What is Time?

Jos Stam

Autodesk Research
University of Toronto

Eurographics keynote talk
Lyon, France
April 28, 2017

Restaurant Eurographics 2017

Menu Prix Fixe

Confit de Nucleus a la sauce Maya

Escaloppe Temporelle cuite avec une Gratine a la sauce Canadienne

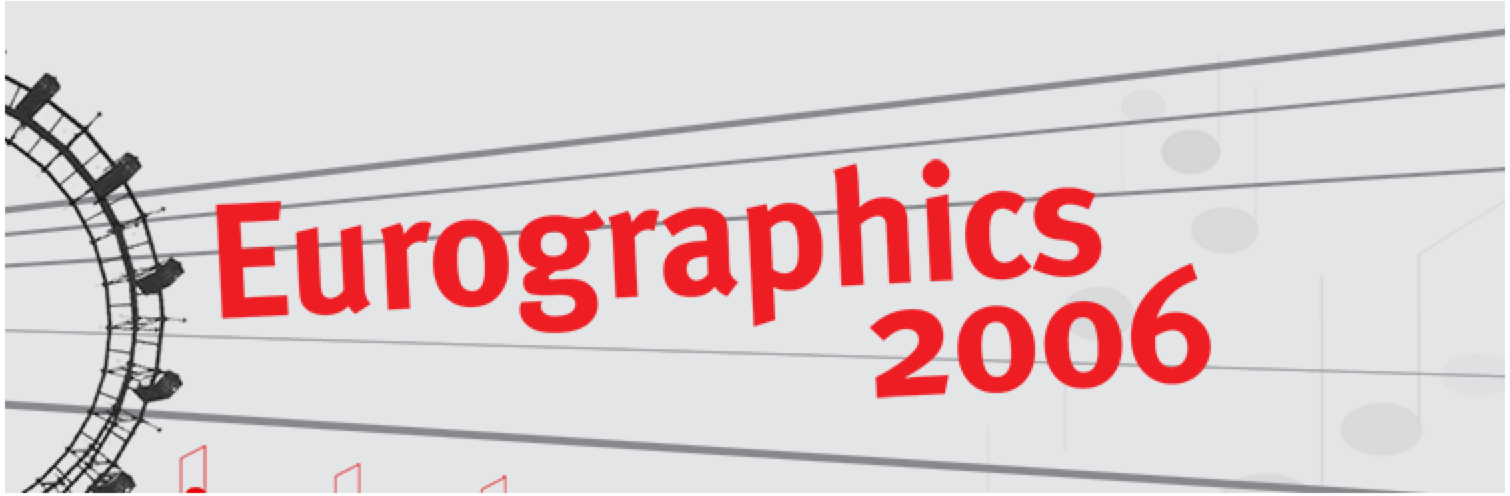
Un Flambe a l'Optimizsation parseme avec des nombres duels

Vin de Table: **The Art of Fluid Animation**, Grand Cru Chinois.

Entree

Entree

Nucleus: 10 years after…



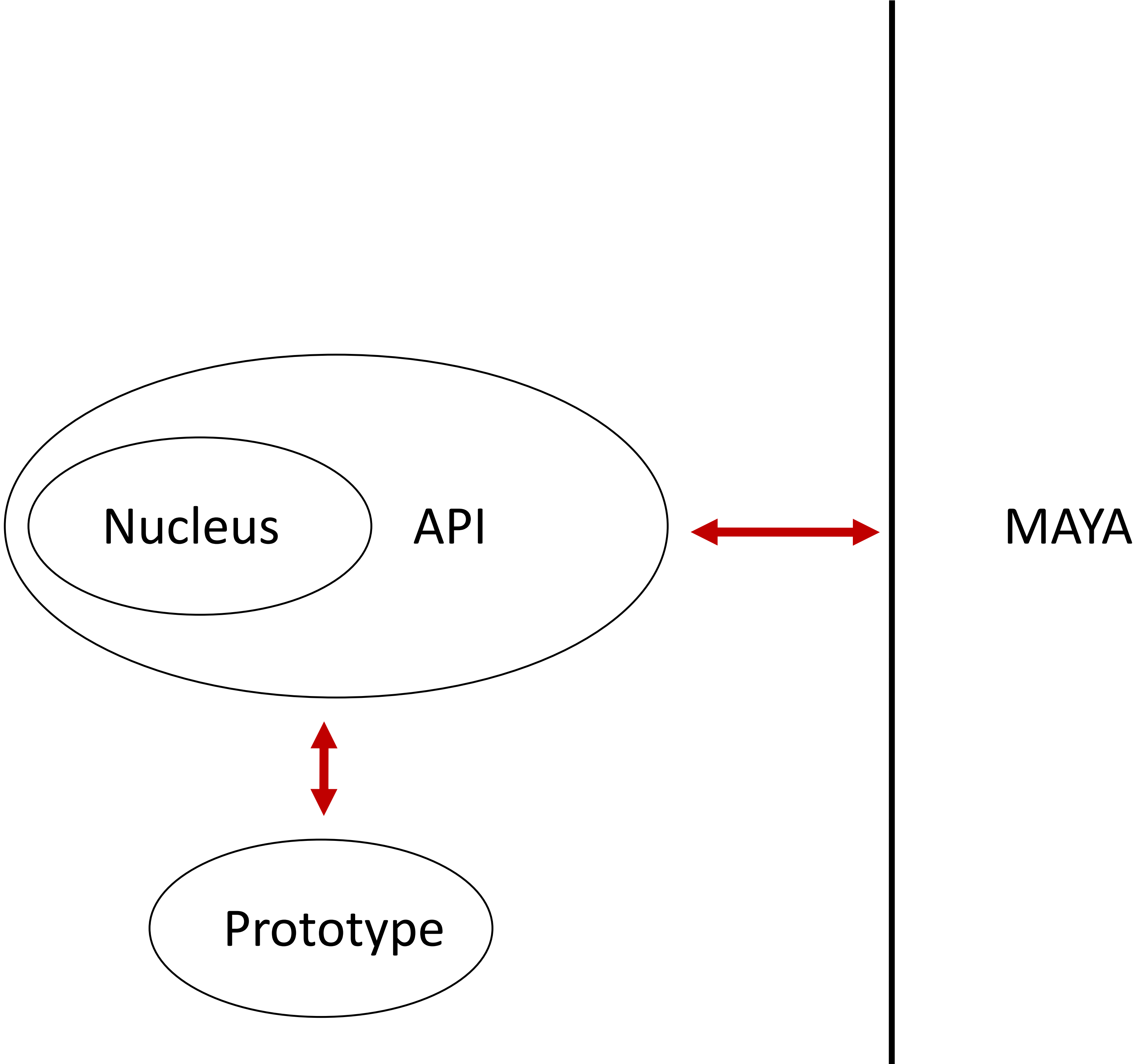
Nucleus World Tour

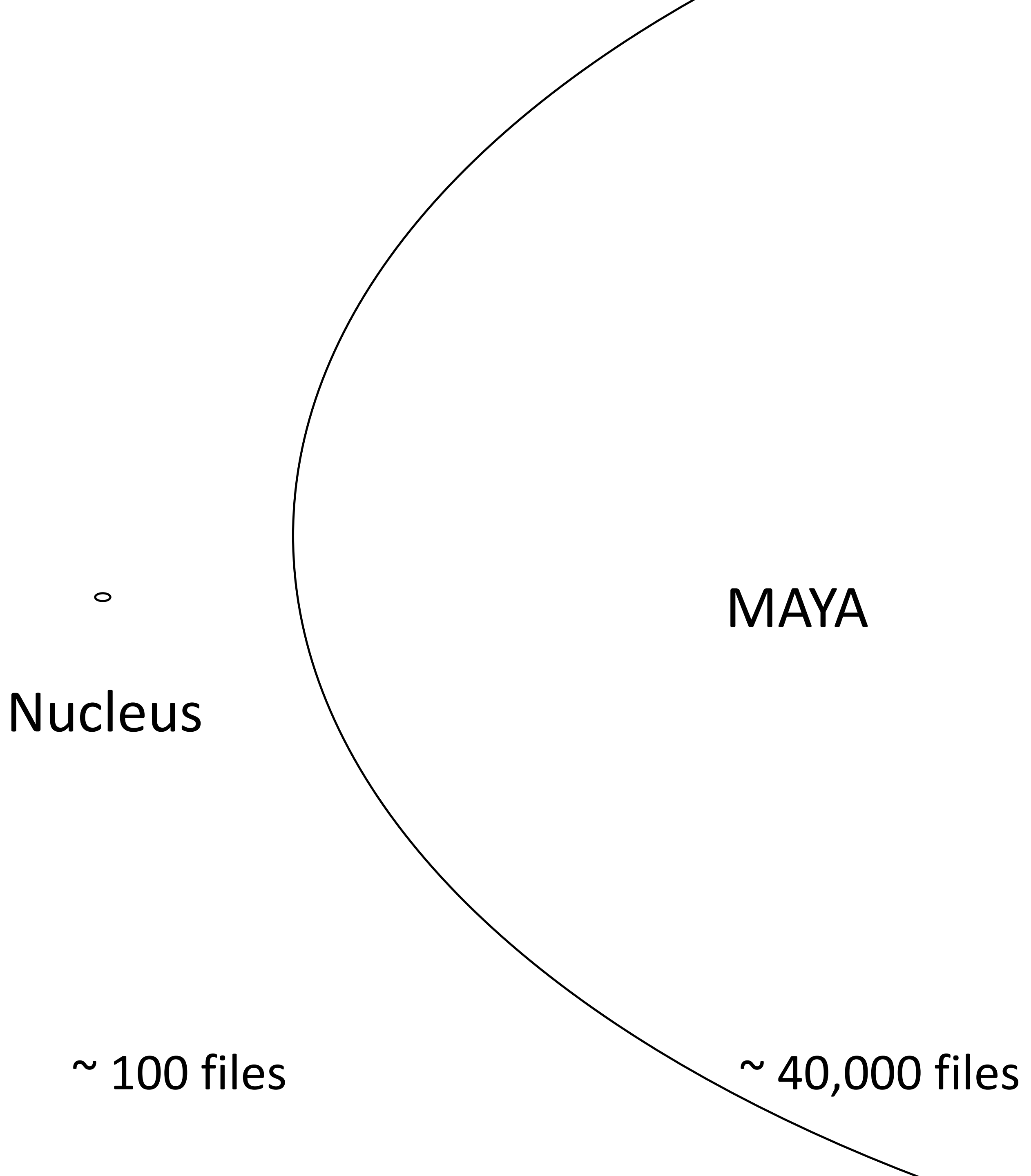
Vienna,	Sep 2006
London,	Sep 2006
Montreal,	Nov 2006
Costa Mesa,	Feb 2007
Los Angeles,	Apr 2007
Norrkoping,	Apr 2008
Toronto,	May 2008
Dublin,	Jul 2008
Los Angeles,	Aug 2008
Campo Grande,	Oct 2008
Rio de Janeiro,	Oct 2008
Vancouver,	Mar 2009
Shanghai,	Aug 2009
Yellow Mountain,	Aug 2009

Las Vegas,	Nov 2009
Ottawa,	Jun 2010
Hangzhou,	Jun 2010
Wellington,	Sep 2010
Melbourne,	Oct 2010
CharlotteVille,	Apr 2011
Honolulu,	Oct 2011
Singapore,	Mar 2012
Tokyo,	Oct 2012
Copenhagen,	May 2013
Vienna,	Sep 2013
Barbados,	Feb 2014
Banff,	Feb 2014
Shenzhen,	Mar 2017
Lyon,	Apr 2017

Computer Graphics, Game Developers, Applied Mathematicians, Undergrad Students , Architecture and the Media.

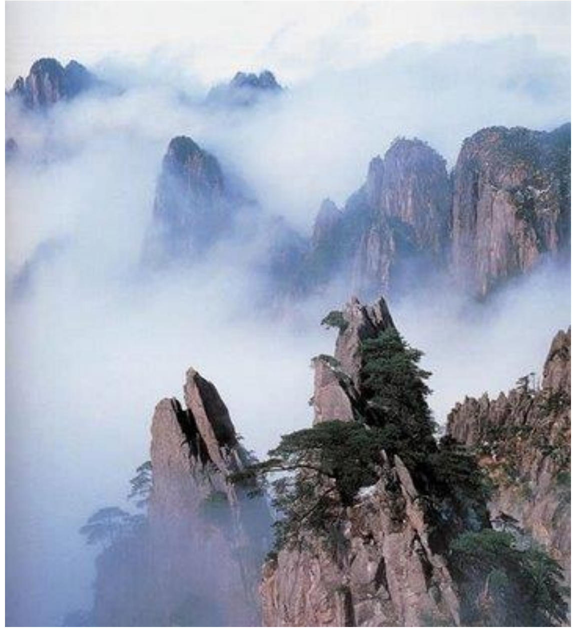
Ruysdael (1628-1682)


Nucleus
API
MAYA
Prototype


MAYA
Nucleus
~ 100 files
~ 40,000 files

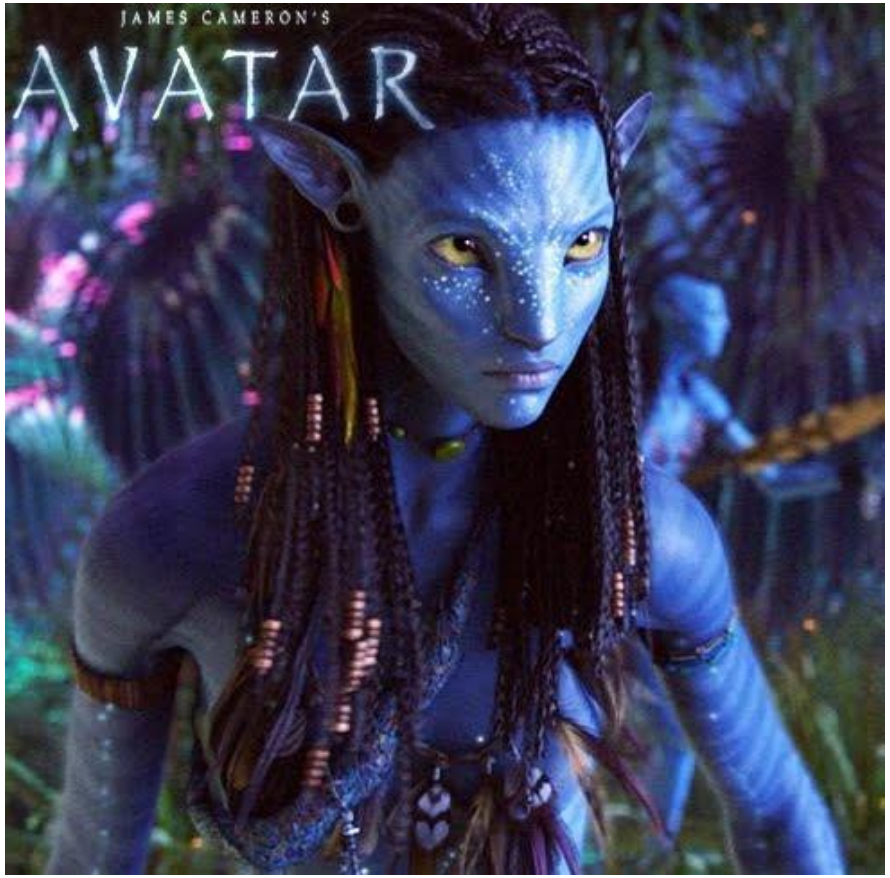
Publication

Jos Stam. (2009).
Nucleus: Towards a Unified Dynamics Solver for Computer Graphics
2009 Conference Proceedings. IEEE International Conference on Computer-Aided Design and Computer Graphics. pp. 1-11.



Yellow Mountain

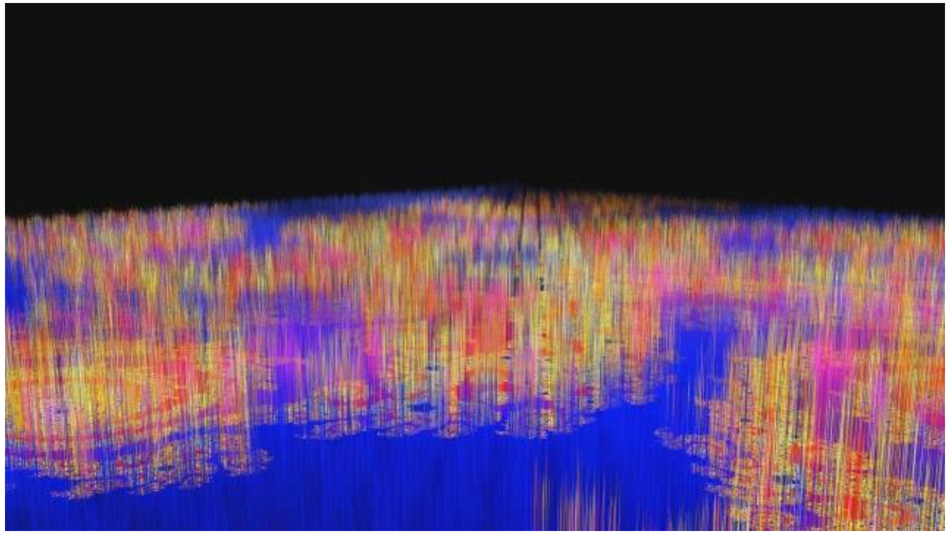
Used in Movies



Weta FX, Wellington, New Zealand

nHair Examples

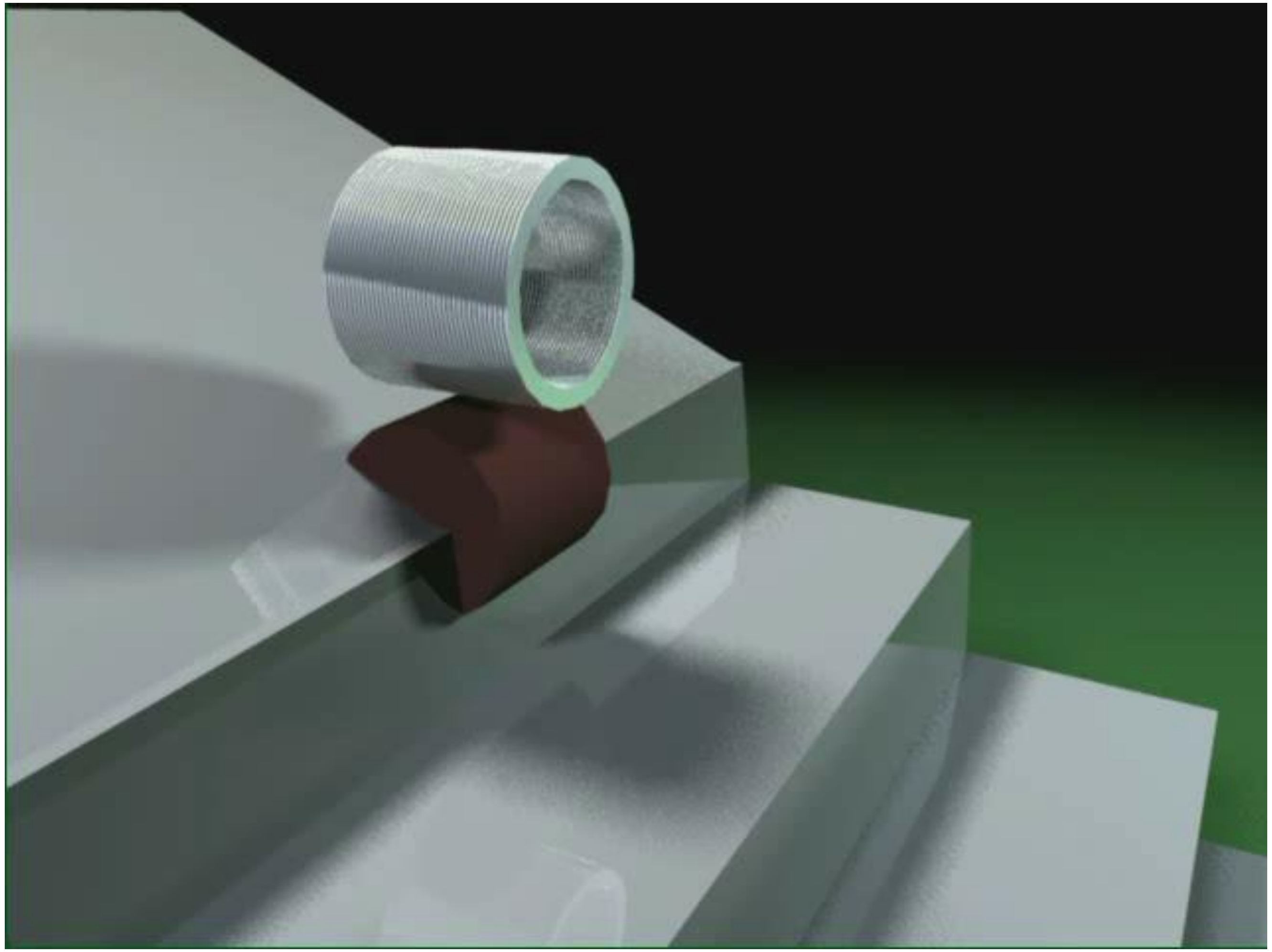
Thanks Ken Taki!



