Arbbot Convergence

Stefan Loesch

December 2023

Contents

1. General issues and choices with numeric algorithms and the practical implications thereof
2. Specific issues with the optimization algorithm used

General issues with numeric algorithms

Numerical algos don’t have bugs…

- **… they are always wrong (but sometimes they are useful)**
- Specific problem: asserting the domain of convergence is a lot of work, and usually not worth the effort ex-ante
  - Subtle effects, not all of which can be easily captured
  - Necessary preconditions not met (but close enough, so it mostly works)
  - Typically spending a lot of work dealing with constellations that are unlikely to become relevant in practice
- Often better solution: monitor algo performance, and have a plan when things go wrong
  - Monitor errors, especially convergence errors
  - Employ independent checks for monitoring (eg marginal prices; RedTeam NB)
  - Have a plan to manually fix the issue in the short run whilst the long run solution is developed
- The good news: it gets better over time the more issues have been found and addressed; but you are never safe!

Qualitative vs quantitative analysis

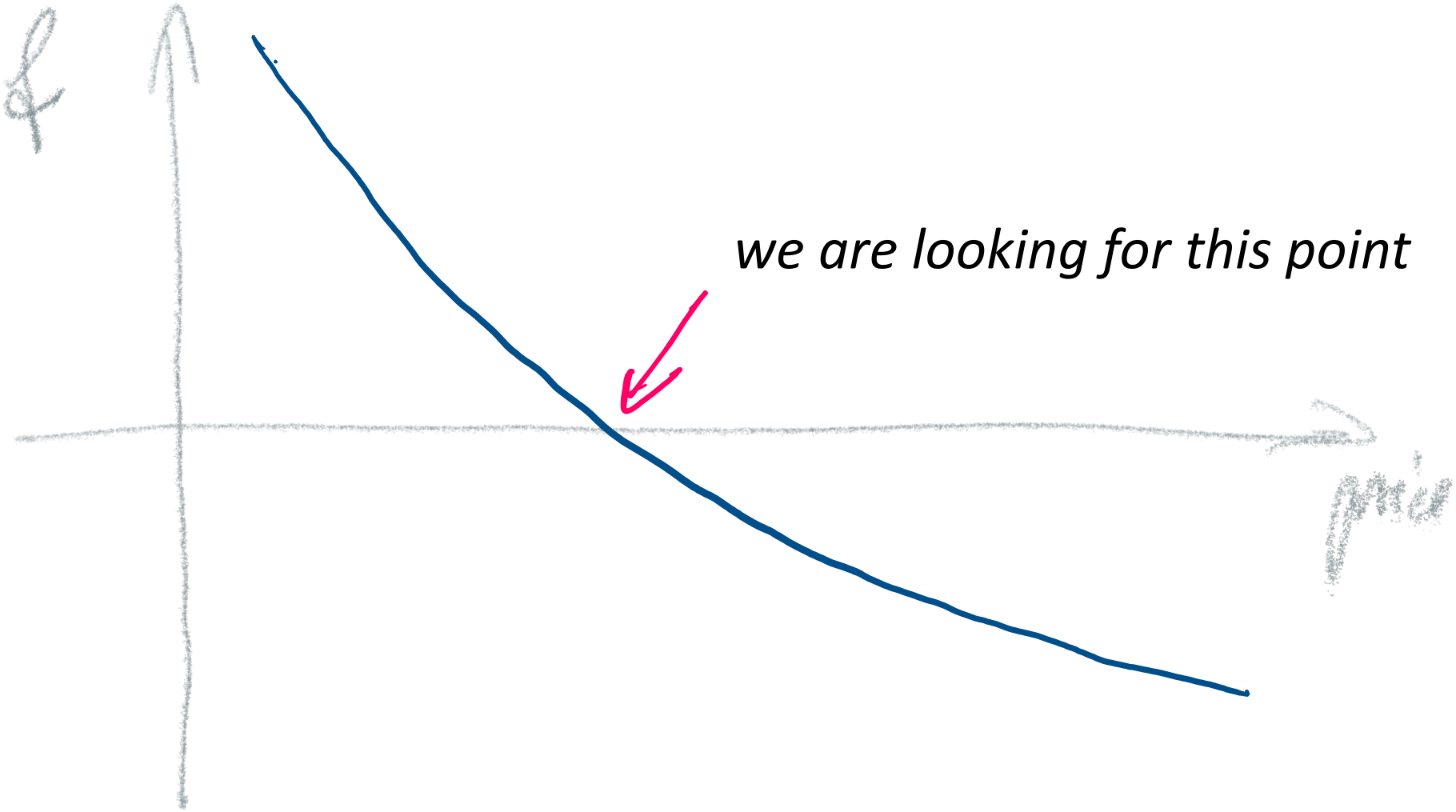
- Often we can – and should – qualitatively understand what situations can cause problems with our specific algorithm
  - Example: “large limit orders can be problematic”
- However – meta parameter choices are compromises
  - **Epsilon** (“accuracy”) – impacts runtime and may prevent convergence
  - **MaxIter** – backstop; too small and viable solution discarded; too big, and overall runtime can be excessive
  - **RP** (”regularization”) – too little has no impact, so no convergence; too much may distort the results beyond reasonable bounds
- Ex-ante meta parameter choice is hard, and may even be “NP hard” – there may be no simplification available
- **TLDR – even if qualitatively we have solved everything, from time to time we WILL run into sub-optimal meta parameter choices**

What does this mean for us specifically?

