Real time Ray-Casting of Algebraic Surfaces

Martin Reimers Johan Seland

Center of Mathematics for Applications
University of Oslo

Workshop on Computational Method for Algebraic Spline Surfaces
Thursday 13. September

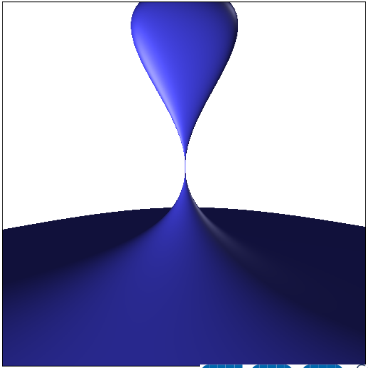
Raycasting

An algebraic surface is the zero set of a polynomial

$$f(x, y, z) = \sum_{0 \le i+j+k \le d} f_{ijk} x^i y^j z^k = 0.$$

Raycasting amounts to "shooting" rays inside a view frustum (VF) and determine if they intersect the surface.

- Can miss thin features
- Conceptually "easy"
- Embarrassingly parallel



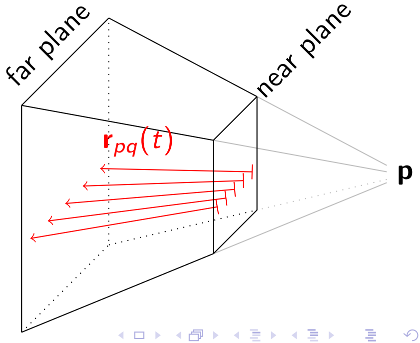
For a ray $\mathbf{r}_{pq}(t)$ raycasting amounts to finding the smallest $t \in [0, 1]$ s. t.

$$f((1-t)\mathbf{n}_{pq} + t\mathbf{f}_{pq}) = f(\mathbf{r}_{pq}(t)) = 0.$$

We would like to work on a univariate polynomial in Bernstein form,

$$f(\mathbf{r}_{pq}(t)) = \sum_{k=0}^{d} c_{pqk} B_k^d(t) = 0.$$

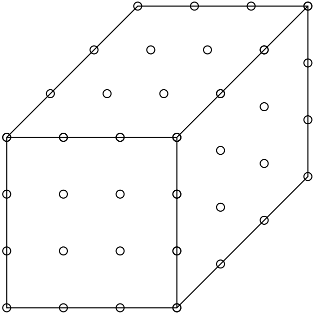
- Assume a screen resolution of $(m+1) \times (n+1)$ pixels.
- Pixel (p, q) corresponds to a ray through $\mathbf{p}$ and the pixel with coordinates $(p/m, q/n)$.



Overview

Our algorithm thus consist of the following steps:

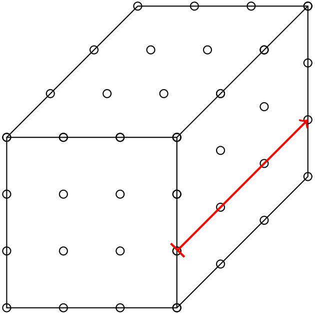
1. Compute ray coefficients $C = (c_{pqk})$



Overview

Our algorithm thus consist of the following steps:

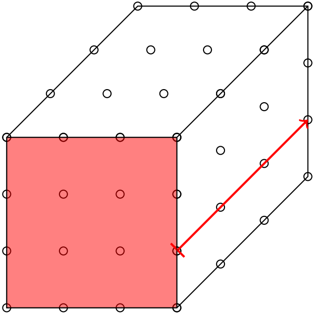
1. Compute ray coefficients $C = (c_{pqk})$
2. For each pixel (p, q)
   2.1 Seek the smallest root $t \in [0, 1]$ of $f(\mathbf{r}_{pq}(t))$.
   2.2 Compute a color for pixel (p, q).



