Building fast numerical solvers

Homotopy Continuation Tutorial

Ricardo Fabbri **Rio de Janeiro State University**

Author of MiNuS github.com/rfabbri/minus

CVPR
JUNE 17-21, 2024
SEATTLE, WA

Building fast HC solvers

Outline of Talk

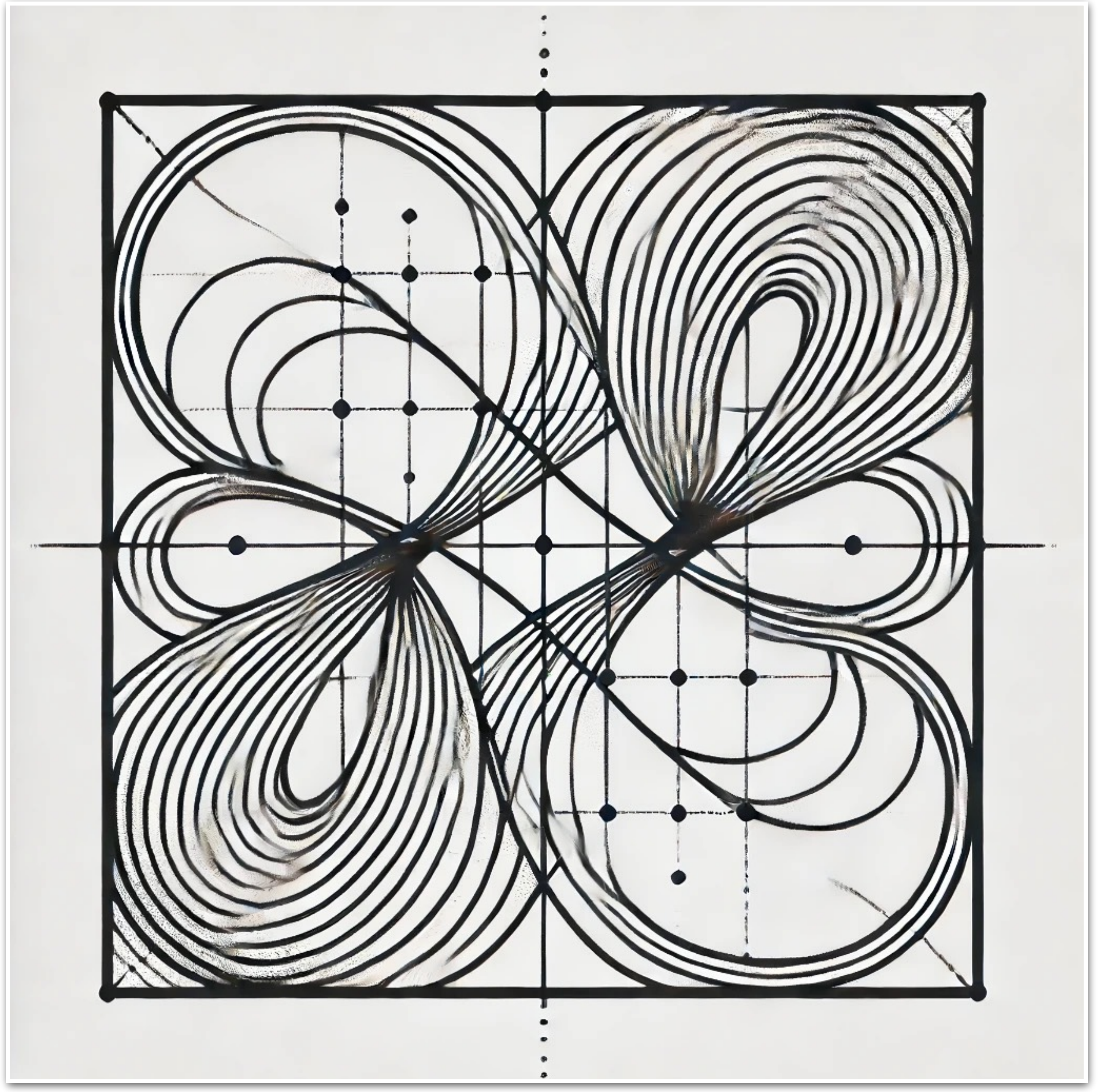
Design principles

Predictor-corrector design

- ODE Solving along levelsets: *an illustration*
- Predictors: design choices for speed
- Correctors: design choices for speed

