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

Now posted at www.eecs.berkeley.edu/~wkahan/7Nov18.pdf

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

Abstract:

We speak different languages. Here are rough translations:

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

Some practitioners of D.L. and A.I. are likely to benefit from an acquaintance with concepts like "Ill-Condition" familiar to Numerical Analysts.

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 he 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

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

◊2 A naive model of the Visual Cortex.

◊3 Inadequate rôle of logical analysis in Deep Learning.

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

◊5 Deep Learning is too slow at ~~learning~~ being indoctrinated.

◊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."**

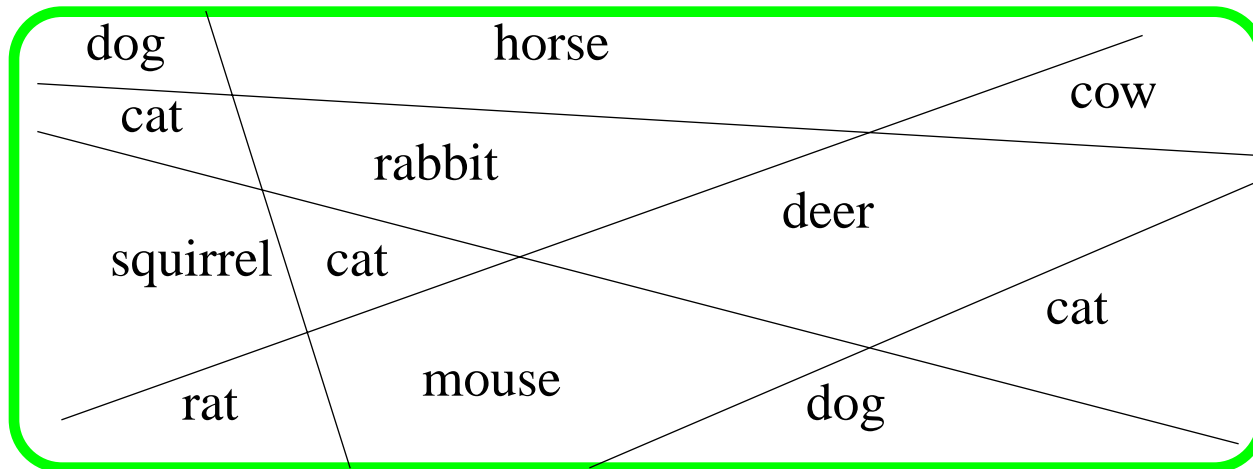pp. 104-5 of the vol. on *Mechanical Intelligence* in Turing's *Collected Works* (North-Holland)

◊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 ?

dog   horse
cat
            rabbit
                        deer   cow
squirrel   cat
                                    cat
        rat   mouse   dog

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 feedack".

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

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 Learnng 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 or squirrel.
· · · · · ·

dog

cat

horse or foal

cow or calf

rabbit

squirrel

cat

deer or fawn

cat

rat

mouse

dog
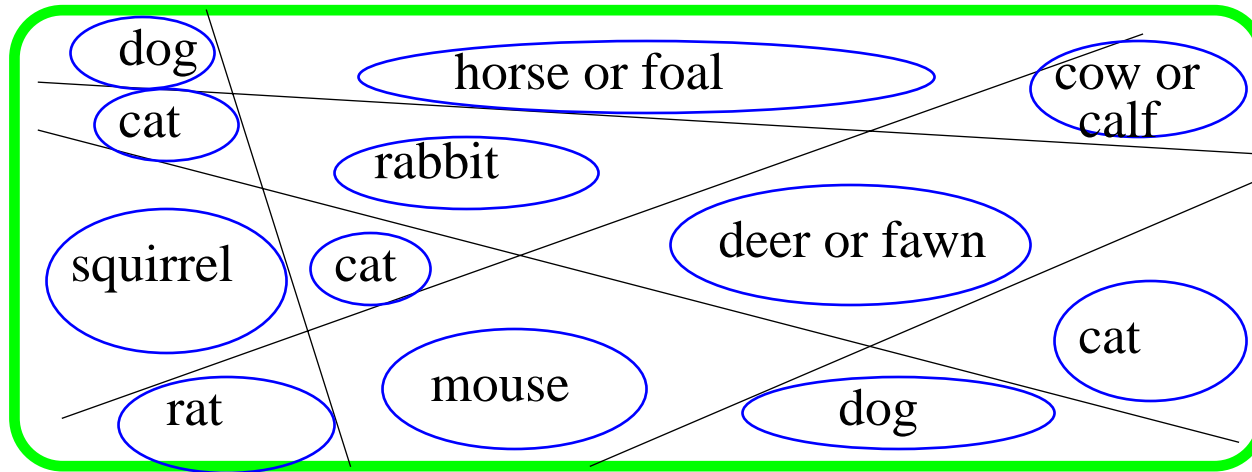
4-legged Animals

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

Outside an Oval **should imply** "I don't know", or else intelligence is dubious.

See

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. The combatants operated flight simulators, 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."

◇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 $\mathbf{y} := f(\mathbf{x})$ be the program we write to approximate f .
               $\mathbf{Y} := F(\mathbf{X})$ is the actual computation our computer performs.

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

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

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

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

Numerical Instability afflicts *program* $f(\mathbf{x})$ when its execution $F(\mathbf{X})$ 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 $\|\partial f(x)/\partial x\|$ gauges how sensitive $f(x)$ is to some small Δx :
$$y + \Delta y := f(x + \Delta x) \approx f(x) + \partial f(x)/\partial x \cdot \Delta x \quad \text{so} \quad \Delta y \approx \partial f(x)/\partial x \cdot \Delta x \ .$$
Every Δx keeps $\|\Delta y\| \leq \|\partial f(x)/\partial x\| \cdot \|\Delta x\|$ with near equality for some Δx .