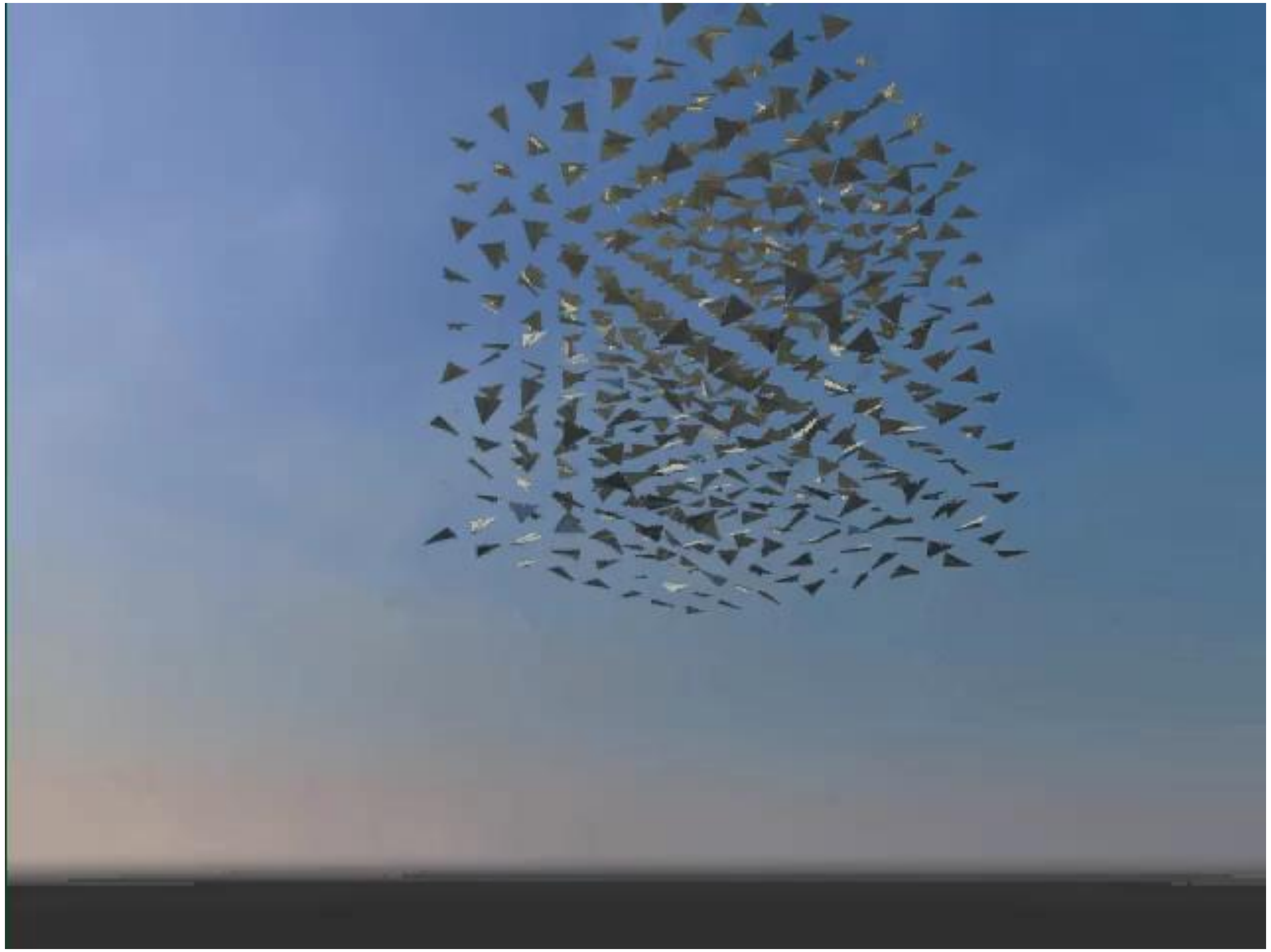
Duncan’s Corner

http://area.autodesk.com/blogs/duncan

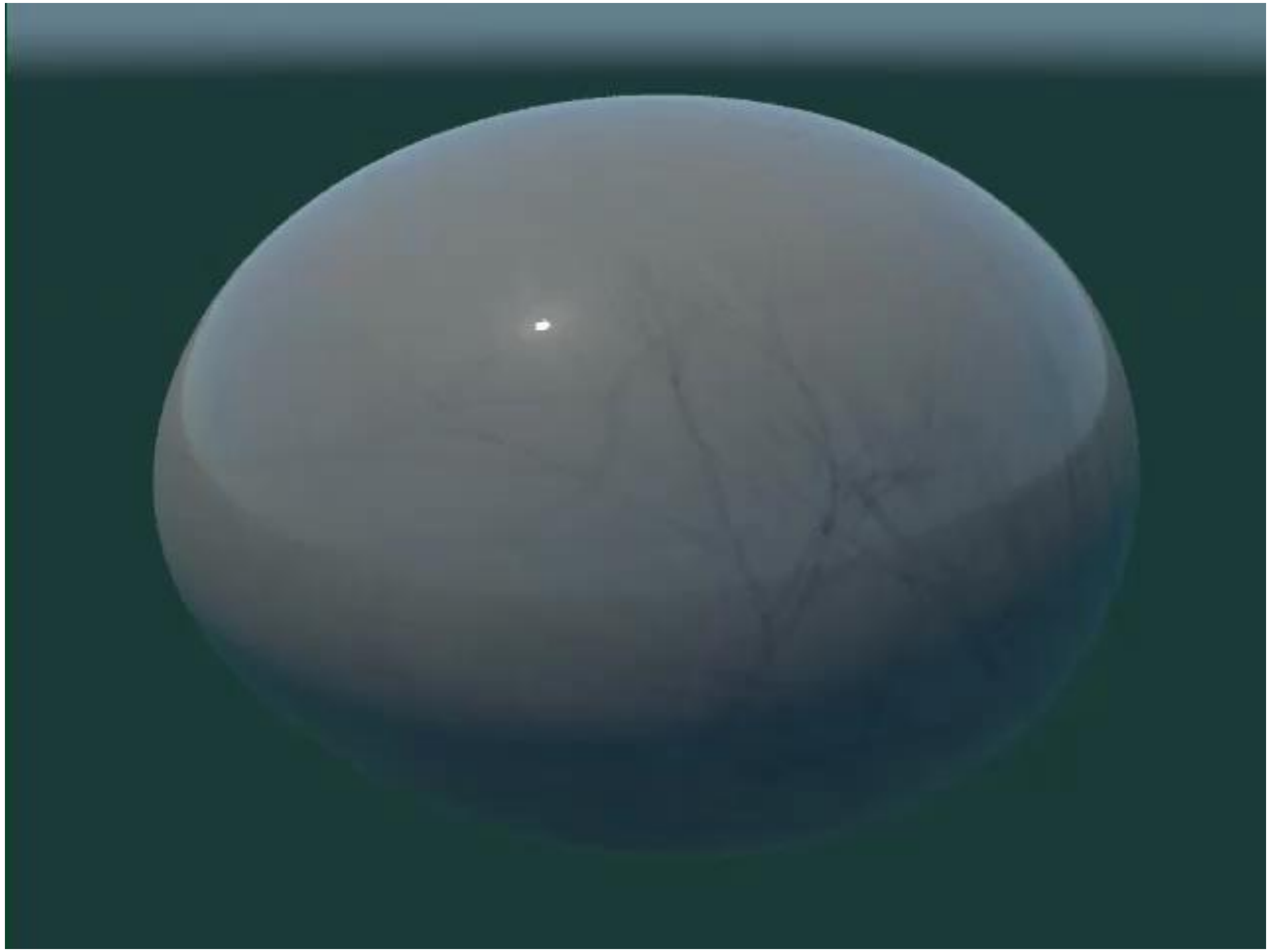
Slinky



Paper Airplanes



The Brain



nParticles and Fluid Effects

Main Course

Main Course

What is Time?

... tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic, toc, tic ...

Au prochain top il sera exactement...

St. Augustine

348-430 AD

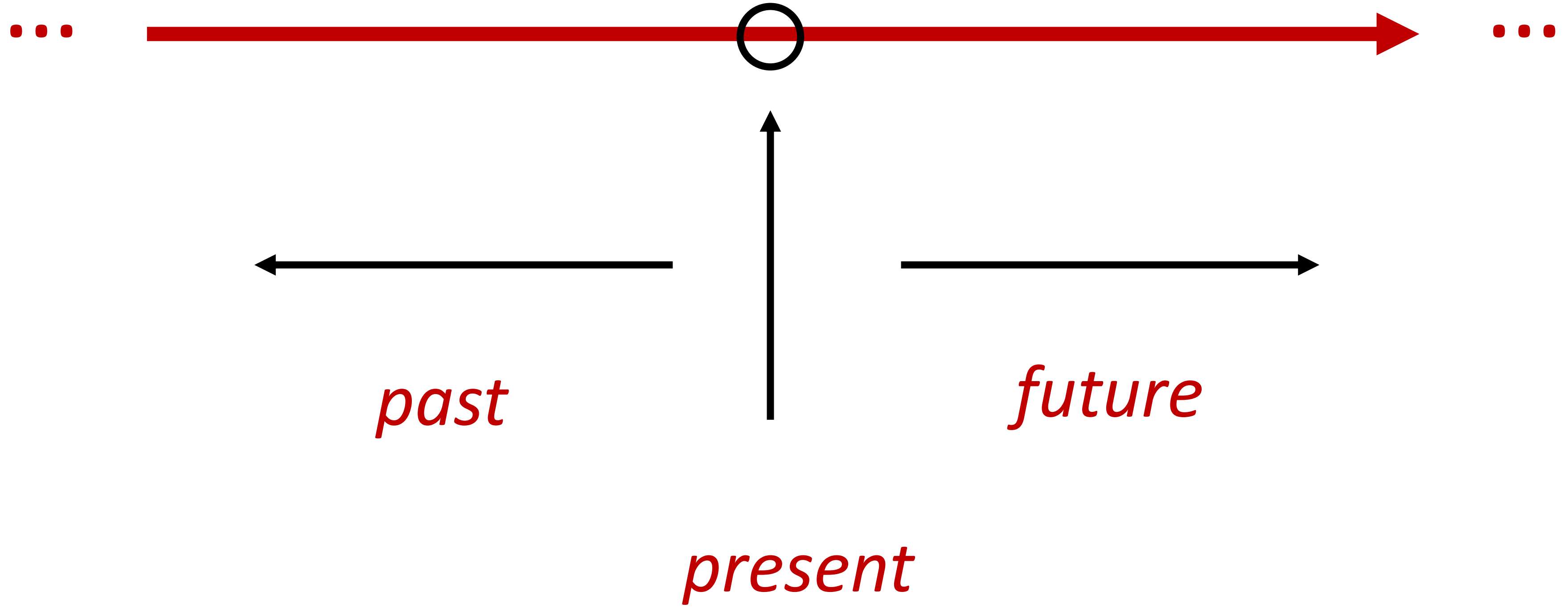
Time Does not Exist

Quid est enim tempus?
Quis hoc facile breuiterque explicauerit? Quis hoc ad uerbum de illo proferendum uel cogitatione comprehenderit?
Quid autem familiarius et notius in loquendo commemoramus quam tempus?
Et intellegimus utique cum id loquimur, intellegimus etiam cum alio loquente id audimus.
Quid est ergo tempus? Si nemo ex me quærat, scio; si quærenti explicare uelim, nescio.
Fidenter tamen dico scire me quod, si nihil præteriret, non esset præteritum tempus, et si nihil adueniret,
non esset futurum tempus, et si nihil esset, non esset præsens tempus.
Duo ergo illa tempora, præteritum et futurum, quomodo sunt, quando et præteritum iam non est et futurum nondum est?
Præsens autem si semper esset præsens nec in præteritum transiret, non iam esset tempus, sed æternitas.
Si ergo præsens, ut tempus sit, ideo fit, quia in præteritum transit, quomodo et hoc esse dicimus, cui causa, ut sit,
illa est, quia non erit, ut scilicet non uere dicamus tempus esse, nisi quia tendit non esse?

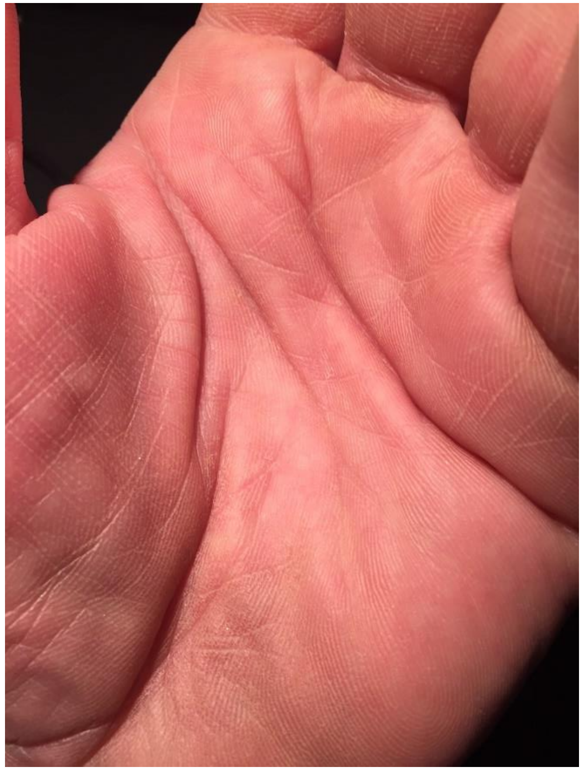
Time *is a fiction created by the mind.*

Infinite Linear Time

Infinite Linear Time

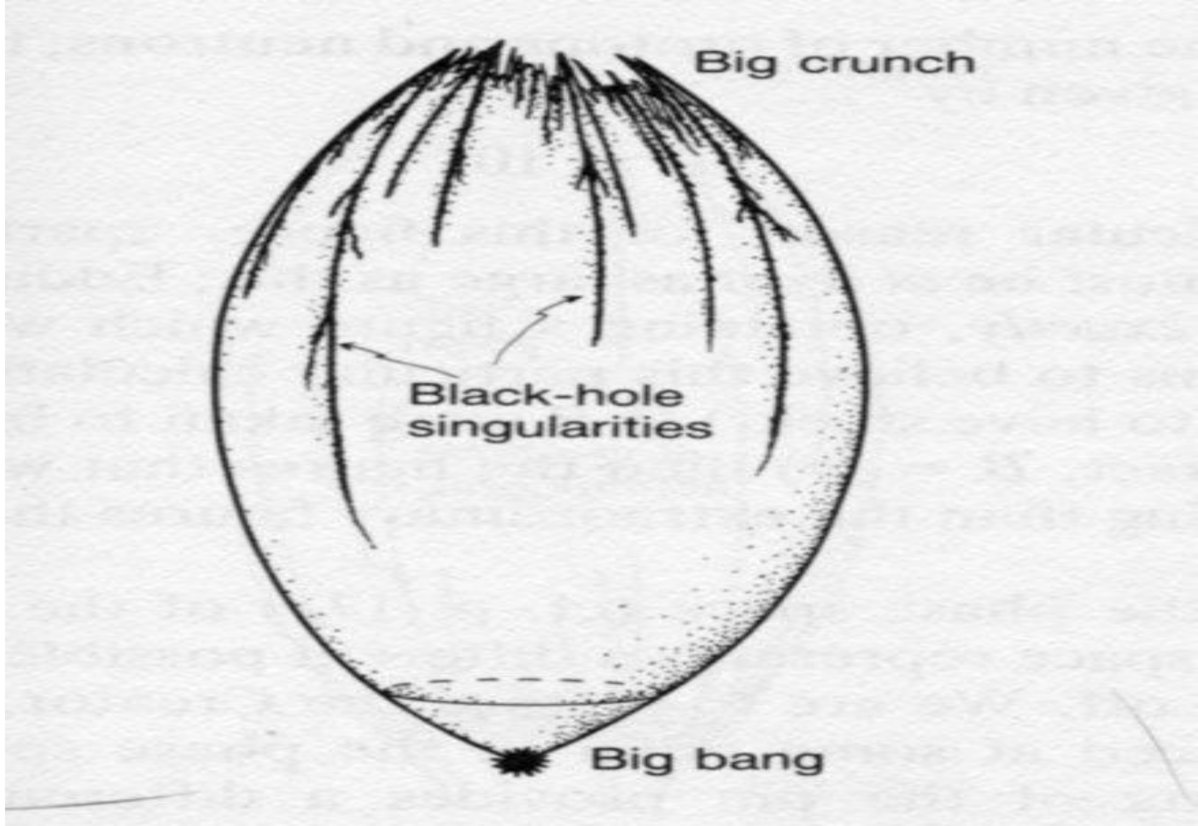


Time is finite



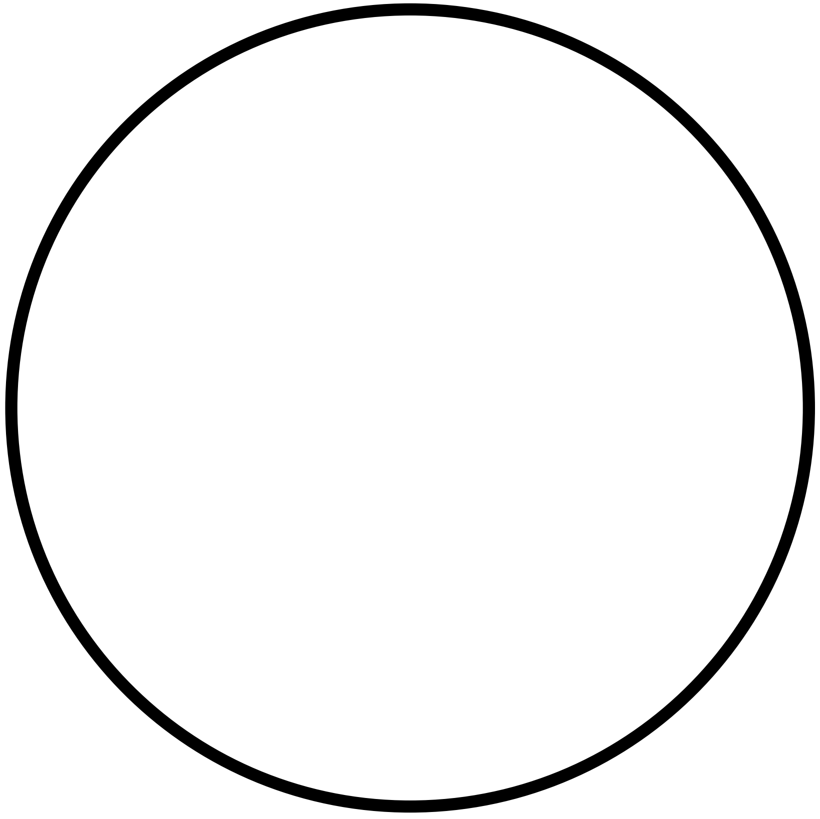
My ugly left hand

Time is finite

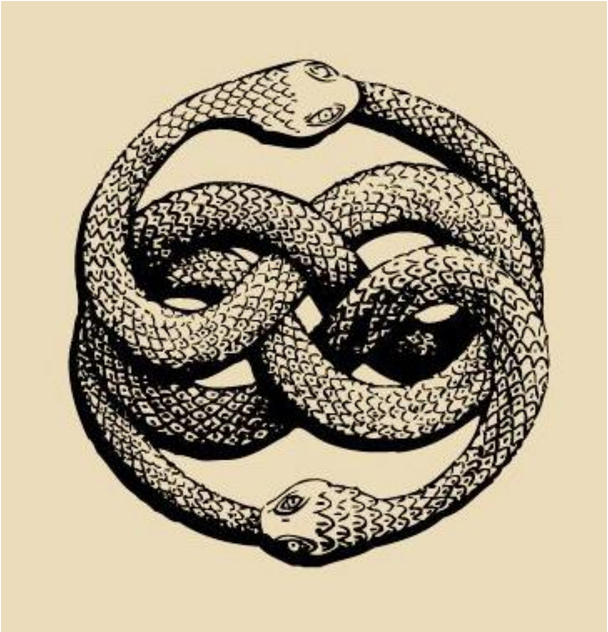


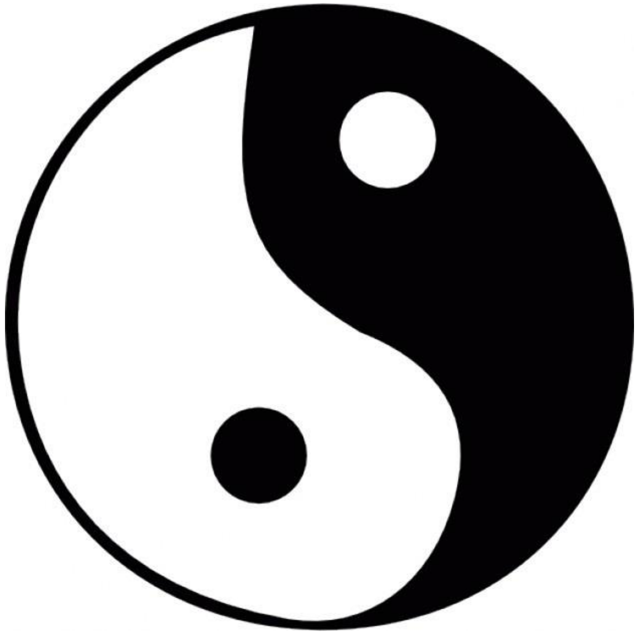
Our ugly Universe

Time is Circular



Time is Circular

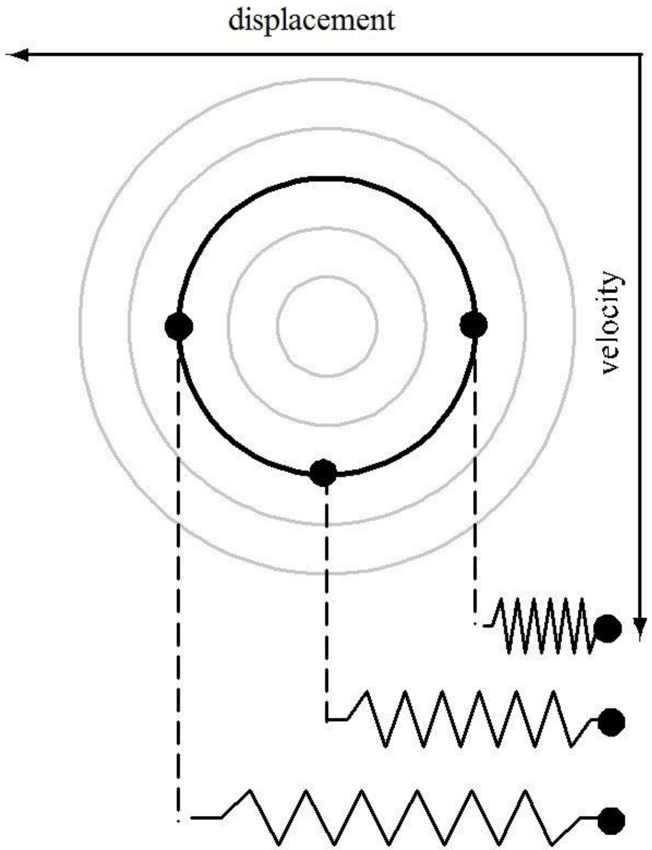




“Zen”

But there is a direction!

A Simple Spring



Fourier Series

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{2\pi nt}{T}}$$

Any function is a sum of ever oscillating ideal springs

Finnegans Wake 1939 (17 Years)

James Joyce, 1882 – 1941.

“If you met on the binge a poor acheseyeld from Ailing..."

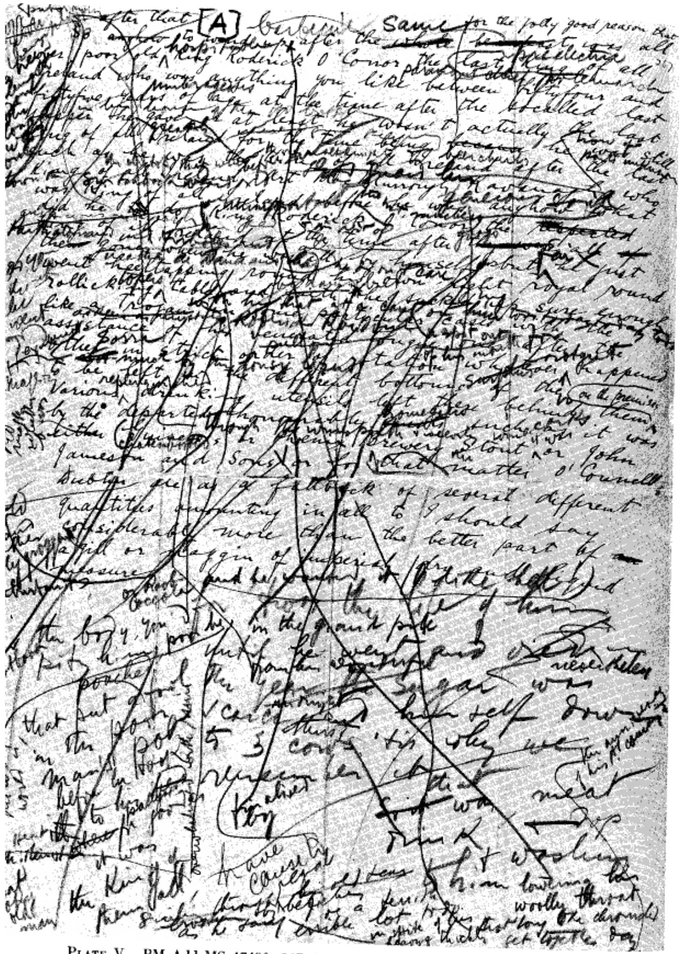


Plate V. BM Add MS 47480, 267, 267 b. The earliest available version of the “Roderick O'Conor” piece (FW 380–382), the first piece written for “Work in Progress.” This was among the last passages to be incorporated in the book. Joyce's

Finnegans Wake

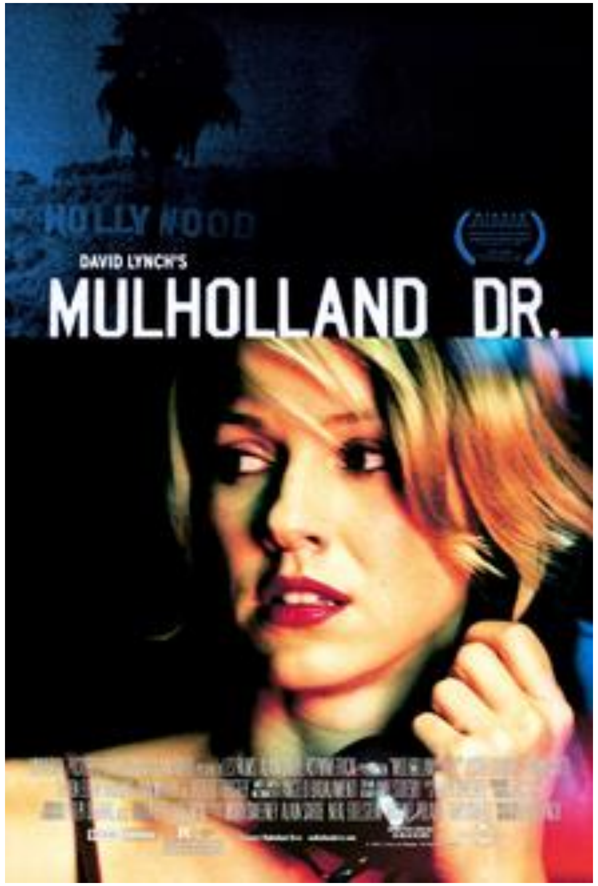
Riverrun, past Eve and Adam's, from swerve of shore to bend of bay,
brings us by a commodius vicus of recirculation back to Howth Castle and Environs.

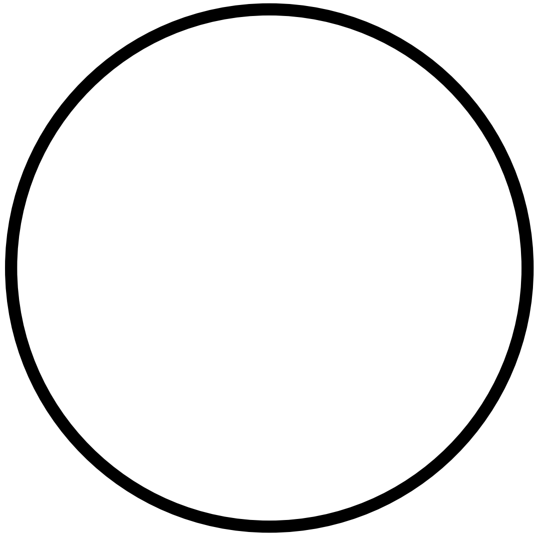
Lots of dense prose and finally:

A way a lone a last a loved a long the

A way a lone a last a loved a long the / riverrun, past Eve and Adam's, from swerve of shore to bend of bay, brings us by a commodius vicus of recirculation back to Howth Castle and Environs.

