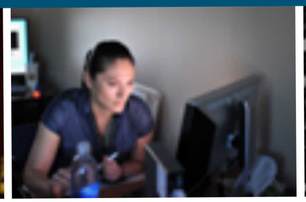
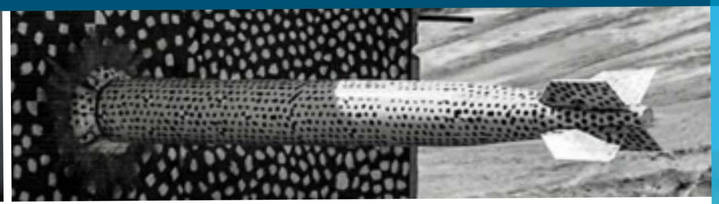
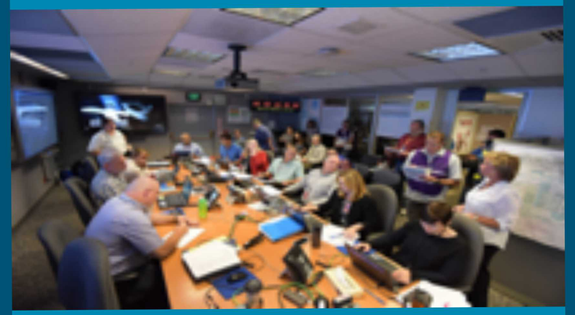
SAND2019-0465PE

Sandia National Laboratories

Advanced Adaptive Time-Stepping for Fundamental Studies of Turbulent, Reacting Flow



PRESENTED BY

Michael A. Hansen, Ph.D., Postdoctoral Appointee

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

“decision-making”
in engineering

COMPUTATIONAL SIMULATION

science
+
mathematics

“decision-making”
in engineering

Design and qualification

COMPUTATIONAL SIMULATION

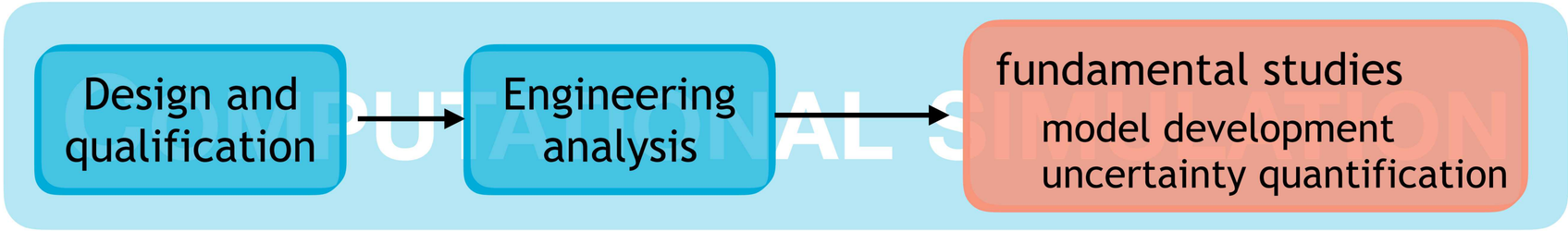
science
+
mathematics

“decision-making” in engineering

science + mathematics

fundamental reacting flow studies rank among the most expensive calculations possible

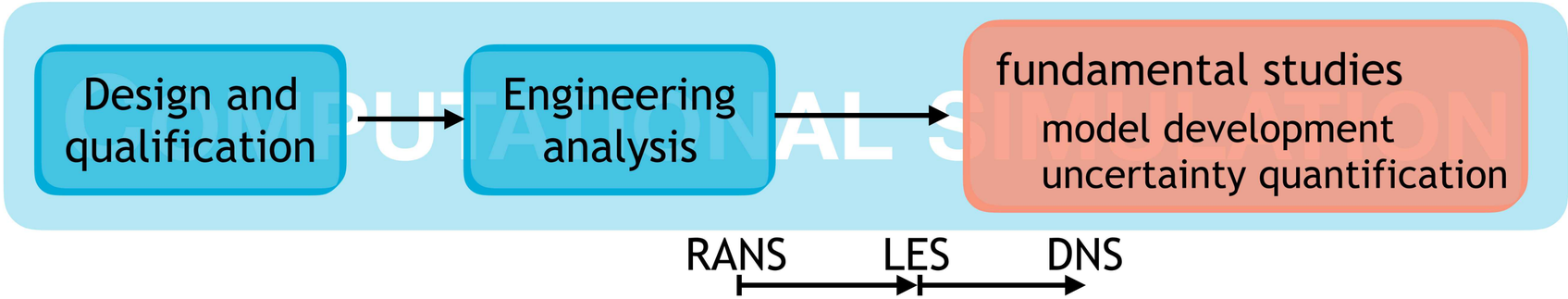
“decision-making” in engineering



science + mathematics

fundamental reacting flow studies rank among the most expensive calculations possible

“decision-making” in engineering



science + mathematics

fundamental reacting flow studies rank among the most expensive calculations possible

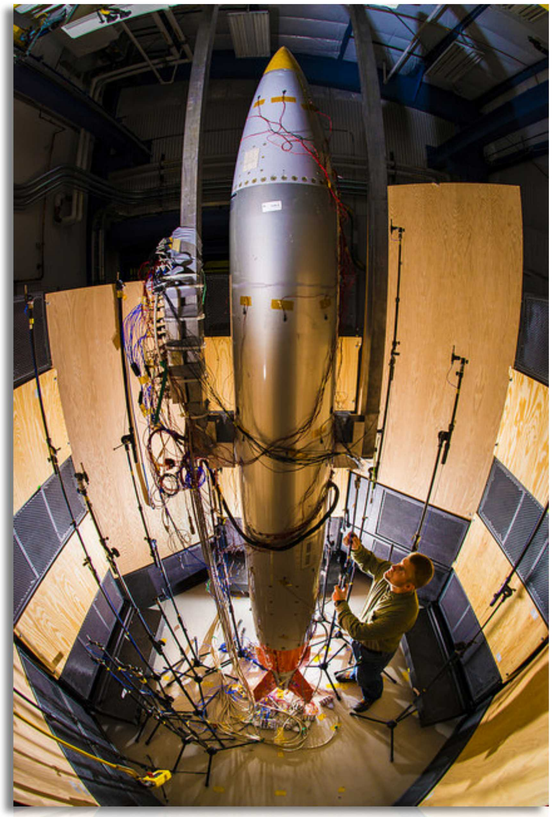
"decision-making" in engineering

COMPUTATIONAL SIMULATION

Design and qualification → Engineering analysis → fundamental studies
model development
uncertainty quantification

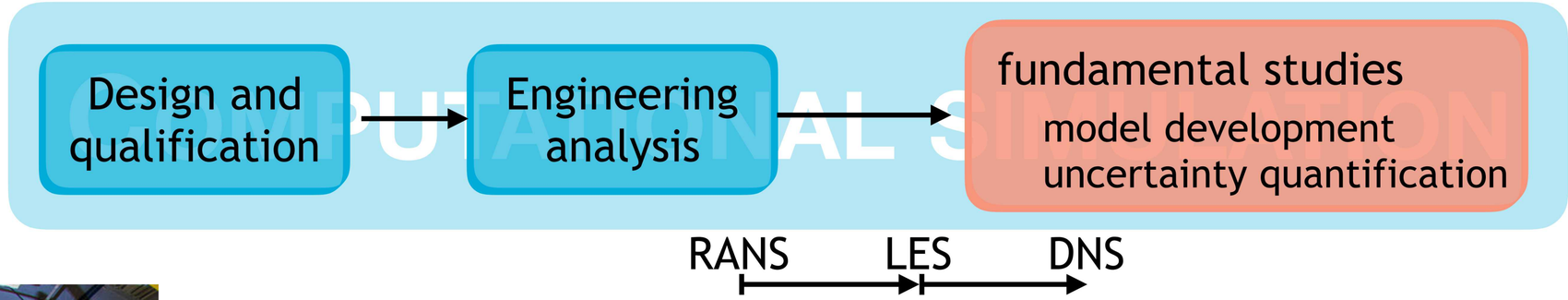
science + mathematics

RANS → LES → DNS



fundamental reacting flow studies rank among the most expensive calculations possible

"decision-making" in engineering



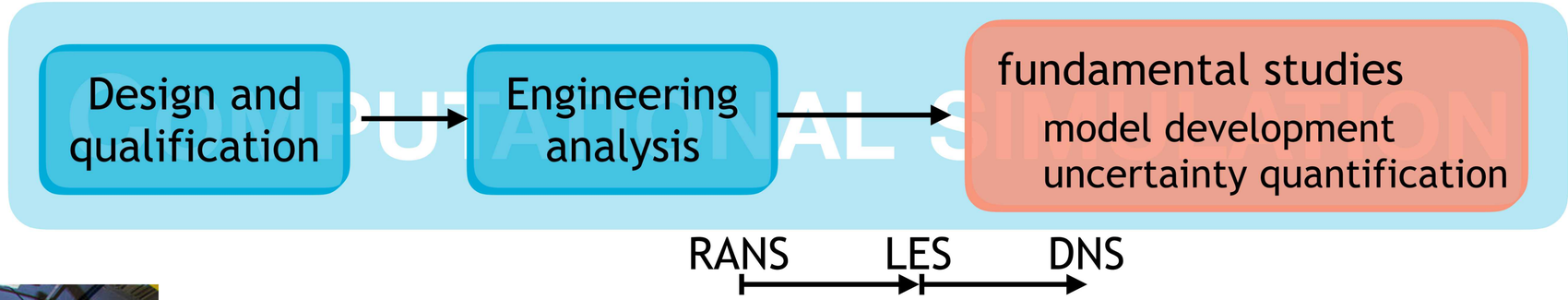
science + mathematics

Direct Numerical Simulation

- Thermochemical nonequilibrium at high speeds
  - *chemical reactions*
  - *multiple temperatures*

fundamental reacting flow studies rank among the most expensive calculations possible