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

Ill-Condition usually arises because data **X** is too close to a singularity or a redundancy or confluence of function f .
e.g.: $f(x)$ is the inverse of a nearly singular matrix 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 $-g(\mathbf{w}) := -\partial\mathbf{£}(\mathbf{w})/\partial\mathbf{w}´$.
(- 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 $\math£(\mathbf{w})$ 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 $\mathfrak{L}(\mathbf{w})$ …   A **Sharp Minimum** of $\mathfrak{L}(\mathbf{w})$ …
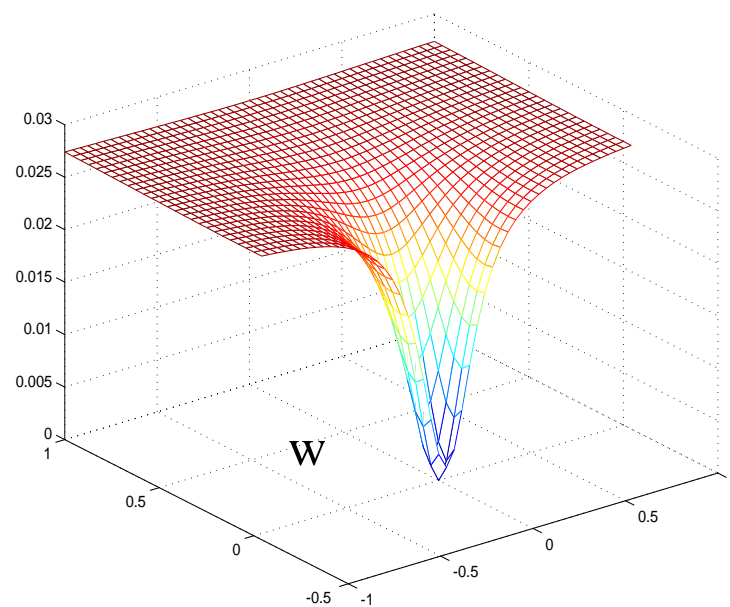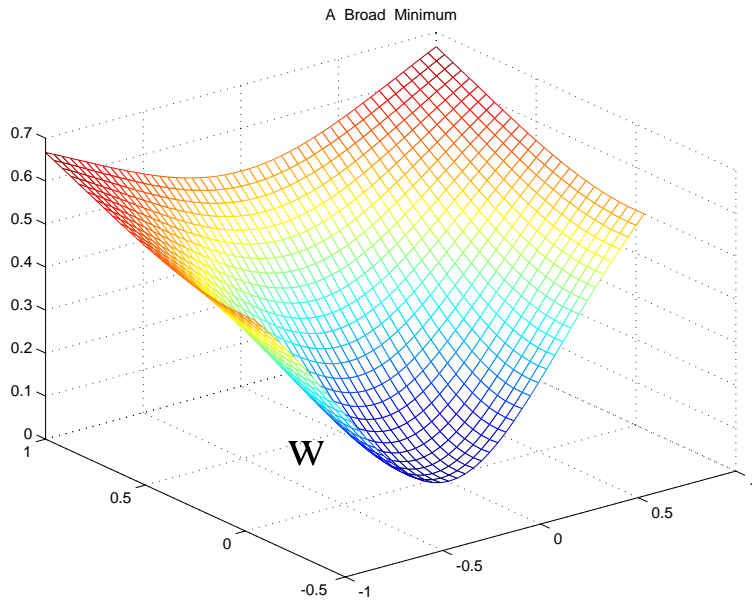
Tolerates …   is Hypersensitive to …

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

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

◊4 Disregard of **_Ill-Condition_** and **_Numerical Instability_**.

These can degrade Deep Learning …

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

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

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 the categories to be chosen 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)**

Spheres

Ellipsoids

w

Ç(H) ≈ 1.2

Ç(H) ≈ 5
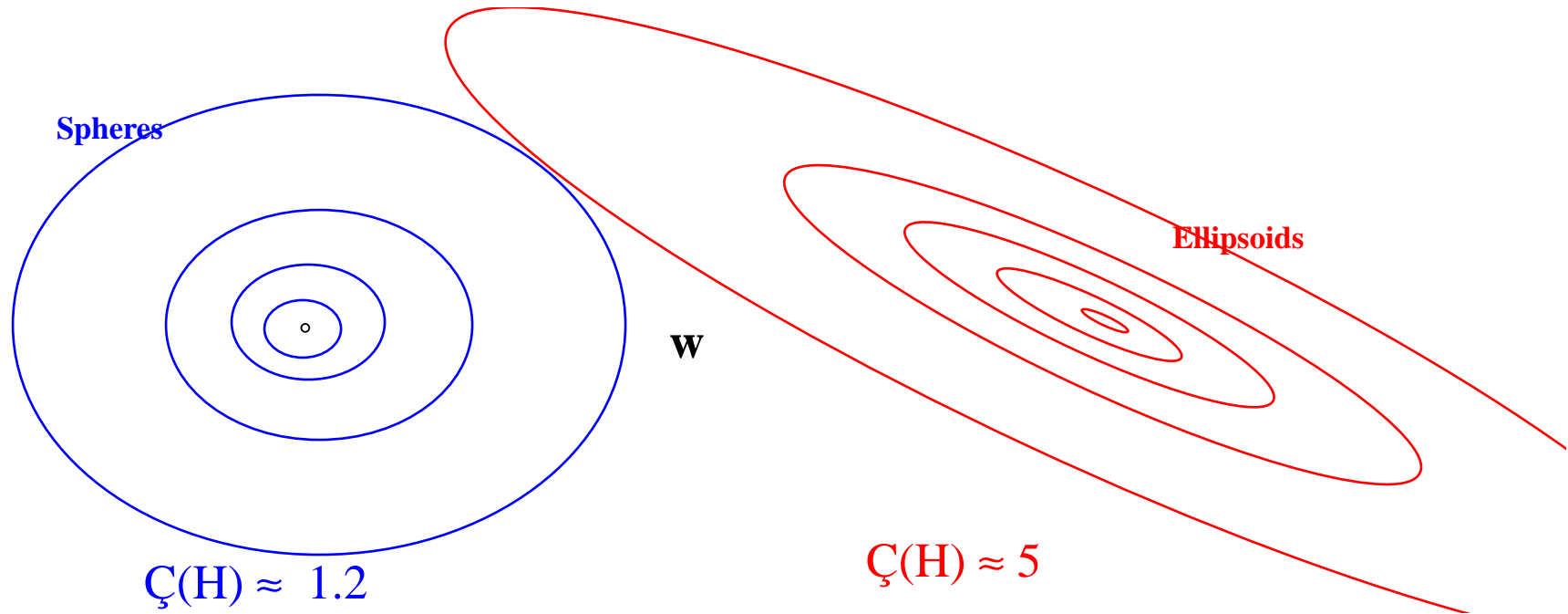
Plots for Ç(H) >> 10 appear as a family of variably spaced parallel lines**:**
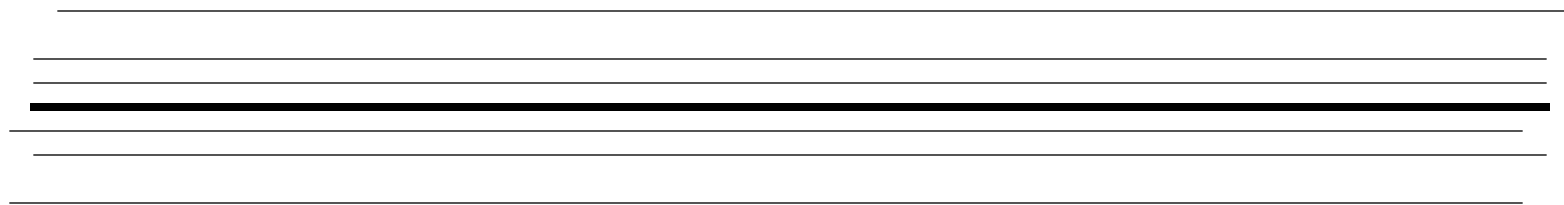
Where is the minimizing **w** ?

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

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

1: An Ill-Conditioned Hessian H($\mathbf{w}$) spawns Numerical Instability
causing minimizing weights $\mathbf{w}$ to be computed inaccurately,
usually much too big.

Weights $\mathbf{w}$ 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 £($\mathbf{w}$) 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 $\mathbf{w}$) carrying more
than twice as much precision as you trust in the data or desire in the weights $\mathbf{w}$.

2: An Ill-Conditioned Hessian H($\mathbf{w}$) protracts the search for a minimizing $\mathbf{w}$,
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 $\pounds(\mathbf{w})$ by following the
Downward gradient $-g(\mathbf{w}) := -\partial\pounds(\mathbf{w})/\partial\mathbf{w}'$.

