A Numerical Analyst thinks about "Deep Learning" and Artificial Intelligence

by Prof. W. Kahan (Retired),
Math. Dept., and
Elect. Eng. & Computer Sci. Dept.,
University of California @ Berkeley

prepared for the Celebration of
U.C. Berkeley's Turing Awardees
7 Nov. 2018

This revised version is now posted at www.eecs.berkeley.edu/~wkahan/7Nov18.pdf

A Numerical Analyst thinks about "Deep Learning" and Artificial Intelligence

Abstract:

We speak such different languages. Here are rough translations:

Artificial Intelligence	≈	Wishful Thinking
Deep Learning	≈	Pattern Matching Approximated
To Train Deep Learning	≈	Seek Convolutions' Coefficients
To Test Deep Learning	≈	Try to detect Erroneous Training
Overfit	≈	Too good to be true
One "Epoch" during Training	≈	One Full Iteration-Step of the search
Learning Rate	≈	Stepsize
Hyper-parameter	≈	Unknowable constant sought by trial

Why might practitioners of D.L. and A.I. benefit from an acquaintance with concepts like "Ill-Condition" familiar to Numerical Analysts ?
Because Ill-Condition limits the speed and accuracy of (re)training.

The Ultimate Artificial Intelligence?

When computers get fast enough and have enough memory, will one be programmed to fool ***everybody*** into believing there is a human inside the box?

To what extent must our human senses and actuators, like smell and fingers, be imitated by any robot before it could imitate also the way we think?

If Rodin's Thinker were a computer programmed to fool us into believing it could really think like us, what would it be thinking about?

(What about the "*Singularity*" ? Ask me later.)

If it too nearly resembles us, ...

I think Artificial Intelligence is still beset by Defects and Difficulties that are more immediate than its threat to our belief in our intellectual primacy.

Some Current Defects and Difficulties of
not all
Artificial Intelligence and Deep Learning

A.I. Strategy:

Tedious Numerical Details:

◊1 A disinclination to admit "I don't know".

Comparing a computer's to a human's thinking (@ London Math. Soc. ***1947***), Alan Turing said …

"I would say that fair play must be given the machine.

. . .

In other words then,

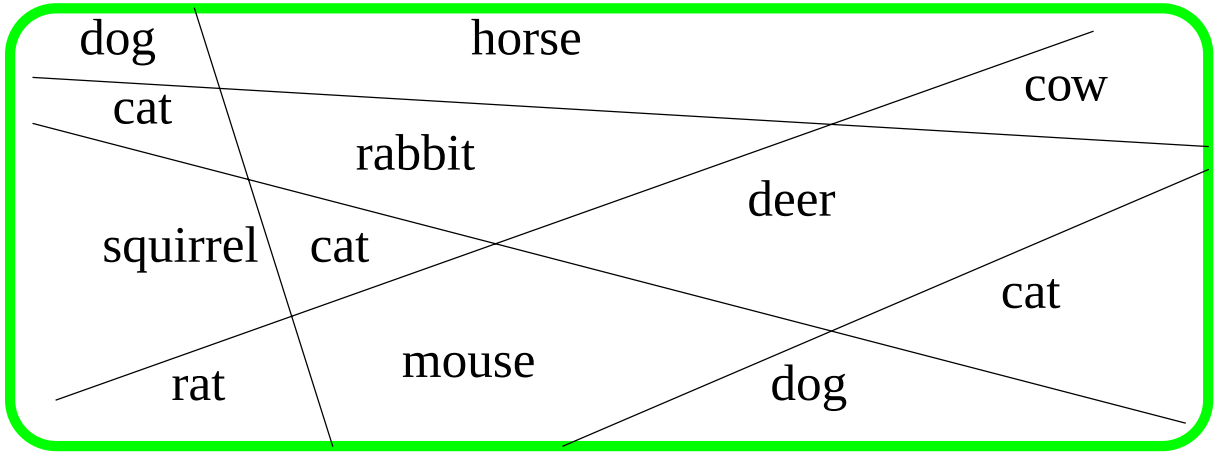
if a machine is expected to be infallible,
it cannot also be intelligent."

pp. 104-5 of the vol. on *Mechanical Intelligence* in Turing's *Collected Works* (North-Holland)

◊1 A disinclination to admit "I don't know".

"… if a machine is expected to be infallible,
it cannot also be intelligent."

e.g.: When "Deep Learning" identifies a situation with a pattern
matched poorly,
how is this classification's uncertainty reported to a user?
Not at all ?



4-legged Animals

How will these be classified:
camel, baby, mole, rhino, chair, table **?**

)

◊1 A disinclination to admit "I don't know".

"… if a machine is expected to be infallible,
it cannot also be intelligent."

e.g.: Any subprogram launched to seek something might not find it.
Perhaps it doesn't exist. Perhaps the search began in an unfortunate place.

Many complicated subprograms have failure modes.

Such a subprogram must communicate its failure to its calling program, which must be prepared to cope with a failure without incurring a disaster. Ideally that communication should have more diagnostic value than just a cryptic code or a fleeting display on a screen.

(An IEEE-754 (1985)-conforming floating-point computation may use *flag*s and *NaN*s to signal a failure and point to its first symptom, but only provided the programming language and system supports their diagnostic use.)

(See my web page's *Boulder.pdf* .)

◊2 A naive model of the Visual Cortex:

Layers of Neurons each connected to adjacent layers by relatively few Synapses.

Deep Learning's Model: A sequence of arrays
each acted on by *windowed* convolutions
that feed to the next array(s).

Scenario data —> 1st array —> convolution(s) —> 2nd array(s) —> ⋯
⋯ —> last array(s) —> Classifications

Learning "Training": Convolutions' coefficients are determined by minimizing a scalar residual that summarizes "errors" in classifications either chosen in advance or exposed with the aid of "guided feedback".

What's Missing? Prompt Motion Detection, and Logical Analysis

Every creature with an optic nerve detects motion of an image
long before it recognizes that image.
e.g.: Slow detection of motion by peripheral vision may have contributed to the collision of an *Uber* with a woman walking her bike across its path.

cf. **"Seeing Blind"** ***Scientific American*** 18 Oct. 2018 p. 14
A woman whose occipital lobe, including the visual cortex, was lost to a stroke can however "see" moving objects. Apparently visual stimuli travel somehow to a still intact "Middle Temporal Visual Area", and do so very quickly, by-passing the woman's lost visual cortex.

Some brain-injured soldiers in WWI displayed similar symptoms; the phenomenon has been known for a century.

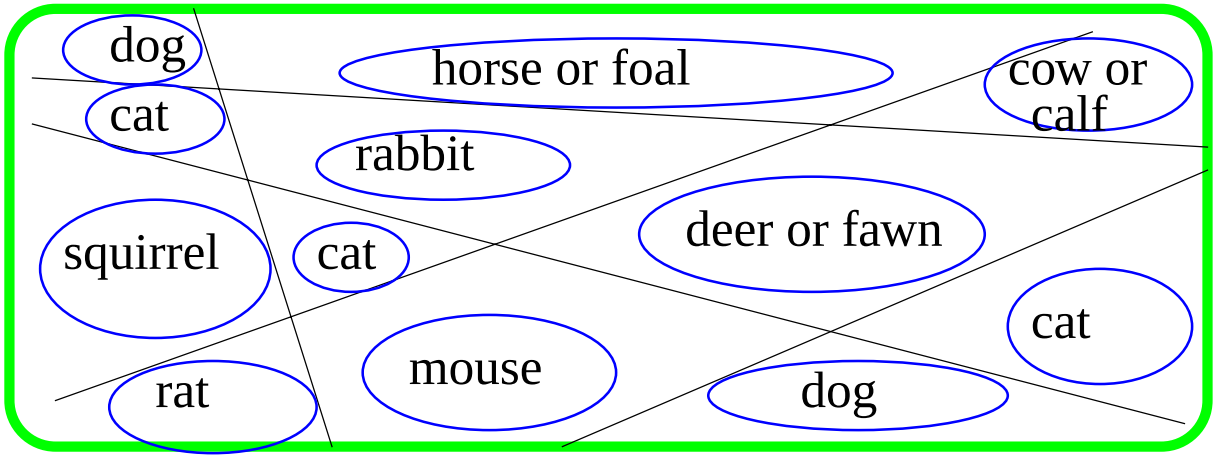
There is more to ***Vision*** than the Visual Cortex or its Imitations by Deep Learning or circuits inspired by it.

There is more to human ***Intelligence*** than ***Imitation***.
Humans can try to ***Understand*** another's mind well enough to ***Simulate*** it, albeit imperfectly.

◊3 Inadequate use of Logical Analysis with Deep Learning.

Medical students are taught "If you hear hoofs, think 'horses' before 'zebras'."

Hoofs on a small animal **imply** a foal or calf or fawn, not a St. Bernard dog.
Long whiskers **imply** a cat or rat or mouse or rabbit, not a dog nor horse nor …



4-legged Animals

How will these be classified:
camel, baby, mole, rhino, chair, table **?**

Outside an Oval should imply *"I don't know"*; else intelligence is in doubt.

cf. "Human-Level Intelligence or Animal-Like Abilities?" by Adnan Darwiche in pp. 58-67 of ***Comm. ACM* 61** #10 (Oct. 2018). He regards D.L. without logical analysis as an elaborate but mindless version of curve-fitting.

See also ...
http://www.ien.com/product-development/news/20864961/ai-shoots-down-fighter-planes-treats-bipolar-disorder

Reported in June 2017:

At the U. of Cincinnati, researchers combined a version of *Fuzzy Logic* with A.I. to improve its accuracy by at least an order of magnitude.

Applied to FMRI scans of brains of patients with Bipolar Disorder, A.I. alone predicted who would benefit from Lithium medication with a "25% error rate" that dropped below 1% when A.I. was combined with Logic.

Applied to simulations of Aerial Dogfights, A.I. alone ***never*** won, but ***always*** won when combined with logical analysis. Each combatant operated a flight-simulator, one operated by the A.I. + Logic software, the other by a U.S. Air Force Colonel with combat experience. He said

> "It seemed to be aware of my intentions and reacting instantly to my changes in flight and my missile deployment, ... It knew how to defeat the shot I was taking. It moved instantly between defensive and offensive actions as needed."

Another way? *Cf.* Judea Pearl (Mar. 2019) "The Seven Tools of Causal Inference, with Reflections on Machine Learning", pp. 54-60 of ***Comm. ACM*** **63** #3.

_____ End of "A.I. Strategy" _____

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability***.

What do these words mean?

Let	$y := f(x)$	be a function we wish we could compute.
Let	$\mathbf{y} := \mathrm{f}(\mathbf{x})$	be the program we write to approximate f .
	$\mathbf{Y} := \mathrm{F}(\mathbf{X})$	is the actual computation our computer performs.

They differ, despite that they share the same name "eff".

$\mathrm{f}(\mathbf{x})$ includes intentional approximations like discretization of the continuum, truncation of infinite processes, probabilistic sampling to reduce dimensions, …

$\mathrm{F}(\mathbf{X})$ includes rounding errors, binary <—> decimal conversion, time-outs. These are almost never explicit in the text of program $\mathrm{f}(\mathbf{x})$.

