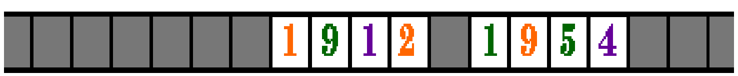
Computing over the Reals: Where Turing Meets Newton

Alan Turing

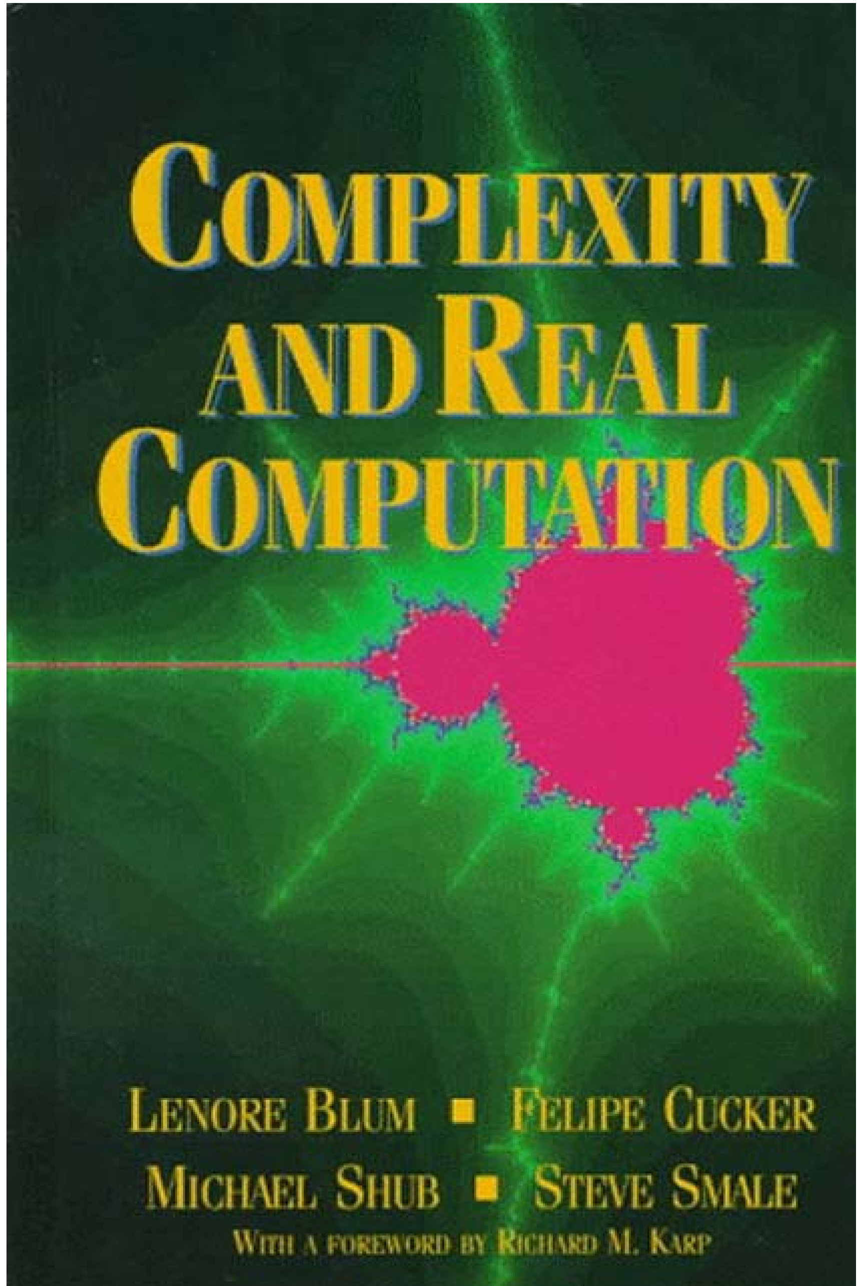


Sir Isaac Newton (1642-1727)

Lenore Blum

Computer Science Department

Carnegie Mellon



COMPLEXITY AND REAL COMPUTATION
LENORE BLUM ▪ FELIPE CUCKER
MICHAEL SHUB ▪ STEVE SMALE
WITH A FOREWORD BY RICHARD M. KARP

Mike Shub, Lenore Blum, Felipe Cucker, Steve Smale

Photo taken by Victor Pan at Dagstuhl, 1995.

Two Traditions of Computation

NumericalAnalysis/ ScientificComputation

- Newton's Method Paradigm Example in Most Numerical Analysis Texts

•No Mention of Turing Machines 

•Real & Complex #'s Math is Continuous

•UnDevelopeding Foundations of CM

Logic/ComputerScience

- Turing Machine Underlying Model in most CS Texts on Algorithms
- No Mention of Newton's method