Three such processes will be examined hereunder. All of them converge almost
always to a local minimum of $\pounds(\mathbf{w})$ at rates ultimately limited by the Hessian's
Condition Number $Ç := \|H\|\cdot\|H^{-1}\|$ where $H(\mathbf{w}) := \partial^2\pounds(\mathbf{w})/\partial\mathbf{w}^2$.
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 $d\mathbf{£}(\mathbf{w}(\tau))/d\tau = -\|g(\mathbf{w}(\tau))\|^2 \leq 0$, so $d\mathbf{£}(\mathbf{w}(\tau)$ 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 $\mathbf{£}(\mathbf{w})$.

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

Cond(H) = 5
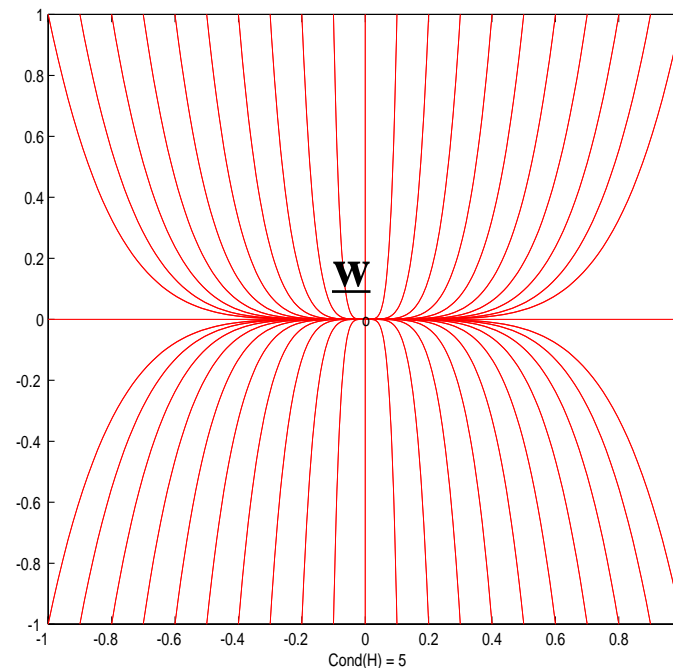
GD is Euler's discretization $\Delta\mathbf{w}/\Delta\tau = -g(\mathbf{w})$ of the Differential Equation.

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:


Cond(H) = 5
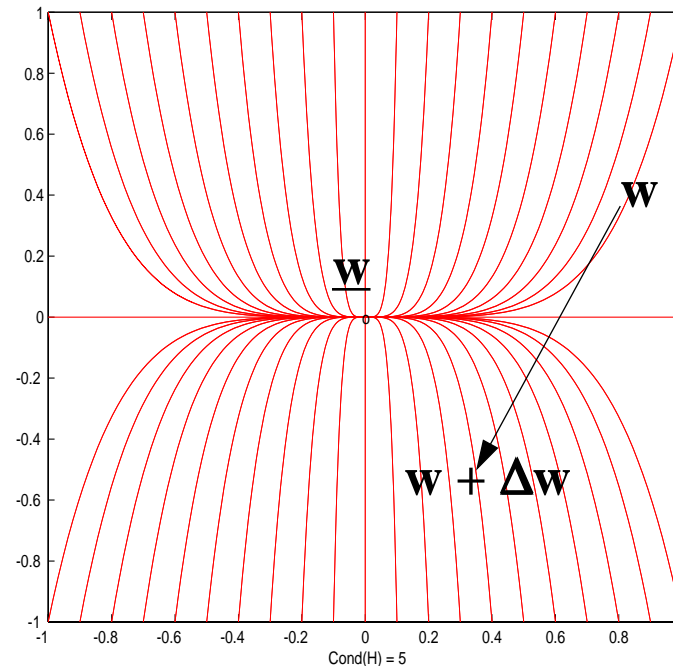
Provided $\Delta\tau < 2/\|H\|$, sucessive steps converge to $\underline{\mathbf{w}}$, very slowly if $\Delta\tau$ is too small, or else slowly by ricocheting. Ricocheting retards convergence severely if the unknown Hessian $H(\mathbf{w}) := \partial^2\pounds(\mathbf{w})/\partial\mathbf{w}^2$ is Ill-Conditioned.

Why ?

Why is Gradient Descent's convergence retarded severely by Ricocheting if the unknown condition number Ç of the Hessian H(**w**) is too big ?

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

Ç = 5   Ç = 50
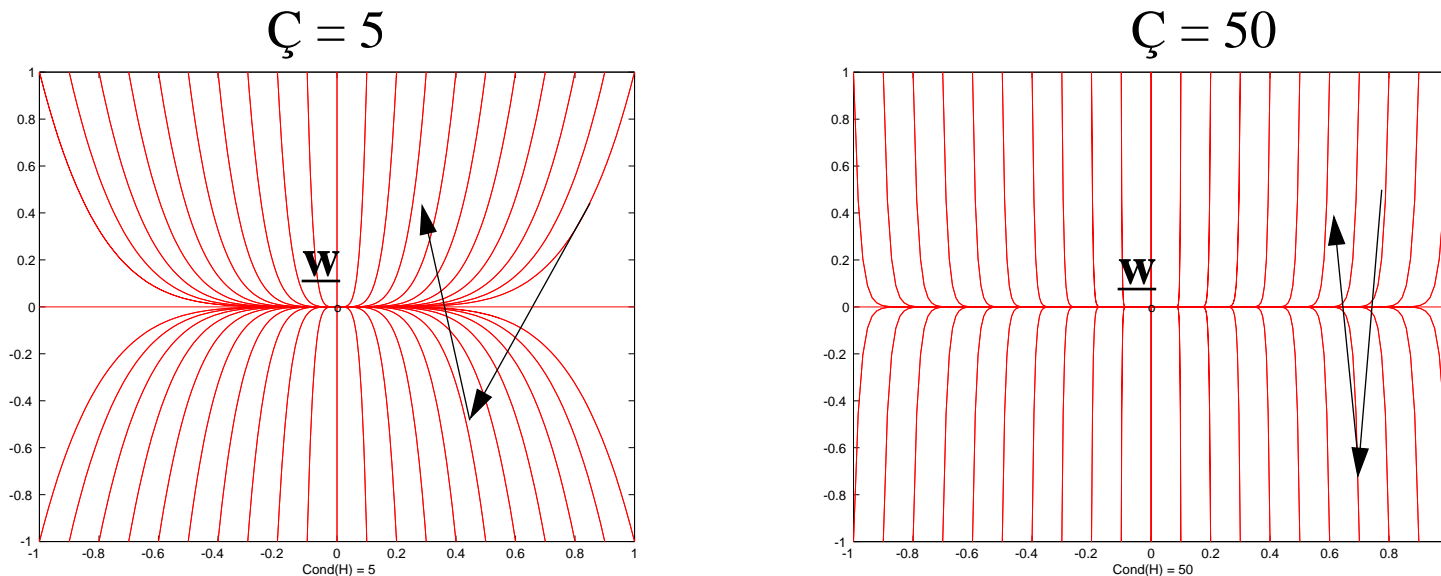
GD can't converge if $\Delta\tau \geq 2/\|H\|$ yet converges fastest if $\Delta\tau = (2/\|H\|)/(1 + 1/Ç)$ which is barely smaller when Ç is big; then the best choice of $\Delta\tau$ is delicate.