Ill-Condition afflicts *function* $f(x)$ if it is hypersensitive to perturbations of x .

Numerical Instability afflicts *program* $\mathrm{f}(\mathbf{x})$ when its execution $\mathrm{F}(\mathbf{X})$ suffers far more uncertainty from errors in F than is inherited from uncertainty in data $\mathbf{X}$.

(When I was a student in the 1950s, many people failed to distinguish the two concepts.)

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability***.

What do these words mean *quantitatively* ?

Condition Number $\|\partial f(\boldsymbol{x})/\partial \boldsymbol{x}\|$ gauges how sensitive $f(\boldsymbol{x})$ is to some small $\Delta\boldsymbol{x}$:

$\boldsymbol{y} + \Delta\boldsymbol{y} := f(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx f(\boldsymbol{x}) + \partial f(\boldsymbol{x})/\partial\boldsymbol{x}\cdot\Delta\boldsymbol{x}$ so $\Delta\boldsymbol{y} \approx \partial f(\boldsymbol{x})/\partial\boldsymbol{x}\cdot\Delta\boldsymbol{x}$.

Every $\Delta\boldsymbol{x}$ keeps $\|\Delta\boldsymbol{y}\| \le \|\partial f(\boldsymbol{x})/\partial\boldsymbol{x}\|\cdot\|\Delta\boldsymbol{x}\|$ with near equality for many a $\Delta\boldsymbol{x}$.

f is deemed "Ill-Conditioned" when its Condition Number is huge, which depends upon f, $\boldsymbol{x}$, and the ***Norms*** $\|\ldots\|$ that measure size or smallness.

Ill-Condition usually arises because given data $\mathbf{X}$ is too close to a singularity or a redundancy or a confluence of function f .

e.g.: $f(\boldsymbol{x})$ is the inverse of a nearly singular matrix $\boldsymbol{x}$.

A Numerically Unstable program f can be extremely difficult to debug.

Some tools I use are described in my web page's *Boulder.pdf*, but they are not available widely.

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability.***

These can degrade Deep Learning in two places:

1» When applying it to fresh data ***after*** D.L. has been "Trained" and "Tested".

2» While computing convolutions' coefficients **w** ***during*** "Training" of D.L.

"**Training**" is accomplished by seeking convolutions' coefficients **w** , called "Weights", to minimize a scalar "Error Measure" **£**(**w**) , actually a ***Residual***, that gauges how far the weights **w** are from identifying all desired patterns among a ***vast*** number of scenarios supplied as training data.

Training is an iterative process that seeks a minimum of **£**(**w**) by following the Downward gradient -g(**w**) := -∂**£**(**w**)/∂**w**´ .
(- Transposed *Jacobian*)

There can be thousands of weights **w** and many local minima **w** of **£**(**w**) .

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability.***

These can degrade Deep Learning in two places:

1» When applying it to fresh data ***after*** D.L. has been "Trained" and "Tested".

2» While computing convolutions' coefficients ***during*** "Training" of D.L.

"**Testing**" is accomplished by offering "Trained" D.L. a sufficient number of scenarios not used for training, and counting how often they are misclassified.

The "**Error-Rate**" is the percentage of misclassifications.

Error-Rates as high as 20% have often been deemed acceptable.

Error-Rates below 6% have almost always been accepted.

But if Testing data resembles Training data too closely, the low Error-Rate can fail to distinguish

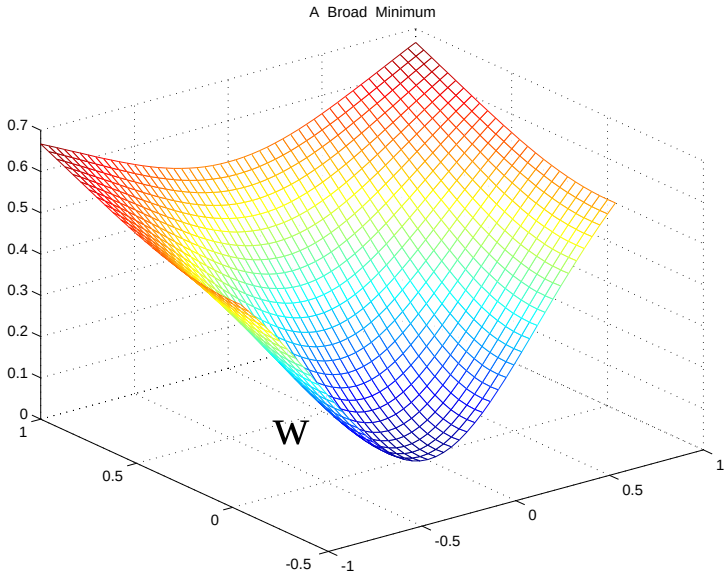
a **sharp minimum** of **£(w)** from a **broad minimum**.

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability.***

Ill-Condition can degrade Deep Learning …

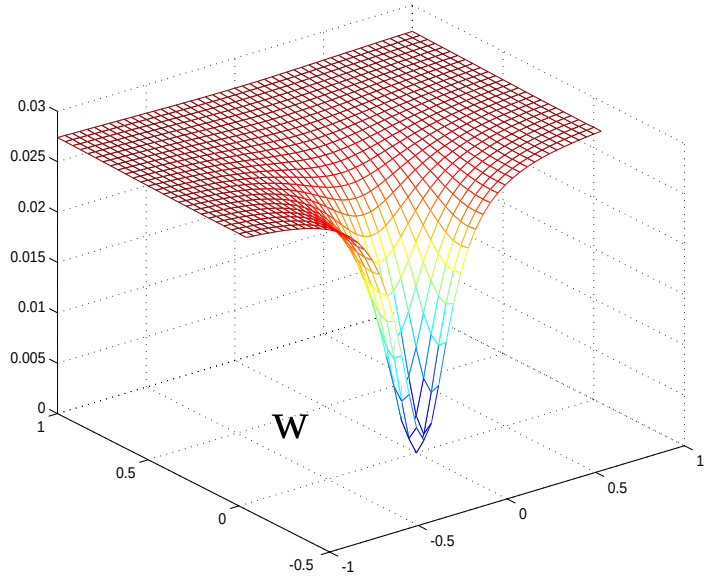
1» … when applied to fresh data ***after*** D.L. has been "Trained" and "Tested".

A **Broad Minimum** of **£(w)** …



Tolerates …

A **Sharp Minimum** of **£(w)** …



is Hypersensitive to …

… small variations in an applied Scenario's data, because of …

… a moderate Hessian $\partial^2 \mathbf{£}(\mathbf{w})/\partial \mathbf{w}^2$ … a huge Hessian $\partial^2 \mathbf{£}(\mathbf{w})/\partial \mathbf{w}^2$.

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability.***

These can degrade Deep Learning …

2» … while computing convolutions' coefficients ***during*** "Training" of D.L.

because of an Ill-Conditioned Hessian array $H(\mathbf{w}) := \partial^2 £(\mathbf{w})/\partial \mathbf{w}^2$

that costs too much to compute, so we rarely know its …

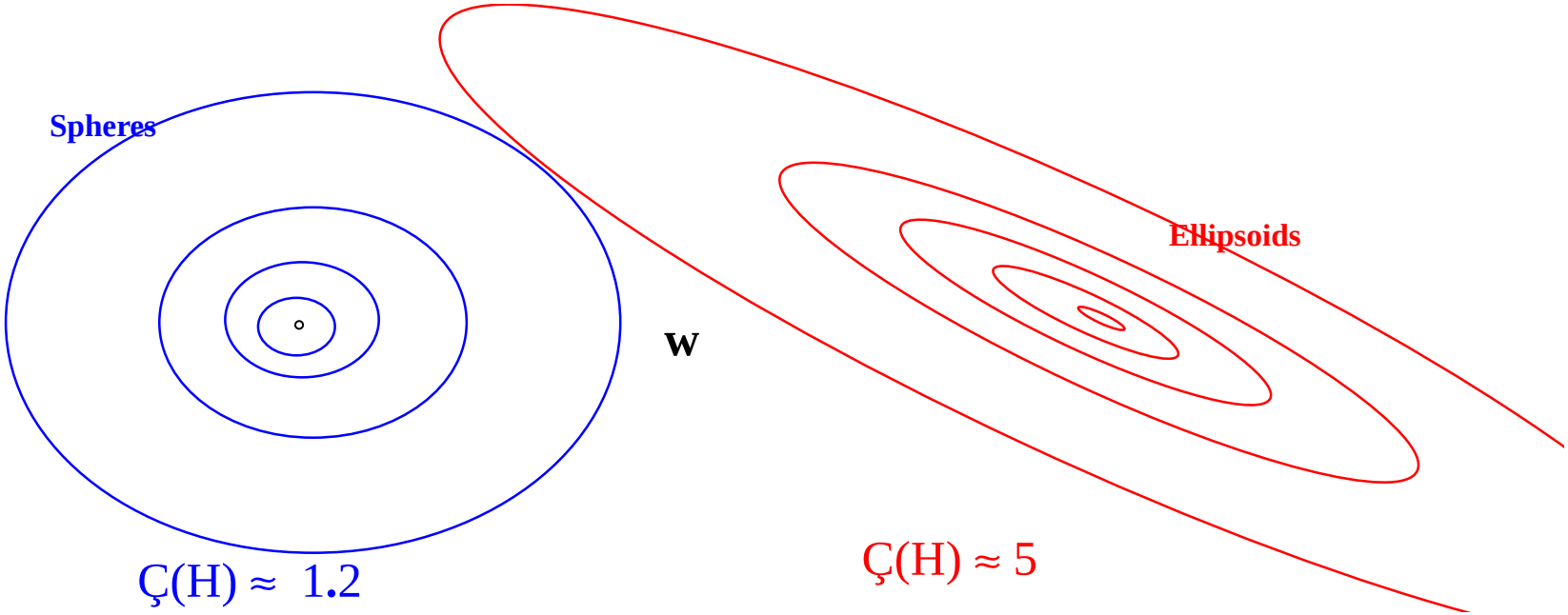
… ***Condition Number*** $Ç(H) := \|H\| \cdot \|H^{-1}\| \geq 1$.

H is deemed "Ill-Conditioned" when its Condition Number Ç(H) is *HUGE*.
It can happen if categories to be distinguished by D.L. are too nearly redundant.

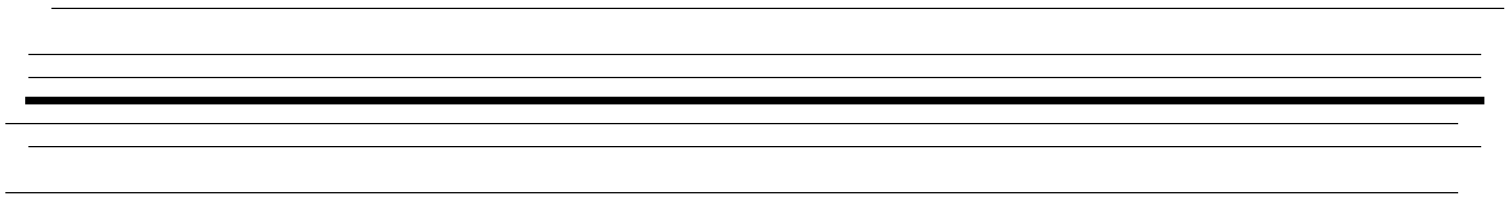
How can you tell whether H is Ill-Conditioned without computing it?
By Ill-Condition's deleterious consequences.