•0's & 1's (bits)  Math is discrete

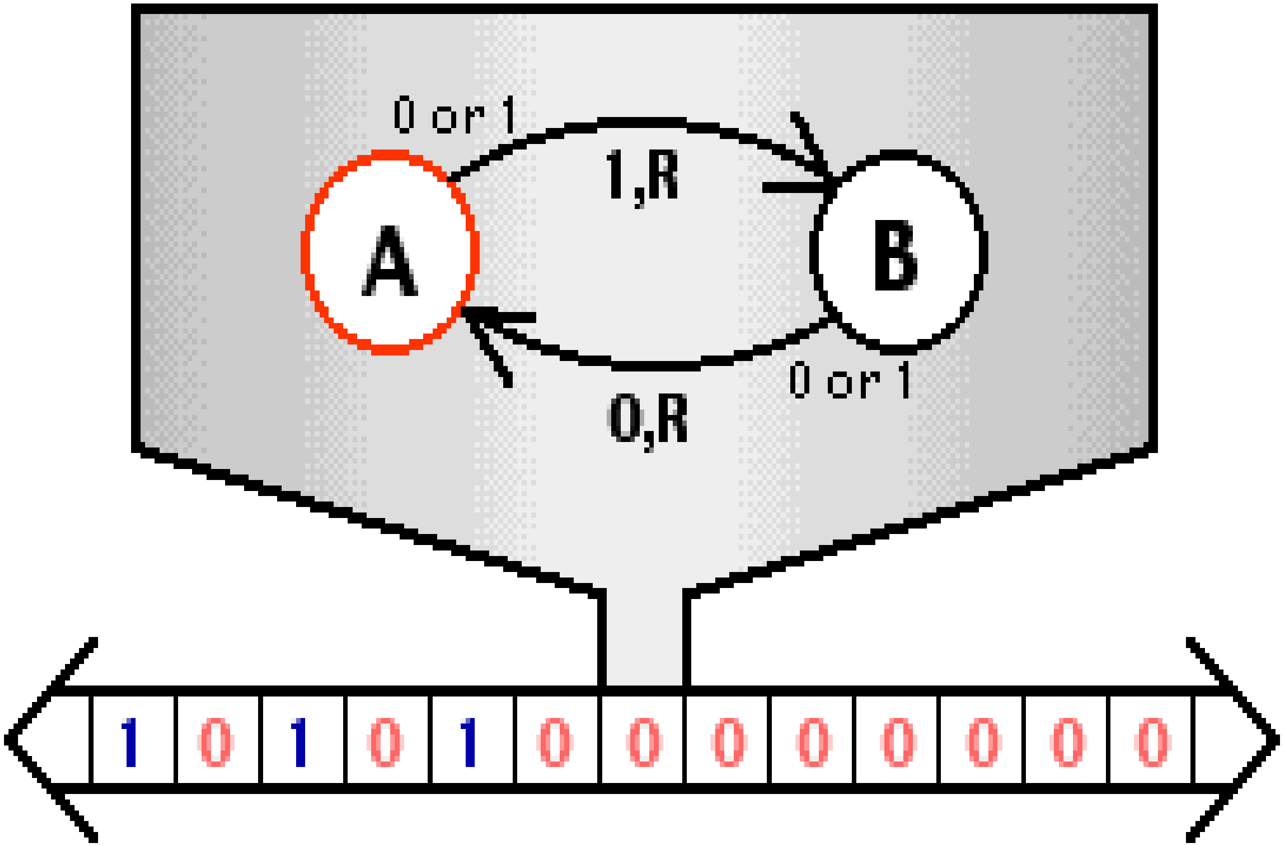
•Highly Developed Foundations Of CS

- Want to reconcile dissonance
- Build bridges
- Unify
- Traditions/tools of each should inform the other*

*Examples

FOCS → FoCM: Complexity Theory

FoCM →FOCS: Condition



The Turing Machine provides the Mathematical Foundation for the Classical Theories of

Computation and **Complexity**

Logicians in the 1930's-40's (Godel, Church, Kleene, ...)

Computer Scientists in the 60's-70's (Rabin, M.Blum, Cook, Karp, Levin...)

*TM courtesy of Bryan Clair

Since the 1930's, many seemingly different models have been proposed for a general theory of (discrete) computing. What has been striking is that all such models have given rise to the *exact same class* of "computable" functions: the class of input-output maps of Turing machines are exactly the computable functions derived from Post production systems, as from flow-chart machines, as from random access machines, as from axiomatic descriptions of *recursive functions*, etc…

The has been essentially true wrt the accompanying complexity theories (up to polynomial time): poly-time and intractability are essentially invariant across platforms. Thus logicians and computer scientists have confidence they are working with a very natural class of functions and justifies the use of their favorite model. Even more, this naturalness/invariance underlie the impact theory has had on technological developments, be it in programming, the design of computers or cryptography.

The classical Turing tradition yields:

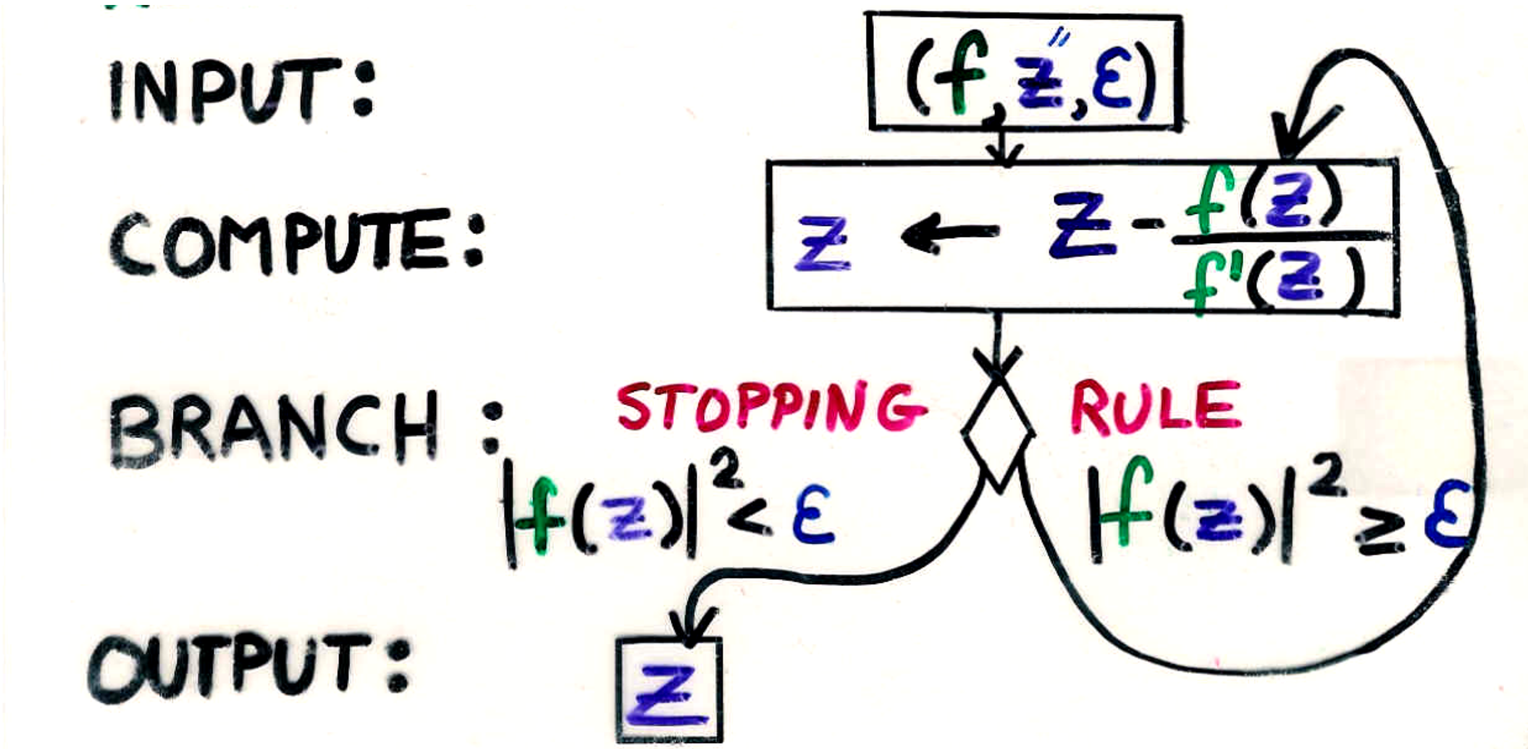
- A highly developed and *rich* (invariant) *theory of computation* and *complexity*
- With *important applications* to computation, cryptography, security, etc
- And, *deep interesting problems.*

Why do we want a new model of computation?

