INTERVAL NUMERICAL METHODS FOR FIXED POINTS

José Eduardo de Almeida Ayres · Luiz Henrique de Figueiredo

Finding the fixed points of a function is important in many contexts. For instance, solving nonlinear equations is frequently cast as finding fixed points. Newton's method is probably the main example of this formulation. Fixed points, and more generally periodic points, are also important in discrete dynamical systems, especially in complex dynamics, where periodic orbits play a key role. There is a large literature on interval methods for solving nonlinear equations, but surprisingly very little that is specific to fixed points.

Let $f\colon \Omega \subseteq \mathbb{R}^d \to \mathbb{R}^d$ be a continuous function defined on a box Ω. We describe a rigorous numerical method based on interval analysis for finding all fixed points of f: attracting, repelling, and indifferent. We specialize this method for finding all attracting periodic points of a complex polynomial.

Our algorithm is a divide-and-conquer algorithm that recursively subdivides Ω and discard boxes that cannot contain a solution to isolate fixed points within a given tolerance ε. Our algorithm is both spatially adaptive, because its search is guided by the location of the fixed points of f, and analytically adaptive because its search is also guided by the nature of the fixed points of f.

Interval analysis is the main tool for rigorous numerical computation. It is based on interval arithmetic, an extension of ordinary arithmetic operations and standard elementary functions to intervals. The basic fact in interval analysis is that for each function $f\colon \Omega \subseteq \mathbb{R}^d \to \mathbb{R}$ expressed by a formula or an algorithm, there is a computable function F automatically built from the expression of f, called the natural interval extension of f, such that $F(X)$ is an interval that estimates the whole range of values taken by f on a box $X \subseteq \Omega$:

$$F(X) \supseteq f(X) = \{f(x) : x \in X\}$$

Finding the exact range $f(X)$ is a hard problem in general. Therefore, the inclusion $F(X) \supseteq f(X)$ is usually proper and interval estimates are usually overestimates. Nevertheless, the estimates $F(X)$ get better as X shrinks to a point in the sense that $F(\{x\}) = \{f(x)\}$ for every $x \in \Omega$. More precisely, we have at least linear convergence for interval estimates: $\operatorname{diam}(F(X)) \leq c \operatorname{diam}(X)$ for some c that depends only on f. Thus, interval methods are typically divide-and-conquer methods that recursively explore the domain of f, getting better information about f as they refine the subdivision, and discarding boxes that cannot contain a solution. For instance, when finding the zeros of f in Ω, we can discard a box X whenever $0 \notin F(X)$. This is a computational proof that f has no zeros in X. However, because of overestimation, we cannot conclude that f has a zero in X when $0 \in F(X)$. In this case, we subdivide X and recursively test the pieces.

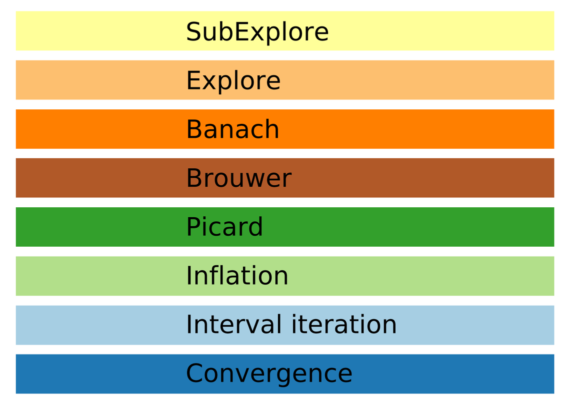
Automatic differentiation is the perfect companion for interval arithmetic and works in a similar fashion. It automatically converts an expression for f into an algorithm that simultaneously computes the value of f and of all its partial derivatives. When fed intervals instead of numbers, this algorithm computes interval estimates for the value of f and of all its partial derivatives. This allows us to reason reliably about both the range of values of f and its regions of monotonicity.

Interval arithmetic and automatic differentiation allow us to check the hypotheses of the fixed-point theorems rigorously in a computer. The existence of fixed points in a box X guaranteed by Brouwer's theorem follows whenever $F(X) \subseteq X$ because then $f(X) \subseteq F(X)$ implies $f(X) \subseteq X$. The existence of a unique fixed point in a box X guaranteed by Banach's theorem follows whenever $F(X) \subseteq X$ and $\|F'(X)\| < 1$ because these imply that f is a contraction in X, thanks to the mean value inequality. Here, F' is an interval extension of the Jacobian matrix of f, which can be computed with automatic differentiation.