If H is well-conditioned, nothing bad happens during Training.

Level Lines/Surfaces/… of Residual £(**w**)



Plots for Ç(H) >> 10 appear as a family of variably spaced parallel lines**:**



Where is the minimizing **w** ?

◊4 Disregard of ***Ill-Condition*** and ***Numerical Instability.***

These degrade Deep Learning ***during*** "Training" in two ways**:**

1: An Ill-Conditioned Hessian H(**w**) spawns ***Numerical Instability***
causing minimizing weights **w** to be computed inaccurately,
often much too big.

Weights **w** that are much too big cause losses of input-data digits to cancellation during applications of D.L., degrading the reliability of its classifications, especially when the minimum of **£**(**w**) is sharp because ||H|| is very big too.

The easiest way to avoid that Numerical Instability is to perform ***all*** arithmetic during the optimization process (searching for a minimizing **w**) carrying more than twice as much precision as you trust in the data or desire in the weights **w**.

2: An Ill-Conditioned Hessian H(**w**) ***Prolongs the Search*** for a minimizing **w**, sometimes by an order of magnitude.
This is most painful during re-Training using newly found data.

◊5 Deep Learning takes too long to Learn, *i.e.*, to be (re)Trained.

Training is an iterative process that seeks a minimum of $\mathbf{£}(\mathbf{w})$ by following the Downward gradient $-g(\mathbf{w}) := -\partial\mathbf{£}(\mathbf{w})/\partial\mathbf{w}'$.

Three such processes will be examined hereunder. All of them converge almost always to a local minimum of $\mathbf{£}(\mathbf{w})$ at rates ultimately limited by the Hessian's Condition Number $Ç := \|H\|\cdot\|H^{-1}\|$ where $H(\mathbf{w}) := \partial^2\mathbf{£}(\mathbf{w})/\partial\mathbf{w}^2$.
Ç is almost never known in advance; sometimes it may be estimated afterwards.

The three processes to be examined are …

GD: Gradient Descent

GD+M: Gradient Descent with Momentum

AGD: Anadromic Gradient Descent

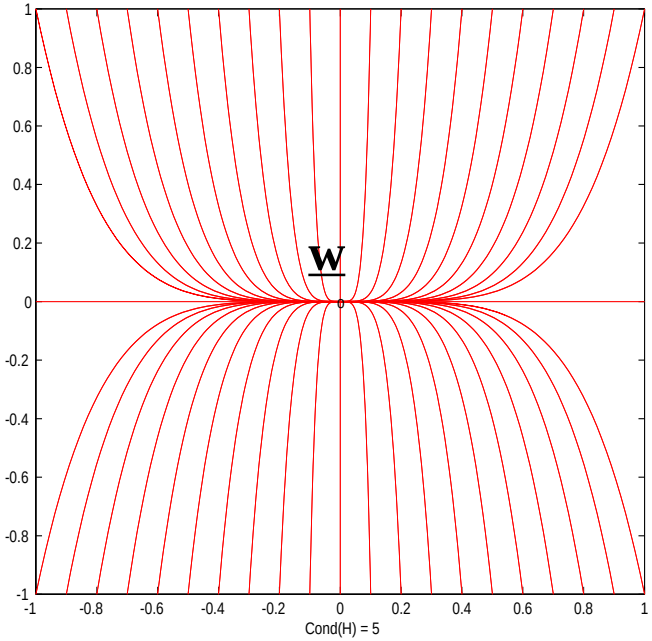
(What about *Stochastic Gradient Descent* ? Ask me later.)

Gradient Descent

This process is motivated by a Differential Equation $d\mathbf{w}(\tau)/d\tau = -g(\mathbf{w}(\tau))$ along whose trajectories $d\mathfrak{L}(\mathbf{w}(\tau))/d\tau = -\|g(\mathbf{w}(\tau))\|^2 \le 0$, so $\mathfrak{L}(\mathbf{w}(\tau))$ decreases along every trajectory until it stops at a ***Stationary Point*** $\underline{\mathbf{w}}$ where gradient $g(\underline{\mathbf{w}}) = \mathbf{o}$.

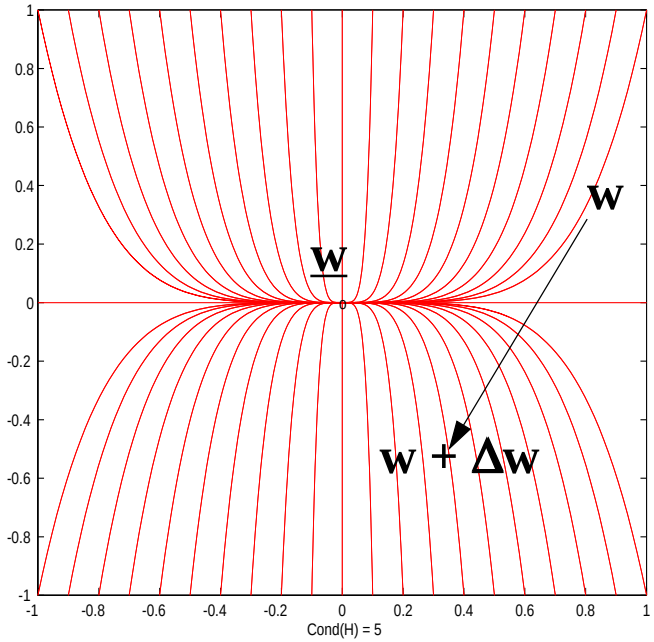
Almost always, that stationary point $\underline{\mathbf{w}}$ is a local minimum of residual $\mathfrak{L}(\mathbf{w})$.

Trajectories of $d\mathbf{w}(\tau)/d\tau = -g(\mathbf{w}(\tau))$



GD is Euler's discretization $\Delta\mathbf{w}/\Delta\tau = -g(\mathbf{w})$ of the Differential Equation.

Each step ("Epoch") of **GD** over-writes $\mathbf{w}$ by a new $\mathbf{w} := \mathbf{w} + \Delta\mathbf{w}$ determined from Euler's discretization $\Delta\mathbf{w} := -g(\mathbf{w})\cdot\Delta\tau$ for some step-size $\Delta\tau$ ("learning rate", a "hyper-parameter") which must not be chosen too big. Each step from $\mathbf{w}$ to $\mathbf{w} + \Delta\mathbf{w}$ amounts to drawing a tangent to the trajectory through $\mathbf{w}$ thus:



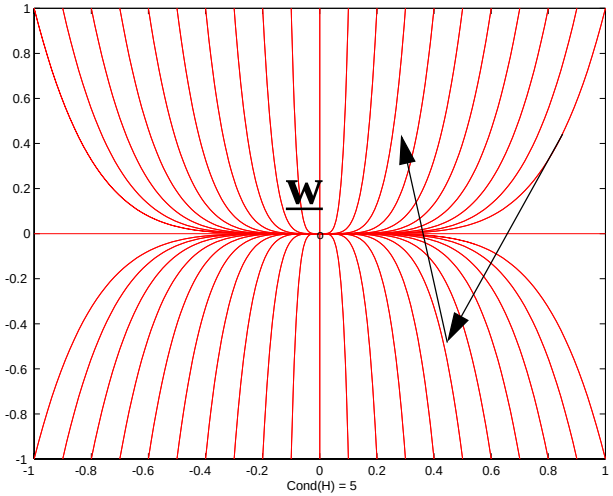
Provided $\Delta\tau < 2/\|H\|$, successive steps converge to $\underline{\mathbf{w}}$, very slowly if $\Delta\tau$ is too small, or else slowly by ricocheting. Ricocheting retards convergence severely if the unknown Hessian $H(\mathbf{w}) := \partial^2 £(\mathbf{w})/\partial\mathbf{w}^2$ is Ill-Conditioned.

Why ?

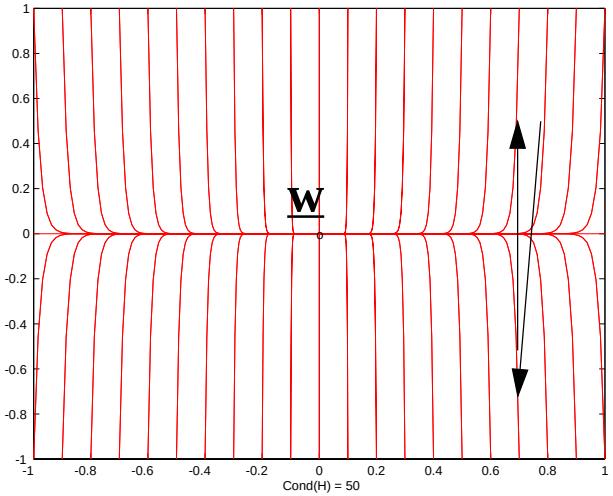
Why is Gradient Descent's convergence retarded severely by Ricocheting if the unknown condition number Ç of the Hessian H($\underline{\mathbf{w}}$) is too big ?

Because almost all trajectories bend sharply, more so if Ç is big.

