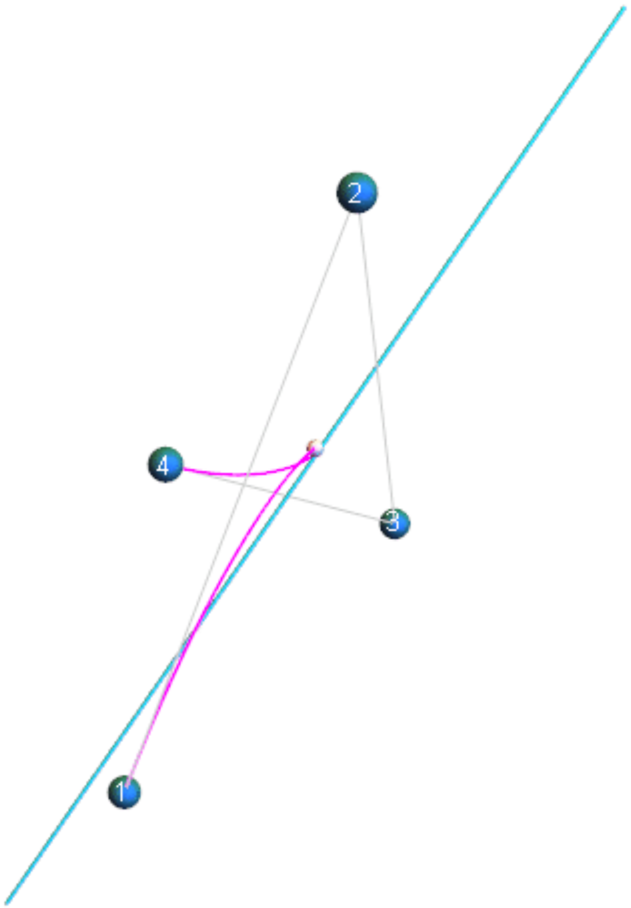
Exploiting Budan-Fourier and Vincent's Theorems for Ray Tracing 3D Bézier Curves

Alex Reshetov
NVIDIA



High-Performance Graphics 2017
Los Angeles | July 28–30, 2017

Why now

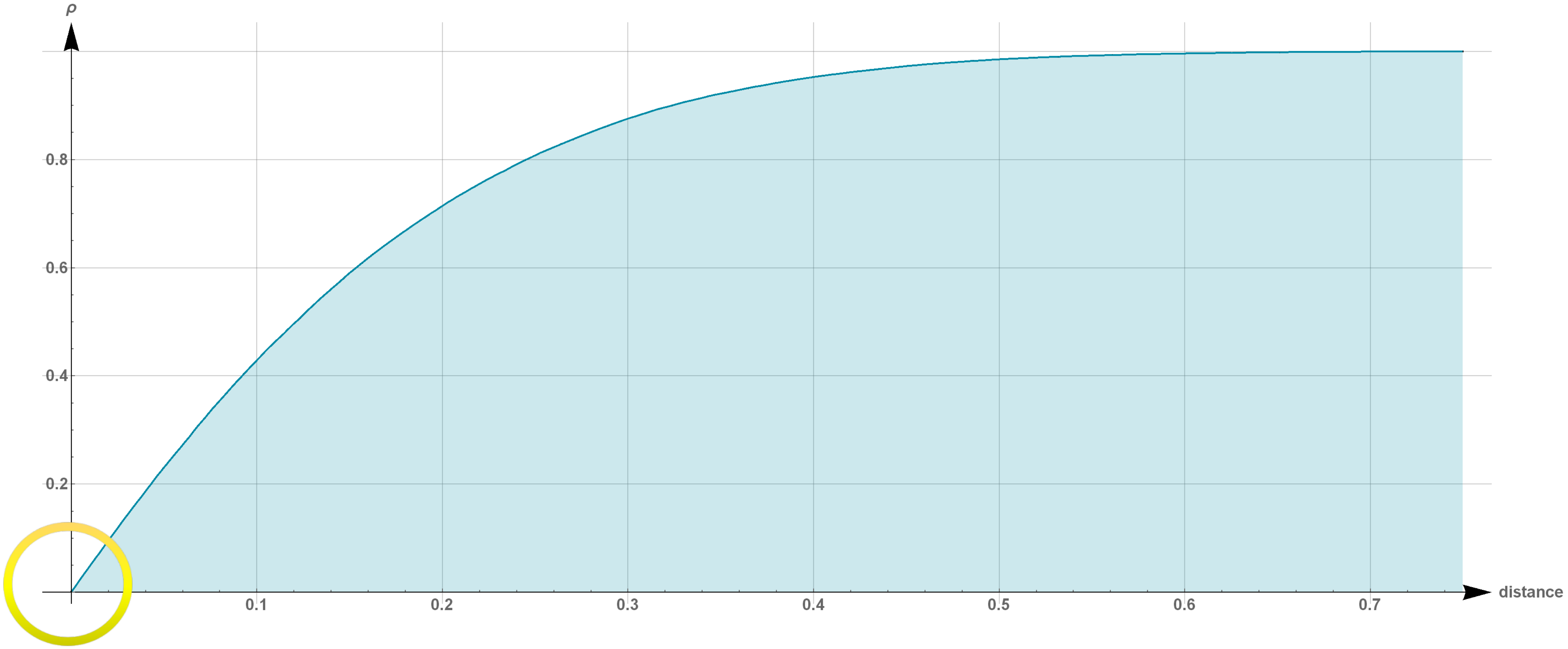
- $\exists$ need
  - Hair, grass, vegetation, contours, etc
  - Linear approximations or alpha blending are running its course
- Significant progress in studies of polynomials since Galois' last letter
  - Vincent–Akritas–Strzeboński (VAS, 2005) is used by Mathematica, Sage, etc
  - Math is mostly elementary
  - Far-reaching similarities between Bernstein polynomials and root-isolation techniques (not fully recognized yet)

What

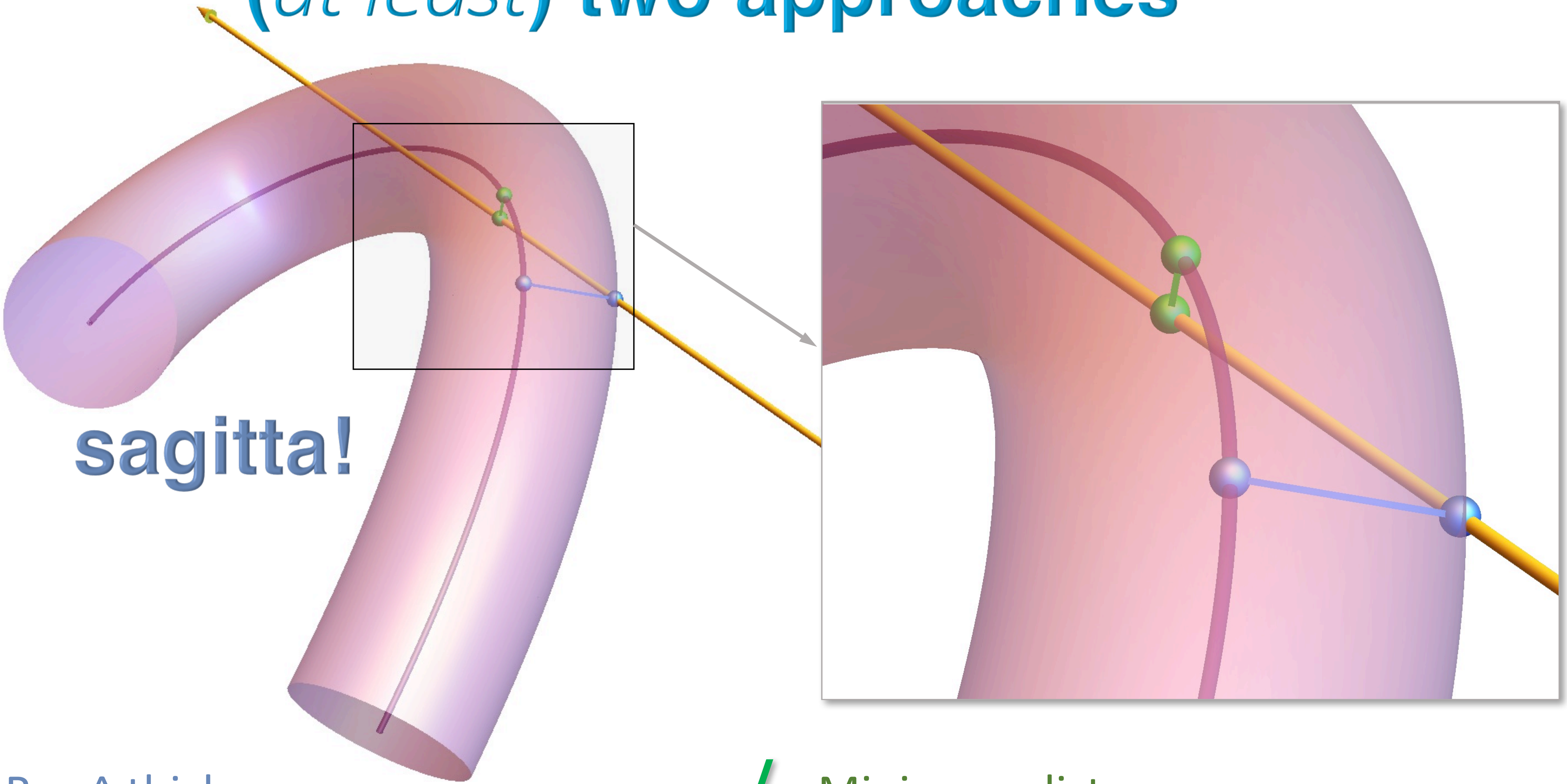
- Ray tracing cubic Bézier curves
  - i.e. finding curve's t for ray/curve intersection(s)
  - Rasterization is quite different ($t \rightarrow$ geometry)
  - Degree 3 is sufficient
    - It is expressive enough (inflection points & smooth connections)
    - Longer curves will be chopped into pieces anyway by acceleration structures
- Will *not* talk about
  - Building acceleration structures
  - Shading

What intersection?

probability of a ray to be closer to a curve than a given distance



(*at least*) two approaches



- Ray ^ thick curve
  more thrilling

√ Minimum distance
more universal (and easier)

Why *t*?

It looks simple…

Why *t*?

…but it isn’t

- $t \rightarrow$ geometry is easy
- geometry $\rightarrow t$ is not
- Yet we have to recover parameters
  to keep everything in an analytical form
  - normals
  - texture maps
  - etc

These approximate 2D methods will not work

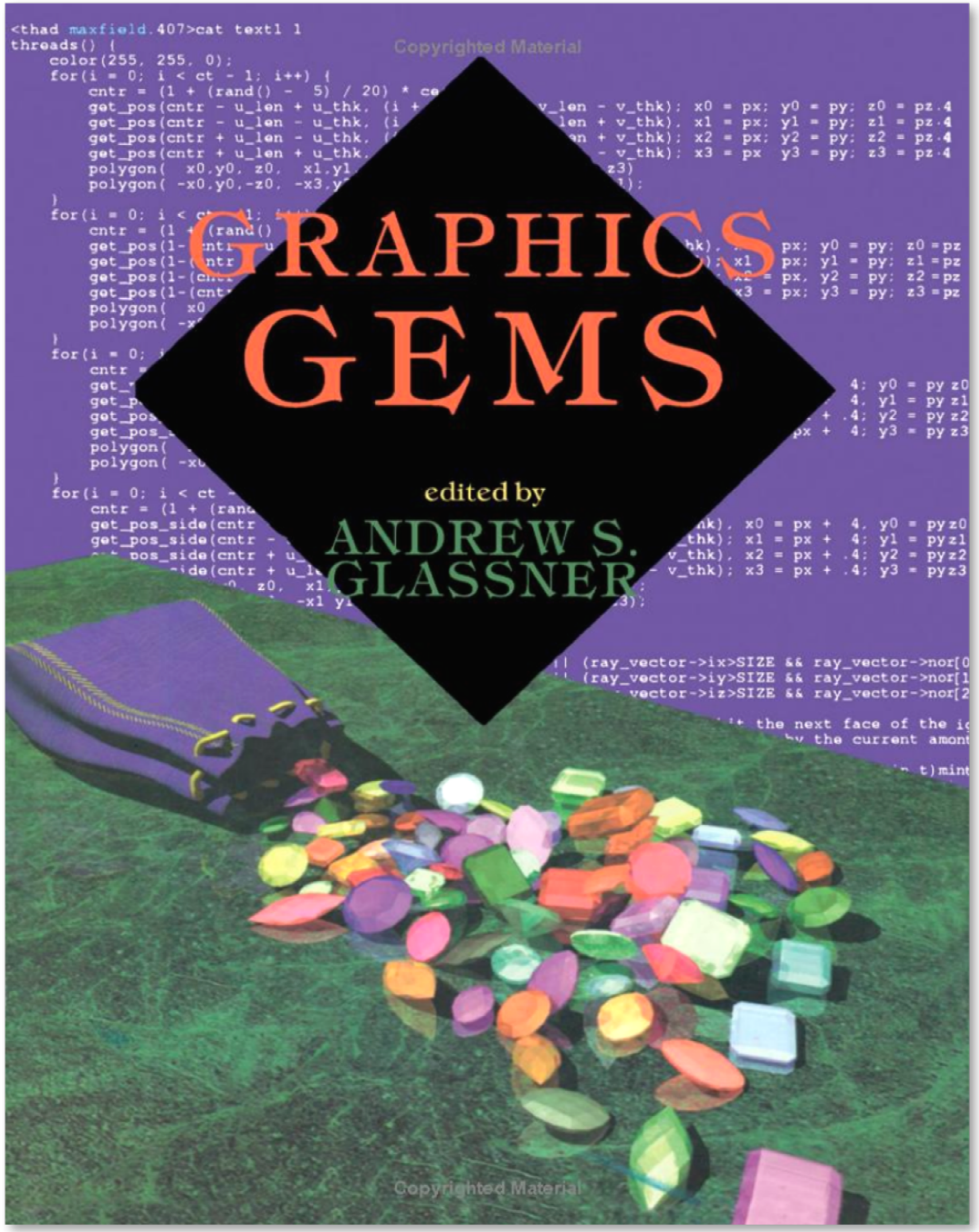
Batra et al. 2015;
Ganacim et al. 2014;
Kilgard and Bolz 2012;
Liao et al. 2012;
Loop and Blinn 2005;
Nehab and Hoppe 2008, 2012;
Qin et al. 2008;
Ray et al. 2005;
Reshetov and Luebke 2016;
Sen 2004;
Sun et al. 2012;
Tarini and Cignoni 2005

not because it is 2D,

these methods will not produce

a (good) curve's parameter ***t***

Project's inspiration



- many excellent articles on parametric representation of geometry
- Once you use a parametric form, everything is "equation solving" (either explicitly or implicitly)
- Hook and McAree [1990] introduced (to graphics community) Sturm sequences for solving equations

(my understanding of Sturm sequences)

DAVE DORMAN

State-of-the-art: adaptive linearization

Sederberg and Nishita [1990];

Nakamaru and Ohno [2002];

Barringer et al [2012];

Qin et al [2014];

Woop et al [2014];

Chiang at al [2015]

