ReadMe file for new version of mixer (VER7.1.2)

L. D. Marks, 7 December 2016

Contents

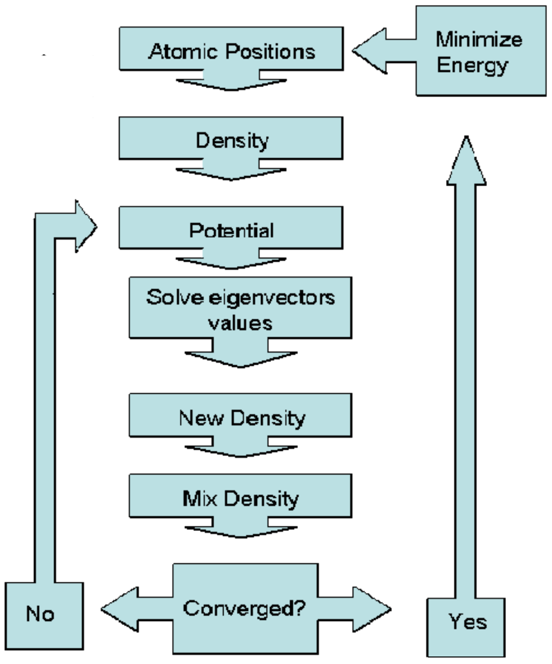
Introduction

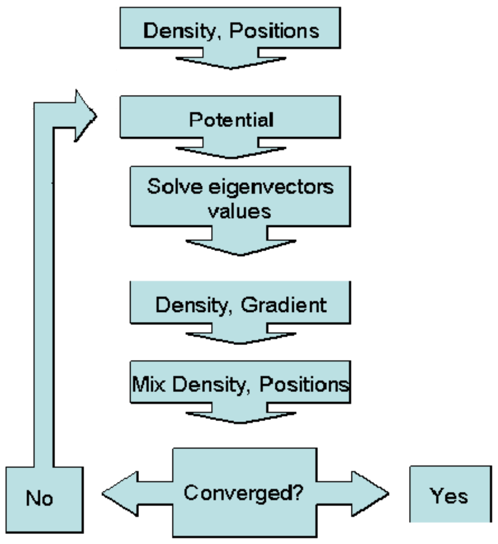
This directory contains Version 7.0 of the mixer, which is an update of 4.1 which was a major rewrite/restructuring with some new algorithms while preserving most of the old code.

For routine calculations, most users will see little difference in the behavior of this version and the one currently in the WIEN2k_10 distribution (version 3), and the MSEC1 from the previous version is preserved intact. In some cases it will be a bit faster, perhaps even 10-20% faster for poorly posed problems. With version 7.0 structures with soft modes should be handled much better than in previous version.

One major innovation with this series of releases is the addition of a new algorithm MSR1 (MultiSecant Rank One) as well as an algorithm for simultaneous “mixing” of densities and atomic positions, MSR1a. In a standard DFT calculation, a double loop is used for finding the lowest energy structure as shown below on the left. What MSR1a does is treat the densities and atom positions as variables, and find the self-consistent solution for the densities and gradients (i.e. gradients small or zero) simultaneously, as illustrated below on the right.

This is faster by a factor of 1.5 to 4.0 than the double loop in large test cases, perhaps even more in general.





This algorithm used is not very different from the older MSEC1 or the new MSEC3. Using standard notation, we define two vectors s_n and y_n for the n’th iteration as

$$s_{n-1} = \rho_n - \rho_{n-1} \quad ; \quad y_{n-1} = R_n - R_{n-1} \tag{1}$$

where the residual is $R_n = F(\rho_n)-\rho_n$, i.e. the difference between the density at the start of iteration n (ρ_n) and that ($F(\rho_n)$) produced by solving the Kohn-Sham equations for the potential generated from ρ_n. We further define two matrices containing some number (m) of prior steps:

$$S = [\ s_{n-m}, s_{n-m+1},\dots s_n\ ] \qquad \text{and} \qquad Y = [\ y_{n-m}, y_{n-m+1},\dots y_n\ ] \tag{2}$$

We describe the original version (MSEC1 as well as MSEC3) via a Jacobian H:

$$H = \beta\,(I - Y\,(Y^TA+\lambda I)^{-1}A^T) + S(Y^TA+\lambda I)^{-1}A^T, \text{ where} \tag{3}$$

β = mixing greed, λ = regularization term
A = Y for the MSEC1 family

The next step is then

$$\rho_{n+1} = \rho_n + H\,R_n \tag{4}$$

For the MSR1 family, equation (3) is changed to:

$$H = \beta\,(I - Y\,\{Reg(Y^TA)\}A^T) + S\{Reg(Y^TA)\}A^T \tag{5}$$

Where $A=(\alpha S+Y)$, and $Reg(Y^TA)$ is a Tikonov regularization of the mxm matrix Y^TA, which can be written as

$$Reg(Y^TA) = Reg(B) = (B^TB++\lambda I)^{-1}B \tag{6}$$

This is a slightly more greedy algorithm, but not as greedy as Broyden's first method (often called "Good Broyden") where A=S. (For mixing Broyden's first method has, to date, failed.) This algorithm can be considered as an adaptive algorithm, since if $|S|>>|Y|$ it becomes Broyden's first method while if $|S|<<|Y|$ it becomes Broyden's second method which is what is used in MSEC1 and MSEC3. In the earlier versions the constant α=1, with later version this is adjusted inside the code.

For the new algorithms where atomic positions are moved at the same time as the densities, all that needs to change is the definitions of s_{n-1} and y_{n-1} which now become:

$$s_{n-1} = (\rho_n - \rho_{n-1}\,,\, x_n-x_{n-1}\,) \qquad ; \qquad y_{n-1} = (\,R_n - R_{n-1}\,,\, g_n-g_{n-1}\,) \tag{7}$$

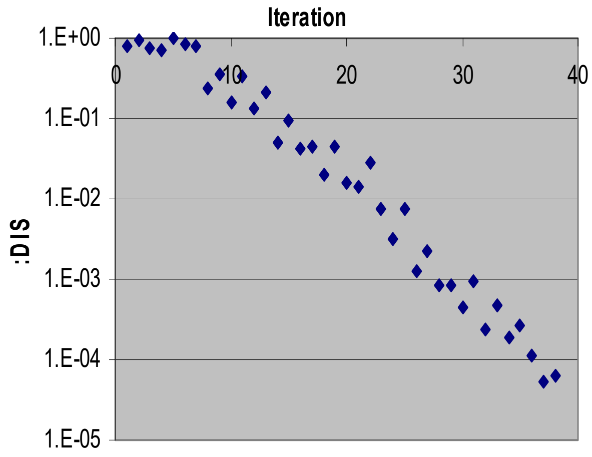
where x_n are the atomic positions, and g_n the (Pulay corrected) forces.

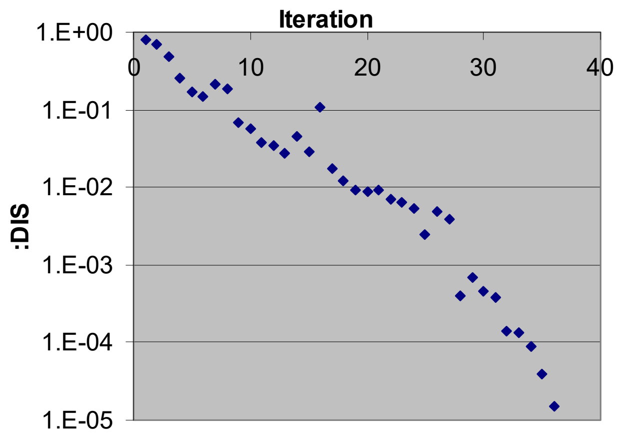
To understand how this can work, realize that the Jacobian used will now contain information not just on how the residue R_n changes as a function of density ρ_n, but also how the gradients (and Pulay corrections) change with density, the density changes with atomic positions as well as how the gradients change with atomic positions. Hence in effect it has a higher-order interpolation of the charge density for changes in the positions (higher-order than the currently very effective clminterpol code).