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 with Momentum

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

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 Δτ at each step from matrix-vector products like H(w)·w , but *only* when the Hessian H is accessible.

Otherwise the best constant values for Hyper-parameters β and stepsize Δτ depend upon the unknown Hessian H($\underline{w}$) and its Condition Number Ç, so they must be determined by experiment during iterations.

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

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

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

$$\rho = 1 - 2/(1 + \sqrt{\overline{Ç}}) .$$

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 retun from the sea to their native streams.

An Anadromic numerical method for approximating the solution of a differential equation like $dw(\tau)/d\tau = -g(w(\tau))$ 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 (like Euler's) are not anadromic.

Anadromic:

So What?
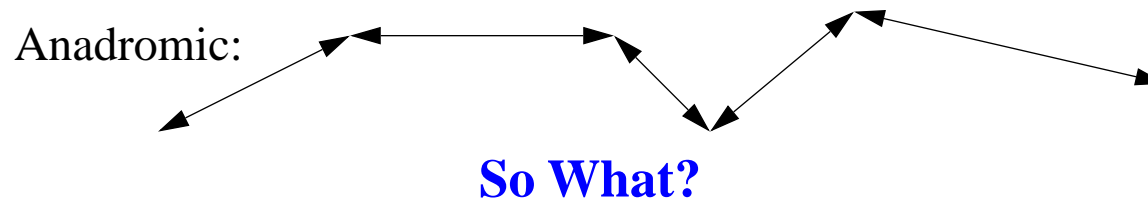
If you find an anadromic numerical method for your differential equation (how to find one is usually unobvious) the accuracy you achieve with it will usually cost less computation than if achieved with a non-anadromic numerical method of similar complexity. *cf*. My web page's mcom2498.pdf and citations there.

Anadromic Gradient Descent

Numerical methods used to solve the *Hamiltonian* differential equations for friction-free mechanical systems are often anadromic incidentally. Starting from a Hamiltonan 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 some 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 $\mu > 0$. Here is how:

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

$$d\mathbf{w}/d\tau = \partial Æ/\partial \mathbf{v}' = \mathbf{v} \quad \text{and, starting from } \mathbf{v} := \mathbf{o} \text{ initially,}$$

$$d\mathbf{v}/d\tau = -\partial Æ/\partial \mathbf{w}' - \mu\cdot\mathbf{v} = -\partial\pounds(\mathbf{w})/\partial\mathbf{w}' - \mu\cdot\mathbf{v} = -g(\mathbf{w}) - \mu\cdot\mathbf{v}.$$

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

Anadromic Gradient Descent's ODE

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

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

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

$\mathbf{v}(\tau)$ cannot ever get very big since $\quad \mu\cdot\int_0^\infty \|\mathbf{v}(\tau)\|^2 d\tau \;=\; Æ(\mathbf{w}(0), \mathbf{o}) - Æ(\underline{\mathbf{w}}, \mathbf{o})$.
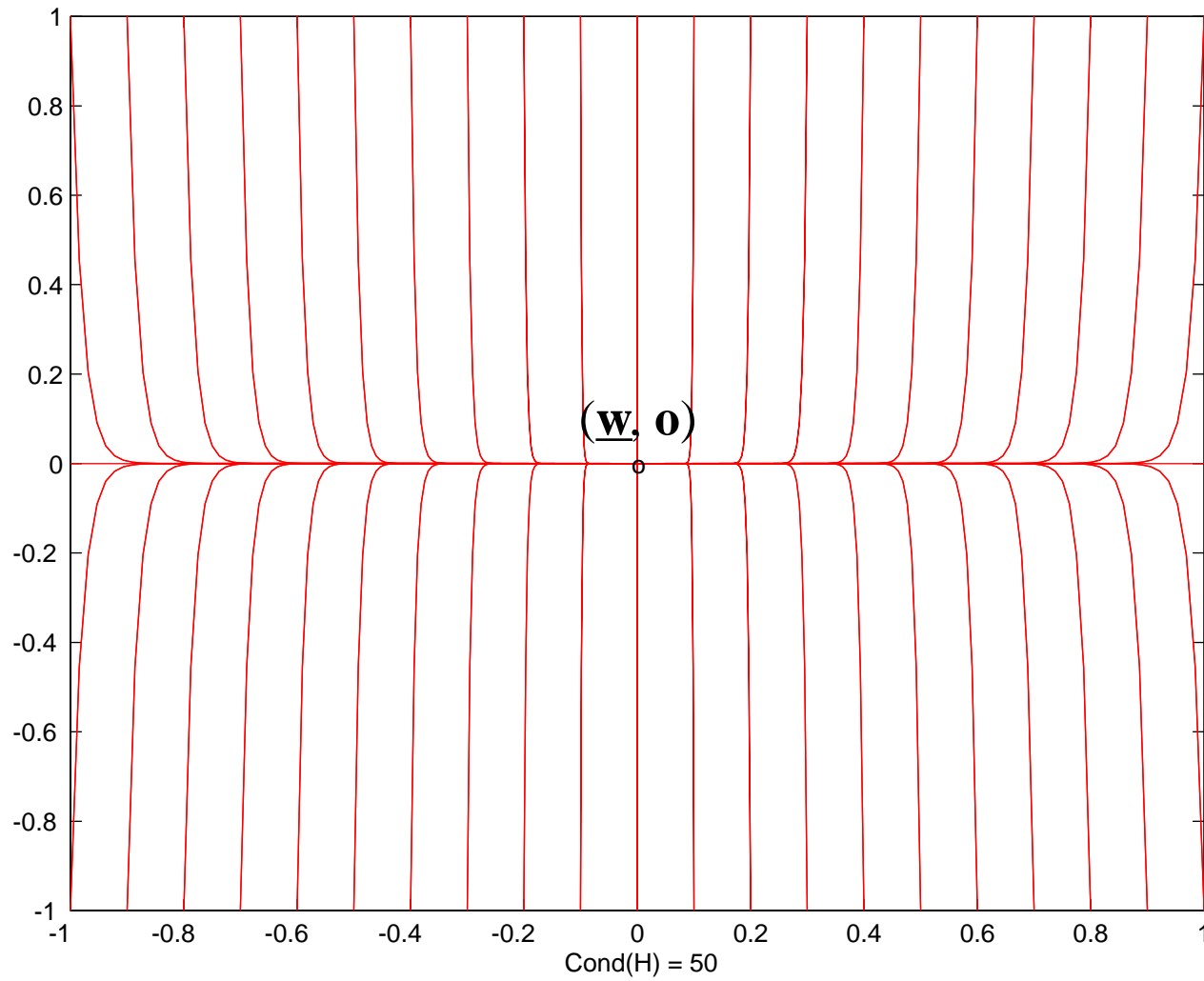
What do the trajectories look like? That depends upon the drag μ and the unknown Condition number Ç of the Hessian. Spirals occur unless $\mu \geq 2\sqrt{\|H\|}$.