"decision-making" in engineering



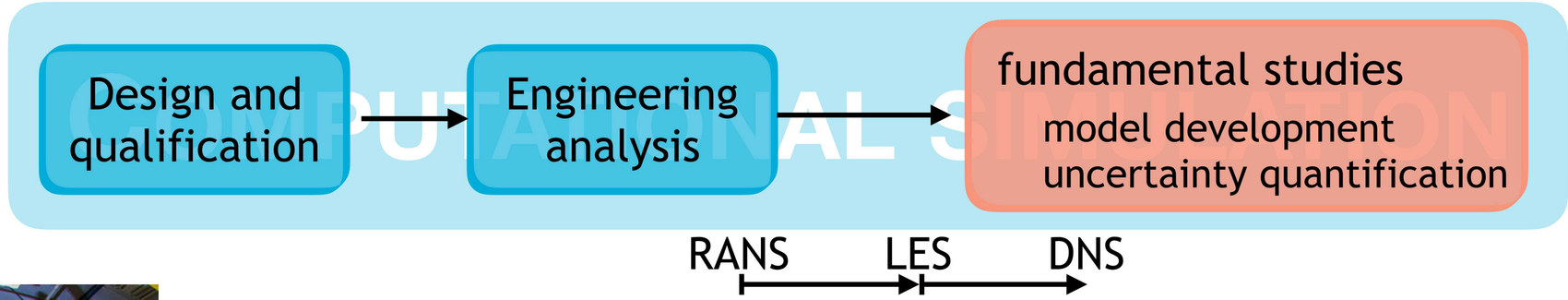
science + mathematics

Direct Numerical Simulation

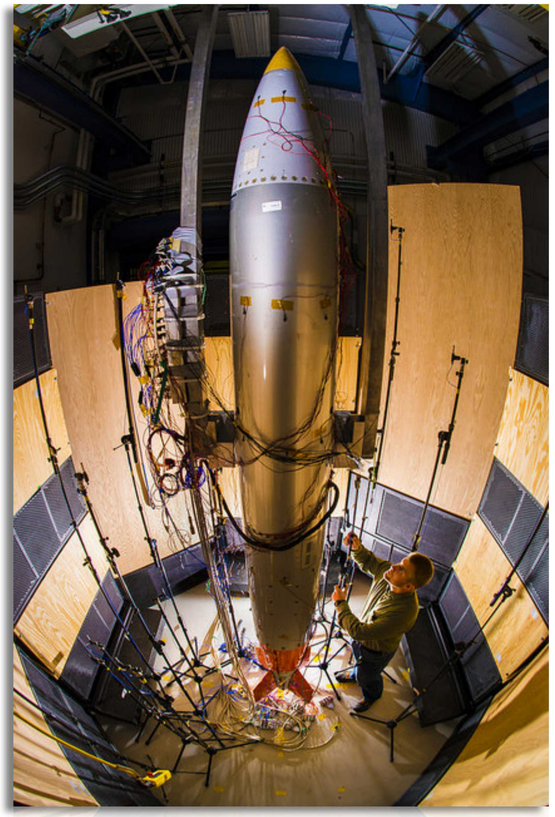
- Thermochemical nonequilibrium at high speeds
  - *chemical reactions*
  - *multiple temperatures*
- Geometry-induced stiffness (e.g. boundary layers)

fundamental reacting flow studies rank among the most expensive calculations possible

"decision-making" in engineering



science + mathematics

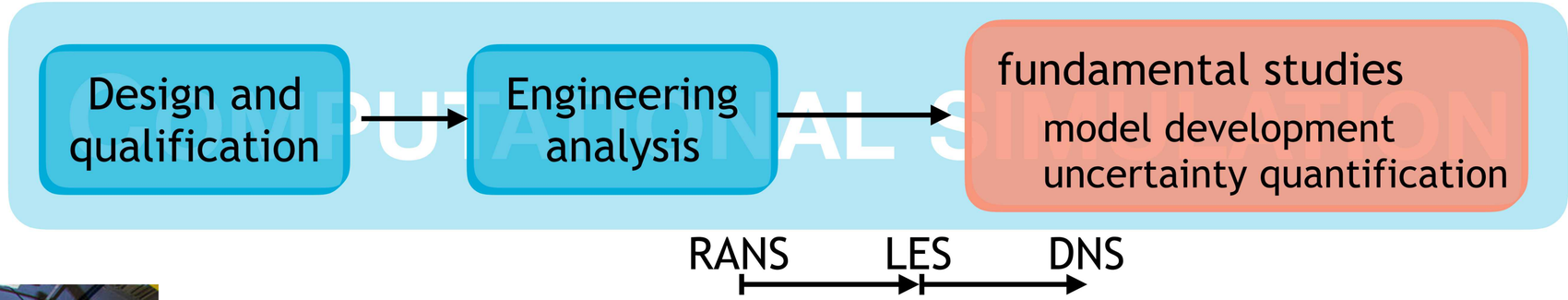


Direct Numerical Simulation

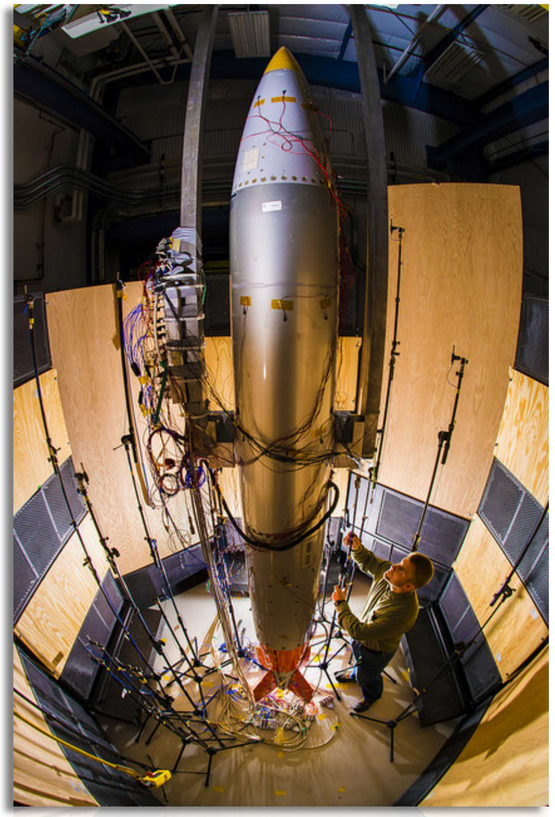
- Thermochemical nonequilibrium at high speeds
  - *chemical reactions*
  - *multiple temperatures*
- Geometry-induced stiffness (e.g. boundary layers)
- Advanced high-order spatial discretizations
  - *high-resolution, low-dissipation schemes*
  - *unclear stability limits*

fundamental reacting flow studies rank among the most expensive calculations possible

"decision-making" in engineering



science + mathematics



Direct Numerical Simulation

- Thermochemical nonequilibrium at high speeds
  - *chemical reactions*
  - *multiple temperatures*
- Geometry-induced stiffness (e.g. boundary layers)
- Advanced high-order spatial discretizations
  - *high-resolution, low-dissipation schemes*
  - *unclear stability limits*
- **Explicit time-integration techniques**

fundamental reacting flow studies rank among the most expensive calculations possible

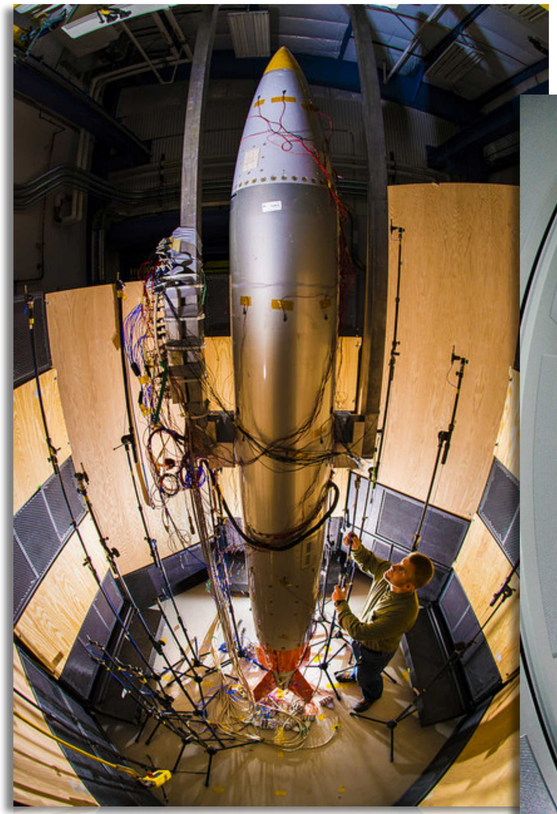
"decision-making" in engineering

COMPUTATIONAL SIMULATION

Design and qualification → Engineering analysis → fundamental studies
model development
uncertainty quantification

science + mathematics

RANS → LES → DNS



Direct Numerical Simulation

- Thermochemical nonequilibrium at high speeds
  - *chemical reactions*
  - *multiple temperatures*
- Geometry-induced stiffness (e.g. boundary layers)
- Advanced high-order spatial discretizations
  - *high-resolution, low-dissipation schemes*
  - *unclear stability limits*
- **Explicit time-integration techniques**
- Millions of cpu-hours on the largest supercomputers

Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

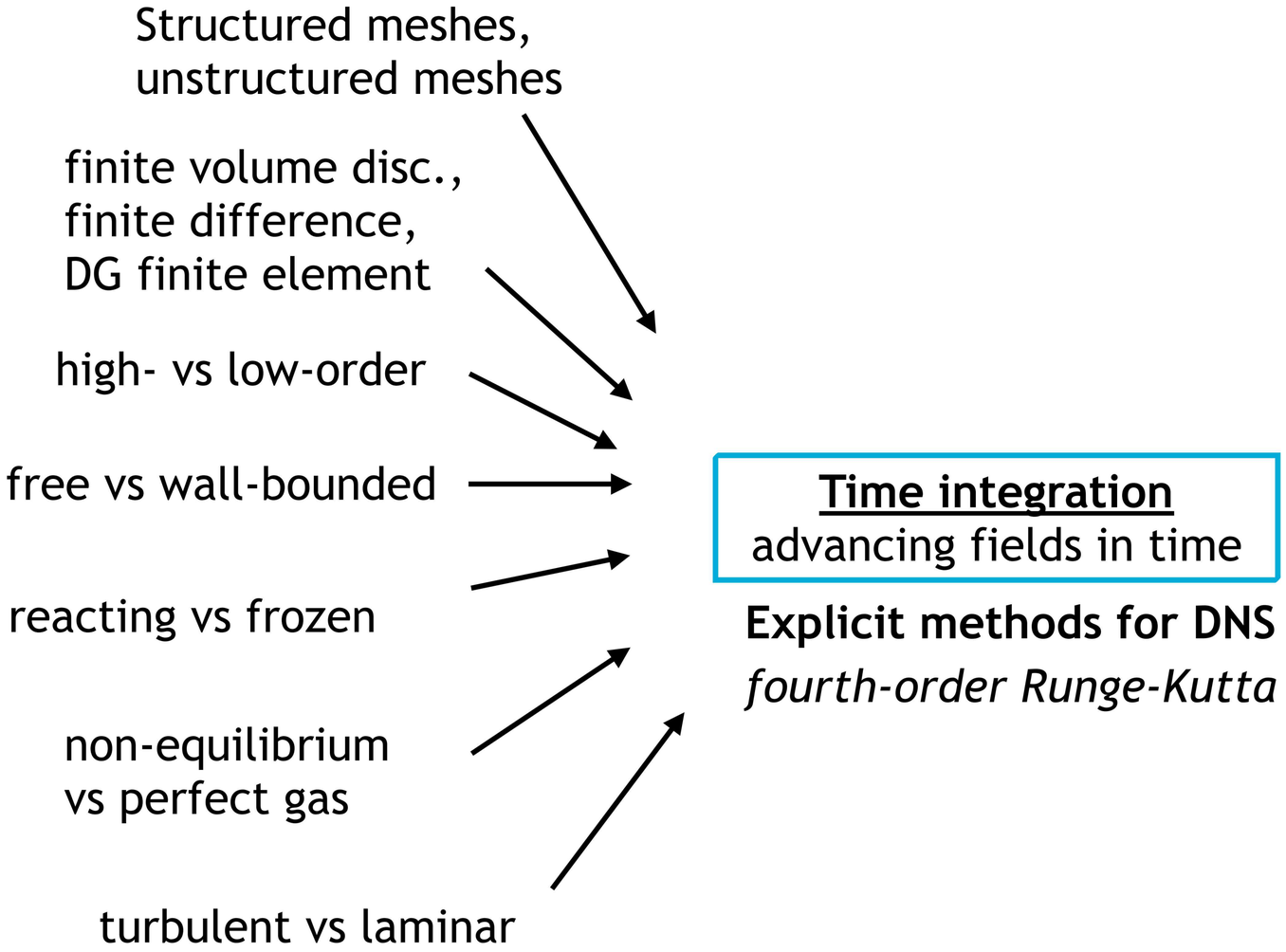
Every variation of DNS calculation runs into the time integrator

