Function Minimization (Continued):

There also exist other, more powerful ***Gradient*** function minimization methods.

One elegant and powerful one is the **Variable Metric Method** (VMM):

This method utilizes concepts from ***differential geometry***, where an M-dimensional ***space*** is characterized by a ***metric*** $ds^2 = \underline{dx}^T \underline{A} \underline{dx}$. The ***components*** of the $M \times M$ ***metric tensor*** $\underline{A}$ transform in a ***covariant*** manner *w.r.t.* a change in coordinate systems, *e.g.* Cartesian/rectangular coordinates (x, y, z) to spherical-polar coordinates (r, θ, φ). {Please see Auxiliary P598AEM Lect. Notes on invariant/covariant/contravariant vectors/tensors if want/need further info on this…}.

In the VMM, we let the properties of the function $F(\underline{\lambda})$ (to be ***minimized***) define a ***metric tensor***. Assuming that $F(\underline{\lambda})$ is a ***scalar*** function, the ***second*** derivatives $\dfrac{\partial^2 F(\underline{\lambda})}{\partial\lambda_i \partial\lambda_j}$ are the ***components*** of a ***covariant tensor***, so we ***define*** the ***metric tensor*** as: $\left(\underline{H}_{\underline{\lambda}}\right)_{ij} \equiv \dfrac{\partial^2 F(\underline{\lambda})}{\partial\lambda_i \partial\lambda_j}$.

Then $ds^2 = \underline{dx}^T \underline{H}_{\underline{\lambda}} \underline{dx}$ is an ***invariant*** quantity.
(*n.b.* $\underline{\lambda}$ behaves like a ***contravariant*** vector under coordinate transformations.)

If $F(\underline{\lambda})$ is for example, the ***scalar*** $\chi^2(\underline{\lambda})$ function, then ds^2 is the (square of the) "***distance***" $\underline{\lambda} \to \underline{\lambda} + \underline{d\lambda}$, *i.e.* the (square of the) number of standard deviations ***difference*** between $\underline{\lambda}$ and $\underline{\lambda} + \underline{d\lambda}$.

As usual, we know that the ***covariance*** matrix for the $\underline{\lambda}$-parameters will be $\underline{V}_{\underline{\lambda}} = \underline{H}_{\underline{\lambda}}^{-1}$ (when at the ***minimum*** of $F(\underline{\lambda})$). The VMM uses the ***covariance*** matrix $\underline{V}_{\underline{\lambda}}$ during the function ***minimization*** process itself. Since $\underline{H}_{\underline{\lambda}}$ behaves as a ***covariant*** tensor under coordinate transformations, then $\underline{V}_{\underline{\lambda}}$ ($= \underline{H}_{\underline{\lambda}}^{-1}$ at the ***minimum*** of $F(\underline{\lambda})$) transforms as a ***contravariant*** tensor.

From $\underline{V}_{\underline{\lambda}}$ we can construct ***another invariant***: $\rho \equiv \underline{g}_{\underline{\lambda}}^T \underline{V}_{\underline{\lambda}} \underline{g}_{\underline{\lambda}}$, where $\underline{g}_{\lambda_i} \equiv \dfrac{\partial F(\underline{\lambda})}{\partial\lambda_i}$ is the ***gradient*** of the function $F(\underline{\lambda})$, and behaves like a ***covariant vector*** under coordinate transformations.

It turns out that $\rho \equiv \underline{g}_{\underline{\lambda}}^T \underline{V}_{\underline{\lambda}} \underline{g}_{\underline{\lambda}}$ is twice the difference between $F(\underline{\lambda})$ (evaluated at the same $\underline{\lambda}$ where $\underline{V}_{\underline{\lambda}}$ and $\underline{g}_{\underline{\lambda}}$ were calculated) and the ***minimum*** of the ***quadratic*** form whose coefficient matrix is $\underline{H}_{\underline{\lambda}} = \underline{V}_{\underline{\lambda}}^{-1}$.

Thus, $\frac{1}{2}\rho$ is the ***expected*** ("***vertical***") distance (= ***difference*** in $F(\underline{\lambda})$ values) to the ***minimum*** if the function $F(\underline{\lambda})$ were truly ***quadratic***. If we have ***approximations*** to the ***covariance*** matrix $\underline{V}_{\underline{\lambda}}$ and ***gradient*** $\underline{g}_{\underline{\lambda}}$, then we can quickly calculate $\frac{1}{2}\rho \equiv \frac{1}{2}\underline{g}_{\underline{\lambda}}^T \underline{V}_{\underline{\lambda}} \underline{g}_{\underline{\lambda}}$ and see if we have converged.