Overview

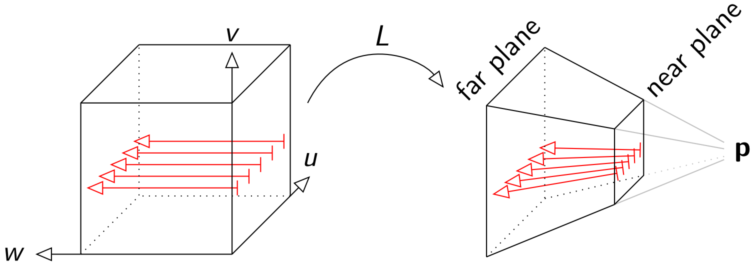
Our algorithm thus consist of the following steps:

1. Compute ray coefficients $C = (c_{pqk})$
2. For each pixel (p, q)
   2.1 Seek the smallest root $t \in [0, 1]$ of $f(\mathbf{r}_{pq}(t))$.
   2.2 Compute a color for pixel (p, q).
3. Optionally, perform postprocessing
   3.1 Detect singularities
   3.2 Antialias



The View Frustum Form

Idea: *Parameterize the view frustum over the unit cube, s. t.* $(u, v, 0)$ *and* $(u, v, 1)$ *maps to points on the near and far plane.*



A ray in the view frustum is given by: $\mathbf{r}_{pq}(w) = L(p/m, q/n, w)$.
We define the View Frustum Form to be:

$$g = f \circ L : [0, 1]^3 \rightarrow \mathbb{R}.$$

Using the composition $g = f \circ L$,

$$f(L(\frac{p}{m}, \frac{q}{n}, w)) = g(\frac{p}{m}, \frac{q}{n}, w) = \sum_{i,j,k=0}^{d,d,d} g_{ijk} B_i^d(\frac{p}{m}) B_j^d(\frac{q}{n}) B_k^d(w)$$

$$= \sum_{k=0}^{d} \underbrace{\left(\sum_{i,j=0}^{d,d} g_{ijk} B_i^d(\frac{p}{m}) B_j^d(\frac{q}{n}) \right)}_{c_{pqk}} B_k^d(w).$$

Yielding univariate ray equations of degree d,

$$f(\mathbf{r}_{pq}(t)) = \sum_{k=0}^{d} c_{pqk} B_k^d(t).$$

Computing VFF Coefficients

The VFF coefficients $G = (g_{ijk})$ can be found in a number of ways:

- Blossoming [DeRose et.al. 1993].
- Recursion [Sederberg/Zundel 1989].
- Interpolation (our preferred approach).
  - Choose $(d+1)^3$ distinct interpolation points (u_p, v_q, w_r) on a grid.
  - Solve

$$\sum_{i,j,k=0}^{d,d,d} g_{ijk} \underbrace{B_i^d(u_p)}_{\Omega_p} \overbrace{B_j^d(u_q)}^{\Omega_q} \underbrace{B_j^d(u_r)}_{\Omega_r} = f(L(u_p, v_q, w_r))$$

  - Needs inverse of Bernstein collocation matrices $\Omega_p = (B_i^d(u_p))$.
  - Use Chebyshev interpolation points for stability.
  - Not dependent on the representation of f.

Benefits of the View Frustum Form

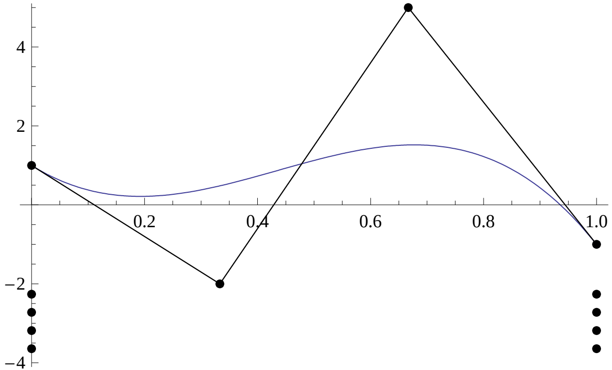
- For fixed m, n, d – precompute basis functions
  - $C_k = MG_kN^T, \quad M = (B_i^d(p/m)), N = (B_j^d(q/n))$.
  - Can pre-evaluate inverse of collocation matrix Ω^{-1}.
- Reduce algorithmic complexity
  - Evaluation of c_{pqk} requires $(d+1)^2$ muls/adds.
  - Evaluation of f requires $(d+1)(d+2)(d+3)/6$ muls/adds.
- Univariate ray equations for root finding.