Time integration
advancing fields in time

Explicit methods for DNS

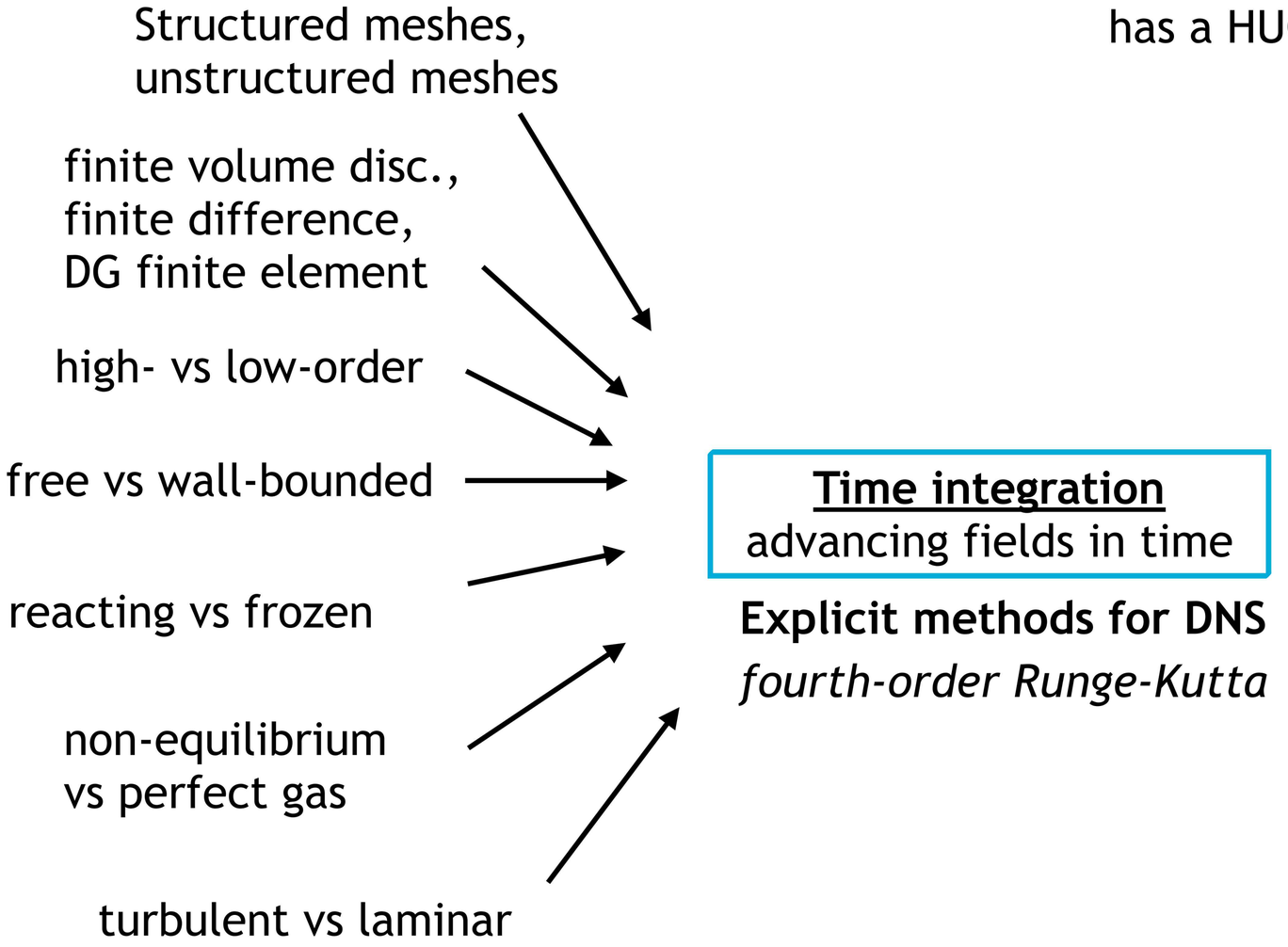
fourth-order Runge-Kutta

Every variation of DNS calculation runs into the time integrator



Every variation of DNS calculation runs into the time integrator

Selection/adaptation of the time step size has a HUGE impact on simulations



Every variation of DNS calculation runs into the time integrator

Structured meshes,
unstructured meshes

finite volume disc.,
finite difference,
DG finite element

high- vs low-order

free vs wall-bounded

reacting vs frozen

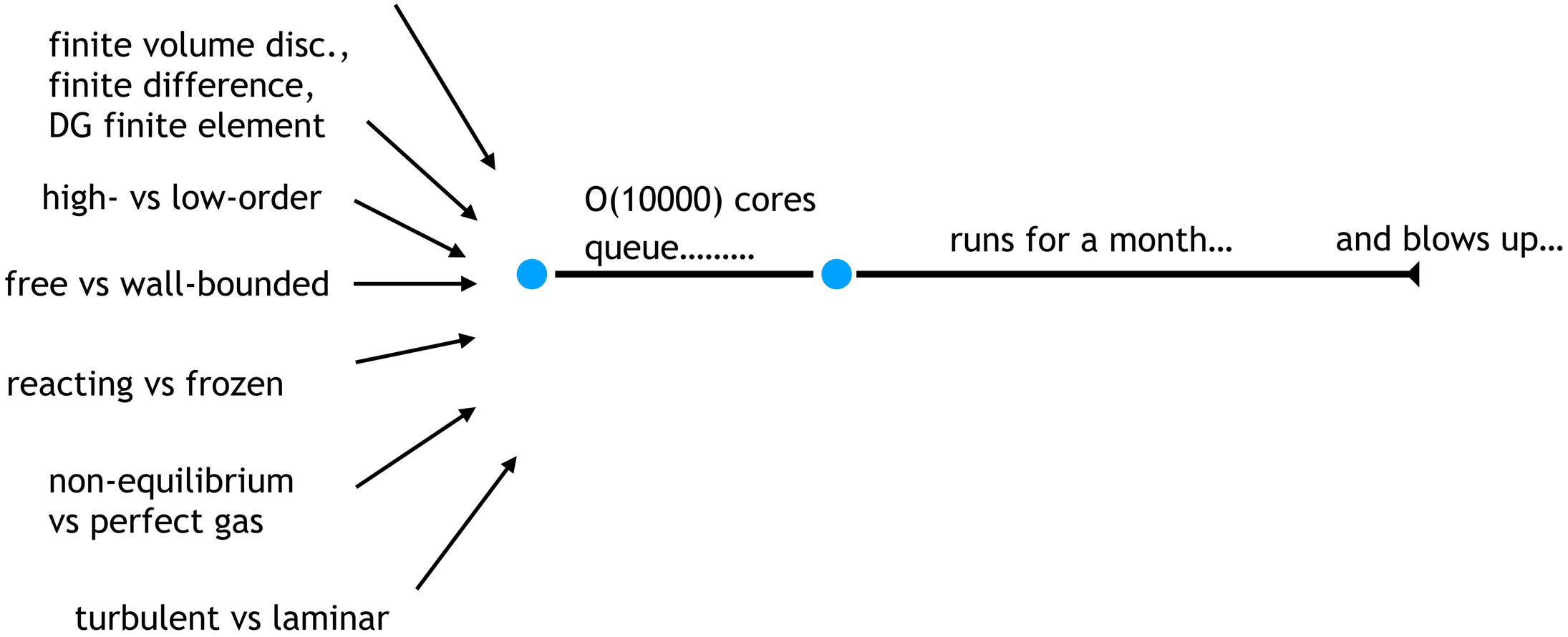
non-equilibrium
vs perfect gas

turbulent vs laminar

Selection/adaptation of the time step size
has a HUGE impact on simulations

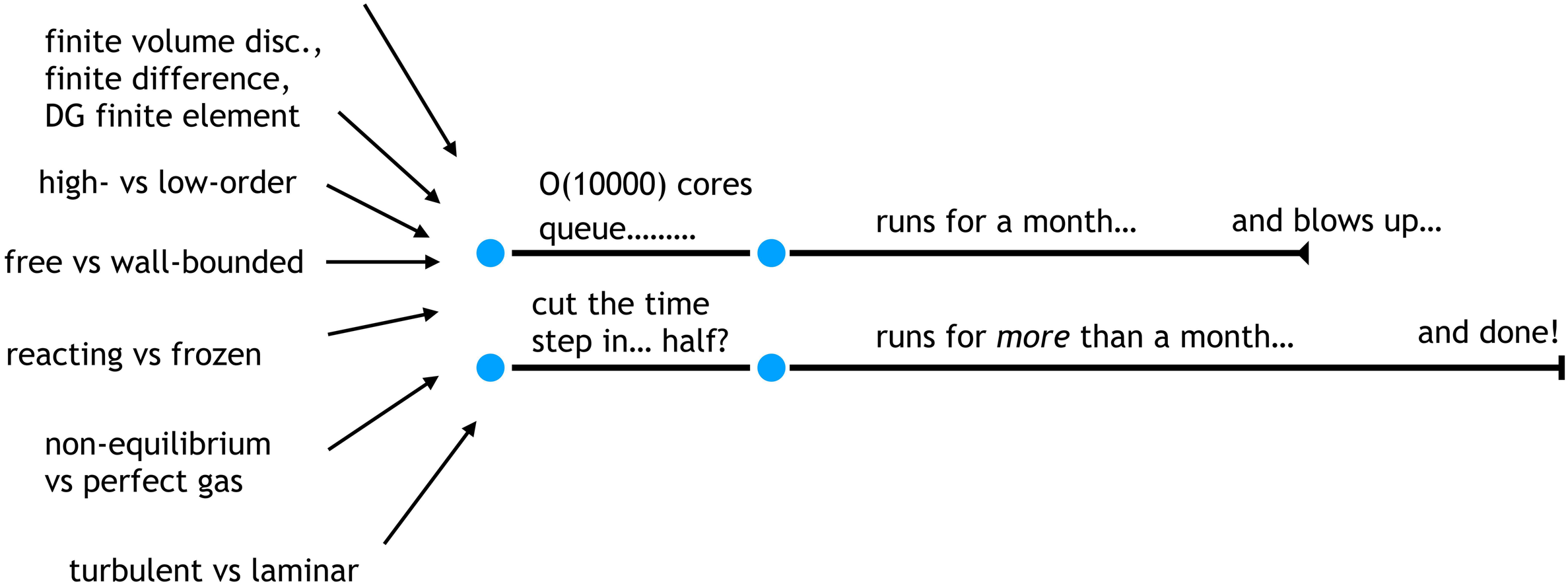
Every variation of DNS calculation runs into the time integrator

Selection/adaptation of the time step size has a HUGE impact on simulations



Every variation of DNS calculation runs into the time integrator

Selection/adaptation of the time step size has a HUGE impact on simulations



Every variation of DNS calculation runs into the time integrator

Selection/adaptation of the time step size has a HUGE impact on simulations

Structured meshes, unstructured meshes

finite volume disc., finite difference, DG finite element

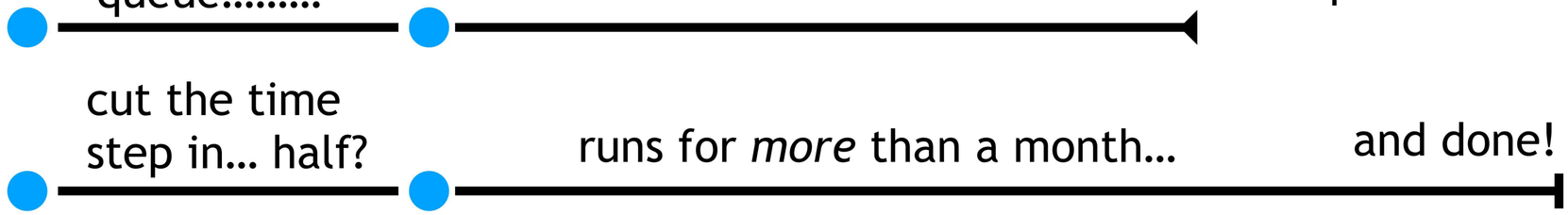
high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium vs perfect gas

turbulent vs laminar



A badly run time integrator can cost many millions of cpu hours!

Not the optimal workflow!

Every variation of DNS calculation runs into the time integrator

Selection/adaptation of the time step size has a HUGE impact on simulations

Structured meshes, unstructured meshes

finite volume disc., finite difference, DG finite element

high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium vs perfect gas

turbulent vs laminar

Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

A badly run time integrator can cost many millions of cpu hours!

Not the optimal workflow!

Every variation of DNS calculation runs into the time integrator

Structured meshes,
unstructured meshes

finite volume disc.,
finite difference,
DG finite element

high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium
vs perfect gas

turbulent vs laminar

Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

Selection/adaptation of the time step size
has a HUGE impact on simulations

Constraints on time integrator

A badly run time integrator can
cost many millions of cpu hours!

Not the optimal workflow!

Every variation of DNS calculation runs into the time integrator

Structured meshes,
unstructured meshes

finite volume disc.,
finite difference,
DG finite element

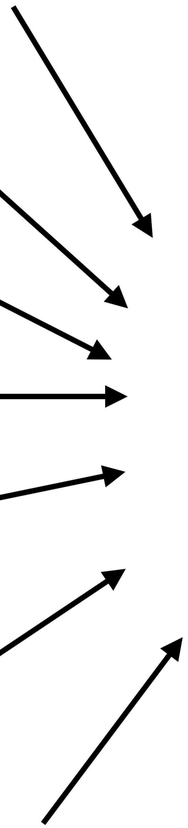
high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium
vs perfect gas

turbulent vs laminar



Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

Selection/adaptation of the time step size
has a HUGE impact on simulations

Constraints on time integrator

- Don't want it to blow up

A badly run time integrator can
cost many millions of cpu hours!

Not the optimal workflow!

Every variation of DNS calculation runs into the time integrator

Structured meshes,
unstructured meshes

finite volume disc.,
finite difference,
DG finite element

high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium
vs perfect gas

turbulent vs laminar

Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

Selection/adaptation of the time step size
has a HUGE impact on simulations

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error

A badly run time integrator can
cost many millions of cpu hours!

Not the optimal workflow!

Every variation of DNS calculation runs into the time integrator

Structured meshes,
unstructured meshes

finite volume disc.,
finite difference,
DG finite element

high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium
vs perfect gas

turbulent vs laminar

Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

Selection/adaptation of the time step size
has a HUGE impact on simulations

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping

A badly run time integrator can
cost many millions of cpu hours!

Not the optimal workflow!

Every variation of DNS calculation runs into the time integrator

Structured meshes,
unstructured meshes

finite volume disc.,
finite difference,
DG finite element

high- vs low-order

free vs wall-bounded

reacting vs frozen

non-equilibrium
vs perfect gas

turbulent vs laminar

Time integration
advancing fields in time

Explicit methods for DNS

fourth-order Runge-Kutta

Selection/adaptation of the time step size
has a HUGE impact on simulations

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!

A badly run time integrator can
cost many millions of cpu hours!

Not the optimal workflow!

Selecting a constant time step is an option but we need somewhere to start.

$$\mathrm{CFL} = \frac{u\Delta t}{\Delta x} \leq 1$$

$$\mathrm{VNN} = \frac{D\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

1D periodic domains
linear problems
constant coefficients

Time step bounds suggested by linear stability analysis are difficult to apply to ‘real’ problems, especially for high-order discretizations

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leqslant \frac{1}{2}$$

Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

1D periodic domains
linear problems
constant coefficients

Time step bounds suggested by linear stability analysis are difficult to apply to ‘real’ problems, especially for high-order discretizations

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leqslant \frac{1}{2}$$

Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

1D periodic domains
linear problems
constant coefficients

real problems

3D arbitrary domains
nonlinear equations (provably nonlinearly stable)
coupled advection-diffusion-reaction
variable coefficients
geometry-induced and model stiffness

Time step bounds suggested by linear stability analysis are difficult to apply to ‘real’ problems, especially for high-order discretizations

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leqslant \frac{1}{2}$$

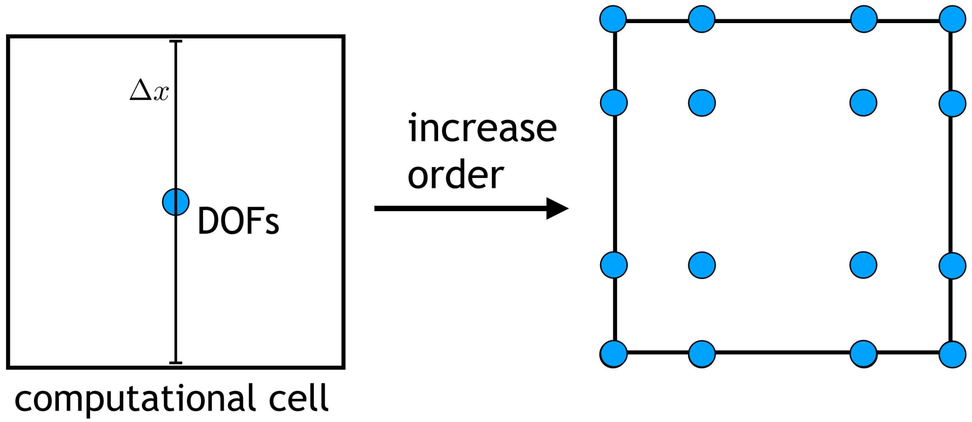
Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

1D periodic domains
linear problems
constant coefficients

real problems

3D arbitrary domains
nonlinear equations (provably nonlinearly stable)
coupled advection-diffusion-reaction
variable coefficients
geometry-induced and model stiffness

High-order spatial discretization (e.g., DG)



computational cell

Time step bounds suggested by linear stability analysis are difficult to apply to ‘real’ problems, especially for high-order discretizations

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leqslant \frac{1}{2}$$

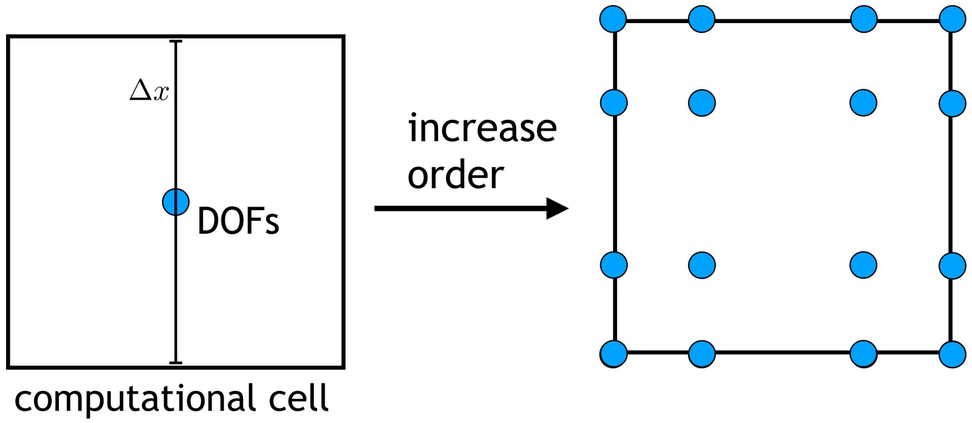
Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

1D periodic domains
linear problems
constant coefficients

<u>real problems</u>

3D arbitrary domains
nonlinear equations (provably nonlinearly stable)
coupled advection-diffusion-reaction
variable coefficients
geometry-induced and model stiffness

High-order spatial discretization (e.g., DG)



computational cell

Characteristic element length is unclear

CFL, VNN are not well defined!