Ç = 5



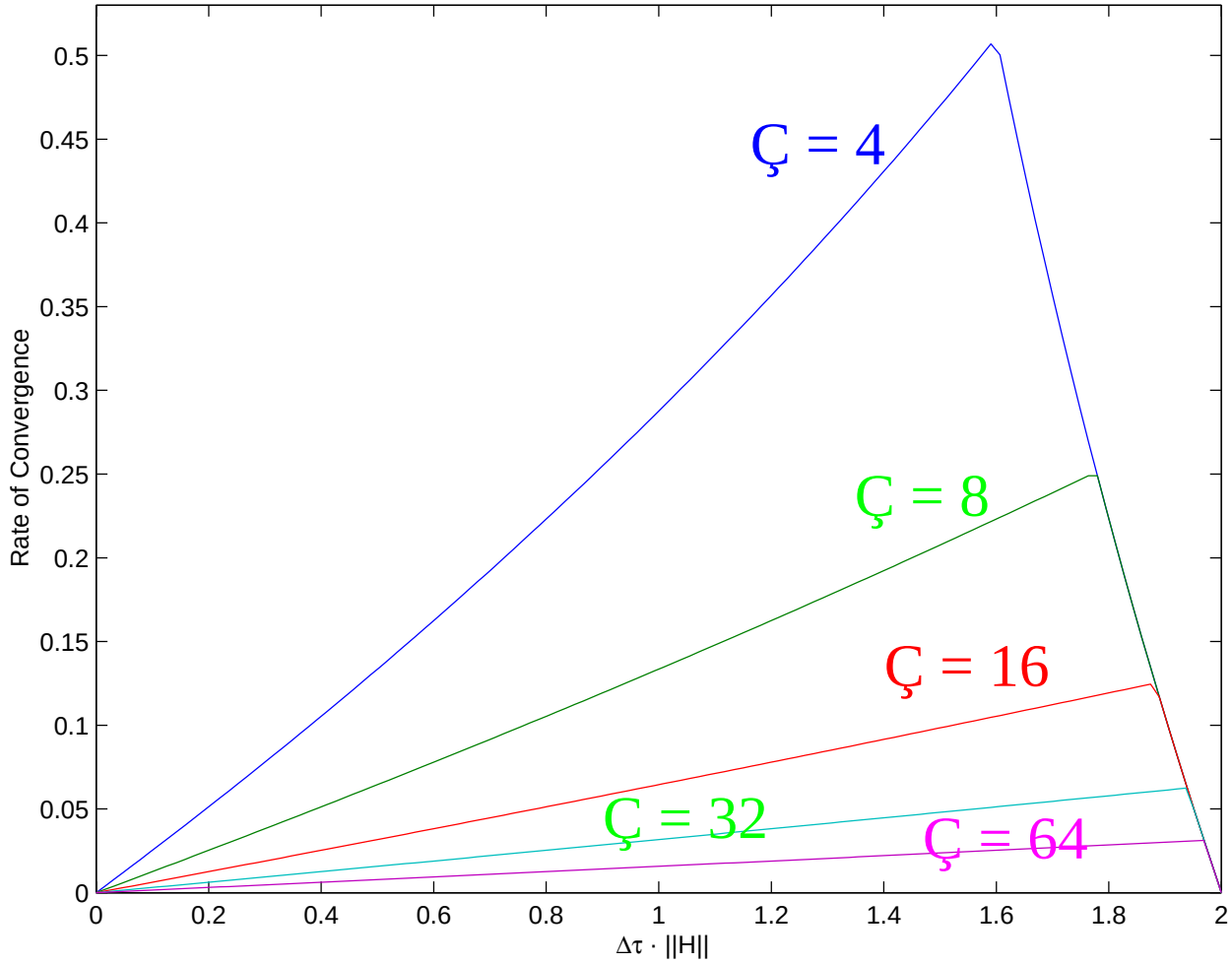
Ç = 50



GD diverges if $\Delta\tau \geq 2/\|H\|$, yet converges fastest for $\Delta\tau = (2/\|H\|)/(1 + 1/Ç)$; it's barely smaller when Ç is big. That makes the best choice for $\Delta\tau$ delicate.

The convergence ratio $\rho := \|\mathbf{w} + \Delta\mathbf{w} - \underline{\mathbf{w}}\|/\|\mathbf{w} - \underline{\mathbf{w}}\|$ for **GD**'s best $\Delta\tau$ is at most $\rho = 1 - 2/(1 + Ç)$.

Gradient Descent's Rate of Convergence -log(ρ) vs. Δτ·||H||



Estimate ||H|| ; set Δτ :≈ 1.25/||H|| ; after a few iterations, estimate Ç .
But **GD** converges too slowly in the long run.

Gradient Descent with Momentum

$$\text{new } \mathbf{w} = \mathbf{w} + \Delta\mathbf{w} := \mathbf{w} - g(\mathbf{w})\cdot\Delta\tau + \beta\cdot(\mathbf{w} - \text{old } \mathbf{w}?)$$

The Momentum := **w** - old **w** . The process can be associated with a differential equation, but was more likely inspired by Conjugate Gradient Iterations which determine varying values for β and stepsize $\Delta\tau$ at each step from matrix-vector products like $H(\mathbf{w})\cdot\mathbf{x}$, but ***only*** when the Hessian H is accessible that way.

Otherwise the best constant values for Hyper-parameters β and stepsize $\Delta\tau$ depend upon the unknown Hessian $H(\underline{\mathbf{w}})$ and its Condition Number Ç, so they must be found by experiment during iterations, subject to $\|H\|\cdot\Delta\tau/2 - 1 \le \beta < 1$.

If $\|H(\underline{\mathbf{w}})\|$ and its Ç were known, **GD+M**'s best Hyper-parameters would be

$$\beta := (\sqrt{Ç} - 1)^2/(\sqrt{Ç} + 1)^2 \quad \text{and} \quad \Delta\tau := 4/(\|H\|\cdot(1 + 1/\sqrt{Ç})^2)$$

and their convergence ratio $\rho := \|\mathbf{w} + \Delta\mathbf{w} - \underline{\mathbf{w}}\|/\|\mathbf{w} - \underline{\mathbf{w}}\|$ would be at most

$$\rho = 1 - 2/(1 + \sqrt{Ç}) .$$

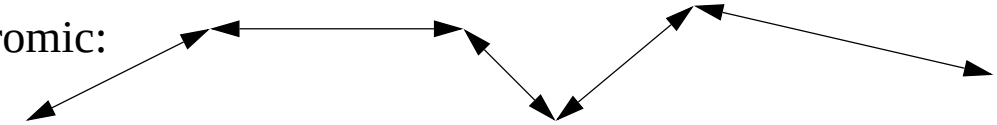
Compare this with Gradient Descent's $\rho = 1 - 2/(1 + Ç)$, which is far slower, especially if Ç is HUGE.

Anadromic Gradient Descent

"Anadromic" is a nonce-word obtained from the Greek for "Running back" like anadromous fish (salmon) that return from the sea to their native streams.

An Anadromic numerical method for approximating the solution of a differential equation like $d\mathbf{w}(\tau)/d\tau = -g(\mathbf{w}(\tau))$ shares its semi-group property of returning from a terminal point to its starting point if the sign of $d\tau$ is reversed. Similarly an anadromic numerical method retraces its steps from its terminus to its origin exactly (but for roundoff) if the steps $\Delta\tau$ (perhaps varying) are reversed in sign and order. Most numerical methods (Euler's, Runge-Kutta, …) aren't anadromic.

Anadromic:



So What?

If you find an anadromic numerical method for your differential equation (how to find one is usually unobvious) the accuracy you achieve with it will usually cost less computation than if achieved with a non-anadromic numerical method of similar complexity. *cf*. My web page's mcom2498.pdf and citations there, & ***Geometric Numerical Integration*** 2d ed. by Hairer, Lubitsch & Wanner (2006).

(What is called "Anadromic" here is called "Symmetric" there; but "symmetric" is a word too overworked.)

Anadromic Gradient Descent

Numerical methods used to solve the *Hamiltonian* differential equations for friction-free mechanical systems are often anadromic incidentally. Starting from a Hamiltonian function $Æ(\mathbf{w}, \mathbf{v})$, the Hamiltonian differential equations are

$$d\mathbf{w}/d\tau = \partial Æ/\partial \mathbf{v}' \quad \text{and} \quad d\mathbf{v}/d\tau = -\partial Æ/\partial \mathbf{w}' .$$

Solutions $\{\mathbf{w}(\tau), \mathbf{v}(\tau)\}$ define trajectories along which $dÆ(\mathbf{w}(\tau), \mathbf{v}(\tau))/d\tau = 0$, whence the Hamiltonian $Æ$ (among other things) is conserved, so trajectories run forever (friction-free) except for a few that collide. This isn't what we want.

We want ***all*** trajectories to terminate at stationary points, minima among them.

To that end we shall introduce friction as a Drag $\boldsymbol{\mu} > 0$. Here is how:

Let $Æ(\mathbf{w}, \mathbf{v}) := \mathbf{£}(\mathbf{w}) + \|\mathbf{v}\|^2/2 = \mathbf{£}(\mathbf{w}) + \mathbf{v}'\cdot\mathbf{v}/2$. The differential equations are

$$d\mathbf{w}/d\tau = \partial Æ/\partial \mathbf{v}' = \mathbf{v} \quad \text{and, starting from } \mathbf{v} := \mathbf{o} \text{ initially,}$$
$$d\mathbf{v}/d\tau = -\partial Æ/\partial \mathbf{w}' - \boldsymbol{\mu}\cdot\mathbf{v} = -\partial \mathbf{£}(\mathbf{w})/\partial \mathbf{w}' - \boldsymbol{\mu}\cdot\mathbf{v} = -g(\mathbf{w}) - \boldsymbol{\mu}\cdot\mathbf{v} .$$

Along a trajectory, $dÆ(\mathbf{w}(\tau), \mathbf{v}(\tau))/d\tau = -\boldsymbol{\mu}\cdot\|\mathbf{v}(\tau)\|^2 \le 0$, so $Æ$ decreases until $\mathbf{v} = \mathbf{o}$. If the trajectory ends at a finite stationary point $\underline{\mathbf{w}}$ then $g(\underline{\mathbf{w}}) = \mathbf{o}$ too.

Anadromic Gradient Descent's ODE

$d\mathbf{w}/d\tau = \mathbf{v}$ and, starting from $\mathbf{v}(0) := \mathbf{o}$ initially,
$d\mathbf{v}/d\tau = -g(\mathbf{w}) - \mu\cdot\mathbf{v}$.

$Æ(\mathbf{w}, \mathbf{v}) := £(\mathbf{w}) + \|\mathbf{v}\|^2/2$ is a pseudo-Hamiltonian;
then $dÆ(\mathbf{w}(\tau), \mathbf{v}(\tau))/d\tau = -\mu\cdot\|\mathbf{v}\|^2 < 0$ until $\mathbf{v} = g(\underline{\mathbf{w}}) = \mathbf{o}$.