Code-level optimizations

- MiNuS: A C++ framework for fast homotopy continuation
- Closer-look at key techniques



DALL-E 3 Prompt
“Homotopy Paths on a Square”

Fast Numerical Algorithms

Design Principles

- **Speed is of the essence** — real-time AR and autonomous cars
- **Floating point is powerful**
    - Continuous modeling to design algorithms — dynamical systems, ODEs, PDEs
    - On the rise with GPUs
- **Specialize generic algorithms but using a general *approach***
    - Numerical algorithms come too generic
    - Simplest possible algorithms = Fast but still too generic
    - Smarter algorithms highly constrained by high-speed requirement

Predictor-Corrector design

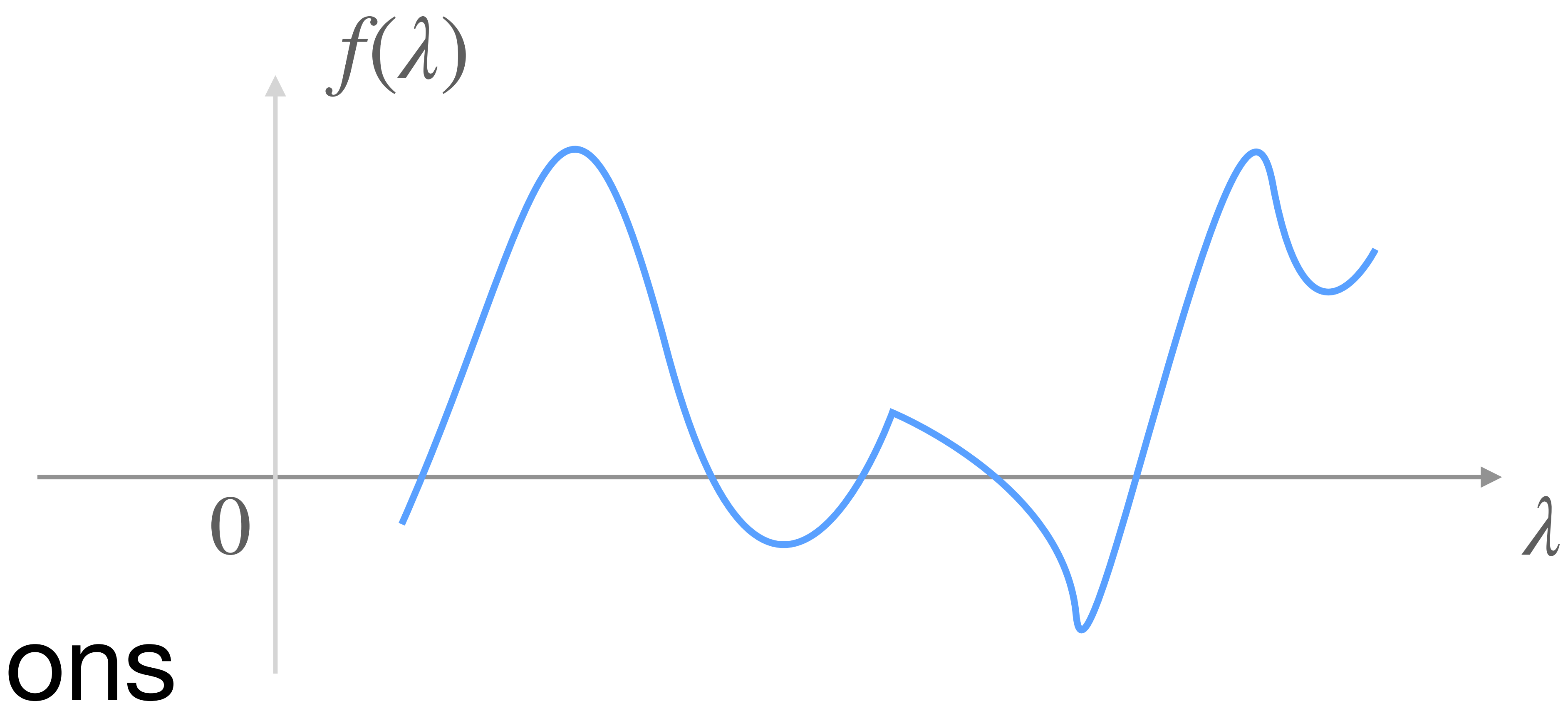
Tracing levelsets with ODE Integration

- We wish to design a fast solver for a system of nonlinear equations

$$f(\lambda) = 0$$

but for any f in a family —> leads to ODE

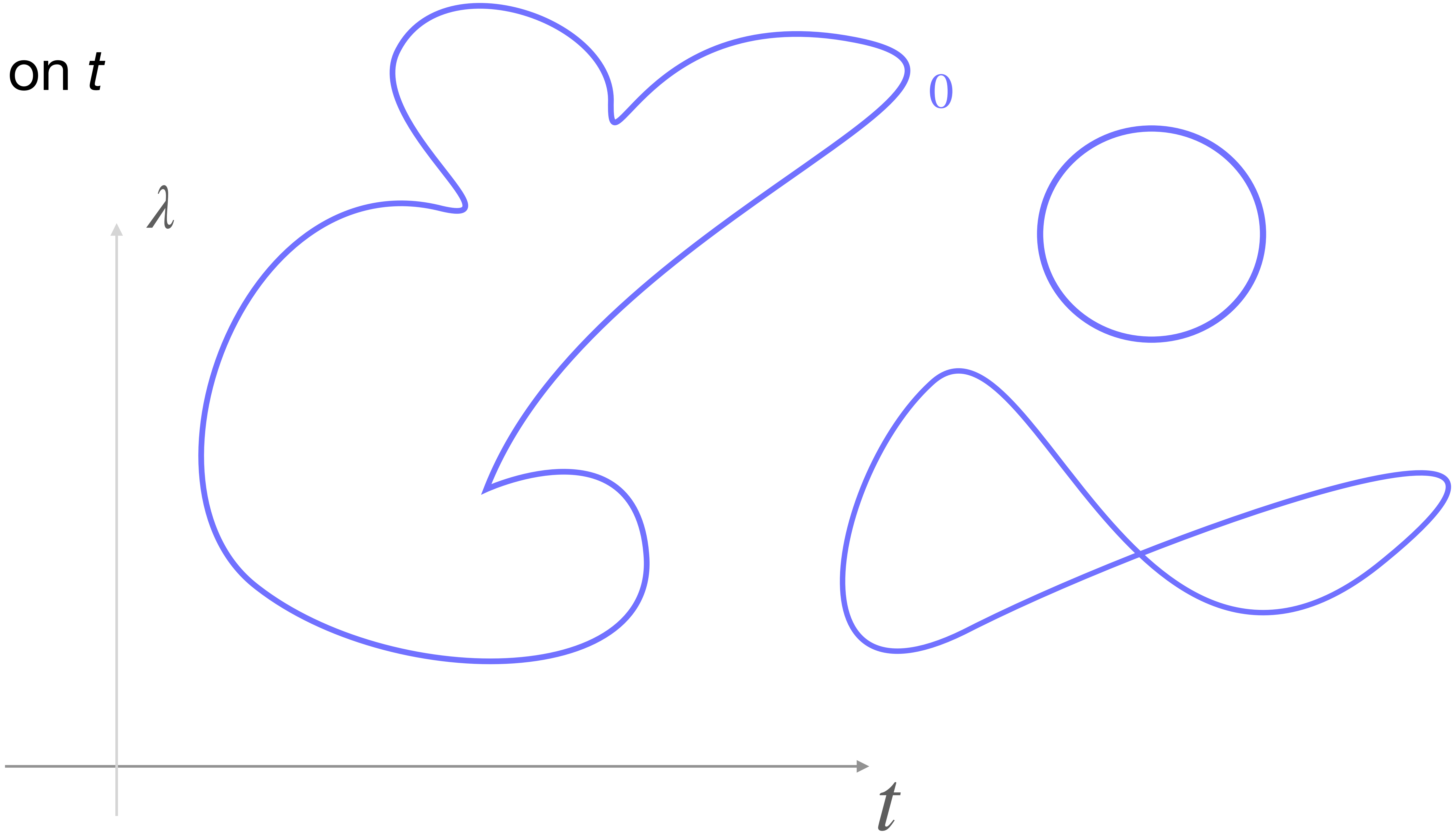
- Minimal problem —> square system
- Let us analyze the 1x1 case in $\mathbb{R}$
- For any f in the family, we have isolated solutions