Depend on order!

Time step bounds suggested by linear stability analysis are difficult to apply to ‘real’ problems, especially for high-order discretizations

Selecting a constant time step is an option but we need somewhere to start.

Classical linear stability analysis

$$CFL = \frac{u\Delta t}{\Delta x} \leq 1$$

Forward Euler, constant Δt
Upwind spatial disc.,
advection problem

$$VNN = \frac{D\Delta t}{\Delta x^2} \leqq \frac{1}{2}$$

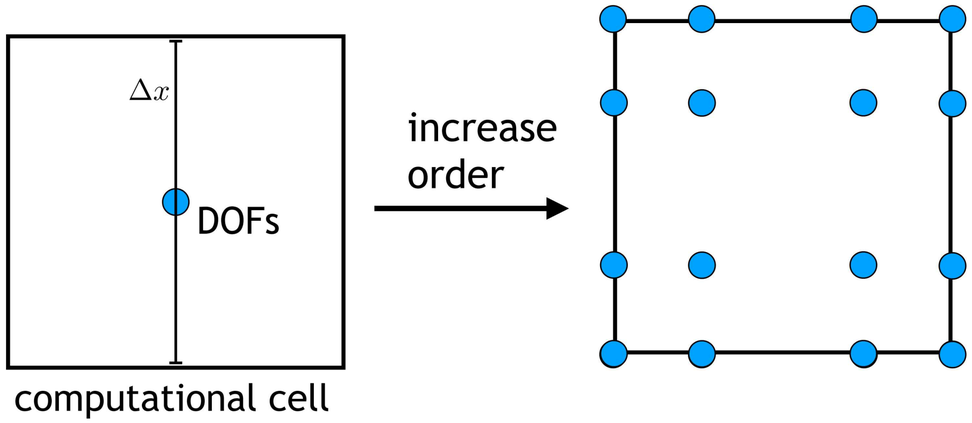
Forward Euler, constant Δt
Centered spatial disc.,
diffusion problem

1D periodic domains
linear problems
constant coefficients

real problems

3D arbitrary domains
nonlinear equations (provably nonlinearly stable)
coupled advection-diffusion-reaction
variable coefficients
geometry-induced and model stiffness

High-order spatial discretization (e.g., DG)



computational cell

Characteristic element length is unclear

CFL, VNN are not well defined!

Depend on order!

Still guess and check!

Runge-Kutta methods allow embedded error estimation

Runge-Kutta methods use the RHS at internal stages to compute a step update

Forward Euler

$$u^{n+1} = u^n + \Delta t(r_1)$$

Trapezoidal

$$u^{n+1} = u^n + \Delta t \left(\frac{1}{2} r_1 + \frac{1}{2} r_2 \right)$$

Ralston's RK4

$$u^{n+1} = u^n + \Delta t \left(\frac{11}{72} r_1 + \frac{25}{72} r_2 + \frac{25}{72} r_3 + \frac{11}{72} r_4 \right)$$

Runge-Kutta methods allow embedded error estimation

Runge-Kutta methods use the RHS at internal stages to compute a step update

Forward Euler

$$u^{n+1} = u^n + \Delta t(r_1)$$

Trapezoidal

$$u^{n+1} = u^n + \Delta t\left(\frac{1}{2}r_1 + \frac{1}{2}r_2\right)$$

Ralston's RK4

$$u^{n+1} = u^n + \Delta t\left(\frac{11}{72}r_1 + \frac{25}{72}r_2 + \frac{25}{72}r_3 + \frac{11}{72}r_4\right)$$

order *p*

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

Runge-Kutta methods allow embedded error estimation

Runge-Kutta methods use the RHS at internal stages to compute a step update

Forward Euler

$$u^{n+1} = u^n + \Delta t\, r_1$$

Trapezoidal

$$u^{n+1} = u^n + \Delta t \left(\frac{1}{2} r_1 + \frac{1}{2} r_2 \right)$$

Ralston's RK4

$$u^{n+1} = u^n + \Delta t \left(\frac{11}{72} r_1 + \frac{25}{72} r_2 + \frac{25}{72} r_3 + \frac{11}{72} r_4 \right)$$

order *p*

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

order *p-1*

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

Low cost: error estimate uses the *same* RHS evaluations

Runge-Kutta methods allow embedded error estimation, and error control offers a problem-independent approach to time step adaptation

order *p*

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

order *p-1*

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} \hat{b}_i r_i$$

Runge-Kutta methods allow embedded error estimation, and error control offers a problem-independent approach to time step adaptation

order *p*
$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

order *p-1*
$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} \hat{b}_i r_i$$

$$e = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) r(u_i)$$

order (p-1) error estimate

→ Manipulate the time step to keep the error estimate at a target value

Sophisticated PID controllers can do this!

$\{b_i\}$ quadrature coeffs for order p (e.g. 4)

$\{\hat{b}_i\}$ quadrature coeffs for order p-1

Runge-Kutta methods allow embedded error estimation, and error control offers a problem-independent approach to time step adaptation

order *p* $$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

order *p-1* $$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} \hat{b}_i r_i$$

$$e = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) r(u_i)$$ order (p-1) error estimate

→ Manipulate the time step to keep the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

$\{b_i\}$ quadrature coeffs for order p (e.g. 4)

$\{\hat{b}_i\}$ quadrature coeffs for order p-1

Runge-Kutta methods allow embedded error estimation, and error control offers a problem-independent approach to time step adaptation

order *p* $$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i r_i$$

order *p-1* $$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} \hat{b}_i r_i$$

$$e = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) r(u_i)$$ order (p-1) error estimate

→ Manipulate the time step to keep the error estimate at a target value

$\{b_i\}$ quadrature coeffs for order p (e.g. 4)

$\{\hat{b}_i\}$ quadrature coeffs for order p-1

Sophisticated PID controllers can do this!

But what do I use for the target error?!

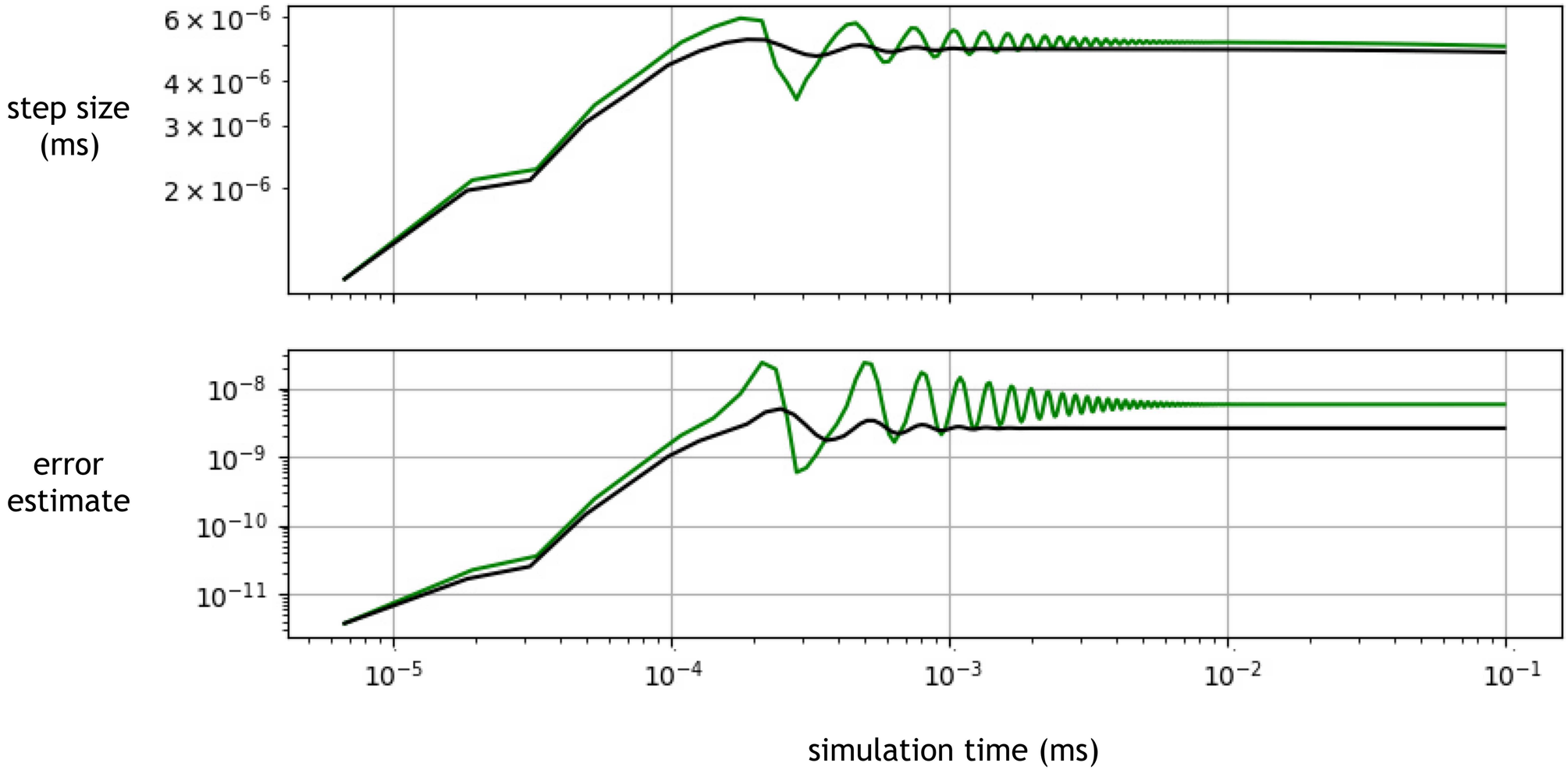
Let's try some values on a well-known problem…

Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

But what do I use for the target error?!

