Algunos ejemplos de éxito en matemáticas computacionales

Carlos Beltrán

Coloquio UC3M
Leganés
4 de febrero de 2015

Solving linear systems

$Ax = b$ where A is $n \times n$, $\det(A) \neq 0$

Input: $A \in \mathcal{M}_n(\mathbb{C})$, $det(A) \neq 0$. $b \in \mathbb{C}^n$. **Output:** $x \in \mathbb{C}^n$ such that $Ax = b$. **Stability of the solution:** Relation between x and x' where:

$$Ax = b \qquad A'x' = b, \qquad A \approx A'.$$

Solving linear systems

$Ax = b$ where A is $n \times n$, $\det(A) \neq 0$

Input: $A \in \mathcal{M}_n(\mathbb{C})$, $det(A) \neq 0$. $b \in \mathbb{C}^n$. **Output:** $x \in \mathbb{C}^n$ such that $Ax = b$. **Stability of the solution:** Relation between x and x' where:

$$Ax = b \qquad A'x' = b, \qquad A \approx A'.$$

Condition number: $\kappa(A) := \|A\|_2 \|A^{-1}\|_2$ [Turing].

$$\frac{\|x - x'\|}{\|x'\|} \leq \kappa(A) \frac{\|A - A'\|_2}{\|A\|_2}.$$

Alternatively: $\kappa_D(A) := \|A\|_F \|A^{-1}\|_2$ [Smale, Demmel].

Solving linear systems

$Ax = b$ where A is $n \times n$, $\det(A) \neq 0$

Input: $A \in \mathcal{M}_n(\mathbb{C})$, $det(A) \neq 0$. $b \in \mathbb{C}^n$. **Output:** $x \in \mathbb{C}^n$ such that $Ax = b$. **Stability of the solution:** Relation between x and x' where:

$$Ax = b \qquad A'x' = b, \qquad A \approx A'.$$

Condition number: $\kappa(A) := \|A\|_2 \|A^{-1}\|_2$ [Turing].

$$\frac{\|x - x'\|}{\|x'\|} \leq \kappa(A) \frac{\|A - A'\|_2}{\|A\|_2}.$$

Alternatively: $\kappa_D(A) := \|A\|_F \|A^{-1}\|_2$ [Smale, Demmel].
Computing κ is more difficult than computing x.

Solving linear systems

$Ax = b$ where A is $n \times n$, $\det(A) \neq 0$

Input: $A \in \mathcal{M}_n(\mathbb{C})$, $det(A) \neq 0$. $b \in \mathbb{C}^n$. **Output:** $x \in \mathbb{C}^n$ such that $Ax = b$. **Stability of the solution:** Relation between x and x' where:

$$Ax = b \qquad A'x' = b, \qquad A \approx A'.$$

Condition number: $\kappa(A) := \|A\|_2 \|A^{-1}\|_2$ [Turing].

$$\frac{\|x - x'\|}{\|x'\|} \leq \kappa(A) \frac{\|A - A'\|_2}{\|A\|_2}.$$

Alternatively: $\kappa_D(A) := \|A\|_F \|A^{-1}\|_2$ [Smale, Demmel].
Computing κ is more difficult than computing x. **What can we hope? Will $\kappa(A)$ be "in general" a small quantity?**

A geometric property of κ_D

[Eckardt, Young, Smicht, Mirski, and even Banach (?)]'s theorem

We note that $\kappa_D(A)$ depends only on the projective class of A.

Let Σ^{n-1} be the set of all singular matrices of $\mathbb{P}(\mathcal{M}_n(\mathbb{C}))$ (which is an algebraic variety.)

Theorem

$$\kappa_D(A) = \frac{1}{d_{\mathbb{P}(\mathcal{M}_n(\mathbb{C}))}(A, \Sigma^{n-1})}.$$

A geometric property of κ_D

[Eckardt, Young, Smicht, Mirski, and even Banach (?)]'s theorem

We note that $\kappa_D(A)$ depends only on the projective class of A.

Let Σ^{n-1} be the set of all singular matrices of $\mathbb{P}(\mathcal{M}_n(\mathbb{C}))$ (which is an algebraic variety.)