$$f(\mathbf{r}_{pq}(t)) = \sum_{k=0}^{d} c_{pqk} B_k^d(t)$$

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

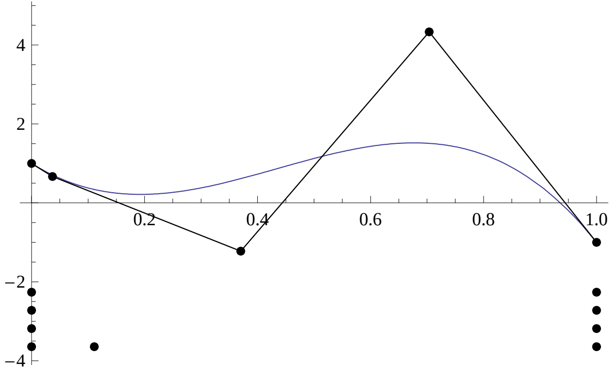


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

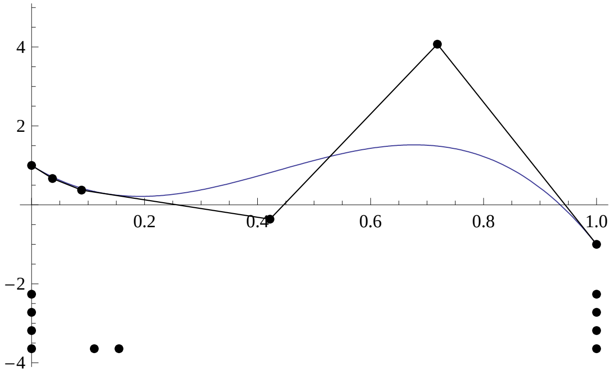


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

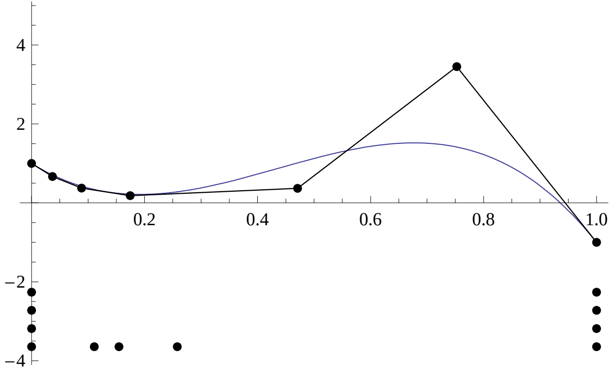


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

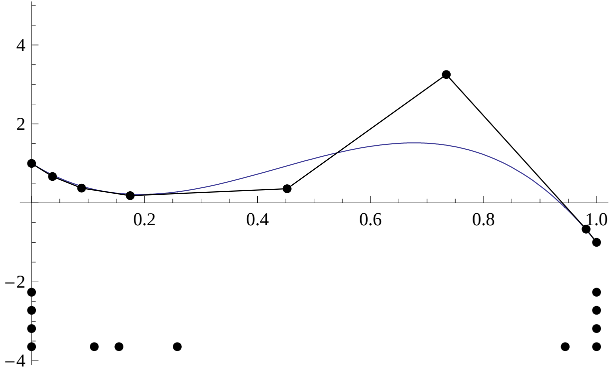


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

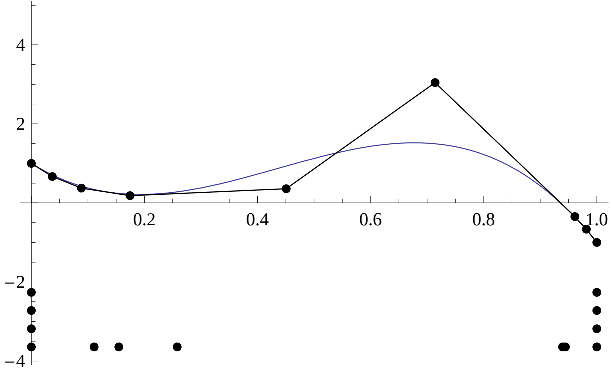


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

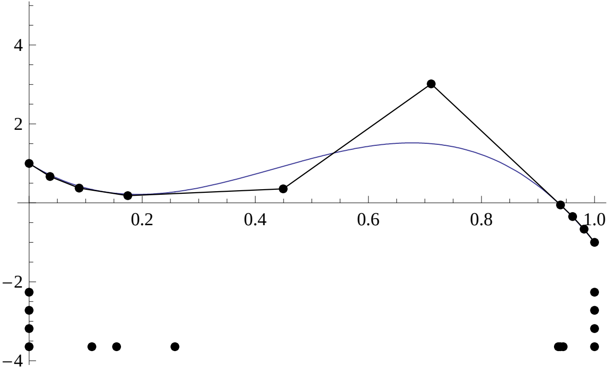


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*

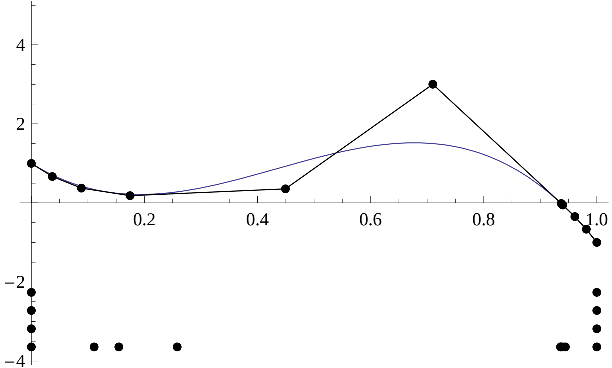


Second order convergence for simple roots [Mørken and Reimers, 2006].

B-Spline Based Root Finding

The univariate ray equations can be expressed as B-Splines.

Idea: *Insert knot(s) at the first intersection of the control polygon. Repeat.*



Second order convergence for simple roots [Mørken and Reimers, 2006].

Properites of rootfinder

- Stable computations (convex combinations)
- Similar to Newtons, but unconditionally convergent to smallest zero (no guessing)
- Quadratic convergence rate to simple zeros