Motivation 1 for New Model

Want model of computation which is more natural for algorithms of numerical analysis, such as Newton's Method.

"Newton Machine"



Paradigm method of numerical analysis.
Translating to 'bit' operations would wipe out the natural structure of Newton's algorithm.

Motivation 2 for New Model

Want model* of computation where it is more natural to pose and answer questions of decidability and complexity about problems and sets over the real and complex numbers.

*uniform model

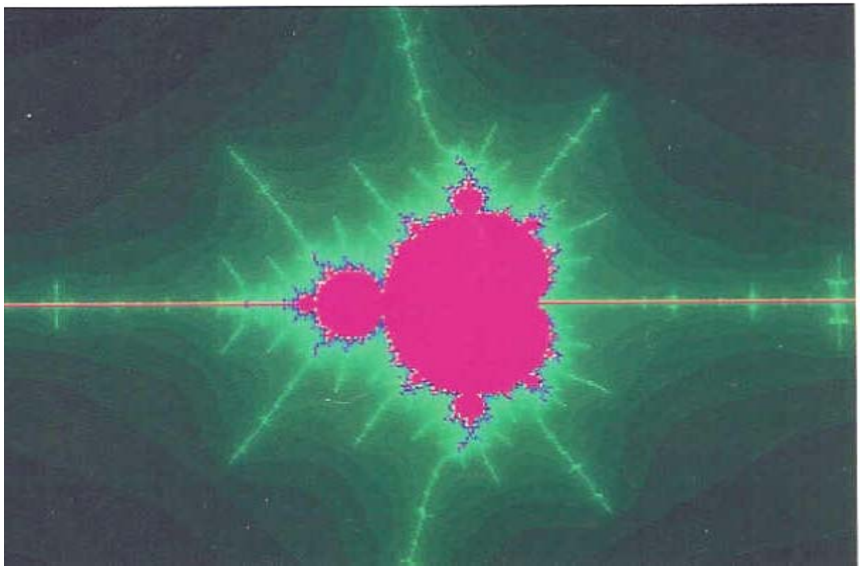
NATIONAL BESTSELLER

THE EMPEROR'S NEW MIND

Concerning Computers, Minds, and the Laws of Physics

Roger Penrose

Roger Penrose (1989).
Is the Mandelbrot set decidable?

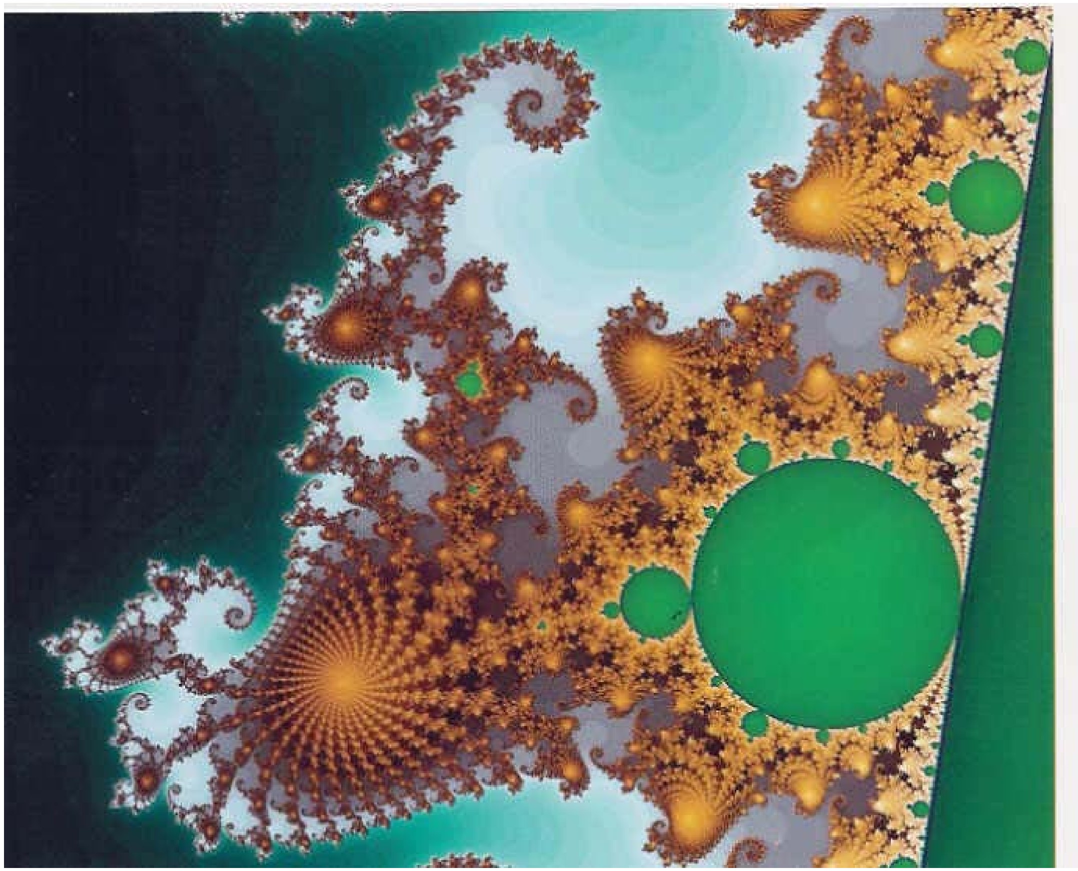


$M=\{c \mid p_c^n(0) \not\to \infty\}$, $p_c(z) = z^2+c$, $c \in \mathbb{C}$, $M \subset \mathbb{C} = \mathbb{R}^2$

"Now we witnessed, …, a certain extraordinarily complicated-looking set, namely the **Mandelbrot** set. Although the rules which provide its definition are surprisingly simple, the set itself exhibits an endless variety of highly elaborate structure.

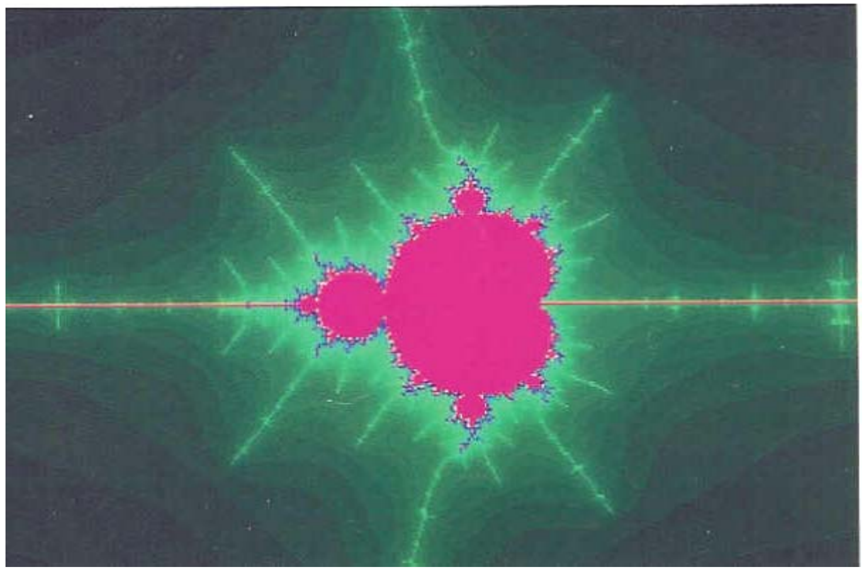
Could this be an example of an undecidable set, truly exhibited before our mortal eyes?"