This can be a better method than using a double-loop since we are solving a variational problem even though we do not have a variational energy – the energy output by WIEN2k will converge to the true energy but only at the fixed point (self-consistent) solution. For fixed atomic positions we are in effect solving a constrained minimization, and the convexity of this (in effect how nice is the valley around the minimum) can be relatively poor. By opening up the problem so the atoms can move the problem can only become more convex, and consequently easier to solve.

Of course, with a code such as WIEN2k this is only true if we do not have ghost-bands. Fortunately the change in how linearization energies are determined in version 10.1 has been a major improvement in this (thanks to Peter Blaha). In addition, because we have an implicit density extrapolation as mentioned above this is less likely. *Caveat: ghost-bands can always occur, so this cannot be guaranteed.*

The MSR1a code heads variationaly downhill in energy, simultaneously making the densities converge and the forces smaller. Sometimes it will appear to be going uphill in energy or with the forces, but if you look carefully this is generally after some large displacements of the atoms and what is being adjusted in the next iterations is the densities. Like all Quasi-Newton methods, it is largely scale invariant so does not depend so much upon how the problem is formulated. That said, it is also known that the more tightly bunched are the eigenvalues of H, the faster will be the convergence. To improve this, there have been some changes in the variables used within the mixing to more appropriately take account of the effect of the multiplicity of the atoms as well as the degeneracy of the (hkl) plane waves and to use the same norm for the L=0 density within the muffin tins as that used for L ≠ 0; more details are given

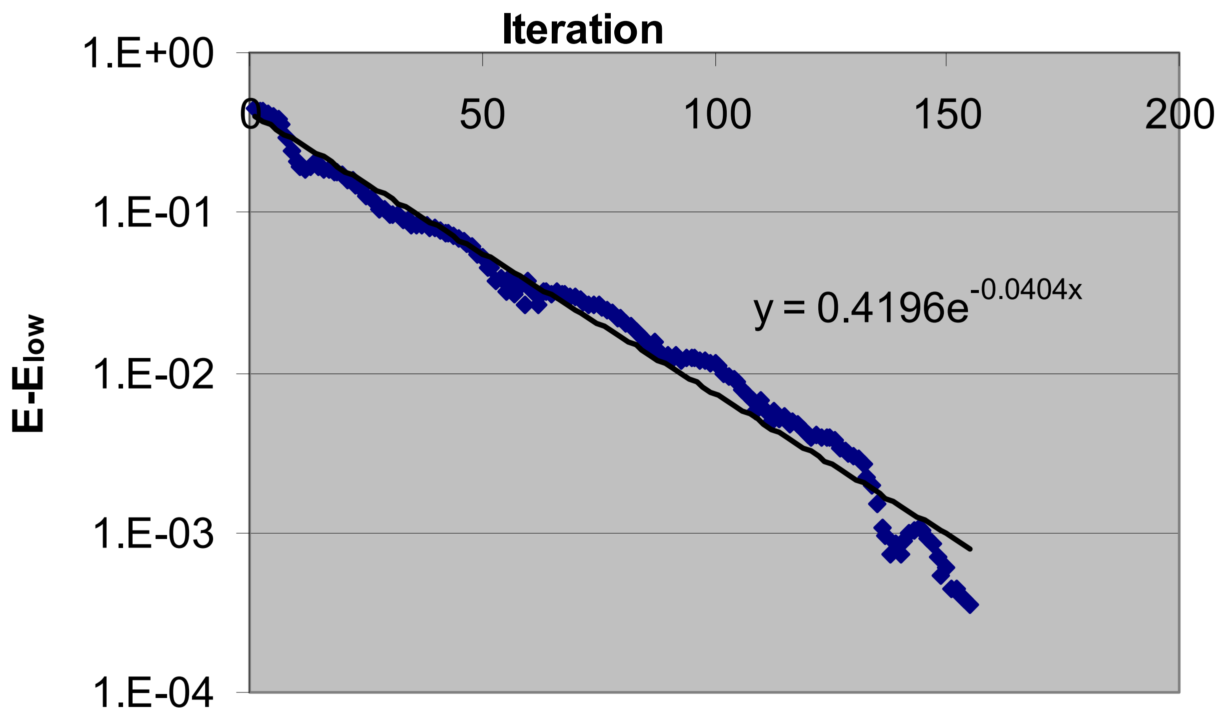




later. Another subtle change (in MSR1 only) in the earlier version was the introduction of a method for dynamically choosing the regularization parameter, but this has now been removed. An illustration of the convergence for MSEC3 is given below on the lower left for a relatively bad problem, a Ni (111) surface, which shows linear convergence; for well posed problems there is superlinear convergence. This can just about be seen for the problem on the lower right, a reduced TiO_2 c2x2 surface. (For reference, in both cases the initial densities were from dstart, i.e. a superimposition of isolated atomic densities.)

Similar to the older MSEC1, the convergence depends very weakly upon the mixing greed or the problem size (number of atoms) and is in the range of 10-50 self-consistent iterations, 10 for a well-posed problem such as bulk MgO, 40 for a Ni (111) surface or similar.

As an example of how MSR1a behaves, shown below is a plot for a relatively large problem of the change in energy referenced to the minimum as a function of energy (in Rydbergs). It shows a reasonable linear convergence with perhaps some superlinear convergence at the end although this is unclear.



One point to be careful about is the forces that are reported by MSR1a, as well as the path that the algorithm takes. It does *not* track along the Born-Oppenheimer surface, i.e. the surface of lowest energy for a given set of positions (which is what PORT does). It tracks on a different surface. Consider the "Surface of Zero Force", i.e. the surface where for a given set of positions the forces on the atoms are zero, which will in general not be an electronic ground state. The algorithm tracks in between this surface and the Born-Oppenheimer, and the forces reported will in general be smaller (sometimes much smaller) than those of the Born-Oppenheimer surface. The two surfaces meet at the true minimum solution.

As a consequence you should always do a standard scf iteration to converge after MSR1a to check that you are close enough to the true solution. In addition, you should always converge with MSR1a for much smaller forces than one would normally use in PORT, coupled with a moderately tight charge convergence (e.g. –cc 0.00025) for a fully converged final result. That said, often it is not needed to converge to such a high level, for instance to test if a particular structure is feasible.

The original version was not bad, and there have been a few changes and hopefully improvements with the different releases. In the next sections the changes are briefly described.