If $F(\underline{\lambda})$ ***were*** truly ***quadratic***, then $\underline{H}_{\underline{\lambda}}$ ***would*** be a ***constant*** (*i.e.* ***independent*** of $\underline{\lambda}$) and we ***would*** be in a space with a "***constant metric***". If $F(\underline{\lambda})$ is not (terribly far from) ***quadratic***, then we expect that $\underline{H}_{\underline{\lambda}}$ will be a ***slowly*** varying function of $\underline{\lambda}$, in which case we have a ***variable metric***.

In **Newton's Method**, $\underline{H}_{\underline{\lambda}}$ is recalculated at ***each*** step, typically a very time-consuming process. In the VMM, the ***covariance*** matrix $\underline{V}_{\underline{\lambda}} = \underline{H}_{\underline{\lambda}}^{-1}$ is ***estimated*** in each step by taking the $\underline{V}_{\underline{\lambda}}$ at the ***previous*** step and ***correcting*** it by applying ***information*** from the ***current*** step.

The "**Matrix Updating Formula**" (MUF) which does this job is different for different VMM's. The MUF procedure is as follows:

a.) Pick ***starting values*** $\underline{\lambda}^0$. Calculate the corresponding starting ***gradient*** $\underline{g}_{\underline{\lambda}}^0 \equiv \partial F(\underline{\lambda})/\partial\underline{\lambda}\big|_{\underline{\lambda}=\underline{\lambda}^0}$ and some ***approximation*** to the ***covariance*** matrix $\underline{V}_{\underline{\lambda}}^0$.
(*n.b.* $\underline{V}_{\underline{\lambda}}^0$ ***may*** be the full second derivative matrix, or simply the unit matrix $\underline{\mathbb{1}}$.)

b.) Take a ***step*** to: $\underline{\lambda}^1 = \underline{\lambda}^0 - \underline{V}_{\underline{\lambda}}^0 \underline{g}_{\underline{\lambda}}^0$.
{Compare this to: $\underline{\lambda} = \underline{\lambda}^0 - \underline{H}_{\underline{\lambda}}^{-1}\underline{g}_{\underline{\lambda}}$ in **Newton's Method,** P598AEM Lect. Notes 23, *p*. 6}.
This ***would*** be the ***exact*** minimum ***if*** $F(\underline{\lambda})$ were ***quadratic*** ***.and.*** if $\underline{V}_{\underline{\lambda}}^0$ were the ***true covariance*** matrix (and not an ***approximation***). {*n.b.* In a more elaborate method, we might look for a ***minimum*** along the "line" from $\underline{\lambda}^0$ to $\underline{\lambda}^1$ and call ***that*** point "$\underline{\lambda}^1$".}

Using $\underline{\lambda}^1$ we calculate the ***new gradient*** $\underline{g}_{\underline{\lambda}}^1 \equiv \partial F(\underline{\lambda})/\partial\underline{\lambda}\big|_{\underline{\lambda}=\underline{\lambda}^1}$.

c.) Now we: • ***correct*** the ***covariance*** matrix $\underline{V}_{\underline{\lambda}}$ using an ***updating formula***:

$$\underline{V}_{\underline{\lambda}}^1 = \underline{V}_{\underline{\lambda}}^0 + f\left(\underline{V}_{\underline{\lambda}}^0, \underline{\lambda}^0, \underline{g}_{\underline{\lambda}}^0, \underline{\lambda}^1, \underline{g}_{\underline{\lambda}}^1\right)$$

• ***replace*** $\underline{g}_{\underline{\lambda}}^0$ by $\underline{g}_{\underline{\lambda}}^1$, $\underline{V}_{\underline{\lambda}}^0$ by $\underline{V}_{\underline{\lambda}}^1$, $\underline{\lambda}^0$ by $\underline{\lambda}^1$

and then repeat steps *b*.) and *c*.) until the process ***converges***, within some specified tolerance.