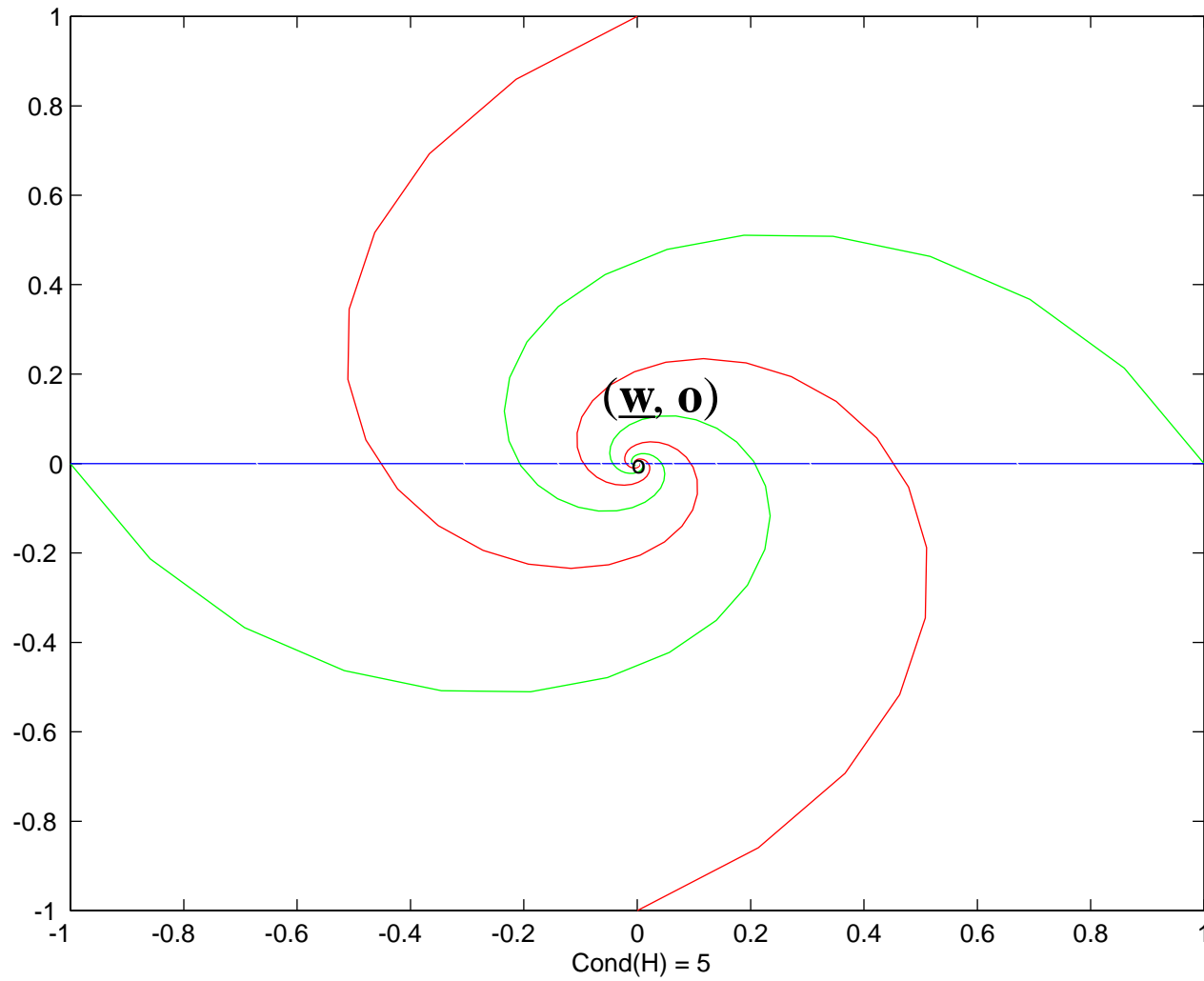The bigger is Ç the sharper almost all trajectories bend or the tighter are spirals.