- Recent extension [Mørken/Reimers] converge quadratically to multiple zeros
- Similar method [Reimers] for computing e.g. $\max f(x)$ or $\min |f(x)|$

Root finding variations

The knot insertion framework is very flexible, and allow for variations:

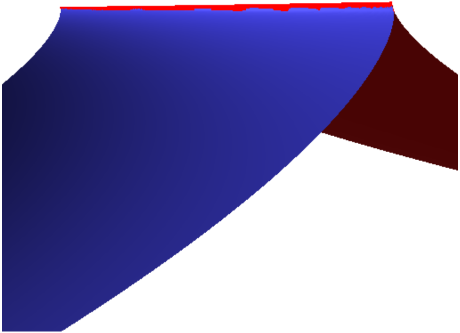
- Can emulate Bézier subdivision by inserting d knots at a time. (Lane/Risenfeld, Rockwood, Schneider).
- “Preconditioning”, insert knots from neighboring rays.
- Estimate root multiplicity and detect roots of n'th derivative. (Strictly alternating control polygon.)
- Detect critical points, use as start value to search for singularities.

Singularity detection

1. For misses, find smallest absolute value along ray, w_0.
2. Flag as singularity if:

$$|g(p/m, q/n, w_0)| + \| \nabla g(p/m, q/n, w_0) \| < \epsilon.$$

- How to determine ϵ?
- Vulnerable to scaling.

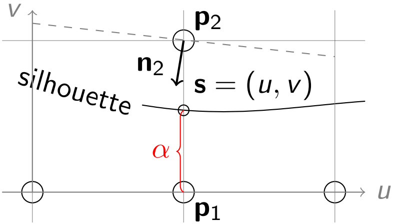


$x^2 - y^3 = 0.$

Antialiasing

Due to discrete sampling, aliasing effects will occur.

- Suppose neighboring pixels $\mathbf{p}_1$, $\mathbf{p}_2$ differ.
  - I.e. $\nabla\mathbf{p}_1 \cdot \nabla\mathbf{p}_2 < \epsilon$.
- We seek a point $\mathbf{s}$ on the separating curve between $\mathbf{p}_1$ and $\mathbf{p}_2$.
- $color(\mathbf{p}_1) :=$
  $(1.0 - \alpha)\ color(\mathbf{p}_1) + \alpha\ color(\mathbf{p}_2)$