One such function $f\left(\underline{V}_{\underline{\lambda}}^0, \underline{\lambda}^0, \underline{g}_{\underline{\lambda}}^0, \underline{\lambda}^1, \underline{g}_{\underline{\lambda}}^1\right)$, due to Davidon (**Comput. J.** 10, 406 (1968)) is:

$$\underline{V}_{\underline{\lambda}}^1 = \underline{V}_{\underline{\lambda}}^0 + \frac{\left(\underline{\delta} - \underline{V}_{\underline{\lambda}}^0\underline{\gamma}\right)\left(\underline{\delta} - \underline{V}_{\underline{\lambda}}^0\underline{\gamma}\right)^T}{\underline{\gamma}^T\left(\underline{\delta} - \underline{V}_{\underline{\lambda}}^0\underline{\gamma}\right)} \quad \text{where: } \underline{\delta} \equiv \underline{\lambda}^1 - \underline{\lambda}^0 \text{ and: } \underline{\gamma} \equiv \underline{g}_{\underline{\lambda}}^1 - \underline{g}_{\underline{\lambda}}^0$$

This version of f has some very nice properties, including the fact that if $F(\underline{\lambda})$ ***is truly quadratic***, then the ***approximate*** $\underline{V}_{\underline{\lambda}}$ can be shown to $\rightarrow$ the ***true covariance*** matrix after a number of iterations.

The VMM technique works well and is ***much*** faster than **Newton's Method**, since it does not have to recalculate $\underline{V}_{\underline{\lambda}}$ from scratch at each step. However, sooner or later computational ***round-off errors will*** accumulate, as will errors due to ***approximations*** . It then makes sense, at the end, to ***recalculate*** $\underline{V}_{\underline{\lambda}}$ from ***scratch*** in order to obtain computationally ***reliable*** error estimates on the $\underline{\lambda}$-parameters, rather than use the ***approximation*** to the ***covariance*** matrix that has been building up all along.

Convergence can be tested after each step by looking at:

$\underline{r} = \underline{V}^1_{\underline{\lambda}} \underline{g}^1_{\underline{\lambda}}$ (which is the distance the $\underline{\lambda}$-parameters will move in the ***next*** step, *i.e.* $\underline{\lambda}^* - \underline{\lambda}^1$)
or at: $\frac{1}{2}\rho \equiv \frac{1}{2}\underline{g}^T_{\underline{\lambda}} \underline{V}_{\underline{\lambda}} \underline{g}_{\underline{\lambda}}$, or at: $F\left(\underline{\lambda}^1\right)$ itself.

We expect $F\left(\underline{\lambda}^1\right) < F\left(\underline{\lambda}^0\right)$ of course. At the ***minimum*** $\underline{r} = 0$, since the ***gradient*** "$\underline{g}_{\underline{\lambda}} \equiv \partial F\left(\underline{\lambda}\right)/\partial \underline{\lambda} = 0$" ***defines*** the ***minimum***. (In "***real***" problems, $\underline{g}_{\underline{\lambda}}$ is never ***exactly*** = 0!) Finally, $\frac{1}{2}\rho \equiv \frac{1}{2}\underline{g}^T_{\underline{\lambda}} \underline{V}_{\underline{\lambda}} \underline{g}_{\underline{\lambda}} = 0$ at a minimum. The value of $\frac{1}{2}\rho$ is usually used as the criterion for ***convergence***, since it ***estimates*** how far $F\left(\underline{\lambda}\right)$ is from its value ***at*** the minimum; when $\frac{1}{2}\rho$ is smaller than a preset limit, the process is stopped. (In "***real***" problems, $\frac{1}{2}\rho$ is never ***exactly*** = 0!)

Function Minimization with Constraints:

Until now, we have only discussed cases where ***no constraints*** are placed on the $\underline{\lambda}$ ***-parameters*** (*e.g.* ***no*** restrictions on their ***allowed*** ranges). Constraints on the $\underline{\lambda}$-parameters can take many forms:

i.) $\left.\begin{array}{c} f_1\left(\underline{\lambda}\right) = 0 \\ f_2\left(\underline{\lambda}\right) = 0 \\ \vdots \end{array}\right\}$ **Constraint Equations**

ii.) $\left.\begin{array}{c} a_1 \le \lambda_1 \le b_1 \\ a_2 \le \lambda_2 \le b_2 \\ \vdots \end{array}\right\}$ **Simple Constant Limits**

iii.) $\left.\begin{array}{c} r_1\left(\underline{\lambda}\right) \le \lambda_1 \le s_1\left(\underline{\lambda}\right) \\ r_2\left(\underline{\lambda}\right) \le \lambda_2 \le s_2\left(\underline{\lambda}\right) \\ \vdots \end{array}\right\}$ **Simple Variable Limits**

iv.) $\left.\begin{array}{c} u_1\left(\underline{\lambda}\right) \le v_1\left(\underline{\lambda}\right) \le w_1\left(\underline{\lambda}\right) \\ u_2\left(\underline{\lambda}\right) \le v_1\left(\underline{\lambda}\right) \le w_2\left(\underline{\lambda}\right) \\ \vdots \end{array}\right\}$ **Implicit Variable Limits**

We shall consider techniques that modify $F\left(\underline{\lambda}\right)$ in such a way that the ***ordinary unconstrained*** χ^2 minimization methods will be applicable.