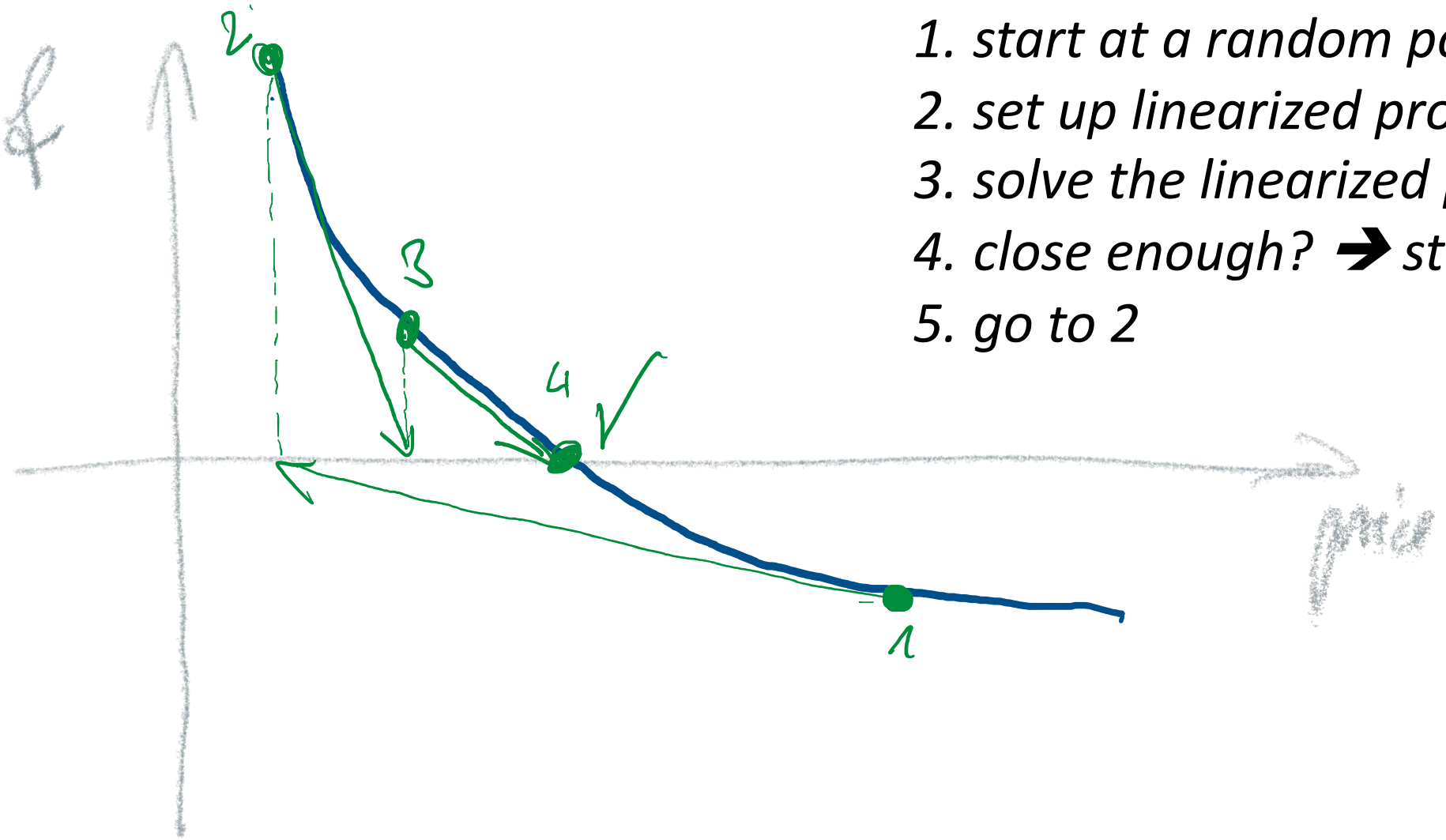
- What kind of “malfunction” is tolerable?
- Monitoring infrastructure
  - Logging
  - Analysis
  - Alerts
- Resolution protocol
  - Manual “quick-fix”
  - Ultimate
- Escalation procedures
  - Responsibilities, channels and timelines