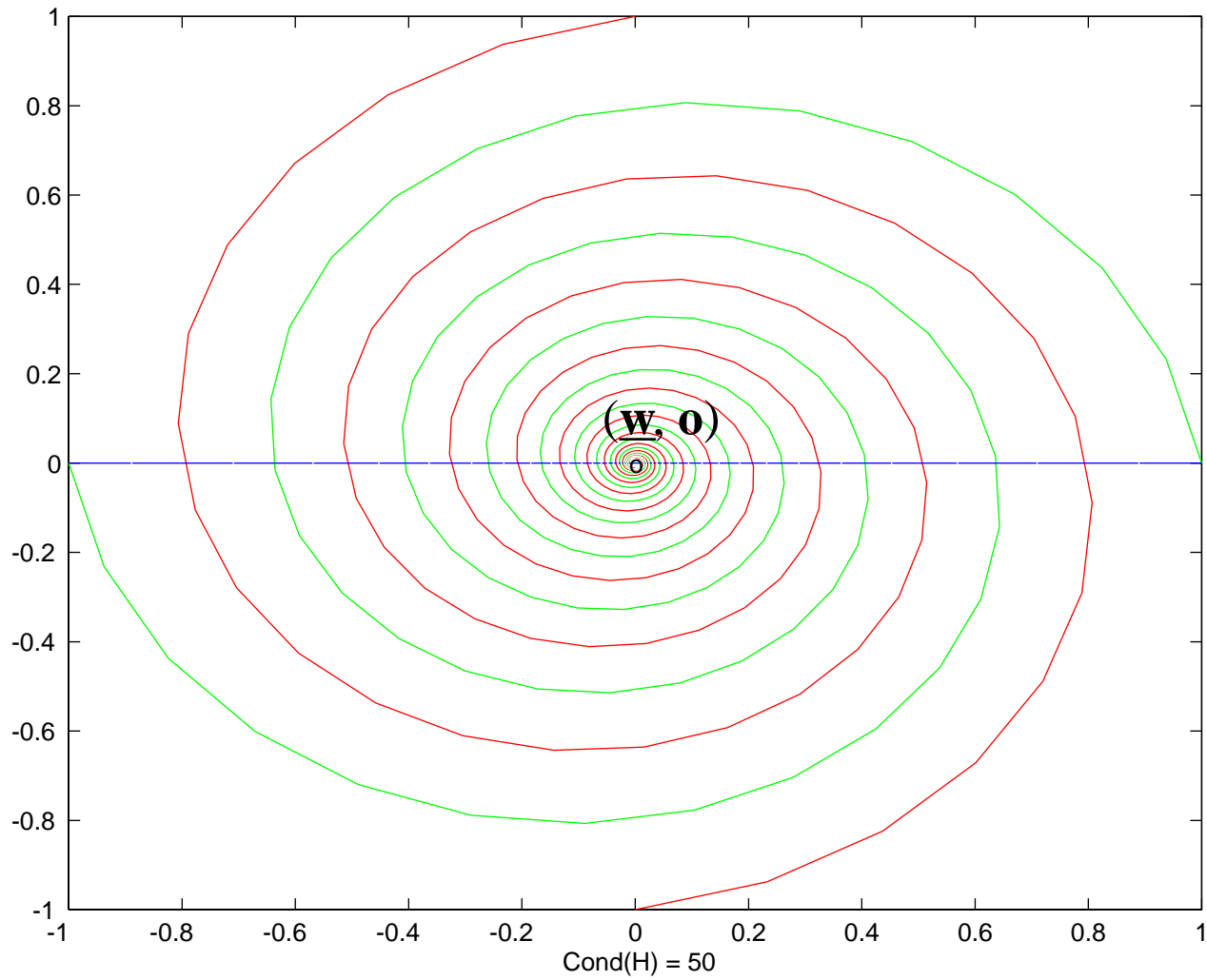## AGD ODE,   Big Drag,   Ç = 5 ,   Gradual bends
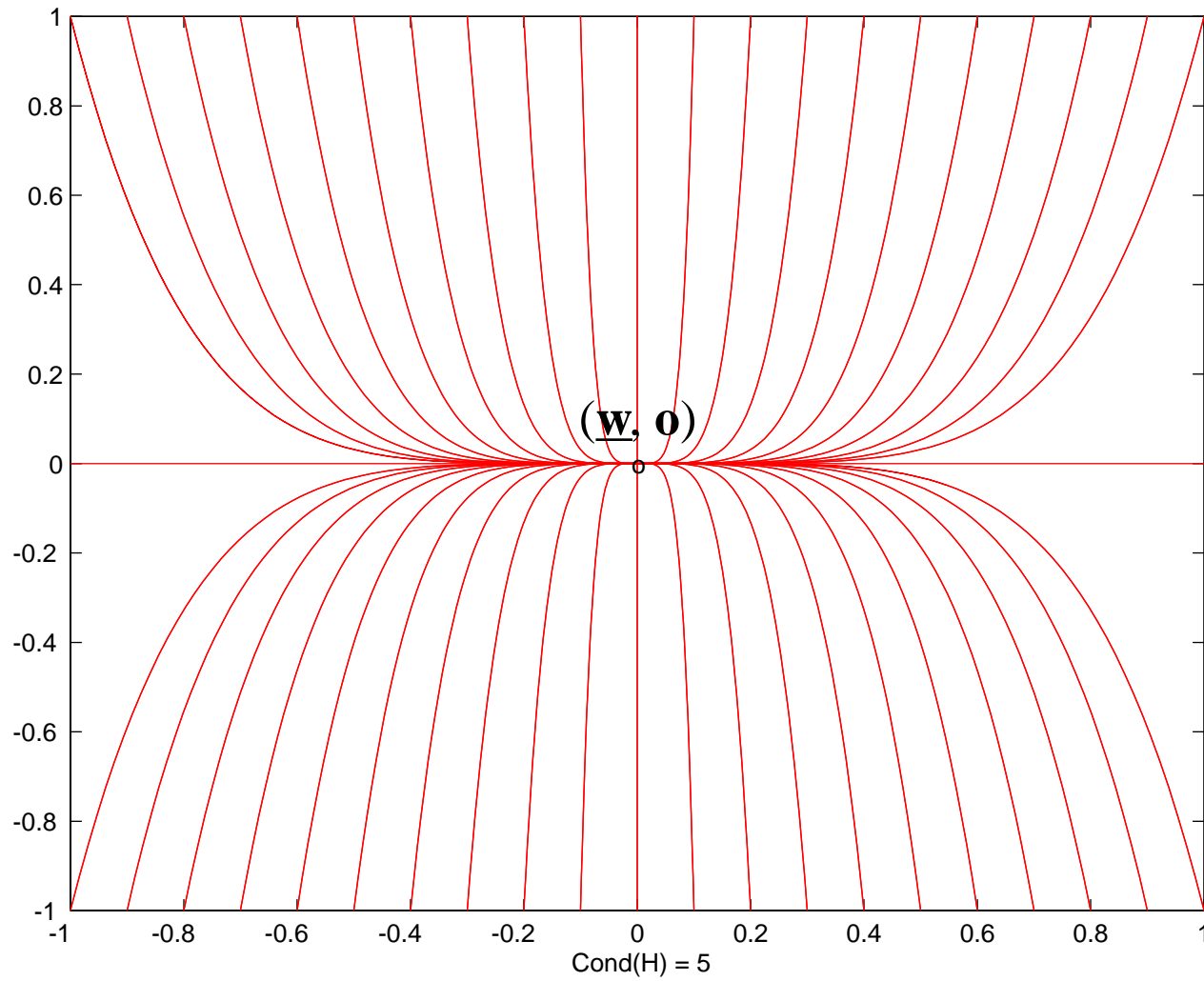
Cond(H) = 5

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

Cond(H) = 50

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

Cond(H) = 5

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

Cond(H) = 50

Anadromic Gradient Descent's Algorithm

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

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

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

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

(Save work by eliding the iteration's last step with the next iteration's first.)