For ***i*.) Constraint Equations**, we have already seen how "***Lagrange Multipliers***" $\underline{\alpha}$ can be introduced. Alternatively, we have also shown that if we can solve each equation for ***one*** of the $\underline{\lambda}$-parameters explicitly, *e.g.* λ_k, then we can ***eliminate*** that specific $\underline{\lambda}$-parameter.

For ***ii*.) Constant Limits**, a useful technique here is to ***change variables***. The idea is to ***replace*** the M $\underline{\lambda}$-parameters by M $\underline{\eta}$-parameters, where the $\underline{\eta}$-parameters are ***functions*** of the $\underline{\lambda}$-parameters, *i.e.* $\underline{\eta} = \underline{\eta}(\underline{\lambda})$ with the property that when λ_k takes on values from a_k to b_k, then η_k takes on values from $-\infty$ to $+\infty$. The function $\underline{\eta}(\underline{\lambda})$ should ***not*** introduce any new minima in χ^2 and should also be "smooth" so that it does not severely ***distort*** or add any ***weird features*** to the existing problem.

Of course, once the problem has been solved in $\underline{\eta}$ space, the ***covariance*** matrix $\underline{V}_{\underline{\eta}(\underline{\lambda})}$ of the $\underline{\eta}$-parameters (as well as the $\underline{\eta}$-parameters themselves) must be transformed back to $\underline{\lambda}$ space.

Examples:

To limit $0 \le \lambda_k \le \infty$, *e.g.* choose: $\lambda_k = \eta_k^2$ or: $\lambda_k = e^{\eta_k}$

To limit $0 \le \lambda_k \le 1$, *e.g.* choose: $\lambda_k = \sin^2 \eta_k$ or: $\lambda_k = \dfrac{e^{\eta_k}}{e^{\eta_k} - e^{-\eta_k}}$

To limit $a_k \le \lambda_k \le b_k$ *e.g.* choose: $\lambda_k = a_k + b_k \sin^2 \eta_k$, *etc.*

Note that the π-periodicity of the $\sin^2 \eta_k$ function implies that a particular value of λ_k^* corresponds to ***many*** η_k^*, each differing by π from each other. This means that while searching in η_k space to minimize χ^2**,** the η_k-steps should be small enough so that the search doesn't ***ever*** jump from the neighborhood of η_k to $\eta_k \pm \pi$, or else the procedure will become ***very*** confused...

This procedure can be extended in many ways to more complex constraints. For example, to minimize $fcn(\lambda_1, \lambda_2)$ with the requirement that $0 \le \lambda_1 \le \lambda_2 \le \infty$ we could define $\lambda_1 = \eta_1^2$, $\lambda_2 = \eta_1^2 + \eta_2^2$. ***This enforces the above requirement.***

For ***iii*.) Simple Limits** and ***iv*.) Variable Limits**, a useful technique employs a "***Penalty Function***". Instead of transforming the ***variables*** in $F(\underline{\lambda})$ as we did above, we modify $F(\underline{\lambda})$ itself in such a way that it is ***forced*** to become very large where we ***don't*** want $\underline{\lambda}$ to go, but is unaffected for values of $\underline{\lambda}$ that ***are*** allowed.

For example, in order to ***minimize*** $fcn(\underline{\lambda})$ with the ***additional constraint*** $p(\underline{\lambda}) \ge 0$, we can define a "penalty function" $T(\underline{\lambda})$ as:

$$T(\underline{\lambda}) = \begin{cases} 0 & \text{if } p(\underline{\lambda}) \ge 0 \\ c[p(\underline{\lambda})]^2 & \text{if } p(\underline{\lambda}) < 0 \end{cases}$$

where c is some "***large***" ***positive*** number, *i.e.* ***large*** compared to ***any*** value of $F(\underline{\lambda})$ in the region being searched near the ***minimum*** of $F(\underline{\lambda})$.