Complexity on the boundary in Seahorse Valley



Roger Penrose (1989).

Is the Mandelbrot set decidable?



$$M=\{c \mid p_c^n(0) \not\to \infty\},\ p_c(z) = z^2+c,\ c \in \mathbb{C},\ M \subset \mathbb{C} = \mathbb{R}^2$$

After considering possible interpretations via the classical theory of computation, Penrose concludes:

"… one is left with the strong feeling that the correct viewpoint has not yet been arrived at."

Motivation 3 for New Model

New perspectives for

$$P = NP?$$

The Model
(BSS,'89)

Inspired by *both* computer science and numerical analysis

From Numerical Analysis…

CIT

THE QUARTERLY JOURNAL OF

MECHANICS AND APPLIED MATHEMATICS

VOLUME I
1948

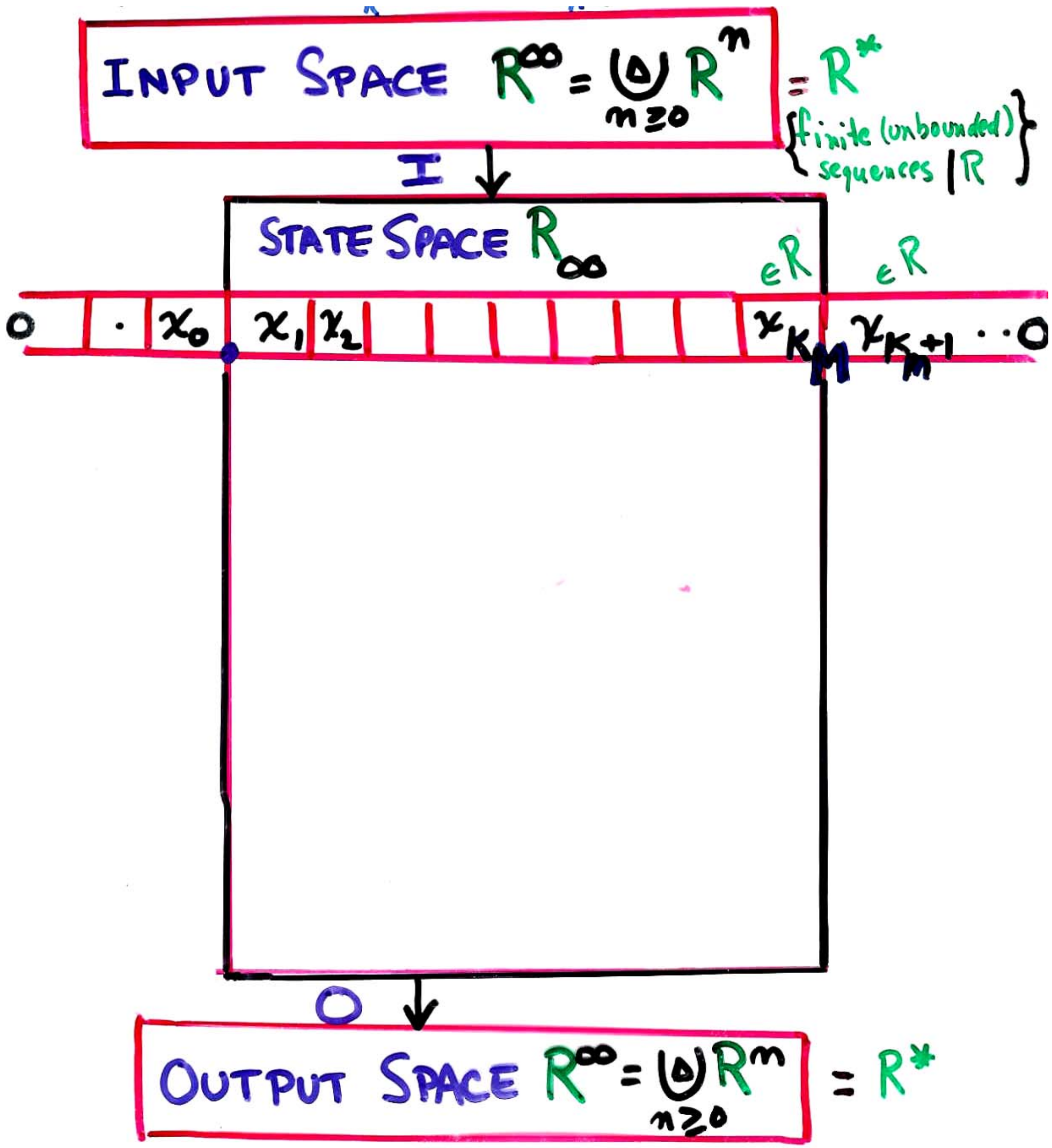
OXFORD
AT THE CLARENDON PRESS

“Rounding-off Errors In Matrix Processes”
The Quarterly Journal of Mechanics and Applied Mathematics, **vol. I, 1948**, pp. 287-308.