Predictor-Corrector design

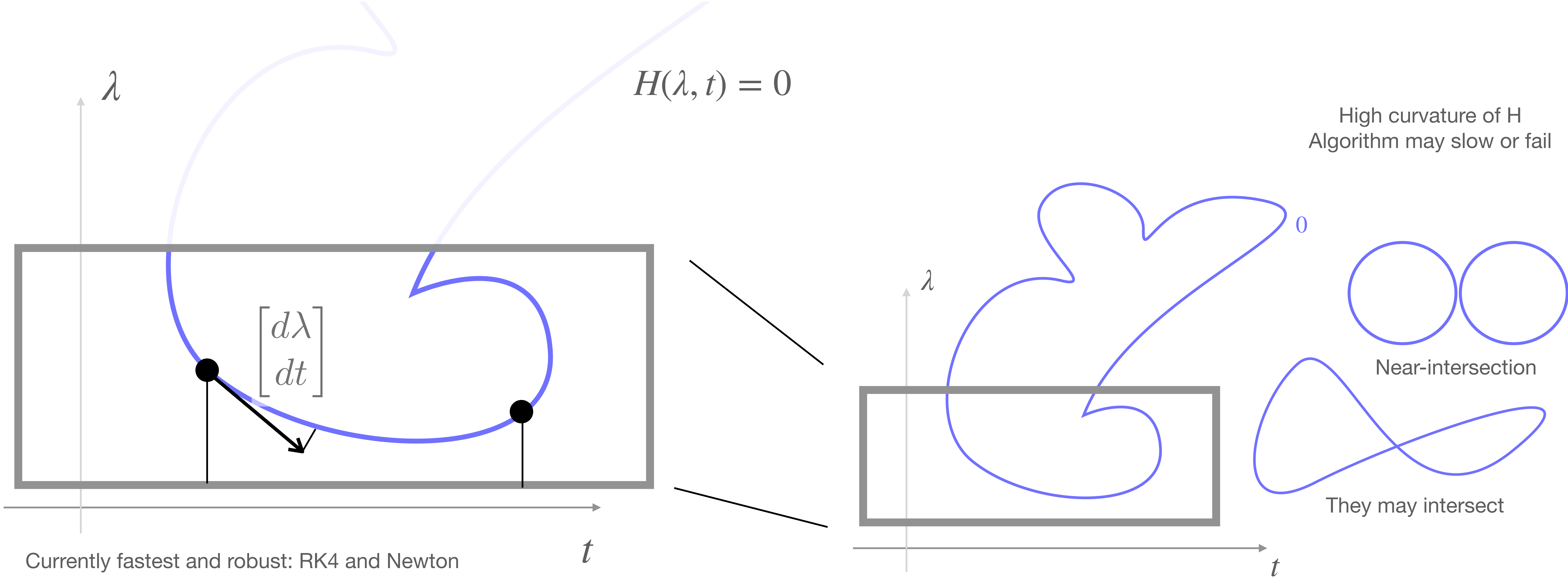
Tracing levelsets with ODE Integration

- We may *locally* trace a curve in the family of systems by introducing an extra variable *t*
- The value of f will now depend on *t*
- family f —> H($\lambda(t), t$) locally



