Git archeology: Stepper benchmark history

Paul Gessinger
CERN
2022-02-08

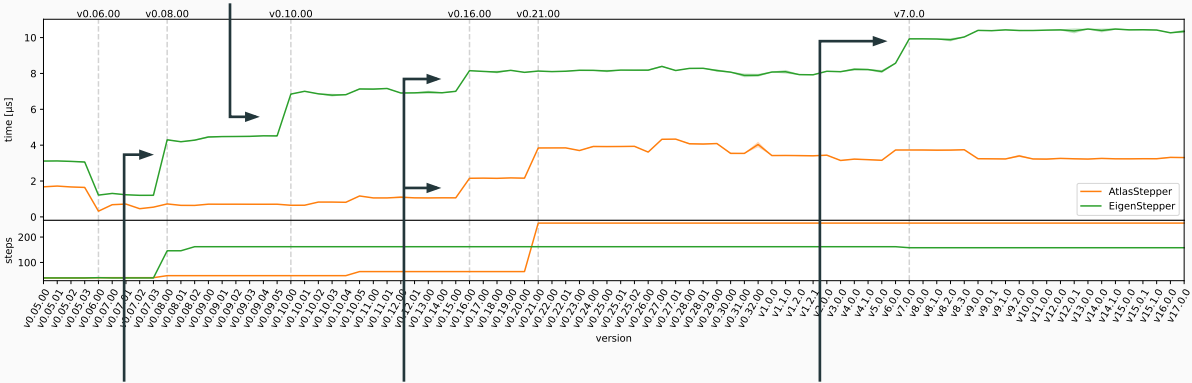
Stepper benchmark

- Sets up arbitrary (fixed) covariance, runs `Propagator::propagate()` a bunch of times (set this to 200k for the following)
- Used because it's available all the way back to v0.05.00 (August 2017)
- **Has increased in execution wall time by factor of 3-4**

Stepper benchmark history: v0.05.00 → v17.0.0

Time added to propagation

MR !583



StepActor merged
MR !473

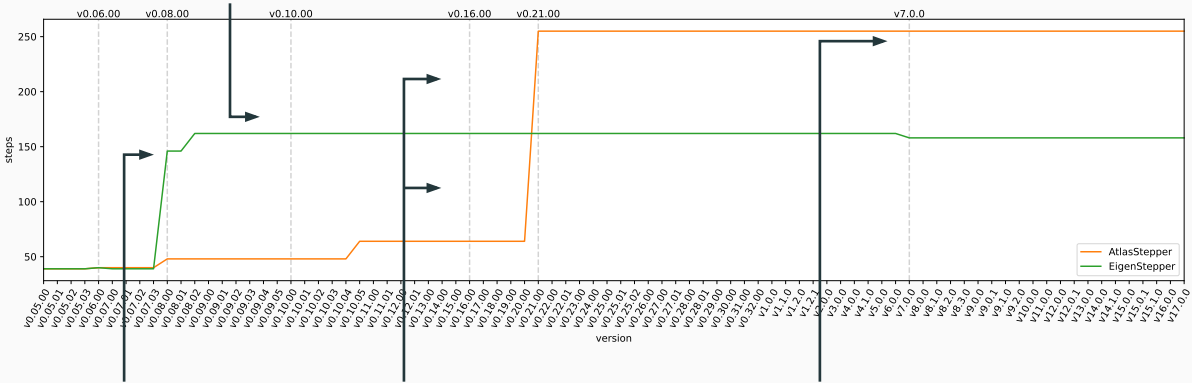
New benchmark tooling
MR !732

EigenStepper step size adjustment
PR !756

Stepper benchmark history: v0.05.00 → v17.0.0

Time added to propagation

MR !583



StepActor merged
MR !473

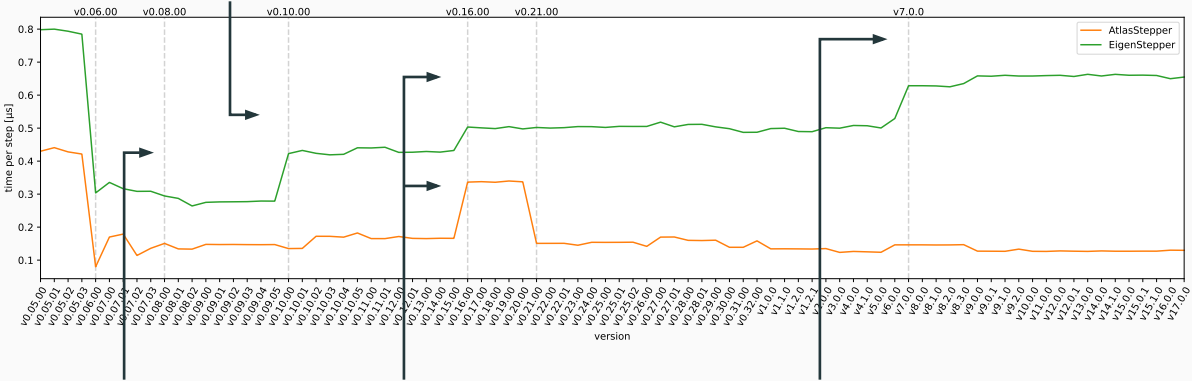
New benchmark tooling
MR !732

EigenStepper step size adjustment
PR !756

Stepper benchmark history: v0.05.00 → v17.0.0

Time added to propagation

MR !583



StepActor merged
MR !473

New benchmark tooling
MR !732

EigenStepper step size adjustment
PR !756

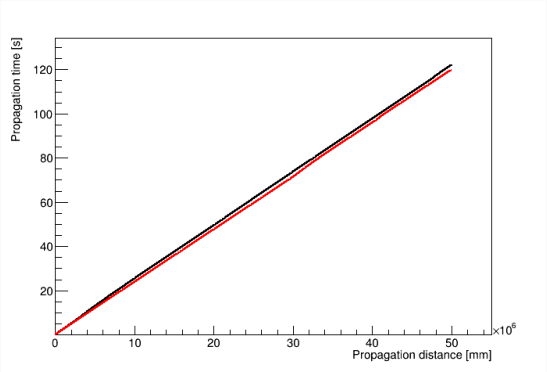
Performance regressions

v0.08.00: *StepActor*

- Merge request !473
- Increases in number of steps
- Time per step roughly constant