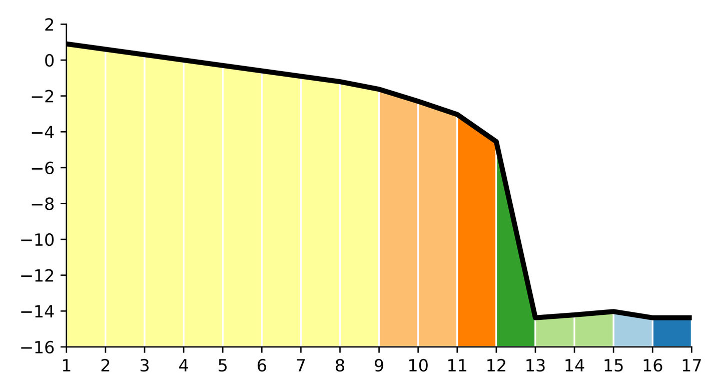
ALGORITHM

```
procedure Explore(X)
  W, W′ ← X, 1
  for k = 1 to n do
    W, W′ ← F(W), F′(W)W′
    if W is outside the escape disk then
      discard X
    end
  end
  X′ ← X ∩ W
  if X′ = ∅ or ‖W′‖ ≥ 1 then
    discard X
  else if diam(X′) < ε then
    accept X′
  else if W ⊆ X and ‖W′‖ < 1 then
    ExploreAttracting(X′)
  else if diam(X′) < λ diam(X) then
    Explore(X′)
  else
    SubExplore(X′)
  end
end
```

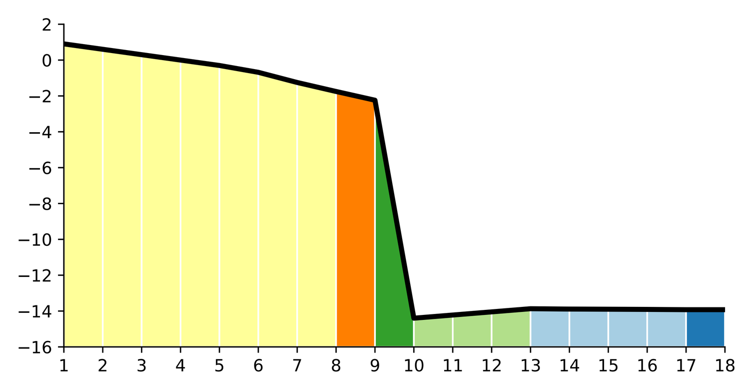
```
procedure
  ExploreAttracting(X)
  x̂ ← mid(X)
  repeat
    x̂ ← fⁿ(x̂)
  until convergence
  X ← [x̂, x̂]
  repeat
    X ← Inflate(X)
  until Fⁿ(X) ⊆ X
  repeat
    X ← Fⁿ(X)
  until convergence
  accept X
end
```