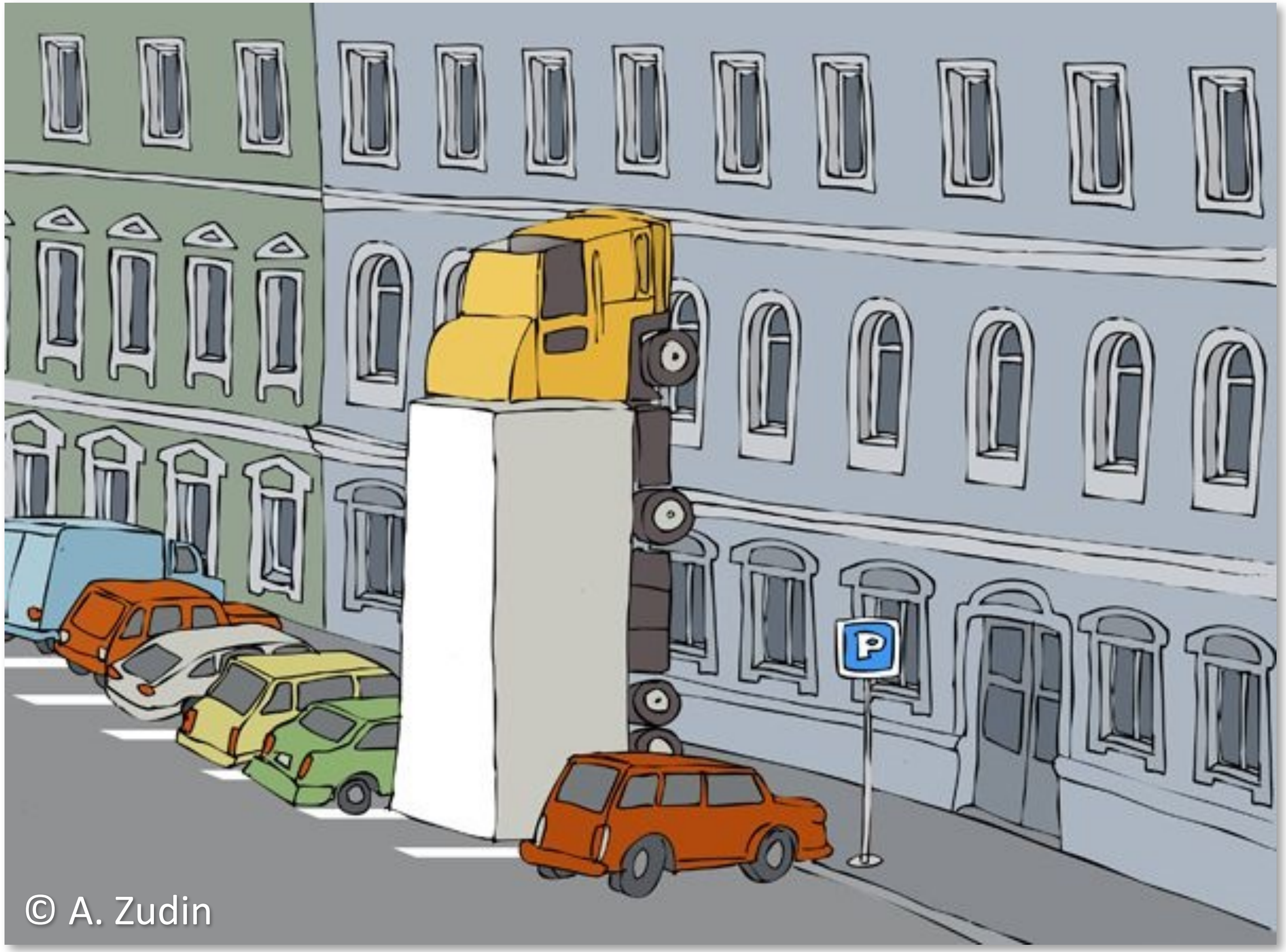
- Just chop the curve into pieces until it can be safely approximated with a line
- “Chopping” is done by (recursively) splitting the curve in half during rendering
- Works best for almost-linear curves; the curves are pre-split by an acceleration structure anyway

Why state-of-the-art is *sm*art

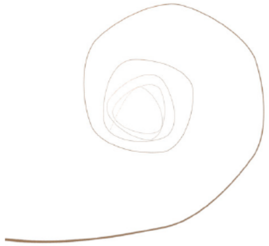
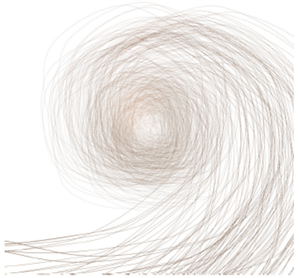
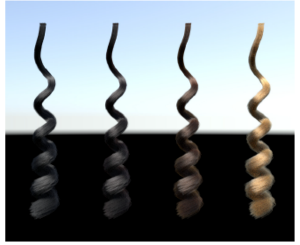
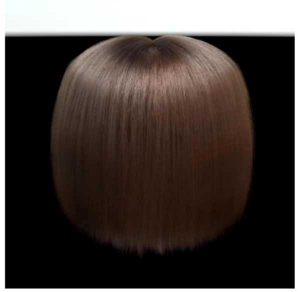
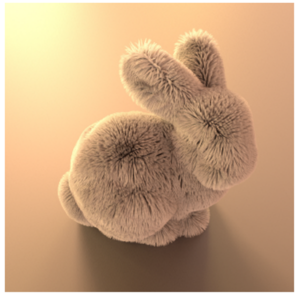
- It is effectively a bisection search of polynomial roots (corresponding to the minimum distance) while simultaneously
- isolating roots of the equation.
- Splitting in half can be done efficiently by using curve’s control points

- Alternatives (~Hook and McAree) are more challenging:
  - Computing Sturm sequences requires a long polynomial division
  - Traditional root-finding techniques (Newton-Raphson, secant, etc) have a lot of issues due to the target function non-linearity

Proposing something drastically different



Budan vs adaptive linearization

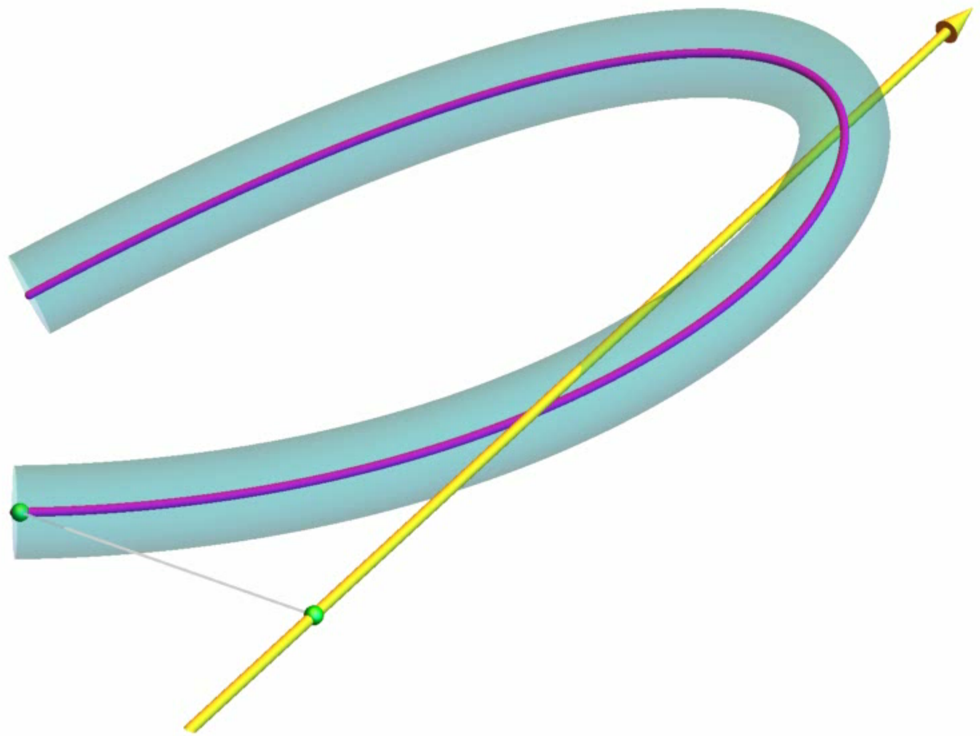
	 1. single hair	 2. lock	 3. curls	 4. straight	 5. curly	 6. fur
average kernel time in nanoseconds	13.6 675.6	33.1 675.2	57.0 697.0	116.0 793.4	145.6 1113.0	90.8 753.9
average distance error	7.7×10^{-10} 2.3×10^{-6}	1.8×10^{-10} 6.8×10^{-7}	2.6×10^{-10} 9.8×10^{-7}	4.1×10^{-10} 7.4×10^{-6}	2.9×10^{-10} 4.5×10^{-7}	5.5×10^{-12} 5.9×10^{-7}
maximum distance error	2.7×10^{-8} 6.9×10^{-5}	8.3×10^{-9} 4.2×10^{-5}	1.2×10^{-5} 2.1×10^{-4}	9.7×10^{-7} 9.1×10^{-4}	1.7×10^{-5} 1.7×10^{-3}	2.0×10^{-8} 1.9×10^{-5}
1000X error reduction as a side effect	best performance improvement			worst performance improvement		

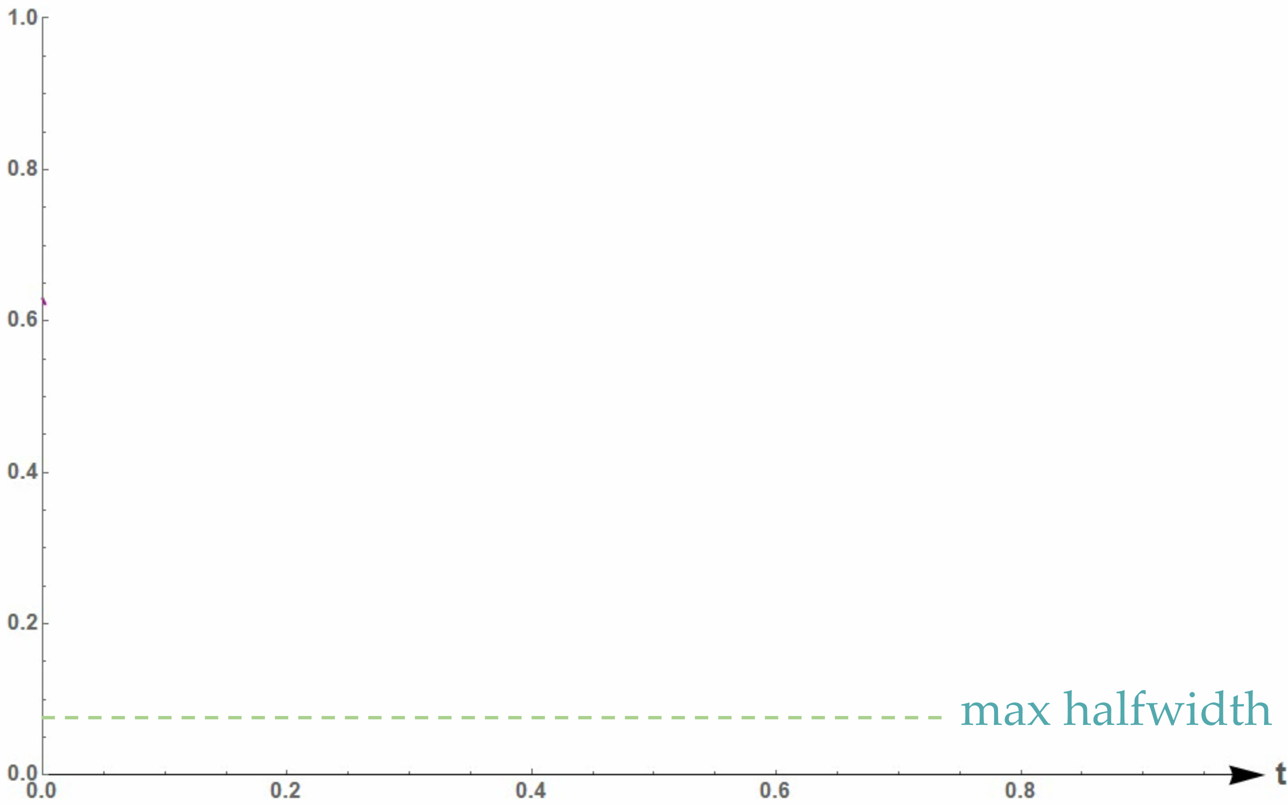
Four (trivial) contributions

1. Raycentric coordinate system
   - Using a sole d.o.f. to allow tight clipping
2. Root localization (the Budan etc thing)
   - Optimizing it for Bézier Curves
   - and allowing efficient conversions
3. Splitting only in the parametric domain $t \in [0,1]$
   - Not splitting control points at all
4. Non-linear root finding
   - Ridders method [1979]

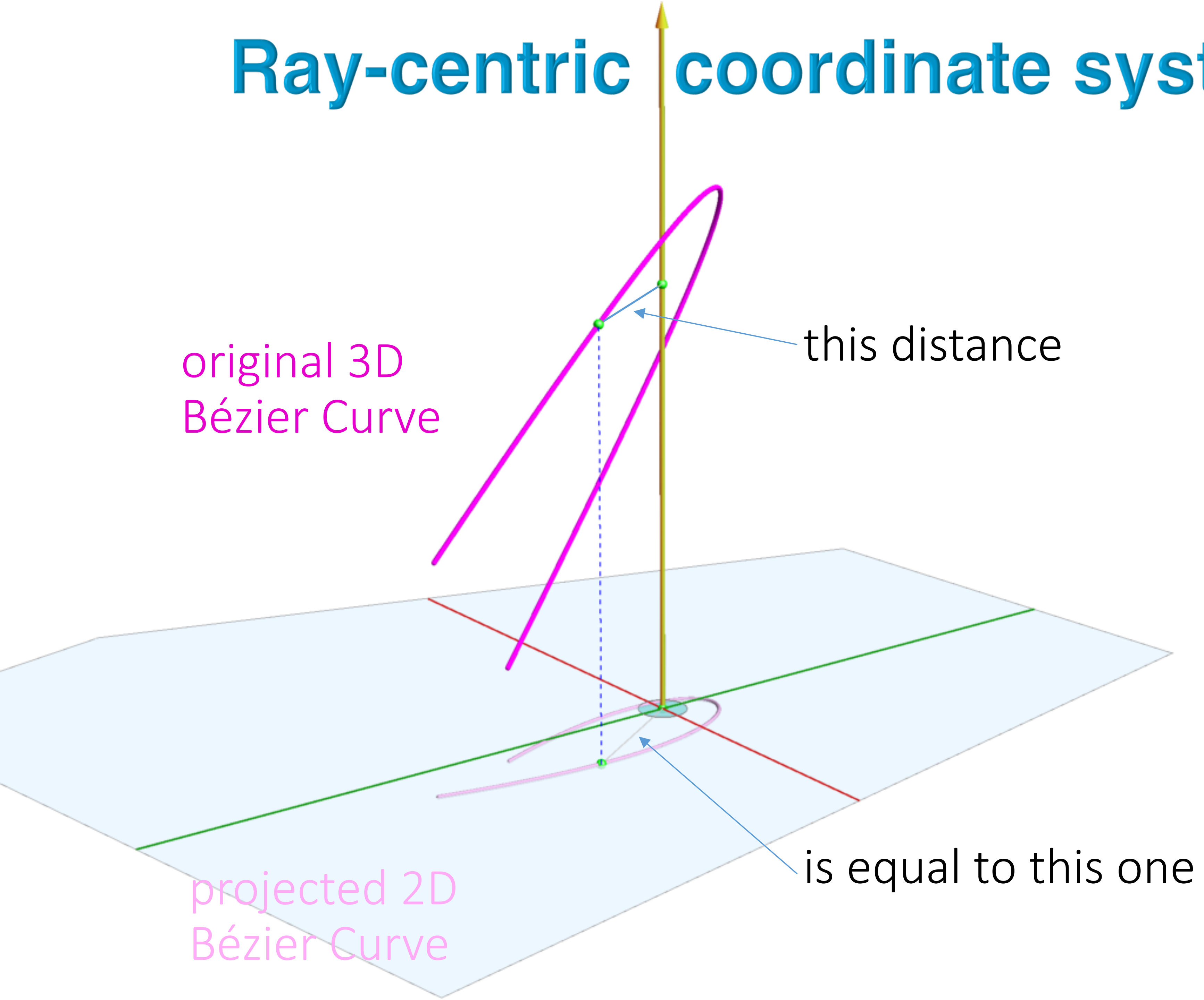
1. **Raycentric coordinate system**
    - Using a sole d.o.f. to allow tight clipping
2. Root localization (the Budan etc thing)
    - Optimizing it for Bézier Curves
    - And allowing efficient conversions
3. Splitting only in the parametric domain $t \in [0,1]$
    - Not splitting control points at all
4. Non-linear root finding
    - Ridders method [1979]