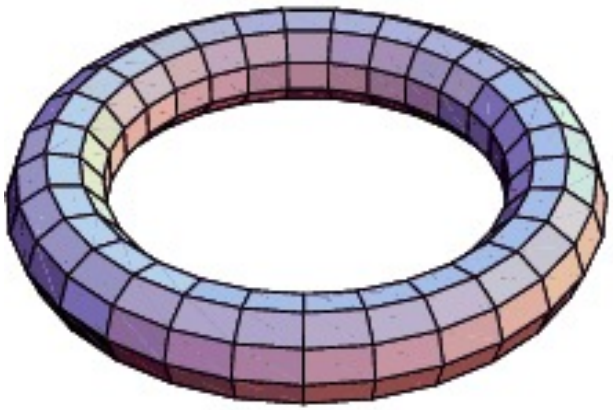
Theorem

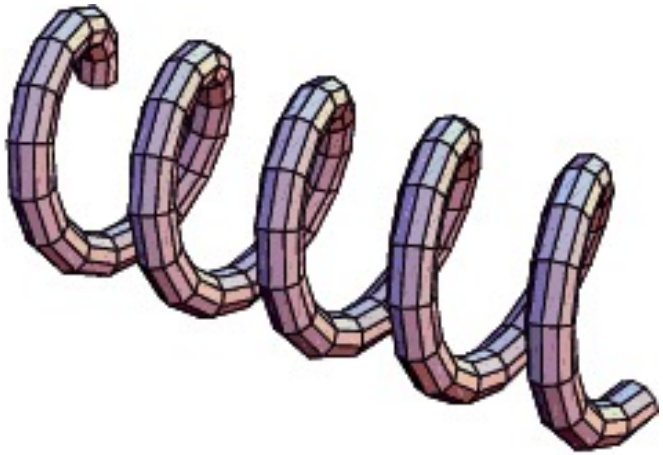
$$\kappa_D(A) = \frac{1}{d_{\mathbb{P}(\mathcal{M}_n(\mathbb{C}))}(A, \Sigma^{n-1})}.$$

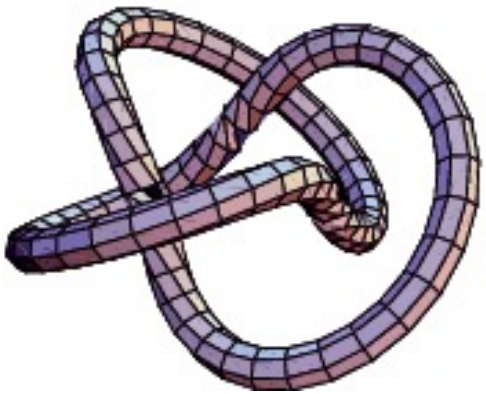
The probability that the condition number is big... equals the probability of being close to Σ^{n-1}.

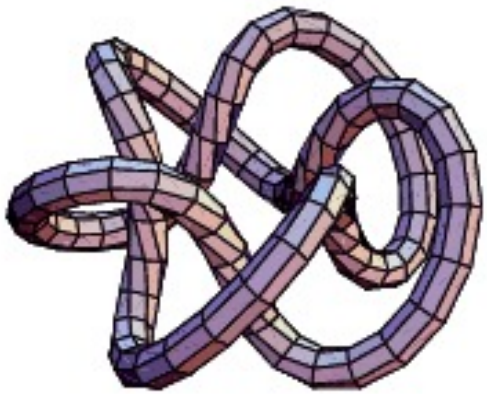
The volume of tubes

The classical result by Weyl (Gray for the projective version) is only valid for smooth varieties and small radius... not our case



