I. STRATEGY AND TACTICS FOR NUMERICAL CONVERGENCE ACCELERATION

In order to solve self consistency problems often it is desirable to spent some memory and time to minimize the number of iteration steps. If the time to be spent for analyzing the available information after each iteration step is still negligible compared to the time one iteration step itself takes, it makes sense to spent it. An algorithm widely used in band structure calculations, which is based on the method of minimizing the mean square deviation, is appropriate for this. The power of this method lies in treating large equation systems with very time consuming evaluation of the right hand side.

Consider the n-dimensional fixed point problem

$$\vec{x} = \vec{G}(\vec{x}) \qquad \Longleftrightarrow \qquad \vec{f}(\vec{x}) = \vec{G}(\vec{x}) - \vec{x} = \vec{0}, \tag{1}$$

where the map is from the space X with elements $\vec{x}$ and $\vec{G}$ into the space F of deviation vectors $\vec{f}$. The fixed point problem is solved if the norm of the n-dimensional vector $\vec{f}$ is zero for a given $\vec{x}$. Because a norm is always ≥ 0 per decret, $\vec{x}$ has to be an absolute minimum for the norm function (not necessarily unique). The idea is to find the kernel of the map $\vec{x} \to \vec{f}$ by local linearization. One defines an $m-1$-dimensional hyper surface in F by linear interpolation between the last m vectors $\vec{f}_k$. Using informations from the last m iterations one determines in this hyper surface the vector $\vec{f}^{min}$ with smallest norm. The inverse of the linearized map (which has to be known of course, this limits our choice of the hyper surface) defines the element $\vec{x}^{min}$, which presumably has its function value close to $\vec{f}^{min}$. Because the norm of $\vec{f}^{min}$ is always smaller than all other norms of the remaining $\vec{f}_k$, the $\vec{x}^{min}$ represents an improvement compared to the last m points, provided $\vec{x}^{min}$ maps sufficiently close to $\vec{f}^{min}$. It is a matter of educated choice of $\vec{x}_k$ to define the $m-1$ dimensional hyper surface in such a way, that it is as close as possible to the origin. If the guess went down the hill, it doesn't matter, because then we just try another guess knowing already about the desaster (and, of course, avoiding it!). And if at the end of the day we found $\vec{f}^{min} \approx \vec{0}$ nobody is going to ask how often we had bad luck in the process of finding it. The main idea is due to Samuel Beckett:

Ever try, ever fail, no matter
try again, fail again, fail better.

To begin with something definite, we start with $m = 1$ and increase m with each iteration by 1, saving the information of each iteration step. At some point one realizes that the memory of the computer flows over. Then it is a good idea not to increase m anymore. So we only increase up to a maximal $m = m_{max}$. Once we arrived at this unpleasant situation (thats the general case), the method lives and dies with a not too unlucky choice of $\vec{x}_k$.

Lets assume, we somehow managed to obtain m pairs $(\vec{x}_m, \vec{f}_m)$, where $\vec{f}_m = \vec{f}(\vec{x}_m)$. Then we obtain our new $\vec{x}$ by the condition that at the $m-1$-dimensional hyper surface in F, given by

$$\vec{f}[g_l] = \vec{f}_m + \sum_{l=1}^{m-1} g_l(\vec{f}_l - \vec{f}_m), \tag{2}$$

the norm of the n-dimensional vector $\vec{f}$ has a local minimum:

$$\vec{f}[g_l]\vec{f}^*[g_l] \to min \qquad . \tag{3}$$

This leads to the following hermitian equation system for g_l (where m is fixed and plays only the role of the iteration index)

$$\sum_{l=1}^{m-1} g_l^{min}(\vec{f}_l - \vec{f}_m)(\vec{f}_k^* - \vec{f}_m^*) = -\vec{f}_m(\vec{f}_k^* - \vec{f}_m^*). \tag{4}$$

Now we realize that we need the vector $\vec{x}^{min}$ which maps on $\vec{f}^{min} = \vec{f}[g_l^{min}]$. The bad news is, we don't know it. The good news is, we can approximize it. We assume that the $m-1$ dimensional hyper surface in X, given by

$$\vec{x}[g_l] = \vec{x}_m + \sum_{l=1}^{m-1} g_l(\vec{x}_l - \vec{x}_m), \tag{5}$$

maps linearly into the corresponding hyper surface F. This is kind of plausible taking into consideration the fact, that all m generators $\vec{x}_k$ of the one hyper surface map onto the m generators $\vec{f}_k$ of the other hyper surface. In other words, we assume our map is locally linear. This assumption becomes gorgeous if we are very close to the fixed point. We define (remember that $\vec{x}^{min}$ maps only to the neighborhood of $\vec{f}^{min}$)

$$\vec{x}^{min} = \vec{x}_m + \sum_{l=1}^{m-1} g_l^{min}(\vec{x}_l - \vec{x}_m). \tag{6}$$