Major Changes with Version 4.1

1) The old Broyden method (qmix5) has been retired.
2) A variant of the Pratt mode (PRAT0) uses the NFL/Trust-region control of MSEC1, so PRAT0 0.5 should be safe (albeit slow).
3) The case.clmsum/up/dn files have one digit higher precision by use of ES rather than E for output. This *does not* make them any bigger.
4) Relevant parts of mixer.F have been vectorized/converted to BLAS. Redundant parts of the code have been eliminated and many of the subroutines used simplified.
5) The prior MSEC1/MSEC2 is still in the code, in subroutines which all have a "7" in them. Newer versions have an "8" in them. The behavior of MSEC1/MSEC2 should be identical to before, with minor changes due to 3) and 4).
6) Newer versions contain vectorized/BLAS calls, more for consistency and accuracy than anything else. Note: W2kinit.F sets high-precision for the vectorization calls. This should reduce/avoid issues with ifort reduced-accuracy optimizations.
7) MSEC3/MSEC4 are new versions of MSEC1/MSEC2. They have a consistent L2 scaling of all the variables including atom multiplicity as well as the plane wave degeneracy. They are slightly better in some cases than the old versions, although with –in1ef it is hard to say. For certain, they should behave the same for different problems (equally good/bad).
8) The algorithms MSR1, MSR2 are new algorithms based upon a rank-one multisecant update; MSR2 uses only LM=0 similar to MSEC2. The MSEC1 and MSR1 families can both be represented in terms of a Jacobian in equations (3) & (5) above. If β =1 MSR1 is similar to a multisecant version of the symmetric rank-one update (SR1). This algorithm is slightly more greedy than MSEC3, and (more importantly) is more appropriate for optimization problems. For methods using simultaneous atom minimization it is much better. Whether MSR1 is better than MSEC3 probably depends more upon the class of problem, and how convex is the physical model/numerics. At present MSR1 appears to be slightly faster and equally robust in most cases.
9) The algorithms MSR1a, MSR2a, MSECa, MSECb do a simultaneous "mixing" of densities and atomic positions. MSR2a is a LM=0 only version of MSR1a (untested), similarly MSECb. They add the gradients (FOR required in case.in2) and atomic positions. See later for notes on MSR1a. MSECa and MSECb are not recommended.
10) A new algorithm is used to choose the regularization parameter within a certain range for MSR1 and MSR1a via a Generalized Cross-Validation, which allows the algorithm to work better both for well-posed problems as well as poorly posed ones. It is not used by default for MSEC3 as a fixed regularization appears to be just as good in this case.
11) The maximum unpredicted greed in the early stages when :DIS is large has been raised, as this was leading to bad convergence as this violated Wolfe conditions.
12) All the new algorithms use a different method of handling the inverse in the Jacobian which gives an estimate of the conditioning (COND) defined as the condition number of the matrix ($Y^TA+\lambda I$) multiplied by the regularization term λ and divided by the number of memory steps. This is useful information; the smaller the condition number, the more convex the problem (probably on a log scale). This is now output in the :DIRM line.

13) All the new algorithms give scaling for the relevant PW/DM/Atoms terms in case.scfm. The Atom modes also print out the positions and displacements (:APOSXXX for grep).
14) The term "MIXING SCHEME WITH XXX" in the old version has been replaced by "MIXING GREED XXX" in the hope that users will start to understand that XXX is *not* what most people think it is. Just as being too greedy is bad, being not greedy enough is also bad – the algorithm starves to death. (More technically, it will not come close to satisfying Wolfe conditions. Note: Wolfe conditions for fixed-point iterations do not appear to be documented in the literature, but everything points to them being real – a paper for a mathematician.)
15) A command "grep -e :DIR -e GREED -e :FRMS -e :ENE -e :CHARG -e PRATT -e :DIS -e "MIXING SC" -e PLANE *.scf $1 | tail -40" will catch most useful output, e.g. giving

```
:DIS  :  CHARGE DISTANCE       ( 0.0001591 for atom   16 spin 1)      0.0000583
:PLANE:  INTERSTITIAL TOTAL      7.77491 DISTAN  5.136E-03 %
:CHARG:  CLM CHARGE   /ATOM     28.25380 DISTAN  3.584E-04 %
:DIRM :  MEMORY 6/8 RESCALE   2.61 RED 0.597 PRED 0.781 NEXT 0.691 COND 1.39E-01
:DIRP :  |BROYD|= 6.586E-04 |PRATT|= 1.510E-04 ANGLE=  18.7 DEGREES
:DIRB :  |BROYD|= 1.220E-03 |PRATT|= 3.694E-04 ANGLE=  15.5 DEGREES
:MIX :    MSEC3 REGULARIZATION 5.00E-05 GREED 0.145  Newton 1.000
:ENE  : ********** TOTAL ENERGY IN Ry =      -116780.86459533
```

The scaling of :PLANE and :CHARG is different with the new modes, only the % values are comparable but these not completely because a L2 norm is now used. The "REGULARIZATION" is the value of λ used, which by default varies (see later). For an atom minimization the output will look more like

```
:DIS  :  CHARGE DISTANCE       ( 0.0007947 for atom   17 spin 1)      0.0002363
:PLANE:  INTERSTITIAL TOTAL      7.77798 DISTAN  1.110E-01 %
:CHARG:  CLM CHARGE   /ATOM     28.25379 DISTAN  1.303E-03 %
:DIRM :  MEMORY 8/8 RESCALES   1.06  0.93 RED 1.276 PRED 1.000 NEXT 1.000 COND 5.50E-02
:DIRA :  |BROYD|= 5.444E-02 |PRATT|= 7.006E-04 ANGLE=  69.7 DEGREES
:DIRP :  |BROYD|= 4.652E-02 |PRATT|= 7.962E-04 ANGLE=  85.8 DEGREES
:DIRB :  |BROYD|= 5.000E-02 |PRATT|= 1.087E-03 ANGLE=  86.8 DEGREES, STEP= 0.51077
:FRMS (mRyd/au)    1.313 :DRMS (au)  7.067E-03
:MIX :    MSR1a REGULARIZATION 3.81E-04 GREED 0.087  Newton 1.000
```

Here :DIRA is the atomic step, two scales are show (one for the PW, one for the atoms) and :FRMS is the RMS forces with :DRMS the RMS atomic displacement of this cycle.

For orbital potentials, density matrices an additional "DIRD" term is show and the 2nd scaler is that for these. For orbital potentials and atoms there are three scalings shown (not tested).
16) The atomic mixing outputs more precise atomic positions (more decimal points) at the end of the case.struct file.

Major Changes with Version 5.3

Version 5.3 is a fairly substantial upgrade of the mixer. While the core alogithms (MSEC3/MSR1/MSR1a) have not changed, a much improved method of controlling the overall step using a trust-region method has been added, and some simple limits added (as defaults) to reduce the algorithm greed for problems involving d or f electrons (which tend to converge poorly in Wien2k due to Fermi-surface sloshing). For simple cases with the standard non-optimizing algorithms (MSEC3/MSR1) in most cases one will only see a small change, reduction of overly aggressive steps. (The trust region is better than MSR1, and since this algorithm appears to be better in most cases than MSEC3 it is now the recommended option and the main target for development.) Particularly for metals involving d or f electrons the performance of MSR1a should be much improved, although it does not compare to MSR1a for insulators which, for reasons associated with the physics of the problem, will almost always converge faster. In more detail:

1) A trust-region method has been added to control overlarge steps (new in 5.3). This, combined with other changes improves the stability for what were hard cases with the initial version
2) The algorithms are now fully controlled by the implicit trust region of the greed in all cases. In effect, the total atom step and the total step cannot exceed a*GREED where a is a constant (set internally for the two cases) and GREED is the locally adapted greed as output in case.scfm (grep –e :MIX *.scf).
3) The code now understands when the user is employing the mBJ potential and will allow mixing of vresp to be controlled by the GREED in case.inm_vresp. (Previously it was always 1.0.) A value of 0.5 in case.inm_vresp is reasonable. Note that in other cases mixing of vresp is over-ridden and a value of 1.0 with PRATT mixing is used. In all cases the density in vresp is not normalized.
4) A different method is used to normalize the density in MSR1a. For version 4.1 any density lost was added to the plane waves; with 5.3 it is added to the valence density instead.
5) Very small forces are now trapping; in principle they could lead to instabilities even with some simple cases (e.g. an Al surface) in 4.1
6) :DIS is now more correct as an L2. The numbers *are different*, but in most cases this will not be apparent.
7) :FRMS is now more correct.
8) Sudden increases in the total step are now trapped.