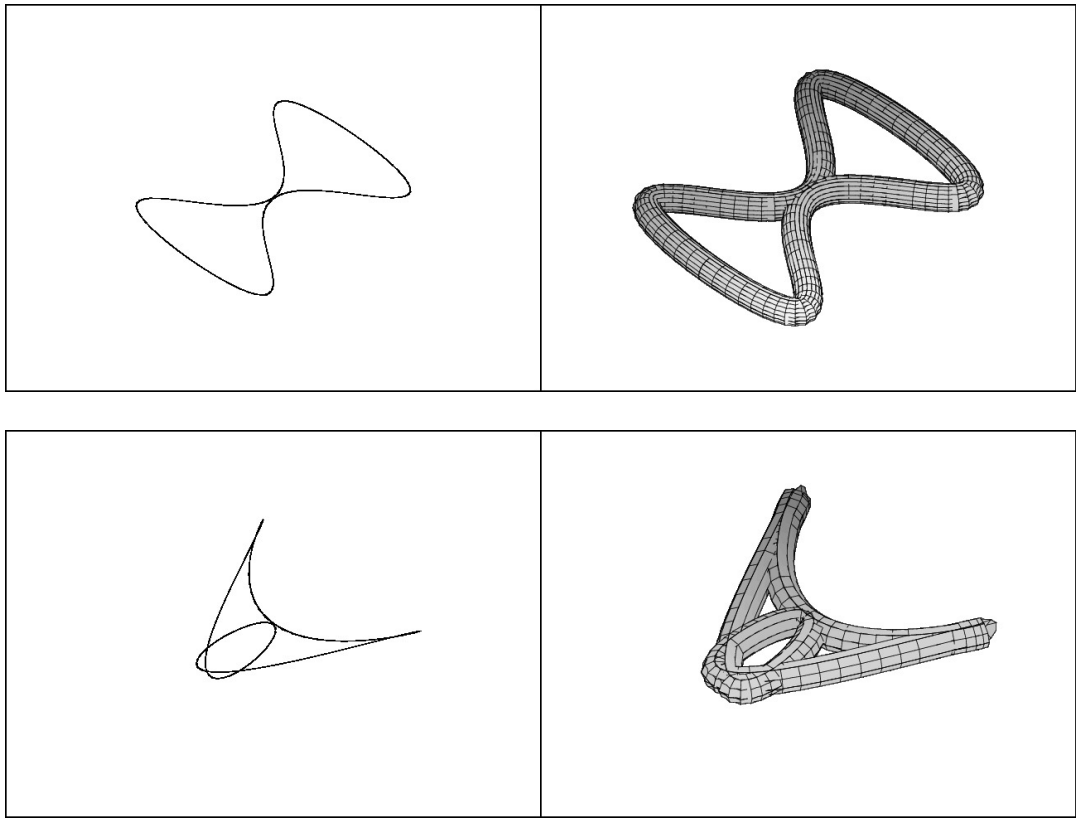






The volume of tubes

The classical result by Weyl (Gray for the projective version) is only valid for smooth varieties and small radius... not our case. But there *is* a way to do it.

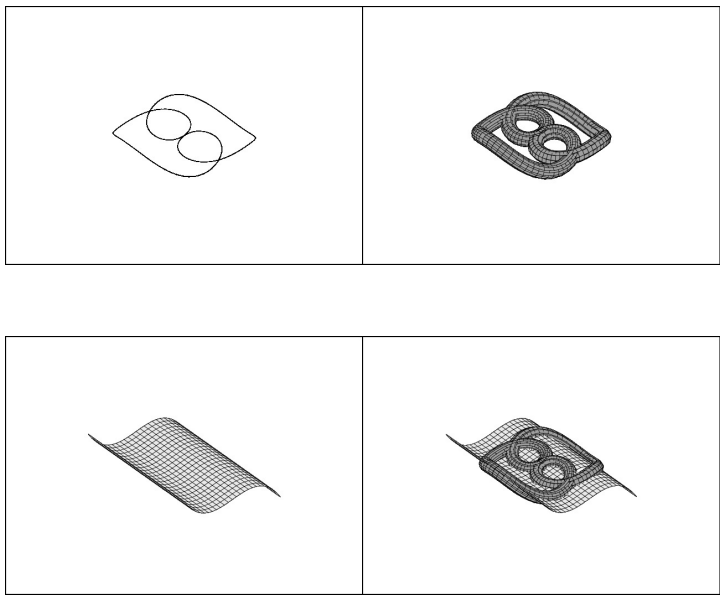


This approach produced the first theoretical statistical study of the condition number. Smale, Renegar, Demmel and others.

Computing kernels

This was probably first observed by Kahan. Let A of size $m \times n$ and $rank(A) = s$

The condition number of A for the problem of computing the subspace such that $Ax = 0$ is the inverse of the distance to the set of one–rank–less matrices.



[B.,Pardo]: a first technique for bounding studying the case $\mathbb{C}$.

The estate of the art in the probabilistic estimation of the condition number of matrices

Edelman, Shub, Sutton, Chen, Dongarra, B.

Real A of size $m \times n$ and rank r. Then,

$$\mathrm{P}\left[\kappa(A) > t\right] = C(m,n,r)t^{-(n+m-2r+1)},$$

where $C(m,n,r)$ is known up to a small constant (and has a simple formula). A similar thing is known for complex matrices.

How much precision should we use for solving "random" problems $Ax = b$ where A is an $m \times n$ matrix of rank r?

Edelman, Shub, Sutton, Chen, Dongarra, B.

$$\mathrm{E}[\log(\kappa(A)) : A \in \Sigma] \leq \log \frac{n+m-r}{n+m-2r+1} + 2.6.$$

How much precision should we use for solving "random" problems $Ax = b$ where A is an $m \times n$ matrix of rank r?