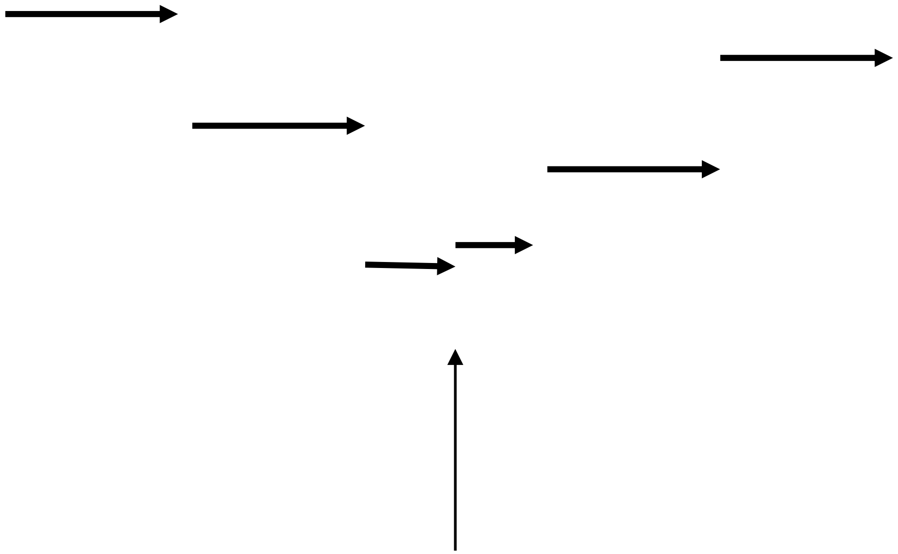
Mulholland Dr. (2001)





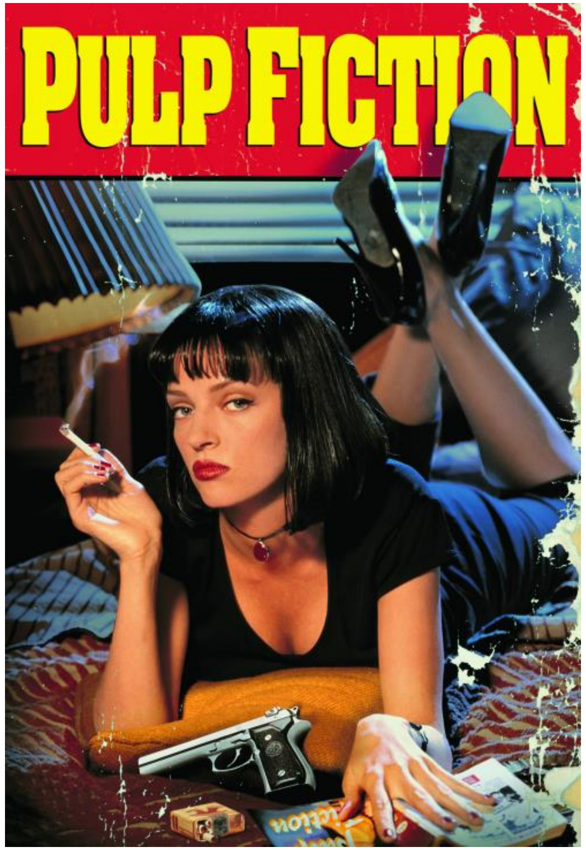
No End of the movie

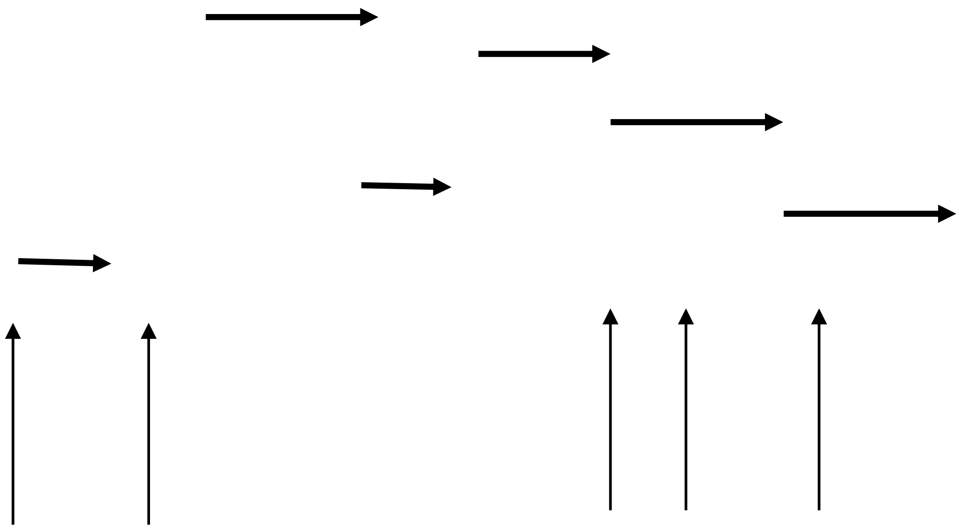
Memento (2000)



End of the movie

Pulp Fiction (1994)





End of the movie?

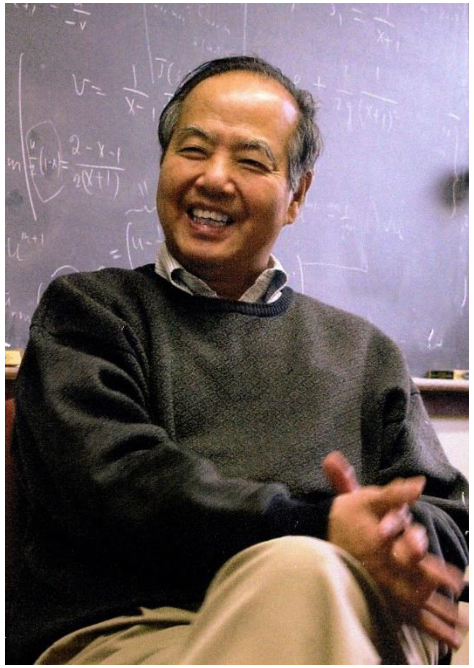
Fundamental Physics: Time has no Direction

Laws are invariant under the transformation

Time → *minus* **Time**

How do we deal with this?

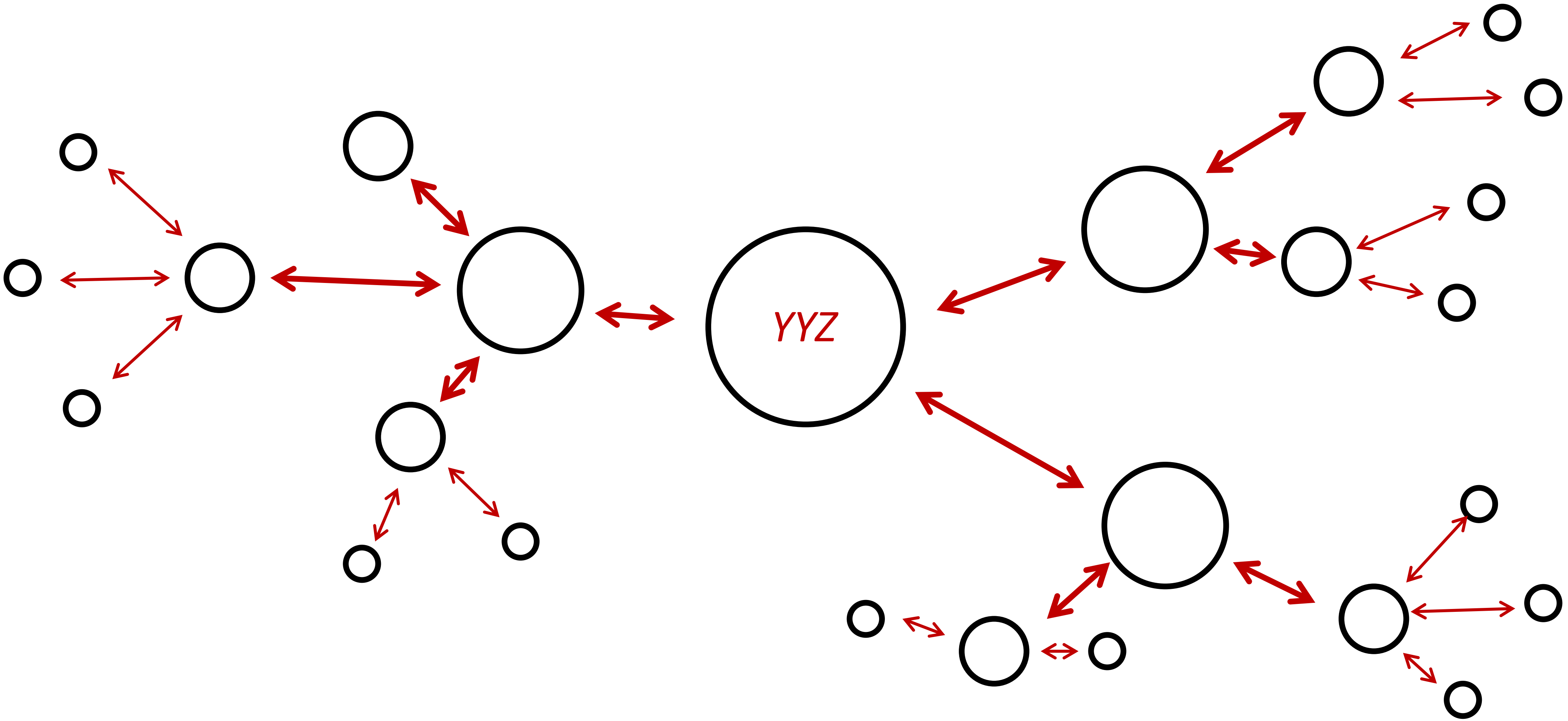
How do we deal with this?



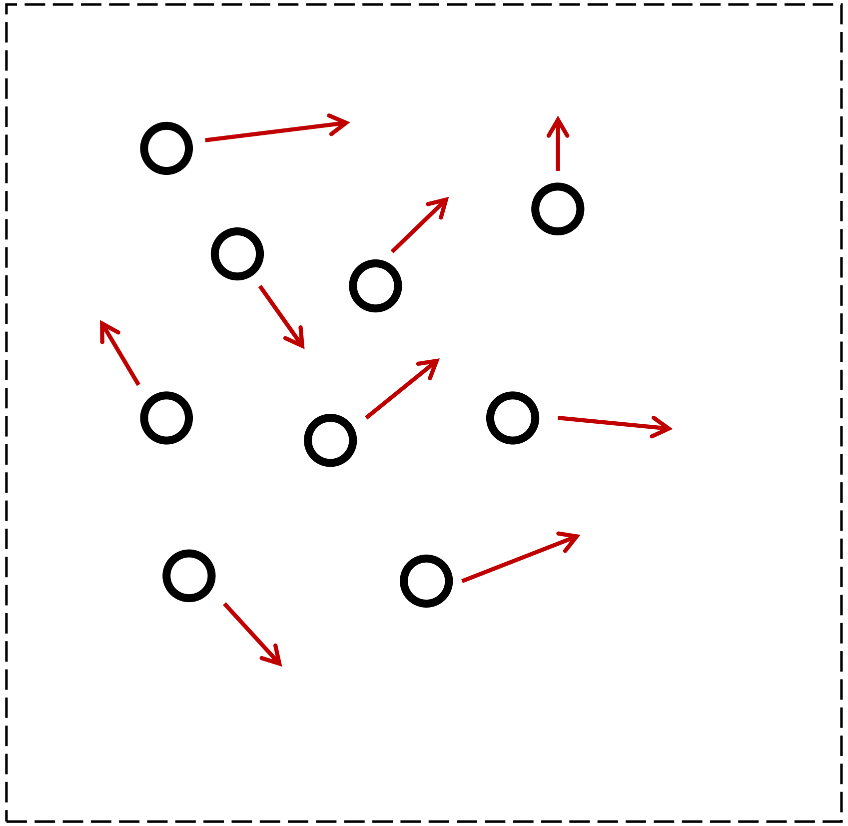
But suppose that in every airport we were to remove all the signs and flight information, while maintaining exactly the same number of flights, as shown in Figure 11(b). A person starting from Easthampton would still arrive in New York, since that is the only airport connected to Easthampton. However, without the signs to guide him, it would be very difficult for him to pick out the return flight to Easthampton from the many gates in the New York airport. The plane he gets on may be headed for San Francisco. If, in San Francisco, he then tries another plane again without any guidance, he could perhaps arrive in Tokyo. If he keeps on going this way, his chance of getting back to Easthampton is very slim indeed. In this example, we see that microscopic reversibility is strictly maintained. When all the airport destination signs and other flight information are given clearly, then macroscopically we also have reversibility. On the other hand, if all such information is withheld, then the whole macroscopic process appears irreversible. Thus, macroscopic irreversibility is not in conflict with microscopic reversibility.

Tsung-Dao (T. D.) Lee 李政道

How do we deal with this?

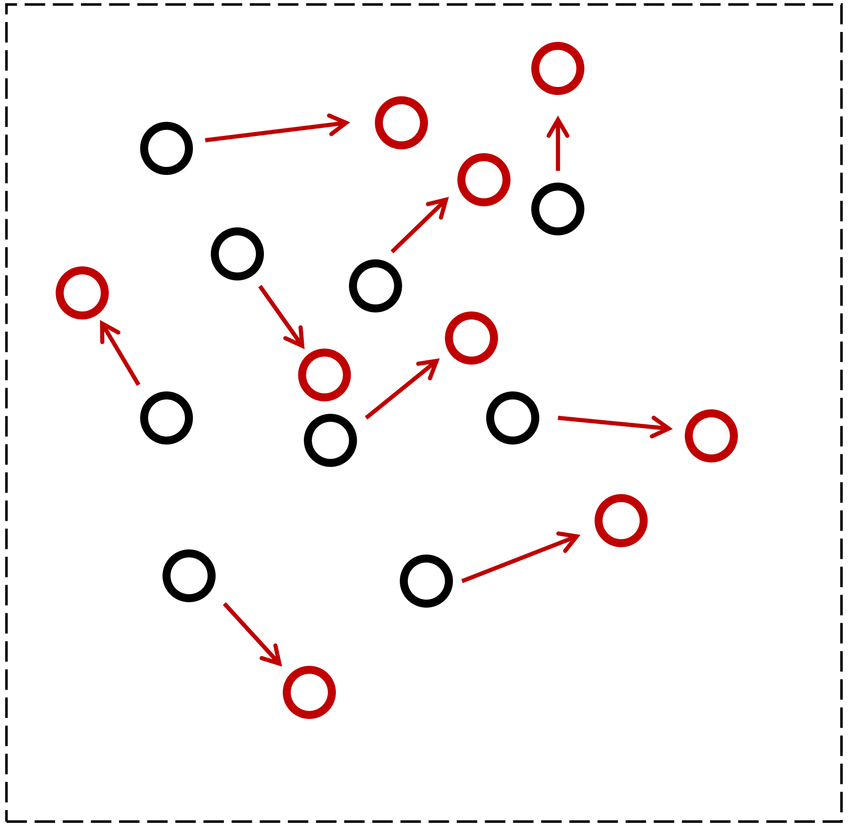


*t*ime is Like Temperature



T = sum of  squared

*t*ime is an emergent phenomenon

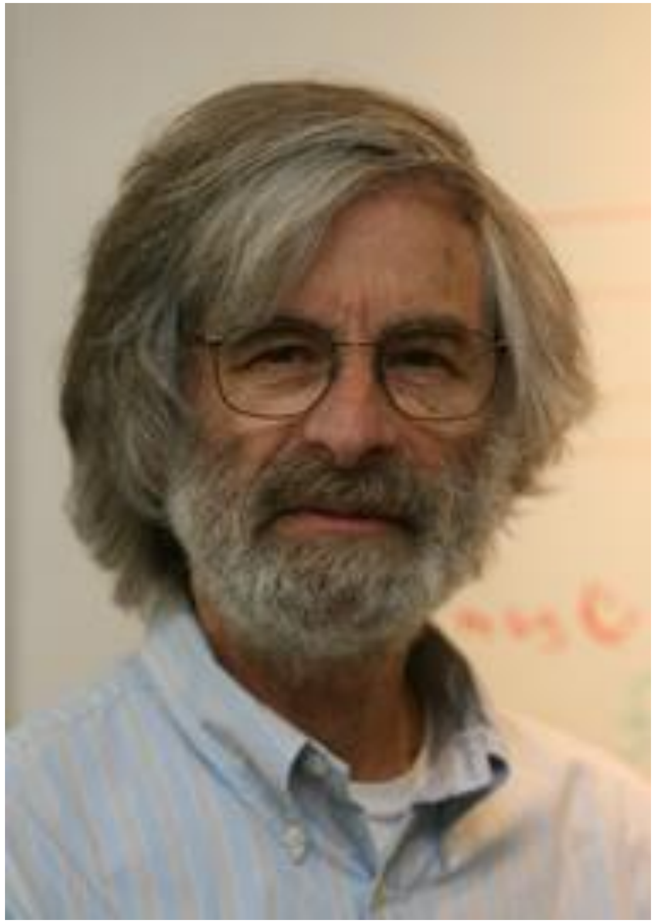


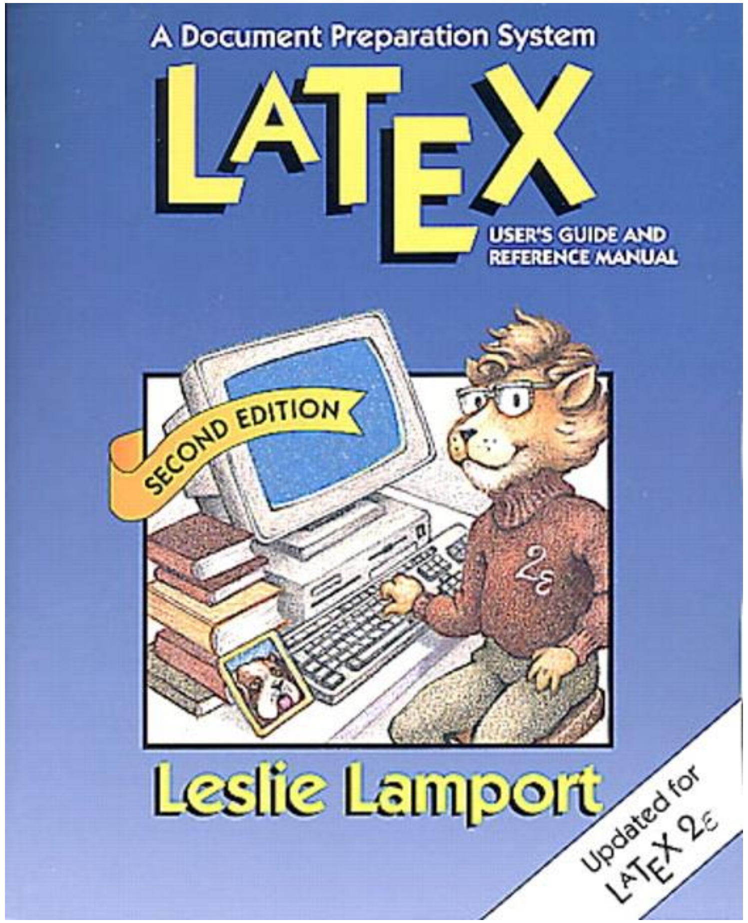
d*t* = sqrt(sum of  squared / energy)

Time ***emerges*** *from changes in* ***positions****!*

Nucleus & Position-Based Dynamics

Logical Time





Operating Systems — R. Stockton Gaines, Editor

Time, Clocks, and the Ordering of Events in a Distributed System

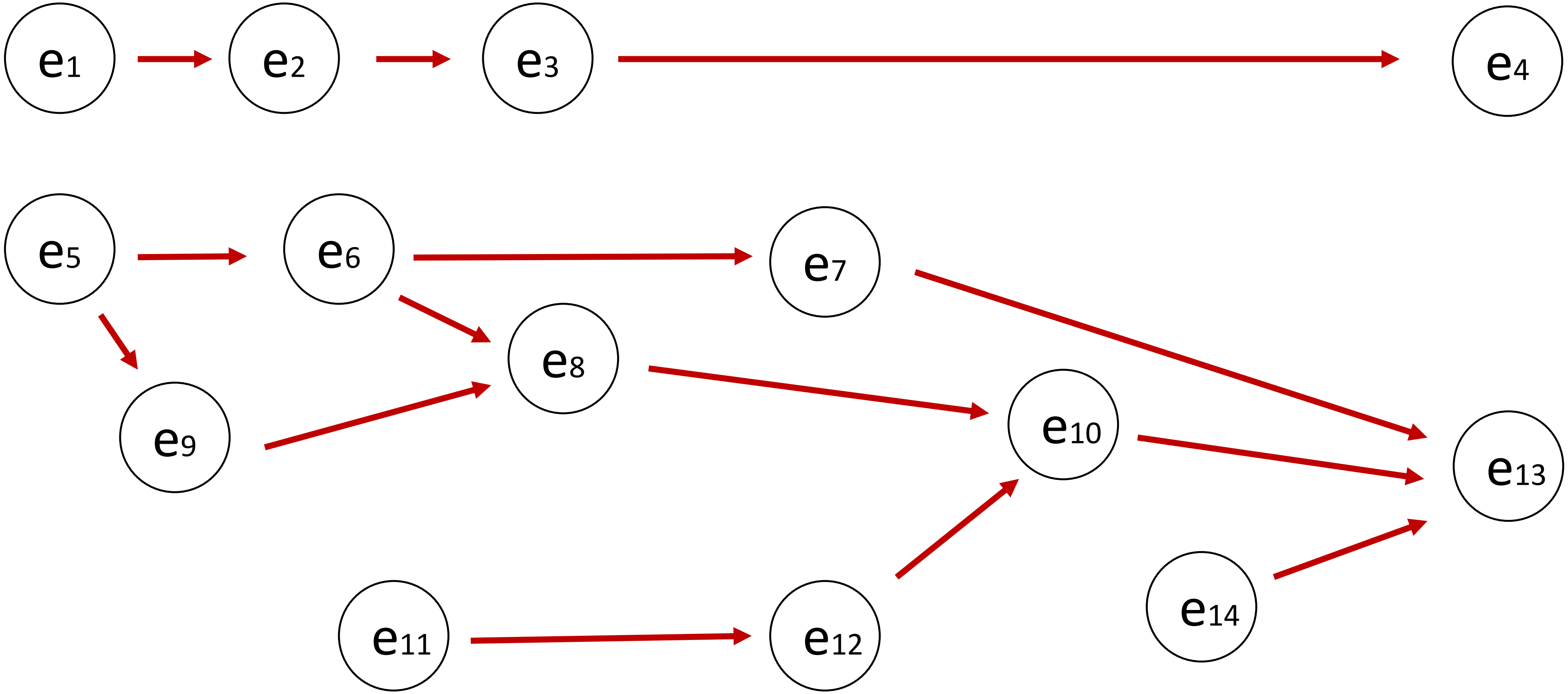
Leslie Lamport
Massachusetts Computer Associates, Inc.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

CR Categories: 4.32, 5.29

Logical Time



Dessert

A Taste of Barbados in February

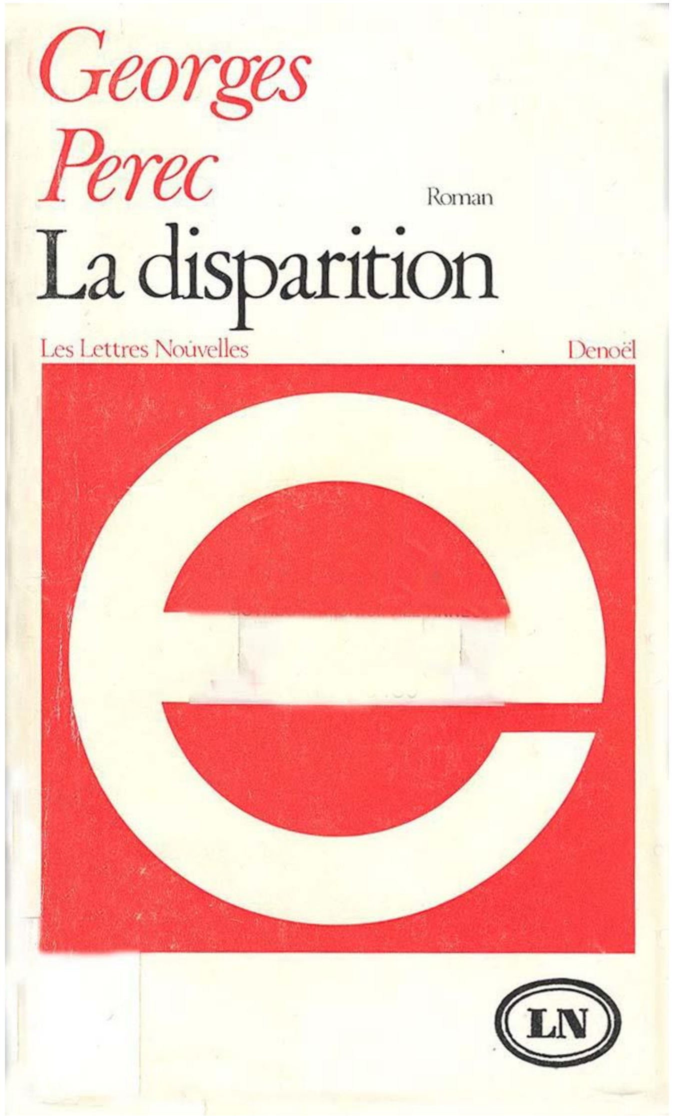
Thanks Paul Kry!