“low” target error: effective control, few step failures

target=2.6e-8
target=1.1e-8

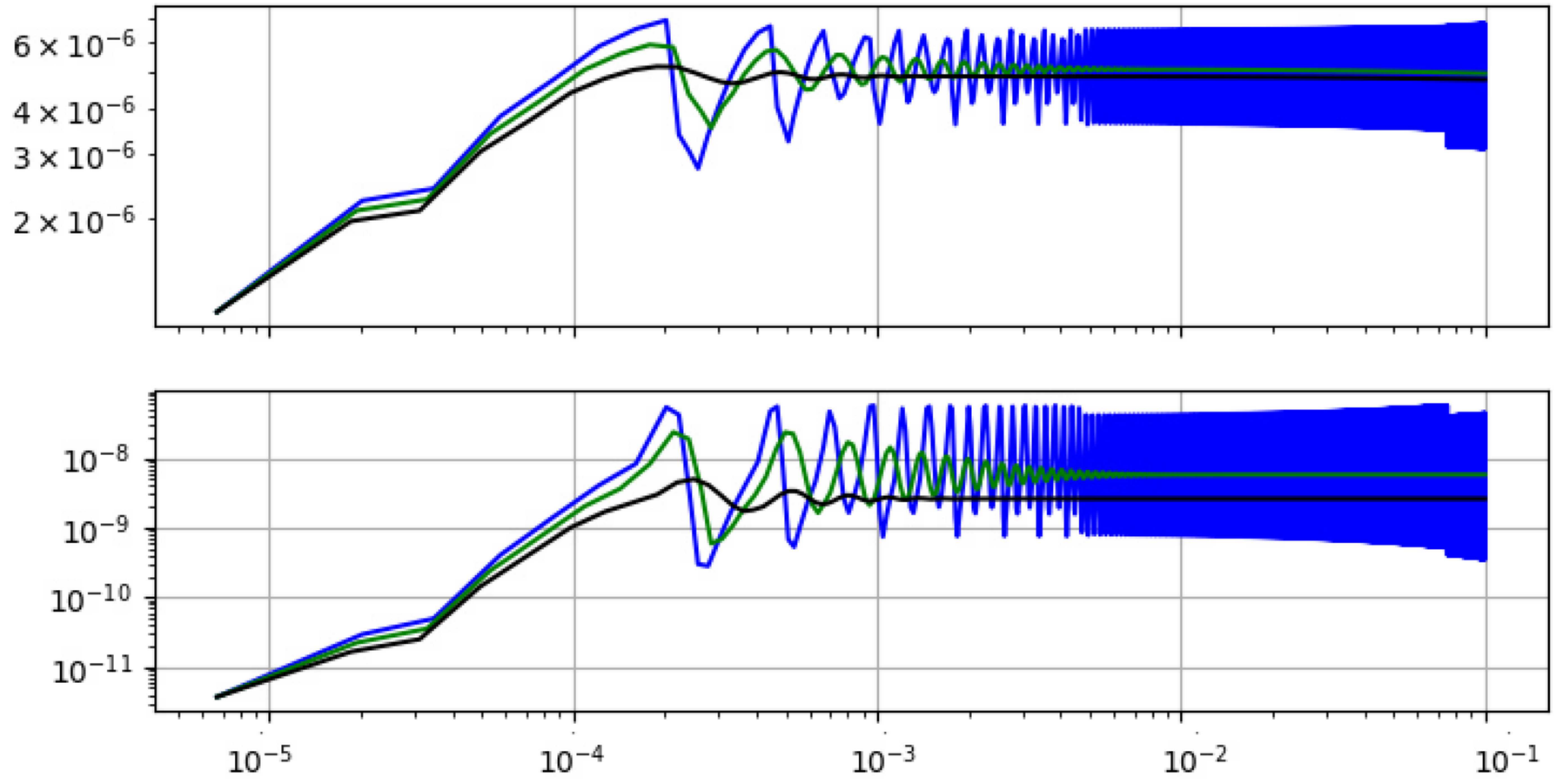


But what do I use for the target error?!

“low” target error: effective control, few step failures
“high” target error: poor control, many failures, SLOW!

target=5.7e-8
target=2.6e-8
target=1.1e-8

Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

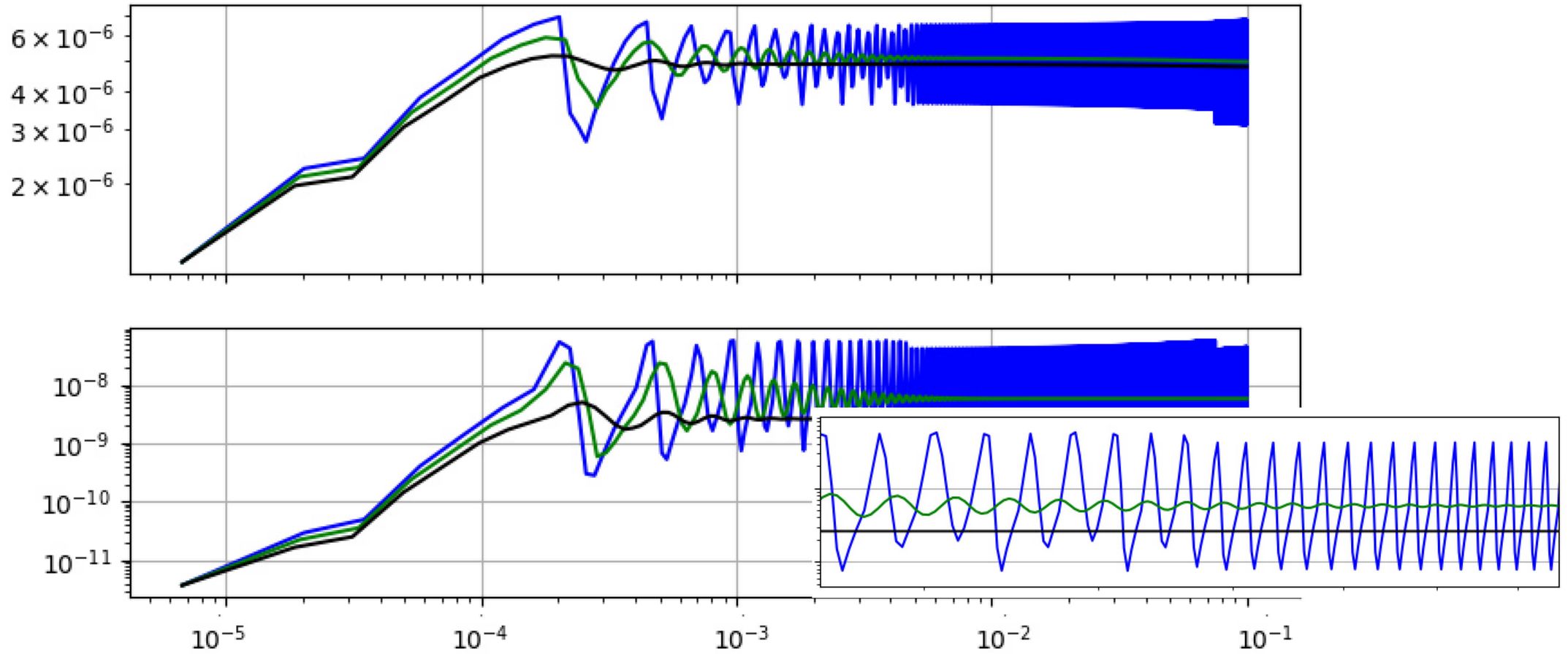


But what do I use for the target error?!

“low” target error: effective control, few step failures
“high” target error: poor control, many failures, SLOW!

target=5.7e-8
target=2.6e-8
target=1.1e-8

Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

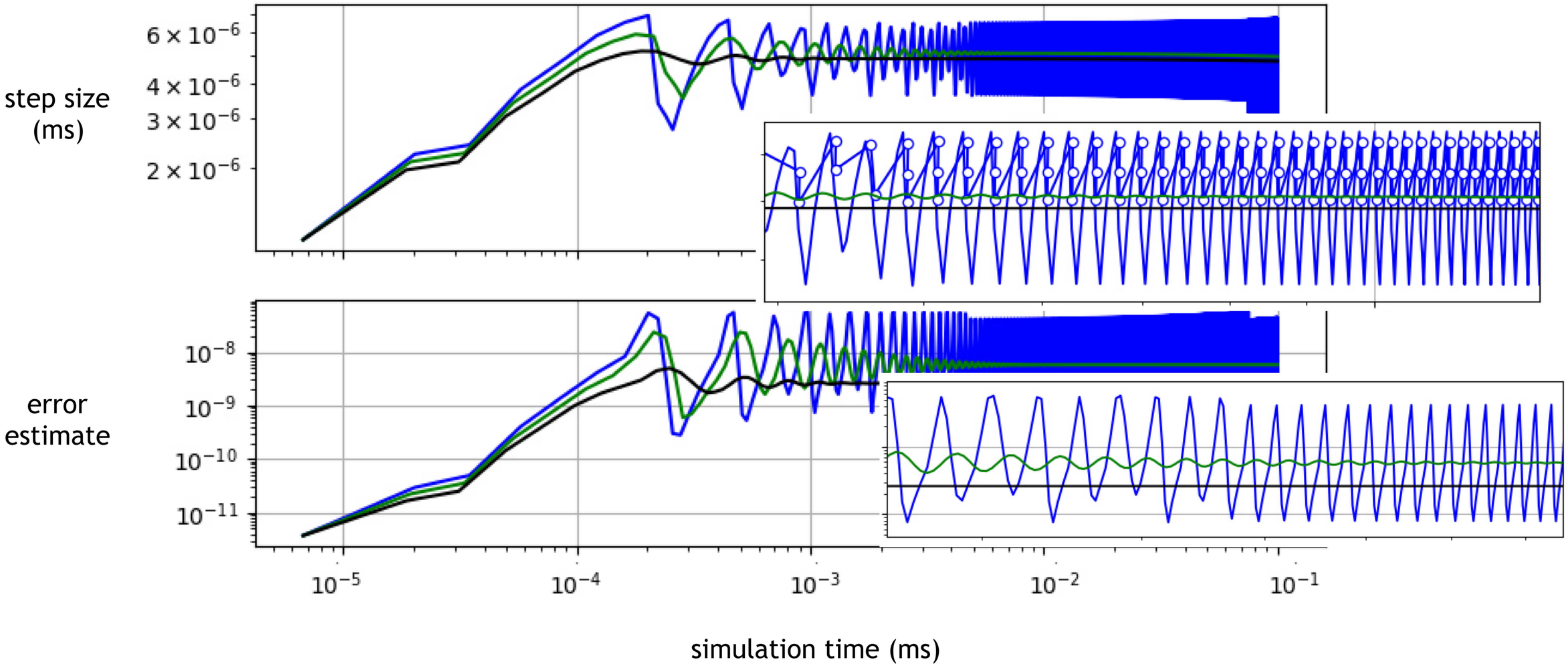


But what do I use for the target error?!