We then ***minimize*** the function $F'(\underline{\lambda}) \equiv F(\underline{\lambda}) + T(\underline{\lambda})$, *i.e.* we ***minimize***:

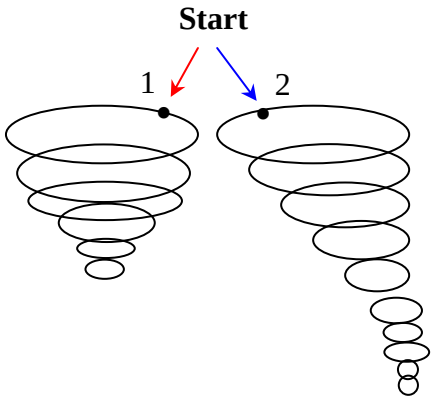
$$F'(\underline{\lambda}) = \begin{cases} F(\underline{\lambda}) & \text{if } p(\underline{\lambda}) \geq 0 \\ F(\underline{\lambda}) + c[p(\underline{\lambda})]^2 & \text{if } p(\underline{\lambda}) < 0 \end{cases}$$

Thus, $F'(\underline{\lambda})$ will be ***continuous*** at the "***boundary***" of the ***constraint***, *i.e.* where $p(\underline{\lambda}) = 0$.

And, as soon as $\underline{\lambda}$ "wanders" into a region where $p(\underline{\lambda}) < 0$, $F'(\underline{\lambda})$ gets large, and so $\underline{\lambda}$ will wander right back out!

Finding Multiple Minima and the Global Minimum:

Although the physical problem that motivates finding a minimum is ***assumed*** to have "***a*** minimum point", *i.e.* a "***best***" solution, the function we use to solve the problem may in fact have ***several*** (local) minima. Ideally, a "solution" should find ***all*** of the minima, leaving us free to think about the results and choose the one that is the "***best***" answer to ***our*** problem. Any of the techniques that we have discussed so far will (at best) find only ***one*** minimum, which may or may not be the ***Global*** minimum, and may not even be the one closest to our starting point.



For example, if we start at "1" we will probably find the left hand shallow minimum, and if we start at "2" we will probably find the deeper right hand one. But the first step ***may*** cross over to find the other minimum!

For physical problems we often assume that the minimum we want is the one nearest the starting values of some (or all) of the parameters (*i.e.* we have foreknowledge of, or approximate values for one or more of them.) It is not even common practice to ***demand*** that the minimum we want be the ***Global*** minimum. It is rare to ask for, and virtually impossible to find, ***all*** of the minima of a complicated function of many parameters if only a ***limited*** amount of computer time is available. Unfortunately, no methods in common use today are ***guaranteed*** to work for finding several minima.

In principle, a "grid search" ***could*** be made in which a grid of ***starting values*** $\underline{\lambda}^0$ was used. For each starting value, a complete minimization could be done. If the grid were fine enough, then we'd ***probably*** find ***all*** of the minima of $F(\underline{\lambda})$ in the region searched. But the "cost" would be high!

One practical possibility is to do a Monte Carlo search once $\underline{\lambda}^*$ has been found. We look at $F(\underline{\lambda})$ at many other points and see whether $F(\underline{\lambda}) < F(\underline{\lambda}^*)$ anywhere, which would prove that $\underline{\lambda}^*$ was ***not*** the ***Global*** minimum. A new minimization could then begin at *that* $\underline{\lambda}$. This will eventually find deeper and deeper minima (if any), but will ***not*** find any *shallower* ones, if these exist/are present.

Some analytic tools have been developed for finding ***other*** minima. For example, since the first derivatives $\partial F\left(\underline{\lambda}\right)/\partial\underline{\lambda}\big|_{\underline{\lambda}=\underline{\lambda}^*}$ vanish at the ***minimum*** of the function $F\left(\underline{\lambda}\right)$, at $\underline{\lambda}=\underline{\lambda}^*$, then ***near*** the minimum, the Taylor series expansion is:

$$F\left(\underline{\lambda}\right)=F\left(\underline{\lambda}^*\right)+\frac{1}{2}\left(\underline{\lambda}-\underline{\lambda}^*\right)^T \underline{H}_{\underline{\lambda}}\left(\underline{\lambda}-\underline{\lambda}^*\right) + \text{ higher-order terms}$$

Now if ***other minima are*** present in $F\left(\underline{\lambda}\right)$, the higher order terms in the Taylor series expansion ***will*** be important, since ***information*** about the other minima ***must*** be contained in these higher-order terms.