We need minimum of distance(curve(t), ray)

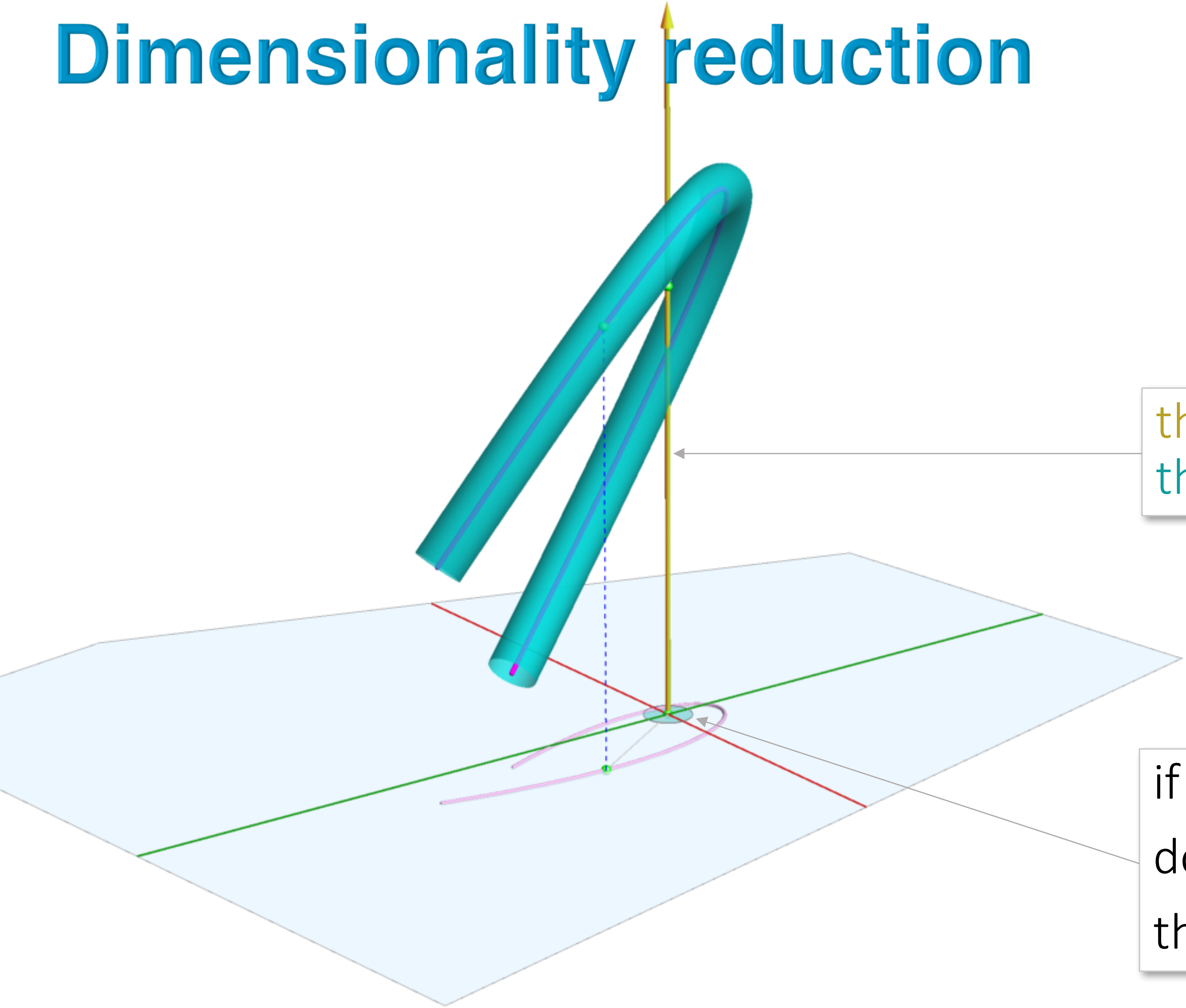




Ray-centric coordinate system



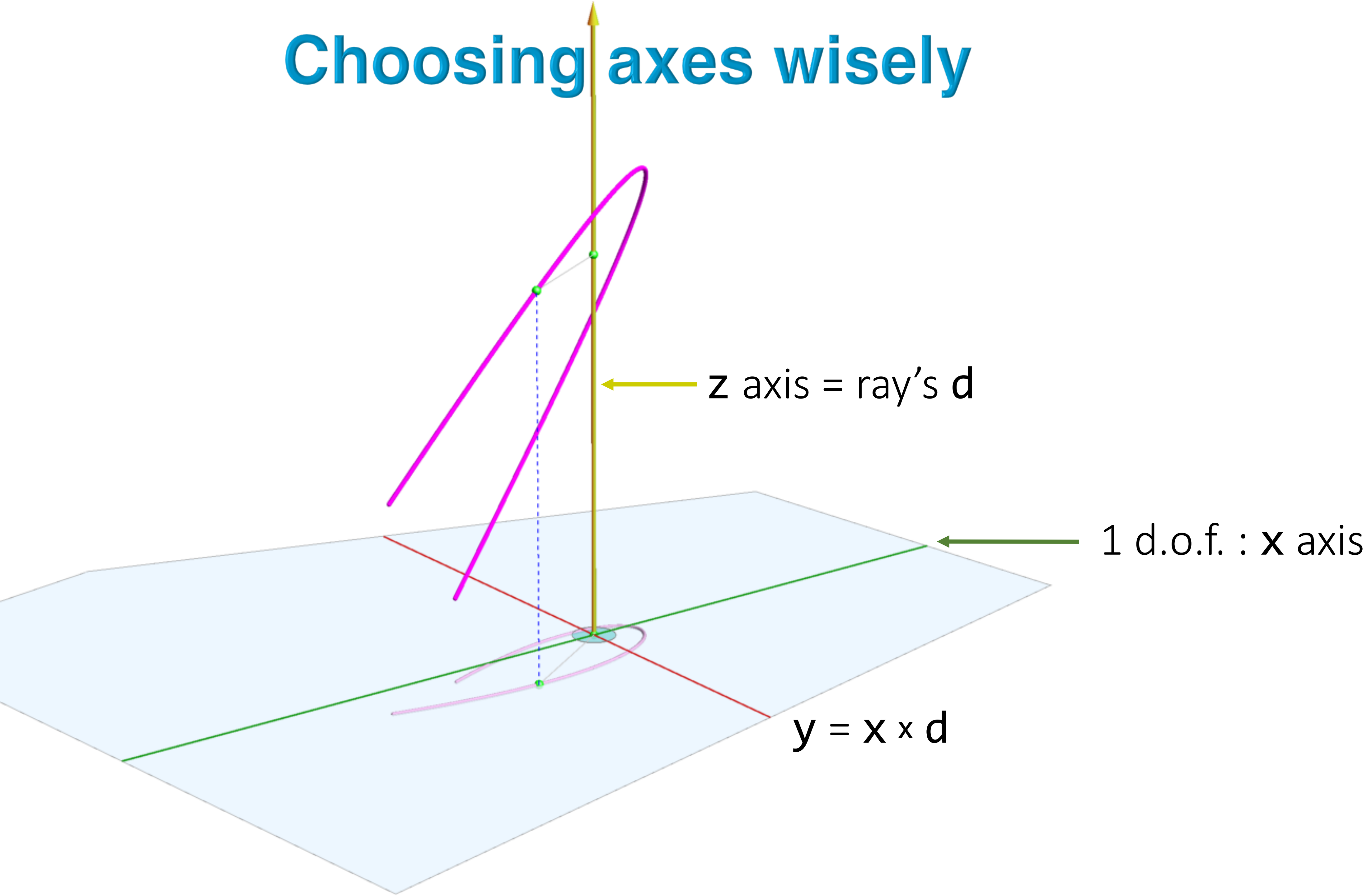
Dimensionality reduction



the ray will not intersect
the swept volume

if the projected curve
does not intersect this circle
then

Choosing axes wisely



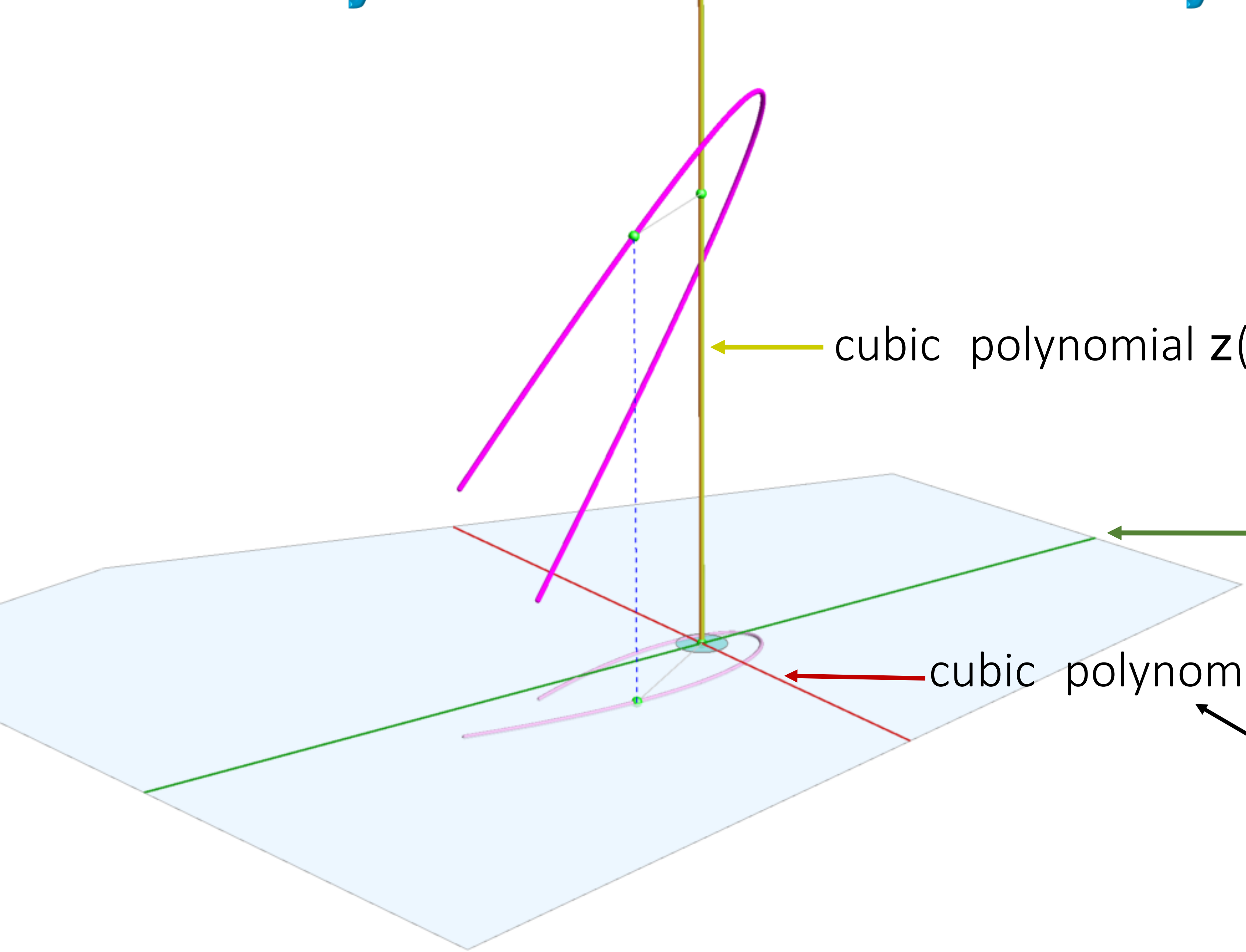
How to choose x axis?