Our specific optimization issue

The problem: find p for which f(p)=0

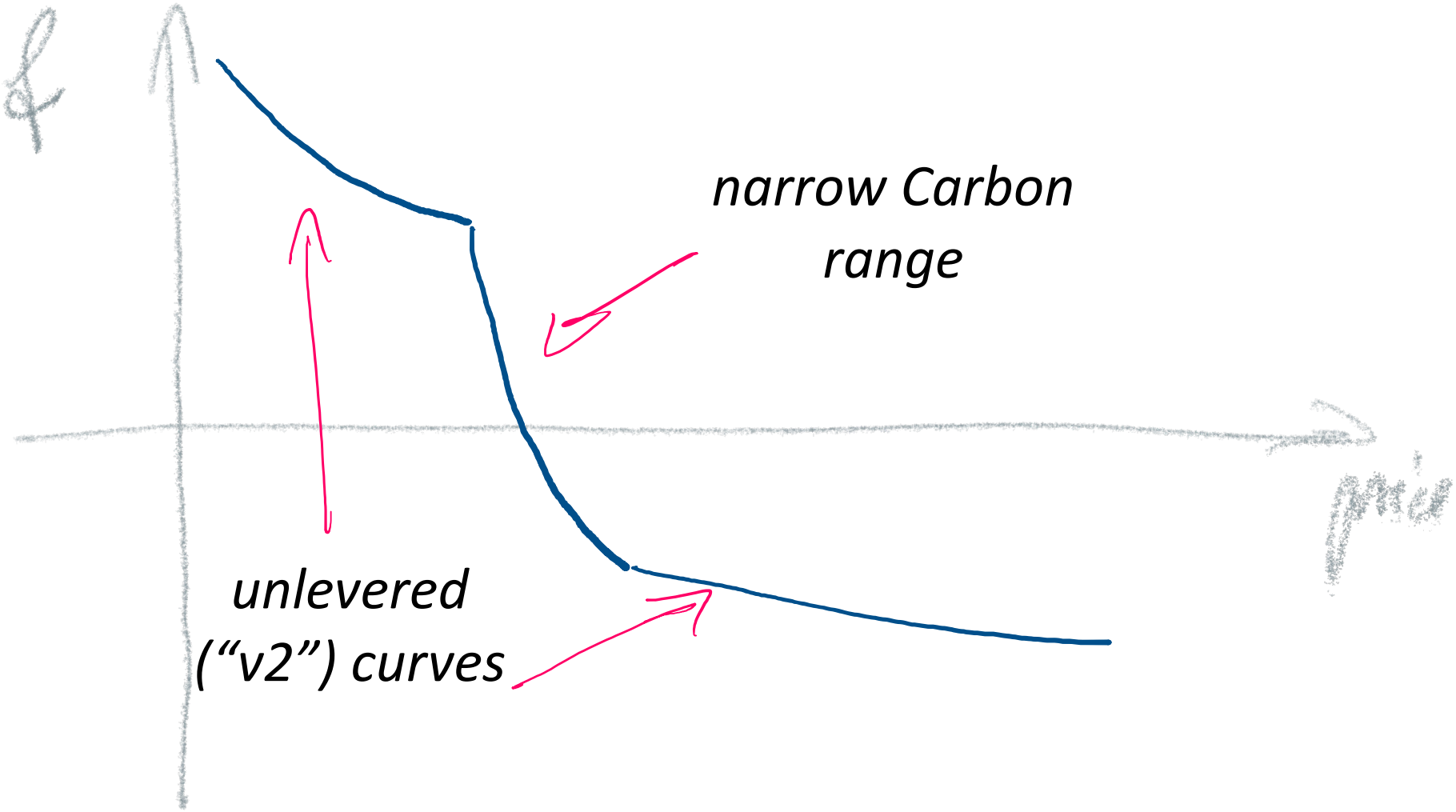


Newton Raphson: use linear approximation