Predictor-Corrector design

Tracing levelsets with ODE Integration



Currently fastest and robust: RK4 and Newton

Predictor-Corrector design

Tracing levelsets with ODE Integration

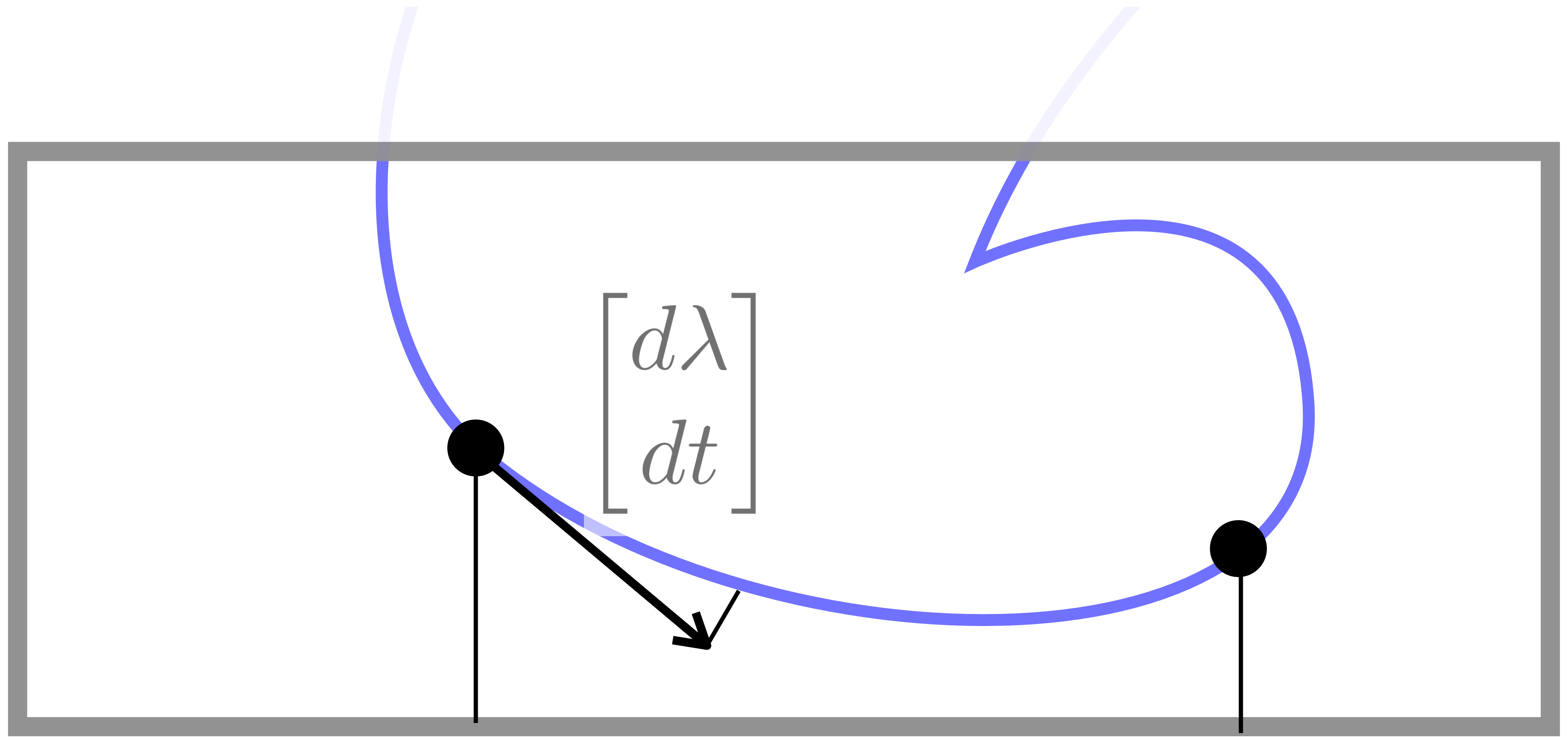
- We now build a linear approximation:

$$dH = \begin{bmatrix} \frac{\partial H}{\partial \lambda} & \frac{\partial H}{\partial t} \end{bmatrix} \begin{bmatrix} d\lambda \\ dt \end{bmatrix} = \frac{\partial H}{\partial \lambda, t} \begin{bmatrix} d\lambda \\ dt \end{bmatrix}$$

- This evaluates linear approximation in any direction
- To get ODE for levelset, write

$$\frac{\partial H}{\partial \lambda, t} \begin{bmatrix} d\lambda \\ dt \end{bmatrix} = 0$$