Major Changes with Version 6.0

Version 6.0 has large changes in many details of the algorithm, the new version hopefully being both more convergent and more robuts. Some issues in the prior version which would not always converge well soft modes appears to have been solved.

A detailed description of this algorithm can be found in *Fixed-Point Optimization of Atoms and Density in DFT.* Marks, L.D., Journal of Chemical Theory and Computation, 2013. **9**(6), 2786-2800.

Major changes in Version 7.1.2

Version 7.1.2 will appear essentially the same to the user, but in operation has some significant differences. The earlier version used mainly the Greed (sometimes inaccurately called the mixing factor) as an implicit trust region control, i.e. it was kept small when the linear model of the expansion of the Jacobian was not that good. An explicit trust region control via the total step length and the movement of the atoms was more a secondary safety constraint.

While this can work, it turned out to be weak for cases where there are soft modes. In these cases the Greed needs to be larger even if the algorithm is not making substantial improvements. To handle this, version 7.1.2 uses larger values of the Greed and the explicit trust region much more and also has an additional explicit trust region control for the total charge change within any of the muffin tins. As such it works much better for soft modes and seems to still work very well for other standard problems.

One addition in Version 7.1.2 is (with MSR1a) a printout via a different method of an estimate of how far the current positions are from the minimum. While this is not an exact metric, since it uses a different algorithm it works as a good check. For a true minimum the density differences, forces as well as this different should all be small; there are cases where the density differences and forces seem to be small but from this other metric it is apparent that there is still some distance to the minimum – a characteristic of soft modes.

Parallel Atomic Minimization Algorithms (a.k.a. the Energizer Bunny)

The MSR1a, MSECa algorithms are strange beasts. They are surprisingly stable, without extreme :DIS divergences and I have never seen ghost-bands with them (except with –it, but these are manageable in most cases tested so far and seem to have been patched by a change in lapw1 and an improvement to vec2pratt). The new –in1ef option (default on) is particularly useful for them. While in principle they can use more accurate position and gradient information, they in fact work almost as well without this.

To use them:

a) Do a preliminary scf convergence to something like –cc 0.1 –ec 0.25 –fc 20 (I do not recommend a full convergence).
b) Do “x pairhess”, since the information in case.inM is used (both the constraints and trust region number). If you forget, the program will do this for you.
c) If you have a badly posed problem, edit the atomic trust region in case.inM to something like 0.1. For a well posed problem the default of 0.2 is fine.
d) Decide upon some convergence level, e.g. –cc 0.001 –ec 0.001 –fc 1, but be careful. How to determine when the algorithm has converged is tricky, and I do not have a full solution.
e) Launch, and be patient since it can take > 100 or more iterations to converge, but this is still faster (1.5 to 2 times, perhaps better) than PORT in many cases. A nickname of “the Energizer Bunny” for these algorithms is appropriate, as they keep going and going downhill in energy.
f) Run a final, conventional scf convergence after it has converged to “some level”, and check that the forces are OK. (This is done automatically in the scripts.)

To date both full diagonalization and iterative diagonalization with –it –vec2pratt –noHinv have been fairly well tested. It is *strongly* recommended that –vec2pratt be used with the iterative diagonalization, it is much better.

Also, ensure that you have small enough R0 values for the atoms; use lstart and look for warnings. This can also lead to problems. Finally, ensure that you have a “good” case.in1. I suggest running lstart then editing case.in1_st for the appropriate RKMAX, energy and number of bands and replacing old versions. Small steps in the linearization energy search turn out to be rather important.

After running the algorithm, it is safest to remove the precise positions at the end, for instance do “x patchsymm” then copy case.struct_new to case.struct. This will ensure that you have a consistent structure file with atomic positions in the normal range of 0-1.0 .

Note: The algorithm uses a Trust Region (see later) to limit steps. The size of the step is printed out in the line where the GREED is also printed (use grep –e :MIX). This might become small for three reasons:

1) The code reads the first line of case.inM (if it exists) and uses ¼ of the trust radius (default 0.35 a.u., damped by the value of :DIS to prevent the algorithm being too greedy far from the Born-Oppenheimer surface) as the largest step to take, and will reduce it if it is larger than this. (If case.inM is not present it will automatically execute pairhess to create it.)
2) It will only take a fraction of a full PRATT step.
3) The requested step would have brought atoms within 0.025 a.u. of each other, i.e. touching spheres.

Only 3) is a real concern, and if it occurs repeatedly you probably need to reduce the RMTs and use clminter; the others can be ignored

I strongly suggest MSR1a over MSECa, it appears to be much, much better – MSECa fails to properly minimize and stagnates near saddle points.

The algorithms find a fixed point of (ρ – F(ρ),g) as a function of (ρ,x), where

ρ	Electron Density
$F(\rho)$	SCF mapping of the density, i.e. CLMNEW, RHONEW
g	Gradients using :FOR
x	Atomic positions

For this to work, the gradient g has to approach the true gradient fairly quickly as the density converges, and it appears that it does. Beyond extending the prior S, Y information as well as the previous density and residue vectors to include x and g, almost nothing is different. For x and g "natural" units of a.u. and mRyd/a.u. appear to be correct.

It is *very important* to recognize that these algorithms are fundamentally different from MSEC1/PORT. For a well posed problem the two give the same result; for a badly posed problem they probably do not. These algorithms are sensitive to the convexity and global nature of the DFT problem being analyzed, and appear to be essentially variational in character (which in itself is a very interesting observation). So far they have been tested with PBE-GGA and PBE0 as well as LDA+U (see the later section on Ghost Forces).

The factor of 1.5-3.0 speed increase it not as good as I would like, and is going to be problem dependent. Some preliminary indications are that systems with soft modes (e.g. hydroxide) are a bit problematic, although these are also not so easy with PORT. Almost certainly there are some subtle noise issues in the current numerics of WIEN2k, not bugs but fundamental limits of the numerical algorithms (perhaps the differentiations/integrations/fits). I am hopeful that the speed will increase in the future.

One caveat: it is not impossible for the algorithm to find a saddle point, rather than going towards a real minimum. Whether this can in fact occur is currently unclear. (No precautions have been taken to force the Jacobian to be positive definite as is done for standard minimization codes.) To date it has not happened in tests with MSR1a. (If you start near a saddle point the progress may be very slow for a while.)

The algorithm can work with a badly posed problem, for instance a metallic surface with only a few k-point sampling of the Brillouin Zone and a small temperature factor (or tetrahedron mode). Note: I do not recommend the use of TEMP for any minimization (PORT as well) as it is not variationally correct, instead TEMPS should be used. For this type of case you can reduce the atomic trust region in case.inM down to perhaps 0.1 and use a GREED of 0.1, or reduce the Trust-Region using “TRUST 0.5” (see later). This, or some variant on this should work although in the long run it is probably better to use, for instance, more k-points.

Similar to a more conventional minimization using PORT, it is recommended that you do not use the largest RKMAX for initial minimizations starting far from a solution, rather a “good enough” value. Again the same, you need to ensure that the muffin tins are far enough apart as otherwise the calculation will get stuck.

Finally, a strong statement and perhaps the most important use of MSR1a. If a particular functional/model combination fails to converge of converges badly with MSR1a, then it is > 98% probable that the combination is inappropriate. This follows from the fact that the algorithm appears to be variational, so a failure to converge or poor convergence implies weak to no convexity of the numerical method, i.e. it does not obey the KS variational principle. The most obvious case where this can occur is problems which are badly posed, for instance a metal without enough k-points for the Brillouin Zone sampling.