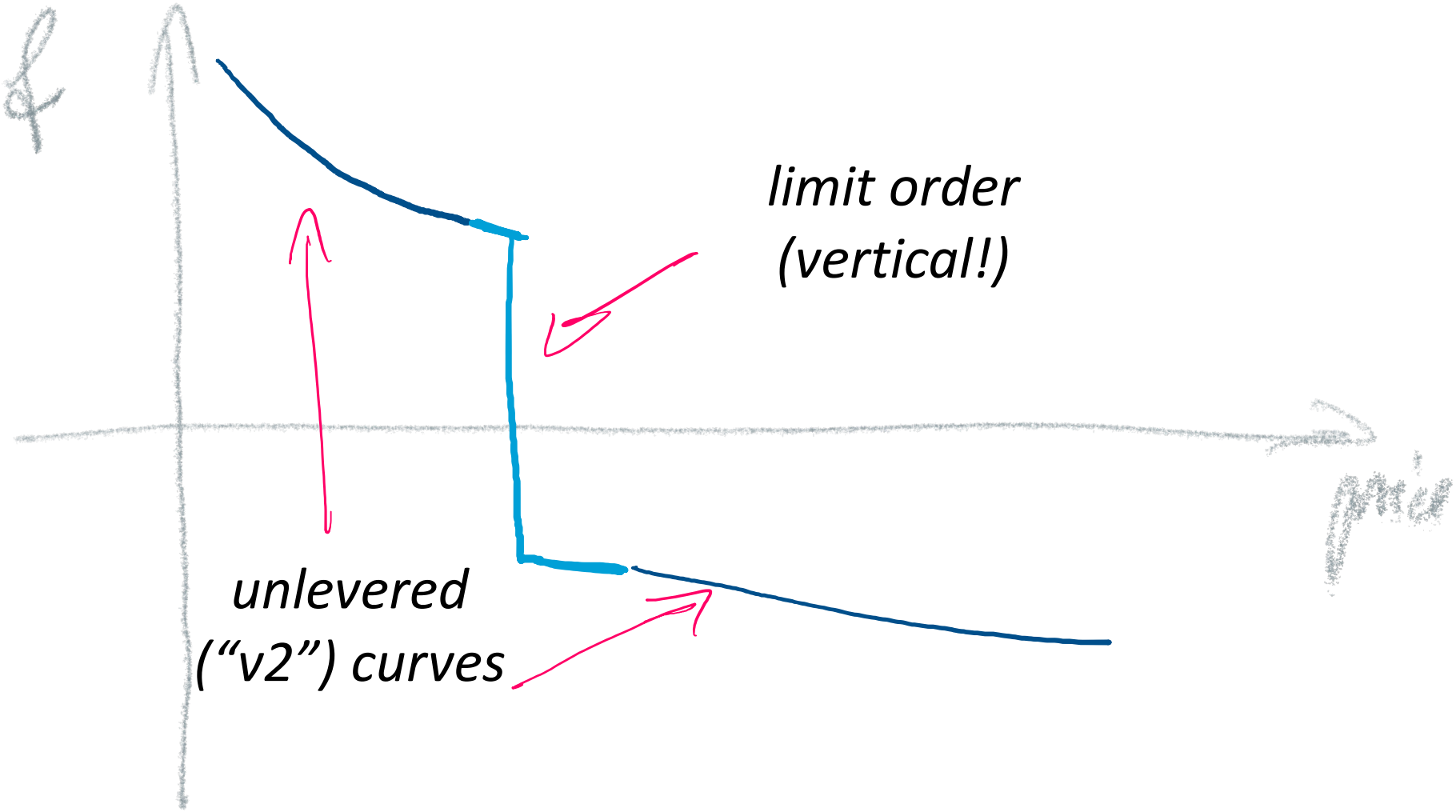


1. *start at a random point*
2. *set up linearized problem*
3. *solve the linearized problem*
4. *close enough? ➔ stop*
5. *go to 2*