- dim ker dH = intersecting branches

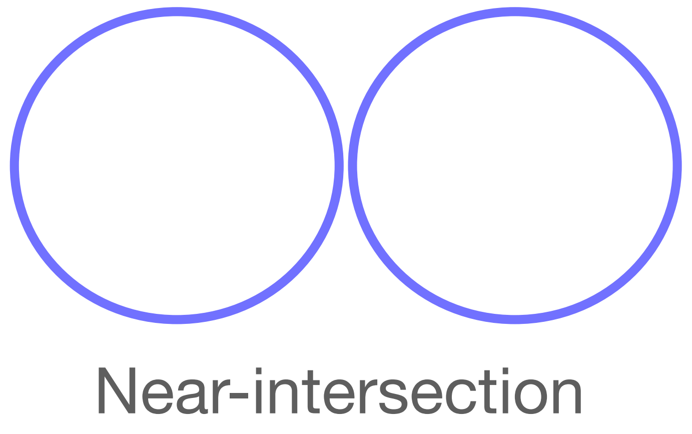


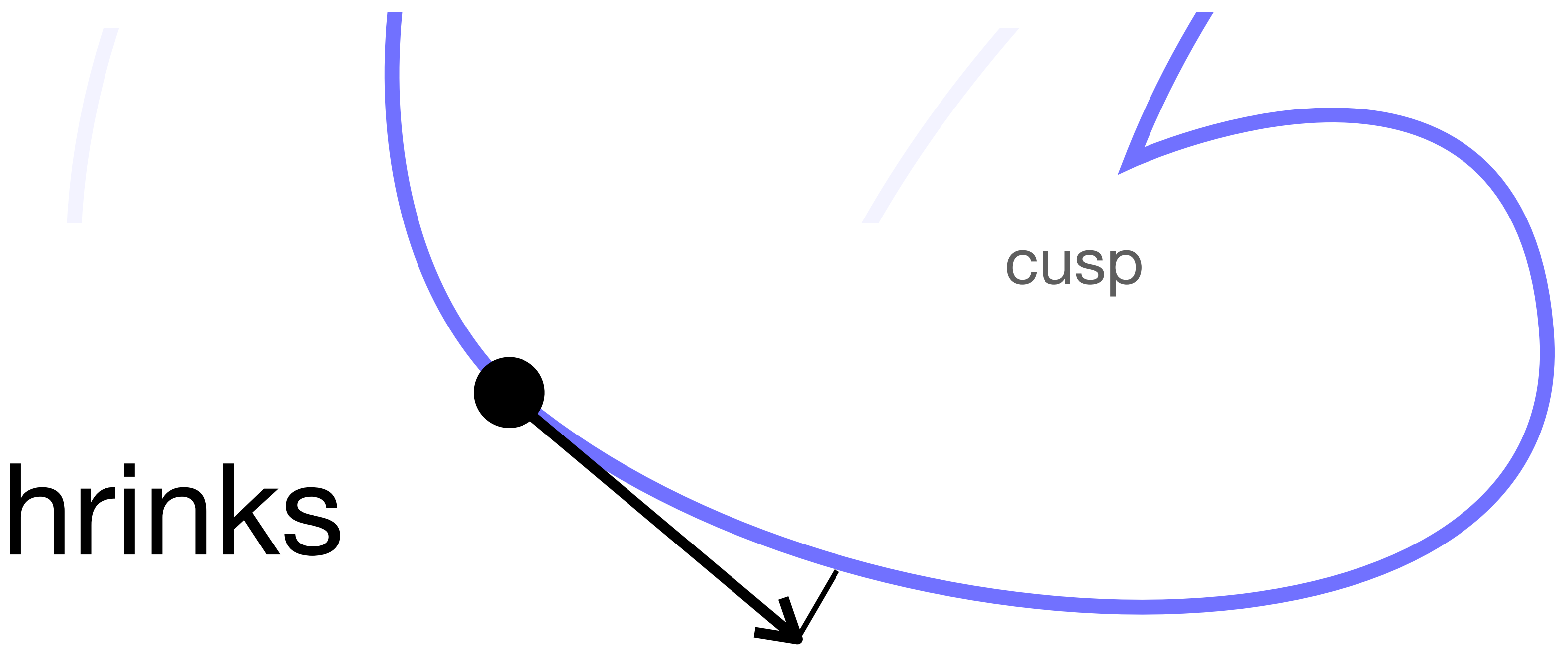
Predictor-Corrector design

Tracing levelsets with ODE Integration

$$\frac{\partial H}{\partial \lambda, t}\begin{bmatrix} d\lambda \\ dt \end{bmatrix} = 0$$