MORE DIFFICULT
LESS DIFFICULT

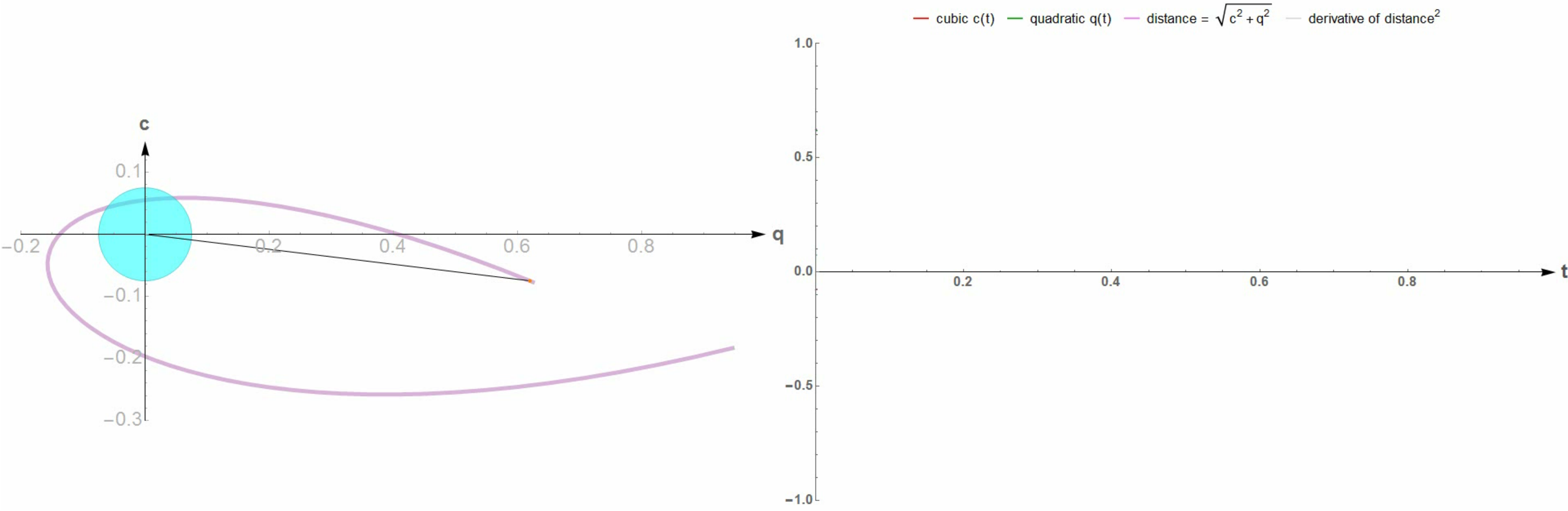
How to choose x axis

it is orthogonal to $\mathbf{d}$	$\mathbf{x}$ = some_vector × $\mathbf{d}$
cubic Bézier curve as a univariate polynomial	$\mathbf{b}(t) = \mathbf{u_0} + \mathbf{u_1}\, t + \mathbf{u_2}\, t^2 + \mathbf{u_3}\, t^3$ where $\mathbf{u_i}$ are 3D vectors and $t \in [0, 1]$
distance from $\mathbf{b}(t)$ to the ray $\mathbf{o} + \mathrm{s}\, \mathbf{d}$ using Πυθαγόρας theorem	$\lvert\mathbf{v}(t)\rvert$, where $\mathbf{v}(t) = (\mathbf{b}(t) - \mathbf{o}) \times \mathbf{d}$ i.e. *cathetus* = *hypotenuse* sin α
x coordinate of $\mathbf{v}(t)$ in $\{\mathbf{x}, \mathbf{y}, \mathbf{d}\}$ orthonormal coordinate system	$\mathbf{v}(t) \cdot \mathbf{x}$
if we choose	$\mathbf{x} = \mathbf{u_3} \times \mathbf{d}$
then the cubic term of $\mathbf{v}(t) \cdot \mathbf{x}$ is	$\mathbf{u_3} \cdot \mathbf{u_3} \times \mathbf{d} = 0$

Ray-centric coordinate system

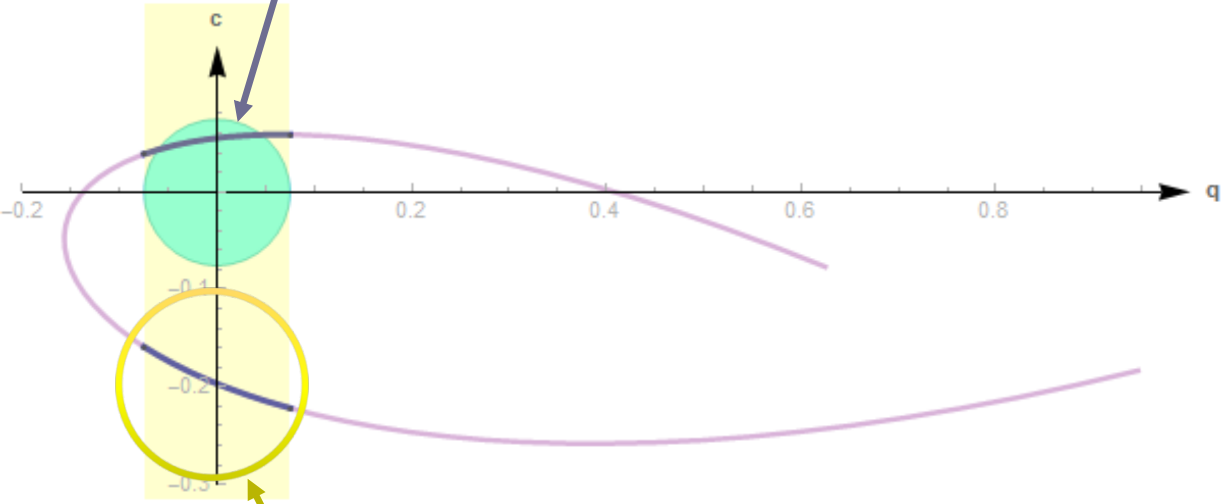


Dual space clipping

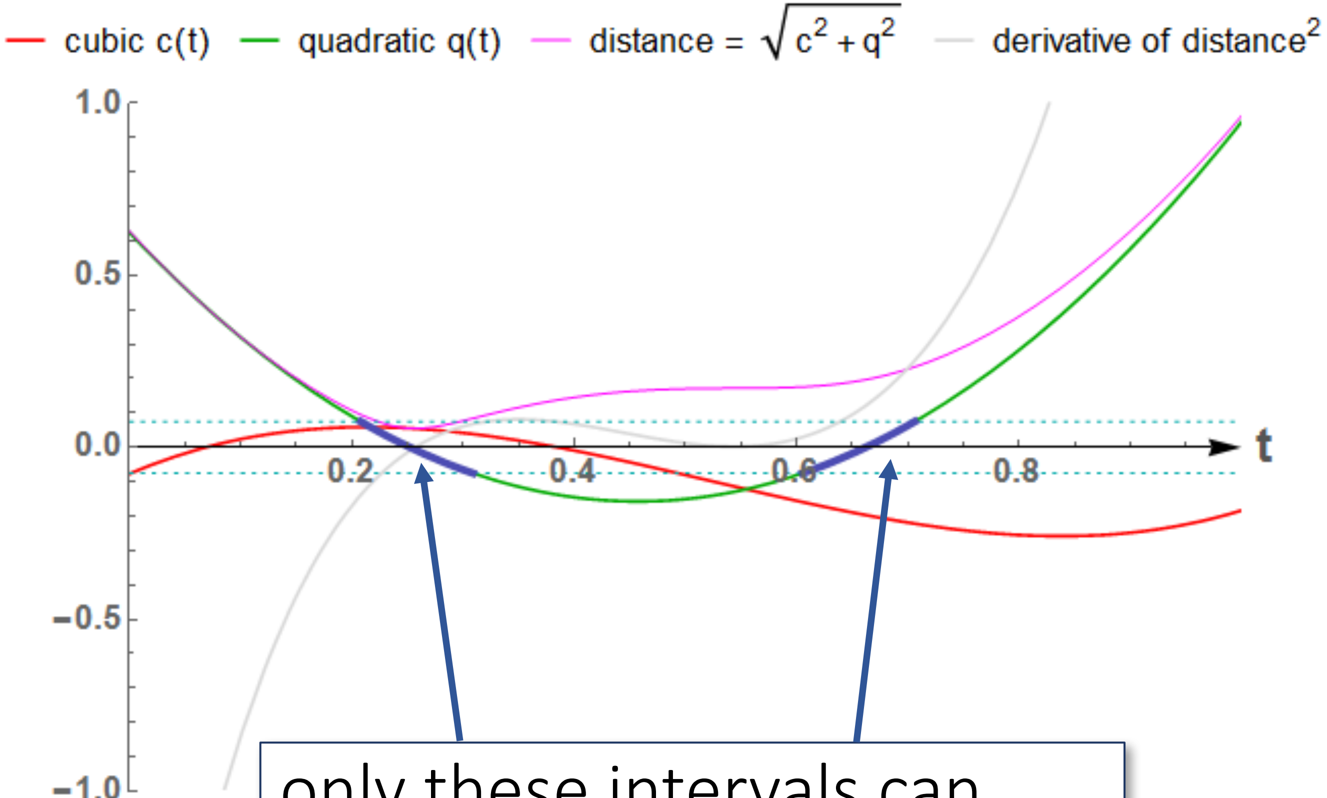


Dual space clipping: interval reduction

and this one can be reduced even further using Budan-Fourier & Vincent's



this interval can be excluded by analyzing the cubic term

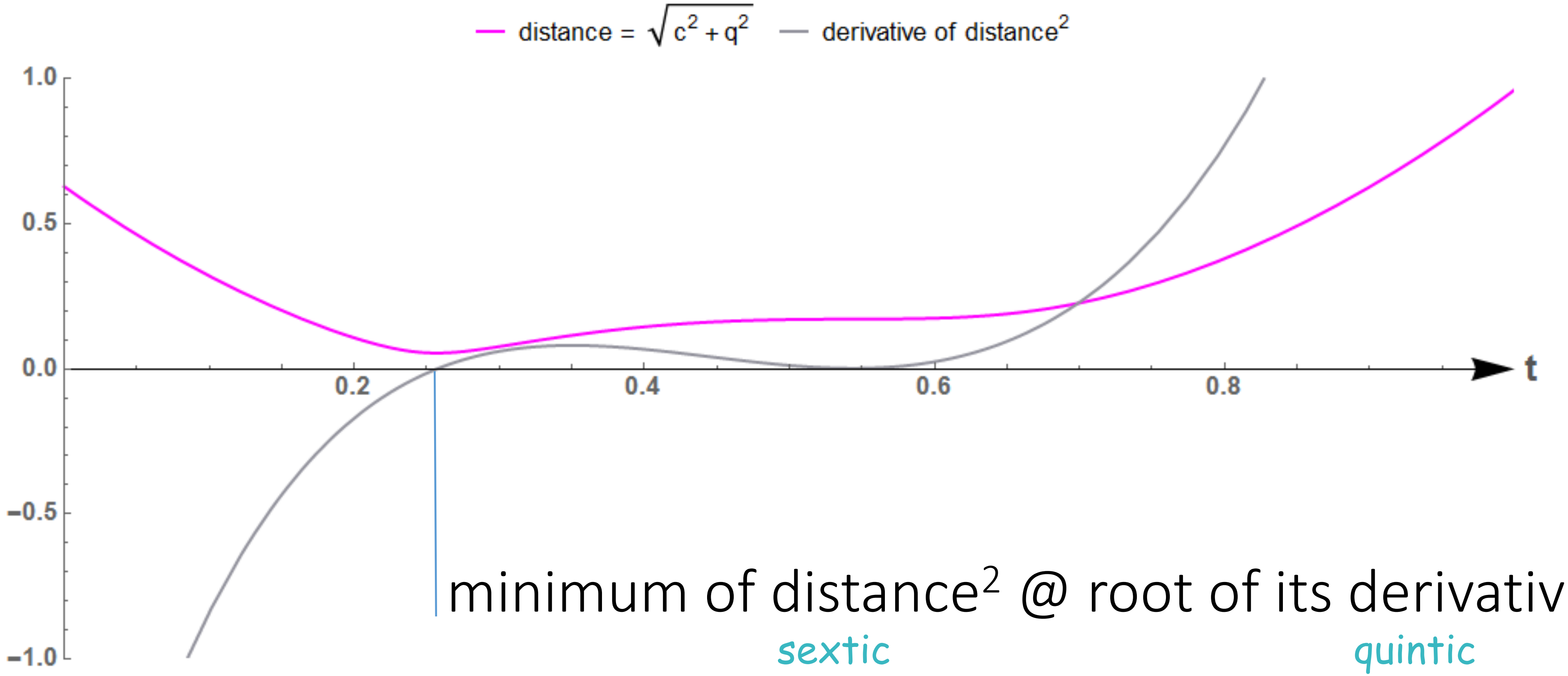


only these intervals can potentially include **min distance < halfwidth**

1. Raycentric coordinate system

- Using a sole d.o.f. to allow tight clipping

2. Root localization (the Budan etc thing)



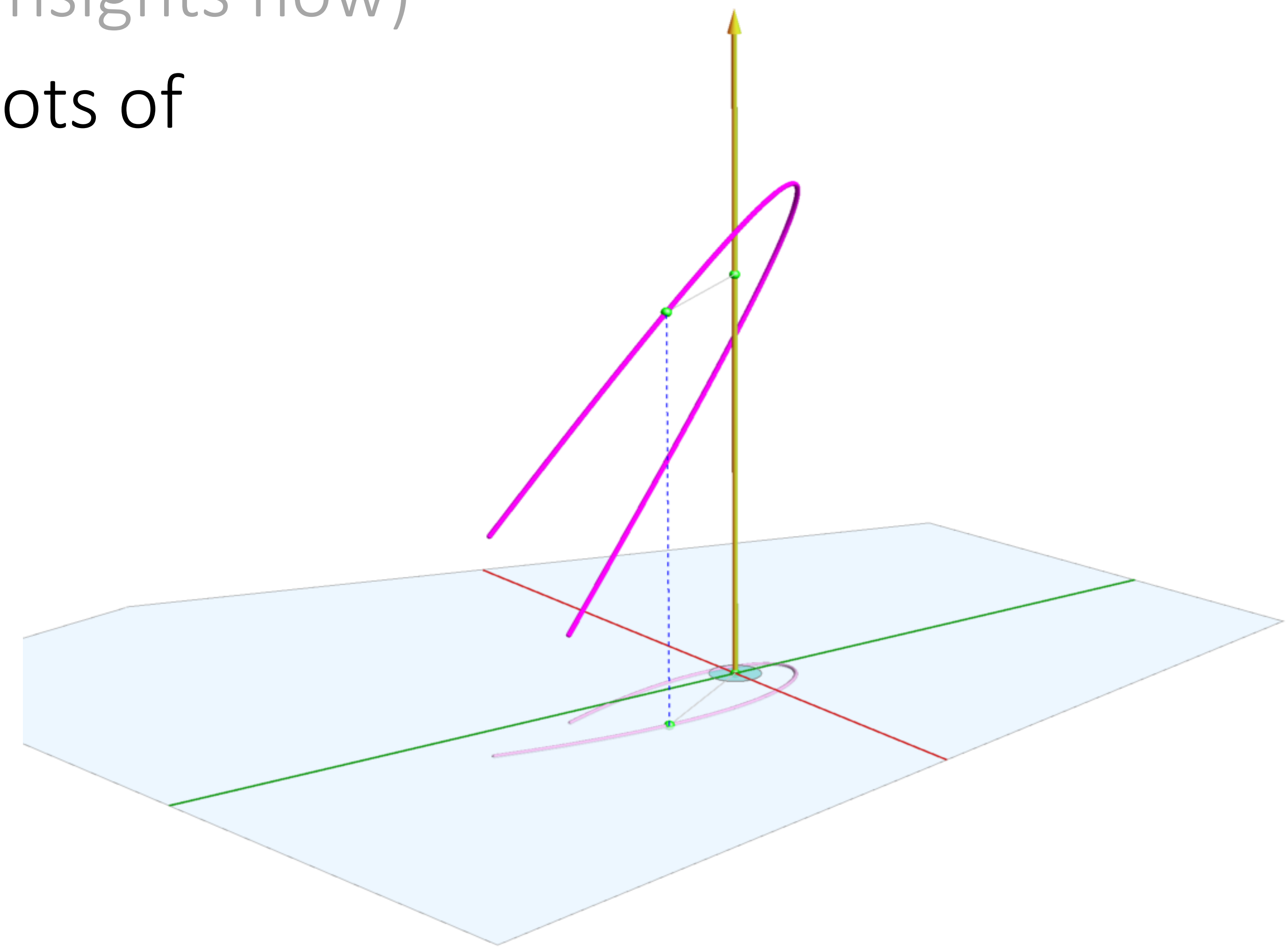
minimum of distance^2 @ root of its derivative