Edelman, Shub, Sutton, Chen, Dongarra, B.

$$\mathrm{E}[\log(\kappa(A)) : A \in \Sigma] \leq \log \frac{n+m-r}{n+m-2r+1} + 2.6.$$

For example, for $10^6 \times 10^6 + 1$ matrices of rank $10^6 - 200$ that quantity is 6 so you should be using at least 8 decimal digits of precision, at least in the output. **This is doable in our double precision IEEE standard.**

How much precision should we use for solving "random" problems $Ax = b$ where A is an $m \times n$ matrix of rank r?

Edelman, Shub, Sutton, Chen, Dongarra, B.

$$\mathrm{E}[\log(\kappa(A)) : A \in \Sigma] \leq \log \frac{n+m-r}{n+m-2r+1} + 2.6.$$

For example, for $10^6 \times 10^6 + 1$ matrices of rank $10^6 - 200$ that quantity is 6 so you should be using at least 8 decimal digits of precision, at least in the output. **This is doable in our double precision IEEE standard.**
But for $10^{14} \times 10^{14}$ matrices of rank $10^{14} - 1$, that quantity is around 17. Too much precision for our machines, right?

How much precision should we use for solving "random" problems $Ax = b$ where A is an $m \times n$ matrix of rank r?

Edelman, Shub, Sutton, Chen, Dongarra, B.

$$\mathrm{E}[\log(\kappa(A)) : A \in \Sigma] \leq \log \frac{n+m-r}{n+m-2r+1} + 2.6.$$

For example, for $10^6 \times 10^6 + 1$ matrices of rank $10^6 - 200$ that quantity is 6 so you should be using at least 8 decimal digits of precision, at least in the output. **This is doable in our double precision IEEE standard.**
But for $10^{14} \times 10^{14}$ matrices of rank $10^{14} - 1$, that quantity is around 17. Too much precision for our machines, right?
And for $10^{20} \times 10^{25}$ matrices of rank 10^{19}?

How much precision should we use for solving "random" problems $Ax = b$ where A is an $m \times n$ matrix of rank r?

Edelman, Shub, Sutton, Chen, Dongarra, B.

$$\mathrm{E}[\log(\kappa(A)) : A \in \Sigma] \leq \log \frac{n+m-r}{n+m-2r+1} + 2.6.$$

For example, for $10^6 \times 10^6 + 1$ matrices of rank $10^6 - 200$ that quantity is 6 so you should be using at least 8 decimal digits of precision, at least in the output. **This is doable in our double precision IEEE standard.**
But for $10^{14} \times 10^{14}$ matrices of rank $10^{14} - 1$, that quantity is around 17. Too much precision for our machines, right?
And for $10^{20} \times 10^{25}$ matrices of rank 10^{19}?The value of this quantity is less than 3. So, that is doable!

Further

That was just a very particular thing!

1. Smooth analysis of the condition number (Spielman & Chen) has been expanded in different directions with similar techniques (Burgisser & Cucker and others).
2. Average results for other problems as linear optimization, polynomial system solving etc. also exist (Smale, Renegar, Shub, B. & Pardo and others).
3. Other probability distributions have been analysed with some success (Tao & Vu and others).
4. Many open questions. For example, average of the condition number when the entries are uniformly distributed in $[-1, 1]$ or in $\{-1, 1\}$.
5. Do you want a paper in Annals of Math.? Prove a version for sparse matrices.

Polynomial systems.

$f = (f_1, \ldots, f_n)$ where $f_i \in \mathbb{C}[X_1, \ldots, X_n]$, $\deg(f_i) = d_i$ is a system of equations.
What do we do to find its solution set $V(f)$?

An example with Groebner basis

Text Math | 2D Input | Times New Roman | 12 | B I U

```
> with(Groebner)
[Basis, FGLM, HilbertDimension, HilbertPolynomial, HilbertSeries, Homogenize, InitialForm, InterReduce, IsProper, IsZeroDimensional,
    LeadingCoefficient, LeadingMonomial, LeadingTerm, MatrixOrder, MaximalIndependentSet, MonomialOrder, MultiplicationMatrix,
    MultivariateCyclicVector, NormalForm, NormalSet, RationalUnivariateRepresentation, Reduce, RememberBasis, SPolynomial, Solve,
    SuggestVariableOrder, TestOrder, ToricIdealBasis, TrailingTerm, UnivariatePolynomial, Walk, WeightedDegree]

> F := [y - 4·x + 3·x·y, x·y - y + 3·y^2 - y^3]
```

$$F := [y - 4x + 3xy,\ xy - y + 3y^2 - y^3]$$ — To solve

```
> B := Basis(F, plex(x, y, z))
```

$$B := [-4y + 16y^2 - 13y^3 + 3y^4,\ -3y^3 + 9y^2 + 4x - 4y]$$ — Groebner basis

```
> L := [solve(B[1], y)]
```

$$L := \left[0, \frac{1}{3}, 2, 2\right]$$ — Solutions of first poly of Groeber basis

```
> for i from 1 to nops(L) do [L[i], simplify(subs(y = L[i], -2y^2 + y^5))] od
```