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

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

If $H(\underline{\mathbf{w}})$ and its Ç were known, AGD's best Hyper-parameters would be
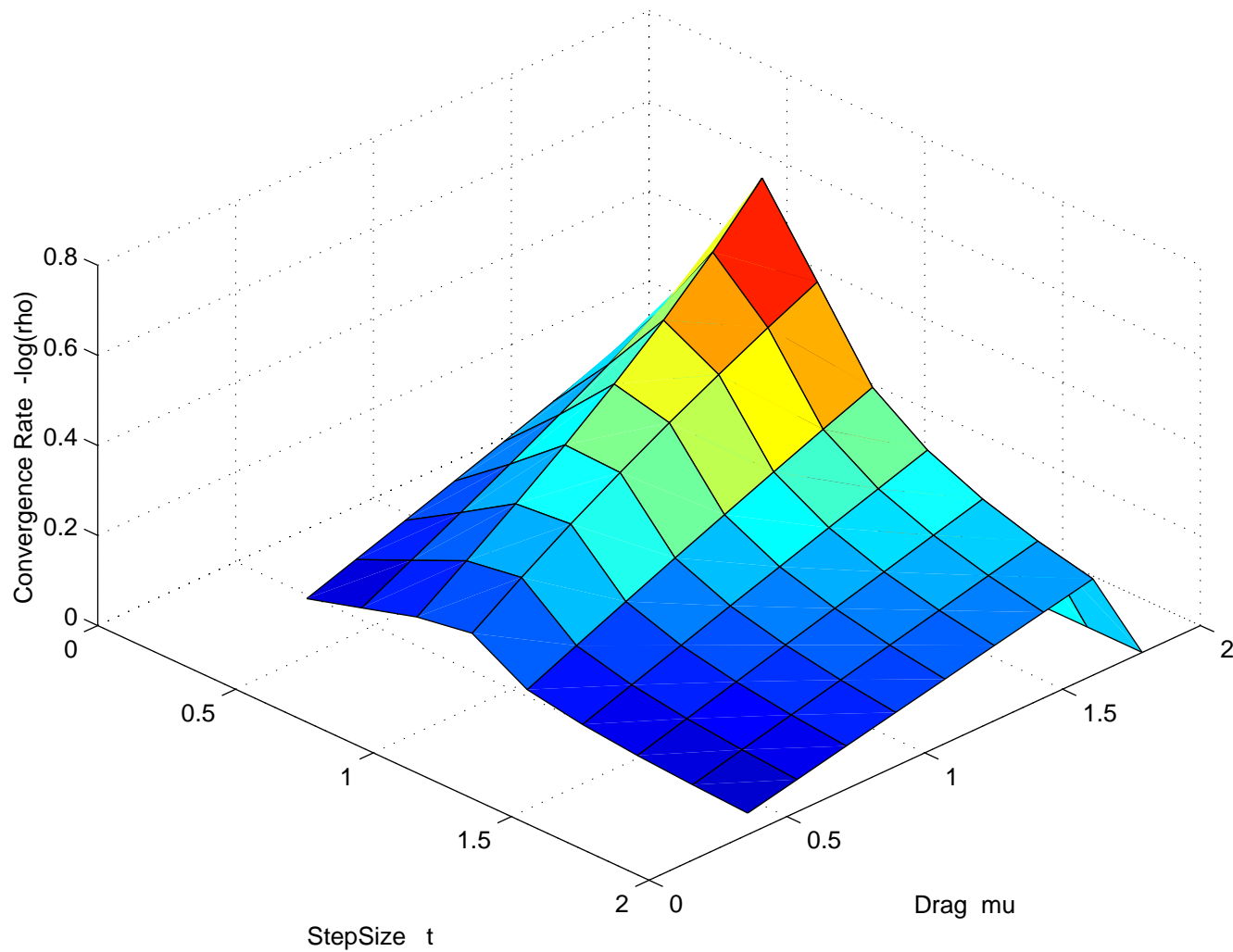$\mu := 2\sqrt{\|H\|}/\sqrt{(1 + Ç)}$ and $\Delta\tau := (2/\sqrt{\|H\|})/\sqrt{(1 + 1/Ç)}$;
and their convergence ratio $\rho := \|\text{new } \mathbf{w} - \underline{\mathbf{w}}\|/\|\mathbf{w} - \underline{\mathbf{w}}\|$ would be at most
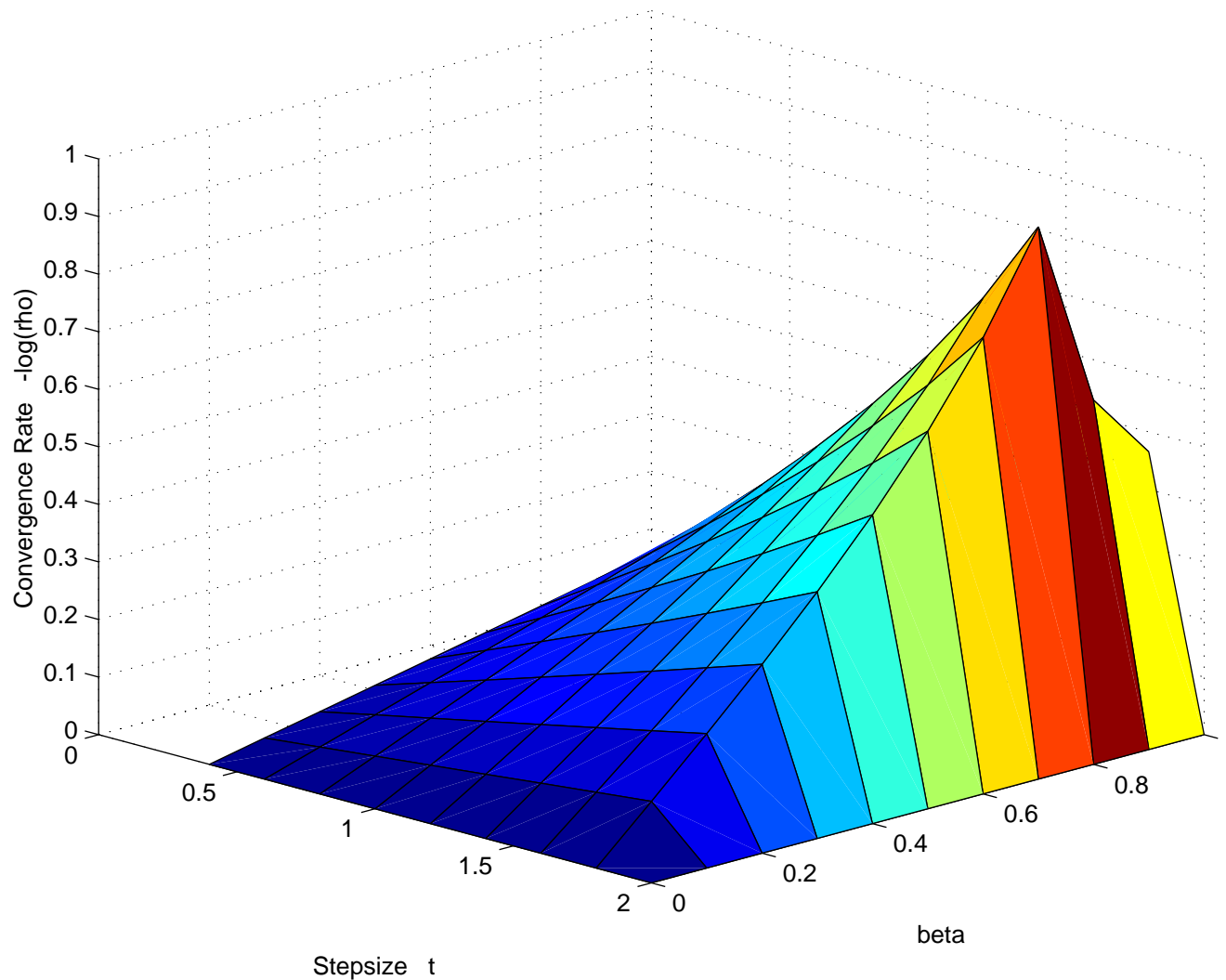$$\rho = 1 - 2/(1 + \sqrt{Ç}) .$$