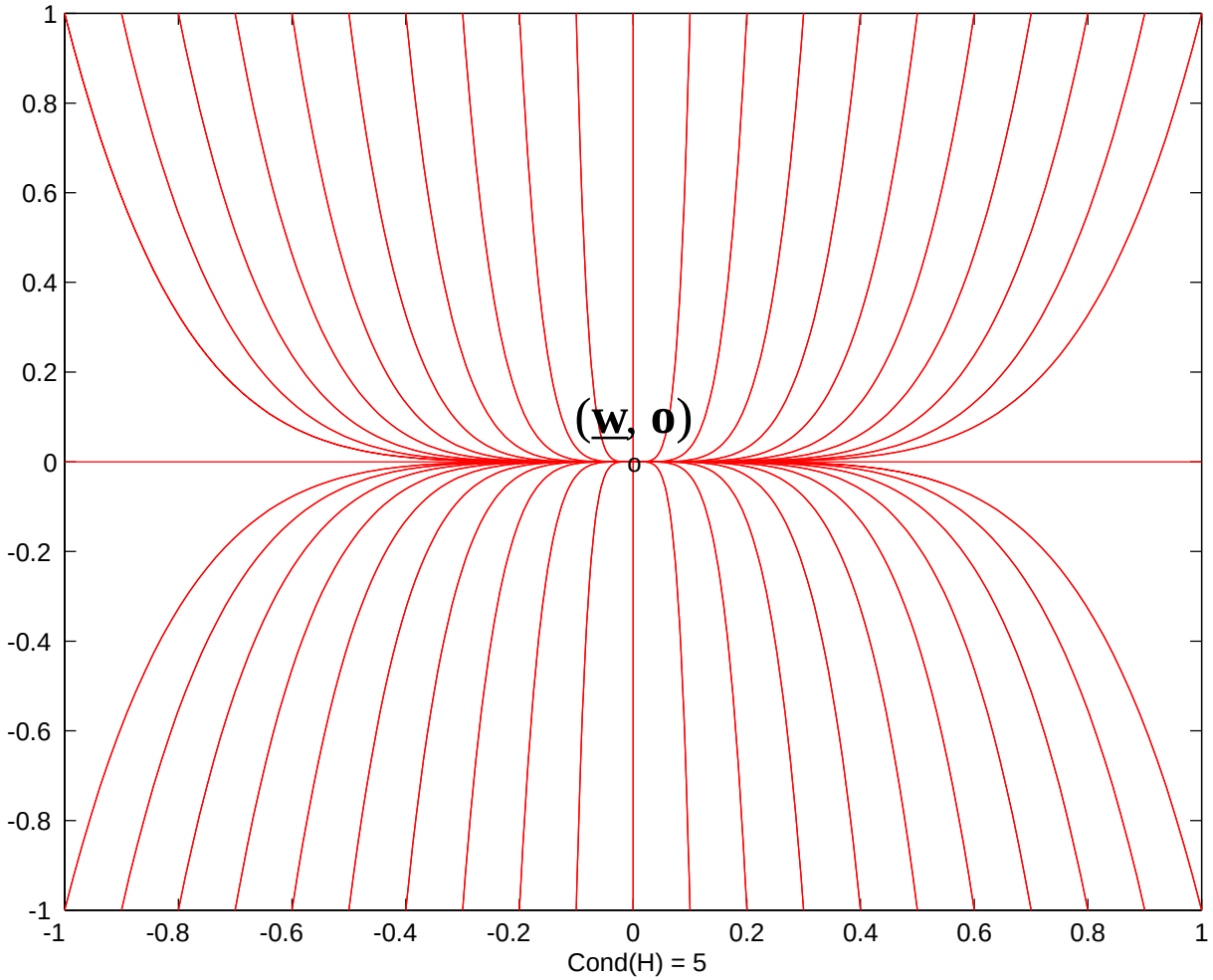
The stationary point $(\underline{\mathbf{w}}, \mathbf{o})$ at a trajectory's end is
almost always a local minimum of residual $£(\underline{\mathbf{w}})$.

$\mathbf{v}(\tau)$ cannot ever get very big since $\mu\cdot\int_0^\infty \|\mathbf{v}(\tau)\|^2 d\tau = Æ(\mathbf{w}(0), \mathbf{o}) - Æ(\underline{\mathbf{w}}, \mathbf{o})$.

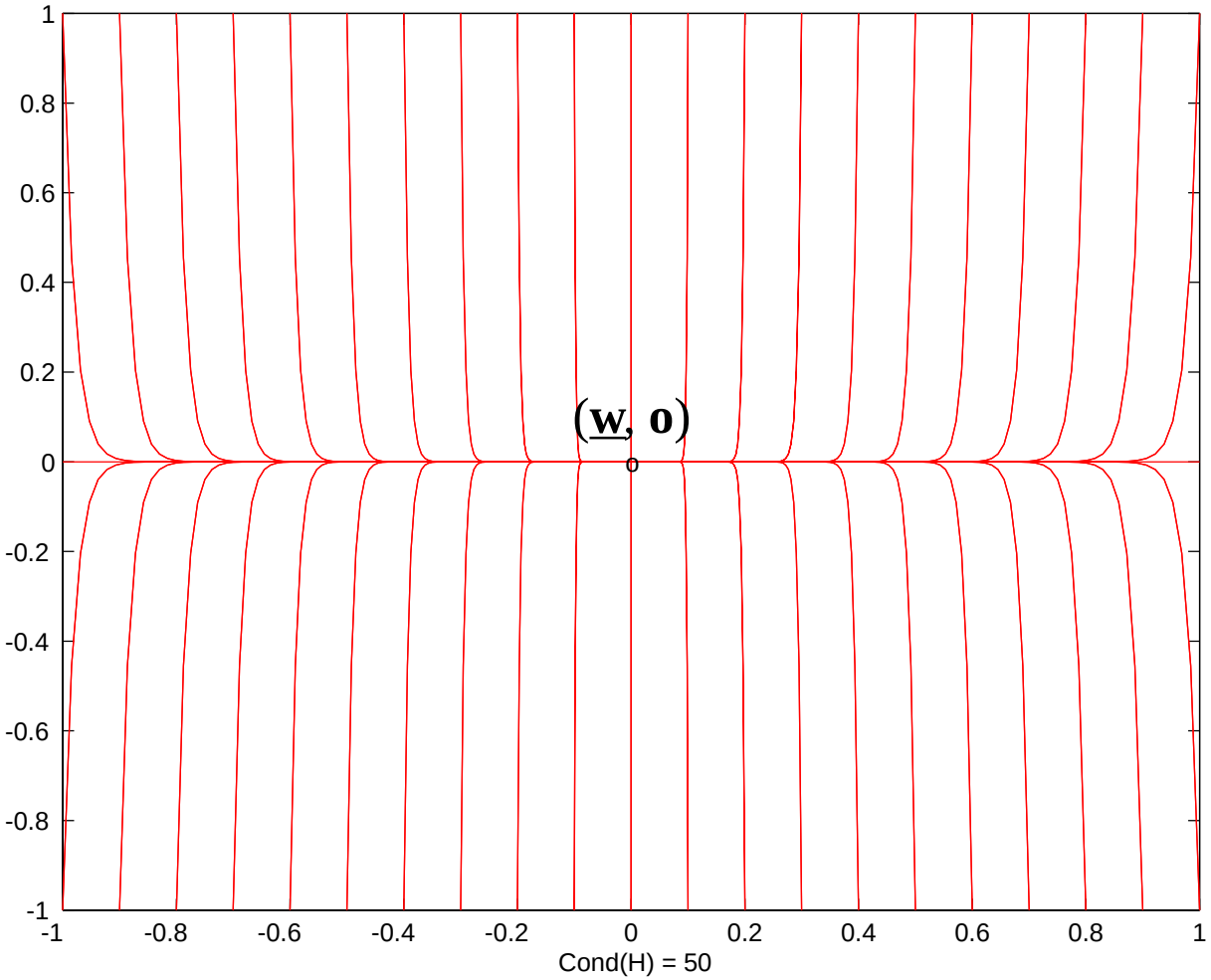
What do the trajectories look like? That depends upon the drag μ and the unknown Condition number Ç of the Hessian. Spirals occur unless $\mu \geq 2\sqrt{\|H\|}$.

The bigger is Ç the sharper almost all trajectories bend or the tighter are spirals.

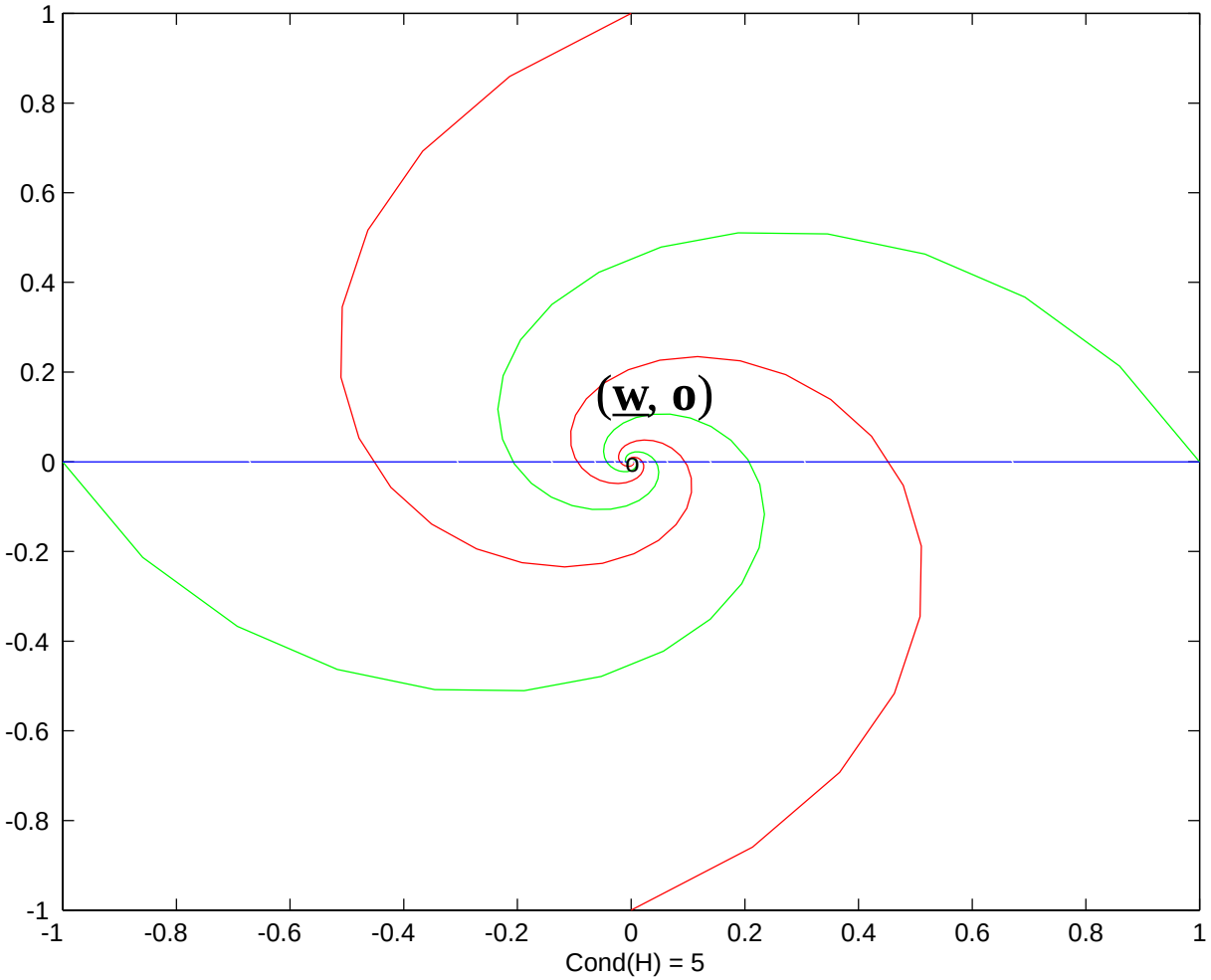
AGD ODE, Big Drag, Ç = 5 , Gradual bends