Doing Math *without* Math

Writing Optimization Code *Without* Continuous Math

David Hilbert

"A mathematical theory is not to be considered complete
until you have made it so clear
that you can explain it to the first man
whom you meet on the street."


Georges
Perec
Roman
La disparition
Les Lettres Nouvelles
Denoël
LN


2012
PLAC
G ORG S
P R C
CRIVAIN FRANÇAIS 1936 - 1982

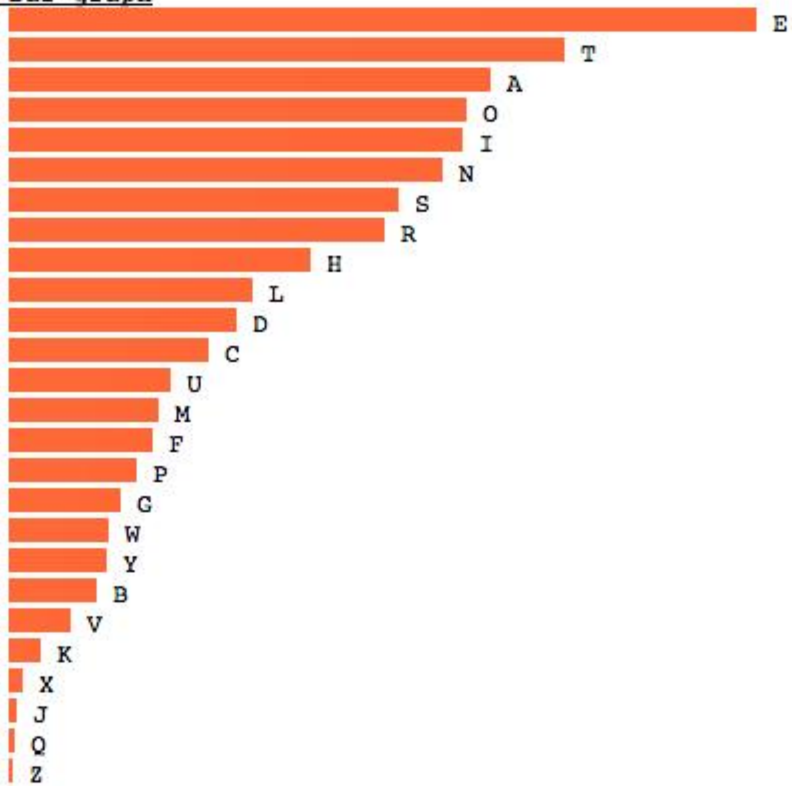

Georges PEREC
A Void

Ou alors, on pourrait agir ainsi : tu irais à un gala nippon.

Il y aurait pour ton grand plaisir, car on sait ton goût pour l'art subtil du Go, un naïf affrontant dans un match amical un champion, un « Kan Shu », sinon un « Kudan » : Kaku Takagawa, mais disposant, pour adoucir la disproportion, d'un fort handicap, non d'un « furin » mais d'un « Naka yotsu ». Kaku Takagawa ouvrirait par un « Moku hadzushi » ; son opposant s'absorbant dans un « Ji dori Go » aussi maladroit 301

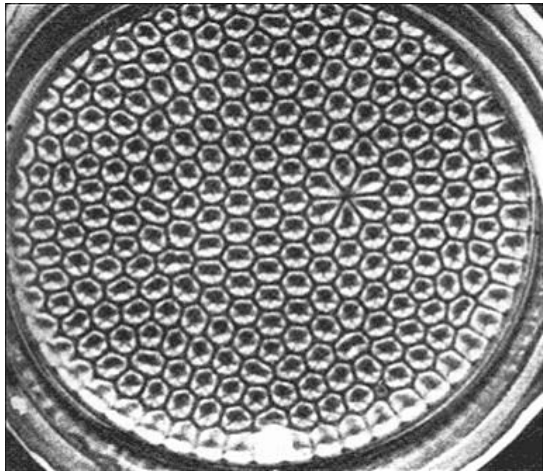
LET	COUNT	PERCENT
E	445.2 B	12.49%
T	330.5 B	9.28%
A	286.5 B	8.04%
O	272.3 B	7.64%
I	269.7 B	7.57%
N	257.8 B	7.23%
S	232.1 B	6.51%
R	223.8 B	6.28%
H	180.1 B	5.05%
L	145.0 B	4.07%
D	136.0 B	3.82%
C	119.2 B	3.34%
U	97.3 B	2.73%
M	89.5 B	2.51%
F	85.6 B	2.40%
P	76.1 B	2.14%
G	66.6 B	1.87%
W	59.7 B	1.68%
Y	59.3 B	1.66%
B	52.9 B	1.48%
V	37.5 B	1.05%
K	19.3 B	0.54%
X	8.4 B	0.23%
J	5.7 B	0.16%
Q	4.3 B	0.12%
Z	3.2 B	0.09%

bar graph

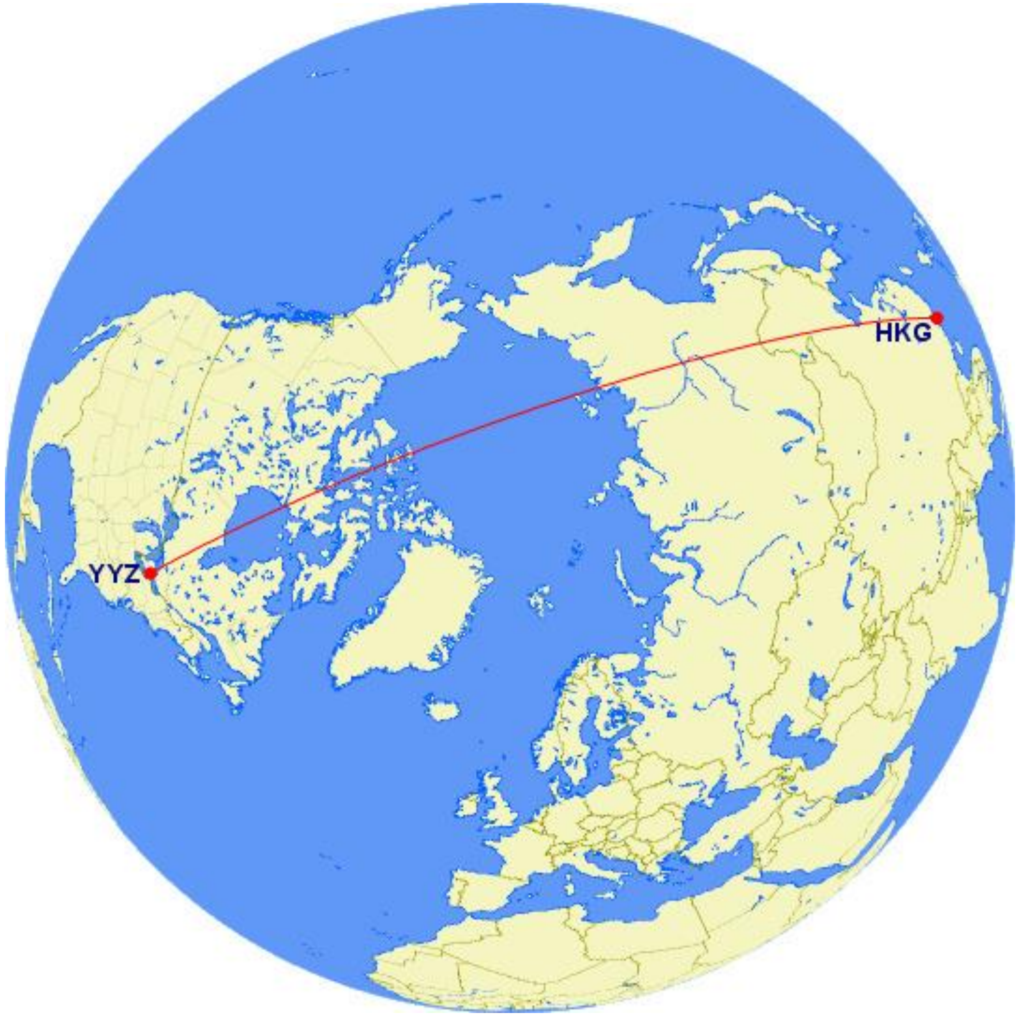


Examples of optimization

In Nature it occurs naturally

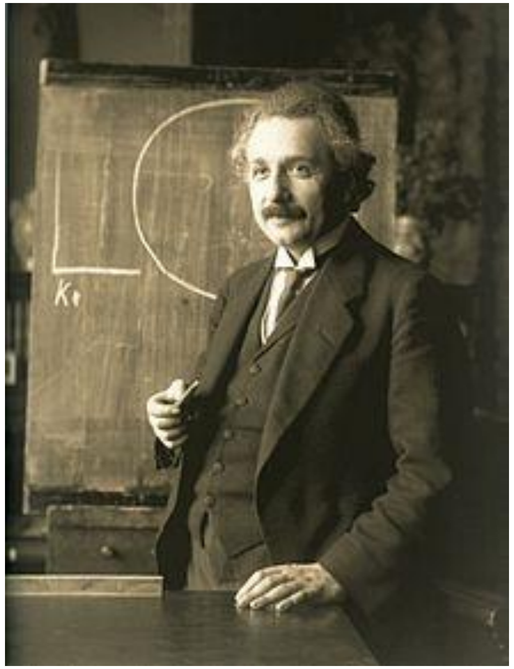


Statics: Geodesics



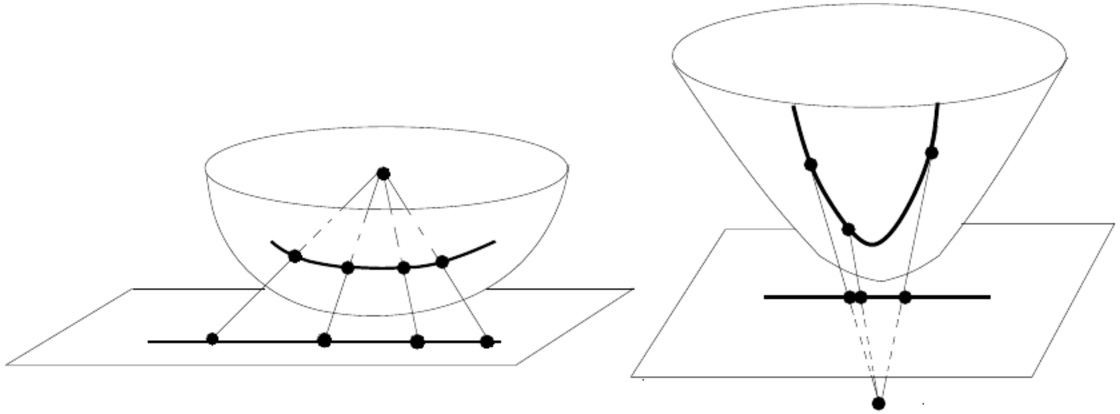
15 and a half hour flight

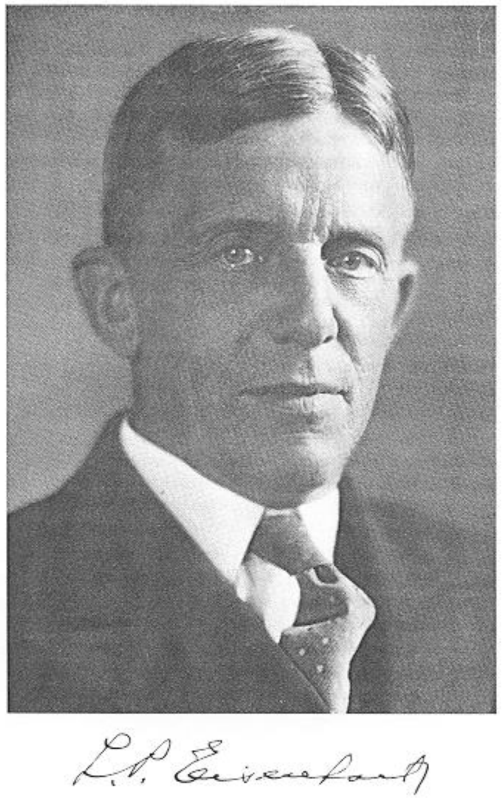
Dynamics: Alembert, Hertz and Einstein



Dynamics is a geodesic in space-time

Eisenhart metric



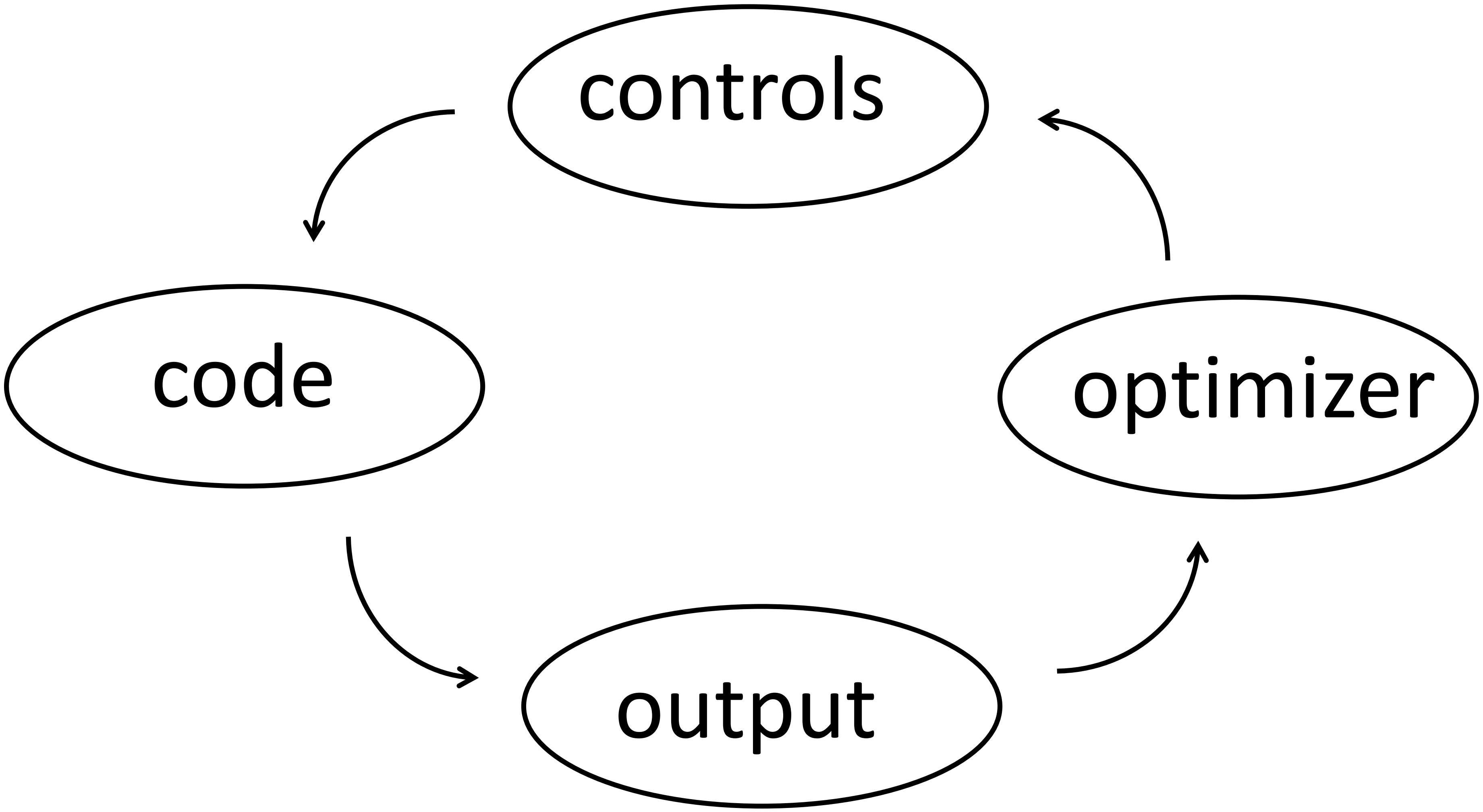


Dynamics is a geodesic in space-time + one extra dimension

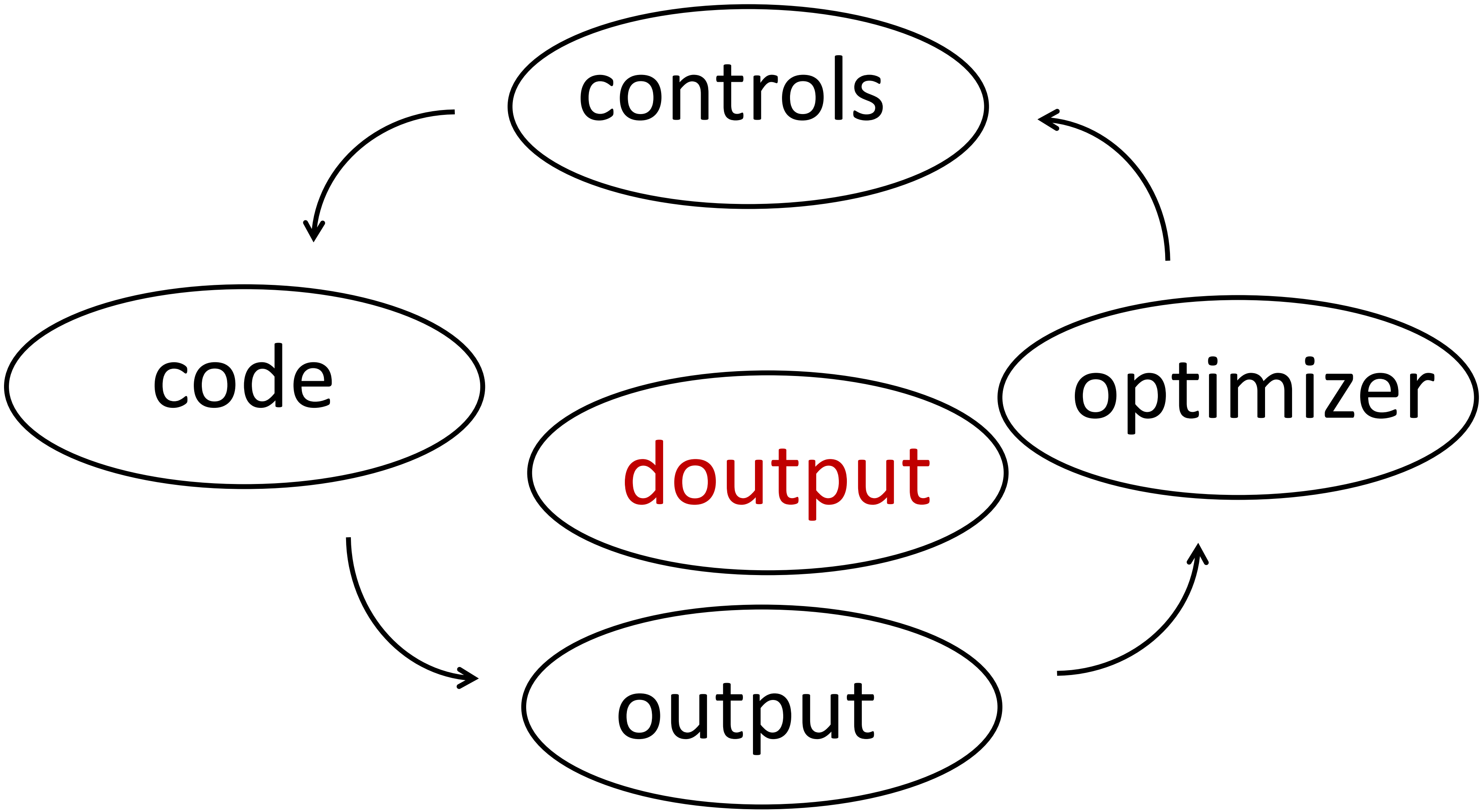
Concept of *lifting*.

Solve a problem in a higher dimensional space where there is more freedom and then project back.