L2 Renormalization

The newer versions all use a L2 renormalization. What is meant by this is that the sum of the squares of ($\rho - F(\rho)$), the density matrices and atomic positions are all correct for the multiplicity of the atomic sites as well as the plane waves. For the densities within the muffin-tins, density matrices and atomic positions this is done by multiplying the current units by sqrt(MULT(JATOM)) where JATOM is the relevant atom. For the plane waves this involves dividing by sqrt(NST) where NST is the number of equivalents for each reflection. The newer versions also correct the norm of the L=0 densities within the muffin tin as this is more consistent.

This has the important advantage that it naturally compensates for different terms in the diagonal approximation of the Jacobian, making it as close to unity as possible with one caveat. This caveat is for the plane waves, where an argument can be made that one should take account of systematic enhancements of the Fourier coefficients as a function of the number of symmetry elements acting on a given (hkl). This is something which is well established for direct methods where one would add an enhancement term ε when calculating the normalized structure factor in statistical methods. At the current moment I cannot say whether one should use an ε term, I have no clear evidence indicating that it is better.

One consequence of this scaling is that the behavior of the mixer should be more independent of multiplicity and symmetry. For instance, for fcc silver the behavior should be the same whether it is calculated with the full fcc symmetry or in any reduced symmetry subgroup.

Like all quasi-Newton methods, to a large extent the algorithms are auto-scaling so rescalings such as this are secondary effects. That said, it is known that if all the eigenvalues of the Jacobian were the same, the convergence would be very, very fast so conditioning such as this does have an effect. Unfortunately there appears to be little detailed mathematical analysis so the only approach that can currently be used is “seat-of-the-pants” intuition. This renormalization has the correct aji (味).

Note: a logical scaling would be to convert to absolute density units so the L2 for each variable in the basis set correctly represents the absolute density change it represents. This did not seem to work, but may have been done wrong.

In a little more detail, the residual R_n (= $F(\rho_n)-\rho_n$) is split up into into parts for the different components of the basis set, i.e. within the muffin tins (CLM), in the interstitial (PW) and (as needed) for atoms, density matrix and orbital potential terms. The code then iteratively finds the scaling terms a^{PW}, a^{Atoms} etc such that (for the simplest case)

$$\sum_n a^{PW} \left\|R_n^{PW}\right\|^2 / \left(a^{PW} \left\|R_n^{PW}\right\|^2 + \left\|R_n^{CLM}\right\|^2 \right) = \sum_n \left\|R_n^{CLM}\right\|^2 / \left(a^{PW} \left\|R_n^{PW}\right\|^2 + \left\|R_n^{CLM}\right\|^2 \right) \quad (8)$$

Where the sums are over the relevant memories used. A renormalized set of residues as well as steps is then generated by rescaling by the relavant terms (e.g. multiplying the PW components by $\sqrt{a^{PW}}$ As a consequence, the different components all have the same relative importance.

In a later step the Y and S matrices are also renormalized using

$$Y = (y_{n-m}/|y_{n-m|}, y_{n-m+1}/|y_{n-m+1}|,\ldots y_n/|y_{n-1}|)\ ;\ S = (s_{n-m}/|y_{n-m|}, s_{n-m+1}/|y_{n-m+1}|,\ldots s_n/|y_{n-1}|) \qquad (9)$$

This converts them effectively into a set of directions so large excursions (due to, for instance, ghost-bands) do not dominate as they would if absolute values were used.

Regularization

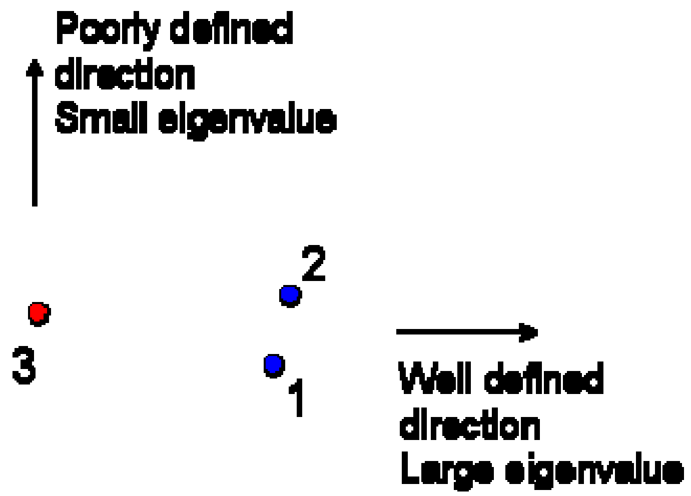
The automatic regularization used in version 5.3 has been retired as it was too strong. This section is left for completeness. (There is code for this using MSEC3, but it does not appear to be as effective as a constant.) Regularization comes in when the algorithm calculates the vector b which satisfies (in a regularized fashion

$$(Y^TA)b = A^TF \tag{10}$$

where F is the current residue vector and A has been previously defined. What the algorithm does is used a SVD to regularize the matrix Y^TA (which is not symmetric) and from this calculate a pseudo-inverse. The exact degree of regularization is chosen via a Generalized Cross-Validation scheme such that b is sufficiently accurate; more details on this will be providede.

As a brief aside, we need to use a regularization as otherwise the inverse of B may inflate eigenvalues/vectors which are mainly noise, leading to large steps in bad directions. To illustrate this, consider the diagram below where we are at point 3 and have prior information about points 1 and 2.

As marked, along the x axis the change (implicitly inside B) is well defined, and the eigenvalue


Poorly defined direction Small eigenvalue
2
3
1
Well defined direction Large eigenvalue

will be large; along the y axis it is less well defined and the eigenvalue will be smaller. When we solve for x_{Reg} above the small eigenvalue can have a large effect which is highly undesirable. Therefore the term λI is added (called a Tikonov regularization, similar to a Wiener filter), see for instance http://en.wikipedia.org/wiki/Tikhonov_regularization.

The current version uses a fixed term for lambda which can be adjusted but there is no indications that this is useful.

Trust Region Control

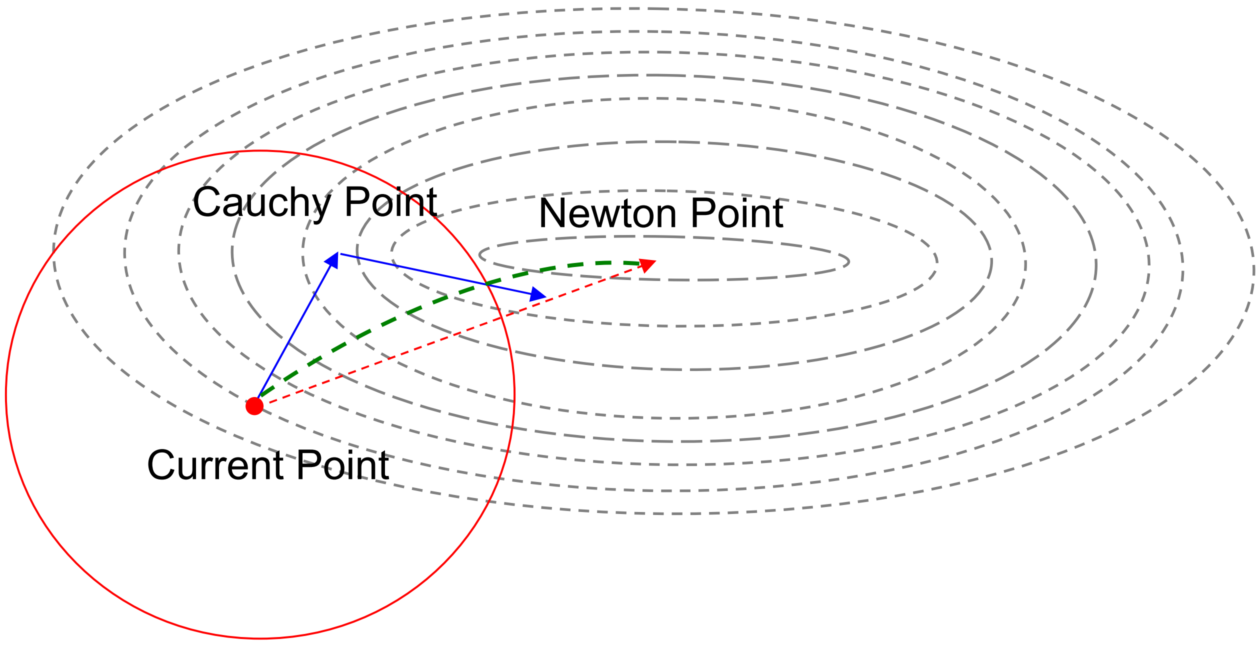
New with version 5.3 is a Trust Region control for overlarge steps. Taking r_i as the residual and p_i as some step, the approach is to use as a model the function