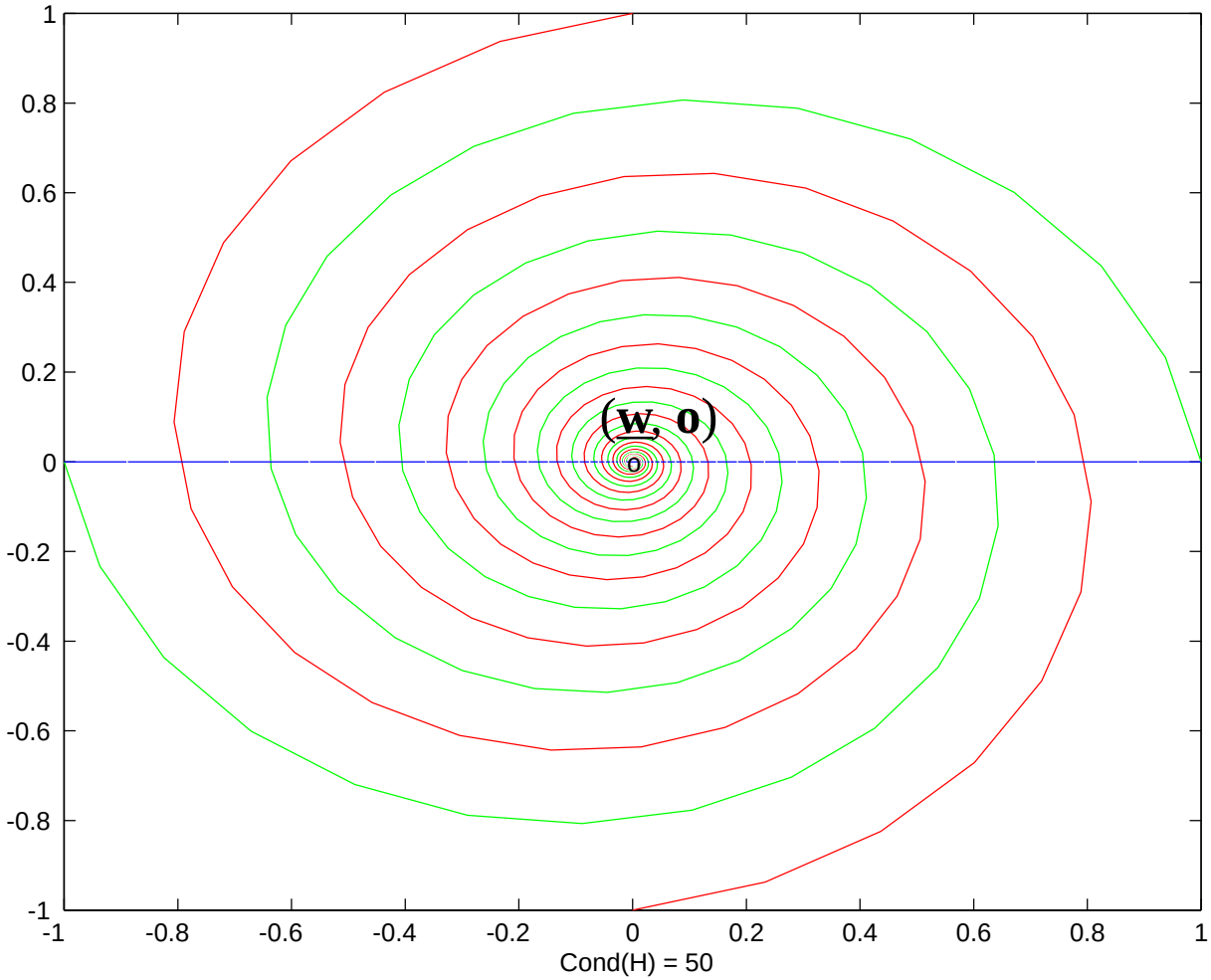
AGD ODE, Big Drag, Ç = 50, Sharp bends



AGD ODE, Small Drag, Ç = 5, Loose spirals



AGD ODE, Small Drag, Ç = 50, Tight spirals



Anadromic Gradient Descent's Algorithm

Here is an ***Anadromic*** discretization of the differential equations:

Temporary $\mathbf{W} := \mathbf{w} + \mathbf{v}\cdot\Delta\tau/2$; *(cf. "Leapfrog method")*

new $\mathbf{v} := \mathbf{v} - \big(g(\mathbf{W}) + \mu\cdot\mathbf{v} \big)\cdot\Delta\tau/(1 + \mu\cdot\Delta\tau/2)$; (sole extra cost)

new $\mathbf{w} := \mathbf{W} + (\text{new } \mathbf{v})\cdot\Delta\tau/2$;

(Save work by eliding each iteration's last $\mathbf{W}$ with the next iteration's first.)

The improved accuracy, or reduced ricocheting, comes from the term $\Delta\tau^2$ in …

$$\big(Æ(\text{new } \mathbf{w}, \text{new } \mathbf{v}) - Æ(\mathbf{w}, \mathbf{v}) \big)/\Delta\tau \;=\; -\mu\cdot\|(\mathbf{v} + \text{new } \mathbf{v})/2\|^2 \;+\; \mathbf{O}(\Delta\tau^2) \;.$$

If $H(\underline{\mathbf{w}})$ and its Ç were known, **AGD**'s best Hyper-parameters would be

$$\mu := 2\sqrt{\|H\|}/\sqrt{(1 + Ç)} \quad \text{and} \quad \Delta\tau := (2/\sqrt{\|H\|})/\sqrt{(1 + 1/Ç)} \;;$$

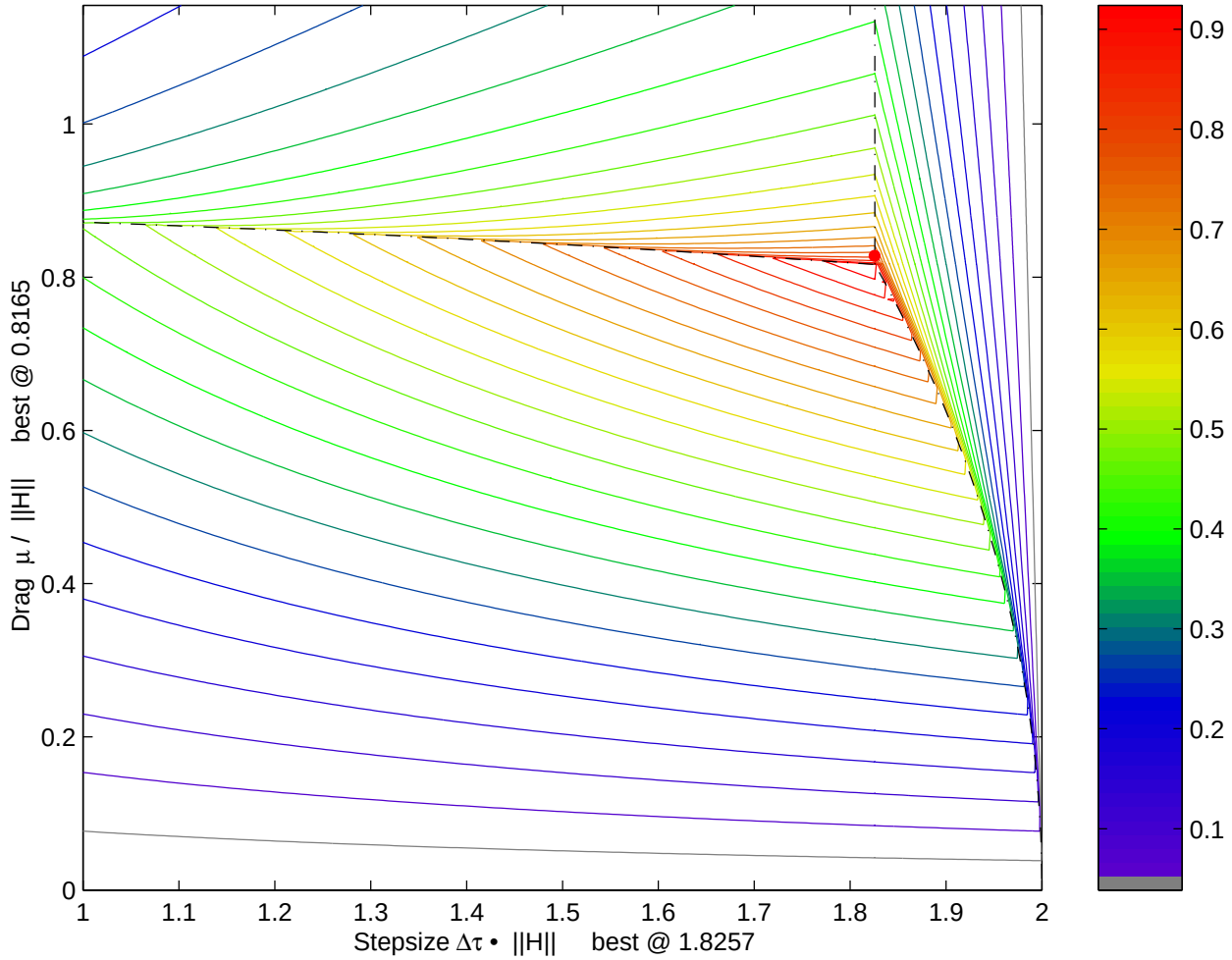
and their convergence ratio $\rho := \|\text{new } \mathbf{w} - \underline{\mathbf{w}}\|/\|\mathbf{w} - \underline{\mathbf{w}}\|$ would be at most

$$\rho = 1 - 2/(1 + \sqrt{Ç}) \;.$$

This ρ is the same as for **GD+M** ; but **AGD**'s Hyper-parameters are easier to choose, as illustrated by the following plots of $-\log(\rho)$ *vs*. hyper-parameters.

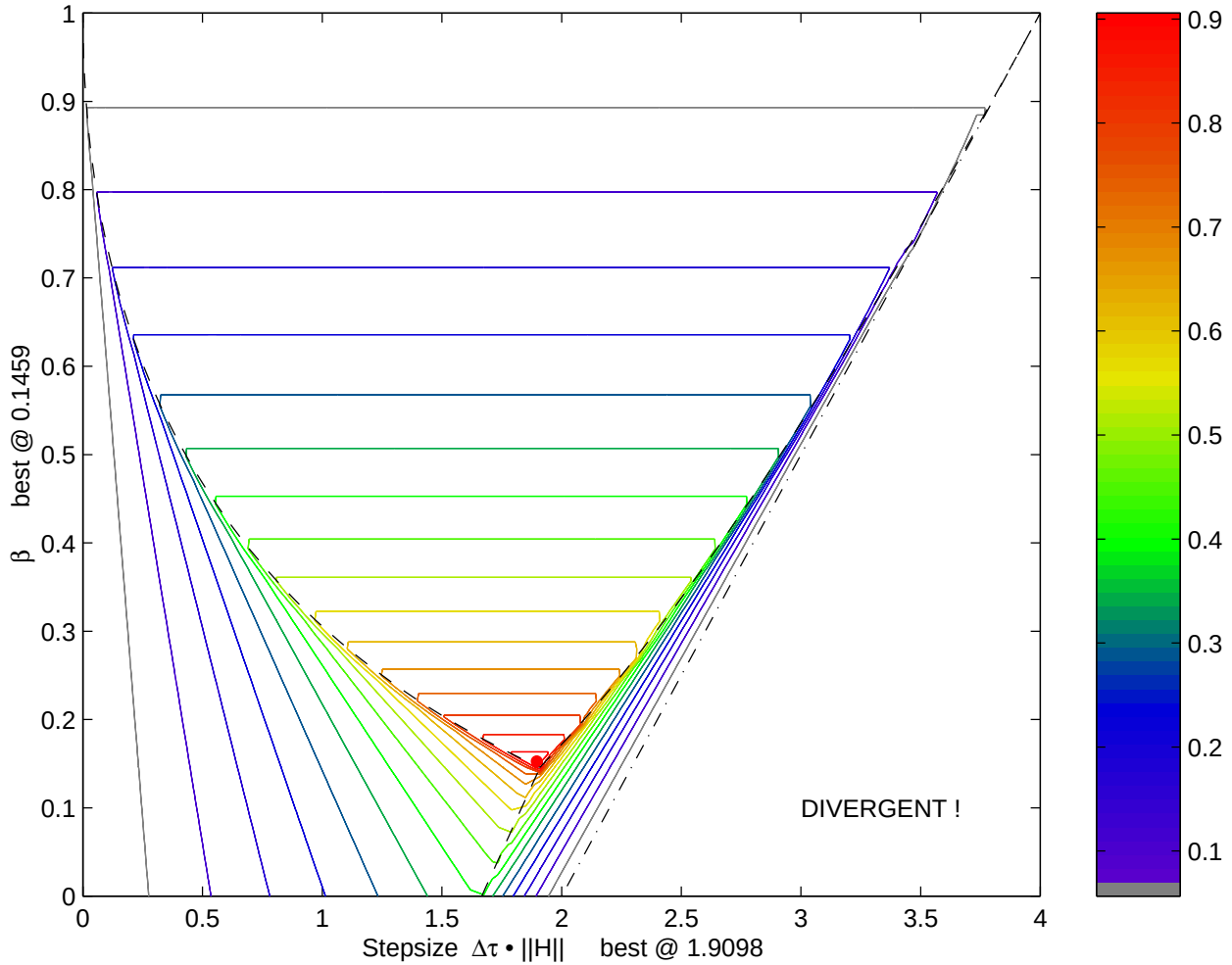
(The bigger is Rate $-\log(\rho)$, the faster is convergence, ultimately,)