1. Measures of work in a process

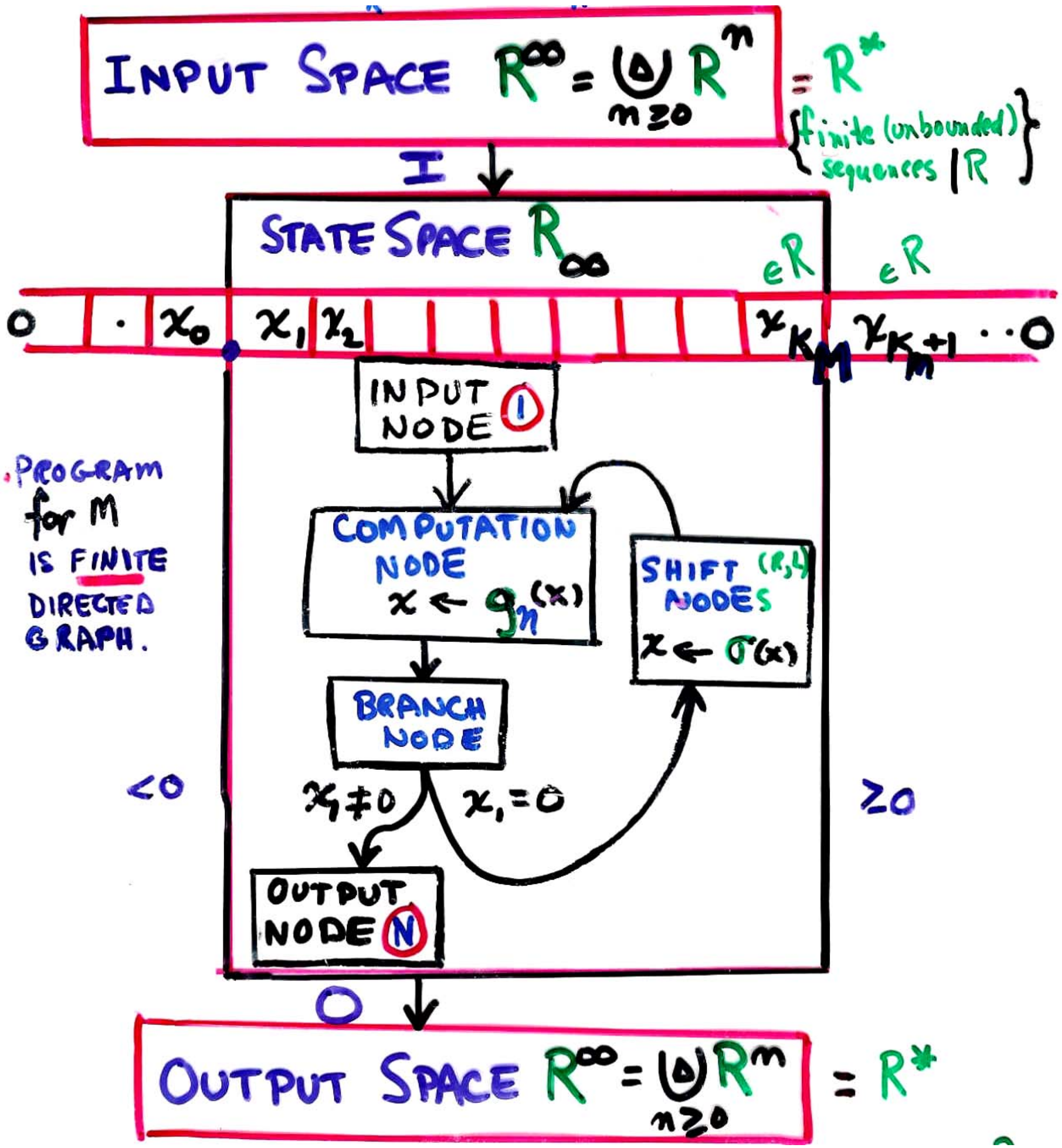
It is convenient to have a *measure of the amount of work involved in a computing process*, even though it be a very crude one…We might, for instance, *count the number of additions, subtractions, multiplications, divisions, recordings of numbers, …*

Machine M over a Ring R (+.x) (BSS '89)



R is commutative with unit. **M** looks like a Turing Machine, but inside …

Machine M over Ring R (+.x) (BSS '89)



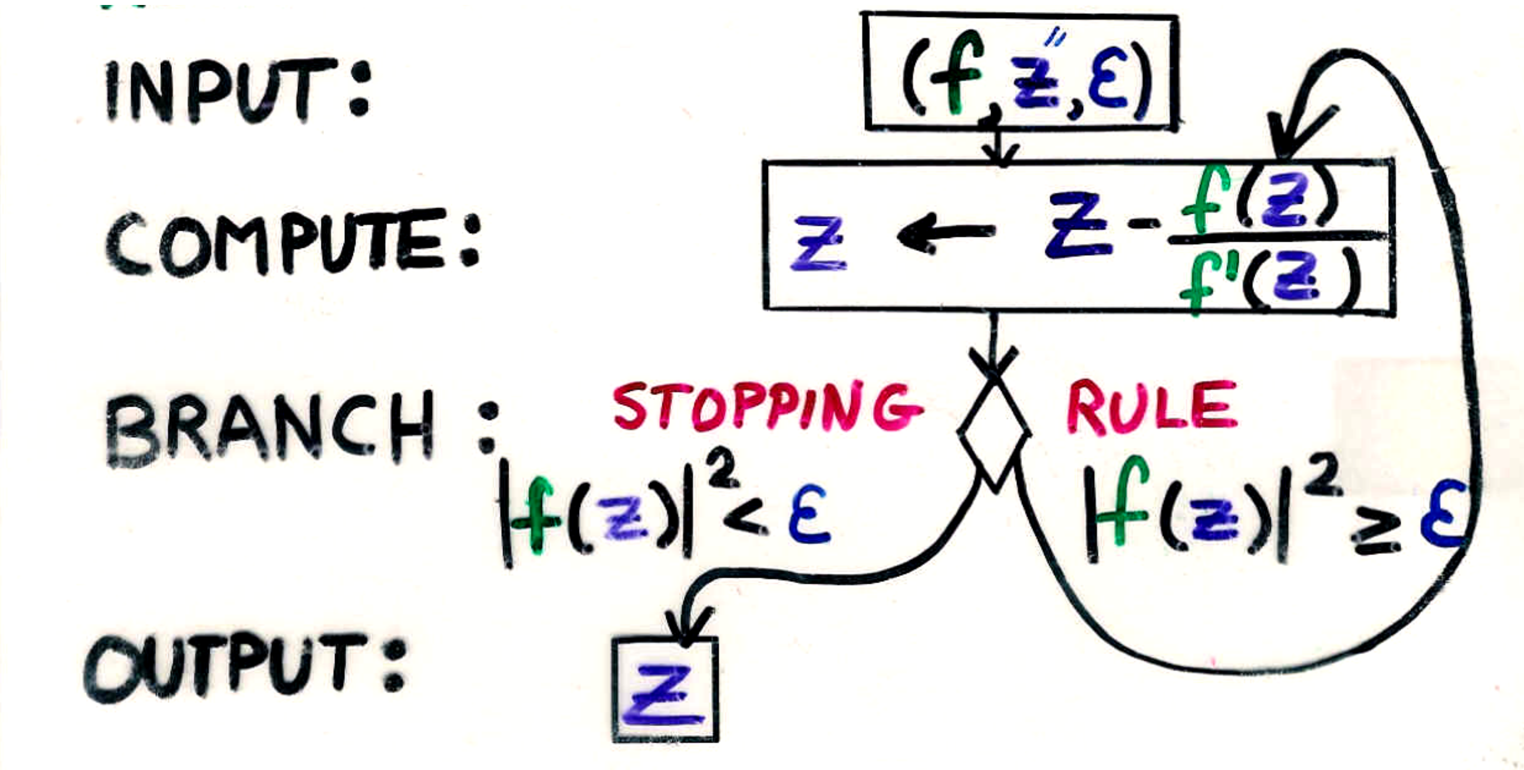
M looks like a Newton Machine inside.