Antialiasing II

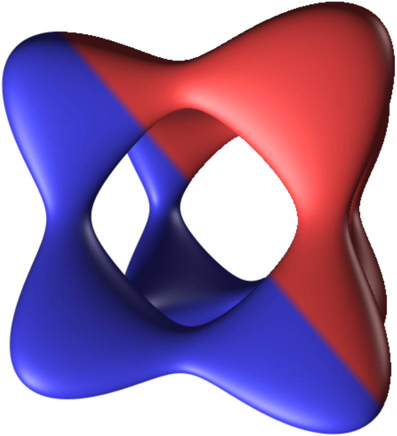
At silhouettes

$$g(\mathbf{s}) = 0 \text{ and } g_w(\mathbf{s}) = 0.$$

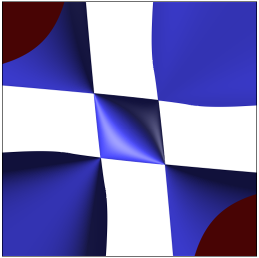
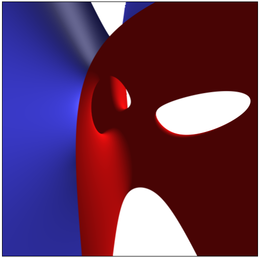
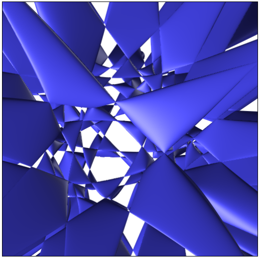
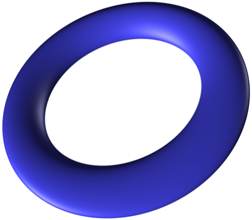
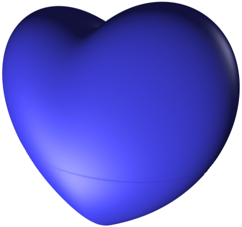
- Use Newton methods on

$$\mathbf{h}(v, w) := (g(\mathbf{s}), g_w(\mathbf{s})) = (0, 0).$$

- Restrict to plane between $\mathbf{p}_1, \mathbf{p}_2$.
- If leaving domain, search for $g(\mathbf{s}) = 0$.



Gallery and performance


Future work

- Interval spline methods:
  - Topological correctness.
  - Empty-space skipping.
- Bounding box calculations
- Efficient data structures for splines

Thank you for listening

Questions?

Contact info

- Johan S. Seland
- <johan.seland@sintef.no>
- Tel: +47 97 18 16 14

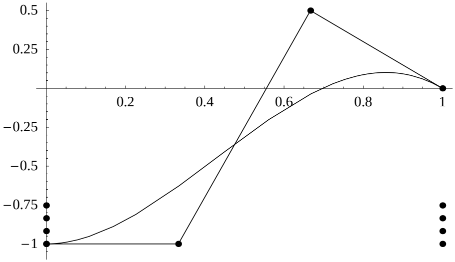
Mørken and Reimers
An unconditionally convergent method for computing zeros of splines and polynomials
Math. of Comp. 76, 2006

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



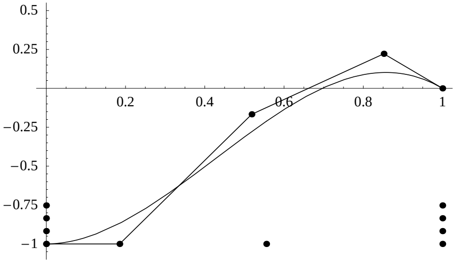
Error $|x_j - z|$: 1.41e-1

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



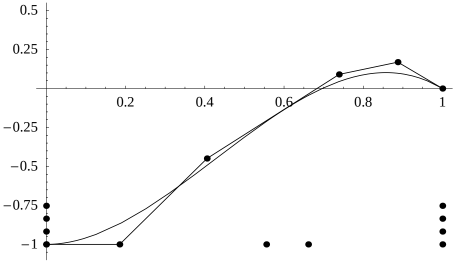
Error $|x_j - z|$: 1.41e-1 3.48e-2

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



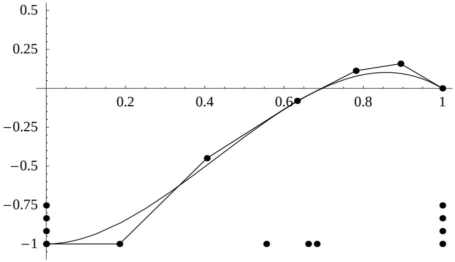
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