The trick, then is to transform $F\left(\underline{\lambda}\right)$ such that it has ***no minimum*** at $\underline{\lambda}=\underline{\lambda}^*$, but it still retains the ***information*** contained in the higher-order terms – *e.g.* define:

$$F_1\left(\underline{\lambda}^*,\underline{\lambda}\right)\equiv\frac{2\left[F\left(\underline{\lambda}\right)-F\left(\underline{\lambda}^*\right)\right]}{\left(\underline{\lambda}-\underline{\lambda}^*\right)^T \underline{H}_{\underline{\lambda}}\left(\underline{\lambda}-\underline{\lambda}^*\right)}$$

Near $\underline{\lambda}^*$, $F_1\left(\underline{\lambda}^*,\underline{\lambda}\right)\simeq 1 +$ higher-order terms, so it (presumably) has ***no minimum*** at $\underline{\lambda}=\underline{\lambda}^*$. Then we can look for a ***minimum***/look for ***minima*** associated with $F_1\left(\underline{\lambda}^*,\underline{\lambda}\right)$....

Methods such as this ***have*** been found to work in ***some*** cases, but their ***general*** validity has not yet been established…

The Metropolis-Hastings Algorithm and Simulated Annealing:

The Metropolis-Hastings algorithm [N. Metropolis, *et al.*, J. Chem. Phys. **21** (6): 1087-92 (1953); W.K. Hastings, Biometrika **57** (1): 97-109 (1970] is a Markov chain (*i.e.* random-walk) Monte Carlo algorithm used for obtaining a sequence of random samples from a function $F\left(\underline{\lambda}\right)$ for which direct sampling may be difficult. It was first proposed for use with the Boltzmann distribution $f\left(E;T\right)\propto e^{-E(T)/k_BT}$. The M-H random walk-type algorithm can be helpful, when used in combination with a function minimization algorithm, such as the Simplex Method, to efficiently do so, and hopefully find the ***Global minimum*** of $F\left(\underline{\lambda}\right)$.

The sequence of random samples of $F\left(\underline{\lambda}\right)$ can also be used *e.g.* to ***approximate*** the function $F\left(\underline{\lambda}\right)$ or to compute an ***integral*** of the function $F\left(\underline{\lambda}\right)$ – *i.e.* a C.D.F., or an expectation value, …

The M-H algorithm has been used to solve the famous ***Traveling Salesman Problem*** – minimizing the total round-trip path for visiting a sequence of cities only once {there are even websites where you yourself can solve your own trip, *e.g.* http://www.gebweb.net/optimap/}. The M-H algorithm has also been used for VLSI chip design, PC board layout – routing traces to minimize their length, using *e.g.* Lagrange multipliers in order to minimize the # PC board jumpers, ***etc***. The M-H algorithm has also found use(s) in calculating many-body spin configurations, DNA & genome sequencing...

The strategy employed by coupling the Metropolis-Hastings algorithm *e.g.* to the Simplex function minimization method is inspired by the microscopic thermodynamical physics associated with the ***slow*** (***i.e.*** adiabatic) ***cooling*** of a heated metal – *i.e.* the ***annealing*** of the metal.

At high temperatures, the number of local microstates available to be "explored" by an atom of the metal is large – due to the high local energy density – which is proportional to the temperature *T* of the metal. The atoms of the metal would all like to be in their lowest possible energy states (thereby maximizing the overall entropy of the system) but ***sometimes***, due to temporal and spatial fluctuations in the (local) thermal energy density, individual atoms ***may*** (momentarily) find themselves in a higher energy state.

As the metal slowly cools, the number of local microstates available to be "explored" by an atom slowly decreases due to the decrease in thermal energy; the thermal fluctuations are also reduced.

For an adiabatically slow cooling process, the atoms in the metal eventually find their lowest energy state, forming a giant single crystal if no defects are present, at the end of the cooling process.

At $T = 0$, the "***internal energy***" E associated with the (scalar) function $F\left(\underline{\lambda}\right)$ to be ***minimized*** is the value of the function itself, *i.e.* $E\left(T=0\right) = F\left(\underline{\lambda}\right)$, at the {*M*-dimensional} "space" point $\underline{\lambda}$.