M can have a finite # of built-in constants from R.

Features of Machine over R

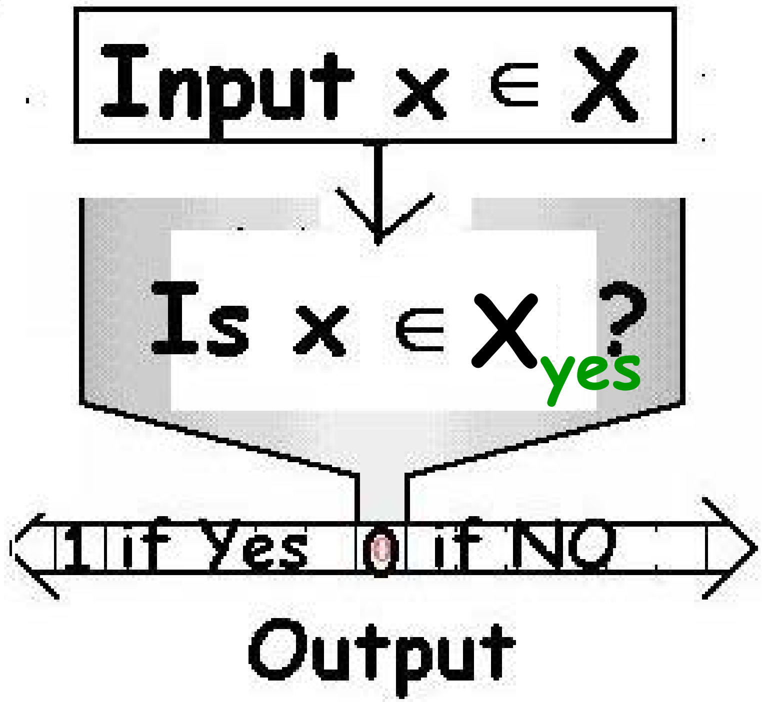
- Computation nodes: built in g_η : $R^n \to R^m$, polynomial or rational map (given by a finite # of polynomials in a finite # of variables).
- Branch: on = if R, unordered ring or field,
  on < if R is ordered.
- Shift nodes: shift one cell right or left
- Machine is uniform over R^n, for all n.
- Computable fns over R: the input-output maps.
- Can construct universal (programmable) machines.
  [We do not use Godel coding. The program itself is (essentially) its own code.]
- If R is Z_2 , we recover the classical theory of computation (and complexity, as we shall see).

"Newton Machine"



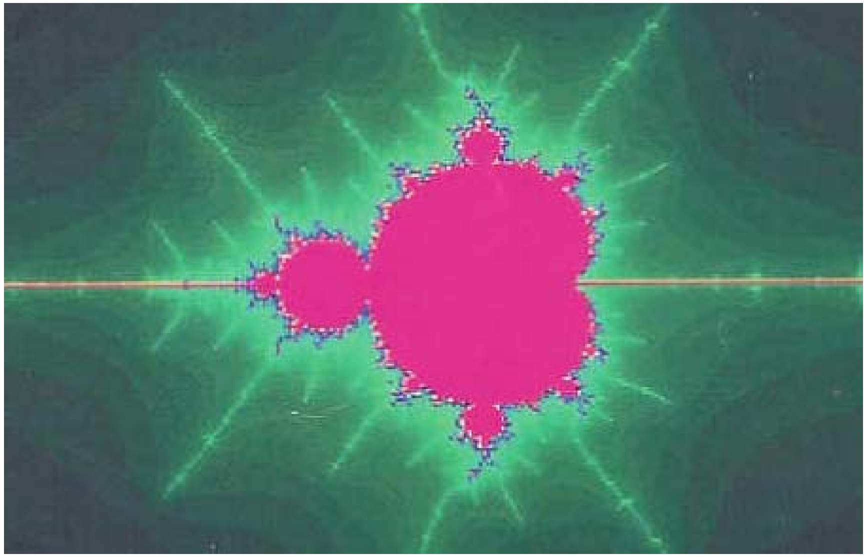
is **Perfectly good Machine** over the **Reals**

A Problem_R (X, X_{yes}) is Decidable if there is a Machine over R such that…



So, now can formally pose Penrose's question (over the Reals): $\boxed{\text{Is } (X, X_{yes}) \text{ decidable over } R?}$

Here R=Reals, $X=R^2$ and X_{yes} = Mandelbrot Set.



---

*<u>Theorem</u> (Blum, Smale).
The Mandelbrot Set is Undecidable/Reals

Complexity Theory over a Ring

For $x \in R^\infty = \cup R^n$

Size $x=n$ if $x \in R^n$ (vector length)

$\text{Time}_M(x)$= # of nodes from *input* to *output*

$(X, X_{yes}) \in P_R$ if $\exists$ Decision Machine M & poly such that for each $x \in X$, $T_M(x) < \text{poly}(\text{size } x)$

$(X, X_{yes}) \in NP_R$ if $\exists\ (Y, Y_{yes}) \in P_R$ & poly such that for each $x \in X$, $\exists$ witness $w \in R^{\text{poly}(\text{size } x)}$

$[x \in X_{yes} \Leftrightarrow (x, w) \in Y_{yes}]$

If $R = Z_2 = \{0,1\}$ then $T_M(x) \sim$ bit cost and so, recover classical complexity theory,

i.e. $P_{Z_2} = P$(classical) and $NP_{Z_2} = NP$(classical)

•**Let HN_R** be the **problem** of **deciding** whether or not a given **polynomial system** over **R** has a solution (**zero**) over **R**.

So, $HN_R = (X, X_{yes})$ where

$X = \{f^* = (f_1,\ldots,f_m) \mid f_i \in \mathbf{R}[x_1,\ldots,x_n], m, n > 0\}$

$X_{yes} = \{f \in X \mid \exists\, \zeta \in \mathbf{R}^n, f_i(\zeta_1,\ldots,\zeta_n) = 0,\ i=1,\ldots,m\}$

•$HN_R \in NP_R$:

If $= (f_1,\ldots,f_m) \in X_{yes}$ then $\exists\, w \in \mathbf{R}^n$ such that $f_i(w_1,\ldots,w_n) = 0$, $i=1,\ldots,m$

(**w=witness** and **checking** that $f(w)=0$ **is** **poly-time**)

- **Theorem** (Cook/Levin'71) $P = NP \Leftrightarrow SAT \in P$

- **Theorem** (Karp'72) $P = NP \Leftrightarrow TSP^* \in P$

*or Hamiltonian circuit or any of 19 other problems.