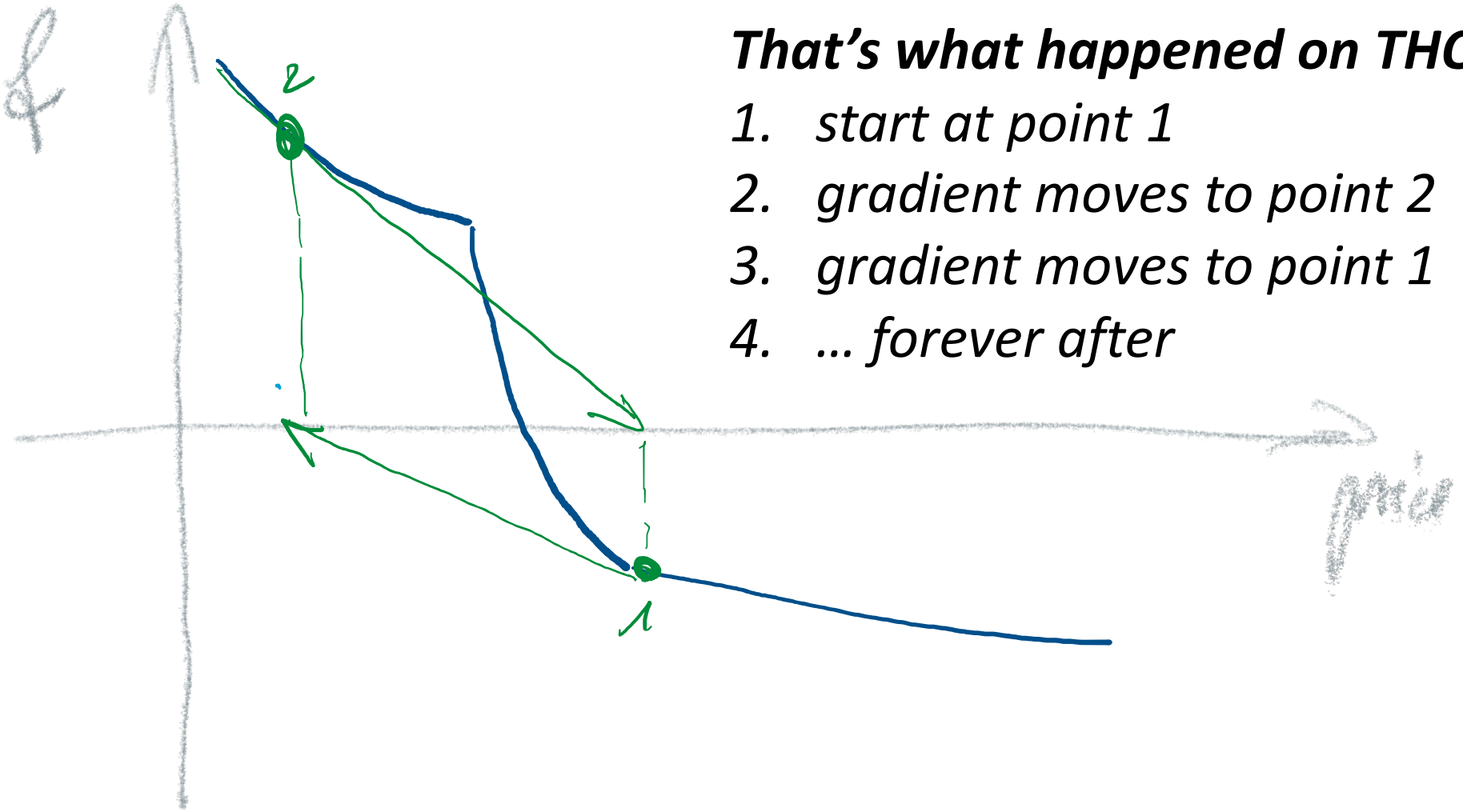
Our problem is a bit tricky on narrow ranges…



…in fact, it even gets worse on limit orders



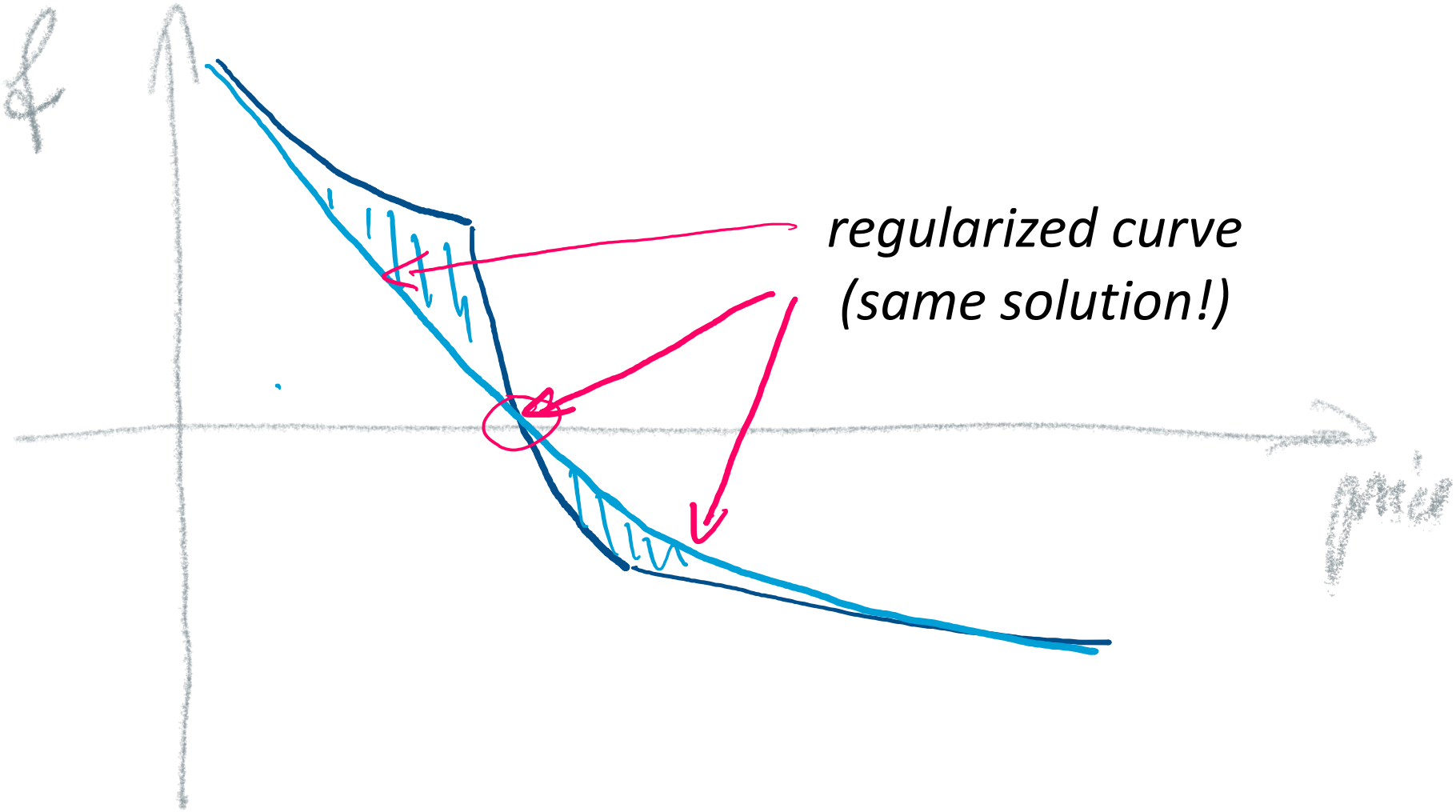
That’s what can happen – the never ending story