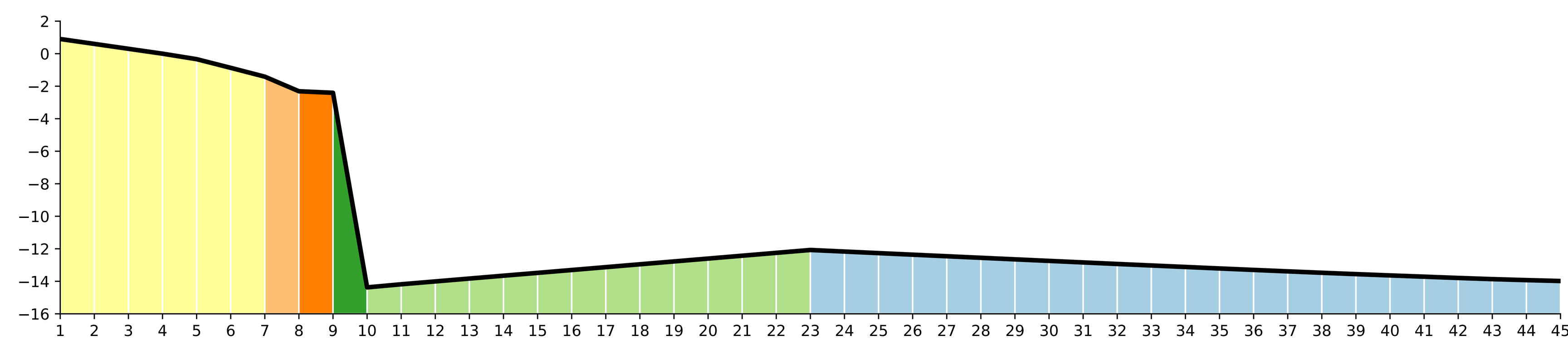
PERFORMANCE



super attracting point

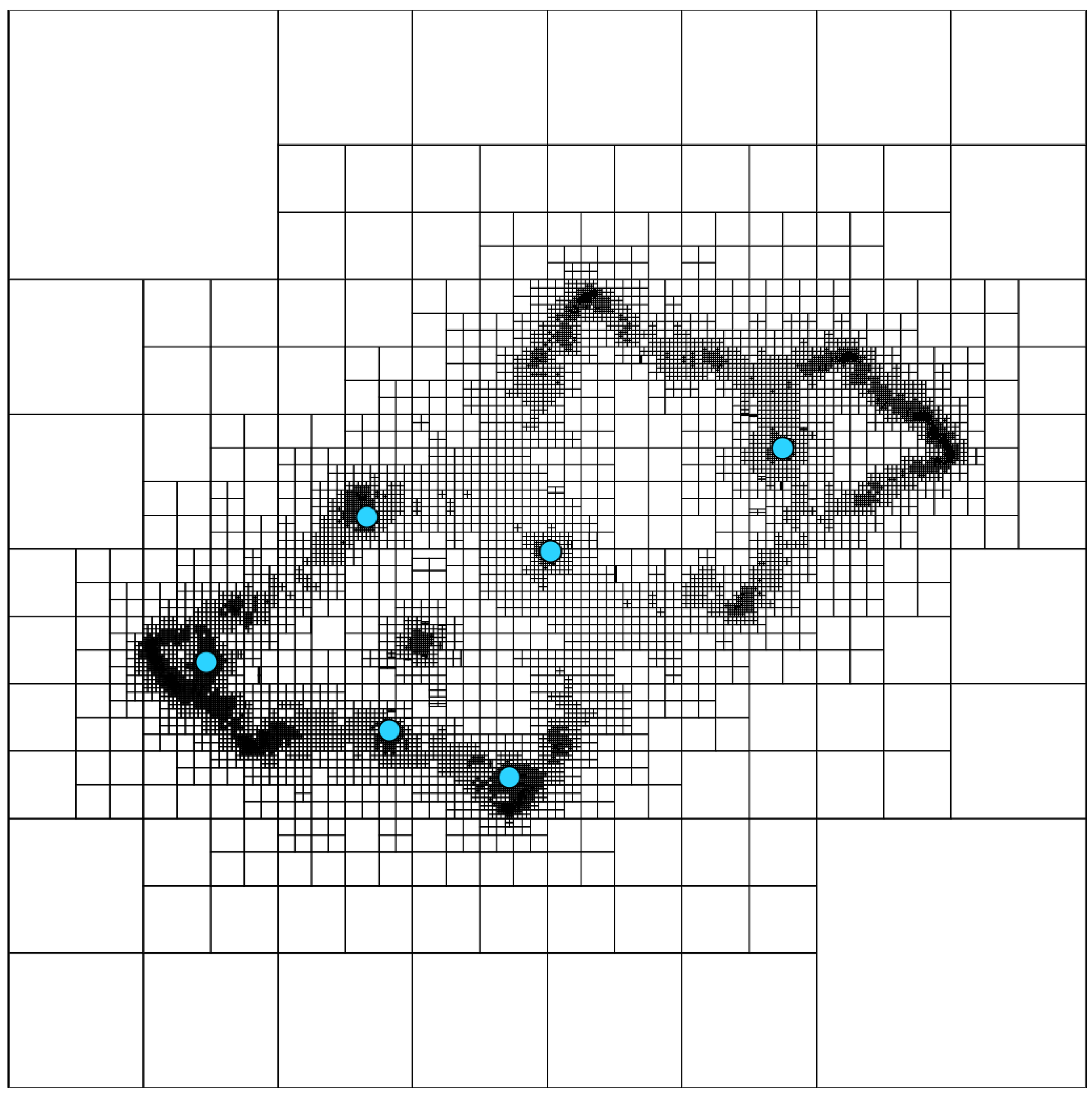


strongly attracting point



weakly attracting point

CONVERGENCE



ACKNOWLEDGEMENTS The first author is partially supported by CNPq and FAPERJ doctoral scholarships. The second author is partially supported by a CNPq research grant. This research was done in the Visgraf Computer Graphics laboratory at IMPA. Visgraf is supported by the funding agencies FINEP, CNPq, and FAPERJ, and also by gifts from IBM Brasil, Microsoft, NVIDIA, and other companies.