Anadromic Gradient Descent Condition # of H : Ç = 5



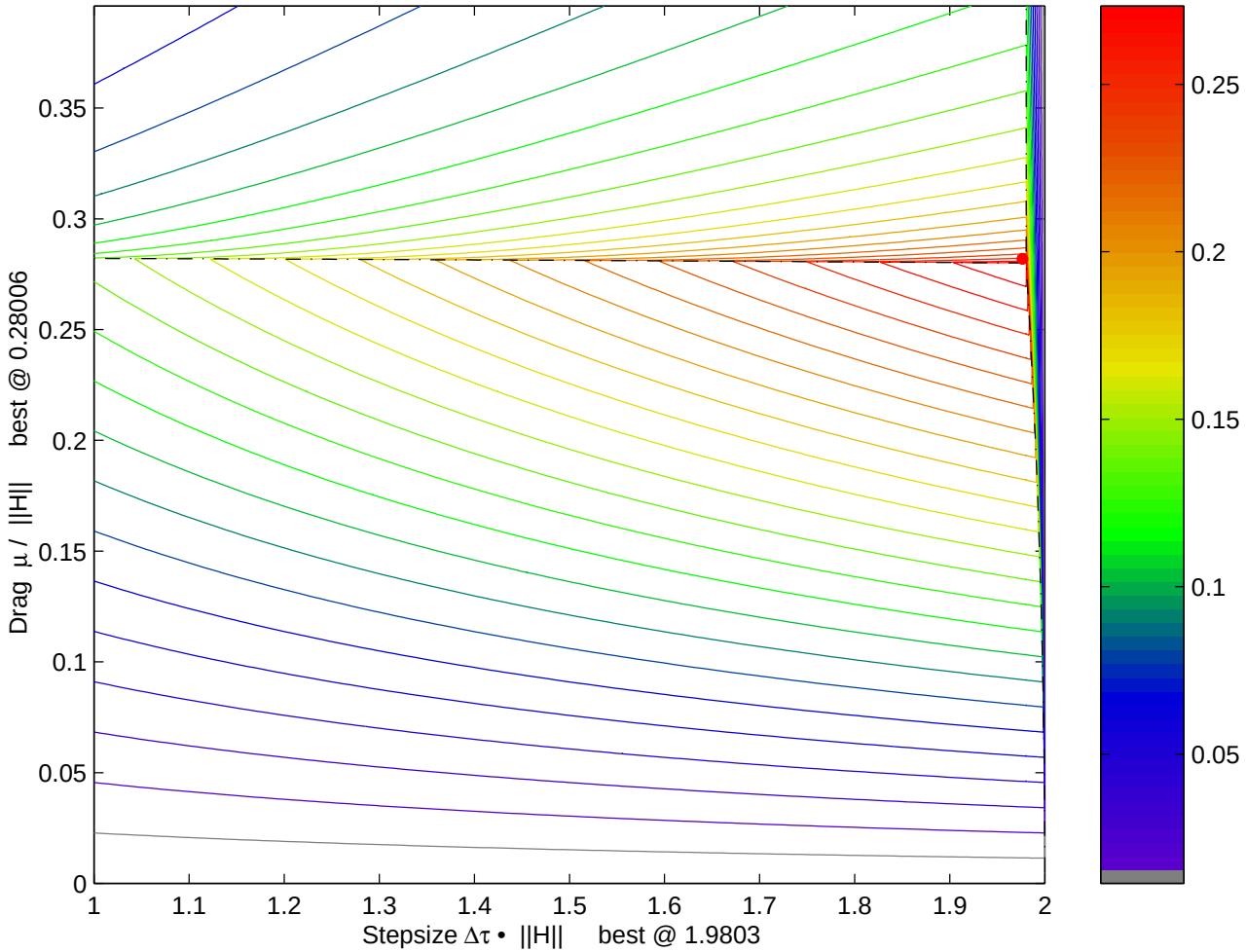
The Rate declines moderately for slightly sub-optimal Δτ and Drag μ .

Gradient Descent + Momentum Condition # of H : Ç = 5



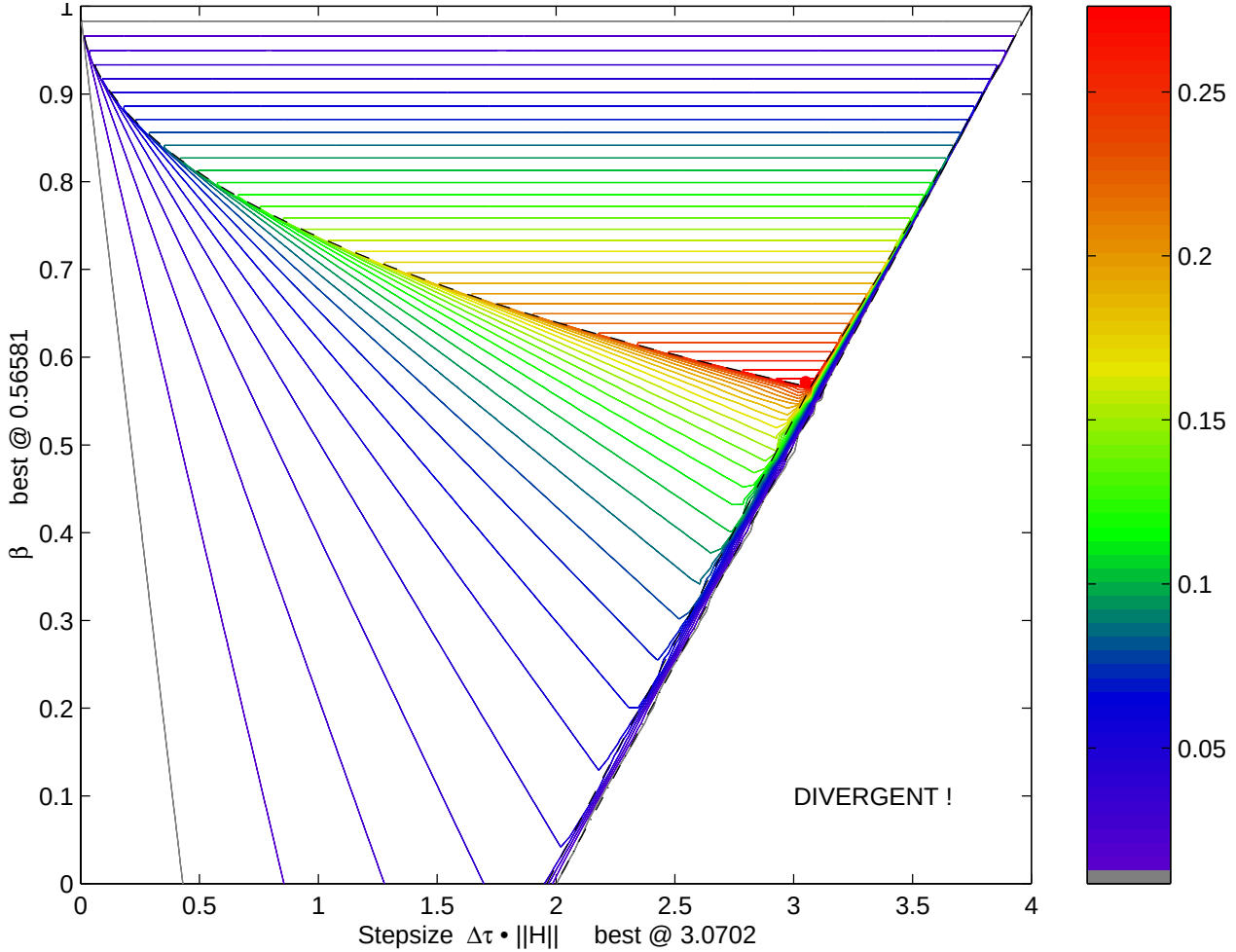
The Rate drops moderately only if point $(\Delta\tau, \beta)$ lies near that hook-shaped arc.