- dim ker dH = intersecting branches
- Linear approximation dH to our H gets singular
- Curvature of embedding function H gets complicated
- Rank/Condition number/determinantal conditions
- Simple numerical methods fast but slow down here
- Near high-curvature, fast convergence neighborhood shrinks
- Adaptive stepsize —> simple is too simple

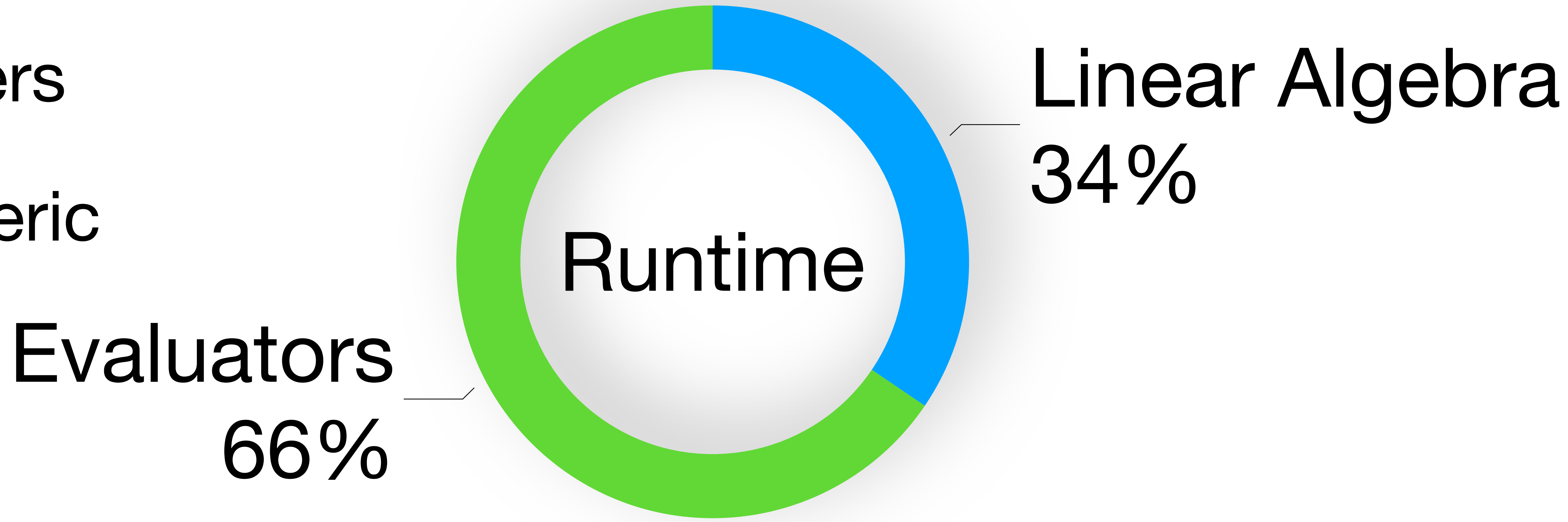




Fast Homotopy Continuation

Code-level optimizations

- MiNuS — C++ Framework for fast HC solvers
- Large trifocal problem 200x faster than generic
- Hardcoded evaluators
- Faster linear algebra
    1. Highly optimized LU from Eigen, e.g. with specialized partial pivoting
    2. Eigen vectorization finely activated for LU decomposition —> very fast
    3. Vectorization of evaluators fine-tested on compilers
- No dynamic allocations — static vectors



github.com/rfabbri/minus

— End of Part 1 —

See part 2: Building fast numerical continuation solvers