The simulated ***annealing*** procedure of the Metropolis-Hastings + Simplex method starts off at a "high" temperature T_{max} {which hopefully is $T_{max} > E_{max}/k_B = F_{max}\left(\underline{\lambda}\right)/k_B$ }. Thus, at ***finite*** temperature *T*, the "***metal***" {the ***ensemble*** of {continuum} values of the {spatial} $\underline{\lambda}$-parameters associated with the (scalar) function $F\left(\underline{\lambda}\right)$ } must have an ***average*** thermal energy ***density*** $\Delta u_{th}\left(T\right)$ and also have thermal ***fluctuations*** in the ***local*** energy ***density*** $\delta u_{th}\left(T,\underline{\lambda}\right)$ associated with it.

Recall that the ***simplex*** function minimization replaces the *M*-dimensional "space" point $\underline{\lambda}$ by an $M+1$ dimensional ***simplex*** – *e.g.* a ***triangle*** in 2-D or a ***tetrahedron*** in 3-D, *etc*.

We thus add a ***positive***, ***logarithmically-distributed random variable***, proportional to the temperature *T*, to the stored function value $F\left(\underline{\lambda}\right)$ associated with ***every*** vertex of the ***simplex***:

$$E\left(T,\underline{\lambda}\right) \equiv F\left(\underline{\lambda}\right) + \Delta U_{th}\left(T\right) + \delta U_{th}\left(T\right)$$

Note that since the thermal contribution $\Delta U_{th}\left(T\right) + \delta U_{th}\left(T\right) = Vol \times \left[\Delta u_{th}\left(T\right) + \delta u_{th}\left(T\right)\right]$ to the internal ***energy*** is ***logarithmically-distributed***, then the corresponding changes in the (local) Maxwell-Boltzmann probability density function $f\left(E;T\right) \propto e^{-E(T)/k_B T}$ will be ***linear***.

In analogy to the annealing process of a real metal, if the simulated thermodynamical system is offered a succession of options, we assume that it will (on average) change its configuration from energy $E_1 \equiv E\left(T,\underline{\lambda}_1\right)$ to $E_2 \equiv E\left(T,\underline{\lambda}_2\right)$ with probability $p = e^{-\Delta E/k_B T} = e^{-\left(E_2 - E_1\right)/k_B T}$.

However, note that if $E_2 < E_1$, then $p = e^{-(E_2 - E_1)/k_B T} > 1$. In order to prevent the overall function minimization process of this algorithm from being "***too greedy***" – *i.e.* finding the minimum of the function $F(\underline{\lambda})$ ***too*** rapidly – thus "***quenching***" the metal instead of "***annealing***" it {and thus increasing the probability of the algorithm getting "trapped" in a local minimum}, the M-H algorithm ***prevents*** $p = e^{-(E_2 - E_1)/k_B T}$ from exceeding 1 by ***clamping*** $p = 1$. This information is then passed to the ***simplex***, informing it that going from $E_1 \equiv E(\underline{\lambda}_1)$ at $\underline{\lambda}_1$ to $E_2 \equiv E(\underline{\lambda}_2)$ at $\underline{\lambda}_2$ is a ***good*** choice, but not (***way***) too good of a choice... The ***simplex*** will thus ***always*** accept *this* $p = 1$ move.

The $M+1$ dimensional ***simplex*** then does its work moving around in the M-dimensional space of the $\underline{\lambda}$-parameters, reflecting, expanding and/or contracting as it evaluates the function $F(\underline{\lambda})$ at the current $M+1$ vertex points of the ***simplex***.

However, for every ***new*** $\underline{\lambda}$-point that the ***simplex*** tries, the M-H algorithm ***subtracts*** a ***logarithmically-distributed random variable*** $\delta E_{th}(T)$, proportional to the temperature *T*, from the stored energy value $E(T, \underline{\lambda})$ associated with that ***new*** $\underline{\lambda}$ vertex-point of the ***simplex***.

Since $p = 1$ for $E_2 < E_1$, then $p \lesssim 1$ for $E_2 \gtrsim E_1$, which will ***sometimes*** encourage the simplex to investigate/probe the $\underline{\lambda}$-parameter space where fluctuations in the local thermal energy density have ***somewhat*** disfavored it.

Thus, because of the ***subtraction*** of thermal energy $\delta U_{th}(T)$ to each ***new*** point of the simplex, in order to ***encourage*** the simplex to ***continue*** with its function-minimization mission, the temperature *T* ***must*** be (slowly/adiabatically) reduced after a period of allowing the simplex to do its minimum-finding thing for a while – this ***temperature reduction*** is known as the ***annealing***, or ***cooling schedule*** for this algorithm.