$[0, 0]$

$\left[\frac{1}{3}, -\frac{53}{243}\right]$

$[2, 24]$

$[2, 24]$

→ Solutions of the original system

Another example with Groebner basis

Text Math | 2D Input | Times New Roman | 12 | B I U

```
> with(Groebner)
```

[Basis, FGLM, HilbertDimension, HilbertPolynomial, HilbertSeries, Homogenize, InitialForm, InterReduce, IsProper, IsZeroDimensional, LeadingCoefficient, LeadingMonomial, LeadingTerm, MatrixOrder, MaximalIndependentSet, MonomialOrder, MultiplicationMatrix, MultivariateCyclicVector, NormalForm, NormalSet, RationalUnivariateRepresentation, Reduce, RememberBasis, SPolynomial, Solve, SuggestVariableOrder, TestOrder, ToricIdealBasis, TrailingTerm, UnivariatePolynomial, Walk, WeightedDegree]

```
> F := [y - 4·x + 3·x·y, x^3 + x·y - y + 3·y^2 - y^3]
```

$$F := [y - 4x + 3xy,\ x^3 + xy - y + 3y^2 - y^3]$$

```
> B := Basis(F, plex(x, y, z))
```

$$B := [-64y + 352y^2 - 627y^3 + 504y^4 - 189y^5 + 27y^6,\ -148y + 681y^2 + 16x - 900y^3 + 459y^4 - 81y^5]$$

```
> L := [solve(B[1], y)]
```

$$L := [0, RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 1), RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z$$
$$index = 2), RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 3), RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 -$$
$$+ 27\_Z^5, index = 4), RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 5)]$$

```
> for i from 1 to 2 do [L[i], simplify(subs(y = L[i], -2y^2 + y^5))] od
```

$$[0, 0]$$

$$\left[RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 1),\ \frac{191}{9}\ RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 1)^2 + \frac{64}{27} - \frac{352}{27}\ RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 1) - \frac{56}{3}\ RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 1)^3 + 7\ RootOf(-64 + 352\_Z - 627\_Z^2 + 504\_Z^3 - 189\_Z^4 + 27\_Z^5, index = 1)^4\right]$$

Ask for less information and you might be faster

A much more modest question: can we approximate just one root, but guaranteeing polynomial running time?

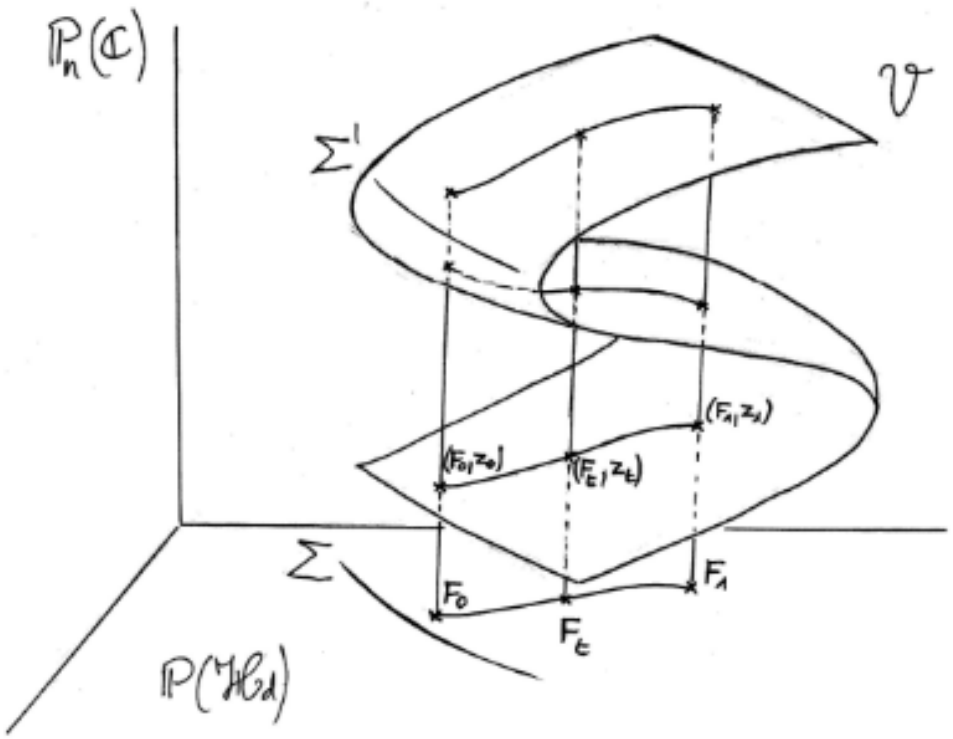
Ask for less information and you might be faster