That’s what happened on THOR:

1. *start at point 1*
2. *gradient moves to point 2*
3. *gradient moves to point 1*
4. *… forever after*

Regularization: easier curve, same results

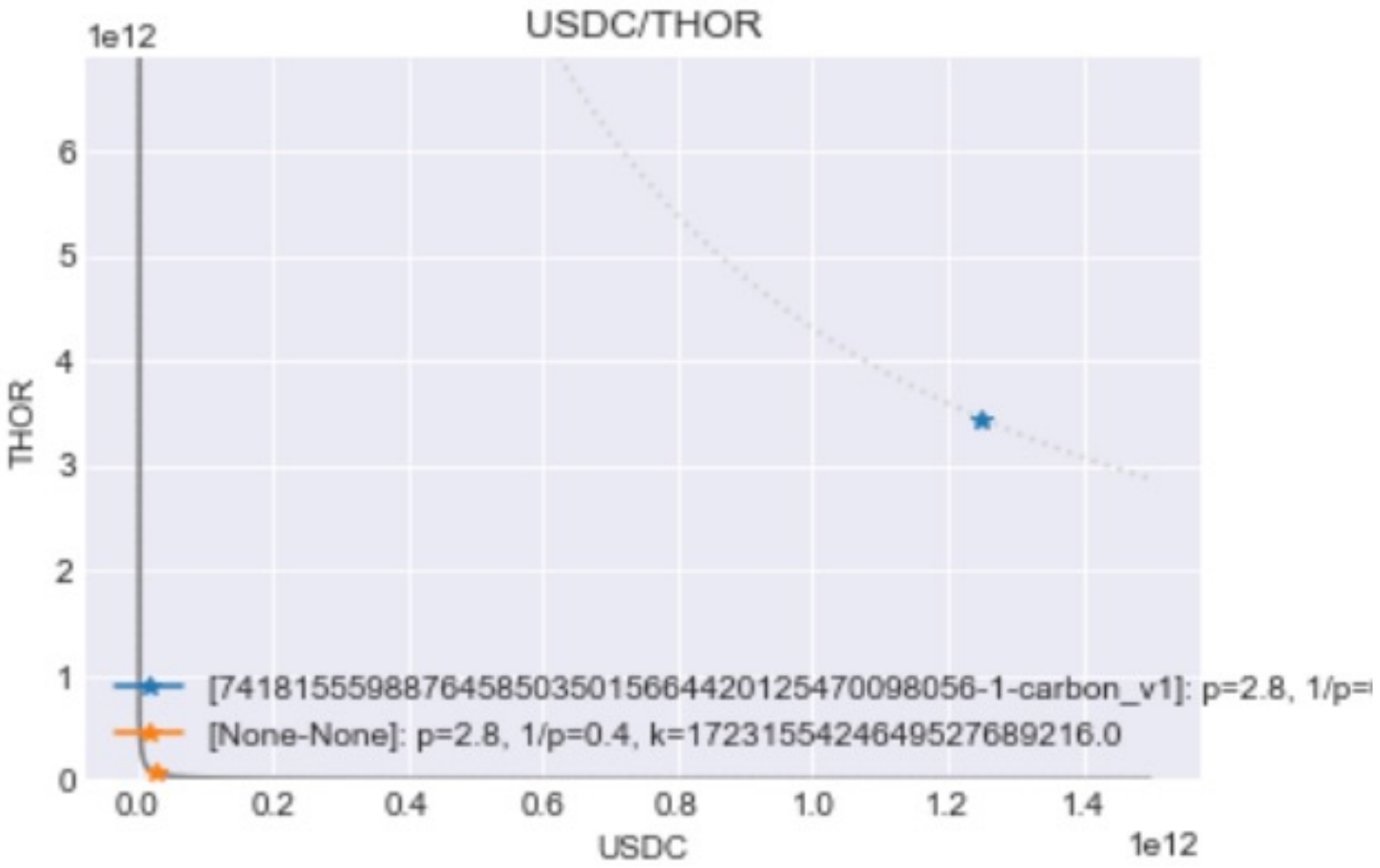


Regularization methods

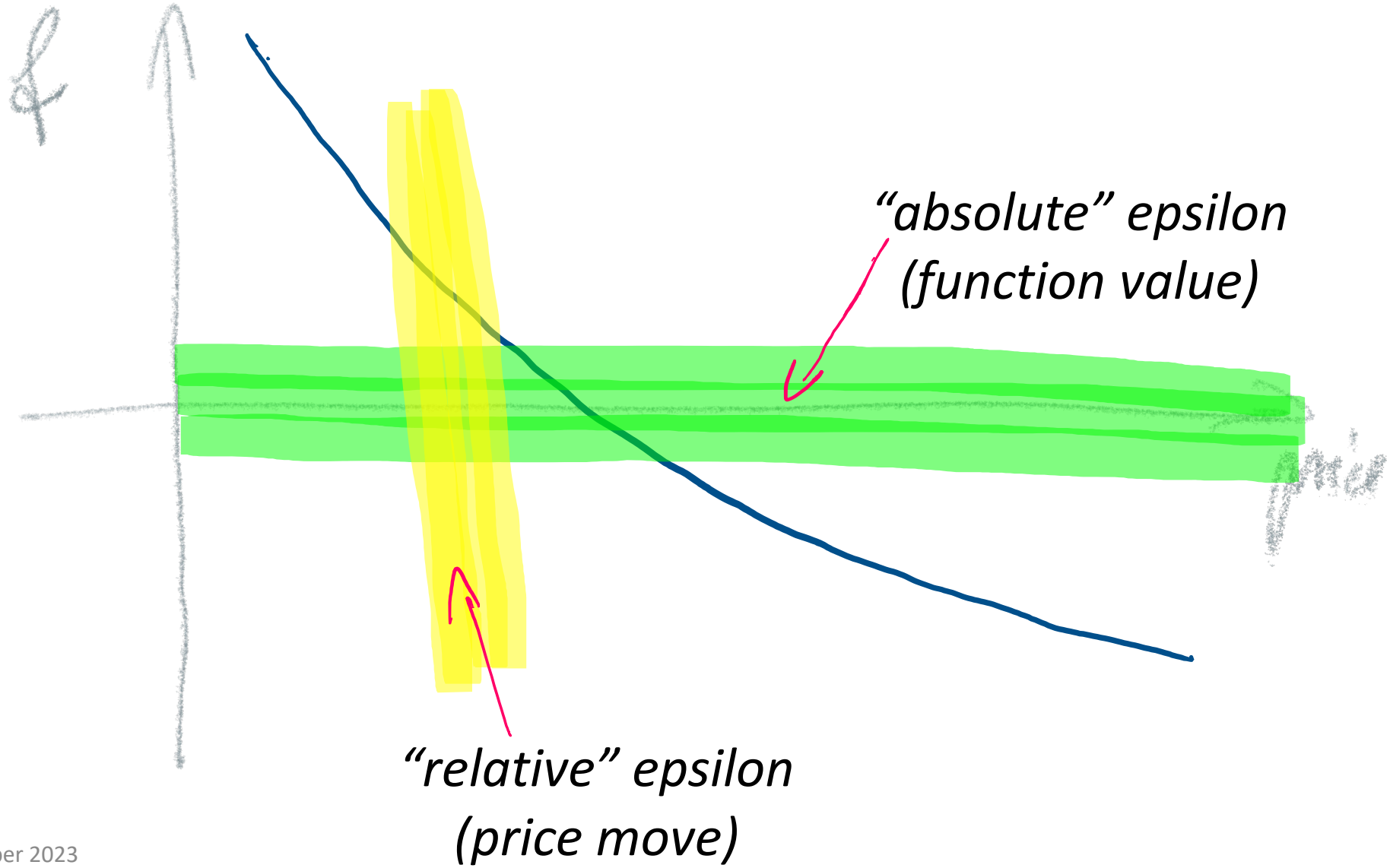
- **Minimum range width (required):** use a very narrow Carbon Range (width 1e-3 to 1e-6) to represent a limit order
  - There is no other way representing a limit order in a xy=k framework (because a proper limit order has k=infty)
  - It's a good start, but unless the range is very wide (10%?) this seems not enough to ensure convergence
- **Sentinel curve**: add an unlevered ("v2") curve with the same parameters as the limit order (margp = limit price; volume scaled down by a factor of [50])
  - The sentinel curve guides the algo over the entire range and improves chances of convergence
  - If sentinel curve is too small convergence may still not happen
  - If too big then it will impact the other curves and post-processing will become harder to do (or yield suboptimal result)