sextic polynomial

quintic polynomial

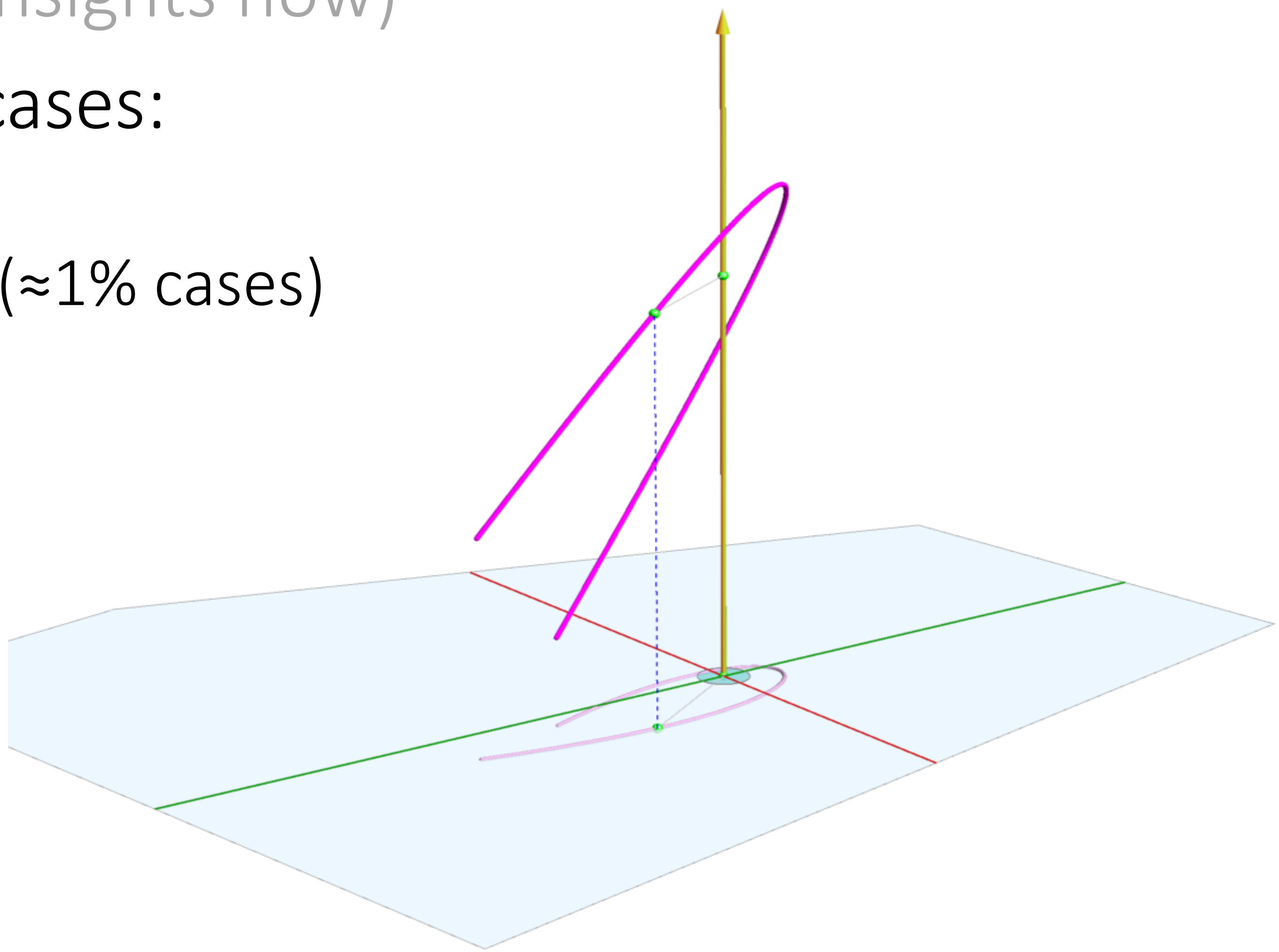
Details are in the paper…

- plus Mathematica code
  (I just will provide some insights now)
- We want to know # of roots of p(t) for $t \in [t_1, t_2]$



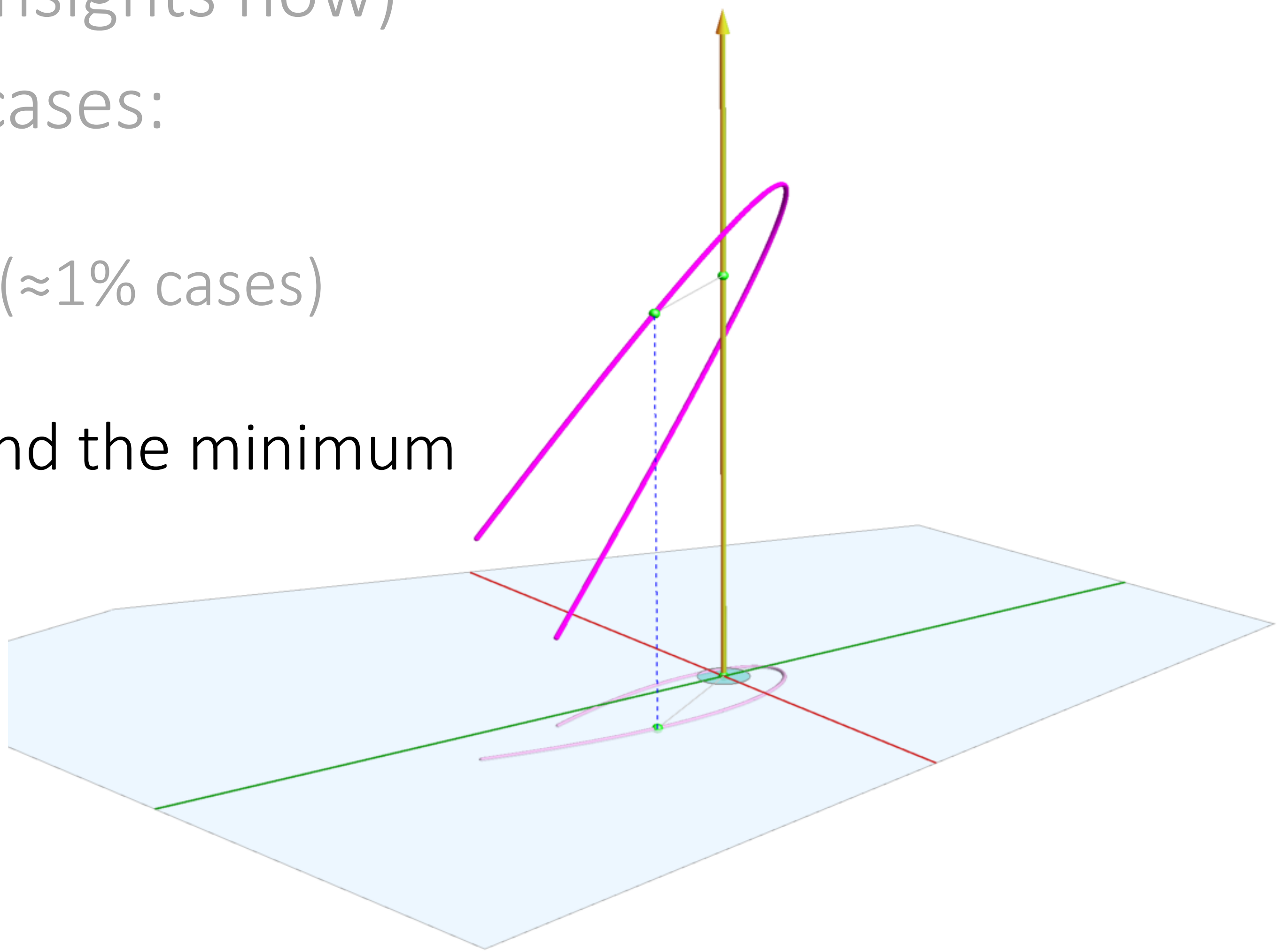
Details are in the paper…

- plus Mathematica code
  (I just will provide some insights now)
- Only interested in three cases:
  0, 1, or many
  - More than 1 root → split (≈1% cases)



Details are in the paper…

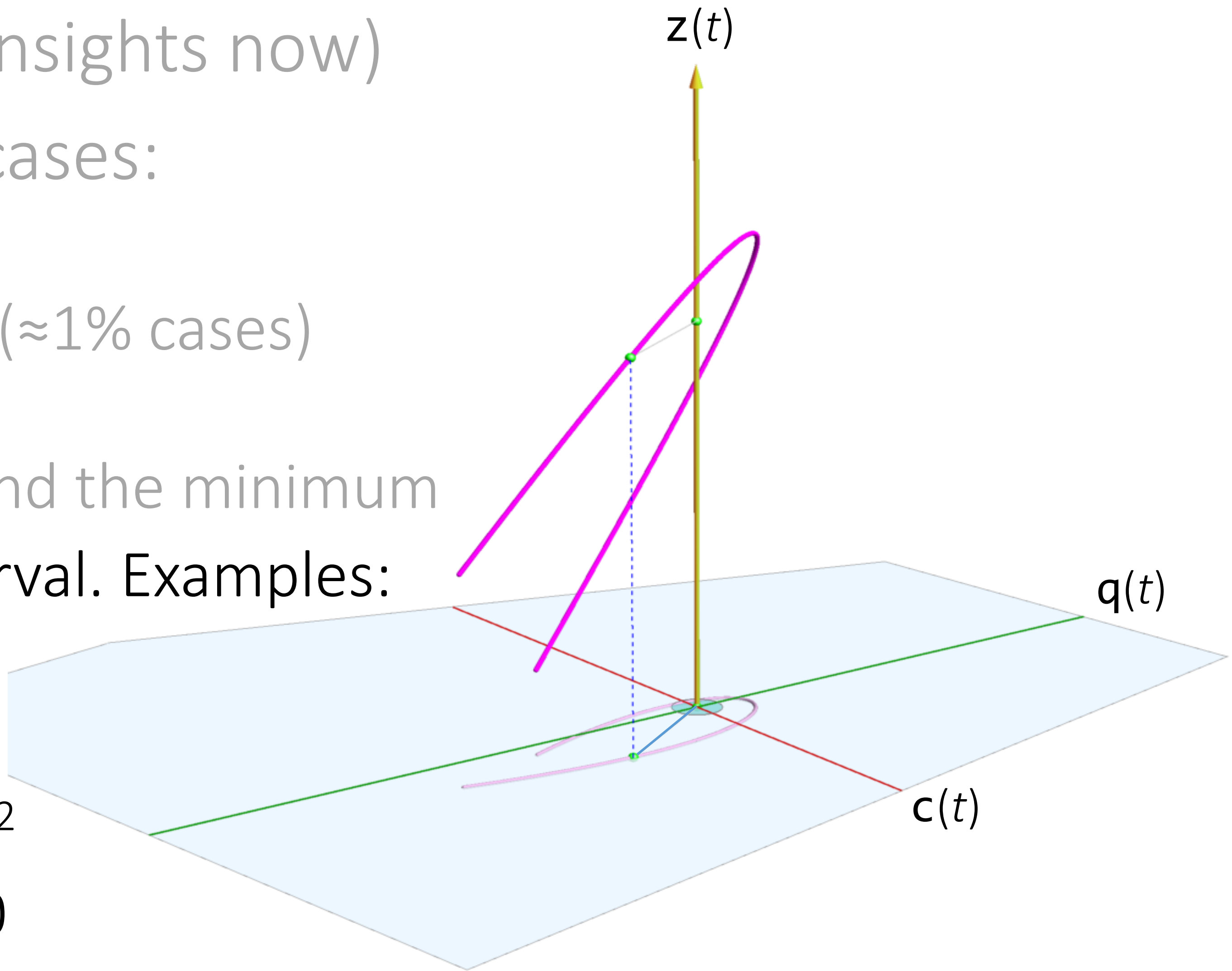
- plus Mathematica code
  (I just will provide some insights now)
- Only interested in three cases:
  0, 1, or many
  - More than 1 root → split (≈1% cases)
  - One root →
    reduce the interval and find the minimum



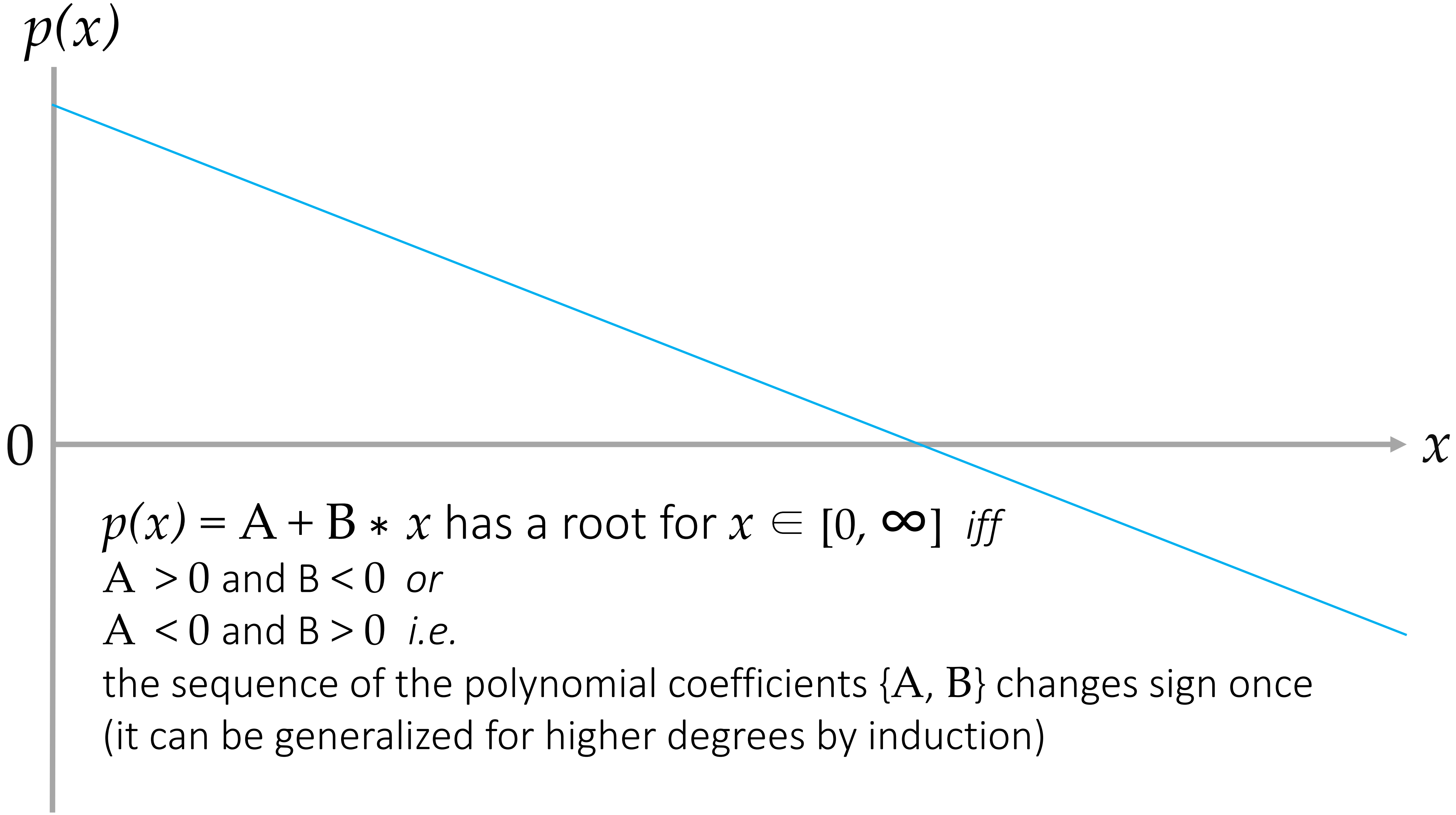
Details are in the paper…

- plus Mathematica code
  (I just will provide some insights now)
- Only interested in three cases:
  0, 1, or many
  - More than 1 root → split (≈1% cases)
  - One root →
    reduce the interval and find the minimum
  - No roots → drop the interval. Examples:

1. $c(t) > |\text{halfwidth}|$
2. $z(t) < 0$
3. $q(t)^2 + c(t)^2 > \text{halfwidth}^2$
4. $q(t)*q'(t) + c(t)*c'(t) \neq 0$



Insights: Descartes' rule of signs



$p(x) = \mathrm{A} + \mathrm{B} * x$ has a root for $x \in [0, \infty]$ *iff*

$\mathrm{A} > 0$ and $\mathrm{B} < 0$ *or*

$\mathrm{A} < 0$ and $\mathrm{B} > 0$ *i.e.*

the sequence of the polynomial coefficients $\{\mathrm{A}, \mathrm{B}\}$ changes sign once

(it can be generalized for higher degrees by induction)

15-16th centuries

- 1637: Descartes' rule of sign
  - # of polynomial roots $\leq$ $\sum$ sign differences between its coefficients
  - no proof, just a few confirming examples
- Wallis' claim: it was proposed by Harriot before
  - 1627–1629: Anglo-French War
- 1707: the rule restated by Newton
  - No proof either
  - Encyclopædia Britannica:
    "he considered its proof too trivial to bother recording"
- 1728 : first (flawed) proof by Segner (written in Latin)
- 1742: first real proof by De Gua

The problem: we need $[a, b]$ and not $[0, \infty]$

- 1807: Budan's theorem

  $p(x)$ has no real roots in [a, a+1]
  if $p(a)$ and $p(a+1)$ have the same number of sign variations

- 1820: Fourier's theorem

  compute # of sign variations lost from *a* to *b* in
  $p(x)$, $p'(x)$, $p''(x)$,… $p^{(n)}(x)$

equivalent formulations

- 1834: Vincent's theorem

  based on continued fractions
  (these 3 formulations yield upper bound on # of roots)

- 1829: Sturm sequence: exact #, though it requires **l o n g** polynomial division

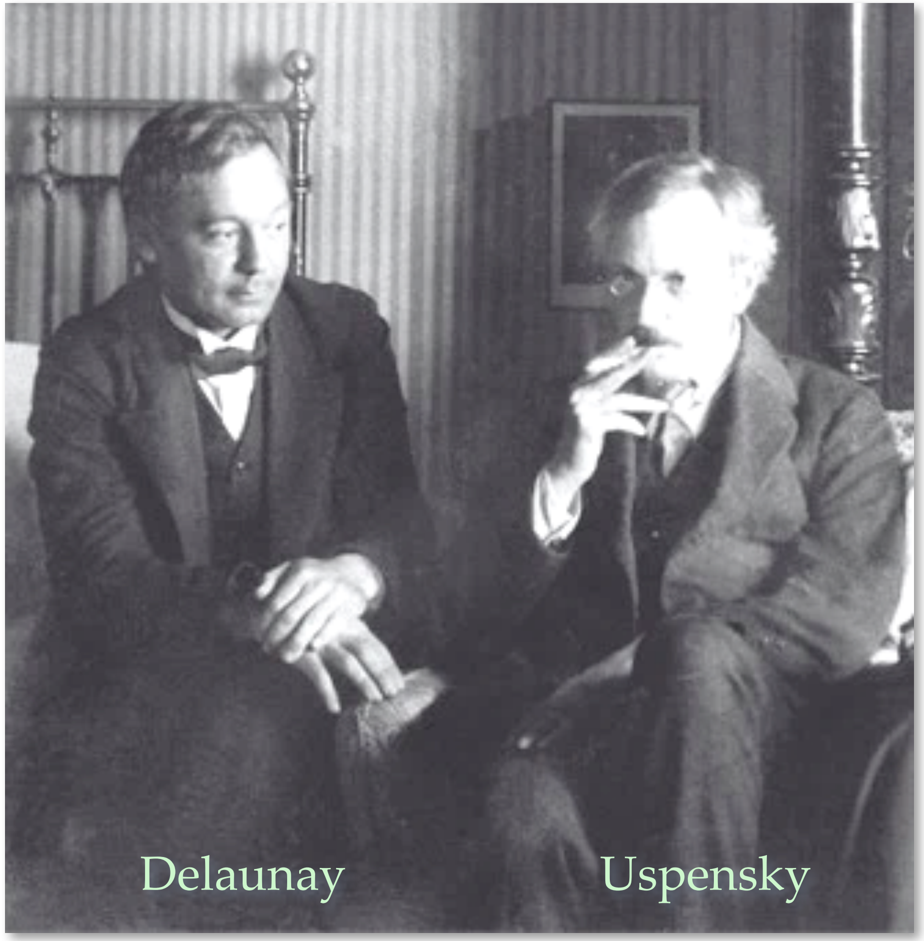
Fast-forward to 20th century

20th century

- 1948: Uspensky's implementation of Vincent's theorem “faster than Sturm’s algorithm”

Historical trivia:

Unfortunately, Uspensky did not have access to Vincent's original paper of 1836, in which Budan’s theorem is cited, and his implementation is actually slower than Sturm’s

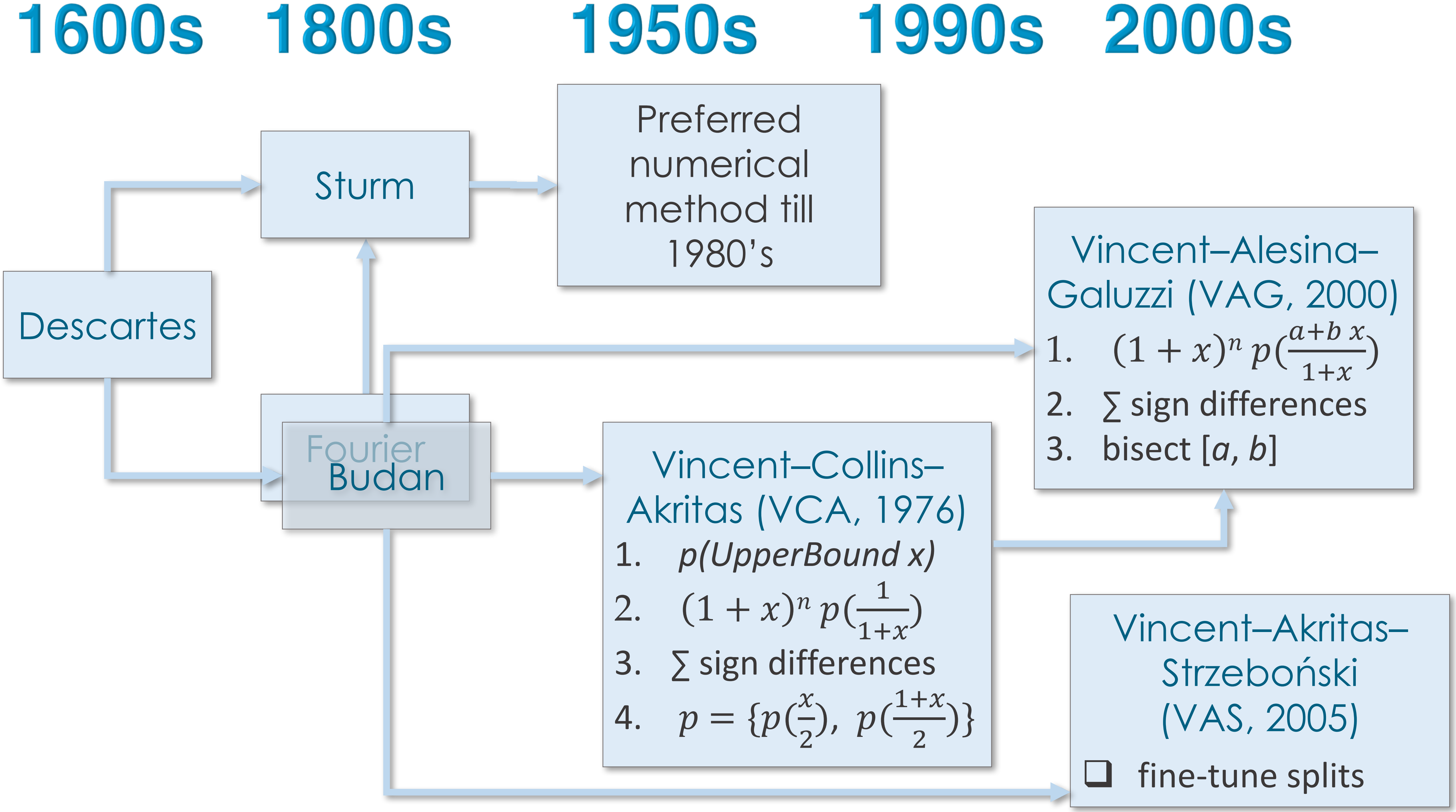


20th century

- 1948: Uspensky's implementation of Vincent's theorem
- 1978: Akritas read about Vincent's theorem in Uspensky's book and made it the topic of his Ph.D. Thesis "Vincent's Theorem in Algebraic Manipulation"

20th century

- 1948: Uspensky's implementation of Vincent's theorem
- 1978: Akritas found Vincent's theorem in Uspensky's book and made it the topic of his Ph.D. Thesis "Vincent's Theorem in Algebraic Manipulation“
- Fortunately for Akritas, the heroic librarian at the University of Wisconsin was able to get ahold of the original Vincent's paper


1600s
1800s
1950s
1990s
2000s
Descartes
Sturm
Fourier
Budan
Preferred numerical method till 1980's
Vincent–Alesina–Galuzzi (VAG, 2000)
1. $(1+x)^n\, p(\frac{a+b\,x}{1+x})$
2. ∑ sign differences
3. bisect [a, b]
Vincent–Collins–Akritas (VCA, 1976)
1. p(UpperBound x)
2. $(1+x)^n\, p(\frac{1}{1+x})$
3. ∑ sign differences
4. $p = \{p(\frac{x}{2}),\ p(\frac{1+x}{2})\}$
Vincent–Akritas–Strzeboński (VAS, 2005)
fine-tune splits

What is important for us

- Vincent–Collins–Akritas form $(1+x)^n\, p(\frac{1}{1+x})$ for $x \in [0,1]$ is easy to compute for Bézier curves in Bernstein form
- yet we already have a reduced interval [*a*,*b*] and Vincent–Alesina–Galuzzi $(1+x)^n\, p(\frac{a+b\,x}{1+x})$ is somewhat more challenging

What we can do though (see the paper)

- Compute Fourier sequences a_i and b_i for $x = 0$ and $x = 1$
- And then on interval [*a*,*b*] using the chain differentiation rule
- Convert to VAG representation for the derivative of distance^2 as (VAG sequence for distance^2 is computed as partial sums)

$$\begin{pmatrix} b_1 \\ 5b_1 - b_2 \\ 10b_1 - 4b_2 + \frac{b_3}{2} \\ 10a_1 + 4a_2 + \frac{a_3}{2} \\ 5a_1 + a_2 \\ a_1 \end{pmatrix}$$

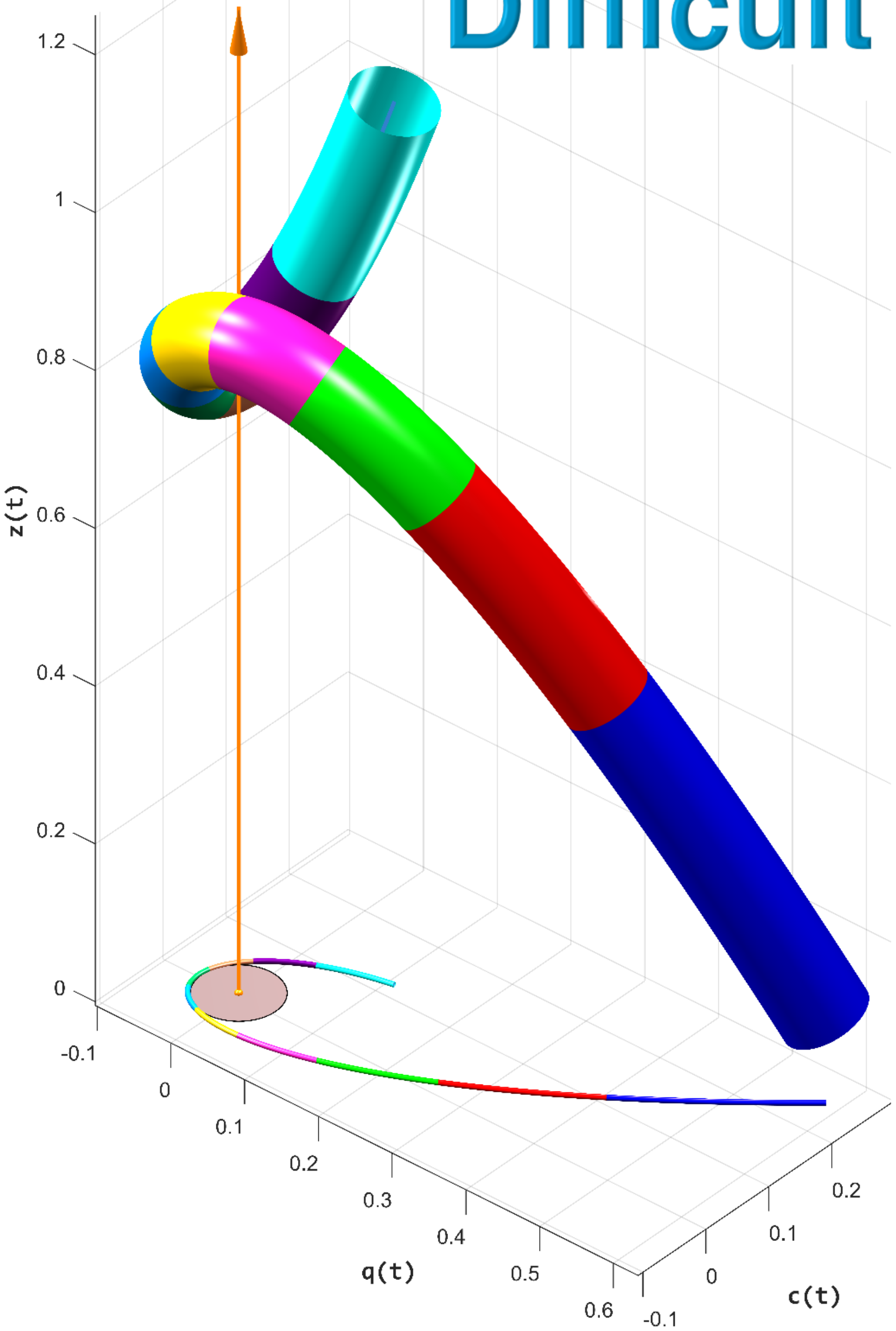
1. Raycentric coordinate system
   - Using a sole d.o.f. to allow tight clipping
2. Root localization (the Budan etc thing)
   - Optimizing it for Bézier Curves
   - And allowing efficient conversions
3. **Splitting only in the parametric domain $t \in [0,1]$**
   - Not splitting control points at all
4. Non-linear root finding
   - Ridders method [1979]