$$T = \tfrac{1}{2} (r_i - H^{-1}p_i)^2 \qquad (12)$$

then, if the step requested is too large chose some appropriate step that minimizes in some sense T. Following standard terminology, there are several plausible steps:

a) Cauchy Step: This is along the direction of steepest descent for (12), with a full magnitude the optimum along this direction.
b) Double Dog-Leg: This starts from the Cauchy step, and via a double dogleg will approach the full (Quasi) Newton step.
c) Hybrid step: This is specific to MSR1, and uses the full MSEC3 step as the starting point (instead of a Cauchy Step) for a double dogleg.
d) Newton step, from 0.8 to 1.0 the nominal optimizer of (12)
e) Levenburg-Marquardt step. This is a more exact solution for a limited size of step projected onto the previous history.

The algorithm will first test a full Newton step; if this is too large or would lead to touching spheres it will move to a the Levenburg-Marquardt step for MSR1 (not implemented for MSEC3), then the reduced Newton step d) above, if this fails then c), b) and finally a) in order. This is illustrated below where (for simplicity) only the Cauchy Point is shown, with the red circle the Trust Radius, blue the double dogleg path and Green the Levenburg-Marquardt step. (Note: this diagram is schematic only.)



This approach appears to be somewhat more stable than earlier versions which just reduced the step length towards the Newton Point.

One additional, related change is that the algorithm now automatically reduces the atomic trust-region size if the structure contains atoms with d or f electrons. This appears to work well to improve stability for some metals which were harder to optimize with version 4.1. As a further step you can make the Trust Region even tighter by including at the end of case.inm a line “TRUST 0.5”, or even “TRUST 0.25”. Be aware that in the later case the algorithm may not properly converge because the Wolfe condition is being violated (i.e. too small steps to obtain meaningful new information) or it may be caught in a trap (a local minimum of the residue rather than a true root. This method is probably better than reducing the mixing GREED below 0.05. However, if you really need a small value here it probably means that there is something wrong with the physical model being used.

The code will output what it uses at every step, for instance (grep –e :MIX)

```
:MIX  :  MSR1a  REGULARIZATION: 2.91E-02  GREED: 0.098  LMStep 0.338
:MIX  :  MSR1a  REGULARIZATION: 2.54E-02  GREED: 0.200  Newton 0.877
:MIX  :  MSR1a  REGULARIZATION: 3.07E-02  GREED: 0.200  Newton 1.000
:MIX  :  MSR1a  REGULARIZATION: 3.50E-04  GREED: 0.100  Hybrid 0.119
:MIX  :  MSR1a  REGULARIZATION: 3.50E-04  GREED: 0.200  Hybrid 0.317
:MIX  :  MSR1a  REGULARIZATION: 3.50E-04  GREED: 0.200  Newton 1.000
:MIX  :  MSR1a  REGULARIZATION: 3.50E-04  GREED: 0.100  DogLeg 0.663
:MIX  :  MSR1a  REGULARIZATION: 1.23E-03  GREED: 0.121  Hybrid 0.263
```

For the Levenburg-Marquardt step the number reported is the absolute magnitude of the step relative to that of a full MSR1 step. In other cases the number is how far along the particular path the step is, starting from (for Hybrid or DogLeg) 0.000 at the MSEC/Cauchy point.

For MSR1 the Levenburg-Marquardt step is the default; it is not implemented for MSEC3. In practice the difference between this step and a simpler DogLeg is often small, but in tests it appears to be rather more stable.

Charge Renormalization in MSR1a

The standard renormalization of the charge density in Wien2k is to rescale the plane wave density so the total is correct. In a normal run this is appropriate for the input density, as any lost charge is due to poorly confined core densities so belongs in the plane waves. In version 4.1 the same method was used in MSR1a for the density output from the algorithm.

This is not really correct or optimal for MSR1a. Any loss of density is due to inadequate interpolation of the effects of changing the positions of the atoms, which will transfer parts of the density from within the RMTs to the plane waves (or vica-versa). To a good approximation the core density will not depend upon the positions of the atoms. With 5.3 the charge renormalization is done with the total valence density (i.e. the density minus the core density), which is better.

Ghost Bands and Ghost Forces

Primarily with iterative modes in WIEN2k one can have Ghost Bands. This can occur if there has been a bad choice of linearization energies in case.in1, but this is largely reduced by the –in1ef switch in the more recent releases. In the iterative modes if can also occur if the linearization energies of the states used from the previous iteration are bad approximations to those for the current iteration. When atoms are moving this is more likely to occur.

Something (undesirable) new is what I am calling "Ghost Forces", for want of a better name. What happens is that the force for some atoms suddenly becomes very large, sometimes > 1000 mRyd/au without anything apparently wrong with the density or energies. This only occurs with the iterative mode. The problem appears to be related to breakdown of orthogonality of the plane waves associated with the LO's in lopw.f and a patch is being tests. A patch to lopw.f appears to solve this problem in most cases, but it is not impossible that it can still occur in some cases.

With normal modes (e.g. PBE) the current mixer appears to be sufficiently robust that it can handle this problem in most cases tested so far. At the time of writing this (August 5, 2010) the switch –noHinv –it appears to be best (-vec2pratt is OK, maybe not so good). The mode where a single-precision approximation to the inverse of H inside lapw1 is stored (the default) does not appear to be cost effective; it reduces the time per scf iteration but this is offset by the number of extra cycles needed, and it does not converge as well. As a caveat, the performance of the different modes with a changed lopw.f has not yet been benchmarked. A few non-iterative cycles should probably be run at the end of a MSR1a calculation just to ensure that nothing strange has happened.