Because no pair without coupling, we define our new pair $(\vec{x}_{m+1}, \vec{f}_{m+1})$ by coupling back to the old one with a parameter p_m,

$$\vec{x}_{m+1} = \vec{x}^{min} + p_m\vec{f}^{min} \tag{7}$$

$$\vec{f}_{m+1} = \vec{f}(\vec{x}_{m+1}). \tag{8}$$

Now, after our tactics is clear, we have to find a strategy which tells us which $\vec{f}_k$ are the most relevant ones for our problem and how strongly do we couple back (p_m). The idea is, that we sort our $\vec{f}_k$ with increasing norm and, because everybody likes throwing away as much as possible, we throw away the pair with the largest norm in order to make place for the fresh calculated $(m+1)$th pair (remember that our computer memory was full). Lets just throw away also some of the pairs

with the larger norms if we did very well one step ago (if $|\vec{f}_{m+1}| \ll \min_{k\le m}|\vec{f}_k|$), in order to eliminate these misleading directions. Also if we did very badly one step ago ($|\vec{f}_{m+1}| > \max_{k\le m}|\vec{f}_k|$) we throw away some of the bad generators (but we shouldn't throw away everything we have because then nothing is left). If the rare case occurs that the new generator $\vec{f}_{m+1}$ degenerates with all m preceding generators, we don't give up. We include a weird (stochastic) step to enforce a new direction.

The guess for the updated back coupling parameter is an educated one as well. In order to go from $(\vec{x}^{min}, \vec{f}^{min})$ to $\vec{x}_{m+1}$, Eq. (7) gives for p_m

$$p_m = \frac{\left|\vec{x}_{m+1} - \vec{x}^{min}\right|}{\left|\vec{f}^{min}\right|}. \tag{9}$$

So we just chose next time a p_m which has a norm equal to that of $\tilde{p}_m$, where $\tilde{p}_m$ would have brought us from $(\vec{x}_m, \vec{f}_m)$ to $\vec{x}^{min}$ via the corresponding equation

$$\tilde{p}_m = \frac{\left|\vec{x}^{min} - \vec{x}_m\right|}{\left|\vec{f}_m\right|} = \frac{\left|\vec{x}^{min} - \vec{x}_m\right|}{\left|\vec{G}(\vec{x}_m) - \vec{x}_m\right|}. \tag{10}$$

If the norm of $\vec{f}_{m+1}$ is larger than that one of $\vec{f}^{min}$ we have chosen p_{m-1} too big the step before. In that case we scale the new p_m down with the factor $|\vec{f}^{min}|/|\vec{f}_{m+1}|$. To summarize, p_m follows from

$$p_m = \tilde{p}_m \cdot q_m \tag{11}$$

where the factor q_m is adjusted in every step,

$$q_{m+1} = \min\left(1, \frac{\left|\vec{f}^{min}\right|}{\left|\vec{f}_{m+1}\right|}\right). \tag{12}$$

Who likes to couple, can do this by coupling back with p_{m-1}. This is supposed to increase the stability during the iteration process. The quality of the method is shown in Fig. 1. In this case the vectors $\vec{x}$ are represented by functions $\Delta(\vec{R})$, $\sigma(\vec{R}, \epsilon)$ (gap function, impurity self energy) of the arguments $\vec{R}$ and ϵ. The dimension n is typically of the order of half a million, so its not done with an Abacus. One function call, $\vec{G}(\vec{x})$, took about 10 minutes CPU on 8 processors of a power challenge (after debugging the compiler, and forever before). Under this conditions it would be hopeless to start to calculate a Jacobi matrix. One could start, but apart from the fact that one never would finish, there is also no space left on the device. It is here, where the outlined method gives us back all our hope. For m_{max} we chose values smaller than 10. That's because we are nice and leave other people some space on our computer too.

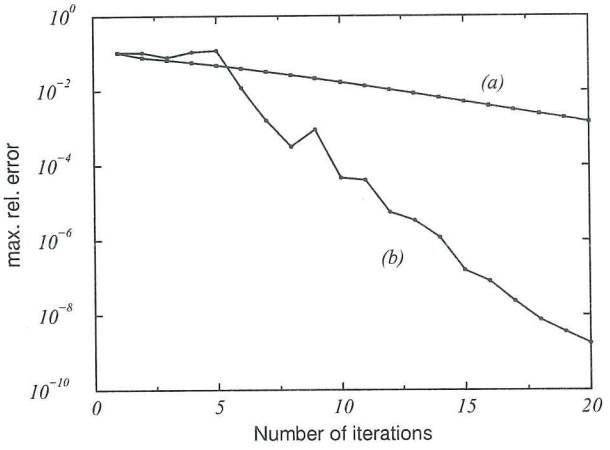

FIG. 1. Note the logarithmic scale. Shown is a comparison between convergence behavior of (a) simple iteration $\vec{x}_{m+1} = \vec{G}(\vec{x}_m)$ and (b) of the outlined method with $m_{max} = 5$. We iterated the self consistency condition of the equilibrium order parameter and impurity self energy of an s-wave vortex with 817 nodes. With nodes we mean here grid nodes. The mean free path was chosen 10 coherence lengths. Note again the logarithmic scale.