“low” target error: effective control, few step failures
“high” target error: poor control, many failures, SLOW!

target=5.7e-8
target=2.6e-8
target=1.1e-8

Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

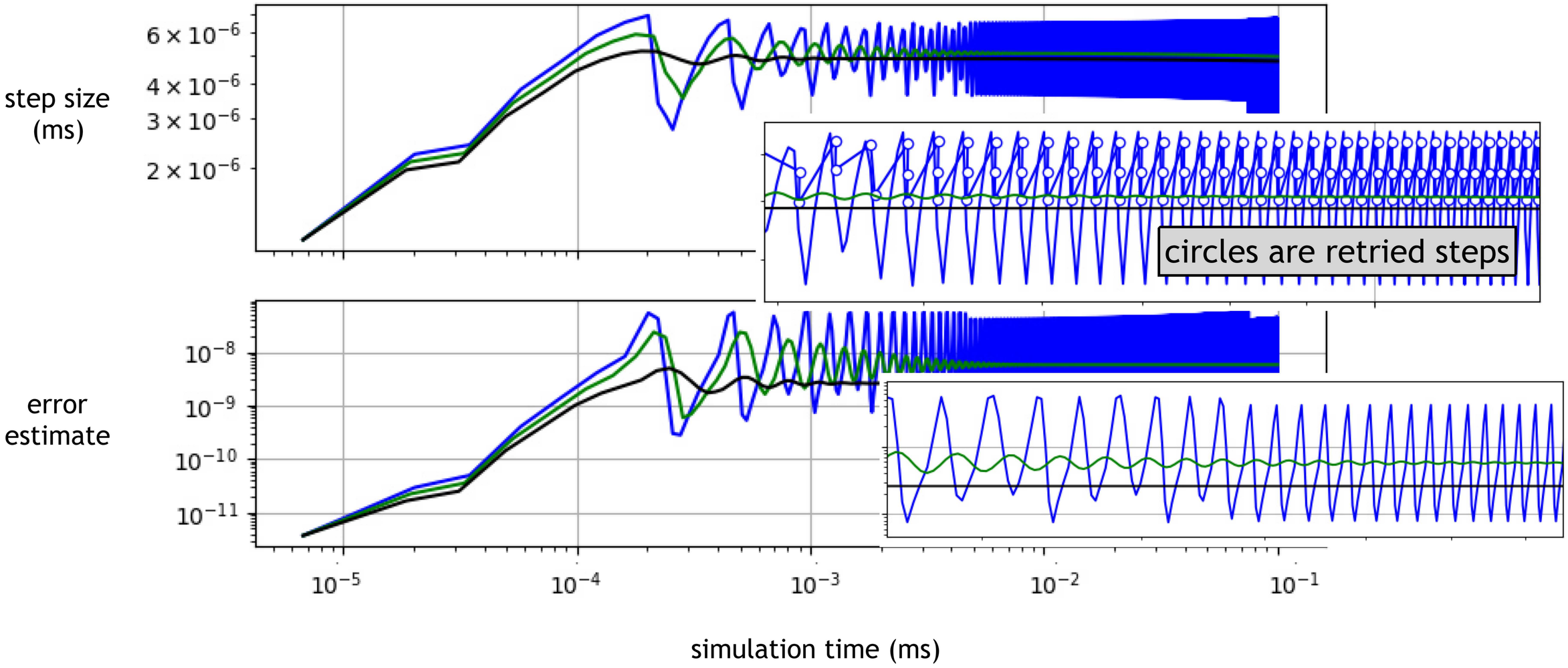


But what do I use for the target error?!

“low” target error: effective control, few step failures
“high” target error: poor control, many failures, SLOW!

target=5.7e-8
target=2.6e-8
target=1.1e-8

Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)



Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

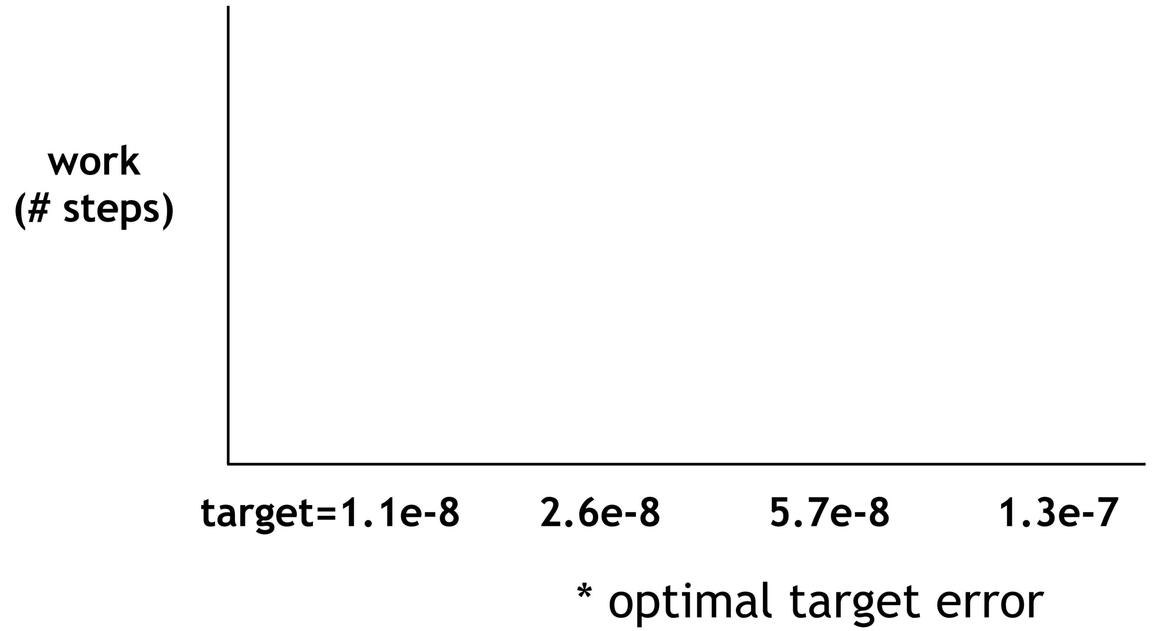
But what do I use for the target error?!

“low” target error: effective control, few step failures
“high” target error: poor control, many failures, SLOW!

Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

But what do I use for the target error?!

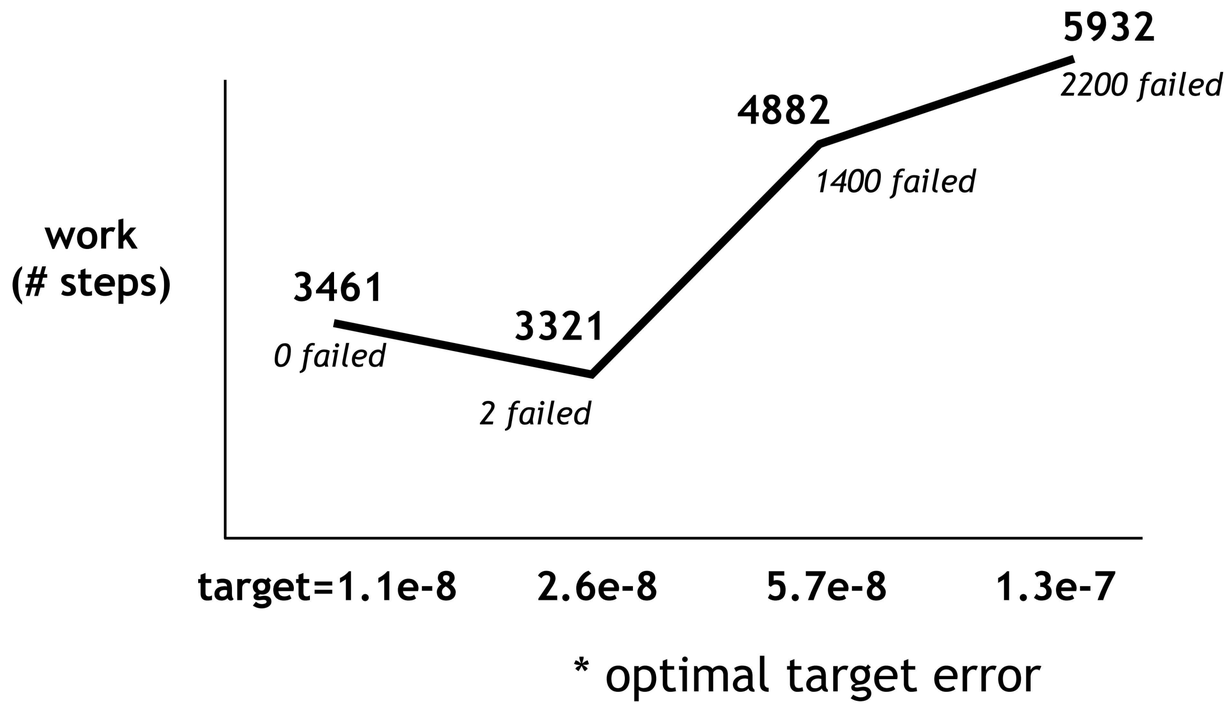
"low" target error: effective control, few step failures
"high" target error: poor control, many failures, SLOW!



Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

But what do I use for the target error?!