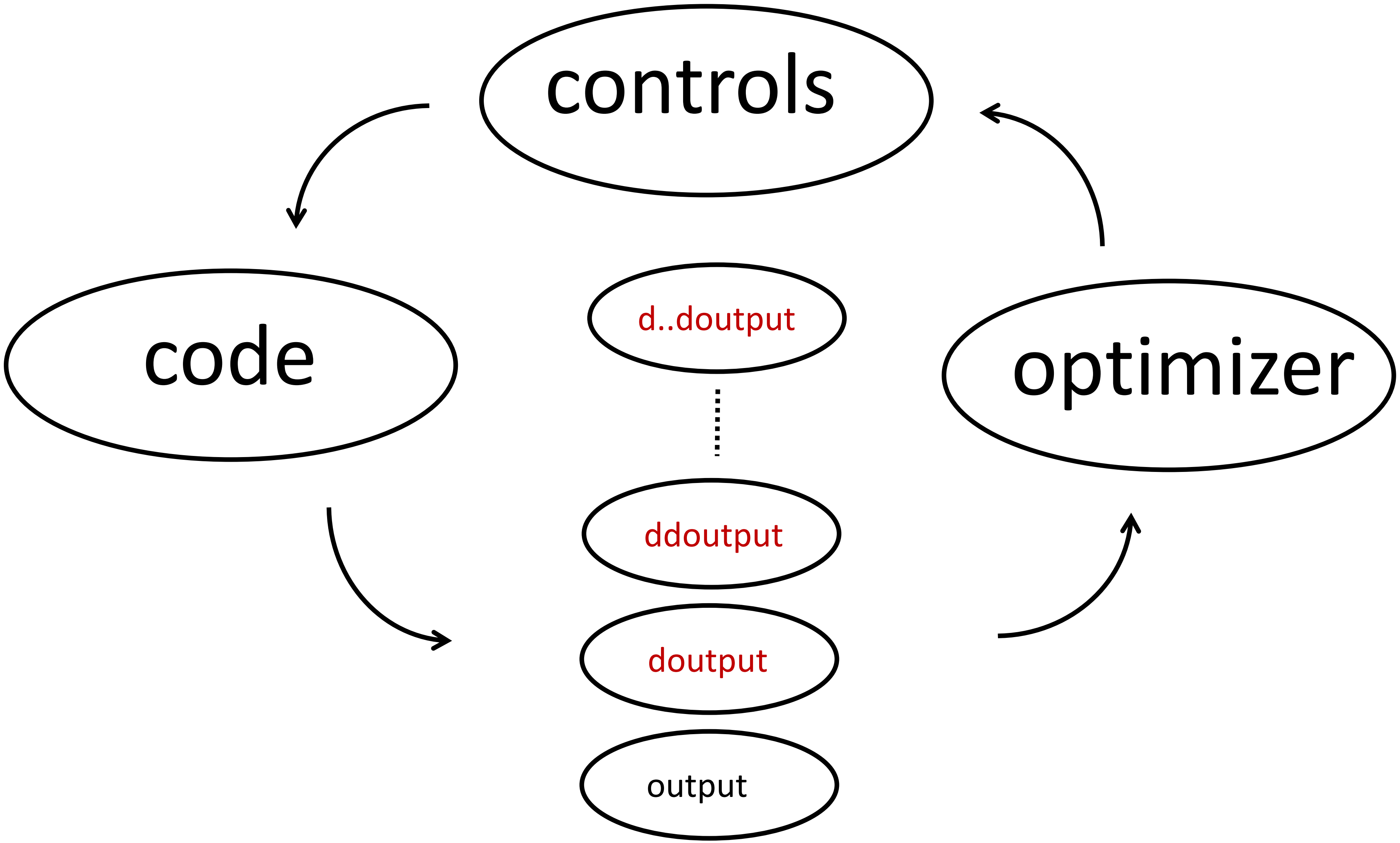
optimization



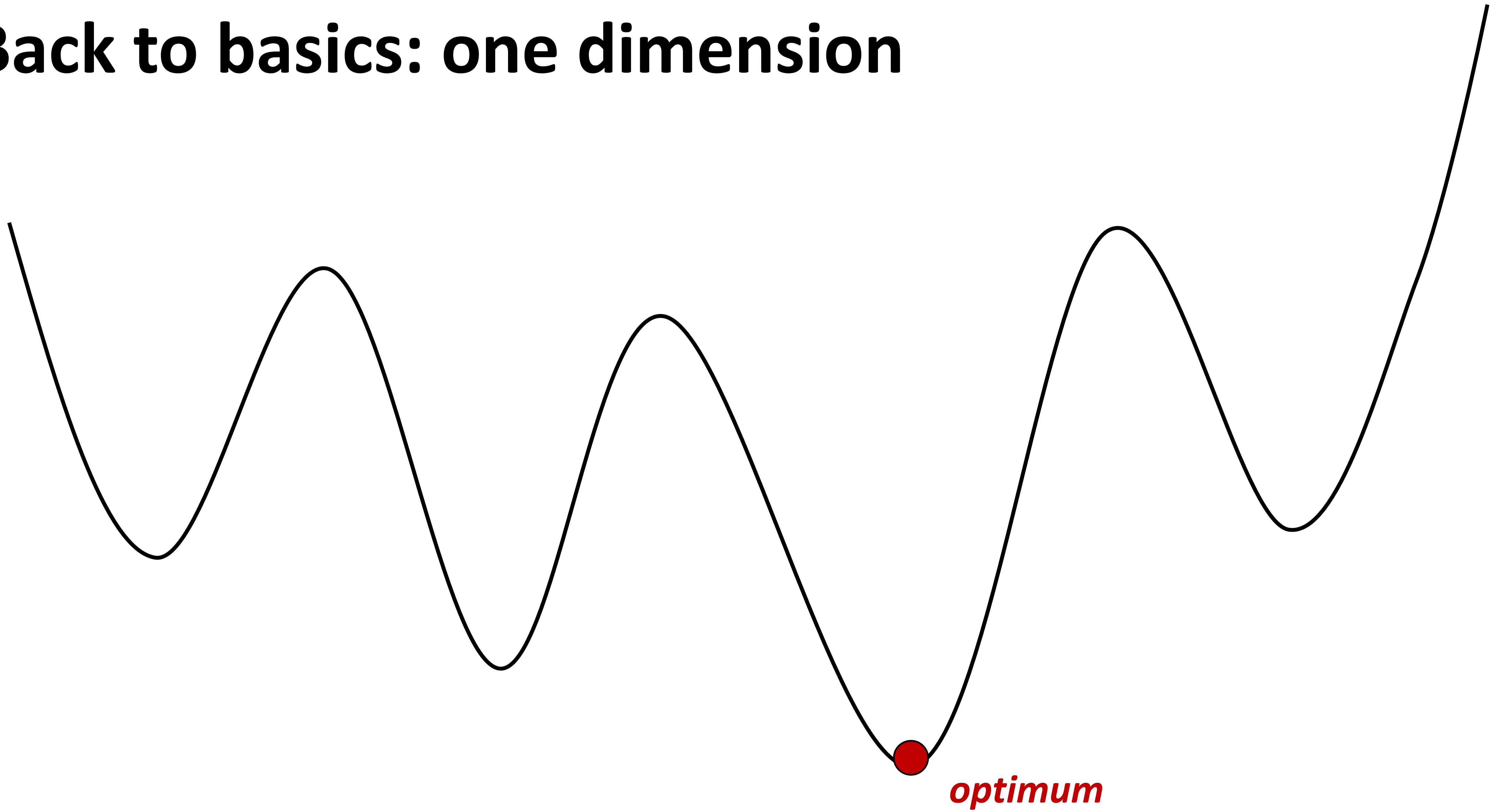
optimization



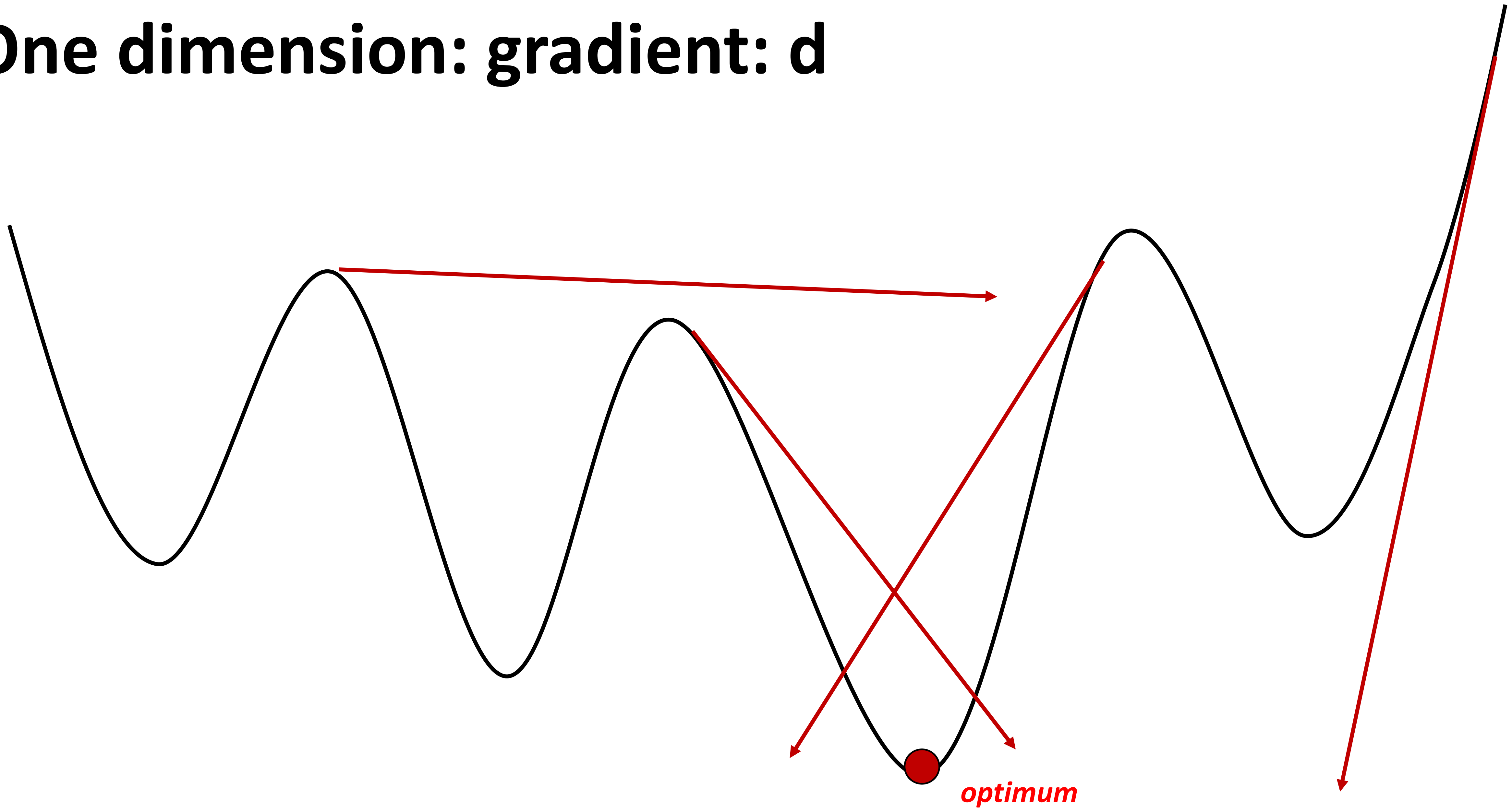
optimization



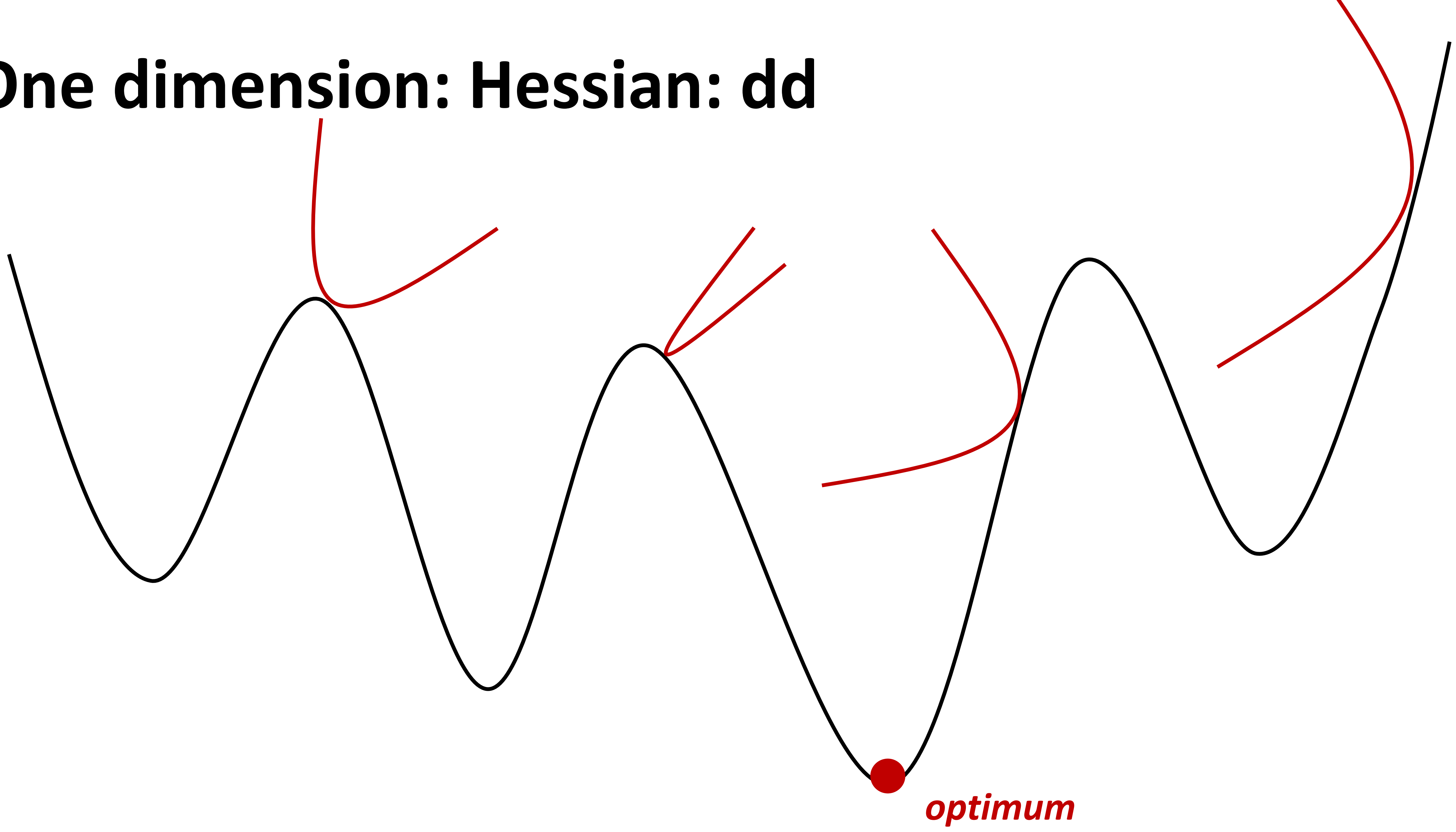
Back to basics: one dimension



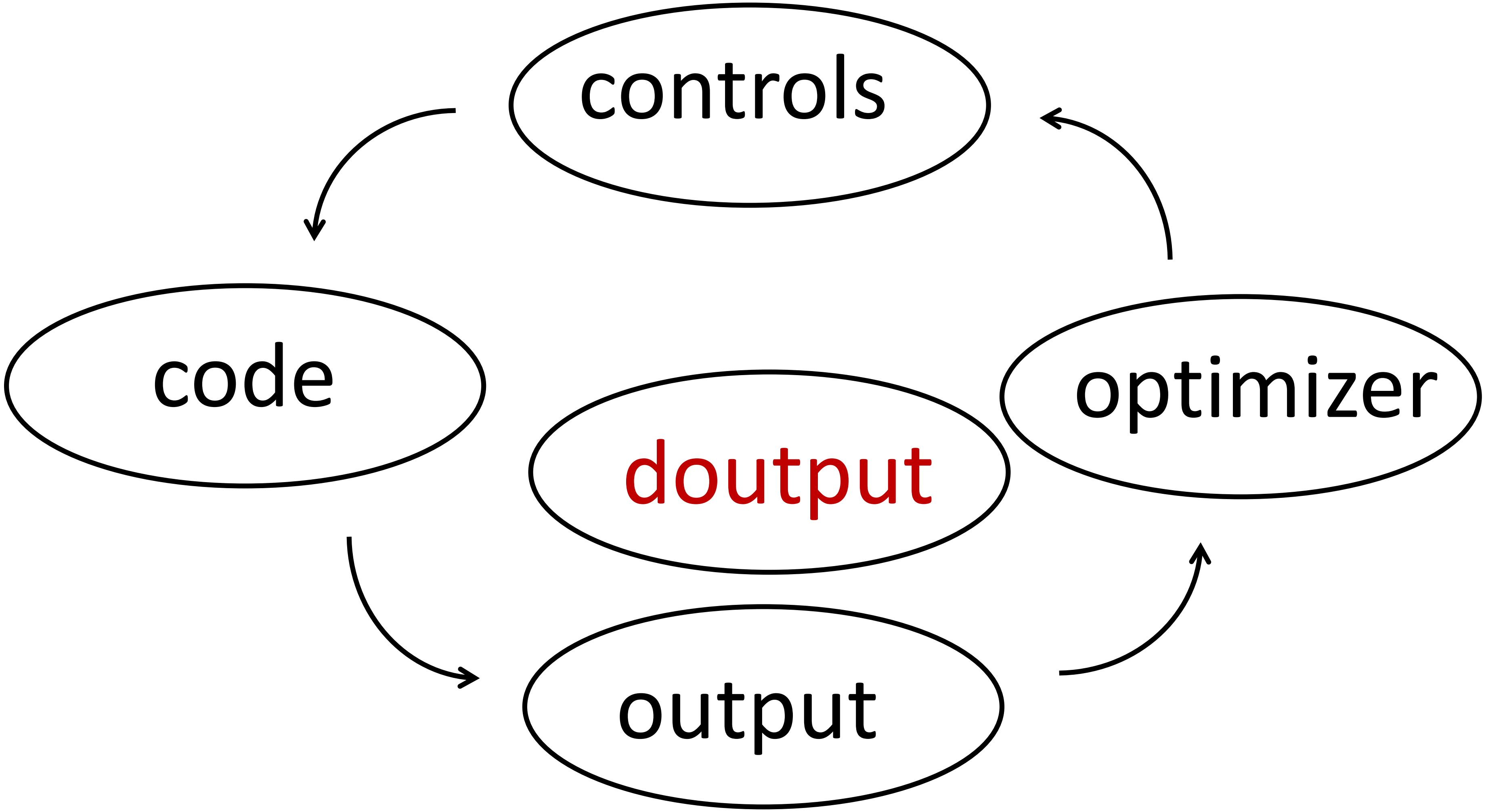
One dimension: gradient: d



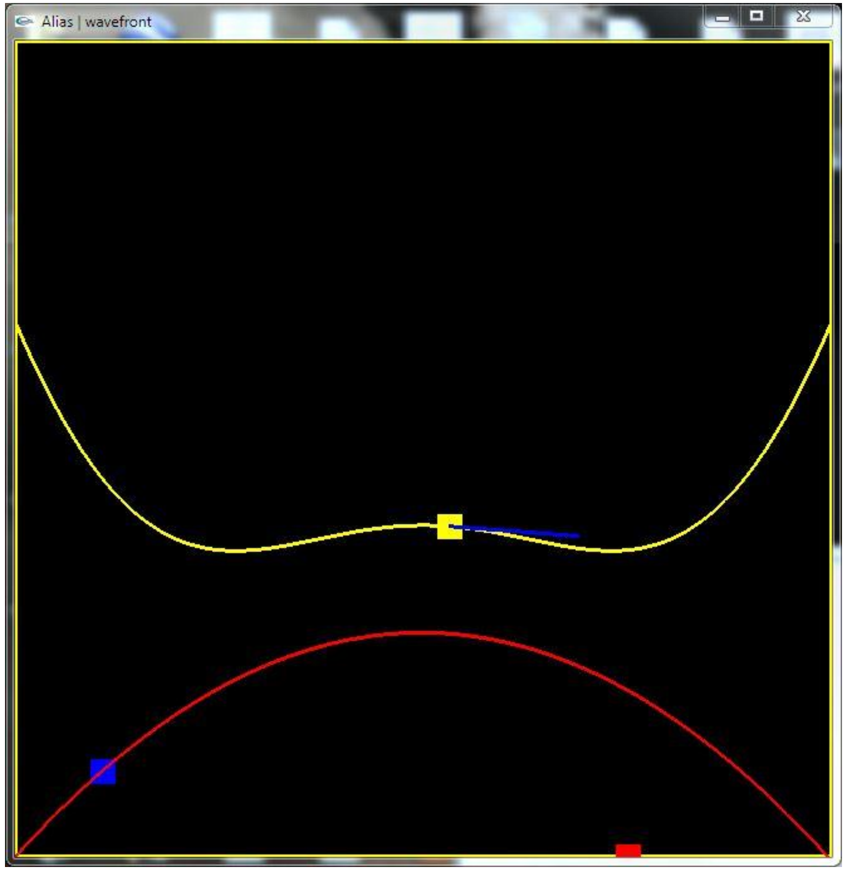
One dimension: Hessian: dd



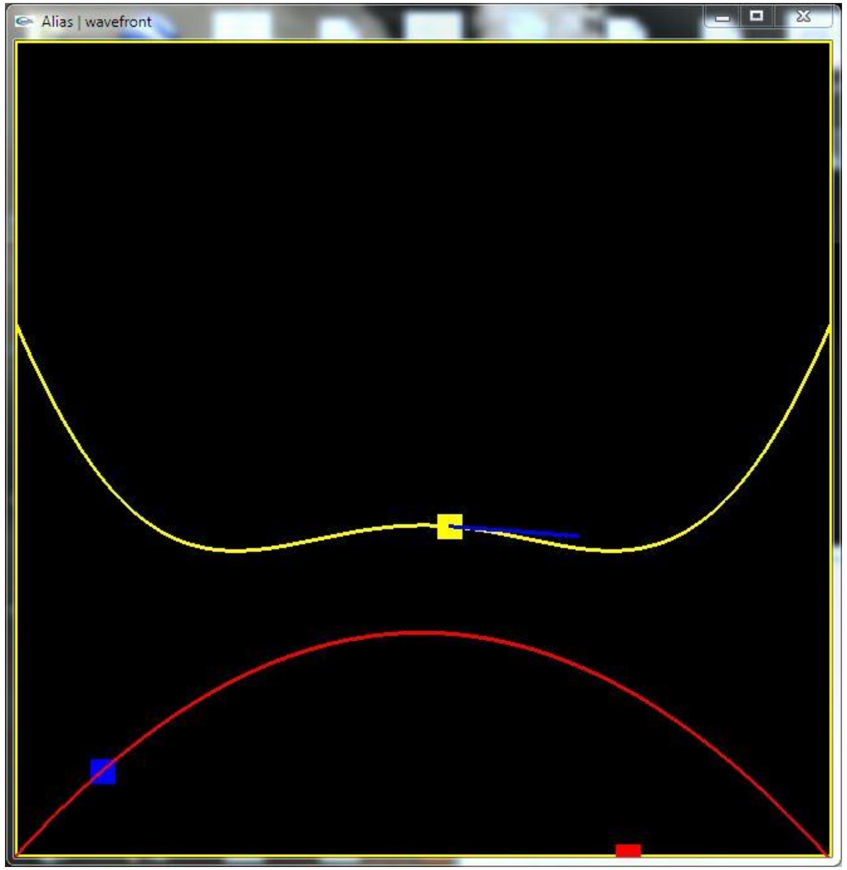
Optimization: *let stick to differential*



Optimization: *cannon balls in 2D*

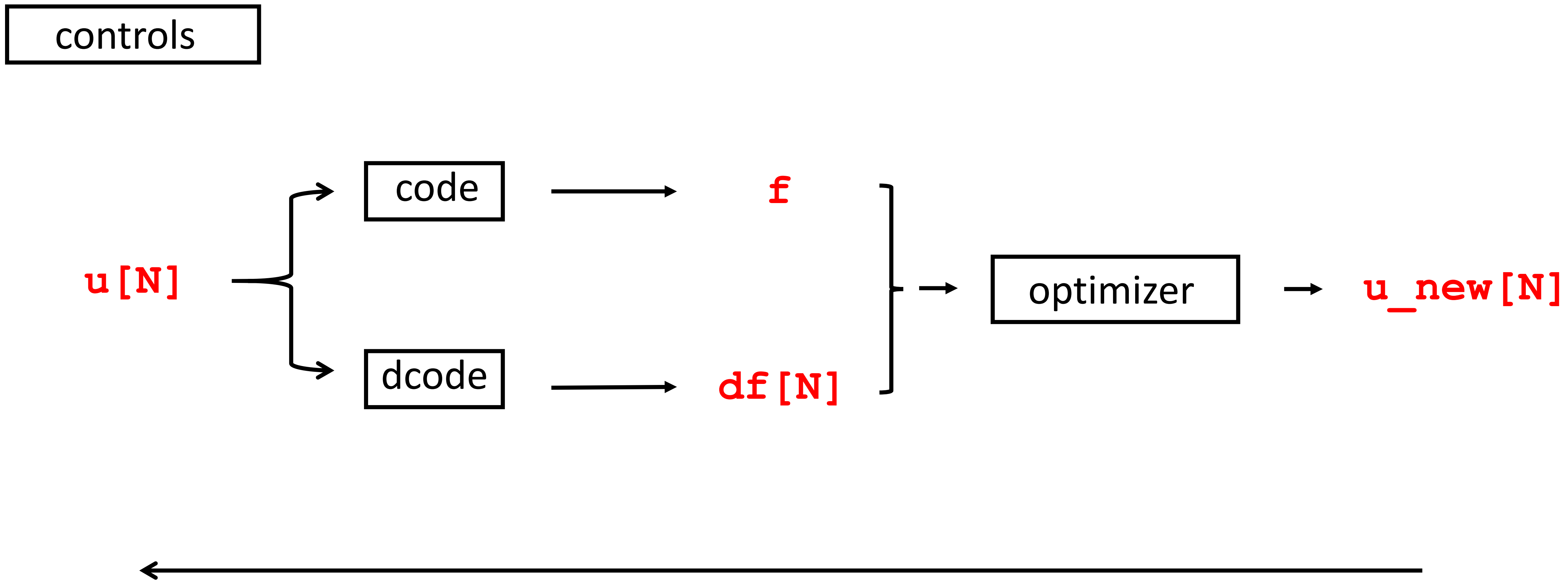


fast



slow

Gradient Based Optimizer



We Need a Discrete Gradient!

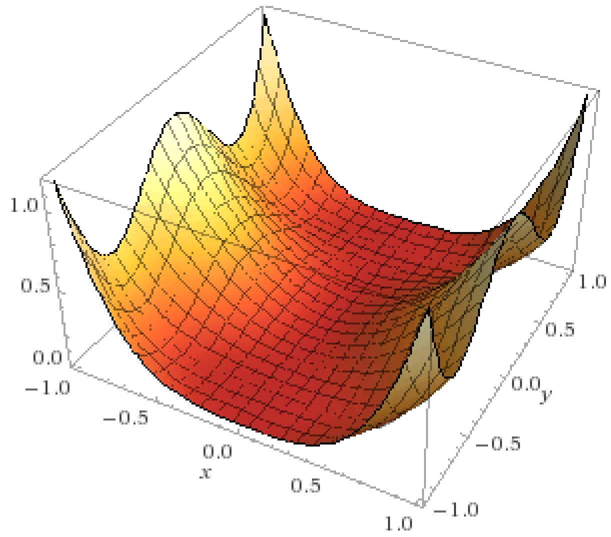
Usually: finite differences, finite elements, etc.

Instead:

Differentiate code *at the* code level

Example

```
float optim_func ( float u[2] )
{
        float s = u[0]*cosf(2*u[1]);
        float t = u[0]*u[0]*u[1];
        float f = s*s+t*t;
        return ( f );
}
```



Hyper-Numbers

$z = a + i\,b$

$i^2 = -1$ Complex numbers ("Awesome Numbers")

$i^2 = 1$ Hyperbolic numbers (Special Relativity)

$i^2 = 0$ **Dual numbers** (Automatic Differentiation)

Dual-Numbers

"Number theoretic phase space"

$z = f + i\, df \qquad i^2 = 0$

With these numbers you compute the function and the differential at the same time!

Good for Optimization!

Also works for higher differentials btw… Hyper-Dual Numbers…

Dual-Numbers encode Calculus

$$(f + i\,df) + (g + i\,dg) = (f + g) + i\,(df + dg)$$

$$(f + i\,df) \times (g + i\,dg) = (f \times g) + i\,(df \times g + f \times dg) \qquad i^2 = 0$$

Etc.

Clifford, W. K., "***Preliminary Sketch of Biquaternions***," Proc. London Math. Soc., Vol. s1-4, No. 1, *1871*, pp. 381–395.