A much more modest question: can we approximate just one root, but guaranteeing polynomial running time?
Smale's 17th Problem:

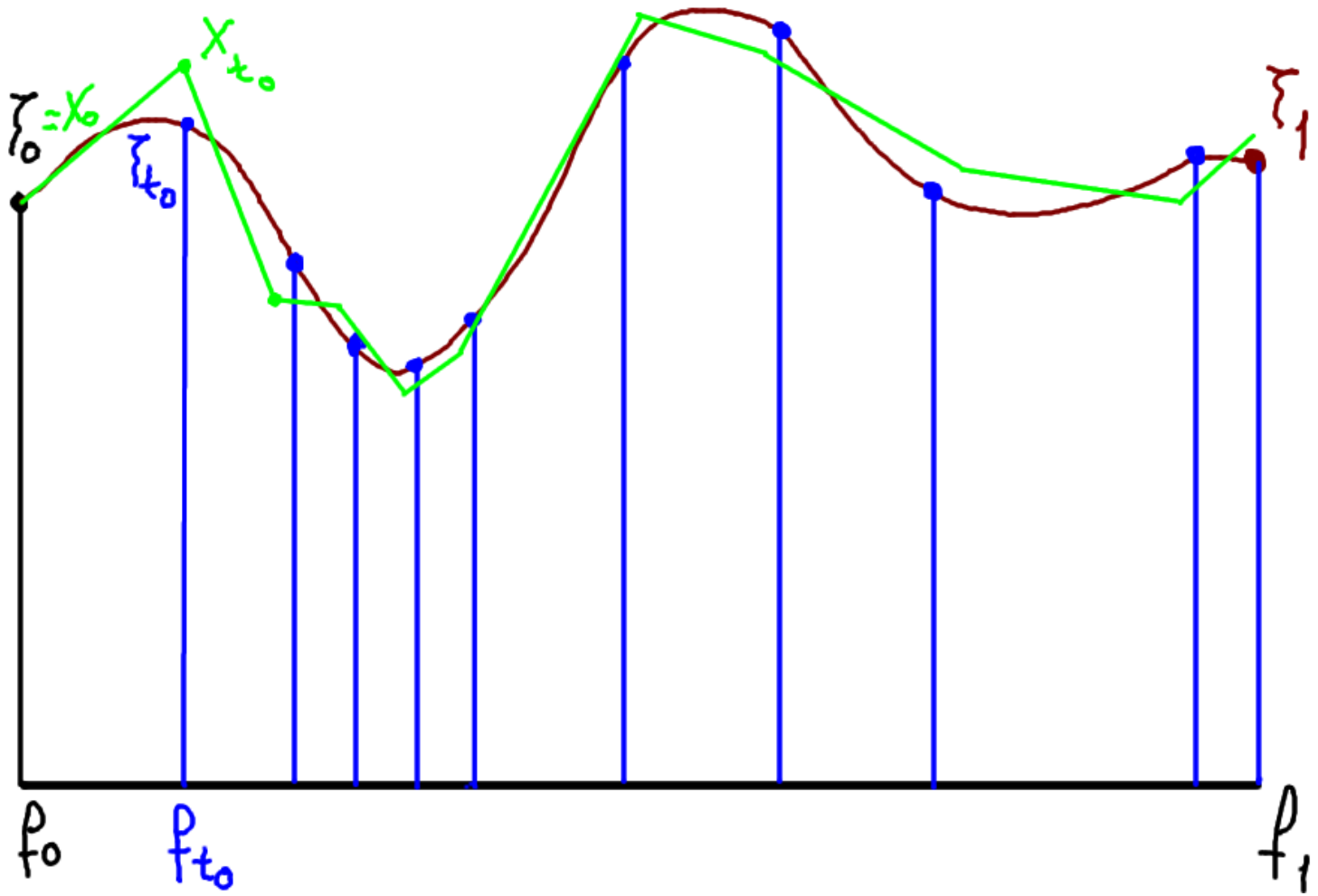
Can a zero of n complex polynomial equations in n unknowns be found approximately, on the average, in polynomial time with a uniform algorithm?