Anadromic Gradient Descent Condition # of H : Ç = 50



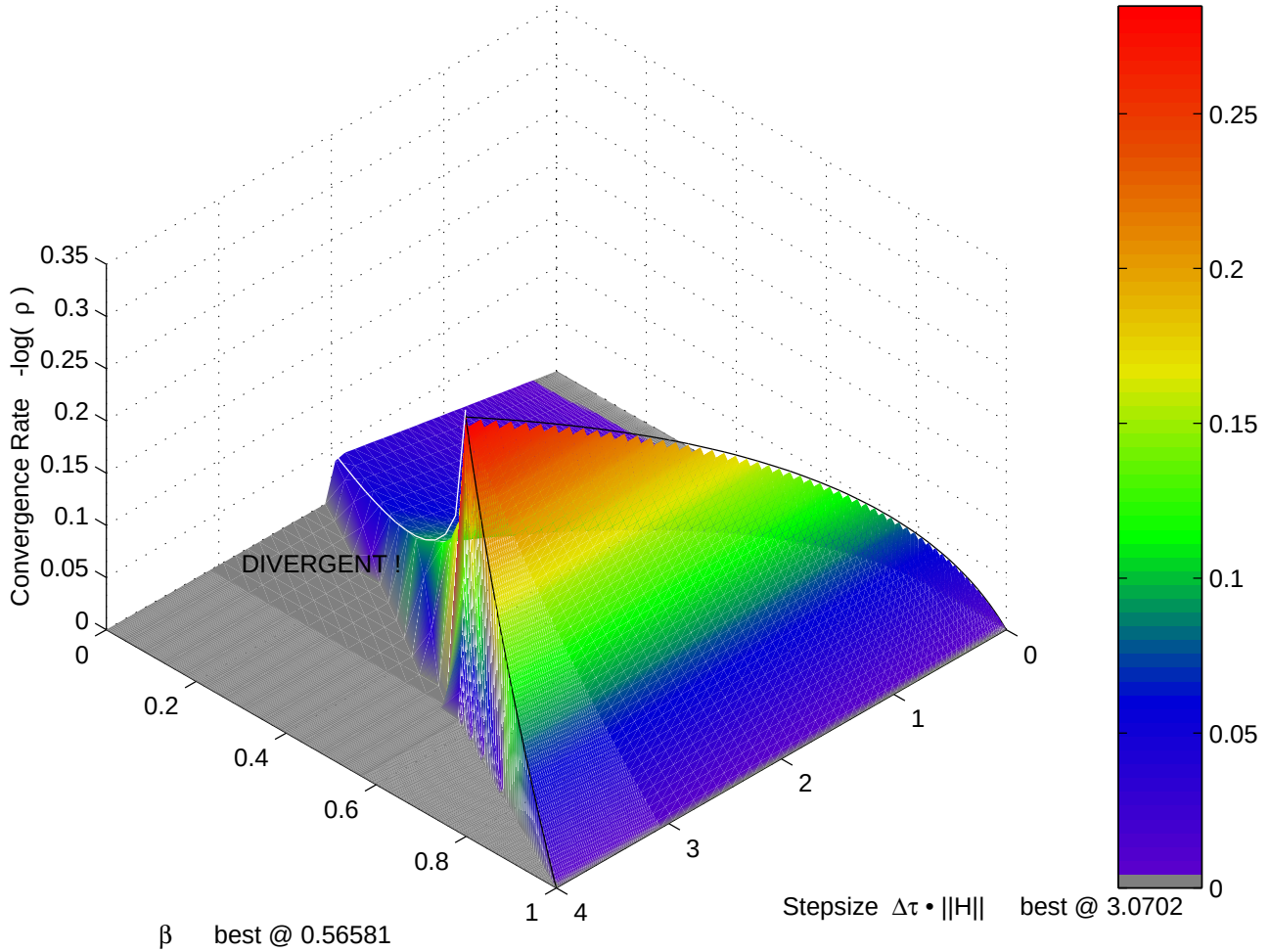
The Rate declines moderately for slightly sub-optimal Δτ and Drag μ .

Gradient Descent + Momentum Condition # of H : Ç = 50



The Rate drops abruptly even if point $(\Delta\tau, \beta)$ lies very near that hook-shaped arc. Divergence occurs if $(\Delta\tau, \beta)$ lies only slightly beyond the optimal • .

Gradient Descent + Momentum Condition # of H : Ç = 50



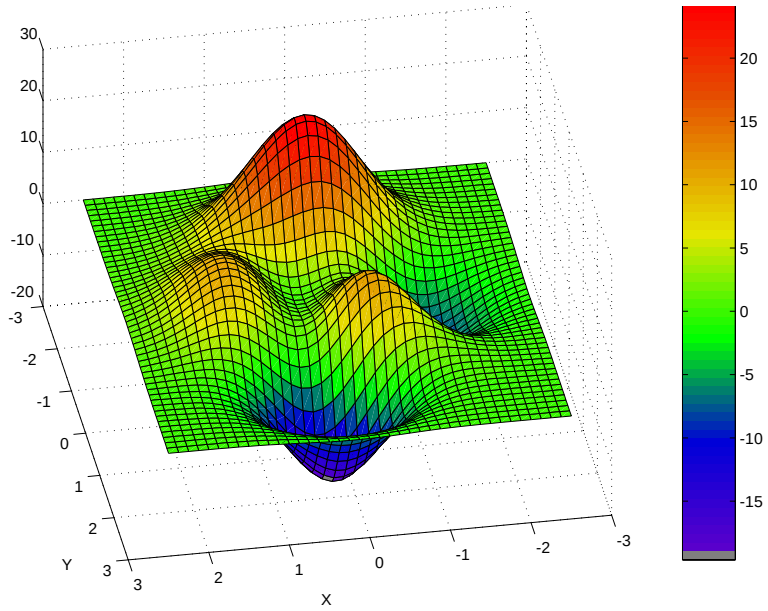
See how abruptly Rate can drop when $(\Delta\tau, \beta)$ is almost optimal.

Example:

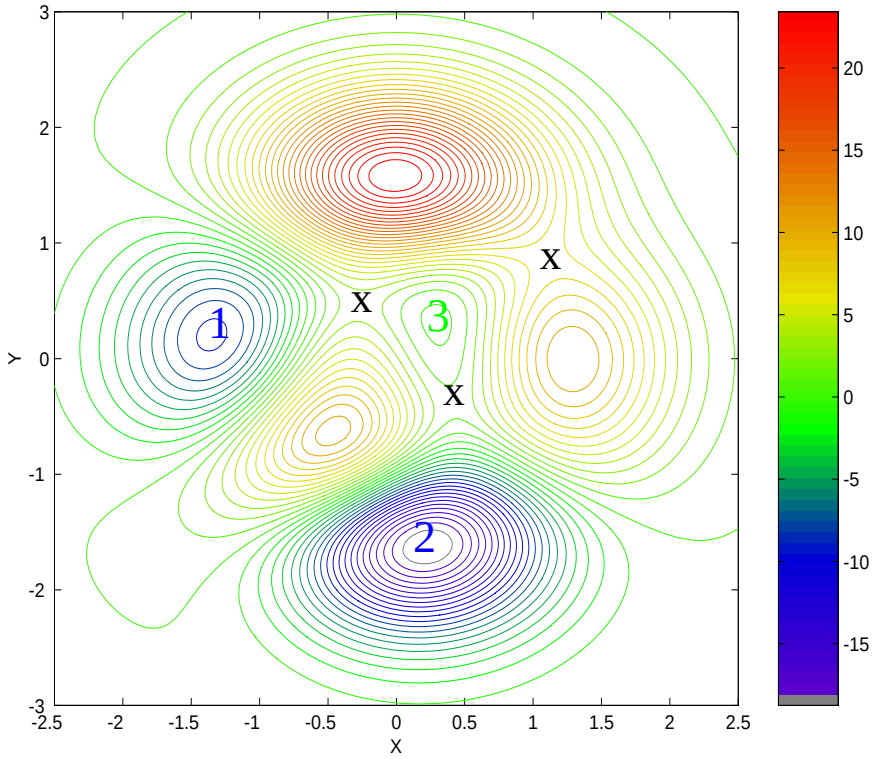
$z(x, y) := 3\cdot\text{peaks}(x, y)$ from Matlab 5.2

$$= 9\cdot(1 - x)^2\cdot\boldsymbol{exp(}-x^2 - (y+1)^2\boldsymbol{)} + (30\cdot x^3 - 6\cdot x + 30\cdot y^5)\cdot\boldsymbol{exp(}-x^2 - y^2\boldsymbol{)} -$$
$$- \boldsymbol{exp(}-(x+1)^2 - y^2\boldsymbol{)} .$$

Beyond the square $\|[x; y]\|_\infty = 3$ this $z(x, y)$ decays rapidly to 0 at infinity. Inside the square this $z(x, y)$ has three maxima, three saddle-points, and three minima of which the third is relatively shallow and hidden among the maxima.



In the contour plot below, the minima are numbered, and the three sadde-points are marked "x" . Searches started too far from them are unlikely to find Min #3.



	Min #1	Min #2	Min #3
x	0.20452	-1.6255	0.32020
y	-1.3474	0.22828	0.29645
z	-9.1495	-19.653	-0.19481

What would the Hyper-parameters' best values be?

	Min #1	Min #2	Min #3
x	0.20452	-1.6255	0.32020
y	-1.3474	0.22828	0.29645
z	-9.1495	-19.653	-0.19481
‖H‖	46.714	90.547	47.395
Ç	1.8208	1.6843	2.4954
GD+M β	0.022116	0.016797	0.050495
GD+M Δτ	0.028247	0.014092	0.031647
AGD μ	8.1389	11.616	7.3645
AGD Δτ	0.23510	0.16649	0.24546
ρ	0.14871	0.12960	0.22471

But normally we would not know them in advance.

They would have to be estimated from experience with experiments.

Let's run an experiment comparing **GD+M** with **AGD** searching for Minimum #3 because it's the one hardest to find.

For our experiment, hyper-parameters will be chosen roughly:

	Min #1	Min #2	Min #3	Chosen
GD+M β	0.022116	0.016797	0.050495	$\beta := 0.05$
GD+M $\Delta\tau$	0.028247	0.014092	0.031647	$\Delta\tau := 0.017$
AGD μ	8.1389	11.616	7.3645	$\mu = 7.5$
AGD $\Delta\tau$	0.23510	0.16649	0.24546	$\Delta\tau := 0.25$

Searches started at 64 points distributed uniformly around the square $\max(|x|, |y|) = 1$.

Searches stopped when $\|\text{Grad}(z(x, y))\|$ had shrunk by a factor 1.0e-5 .

	GD+M	**AGD**	**GD+M**	**AGD**	**GD+M**	**AGD**
	Min #1	Min #1	Min #2	Min #2	Min #3	Min #3
How often	36%	36%	34%	34%	30%	30%
Max # It'ns	34	22	24	29	37	20
Min # It'ns	14	11	12	20	21	12
Max. error	1.01e-5	1.67e-5	4.40e-6	7.73e-5	1.50e-5	1.33e-5
Min. error	2.90e-6	3.78e-6	1.09e-6	2.77e-6	5.02e-6	3.38e-6

Conclusion: Ill-Condition is best Avoided.

How?

Train Deep Learning to seek patterns that have structural meanings from which logical analyses can draw inferences more reliably.

Such patterns are less likely to be nearly redundant and consequently less likely to induce Ill-Condition.

And if $\mathfrak{L}(\mathbf{w})$ has more than one local minimum, use **AGD** to seek them.

Some may be much broader than others; distinguish them by noticing how fast gradient $g(\mathbf{w})$ changed.

∞

This document was typed into *Framemaker* 5.1.1 on a *Power Mac G4* running *OS* 9.2.2; *Matlab* 5.2 produced all the graphs. After it was "printed" to a Postscript file, it was turned into this .pdf file by *Distiller* 3.

Appendix

The time allocated to the foregoing presentation left no time for some important questions:

1» Training is an iterative process; when should it be stopped?

2» Testing Data may fail to reveal hypersensitivity to a little noise in the data; see p. 17 above. How might a *Sharp Minimum* of residual $\mathbf{£}(\underline{\mathbf{w}})$ be revealed more reliably ?

3» How should Hyperparameters like $\Delta\tau$, μ and β be chosen …

» ***Before*** $\mathbf{w}$ is much closer to a minimum of $\mathbf{£}(\mathbf{w})$ than to any other stationary point (maximum or saddle-point) ?

» ***After*** $\mathbf{w}$ is so much closer to a minimum of $\mathbf{£}(\mathbf{w})$ than to any other stationary point that graphs on pp. 35-9 are relevant ?