Automatic Differentiation: AD.h

```
template <int N> class dfloat
{
public:

    float v[N+1];

    // implementation
}
```

```
dfloat<2> u[2], x, f;
```

Automatic Differentiation: AD

```
template <int N> class dfloat
{
public:

    float v[N+1];

    dfloat (){
        for ( int i=0 ; i<=N ; i++ ) v[i] = 0.0f;
    }
    dfloat ( float s ){
        v[0] = s;
        for ( int i=1 ; i<=N ; i++ ) v[i] = 0.0f;
    }
}
```

```
dfloat<2> u[2], x(0.05f), f(3.14f);
```

Automatic Differentiation: AD

```
template <int N> class dfloat
{
public:

    float v[N+1];

    dfloat & operator = ( const dfloat & a ){
        for ( int i=0 ; i<=N ; i++ ) v[i] = a.v[i];
        return ( *this );
    }
    dfloat & operator = ( const float s ){
        v[0] = s;
        for ( int i=1 ; i<=N ; i++ ) v[i] = 0.0f;
        return ( *this );
    }
    void val ( int i, float s ){
        v[i] = s;
    }

}
```

```
dfloat<2> u, x, f(0.0001f);

u[0].val(1,1.0f);
u[1].val(2,1.0f);
x = 2.03f;
f = x;
```

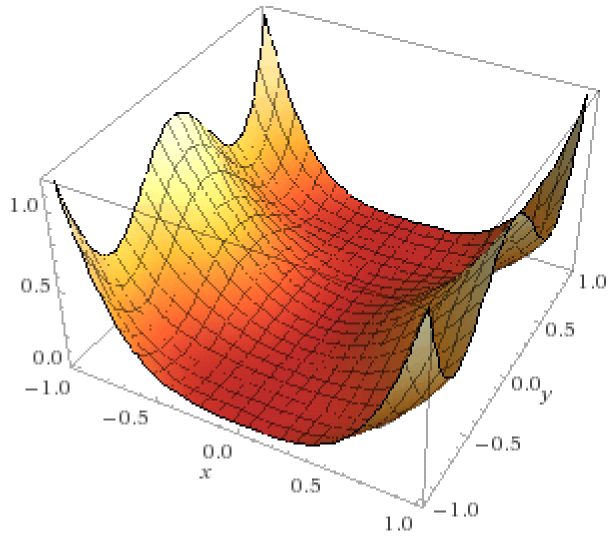
Automatic Differentiation: AD

```
template <int N> class dfloat
{
public:

     float v[N+1];

     friend dfloat operator + ( const dfloat a, const dfloat b ){
          dfloat c;
          for ( int i=0 ; i<=N ; i++ ) c.v[i] = a.v[i] + b.v[i];
          return ( c );
     }
     friend dfloat operator - ( const dfloat a, const dfloat b ){
          dfloat c;
          for ( int i=0 ; i<=N ; i++ ) c.v[i] = a.v[i] - b.v[i];
          return ( c );
     }
     friend dfloat operator * ( const dfloat a, const dfloat b ){
          dfloat c;
          c.v[0] = a.v[0] * b.v[0];
          for ( int i=1 ; i<=N ; i++ ) c.v[i] = a.v[i]*b.v[0] + a.v[0]*b.v[i];
          return ( c );
     }
     friend dfloat operator / ( const dfloat a, const dfloat b ){
          dfloat c;
          c.v[0] = a.v[0] / b.v[0];
          float g = b.v[0]*b.v[0];
          for ( int i=1 ; i<=N ; i++ ) c.v[i] = (a.v[i]*b.v[0] - a.v[0]*b.v[i])/g;
          return ( c );
     }
}
```

```
dfloat<2> u, x(0.45f), f(0.0001f);

u[0].val(1,1.0f);
u[1].val(2,1.0f);
x = u*u + f;
f = x/u*f;
```

Example

```
static float optim_func ( float x, float y )
{
        float s = x * cosf(2*y);
        float t = x * x * y;
        float f = s*s+t*t;
        return ( f );
}
```



Example

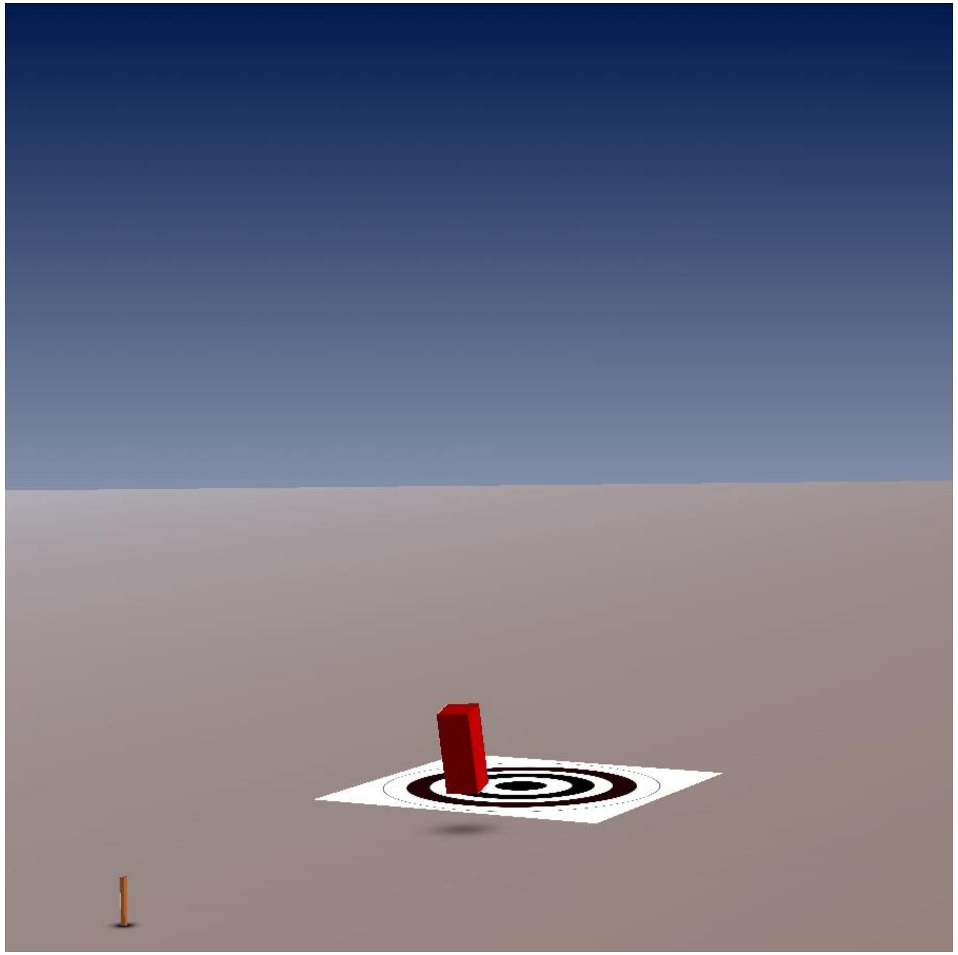
```
static float optim_func ( float x, float y )
{
        float s = x * cosf(2*y);
        float t = x * x * y;
        float f = s*s+t*t;
        return ( f );
}
```

```
static dfloat<2> doptim_func ( dfloat<2> x, dfloat<2> y )
{
        dfloat<2> s = x * dcos(2*y);
        dfloat<2> t = x * x * y;
        dfloat<2> f = s*s + t*t;

        return ( f );
}
```

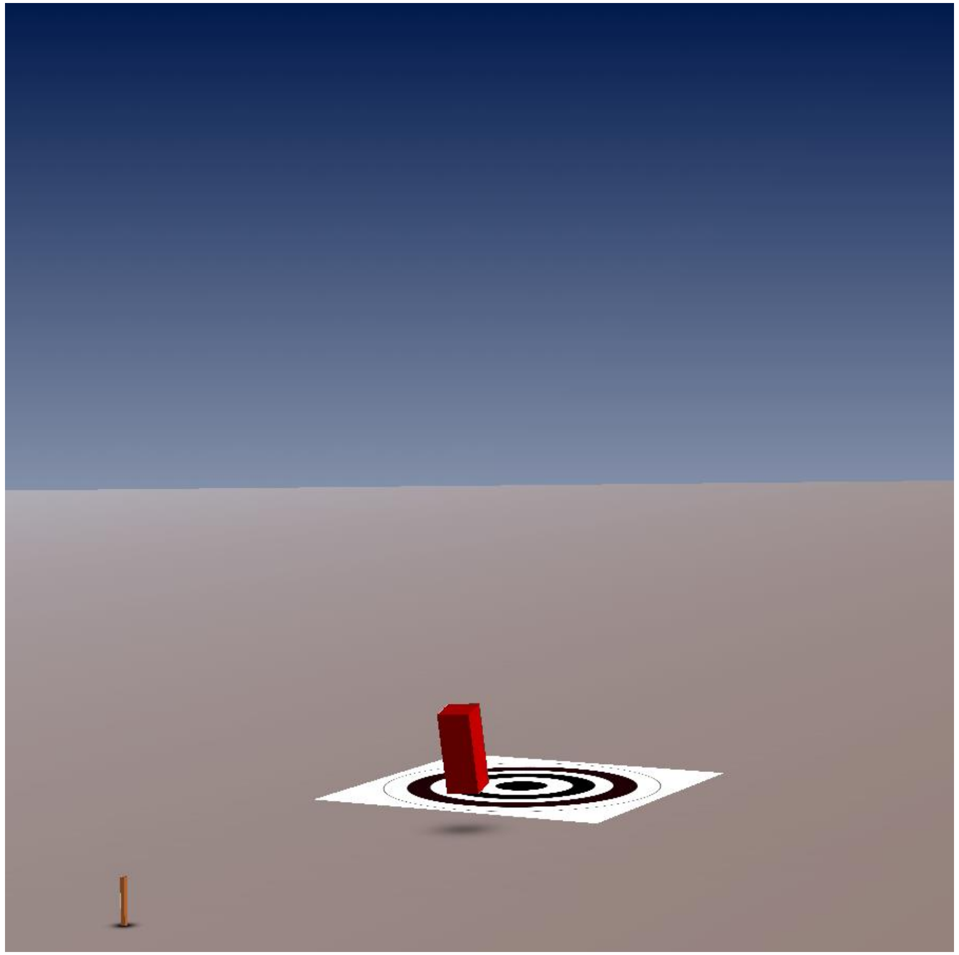
Example

```
static dfloat<2> doptim_func ( dfloat<2> x, dfloat<2> y )
{
        dfloat<2> s = x * dcos(2*y);
        dfloat<2> t = x * x * y;
        dfloat<2> f = s*s + t*t;

        return ( f );
}
```

Fun Example: Rigid Bodies

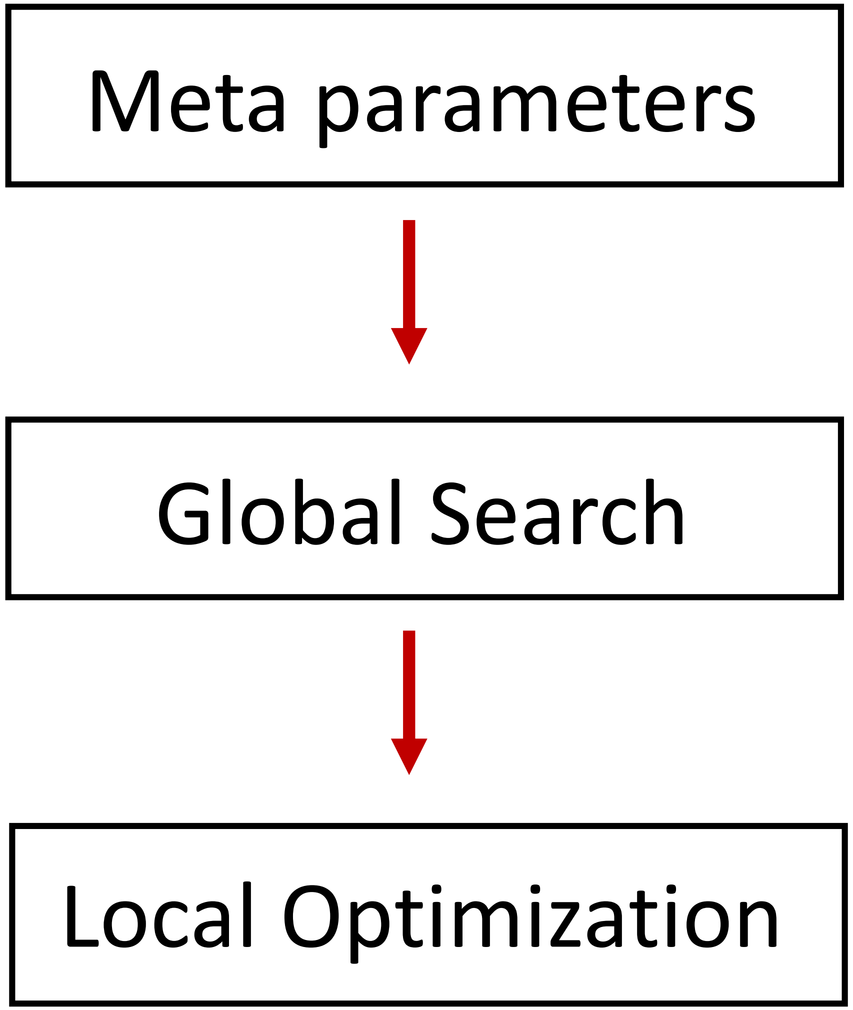


fast



slow

Three-Level Optimization



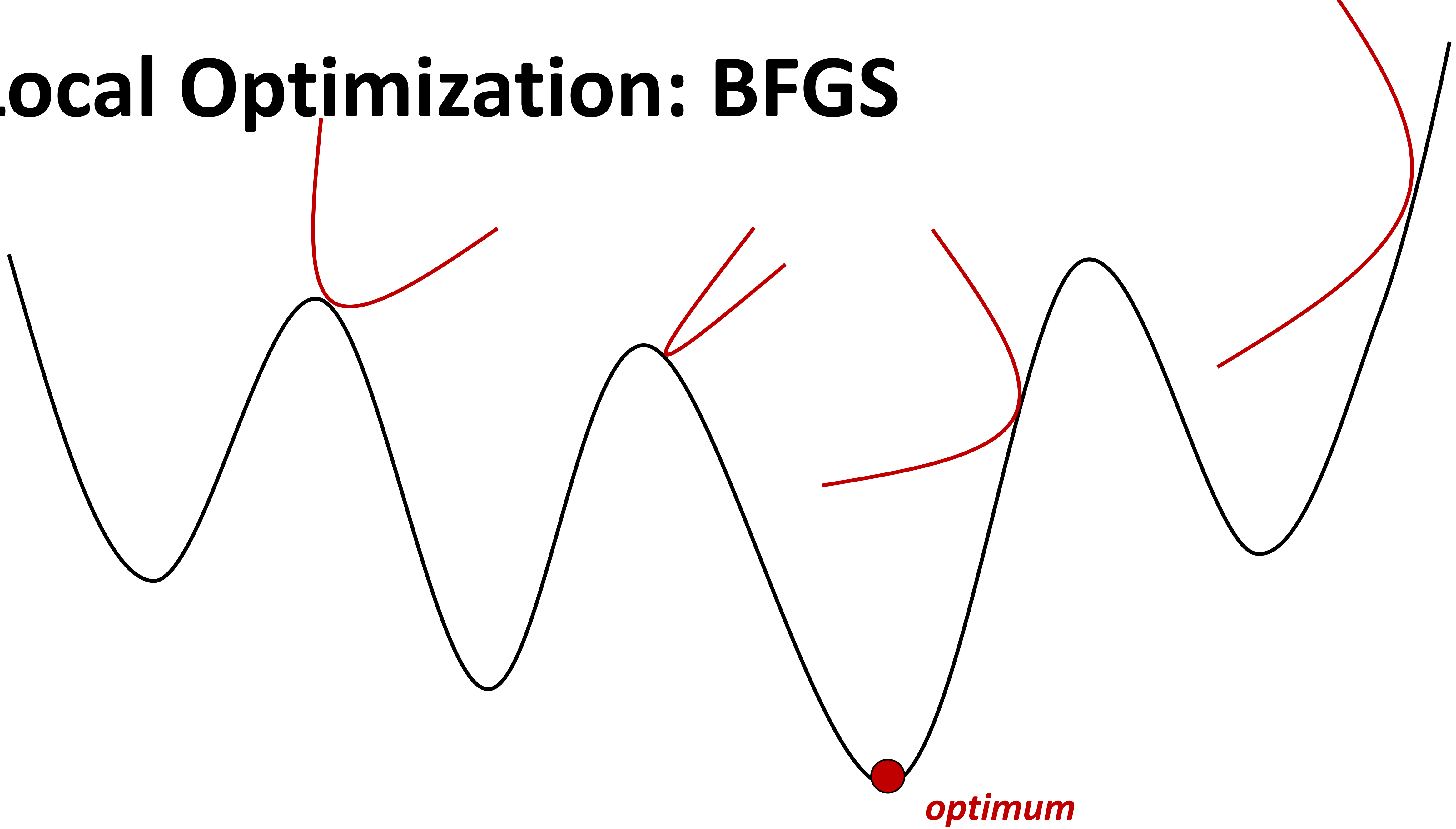
Three-Level Optimization

信 达 雅

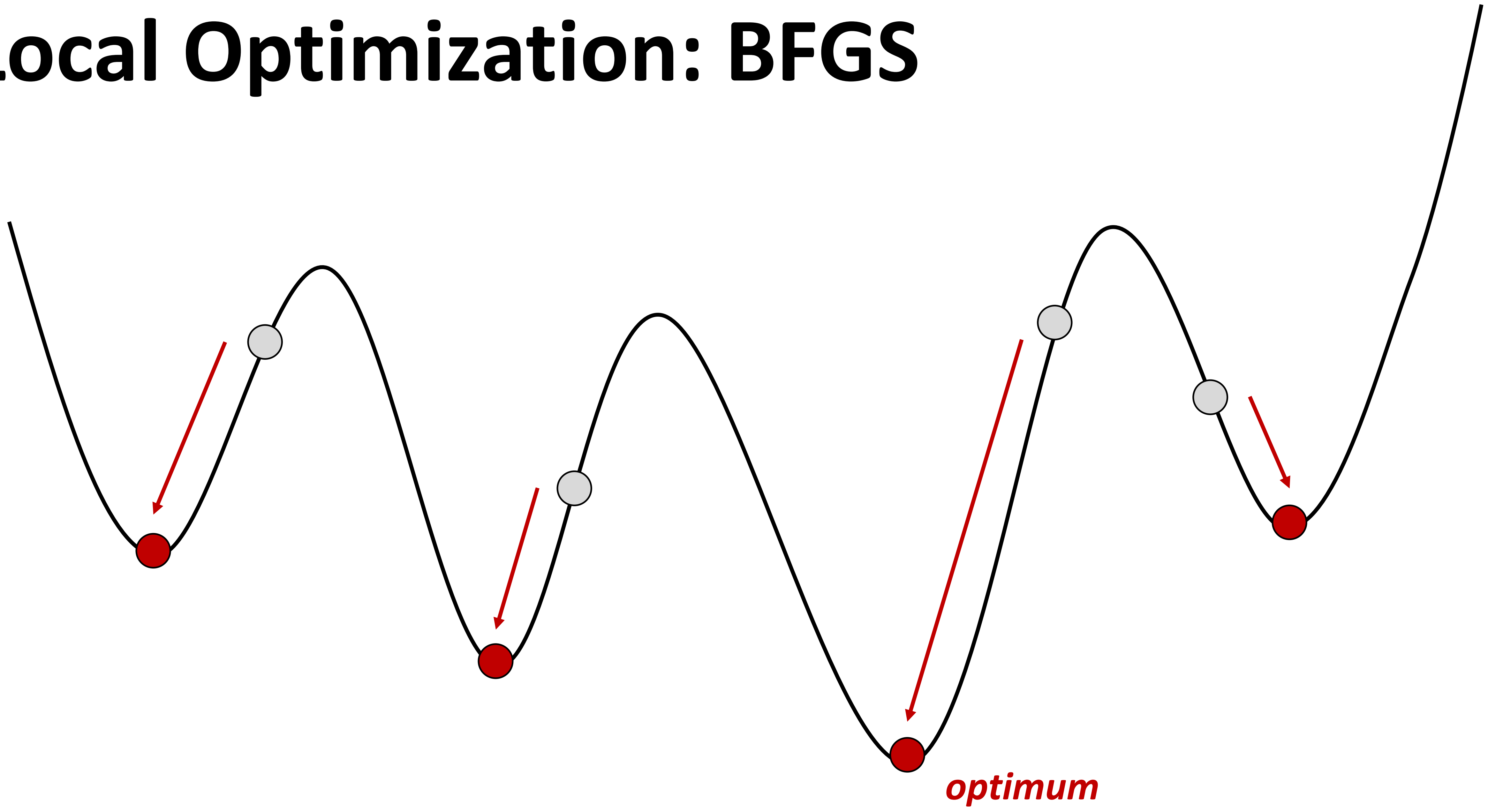
— Yan Fu (1853 - 1921)