Stephen Smale, *Mathematical problems for the next century*.
Mathematics: frontiers and perspectives.
American Mathematical Society, 2000.

Homotopy methods



Homotopy method



Homotopy method

Homotopy method

- Convergence results for Newton's Method (Kantorovich, Shub & Smale, Dedieu & Malajovich, Wang & Hang).
- Precise statements about the length of the homotopy steps by Shub & Smale.
- Required precision in operations and best choice of Newton's method by Malajovich.
- Different techniques to take advantage of sparsity, detect singularities...
- Software, theory and heuristics by Sommerse, Vampler, Verschelde, Li.
- Enormous systems are solved this way.

Approximate zero

Let f be a system and ζ a projective zero of f. Let z be a projective point. We say that z is an approximate zero of f with associated zero ζ if for every $k \geq 0$

$$distance(N_f^k(z), \zeta) \leq \frac{1}{2^{2^k}} distance(z, \zeta),$$

where N_f^k is the result of applying k times the Newton iteration.

Condition number $\mu(f,\zeta)$ and approximate zeros

Let f be a system with a zero ζ and let z be a projective point. Assume that

$$distance(z,\zeta) \leq \frac{3-\sqrt{7}}{2d^{3/2}\mu(f,\zeta)}.$$

Then, z is an approximate zero of f with associate zero ζ.

Condition number $\mu(f,\zeta)$ and approximate zeros

Let f be a system with a zero ζ and let z be a projective point. Assume that

$$distance(z,\zeta) \leq \frac{3-\sqrt{7}}{2d^{3/2}\mu(f,\zeta)}.$$

Then, z is an approximate zero of f with associate zero ζ.Here,

$$\mu(f,\zeta) = \|Diag(d_1,\ldots,d_n)\left(Df(\zeta)\mid_{\zeta^\perp}\right)^{-1}\|$$

is the **condition number** for polynomial system solving.

Condition number $\mu(f,\zeta)$ and approximate zeros

Let f be a system with a zero ζ and let z be a projective point. Assume that

$$distance(z,\zeta) \leq \frac{3-\sqrt{7}}{2d^{3/2}\mu(f,\zeta)}.$$

Then, z is an approximate zero of f with associate zero ζ.Here,

$$\mu(f,\zeta) = \|Diag(d_1,\ldots,d_n)\left(Df(\zeta)\mid_{\zeta^\perp}\right)^{-1}\|$$

is the **condition number** for polynomial system solving.Shub & Smale, Beltrán & Pardo:

$$\mathrm{E}_{f\in\mathbb{P}(\mathcal{H}_{(d)}),\zeta:f(\zeta)=0}(\mu(f,\zeta)^2) \leq nN.$$

Condition number and number of homotopy steps

[Shub, B., B. & Leykin]
The number of Newton homotopy steps necessary to follow a homotopy path $\Gamma_t = (f_t, \zeta_t),\ 0 \leq t \leq 1$ is bounded by

$$\textit{Constant}\ d^{3/2} \int_0^1 \mu(f_t, \zeta_t) \|(\dot{f}_t, \dot{\zeta}_t)\|\ dt$$

The algorithm which performs this task is included in the Numerical Algebraic Geometry package of Macaulay 2, NAG4M2.

Three ways to choose the initial pair (f_0, ζ_0):

1) Choose (f_0, ζ_0) at random, which guarantees average number of Newton steps $O(nN)$.
2) Use the "most simple" ie best conditioned (system,root) pair:

$$g = \begin{cases} d_1^{\frac{1}{2}} X_0^{d_1-1} X_1 = 0, \\ \cdots \\ d_n^{\frac{1}{2}} X_0^{d_n-1} X_n = 0, \end{cases} \qquad e_0 = (1, 0, \ldots, 0)$$

Conjectured by [Shub & Smale] to be "good".
3) Burgisser and Cucker proved time $O(N^{\log\log N})$ for:

$$h = \begin{cases} X_0^{d_1} - X_1^{d_1} = 0, \\ \cdots \\ X_0^{d_n} - X_n^{d_n} = 0, \end{cases} \qquad e_0 = (1, 1, \ldots, 1)$$

Experiments (Beltrán and Leykin, 2012) suggest 2) is best.

Open questions

- Prove that for some fixed choice of initial pair the running time is polynomial in N. Deterministic Smale's 17th problem.
- Find a polynomial f of degree N such that $\mu(f, \zeta) \leq poly(N)$ for all of its roots. Related to Smale's 7th problem.