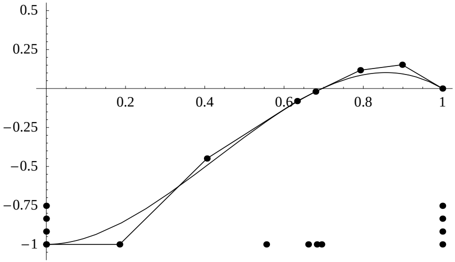
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



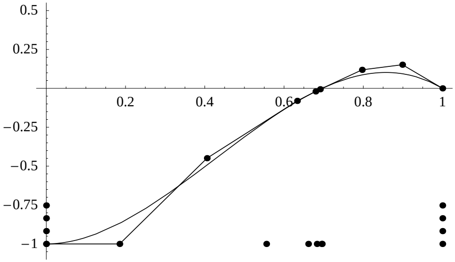
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3 1.46e-4

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



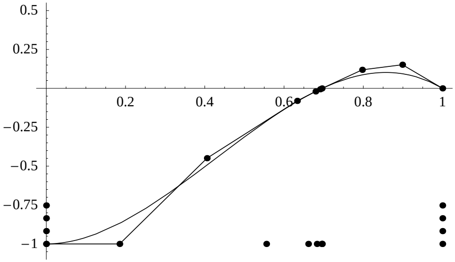
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3 1.46e-4 1.42e-6

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



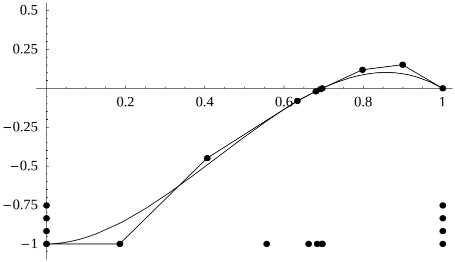
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3 1.46e-4 1.42e-6 1.70e-8

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



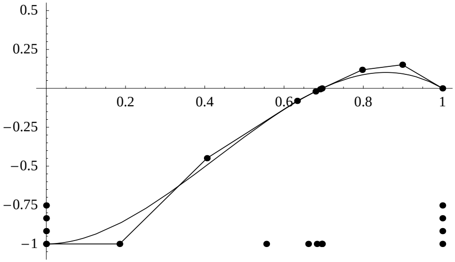
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3 1.46e-4 1.42e-6 1.70e-8 1.61e-12

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



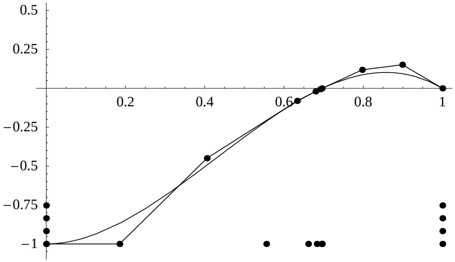
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3 1.46e-4 1.42e-6 1.70e-8 1.61e-12 2.30e-16

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



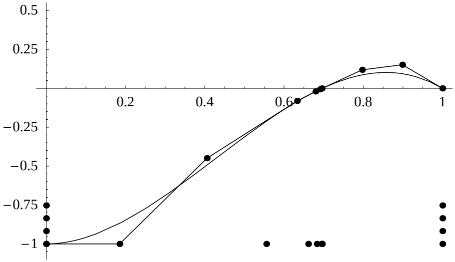
Error $|x_j - z|$: 1.41e-1 3.48e-2 1.31e-2 1.26e-3 1.46e-4 1.42e-6 1.70e-8 1.61e-12 2.30e-16 2.08e-24

Zero Algorithm [MrkenReimers 2007]

Idea: Repeated knot insertion at zeros of $F_{\mathbf{t}}$

Repeat for $j = 0, 1, \cdots$ until convergence or $F_{\mathbf{t}^j}$ has no zeros

1. Find the smallest value x_{j+1} such that $F_{\mathbf{t}^j}(x_{j+1}) = 0$ or stop
2. Let $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and form $F_{\mathbf{t}^{j+1}}$



Error $|x_j - z|$: **1.41e-1** 3.48e-2 **1.31e-2** 1.26e-3 **1.46e-4** 1.42e-6 **1.70e-8** 1.61e-12 **2.30e-16** 2.08e-24

UNIVERSITETET I OSLO

Centre of Mathematics for Applications