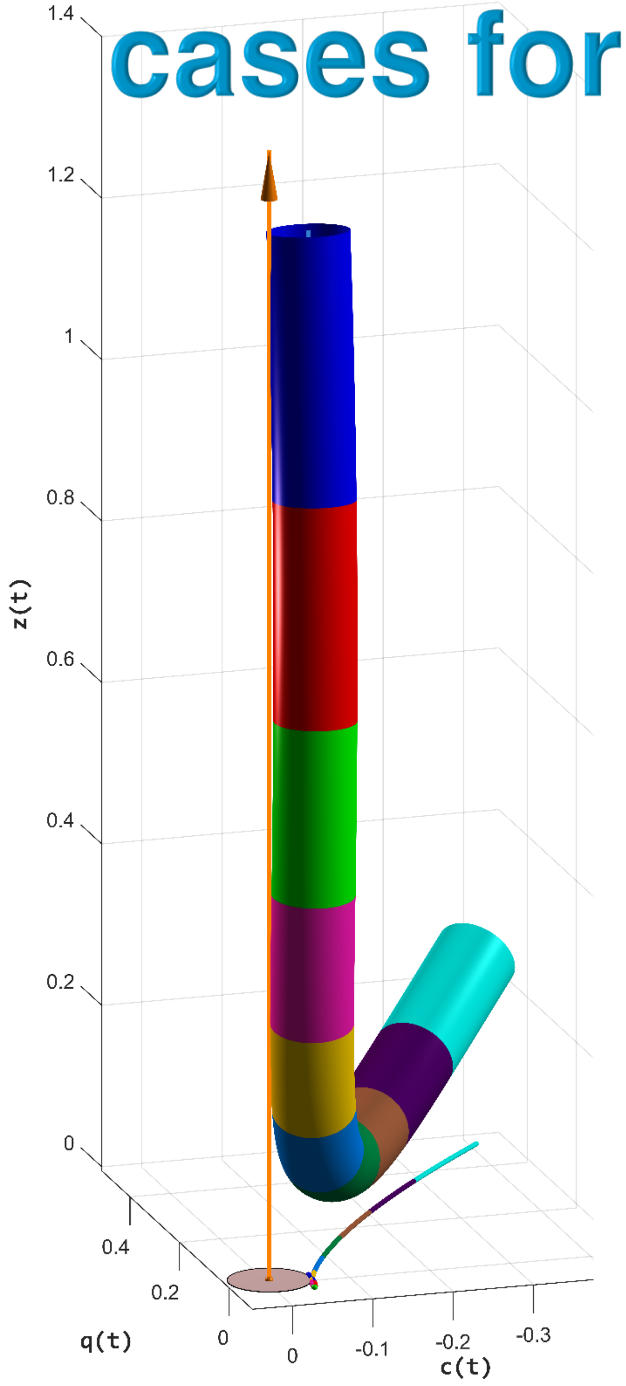
1. Raycentric coordinate system
   - Using a sole d.o.f. to allow tight clipping
2. Root localization (the Budan etc thing)
   - Optimizing it for Bézier Curves
   - And allowing efficient conversions
3. Splitting only in the parametric domain $t \in [0,1]$
   - Not splitting control points at all

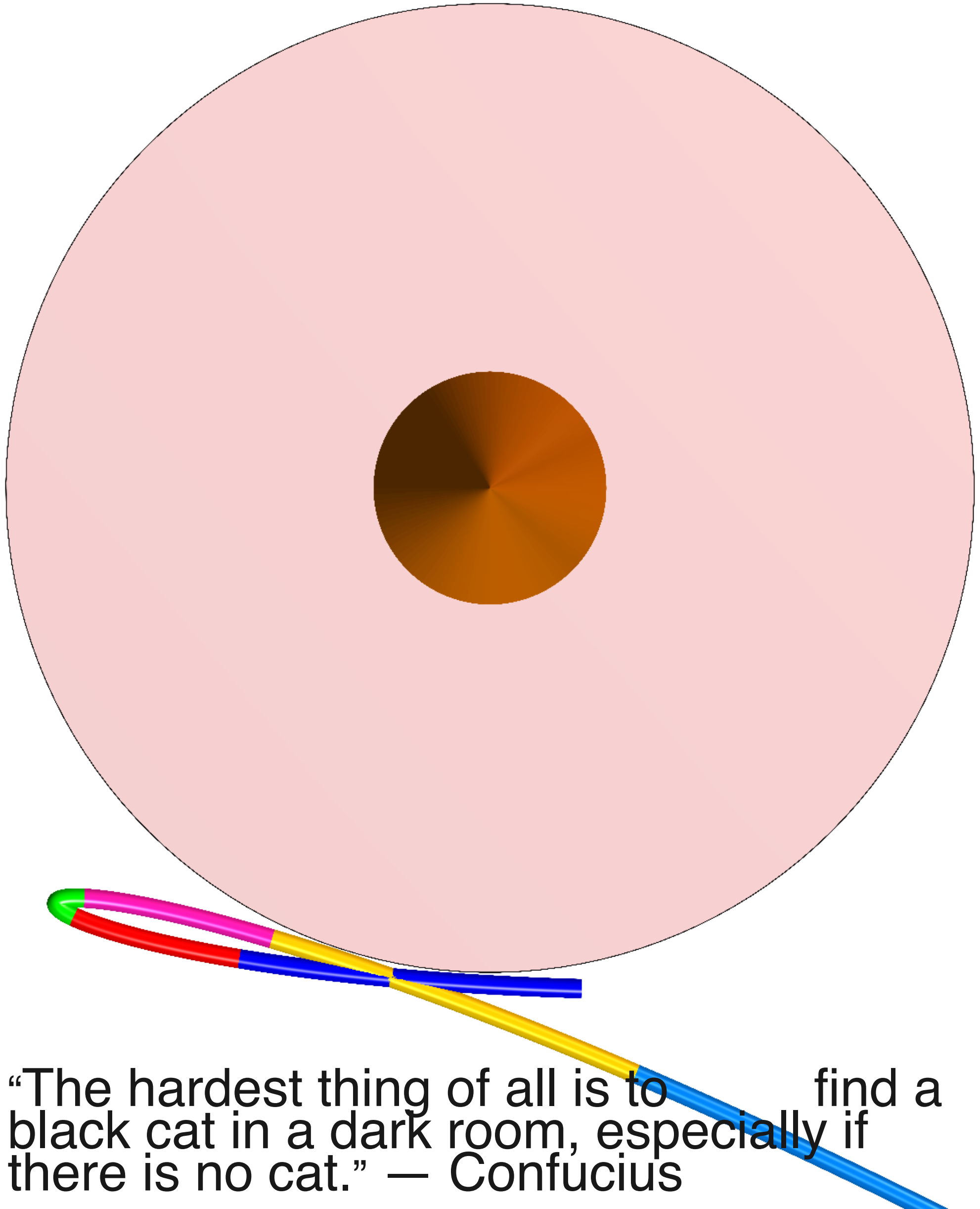
4. Non-linear root finding

Go over 37 entries at https://en.wikipedia.org/wiki/Category:Root-finding_algorithms and choose the best one (see the paper for the discussion)

Difficult cases for linear methods

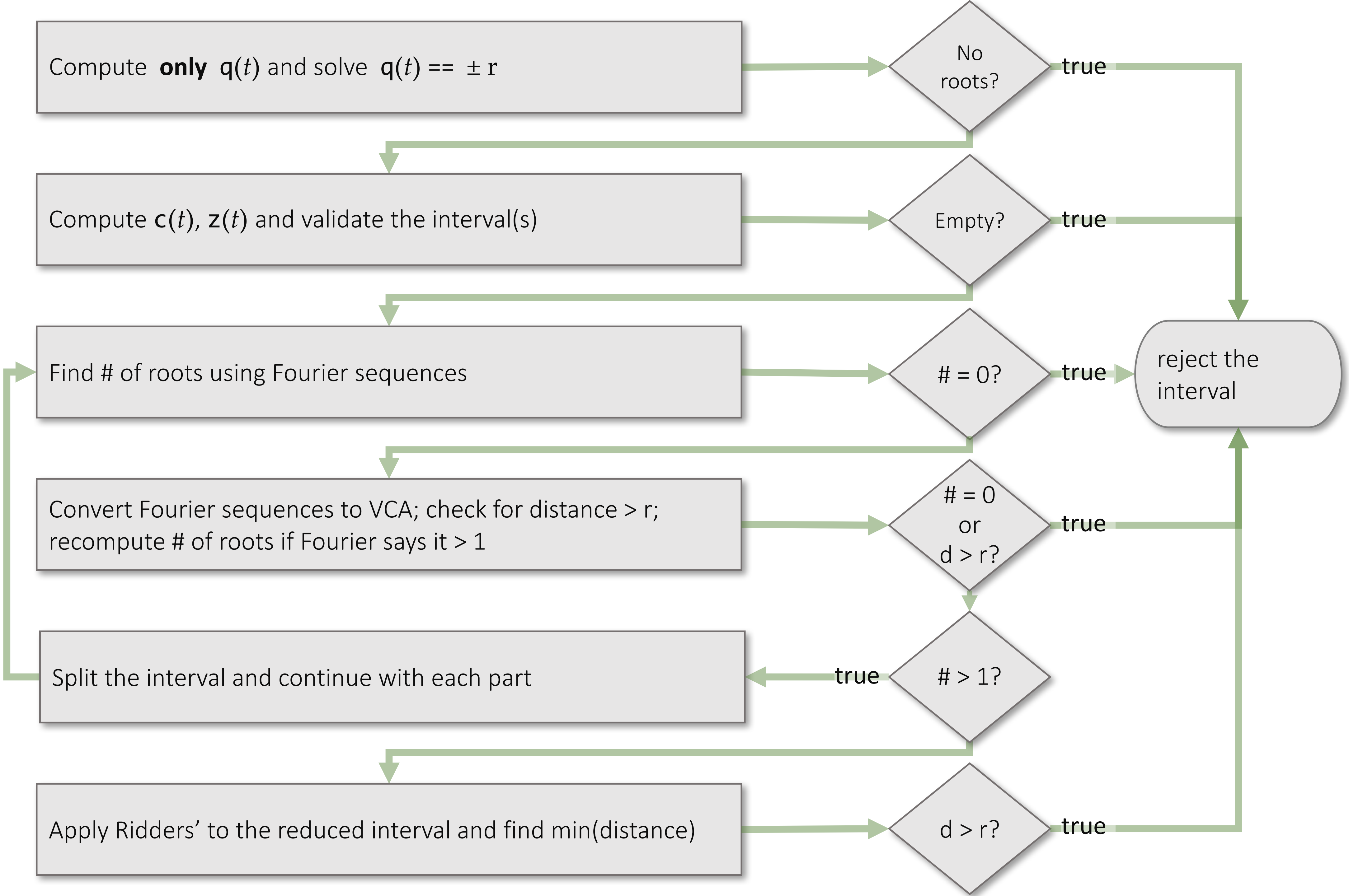






"The hardest thing of all is to find a black cat in a dark room, especially if there is no cat." — Confucius

Summary


Compute **only** $q(t)$ and solve $q(t) == \pm r$
No roots?
true
Compute $c(t)$, $z(t)$ and validate the interval(s)
Empty?
true
Find # of roots using Fourier sequences
= 0?
true
reject the interval
Convert Fourier sequences to VCA; check for distance > r; recompute # of roots if Fourier says it > 1
= 0 or d > r?
true
Split the interval and continue with each part
true
> 1?
Apply Ridders' to the reduced interval and find min(distance)
d > r?
true

Scripts & backup

How to choose x axis

it is orthogonal to $\mathbf{d}$	$\mathbf{x}$ = some_vector × $\mathbf{d}$
cubic Bézier curve as a univariate polynomial	$\mathbf{b}(\mathrm{t}) = \mathbf{u}_0 + \mathbf{u}_1 t + \mathbf{u}_2 t^2 + \mathbf{u}_3 t^3$ where $\mathbf{u}_\mathbf{i}$ are 3D vectors and $t \in [0, 1]$
distance from $\mathbf{b}(\mathrm{t})$ to the ray $\mathbf{o} + \mathrm{s}\,\mathbf{d}$ using Πυθαγόρας theorem	$\lvert\mathbf{v}(t)\rvert$, where $\mathbf{v}(t) = (\mathbf{b}(t) - \mathbf{o}) \times \mathbf{d}$ i.e. cathetus = hypotenuse sin α
x coordinate of $\mathbf{v}(t)$ in $\{\mathbf{x}, \mathbf{y}, \mathbf{d}\}$ coordinate system	$\mathbf{v}(t) \cdot \mathbf{x}$
if we choose	$\mathbf{x} = \mathbf{u}_3 \times \mathbf{d}$
then cubic term of $\mathbf{v}(t) \cdot \mathbf{x}$ is	$\mathbf{u}_3 \cdot \mathbf{u}_3 \times \mathbf{d} = 0$