- **Theorem** (BSS '89) $P_R = NP_R \Leftrightarrow HN_R \in P_R$
  where $R = Z_2$ or the reals $\mathbb{R}$ or the complex #s $\mathbb{C}$ or....any field.

Proof* Given $(X, X_{yes}) \in NP_R$ and instance $x \in X$,

Code (in poly-time): $x \rightarrow f_x$ (poly system/R)

such that $x \in X_{yes} \Leftrightarrow f_x$ has a **zero** over R.

(*Poly-time Reduction)

- **Theorem** (Cook/Levin'71) $P = NP \Leftrightarrow SAT \in P$

- **Theorem** (Karp'72) $P = NP \Leftrightarrow TSP^* \in P$

*or Hamiltonian circuit or any of 19 other problems.

- **Theorem** (BSS '89) $P_R = NP_R \Leftrightarrow HN_R \in P_R$
  where $R = Z_2$ or the reals $\mathbb{R}$ or the complex #s $\mathbb{C}$ or....any field.

So, HN_R is Universal NP-complete Problem.

New Problem: Does $P_R = NP_R$?

Complexity and Real Computation

(BCSS, 98, Springer-Verlag)

Preface by **Dick Karp** (last paragraph):

"It is interesting to speculate as to whether the questions of *whether* $\mathbf{P_R = NP_R}$ and *whether* $\mathbf{P_C = NP_C}$ are related to each other and to the *classical* **P versus NP question**.

… I am inclined to think that the three questions are very different and need to be attacked independently. …"

But… Transfer Principles (BCSS)

Transfer (of **complexity** results) from one domain to another.

Theorem (BCSS) $P_{\mathbb{C}} = NP_{\mathbb{C}} \Leftrightarrow P_{\tilde{\mathbb{Q}}} = NP_{\tilde{\mathbb{Q}}} \Leftrightarrow P_K = NP_K$

(K is algebraically closed and of characteristic 0)

Theorem $P_{\mathbb{C}} = NP_{\mathbb{C}} \Rightarrow BPP \supseteq NP$

Pascal Koiran

(hidden in '93 paper) …

• **Transfer Results** provide important **connections** between the **two** approaches to computing.

Transfer Principles (BCSS)

Transfer (of **complexity** results) from one domain to another.

Theorem (BCSS) $P_{\mathbb{C}} = NP_{\mathbb{C}} \Leftrightarrow P_{\tilde{\mathbb{Q}}} = NP_{\tilde{\mathbb{Q}}} \Leftrightarrow P_K = NP_K$

(K is algebraically closed and of characteristic 0)

1.

Theorem $P_{\mathbb{C}} = NP_{\mathbb{C}} \Rightarrow BPP \supseteq NP$

Proof

2

1. $NP \subseteq$ BooleanPart($NP_{\mathbb{C}}$) via $x(x-1)$
2. $BP(P_{\mathbb{C}}) \subseteq BPP$ via coin tosses:

Can eliminate complex constants probabilistically with small number of small numbers (Use Schwartz's Lemma and Prime Number Theorem.)

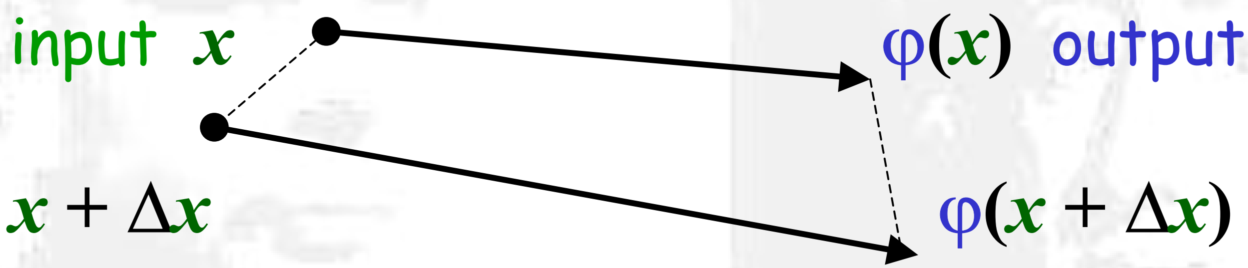
Introducing Condition and Round-off into ComplexityTheory

or where

Turing meets Newton!

Introducing Condition and Round-off into ComplexityTheory *or where* Turing meets Newton!

The **condition** of a problem (instance) measures how small perturbations of the **input** will alter the **output**.



$$\frac{\|\varphi(x + \Delta x) - \varphi(x)\|}{\|\Delta x\|} \quad \text{or } \textit{relative} \quad \frac{\|\varphi(x + \Delta x) - \varphi(x)\| / \|\varphi(x)\|}{\|\Delta x\| / \|x\|}$$

If quotient is large, instance is ill-conditioned so requires more accuracy and hence more resources to compute with small error.

Return again to…

THE QUARTERLY JOURNAL OF

MECHANICS AND APPLIED MATHEMATICS

***The Quarterly Journal of Mechanics and Applied Mathematics*, vol. I, 1948**

ROUNDING-OFF ERRORS IN MATRIX PROCESSES

By A. M. TURING

(*National Physical Laboratory, Teddington, Middlesex*)

[Received 4 November 1947]

SUMMARY

A number of methods of solving sets of linear equations and inverting matrice
e discussed. The theory of the rounding-off errors involved is investigated fo
me of the methods. In all cases examined, including the well-known 'Gaus
imination process', it is found that the errors are normally quite moderate: n
ponential build-up need occur.

Included amongst the methods considered is a generalization of Choleski's metho
hich appears to have advantages over other known methods both as regard
curacy and convenience. This method may also be regarded as a rearrangemen
the elimination process.

HIS paper contains descriptions of a number of methods for solving set
linear simultaneous equations and for inverting matrices, but its mai
ncern is with the theoretical limits of accuracy that may be obtained i
e application of these methods, due to rounding-off errors.

The best known method for the solution of linear equations is Gauss

$$(8.1)\qquad \begin{aligned} 1.4x + 0.9y &= 2.7 \\ -0.8x + 1.7y &= -1.2 \end{aligned}$$

$$(8.2)\qquad \begin{aligned} -0.786x + 1.709y &= -1.173 \\ -0.8\,x + 1.7\,y &= -1.2 \end{aligned}$$

The set of equations (8.2) is *fully equivalent* to (8.1), but clearly if we attempt to solve (8.2) by numerical methods involving rounding-off errors we are almost certain to get much less accuracy than if we worked with equations (8.1).