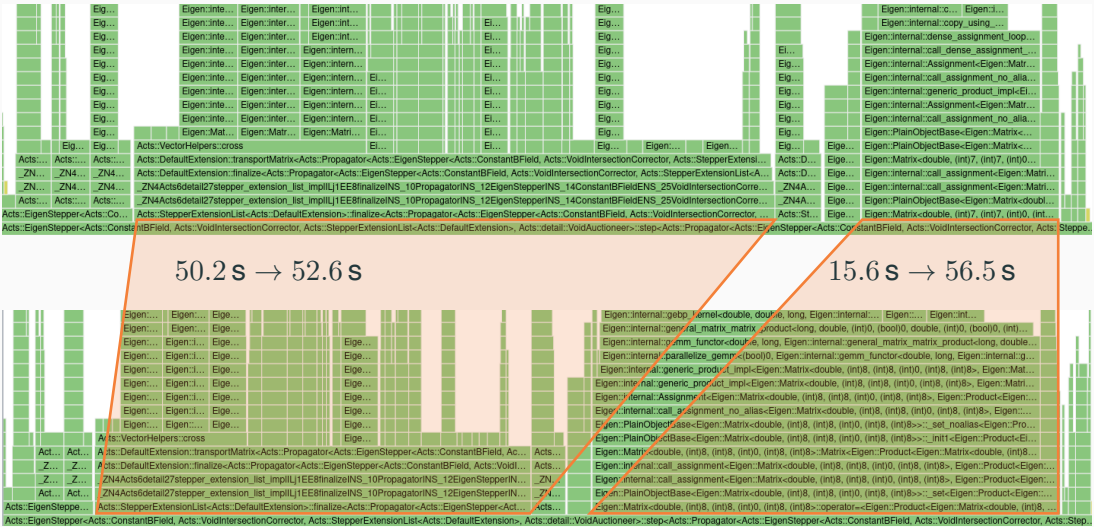
v0.10.00: time introduction

- Merge request !583
- Only added timing to covariance transport and parameters
- Actual time of flight propagation introduced later: MR !590
  - Didn't seem to have an effect $\Longrightarrow$
- Unsure how the range in propagation distance was implemented
- Benchmark at that commit always runs 5000 mm
- Total wall time in my testing now: goes from about 8.7 s $\rightarrow 13.5$ s



Comparing 05fe134a (pre merge of MR !583) against (effectively) post merge of MR !590

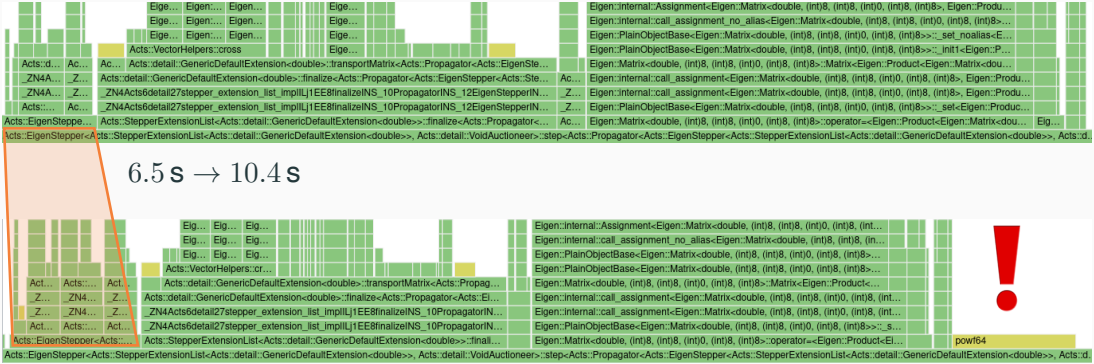
v0.10.00: time introduction



v0.6.0: virtual methods for B-field access

- Introduction of `MagneticFieldProvider`: PR #666: +1.2%
- Steppers use `MagneticFieldProvider`: PR #675: +1.8%
- Allow the build to be configured in ”fail on error” mode PR #687: +1.2%

v0.7.0: new step size scaling



- k_1: ?
- k_2: $0.4\,\mathrm{s} \rightarrow 2.6\,\mathrm{s}$
- k_3: $3.1\,\mathrm{s} \rightarrow 3.4\,\mathrm{s}$
- k_4: $1.9\,\mathrm{s} \rightarrow 2.6\,\mathrm{s}$

Overall steps:
∼same, but we call `tryRungeKuttaStep`
∼ 1.5x more:
$167 \rightarrow 251$

```
stepSizeScaling = std::min(4., std::max(0.25,
  std::pow(state.options.tolerance /
           std::abs(2. * error_estimate), 0.25)));
```

Mitigate by approximation?
($\sim 6\,\%$ wall time speedup)

Next steps

- Optimize obviously
- `powf64` is a low-hanging fruit
- We should probably thoroughly investigate whether we can improve our step sizing
- Eigen is slow at 6x6 and 8x8 operations, can we replace this?