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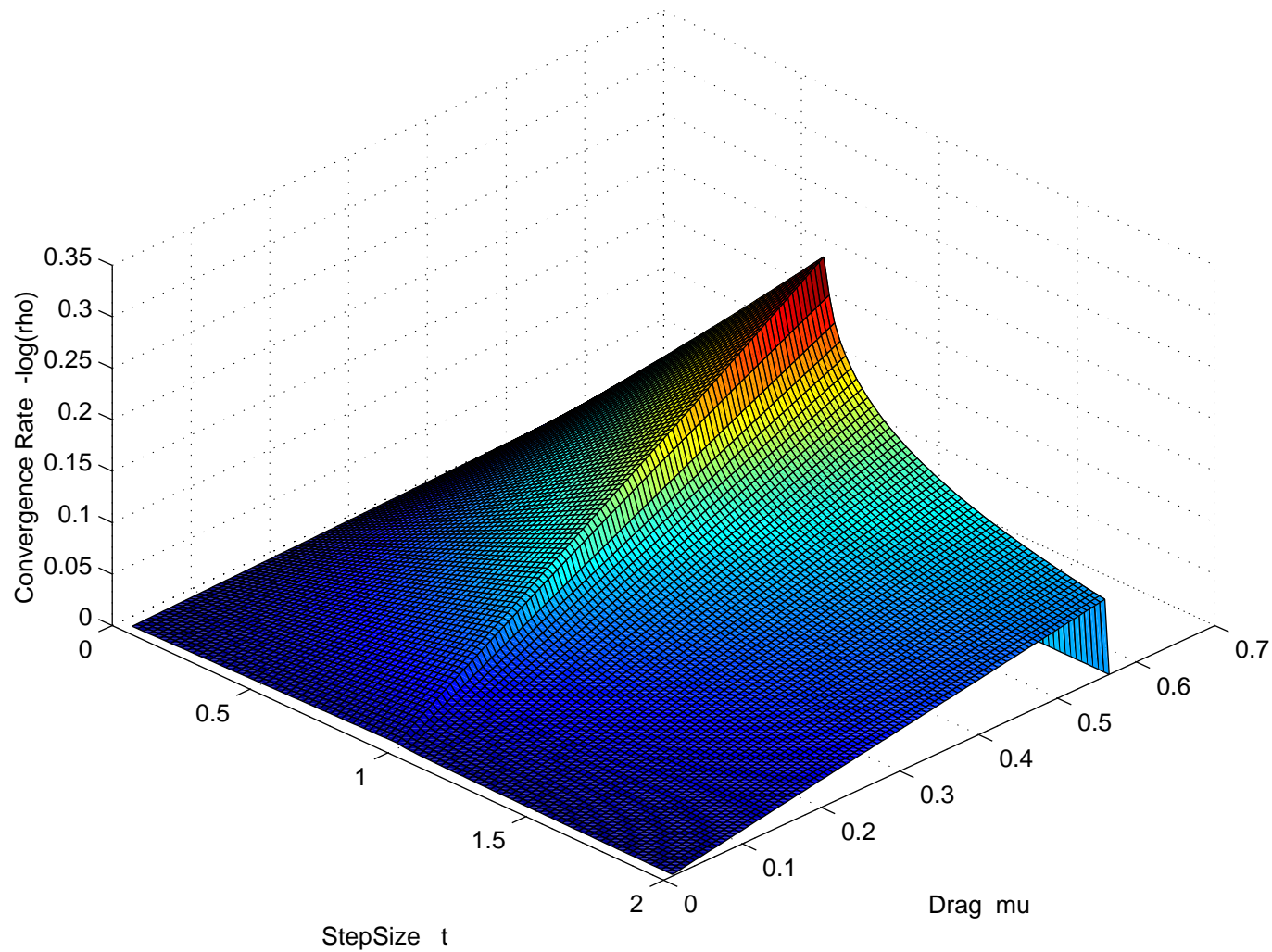(The bigger is $-\log(\rho)$, the faster is convergence,)

AGD   Ç = 5

Anadromic Gradient Descent: ||h|| = 1,   Cond'n = 5

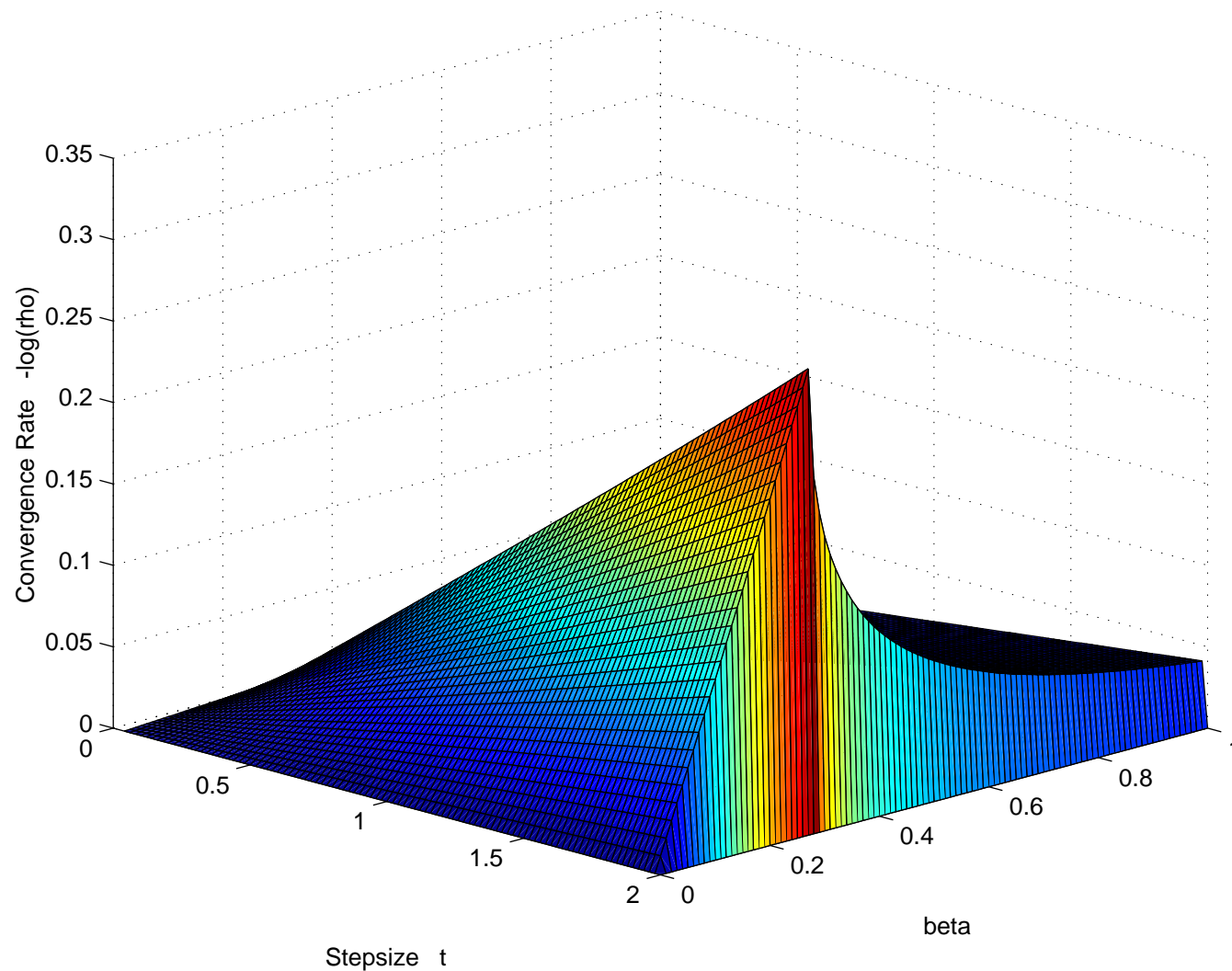Notice broad maximum as a function of stepsize $\Delta\tau$

GD+M   Ç = 5

Gradient Descent + Momentum: ||H|| = 1 ,   Cond'n = 5

Notice precipitous drop in rate if Δτ is slightly too big.

AGD   Ç = 50

Anadromic Gradient Descent: ||h|| = 1,   Cond'n = 50

Note relatively broad maximum as a function of $\Delta\tau$

GD+M   Ç = 50

Gradient Descent + Momentum: ||H|| = 1 ,   Cond'n = 50

Notice precipitous drop in rate if Δτ is slightly too big.

Conclusion: Ill-Condition is best Avoided.

How?

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

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

And if £(w) has more than one local minimum,
use AGD to look for them.

Some may be much broader than others;
distinguish them by watching how g(w) changed.

∞