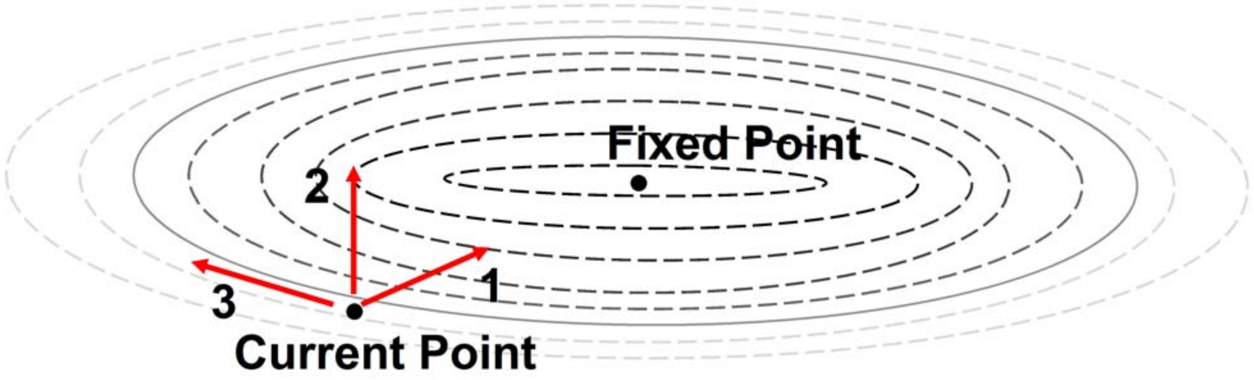
For weakly correlated systems, e.g. the Ti d-electrons in TiO_2 or $SrTiO_3$ compounds/surfaces the Atoms mode appears to work fine in iterative mode using PBE0 (-it –noHinv –vec2pratt). However, for strongly correllated systems such as a NiO surface using GGA+U or PBE0 the algorithm can fail due to the ghost forces if lapw1 has not been patched .

Dealing with Hard Problems

In some cases the convergence of the mixer is poor, independent of the mode used (MSEC1, MSEC3, MSR1), and the behavior of the atom movement algorithm MSR1a can be bad with severe oscillations, although this is less likely with version 5.3. This is not a consequence of a bad algorithm and may not be a consequence of a badly-posed problem, rather it can be intrinsic.

To expand a little, all methods both in Wien2k as well as in the literature use some variant of a expansion of the information from previous steps to form a Jacobian. Different algorithms do this in slightly different ways, but they are not as different in this as they might appear from the literature. Which directions are used to collect this information comes in large part from the residue, the difference between the input density at the start of an iteration and the new one after, the Pratt step. What matters is the properties of this step.

For problems which converge quickly this step is very strongly downhill; for problems which converge badly it is not. For the “good” problems which converge well the algorithm is rapidly obtaining information about the main directions towards the self-consistent solution. Opposite to this, for bad problems it is obtaining much less information about where the solution is, rather a lot of information about moderately useless directions. This is illustrated below: if the Pratt step is along 1 as marked the fixed-point solution will be found quickly; if along 2 it will still be found but a bit slowly; for 3 it will be slow.

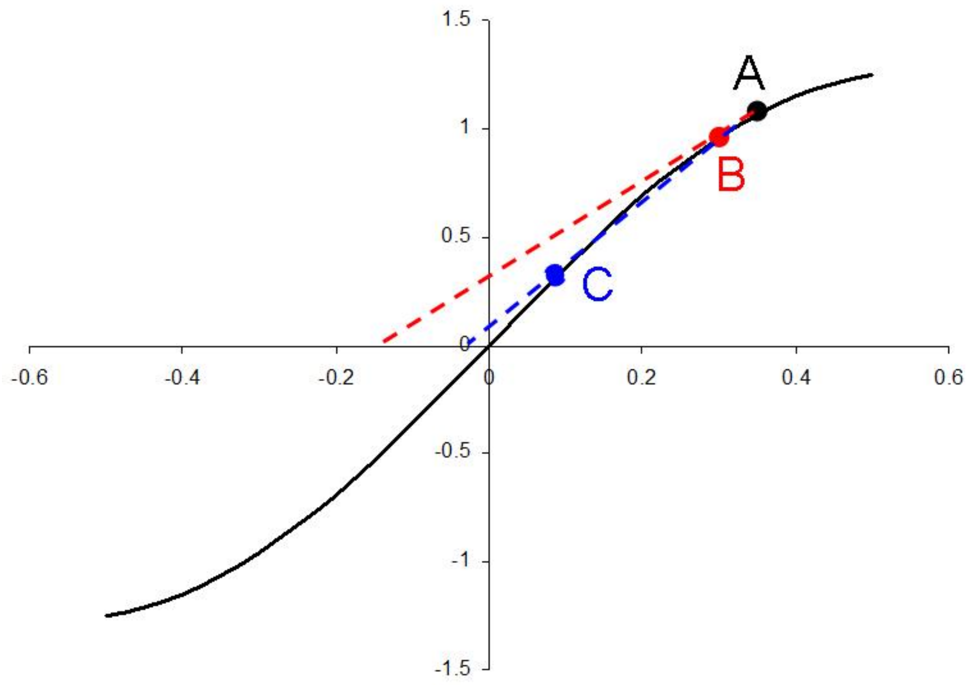


You can estimate what you have for a given problem by looking at the angles between the Broyden and Pratt steps which are printed in case.scfm. If these are near 90 degrees or are larger than this expect the convergence to be slow.

Changing from TETRA to TEMPS (or perhaps TEMP) is known, empirically, to make convergence better by improving the Pratt step – exactly how is not known. Similarly increasing the number of k-points for a metallic system seems to help, although here the effect may be a reduction of noise. Reducing the upper bound of the GREED can also be useful, perhaps 0.1 or in very bad cases 0.05. Reducing this more than this is normally a very, very bad idea as the algorithm will starve to death.

If you encounter very poor convergence, the first thing you should analyze is whether you have posed the problem correctly in terms of the physics; in many cases changing this solves all problems. It may be that for this type of problem it is better to use the double-loop method with PORT rather than MSR1a, certainly for inexperienced users. The second thing to due is check whether you are obtaining bad convergence because the GREED is too small, e.g. stuck around 0.025. Often increasing it for a step by doing "echo 0.05 > .msec" will cure this problem. Third you can consider reducing the maximum GREED to 0.1 or maybe 0.05, and if you are using MSR1a reduce the trust-region step by adding "TRUST 0.5" to case.inm and/or add SLOW or VSLOW. DO NOT REDUCE THE GREED TO 0.01 OR SMALLER AS SUGGESTED FOR OTHER DFT CODES. This is actually deeply wrong in all cases (even with other codes) due to some misconceptions in the literature about what the GREED really does. For other codes it might work (perhaps more by luck than mathematical rigor), for Wien2k it does not. (In fact reducing the GREED below 0.025 is ignored unless a line "DBASE XX" is included in case.inm – this is deliberate.)

To illustrate how too small a greed can be bad, consider the function below where we start from the point A and form numerical derivatives by taking a step to either B or C then use Newton's method to find the next candidate point along the x-axis – this is equivalent to a Broyden method. Obviously the larger step to C is better, and it is hard to generate similar problems where taking a very small step will diverge whereas larger steps converge. Note that of course taking very large steps can readily diverge, and this is a gross oversimplification.



Being slightly more formal, what we require is that the Dennis-Moré condition holds, which is equivalent to requiring that the errors in the estimate of the Jacobian or Broyden matrices are decreasing as the algorithm progresses, which requires steps which are neither too large nor too small.

9 MIXER (adding and mixing of charge densities)

In `mixer` the electron densities of core, semi-core, and valence states are added to yield the total new (output) density (in some calculations only one or two types will exist). Proper normalization of the densities is checked and enforced (by adding a constant charge density in the interstitial). As it is well known, simply taking the new densities leads to instabilities in the iterative SCF process. Therefore it is necessary to stabilize the SCF cycle. In *WIEN2k* this is done by mixing the output density with the (old) input density to obtain the new density to be used in the next iteration. Several mixing schemes are implemented, but we mention only:

1. straight mixing as originally proposed by Pratt (52) with a mixing greed Q

$$\rho_{new}(r) = (1 - Q)\rho_{old}(r) + Q\rho_{output}(r)$$

2. a Multi-Secant mixing scheme contributed by L. Marks (see Marks and Luke 2008), in which all the expansion coefficients of the density from several preceding iterations (usually 6-10) are utilized to calculate an optimal direction in each iteration. This version is by far superior to the other schemes making them quite obsolete. It is very robust and stable (works nicely also for magnetic systems with 3d or 4f states at EF, only for ill-conditioned single-atom calculations you can break it) and usually converges at least 30 % faster than the old BROYD scheme.
3. Two new variants on the Multi-Secant method including a rank-one update (see Marks 2013) which appear to be faster and equally robust

At the outset of a new calculation (for any changed computational parameter such as k-mesh, matrix size, lattice constant etc.), any existing **`case.broydX`** files must be deleted (since the iterative history which they contain refers to a ``different`` incompatible calculation).