We should describe the equations (8.2) as an ***ill-conditioned*** set, or, at any rate, as ill-conditioned compared with (8.1). It is characteristic of ill-conditioned sets of equations that **small percentage errors in the coefficients given may lead to large percentage errors in the solution.**

p. 297

$$(8.1) \qquad \begin{aligned} 1.4x + 0.9y &= 2.7 \\ -0.8x + 1.7y &= -1.2 \end{aligned}$$

$$(8.2) \qquad \begin{aligned} -0.786x + 1.709y &= -1.173 \\ -0.8\,x + 1.7\,y &= -1.2 \end{aligned}$$

The set of equations (8.2) is *fully equivalent* to (8.1), but clearly if we attempt to solve (8.2) by numerical methods involving rounding-off errors we are almost certain to get much less accuracy than if we worked with equations (8.1).

3rd equation = .01(1st equation) + 2nd equation

Will illustrate reconciliation of 2 Traditions with

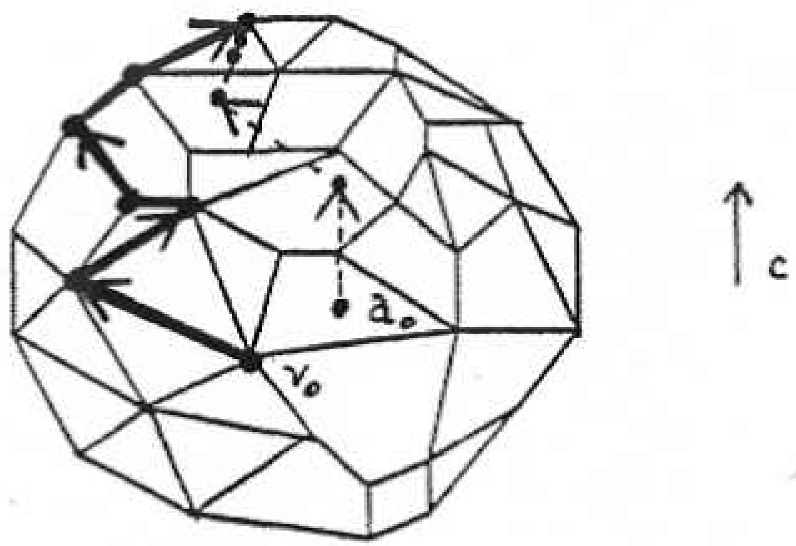
My Favorite Example

Linear Programming Problem

- max $c \cdot x$
- such that $Ax \leq b$
- $x \geq 0$

$c \in R^n$, $b \in R^m$

A, real m x n matrix



My Favorite Example

Linear Programming Problem

- max $c \cdot x$
- such that $Ax \leq b$
- $x \geq 0$

$c \in R^n$, $b \in R^m$

A, real m x n matrix

- Simplex Method (Dantzig '47)
  Exponential (n, m) (Klee-Minty '72)
- Ellipsoid Method (Khachiyan '79)
  Poly in Input Word size (in bits)
- Interior-Point Method (Karmarkar 84)
  *Polynomial (input word bit size)

*Recall, small perturbations (of input) can cause large differences in (input) WordSize:

$1 \sim 1+\frac{1}{2}^n$ but WordSize$(1)=1$ ~ WordSize$(1+\frac{1}{2}^n)=n+1$

My Favorite Example

Linear Programming Problem

- max $c \cdot x$
- such that $Ax \leq b$
- $x \geq 0$

$c \in R^n$, $b \in R^m$

A, real m x n matrix

- Simplex Method (Dantzig '47)

Exponential (n,m) (Klee-Minty '72)

- Ellipsoid Method (Khachiyan '79)

Poly in Input Word size (in bits)

- Interior-Point Method (Karmarkar 84)

*Polynomial (input word bit size)

Not paying attention to the distinction between these metrics has caused both an incompleteness in the analysis and a confusion in the comparison of different algorithms for the LPP.

Condition Numbers and Complexity

<u>**Linear Systems:**</u> **Given** $Ax=b$. **Solve** for **x**.

Turing: $\kappa(A) = ||A|| \; ||A^{-1}||$

- **Theorem** (Eckart-Young 1936). $\kappa(A) \sim 1/d_F(A, \Sigma)$

where Σ is space of *ill-posed* problem instances. i.e. Σ is the space of *non-invertible* matrices, and distance d_F is wrt the Frobenious norm.

Condition Numbers and Complexity

Linear Systems: **Given** $Ax=b$. **Solve** for x.

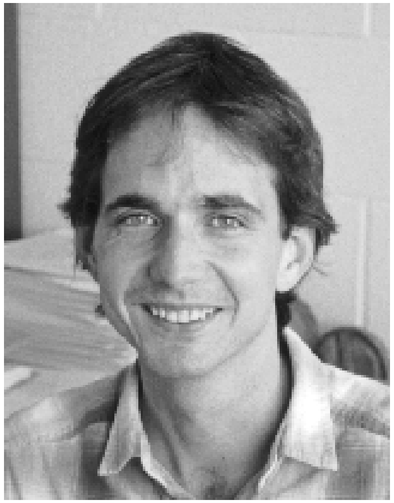
Turing: $\kappa(A) = ||A|| \, ||A^{-1}||$

•**Theorem** (Eckart-Young 1936). $\kappa(A) \sim 1/d_F(A, \Sigma)$

where Σ is space of *ill-posed* problem instances.

•**Linear Programming:** **Given** $Ax=b$, $x \geq 0$. **Solve** for x.

(Blum89) (Renegar95) $C(A, b)=||(A,b)||/d((A,b),\Sigma_{m,n})$



$\Sigma_{m,n}$ is the boundary of the space of feasible pairs (A,b)

Condition Numbers and Complexity

Linear Systems: **Given** $A x = b$. **Solve** for $\mathbf{x}$.

Turing: $\kappa(A) = \|A\| \, \|A^{-1}\|$

- **Theorem** (Eckart-Young 1936). $\kappa(A) \sim 1/d(A, \Sigma)$ where Σ is space of *ill-posed* problem instances.

- **Linear Programming:** **Given** $A x = b$, $\mathbf{x} \geq 0$. **Solve** for $\mathbf{x}$.
(Blum89)(**Renegar**95) $C(A, b) = \|(A,b)\| / d((A,b), \Sigma_{m,n})$

- **Theorem** (**Renegar** Interior Point Algorithm): If feasible, **#** of iterations to get ε-**approximation** of the optimal value is **poly** in **n**, **m**, **log** $C(A,b,c)$ and $|\log \varepsilon|$.

- **Theorem** (**Cucker-Pena** Algorithm with Round-Off): If feasible, produces δ-**approx** to a feasible point in (bit) time $O((m+n)^{3.5}(\log(m+n)+\log C(A)+|\log \delta|)^3)$. The finest precision required is a *round-off* unit of $1/c(m+n)^{12}C(A)^2$.

Computing over the Reals: Where Turing Meets Newton

~~The END~~