Everybody knows...

that it is easy to compute eigenvalues of huge matrices. Is this actually true?

Everybody knows...

that it is easy to compute eigenvalues of huge matrices. Is this actually true?

- The unshifted QR algorithm terminates with probability 1 but is probably infinite average cost if approximations to the eigenvectors are to be output Kostlan, 1988).
- The QR algorithm with Rayleigh Quotient shift fails for open sets of real input matrices (Batterson & Smillie1989).
- We do not know whether the Francis (double) shift algorithm converges generally on real or complex matrices, nor an estimate of its average cost.
- Other algorithms in modern texts are analyzed but avoiding to estimate the (necessarily infinite in the worst case) number of iterations, which usually relies on experimental results.

Everybody knows...

that it is easy to compute eigenvalues of huge matrices. Is this actually true?

- The unshifted QR algorithm terminates with probability 1 but is probably infinite average cost if approximations to the eigenvectors are to be output Kostlan, 1988).
- The QR algorithm with Rayleigh Quotient shift fails for open sets of real input matrices (Batterson & Smillie1989).
- We do not know whether the Francis (double) shift algorithm converges generally on real or complex matrices, nor an estimate of its average cost.
- Other algorithms in modern texts are analyzed but avoiding to estimate the (necessarily infinite in the worst case) number of iterations, which usually relies on experimental results.

So, **is it really easy to compute eigenvalues of matrices? If so... prove it! An old question by Jim Demmel**

With eigenvector is it worse

Any pair v, w of orthogonal vectors is the set of eigenvectors of some arbitrary

$$\begin{pmatrix} 1+\epsilon_1 & \epsilon_3 \\ \epsilon_3 & 1+\epsilon_2 \end{pmatrix}$$

for ϵ_i arbitrarily small.
So, we have two problems:

- Number of iterations of iterative algorithms.
- Precision of the input and output.

Similar situation to the polynomial system case

Condition number for eigenvalue and eigenvector (classical, revisited by Armentano recently)

$$\mu_\lambda = \frac{\|u\| \cdot \|v\|}{|\langle u, v\rangle|}, \quad \mu_v = \|A\|\|(\pi_{v^\perp}(\lambda I - A)\pi_{v^\perp})^{-1}\|.$$

Similar situation to the polynomial system case

Condition number for eigenvalue and eigenvector (classical, revisited by Armentano recently)

$$\mu_\lambda = \frac{\|u\| \cdot \|v\|}{|\langle u, v\rangle|}, \quad \mu_v = \|A\|\|(\pi_{v^\perp}(\lambda I - A)\pi_{v^\perp})^{-1}\|.$$

Similarly as before (Armentano, B., Burgisser, Cucker, Shub):

- If (ζ, w) is *constant*$/\mu_v$ close to an actual eigenpair... then Newton method converges quadratically.

Similar situation to the polynomial system case

Condition number for eigenvalue and eigenvector (classical, revisited by Armentano recently)

$$\mu_\lambda = \frac{\|u\| \cdot \|v\|}{|\langle u, v\rangle|}, \quad \mu_v = \|A\|\|(\pi_{v^\perp}(\lambda I - A)\pi_{v^\perp})^{-1}\|.$$

Similarly as before (Armentano, B., Burgisser, Cucker, Shub):

- If (ζ, w) is *constant*$/\mu_v$ close to an actual eigenpair... then Newton method converges quadratically.
- The homotopy method for continuing (A_0, λ_0, v_0) to (A, λ, v) requires a number of steps:

$$\textit{Constant} \cdot \int_0^1 \mu_v(A_t, \lambda_t, v_t)\|(\dot{A}_t, \dot{\lambda}_t, \dot{v}_t)\|\, dt.$$

So, μ_v controls both precision of output and running time.

Similar situation to the polynomial system case

Moreover,

$$\mathrm{E}_{A\in\mathbb{SC}^{n\times n},\lambda,v:Av=\lambda v}(\mu_v(A,\lambda,v)^2)\leq n^3.$$

Conclusion:

- Computing all eigenpairs has average running time at most $O(n^9)$ with a deterministic algorithm.
- Computing one eigenpair has average running time at most $O(n^7)$ with a probabilistic algorithm.

This answers Demmel's old question.

Open questions

- Combine QR and homotopy/Newton to get better complexity bounds.
- Find a matrix of order n such that $\max(\mu(A, \lambda, v))$ is minimized. Packing problem possibly related to Smale's 7th problem.

Thank you