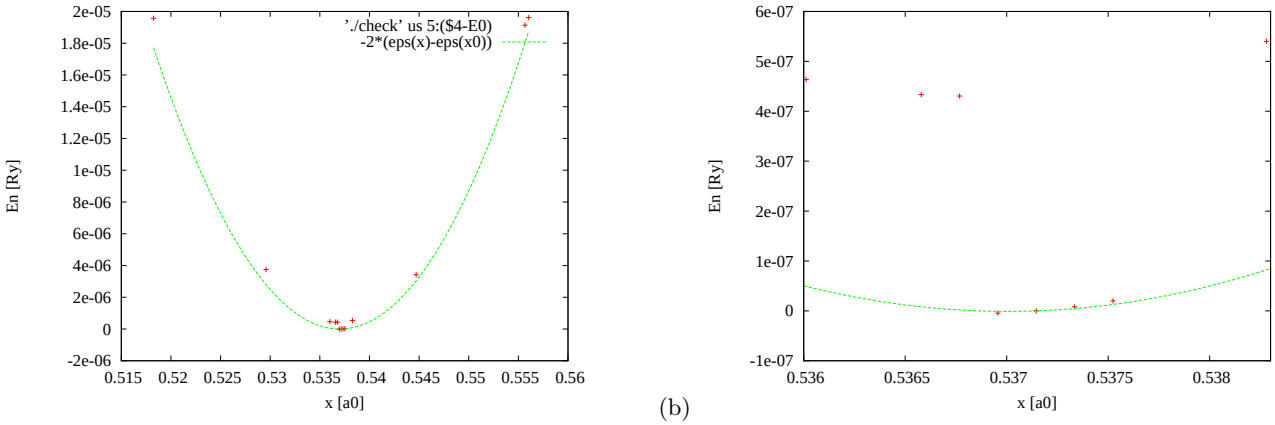


FIG. 2: Crosses: the total energy computed by pw as a function of the x position of the two central atoms. Line: the energy variation obtained by integrating the parabolic fit of the force data in Fig. 1. The reference energy is taken at the $x_0 = 0.284245261/0.52917720859 = 0.5371456978606$ point, where the pw computed energy equal $E(x_0) = -138.3052616101$ Ry. (a) the full $\pm 1$ pm range. (b) a zoom of the central region where the equally spaced points are separated by 10 fm.

could help diagnosing (and then hopefully fixing) this problem. It may turn out that this problem is incurable, but I think not, since over small intervals the total energy is indeed smooth, so there must be some explicit discreteness somewhere (Ewald sums? pseudo?) introducing the discussed jump instabilities in the calculation of the total energy.

Coming back to minimizations: most likely the "good cases" where minimization is extremely accurate happen to have the random energy jumps around the minimum not too close to it, and most likely upward in energy. In contrast, the failure cases happen to have such random jumps very close to the true minimum (as dictated by the forces), with some of them downward in energy, thus erroneously presenting as energetically convenient some configuration which should instead be energetically unfavored. The present calculations clarify in part why the damped dynamics often works better than bfgs minimization near the minimum: the forces seem fine, and thus lead to the actual relaxed geometry, while energy might lead to get lost, and bfgs does indeed do regular checks on energy, while damp just follows some (which?) dissipative dynamics following the forces only. If this jump noise problem was fixed, then probably the bfgs minimization algorithm would highly benefit: it would stop failing due to finding the energy going up while moving according to the forces.

Independently of the above considerations on total energy, the "damp" algorithm should be revised in detail: far away from the minimum its behavior is appropriate (see `test_ok.in test_ok.out`), while near the minimum the reported oscillatory behavior is clearly pathological and simply indicates that the algorithm "gets too eager" to converge. A demonstration of this problem is in `test_bad.in test_bad.out` which is the same as `test_ok` except that it starts off *nearer* to the minimum! Energy remains the same, but damp is clearly leading away from the minimum, as `grep Tota test_bad.out` proves, and the forces are oscillatory. Maybe `proj_verlet` uses a too large

time step (by roughly a factor 2). I looked into the code but could not find where variable `dt` is set.

Even better than a damped dynamics, probably the best minimization type near a minimum would be a plain conjugate gradients approach. Would that one be too hard to implement? I see it is already implemented in the cp code, so it may be simple to carry it over into pw...