Faithfulness Fluency Elegance

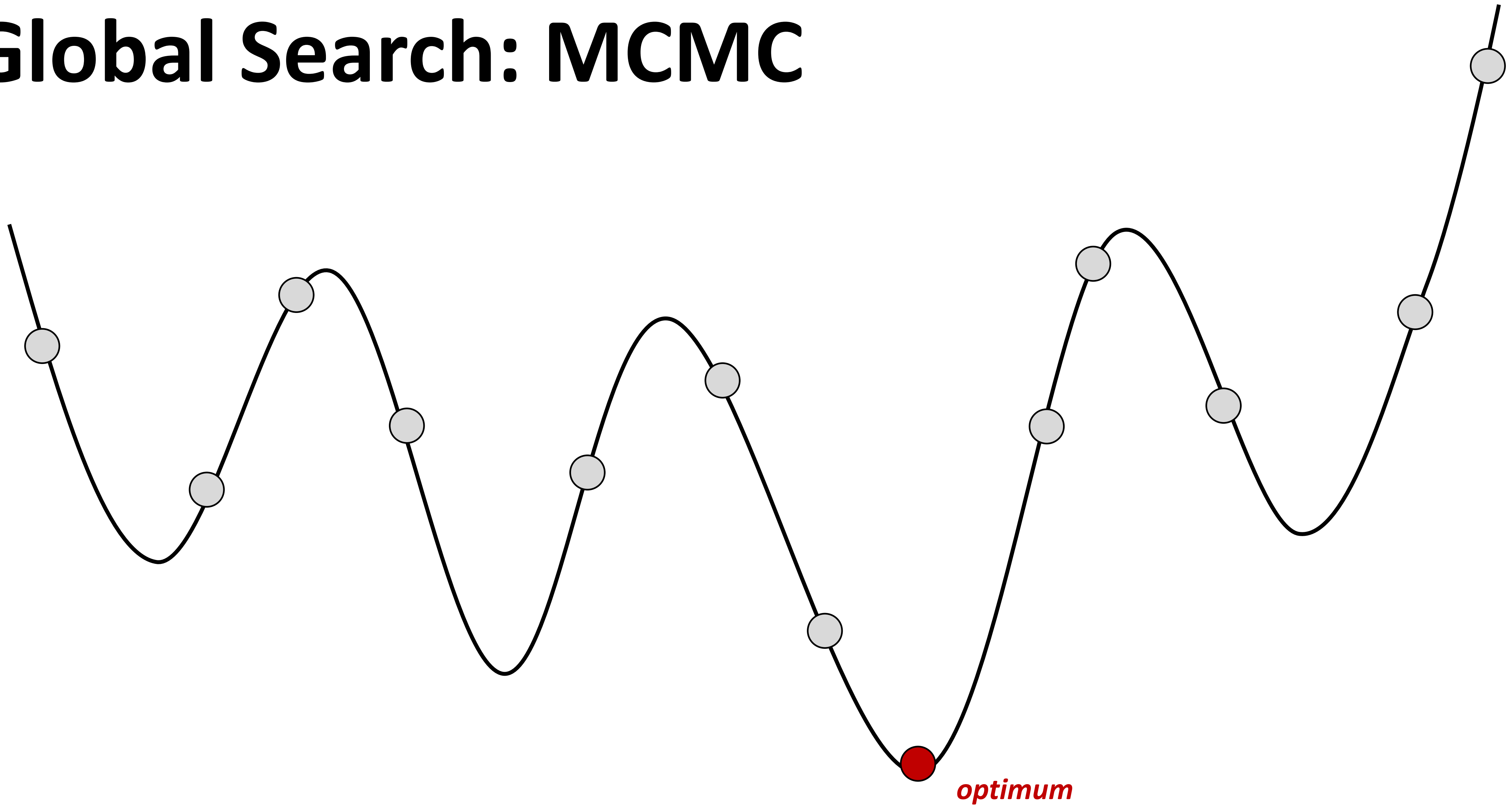
Local Optimization: BFGS



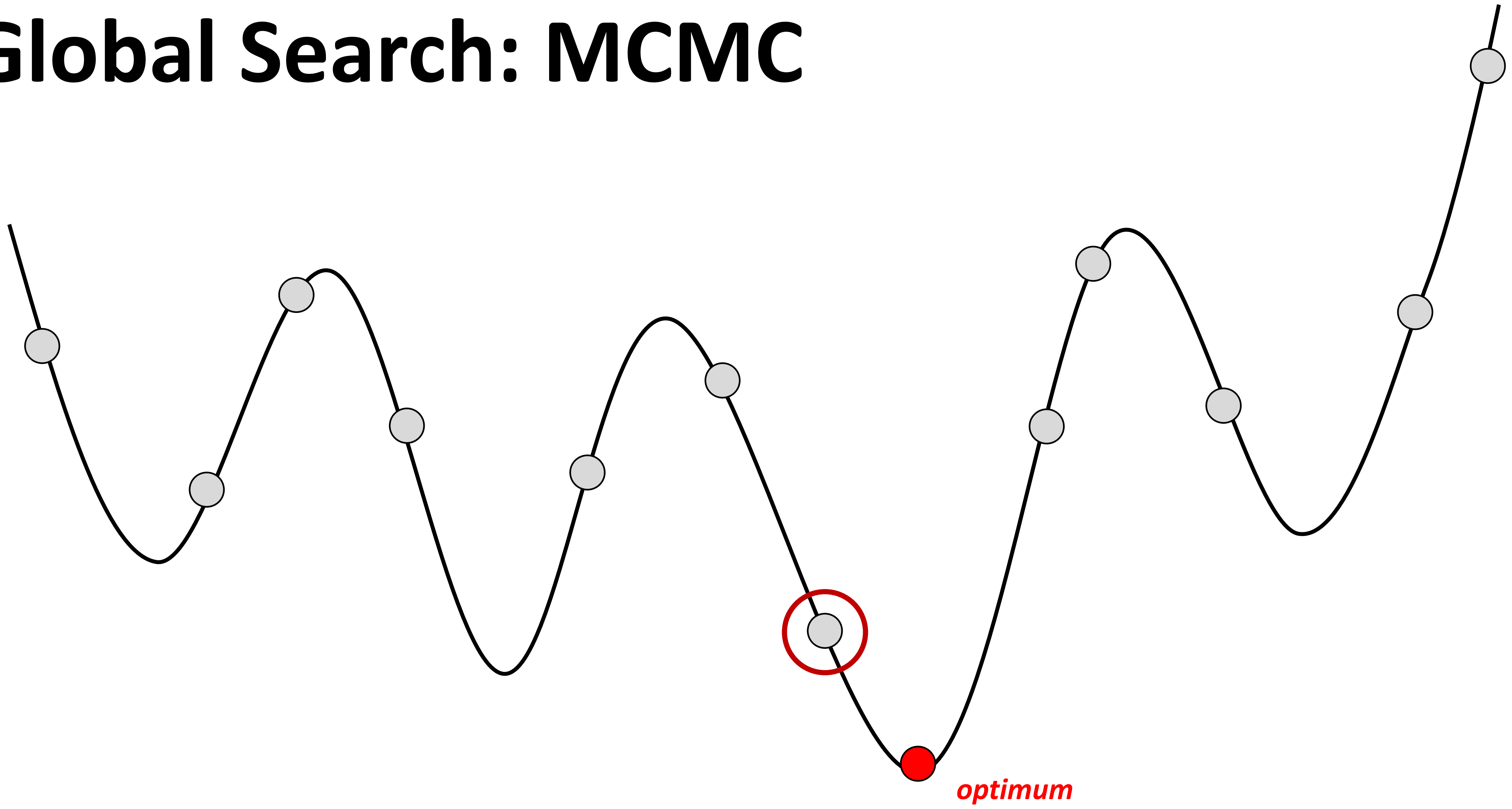
Local Optimization: BFGS



Global Search: MCMC



Global Search: MCMC



What is BFGS?

Broyden, **F**letcher, **G**oldfarb and **S**hanno

No Really what is BFGS?

Algorithm 2.2 BFGS算法

Input: $t \leftarrow 0, \quad \boldsymbol{\theta}_0 \leftarrow rand(0,1), \quad \mathbf{g}_0 \leftarrow \mathbf{g}(\boldsymbol{\theta}_0), \quad \mathbf{D}_0 \leftarrow \mathbf{H}(\boldsymbol{\theta}_0)^{-1};$

Output: $\boldsymbol{\theta}^{t+1}$ minimize cost function $\mathscr{L}_e(\boldsymbol{\theta})$;

while $\|\mathbf{g}_t\| > \varepsilon$ **do**

$\quad \mathbf{d}_t \leftarrow -\mathbf{D}_t\mathbf{g}_t;$ # compute search direction

$\quad \lambda_t \leftarrow \arg\min_{\lambda>0} \mathscr{L}_e(\boldsymbol{\theta}_t + \lambda \mathbf{d}_t);$ # line search meeting Wolfe conditions

$\quad \mathbf{s}_t \leftarrow \lambda_t \mathbf{d}_t, \quad \boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \mathbf{s}_t;$ # update parameters

$\quad \mathbf{g}_{t+1} \leftarrow \mathbf{g}(\boldsymbol{\theta}_{t+1}), \quad \mathbf{y}_t \leftarrow \mathbf{g}_{t+1} - \mathbf{g}_t;$

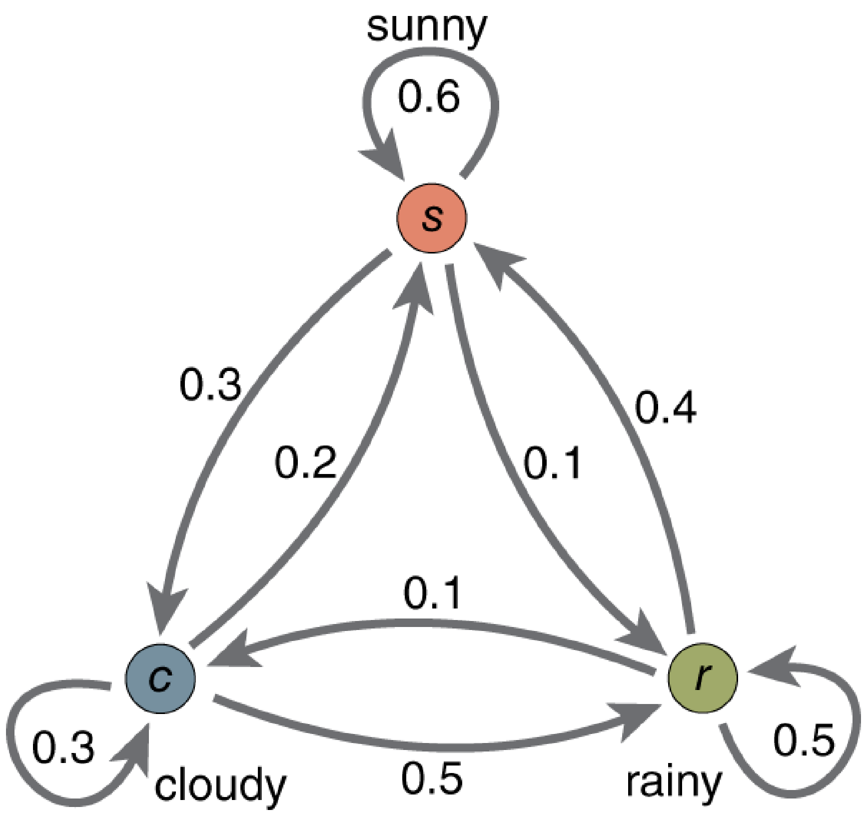
$\quad \mathbf{D}_{t+1} \leftarrow \left[\mathbf{I} - (\mathbf{y}_t^T\mathbf{s}_t)^{-1}\mathbf{s}_t\mathbf{y}_t^T\right]\mathbf{D}_t\left[\mathbf{I} - (\mathbf{y}_t^T\mathbf{s}_t)^{-1}\mathbf{y}_t\mathbf{s}_t^T\right] + (\mathbf{y}_t^T\mathbf{s}_t)^{-1}\mathbf{s}_t\mathbf{s}_t^T;$

$\quad t \leftarrow t+1;$

end while

What is MCMC?

Markov **C**hain **M**onte **C**arlo



Related to Subdivision Surfaces

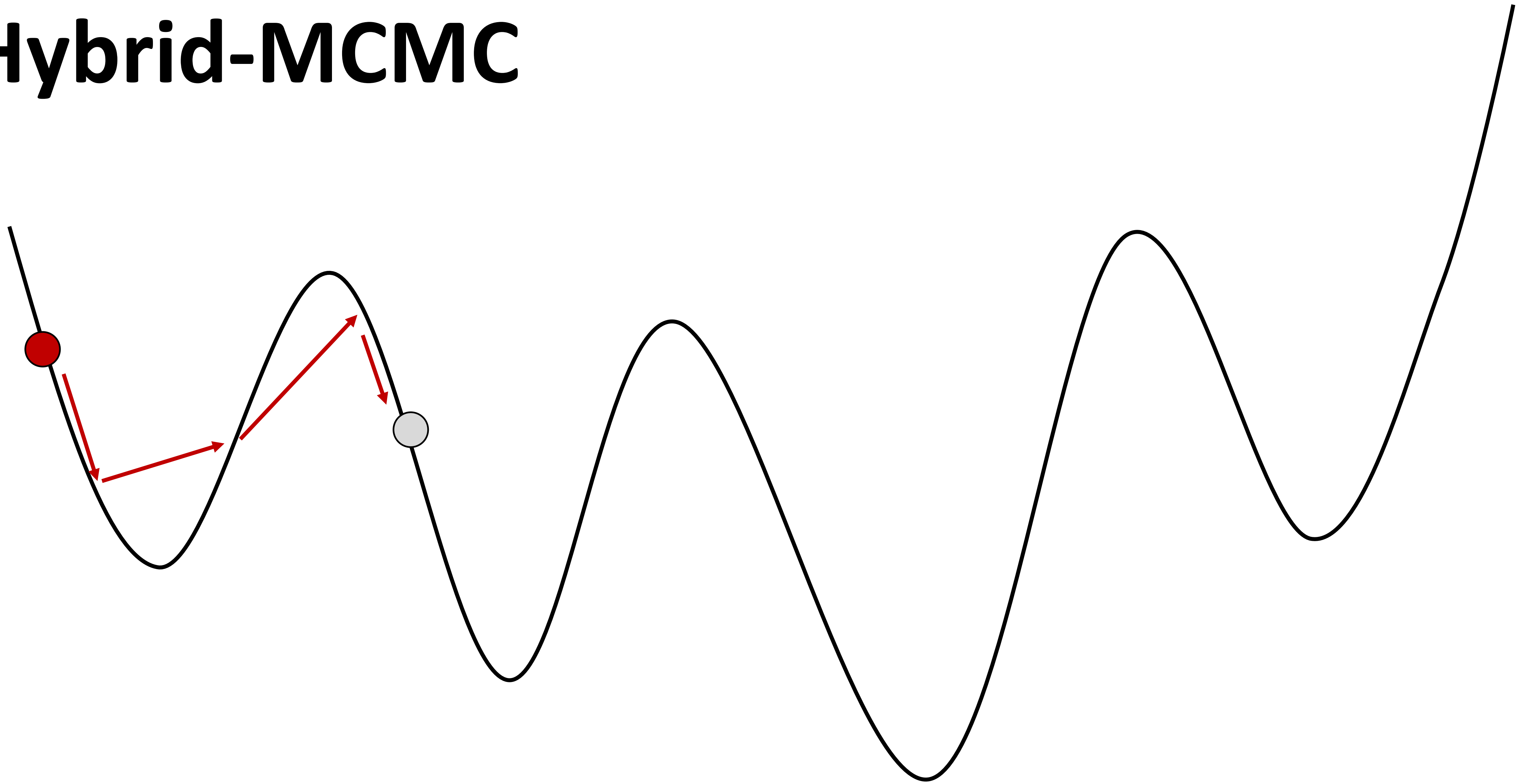
Limit probability distribution	Limit surface
Probabilities add to one	Affine invariance
Transition matrix	Subdivision matrix
eigenvalues	eigenvalues

Iterations of an operator reduced to powers of scalars!

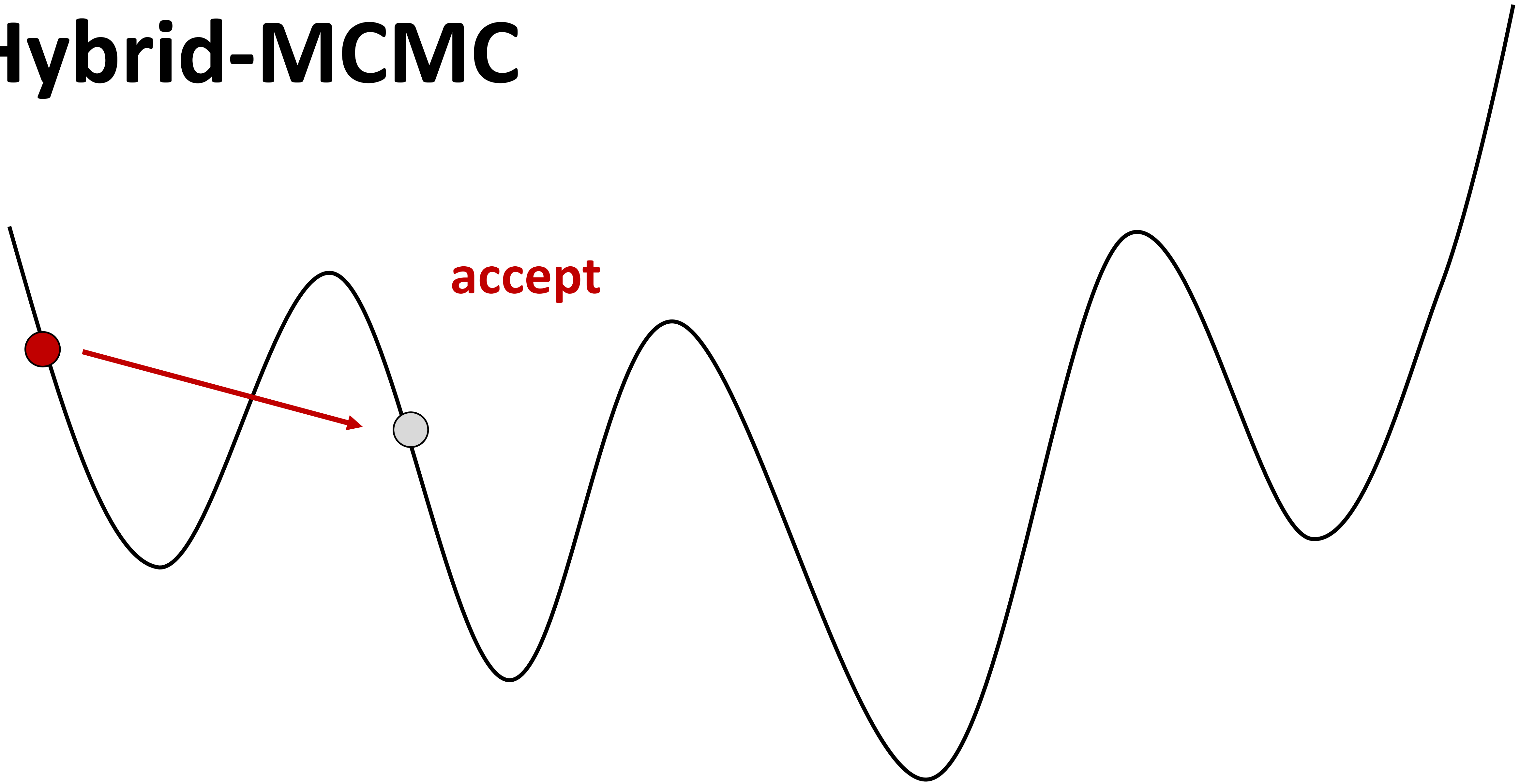
Hybrid-MCMC

Use the Gradient!