If the file **`case.clmsum_old`** can not be found by **`mixer`**, a ``PRATT-mixing`` with mixing greed 1.0 is done.

*Note: a **`case.clmval`** file must always be present, since the LM values and the K-vectors are read from this file.*

The total energy and the atomic forces are computed in mixer by reading the **case.scf** file and adding the various contributions computed in preceding steps of the last iteration. Therefore **case.scf** must not contain a certain ``iteration-number'' more than once and the number of iterations in the scf file must not be greater than 999.

For LDA+U calculations **case.dmatup/dn** and for hybrid-DFT (switch -eece) **case.vorbup/dn** files will be included in the mixing procedure. With the new mode MSR1a (or MSECa) atomic positions will also be mixed (effectively optimized).

1 Execution

The program **mixer** is executed by invoking the command:

mixer mixer.def or **x mixer [-eece]**

A spin-polarized case will be detected automatically by **x** due to the presence of a case.clmvalup file. For an example see fccNi (sec. 10.2) in the *WIEN2k* package.

2 Dimensioning parameters

The following parameters are collected in file **param.inc**, :

NCOM	number of LM terms in density
NRAD	number of radial mesh points
NSYM	order of point group
traptouch	minimum acceptable distance between atoms in full optimization model

3 Input

Below a sample input (written automatically by **lstart**) is provided for TiO_2 (rutile), one of the test cases provided with the *WIEN2k* package.

```
------------------ top of file: case.inm --------------------
MSEC3 0.d0  YES  (PRATT/MSEC1 background charge (+1 for additional e), NORM
  0.2            MIXING GREED
```

```
1.0 1.0          Not used, retained for compatibility only
999 8           nbroyd nuse
------------------- bottom of file -----------------------
```

Interpretive comments on this file are as follows:

line 1:

(A5,*)
switch, bgch, norm

switch	MSEC1	Multi-Secant scheme (Marks and Luke 2008) (also the old BROYD switch points now to this method).
	MSEC2	Similar to MSEC1 (above), but mixes the higher LM values inside spheres by an adaptive PRATT scheme. This leads to a significant reduction of program size and file size (**`case.broyd*`**) for unit cells with many atoms and low symmetry (factor 10-50) with only slightly worse mixing performance.
	MSEC3	Similar to MSEC1, but with updated scaling, regularization and other improvements. For MSEC3a see later.
	MSEC4	Similar to MSEC3, but only mixes the L=0 LM value similar to MSEC2
	MSR1	Recommended: A Rank-One Multisecant that is slightly faster than MSEC3 in most cases. For MSR1a see later.
	MSR2	A variant of MSR1 which only mixes the L=0 values inside the spheres, similar to MSEC2
	PRATT	Pratt's scheme with a fixed greed
	PRAT0	Pratt's scheme with a greed restrained by previous improvement, similar to MSEC3
bgch		Background charge for charged cells (+1 for additional electron, -1 for core hole, if not neutralized by additional valence electron)
norm	YES	Charge densities are normalized to sum of Z
	NO	Charge densities are not normalized

line 2:

free format

greed	mixing greed Q. Essential for PRAT0 and PRATT, rather less important for Multisecent methods. Q is automatically reduced by the program depending on the average charge distance :DIS and the relative improvement in the last cycle. In case that the scf cycle fails due to large charge fluctuations, this can be further reduced but this can lead to stagnation. One should rarely reduce this below 0.05.

line 3 (optional):

(free format)
f_pw, f_clm

f_pw	Not used, retained for compatibility only.
f_clm	Not used, retained for compatibility only..

line 4 (optional):

(free format)
nbroyd, nuse

nbroyd	Not used, retained for compatibility only.
nuse	For all the Multisecant Methods: Only nuse prior steps are used (this value has some influence on the optimal convergence. Usually 6-10 seems reasonable and 8 is recommended).

Line 5 or more (optional), additional switches:

SLOW	Turns on some additional damping, for hard problems
PRATT	Uses PRATT mode with the step size control of MSR1
HYBR X	Changes the degree of greed in the multisecant to X, default 1.0. Values of 0.25-10.0. Smaller values are closer to MSEC3
TMAX X	Reduces the maximum step. For very hard cases X=1 or 0.5 may be useful. To be used with care as the algorithm may starve to death for small values.

Control files

.msec	The existence of the file .msec forces the next step to use a mixing greed which is read from the file. The code deletes the file if it is present. In some cases the algorithm might get trapped with small greed terms (e.g.

	0.01), and this allows it to be reset higher so avoiding Wolfe condition starvation.
.pratt	The existence of the file .pratt forces the next step to be a Pratt step, without a restart. If the file contains a number, this is the mixing greed used. The code deletes the file if it is present.
.push	Performs a single step with more scaling of atoms and orbital terms. Can be useful at the start when changing mode.
.Blimit	Increases the trust region to value in the file for the next iteration – do grep :TRUST case.scf
.Climit	Increases the charge trust region to the value in the file fir tge bext uteration – do grep :TRUST case.scf

Undocumented Options

DAMP	Damps the atomic motion. May lead to very slow convergence in hard cases
DBASE X	Changes the lowest **GREED** to X, typical value 0.025
FAST	Turns off some additional damping, not recommended in general.
LAMBDA XX	Fixes the regularization value to **XX**, for instance 1D-4
NODRI	Turns off automatic drift correction for non-centrosymmetric structures
NOSAFE	Turns of automatic switching into SAFE mode
RESTART X	Restarts from a converged density with an initial **GREED** of **X**
TRAD X	Changes the maximum atomic movement to X au, default 0.1
TRUST XX	Applies a tighter Trust-Region control, with XX default of 1.0. One can even relax the Trust-Region control by making XX larger or smaller. For expert use only
VERBOSE	Outputs additional information that may not be useful

Additional Experimental Controls (subject to change without notice)

CHESS	A different version of HESS, similarly sometimes better, sometimes not, needs work.
HESS	Experimental use of the Hessian created by pairhess. Sometimes better, sometimes not, needs work.
TESTX	Used during testing for various options, with 1 < X < 20.

Cases where MSR1a works well

1. Volume relaxation: forces converge cleanly as the density converges.
2. Change of RMTs by a small amount – similar to 1. Caveat, when changing RMTs by a lot it may be necessary to create a new density, as clminter is not good in such cases.
3. Change of RKMAX, e.g. after a minimization – similar to 1.
4. Change of functional after having done a previous run, e.g. moving to PBE0. Recommendation: use clmextrapol with appropriate lattice changes to accommodate any volume differences, in general it seems that a homogeeous expansion/contraction is the leading change.
5. Minimizations where a good scaling of the initial estimate in pairhess is not known, or when there are large changes in the Hessian during the minimization, e.g. bonding changes. In such a case PORT can take some time to lose the old Hessian information (a number of fully converged outer loops) whereas MSR1a will lose this old information in a few, the number of memory steps used.