Regularization in practice

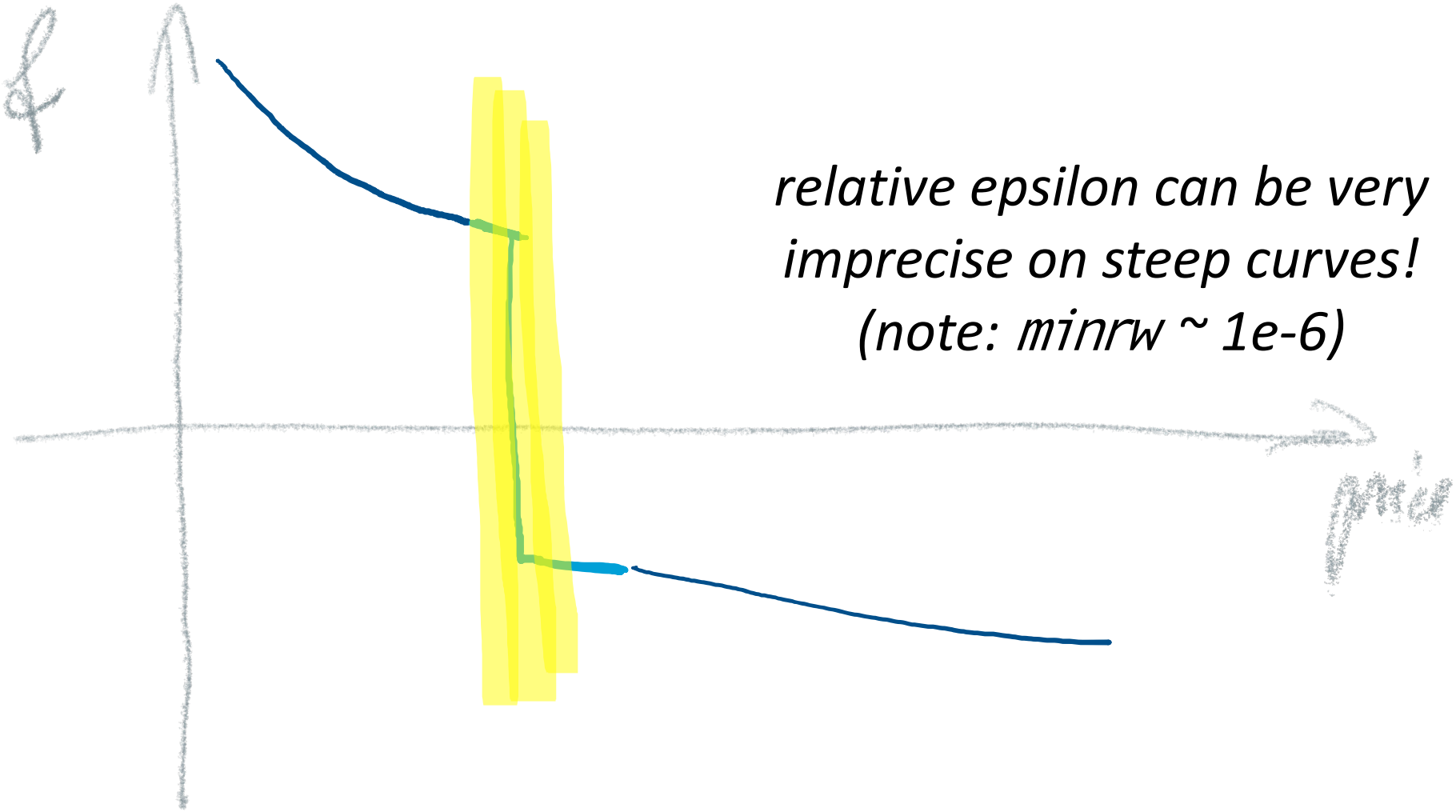
```
pair = USDC/THOR
```



Interlude -- what is close enough?



Our problem with relative epsilon



relative epsilon can be very imprecise on steep curves!
(note: minrw ~ 1e-6)

What we are planning to do

- Better regularization
  - Allow for more control of the imputed width of limit orders
  - Add sentinel curves to the mix that guide the algorithm
- Better convergence assessment
  - Absolute epsilon, expressed in USD terms (requires USD prices!)
- Better post-processing
  - More sophisticated control over actual transaction amounts
  - Question: maximize profit or execution volume?
- Better monitoring
  - Monitor all non-convergence events
  - Triangulation systems (eg RedTeam, margp; unlevered optimization)

Pipeline

- **Pre-processing**
  - Identify suitable arbitrage candidates
  - Add regularization where needed (curve width, sentinel curves)
  - Feed triangulation pipelines
- **Optimization**
  - Run optimization algorithm based on data and parameters provided
- **Post-processing**
  - Check convergence
  - Check triangulation
  - Fine tune transaction
  - Submit transaction