Note that the "size" of the ***simplex*** in $\underline{\lambda}$-parameter space is proportional to the temperature *T*. At "high" temperatures, the ***simplex*** expands to a "big" scale, able to "freely" sample/range over much of the $\underline{\lambda}$-parameter space. It then executes a stochastic, tumbling-type of Brownian motion within that region, sampling new, approximately random points as it does so. The ***efficiency*** with which a region of the $\underline{\lambda}$-parameter space is explored is independent of its narrowness and its orientation. If the temperature T is reduced ***sufficiently slowly*** (*i.e.* allowing the "metal" to ***anneal***), the size of the ***simplex*** also "shrinks" commensurately, enabling it to fit into/sample various local minima and thus also give it a good chance of finding the ***Global minimum*** of the function $F(\underline{\lambda})$.

As one might anticipate, some non-trivial optimization of parameter(s) is required for the M-H + ***simplex*** algorithms in order to successfully find the Global minimum of the function $F(\underline{\lambda})$.

Here again, a "one-size-fits-all" approach will ***not*** be optimal for minimizing ***all*** functions $F\left(\underline{\lambda}\right)$.

- The ***starting temperature*** T_{max} ***relative*** to the function $F\left(\underline{\lambda}\right)$ must be determined/optimized, *i.e*. the numerical value of Boltzmann's constant k_B must be "tuned" since it is the proportionality constant between E and T: $E = k_B T$
- The ***scale*** of the (***mean***) ***thermal energy*** $\Delta U_{th}\left(T\right)$ and (***logarithmic***) ***random fluctuations*** in ***thermal energy*** $\delta U_{th}\left(T\right)$ relative to the ***dynamic range*** of numerical value(s) of the function $F\left(\underline{\lambda}\right)$ must be determined/optimized, ***.and.*** the commensurate ***annealing/cooling schedule*** must also be determined/optimized.
- The ***temperature dependence*** of the simplex ***step size*** $\delta\underline{\lambda}_{th}\left(T\right)$ must also be optimized.

There likely will be detailed issues with "sufficiently slowly" ***cooling/annealing*** of the "metal" and the optimization of the ***annealing/cooling schedule***, as success and/or failure of this function minimization is quite strongly dependent on the details of the ***annealing/cooling schedule***.

Some algorithmic ***annealing/cooling schedule*** possibilities to try here are:

- Reduce the temperature T to $\left(1-\varepsilon\right)T$ every N moves of the simplex, where ε/N is empirically-determined – *i.e*. optimize ε/N by trial & error…
- Budget a total of N_{tot} moves of the ***simplex***, and reduce T after every N moves to a value $T = T_{max}\left(1-N_{cum}/N_{tot}\right)^{\alpha}$ where N_{cum} is the cumulative # simplex moves thus far and the exponent α is a constant ranging from $1 \lesssim \alpha \lesssim 4$. Thus, the parameters N_{cum} and α need to be determined/optimized.
- After a set of N moves, set $T = \beta\left(F\left(\underline{\lambda}_1\right) - F\left(\underline{\lambda}_{best}\right)\right)$ where β is an empirically-determined (*i.e*. optimized) constant of $O\{1\}$ and $F\left(\underline{\lambda}_1\right)$ is the smallest current value of the ***simplex***, whereas $F\left(\underline{\lambda}_{best}\right)$ is the best-ever encountered value of the ***simplex***. However, ***never*** allow the temperature T to be reduced by ***more*** than some ***small*** fraction $\gamma \lesssim 1$ at a time.

Another (sometimes) useful thing to do is to allow an occasional ***complete restart*** of the M-H + Simplex function minimization algorithm {making certain that the ***best-ever*** encountered value of the ***simplex*** is ***not*** currently in the ***restart*** of the ***simplex***!}.

For ***some***, but not ***all*** function minimization problems, when the temperature T has been reduced by a factor of ~ 3× from its initial starting value, a ***restart can*** be beneficial/helpful.

The figure below shows the results of three random-walk Markov chains (yellow, green and blue) running on the ***3-D Rosenbrock function***

$$F\left(x,y,z\right) = \left(1-x\right)^2 + \left(1-y\right)^2 + 100\left(y-x^2\right)^2 + 100\left(z-y^2\right)^2$$

using the Metropolis-Hastings algorithm.

The M-H algorithm samples regions of $F(x, y, z)$ and the Markov chains begin to mix in these regions. The approximate position of the minimum has been illuminated – as white light. Note that the red points are the ones that remain after a "burn-in" process. The earlier ones (yellow, green and blue) have been discarded.