(*at least*) two approaches

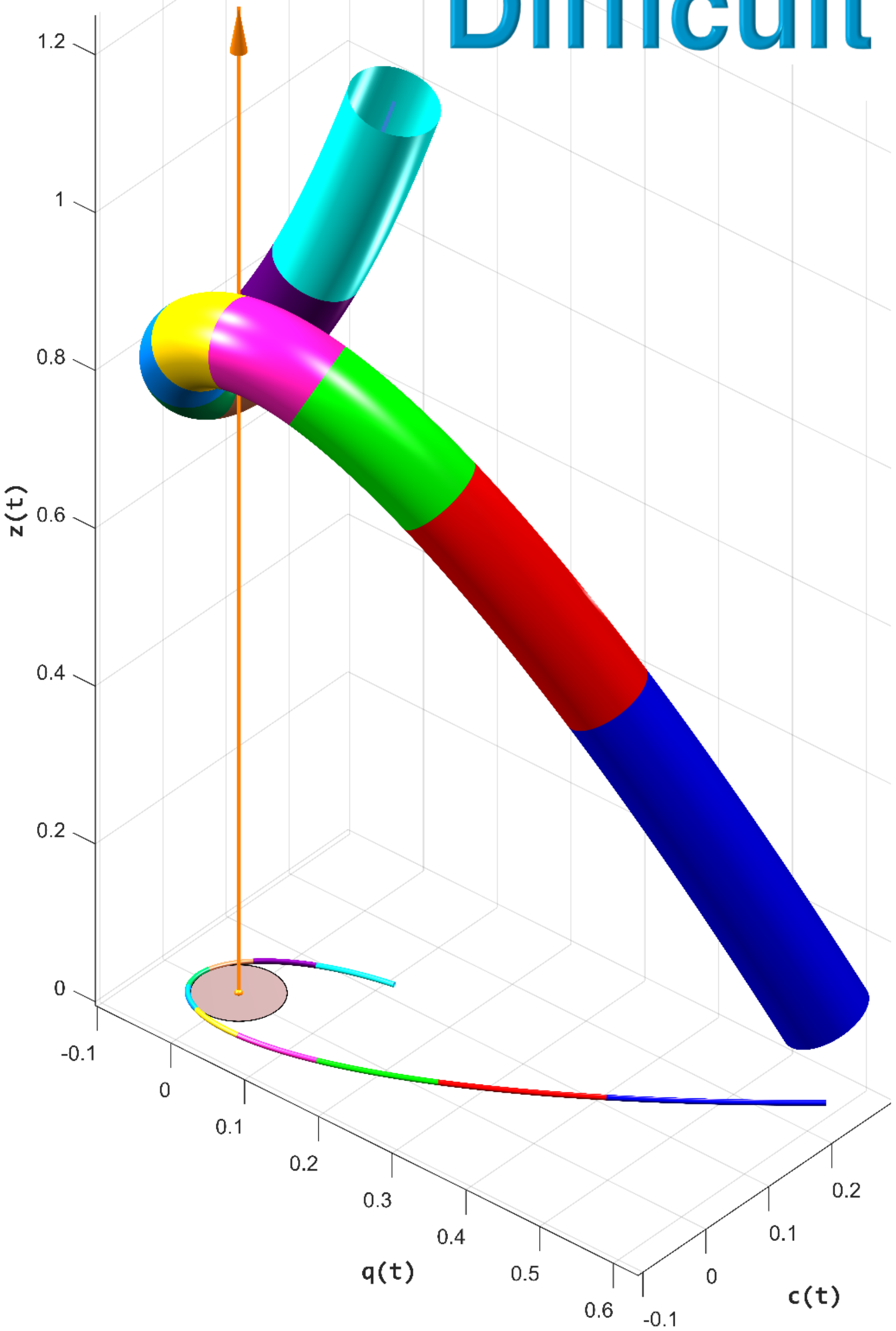
% std::istringstream("1685204011 2815682195 385126455 355161270 2741490130 2213418773 3437980441 3135777609 3486594975 1234267285 1099369070 1004897494 1602088570
2058086804 2766595287 3177268504 2702257490 3124146337 1418718049 1365130483 1265208718 1884250240 3648855035 639353071 4256957795 1605271289 982700018
1959955605 207137408 4223255709 3268747380 62291474 720723770 1432001539 1179295468 4277700772 3384822023 789312185 2745877539 1044122156 2253754244 1044010503
322875373 4015799724 1345322295 99273851 2020442037 2323035699 3484228229 3559337231 4125372910 2067433396 44722077 1341711910 2137959577 1496281135 201449566
2288112463 1808241307 1298612477 3798433704 1473569204 465168807 1173655074 545378063 1215215660 3648634627 759126851 2225790131 2086325018 2554369261 2276960658
941880856 1066069953 1717952623 1984850353 3343835836 2695933096 1656822112 3932875924 3773638371 931018417 2598849831 2643279084 3503712314 132947922 3097522224
3702851089 3056142194 1195040649 412219650 1994870821 337765008 93156141 95192607 1336967578 3577484679 2396266869 4258980693 3395204945 582432190 1581048411
3952965608 2780435230 1281853044 2499783058 4253808186 3062605288 38318014 3270247523 3558152782 729912784 1543269248 3164959574 2404021398 13578967 1272258119
1412589587 2719134096 1701130513 1067314120 1861488225 1184566362 2403701602 1240565084 1311990574 3835906313 3452447601 2539071098 104883673 1121472543
4211714649 3614466054 3738931070 349471191 842287450 3277331210 3245964310 2565353939 519799040 1480779148 1645741614 2383583257 2373547574 4237915924 3202894756
3148590864 1290450093 1945982064 1345964586 1059767949 3120352056 2954115194 1970962417 2807818826 4288411395 280458140 3196262697 643690901 3765177704
3753676503 2241842757 3579928997 3628321 781543337 1997166418 3876904805 2056501733 2952747932 1092137983 4211339744 2251680314 3575996612 2985995824 981302376
2179841463 155498009 4097718670 3090426451 3950054312 3632602667 1556106877 4148384674 3401066556 904566931 20569528 595849810 3623517461 4133363818 3243899690
2584217835 2622436780 4005317734 2817470874 3188779450 231190107 1955823019 337022679 2604165785 3163292398 4114707428 1764383212 2572177483 1973796504
3750958121 804745055 1993459785 4276784439 3834022484 2776507428 3508967888 1699847186 3997024083 82725733 606063791 2984143074 2432647480 254985915 2122283937
47474427 2110125283 2572455943 3415375831 2785076099 2627783621 1592714596 784764603 2178245546 85801516 3487400001 3855936752 2658437062 1191015740 2523391738
3797997762 2105265680 2442552993 2142812732 322379775 2295863178 2794525080 3437059251 314557026 2908496797 3250791855 2027612625 4197355526 2878636016
3249339762 2532865934 3704952630 670074340 1388828656 788474898 3854566936 4140031142 3504167705 1389091907 2702915852 2561147273 3521707995 2696907069
1086119393 778070631 1103895211 2725347355 3775130599 2198148607 3216052957 4228827656 2252511845 1236863666 3953132719 1230365933 3101269550 1186264175
3543486074 206944210 1542383856 713154860 4277298036 958197260 2518833110 1935219647 473080082 933900087 2280263889 1545506654 2992728219 3918997118 3584431475
34563667 3726297827 54782853 796465567 1356215266 3044799671 4241451410 3427418207 3896549751 2450704368 1040672167 2850192145 2612837038 470790924 1851199725
651950670 3453976569 3430959318 795458110 2299767212 2790396125 327221299 1080385624 3606947159 2228760415 2720953837 3875172232 3612108309 3865054701 349953574
4289395775 2501222638 1528482399 2020615770 4181906988 3410405585 955711009 2253974861 1765664884 1577473139 3805172805 3161110250 1381257941 1331962610
1641731630 1732159707 3658172755 3521101282 2802471960 2999654806 1574784185 795196910 3209317949 4092172083 2673956681 2856177 578665840 1410117260 876958718
759108244 2363460035 1697686405 474884213 1974441099 2894689749 2402061879 2485221710 3157428322 2240491335 2467861665 78831800 1507664386 4253484954 176546249
3307579066 2300758379 2254547164 2381868806 2944987653 482229557 776111196 1201108416 4044144538 3049224659 2909989356 3137907294 3860493253 2731451544
1238588477 308511016 3916979781 2732239132 2358757461 3766065875 2676188561 188202881 3942750312 3776931515 241434653 1850642935 1456881045 143032556 1854089893
638425150 2609397761 1259855291 3995665903 2748228522 2410904279 4133385369 3890244691 1269927202 2173266786 357416564 667171161 3498721971 1460437255 405085942
82894503 1690318973 2037687943 1842178682 224646948 3363972818 4258671380 2754564275 4189098102 1328192715 2336094037 2419140087 2523338830 1019298722 2531222161
3353841831 4013938269 309977792 3081713515 212274969 1521244042 161803909 3009038632 3421264194 3435790678 3418096955 3296790731 840138908 2244557633 1542471205
3120941547 1320508325 3440635245 2028899360 536998848 1450634364 4108661774 799244255 3725118394 3968688829 3637617847 751194310 2296501729 2042945363 426573599
1784758273 3281928647 1526585372 3818441274 588204871 121811918 1647063058 3048088652 3309572756 2012161634 496640479 721991116 2890786638 244838095 3966173311
3075051201 3636506565 4169158901 3031275750 2374719437 614493573 1753900116 3902674021 4128416795 2882770774 2579671555 3905854789 1083737215 3800413592
1210948362 3276017723 2282774525 2781763713 2270955670 1408407628 2555530570 2697725778 2476258427 4012045993 2238860543 2822162111 3642920832 1510259615
3583481159 680217030 366547729 2068816041 2281458749 1427164625 4000679175 1269342348 4149071514 1735484759 2354184937 2019027891 296763703 1240542129 2924965487
2475812865 3922667056 2844811041 2262058450 1942740244 2958129230 2207851522 3685692935 2586080668 849737567 2759160197 3402107726 3658722413 2953570163
3550792355 3134427951 4051118033 3071285739 3633769471 4223203455 1750624000 3070701178 4076572526 1694231832 2968907253 589370252 676766806 242933257 1435200681
901761125 3987414775 549440611 3938557999 776978301 3948639091 871071059 4080983779 892588165 2276160415 2899223576 940485226 1474426875 3652297164 3008733142
1646036176 3322626466 3757510579 1101947551 2531620998 1369649733 2602908422 1868656327 2770512978 722783685 3719732719 352558163 2203720183 472079586 3906378084
1519625950 1688678199 1085507723 2393684656 3974950009 861706338 1405260523 3905499796 2576664278 367388648 3827804365 3265124748 2292014770 4103245240
2949120016 4201622372 3029975278 1599826792 3345716154 1435388747 779374721 847236661 1362737722 3960278681 2083335957 3851503928 924809628 3133372642 418288311
3114587019 133875814 3724126360 787970690 700508582 565184219 303753669 203139608 1536751064 1138303486 2043994877 2203741135 2767732549 2663315560 1462681144
2831978468 151051814 524549479 440408845 3810501516 1458311383 1577004251 3601030768 891605198 ") >> mt;

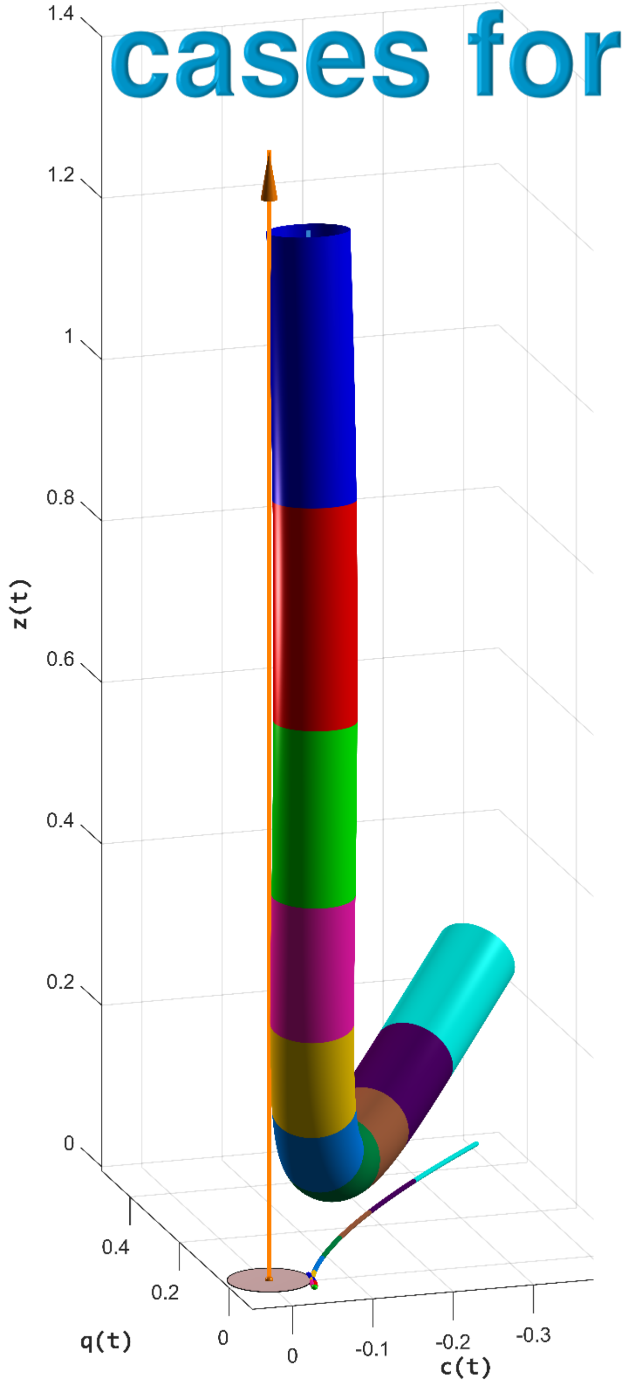
Why *t*?

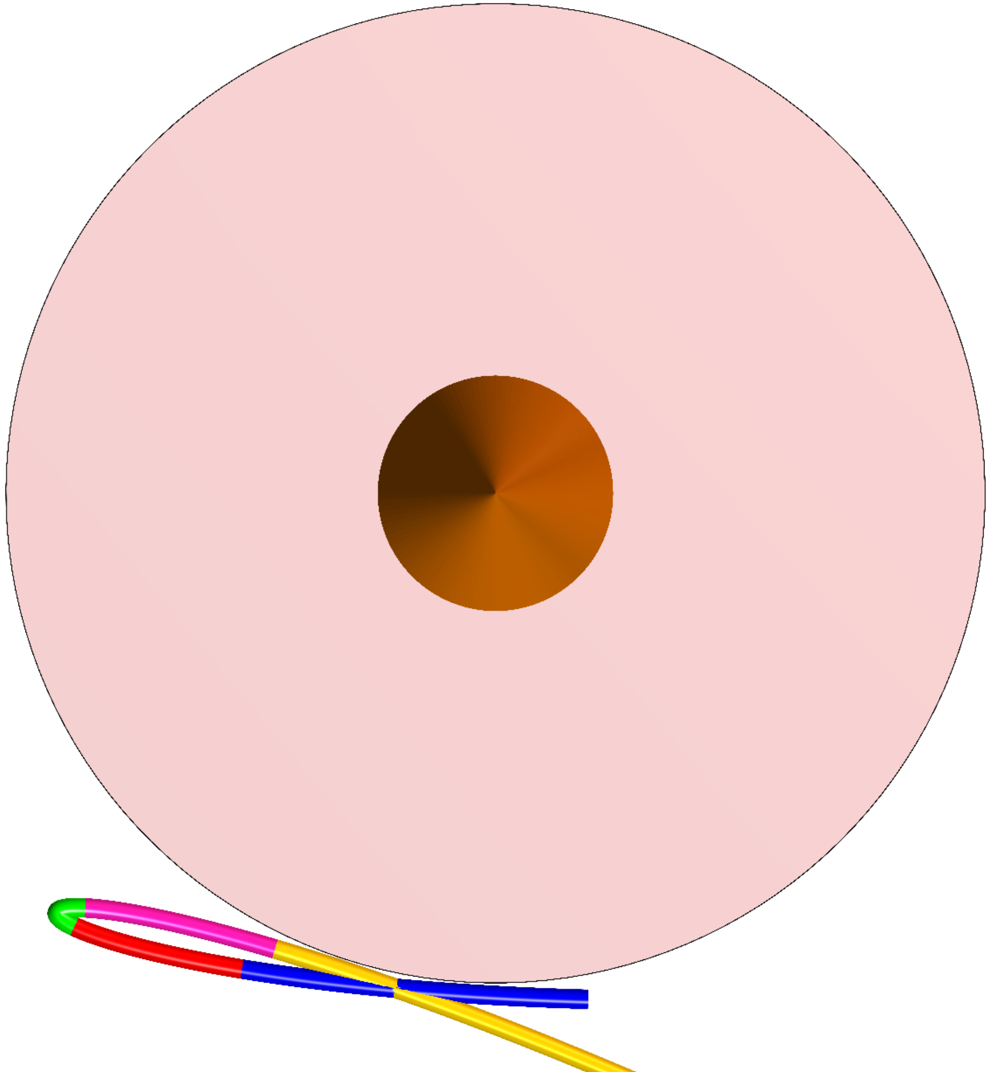
```
par.halfwidth = 0.005; bc(:,1) = [0,0,0]; bc(:,2) = [1,0.01,0.01]; bc(:,3) = [0.5,0,0]; bc(:,4) = [1,0,0];
clf; hsShowBC(bc, par, [0.2,0.4,0.8], 0); axis vis3d; axis off; material metal; lighting phong; axis tight;
printcam(cam); light('Position',[0.5,-3,2], 'Style','local', 'Color','c'); camlight headlight; figure(4);

print('-dpng', '-r768', 'w:\ortvet\hair\rayBC\paper\HPG 2017 presentation\\straight curve 1 color.png');
```

Difficult cases for linear methods







“The hardest thing of all is to find a black cat in a dark room, especially if there is no cat.” — Confucius