

## Re: Step Control for Gradient Descent ?

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2009-02/msg02454.html>

---

- *From:* [rgvickson@xxxxxxxxxx](mailto:rgvickson@xxxxxxxxxx)
  - *Date:* Wed, 18 Feb 2009 15:31:41 -0800 (PST)
- 

On Feb 18, 1:48 pm, dave.rud...@xxxxxxxxxx wrote:

Hi all,

I have a constrained global optimization problem to solve. Some constraints are sometimes based on equality, e.g.,  $a = f(b)$ . Others are inequality constraints, such as  $c \geq f(d)$ . So, life can be quite difficult.

We have a naive implementation of a minimizer for this problem. It simply uses a gradient descent algorithm, i.e.

$$x[n+1] := dt * \text{grad}(f(x[n])) + x[n]$$

and after each step it manually enforces any constraints that have been violated, i.e.

$$\begin{aligned} a &:= f(b) \\ \text{if}(c < f(d)) \ c &:= f(d) \end{aligned}$$

How large is the problem? (That is, how many variables and constraints do you have?) If it is small (say, no more than 200–300 variables and 100–200 constraints) you can try using readily-available software, such as the built-in EXCEL Solver. Alternatively, if the problem is much smaller, you could try using the latest Global optimization package in the LINGO solver, which is available from the LINDO Corp. I think that limited-capacity "student" versions are available for free, while larger-capacity implementations are available for a modest cost—or much larger versions for around \$1000. If you are at a university that has a site licence for some optimization packages, why not try using them?

Why am I making an issue of this? Well, there are two aspects to your problem: (1) getting a local optimum in reasonable time and with reasonable accuracy; and (2) getting a global optimum. Let's just look at (1) for the moment. It has long been known through examples that simple gradient searches (of the type you seem to be using) are dangerous: you can have convergence to a NON-OPTIMAL point.

## Re: Step Control for Gradient Descent ?

Furthermore, even in the absence of constraints, the method can be incredible slow, and can even converge to a non-optimal point when using finite-wordlength arithmetic on a real computer (even though it converges theoretically if one uses infinite-precision computations). That is why numerous teams of researchers working for decades have devoted so much effort to dealing with such problems. Typical, powerful implementations would include better unconstrained optimization methods (such as conjugate-gradient or quasi-Newton methods) having quadratic convergence or better. And there is the important issue of how to handle constraints, especially of inequality type. There a typical, powerful approach is the reduced-gradient or generalized reduced-gradient method, perhaps with modern "augmented Lagrangian" methods. For example, some algorithms (such as MINOS) for nonlinearly-constrained problems use generalized reduced-gradient methods to handle linear constraints and augmented lagrangians to deal with nonlinear constraints.

As I said, teams of researchers have spent decades developing powerful methods to handle such problems—because real-works problems are too hard to be handled naively—and have come up with techniques that will be orders of magnitudes better than what you are attempting. Why re-invent the wheel, and ineffectively at that? Even a tool such as the EXCEL Solver (also available on open-source<sup>3</sup> spreadsheets) embody much of this advanced research, and utilize quasi-Newton methods together with generalized reduced-gradient methods. The LINGO package uses similar recent research results.

Next, there is the issue (2): the question of global optimization. It would likely pay you to read some of the extant research on this topic; even doing a Google search will turn up a lot of useful material free of charge. It is still a difficult problem, and without some special structure to exploit, one can probably never be guaranteed that a true global optimum has been reached. There are, however, numerous effective heuristics that seem to have good performance. Often one must apply (local) optimization packages numerous times, perhaps using several randomly-generated starting points. Other techniques, such as simulated annealing, taboo search or genetic algorithms are brought to bear on such problems.

You might get better information and advice by posting your message to the Operations Research newsgroup 'sci.op-research'. The majority of postings over there deal with issues related to optimization.

Good luck.

R.G. Vickson  
Adjunct Professor, University of Waterloo

## Re: Step Control for Gradient Descent ?

To find the global minimum, we just do the above local search on several places.

Yes, this is a pretty weak method, but it actually works quite well for our purposes. The main problem is that it is REALLY slow, as you might guess.

You might think that we could reduce the number of drop points to speed things up, but in practice, we need relatively few of these drop points. Like, 5.

However, each run with the gradient descent method could take thousands of iterations to converge to something reasonable. So, a better way to speed it up would be to decrease the number of iterations. At the moment, we are evolving the solution with a constant step size ( $dt$ ).

I have heard rumours of ways to derive an adaptive step-size, much like you can with numerical integrators. However, I haven't seen the details of how this is done. Can someone point me in the right direction?

Alternatively, what other method might be useful for doing this kind of minimization? The shape of the energy field is such that there is one deep parabolic canyon with a few small craters along the way. Part of the reason we like our crappy little solver is that we can prevent it from getting caught in a crater way up the side of the wall by starting each local search near the estimated bottom of that canyon. So, if there is a method with similar properties, I would be especially interested.

Thanks for reading.

Dave