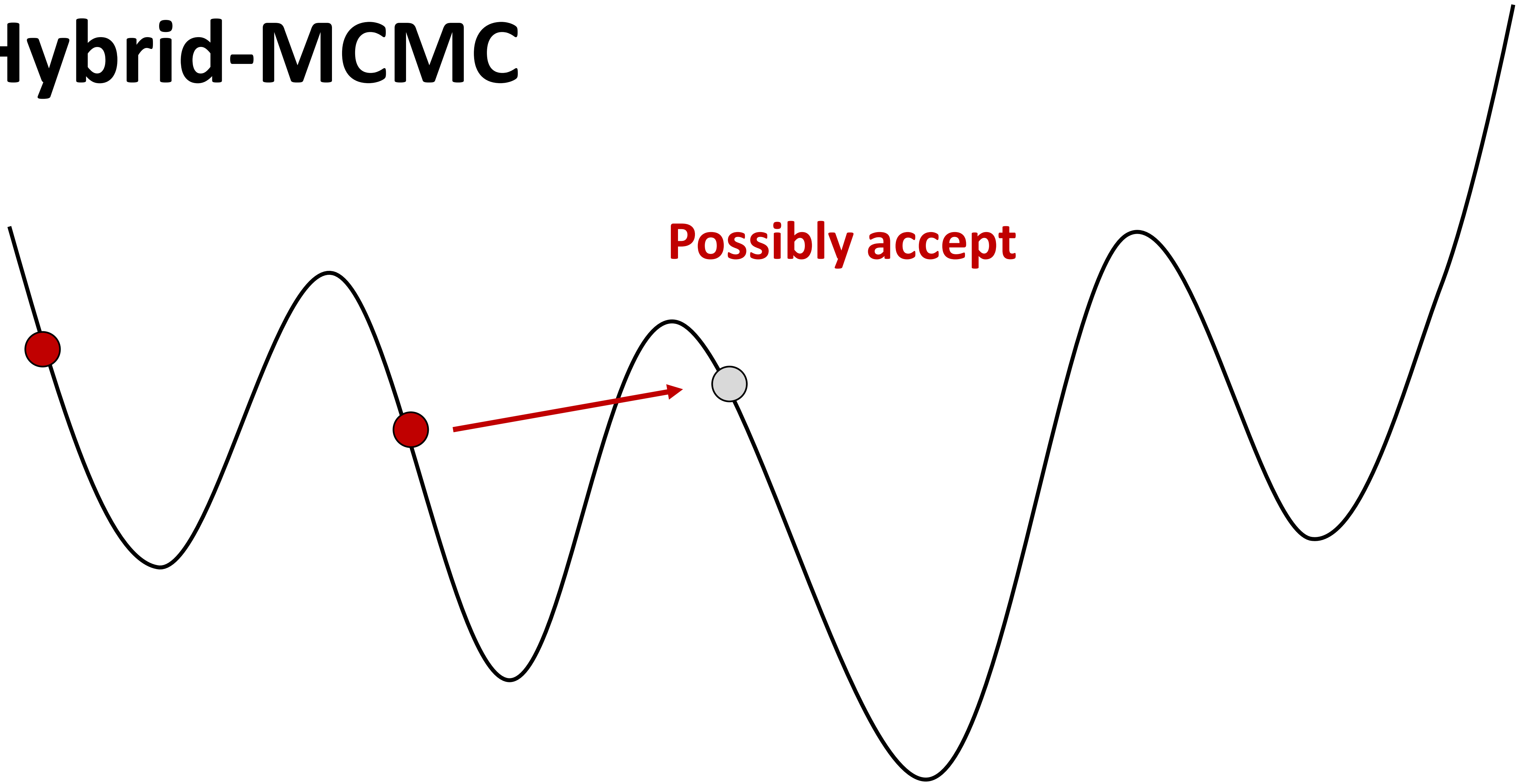
Hybrid-MCMC



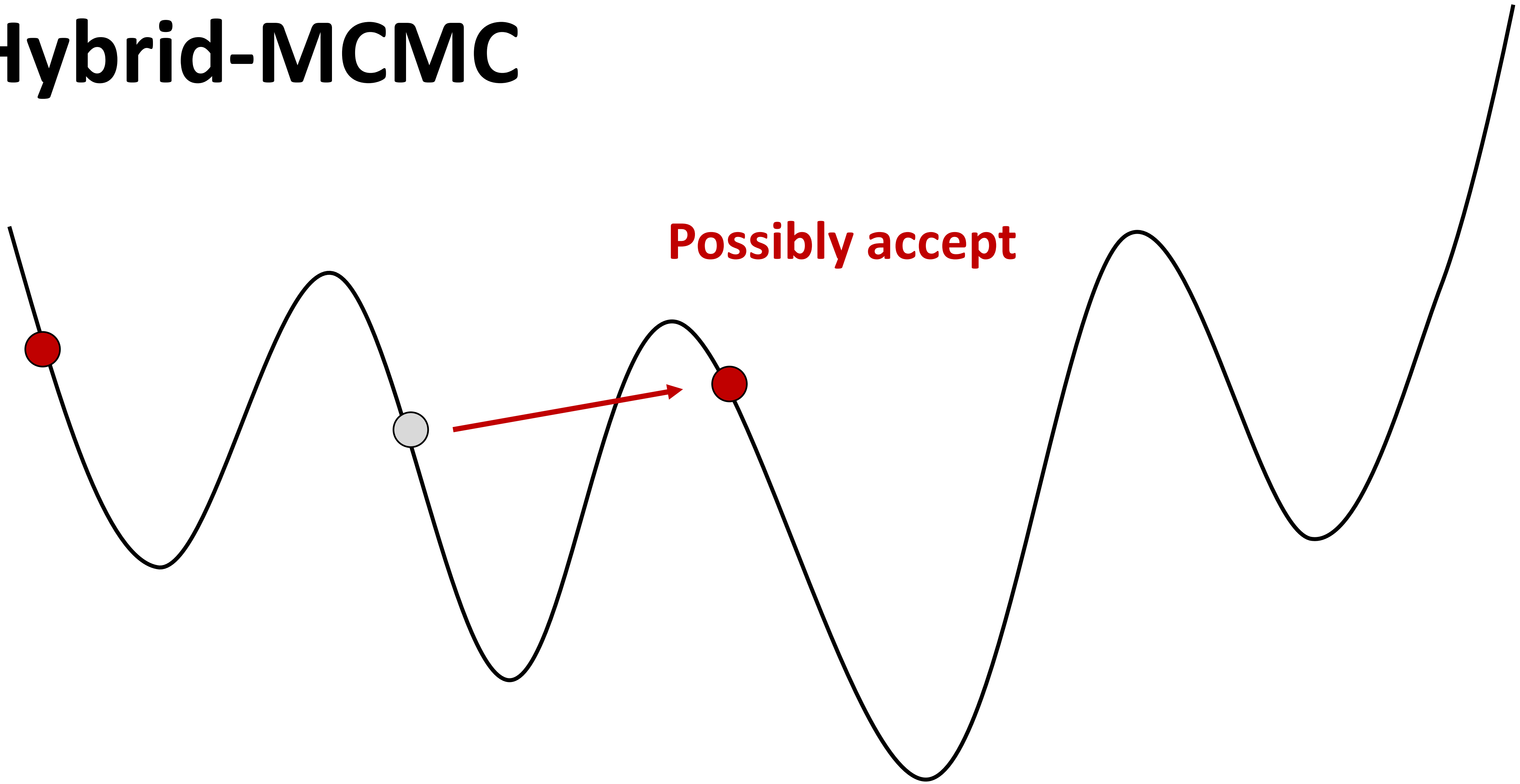
Hybrid-MCMC



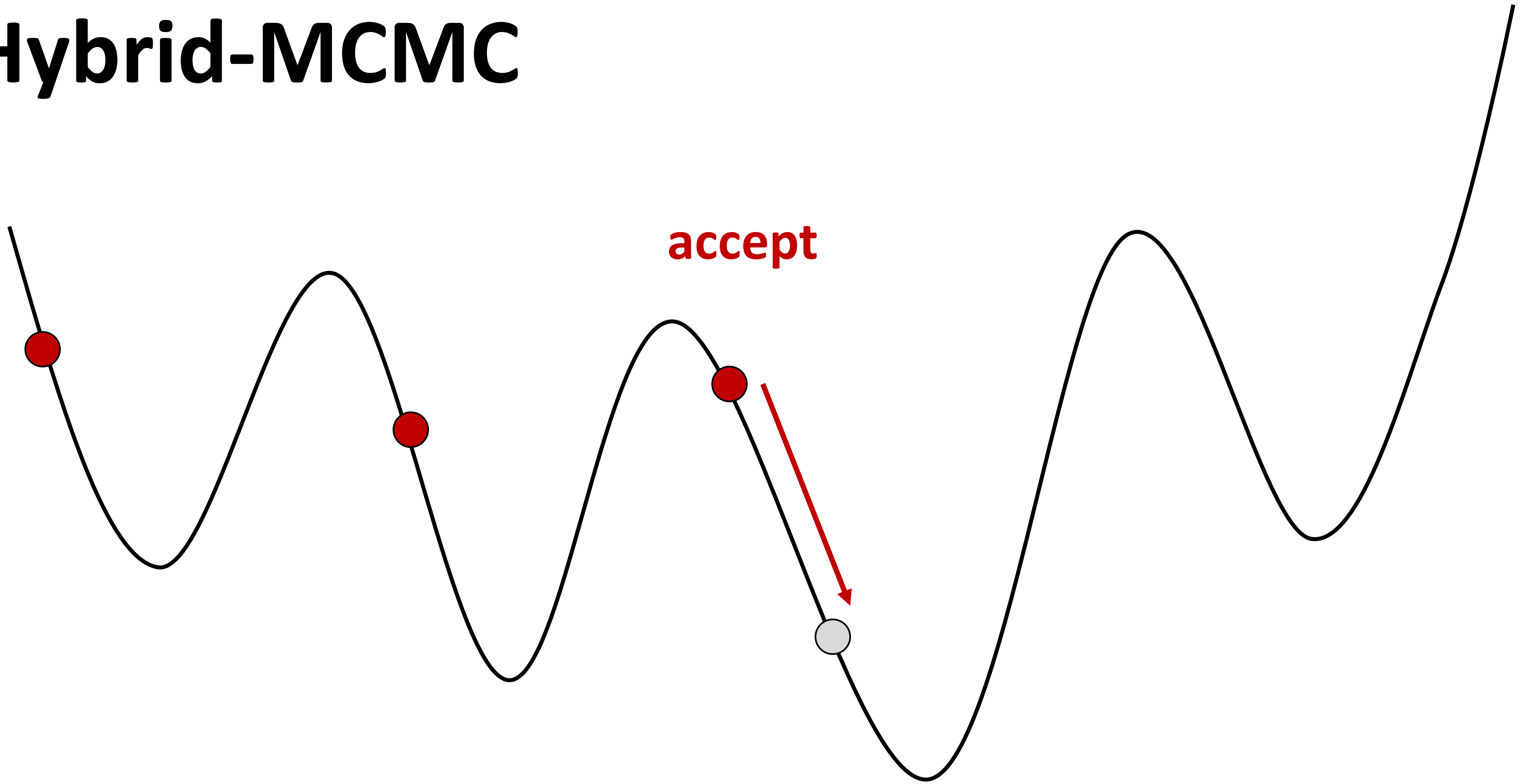
Hybrid-MCMC



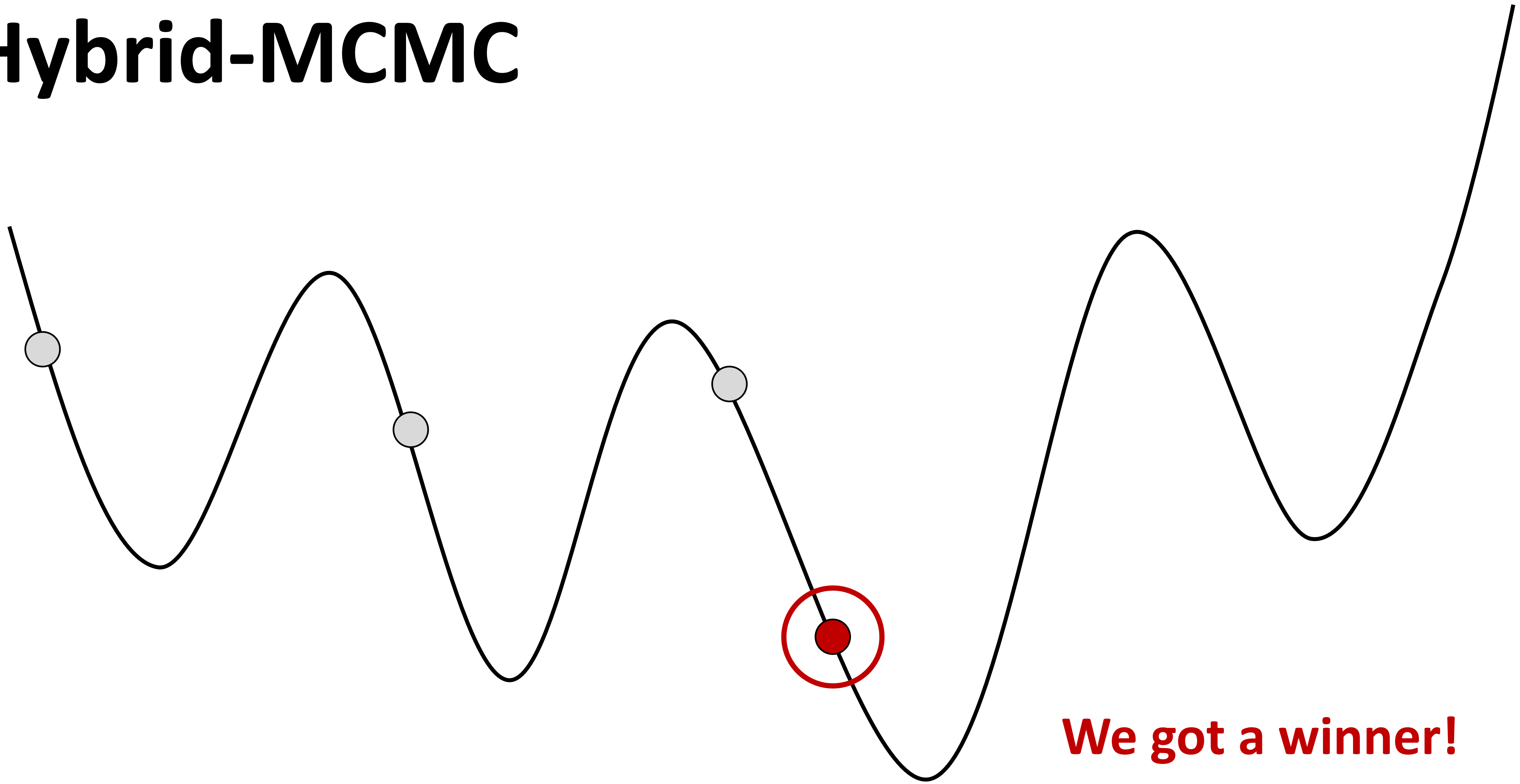
Hybrid-MCMC



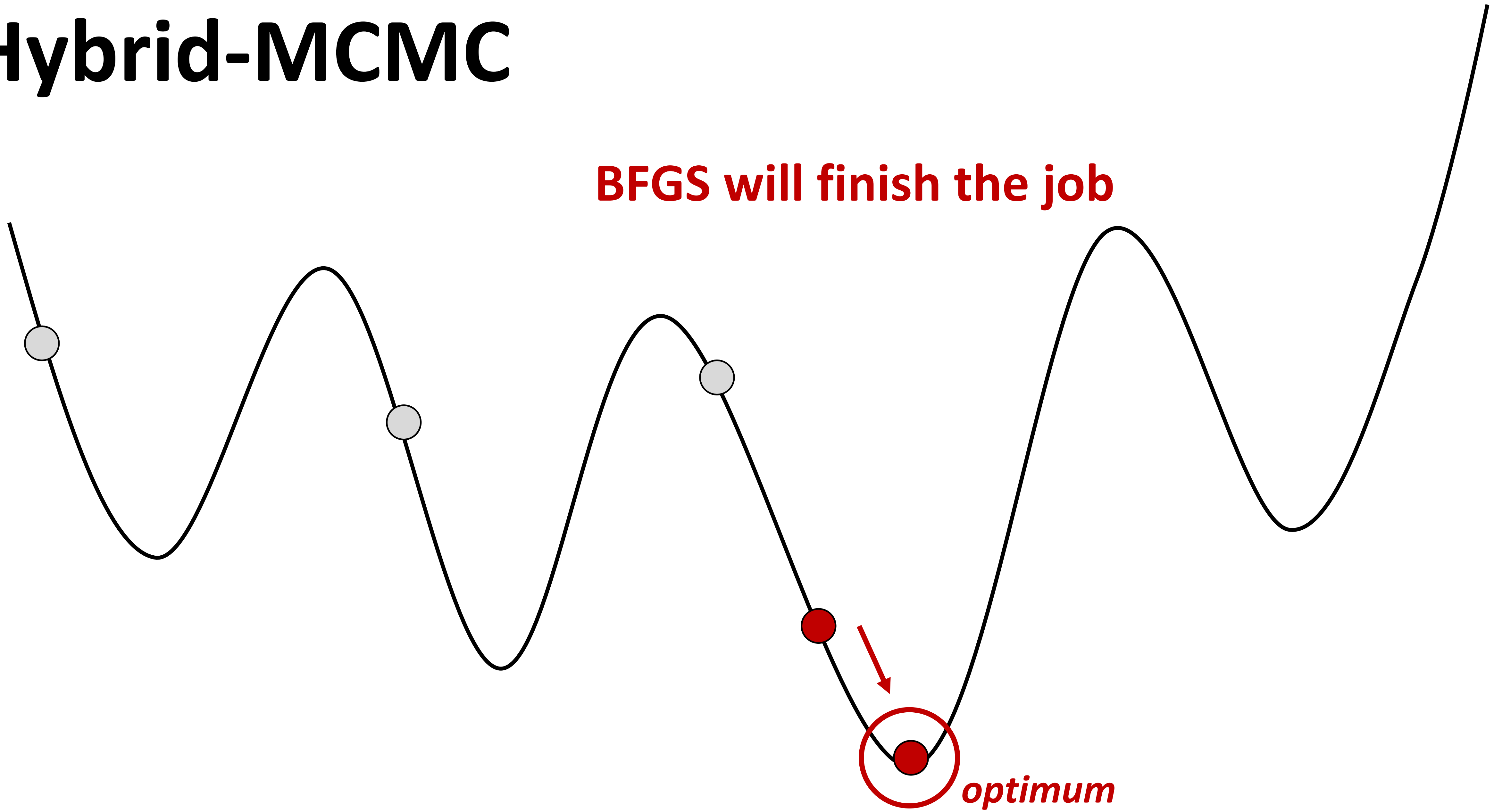
Hybrid-MCMC



Hybrid-MCMC

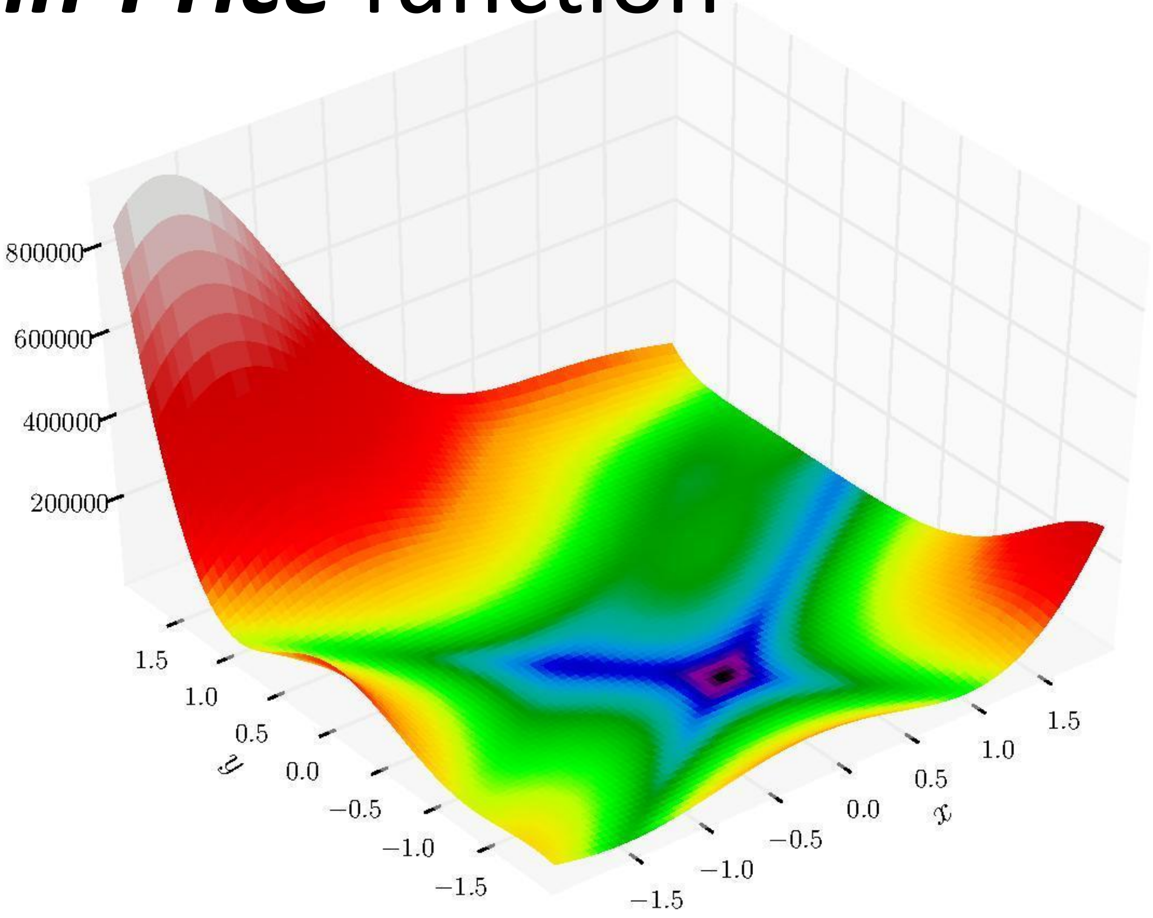


Hybrid-MCMC



Experiments: Validation

Goldstein-Price function

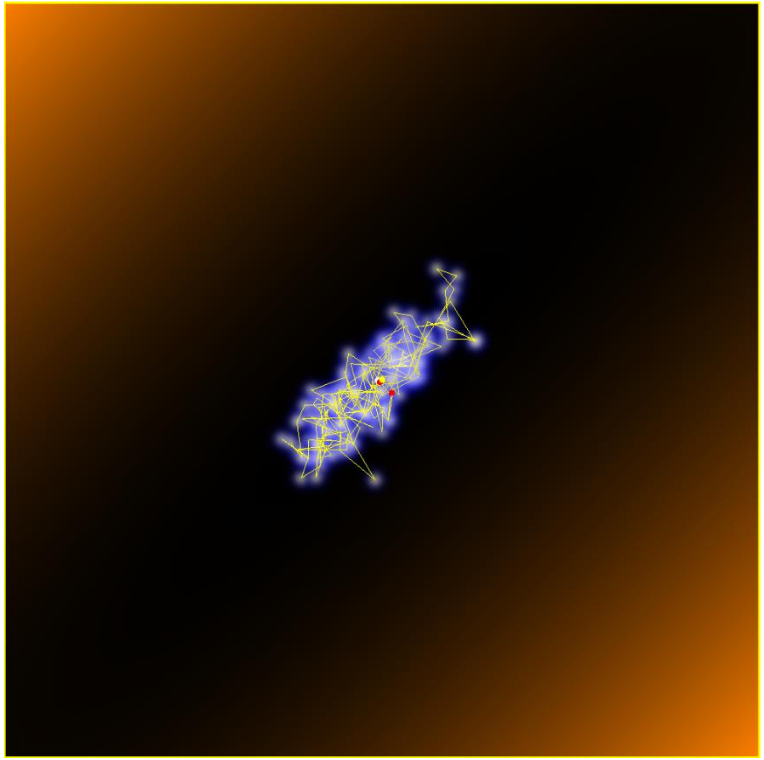


Experiments: Validation

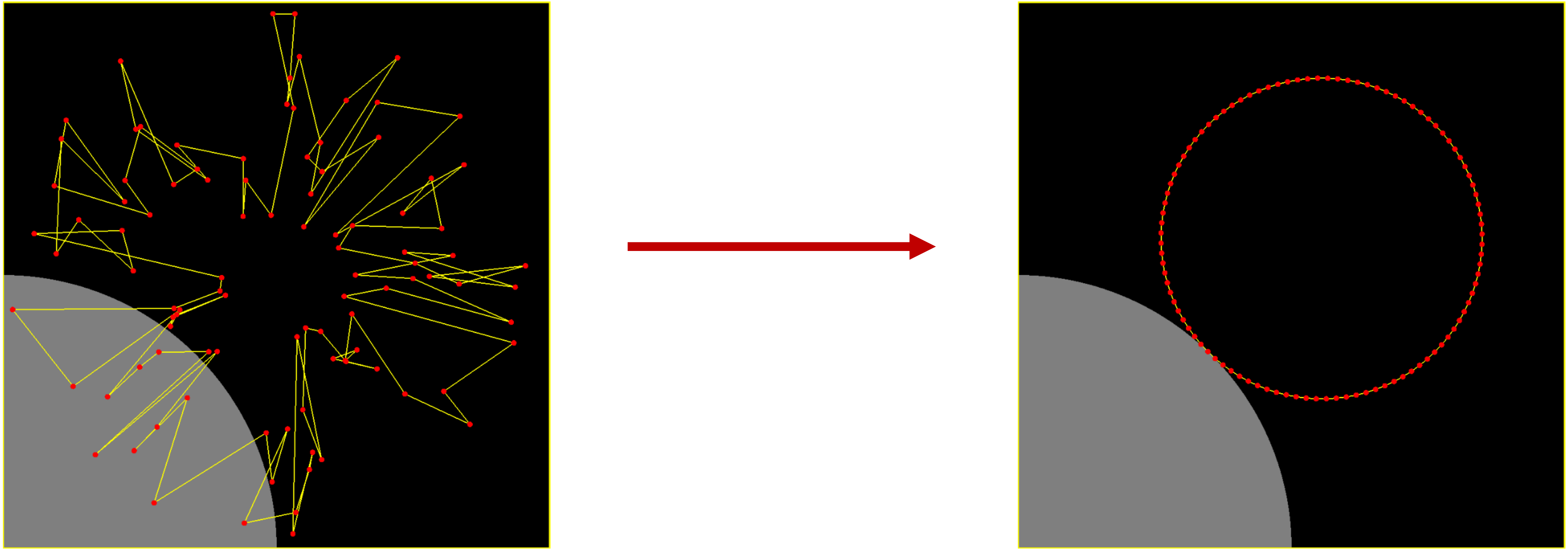
Goldstein-Price function

```
static dfloat<2> f_func ( dfloat<2> x, dfloat<2> y )
{
        dfloat<2> z;

        z = ( 1.0f + dsqr(x+y+1.0f) * (19.0f-14.0f*x+3.0f*dsqr(x)) ) *
        ( 30.0f + dsqr(2.0f*x-3.0f*y) *
        (18.0f-32.0f*x+12.0f*dsqr(x)+48.0f*y-36.0f*x*y+27.0f*dsqr(y)) );

        return ( z );
}
```

Experiments: Validation



Shape Optimization



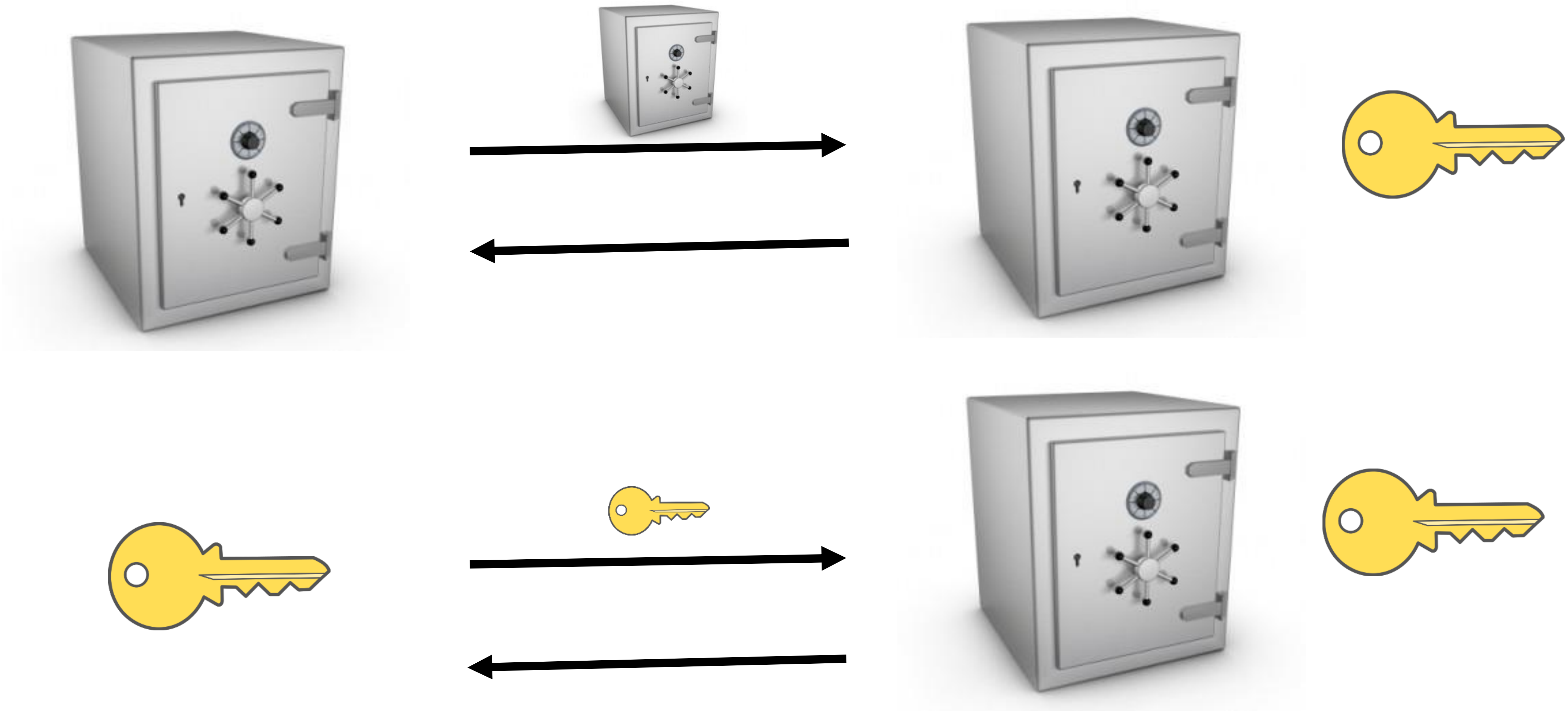
Discussion

It is great to prototype and good for small problems

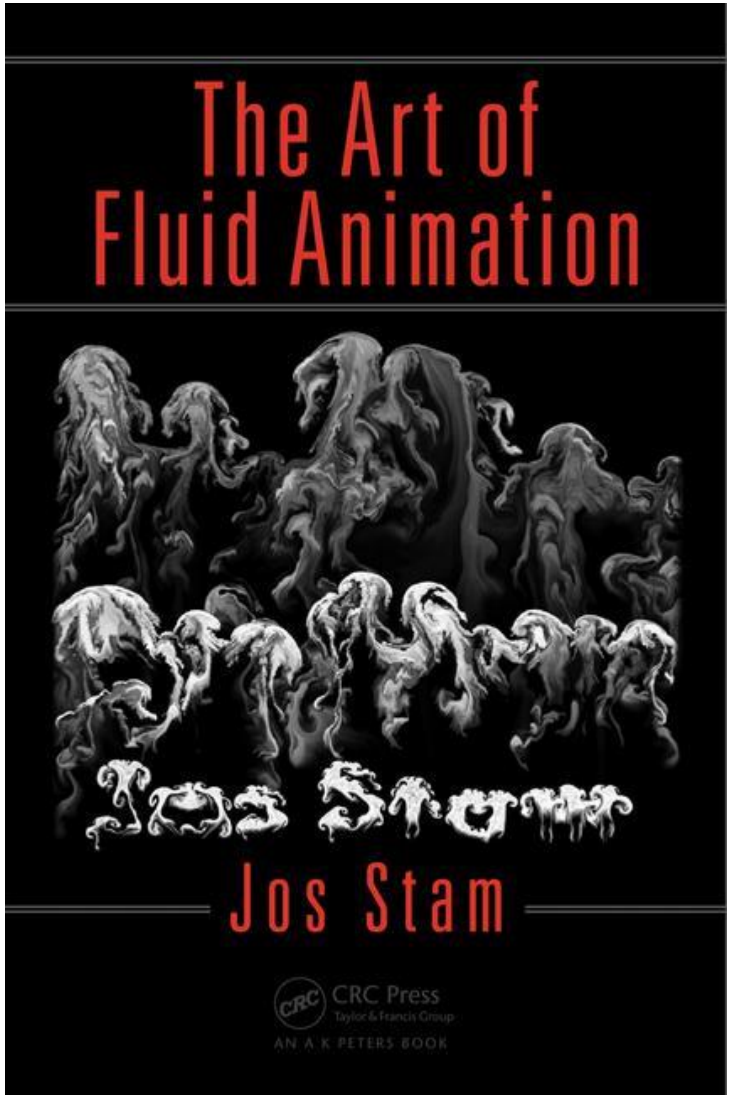
AD is not efficient for large problems

But it is still cool!

The Adjoint Method to the Rescue



Vin de Table



In Pin Yin: “you shi dan” : 尤 = kind of sounds like “Jos” 士 = “respected scholar” 丹 = “red.”

Merci!

a suivre…