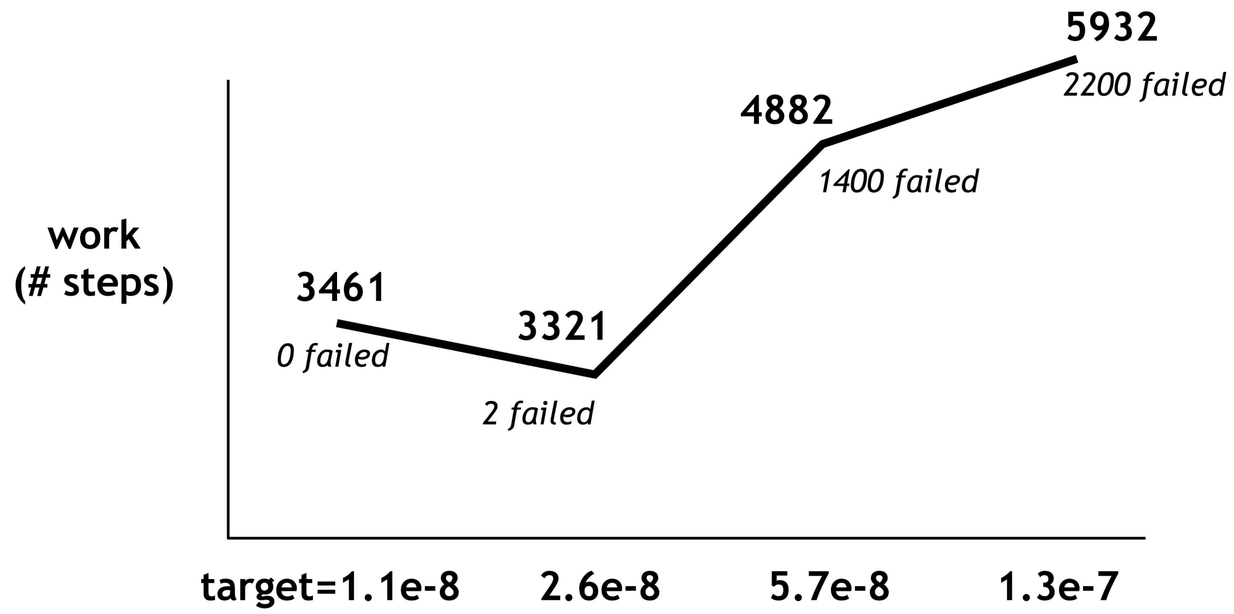
"low" target error: effective control, few step failures
"high" target error: poor control, many failures, SLOW!



Viscous shock, 200x1x1
order 4 HOFD disc.
IMEX-643 solver
reference error est.
(*Kanevsky et al.*)

But what do I use for the target error?!

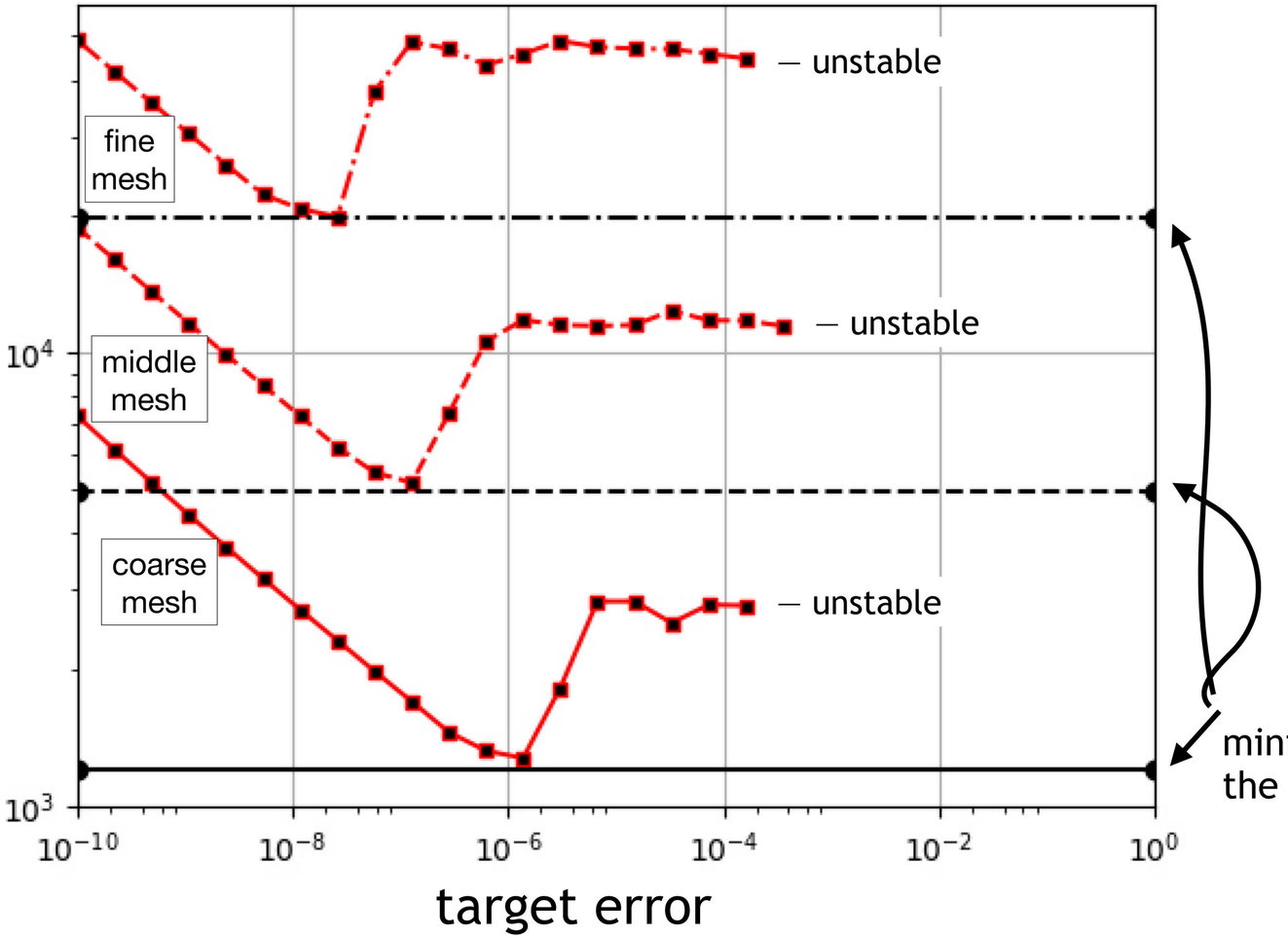
"low" target error: effective control, few step failures
"high" target error: poor control, many failures, SLOW!



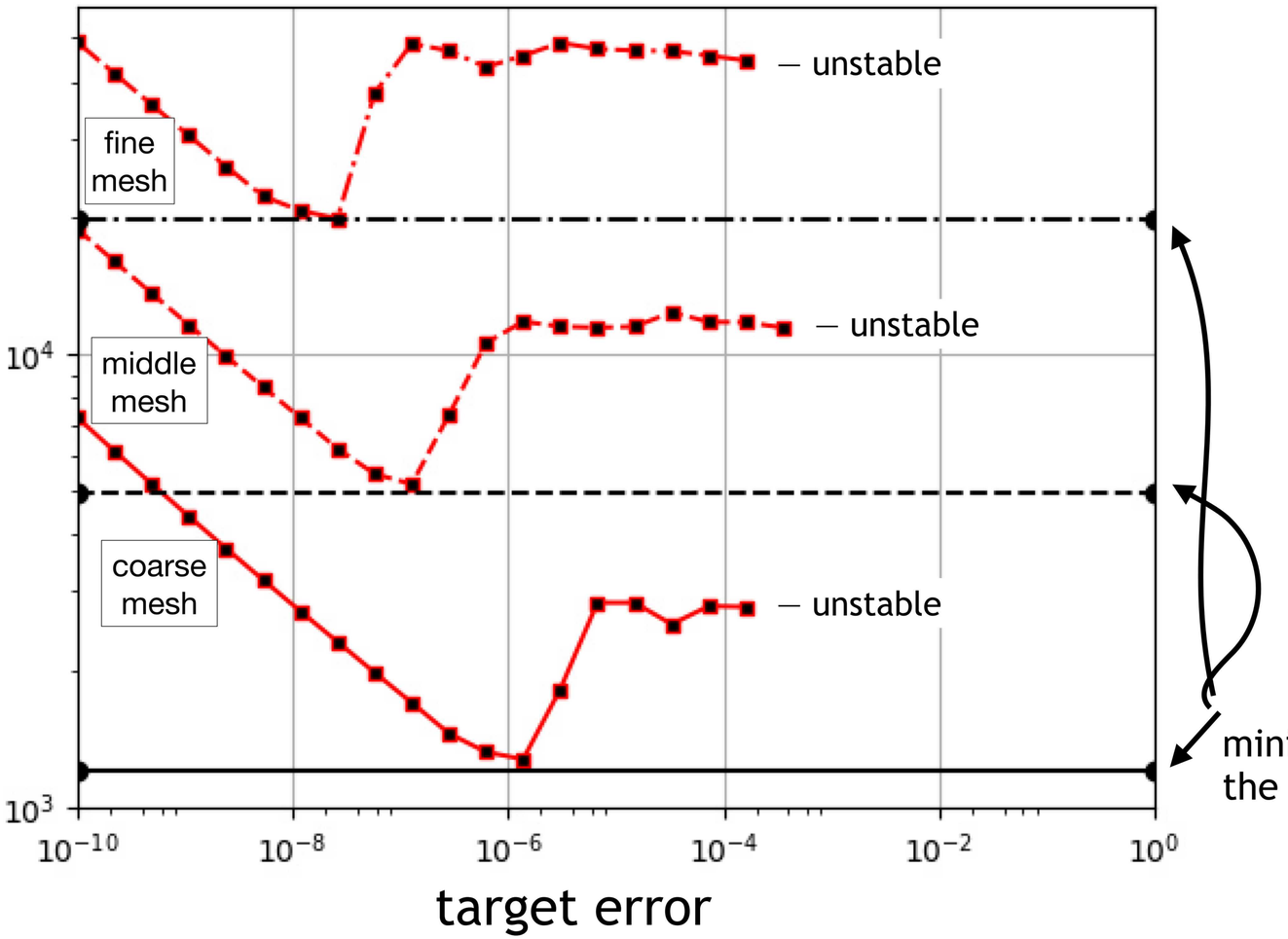
* optimal target error

only a factor of ~2 too large and simulation is *far* slower

total work (#RHS)

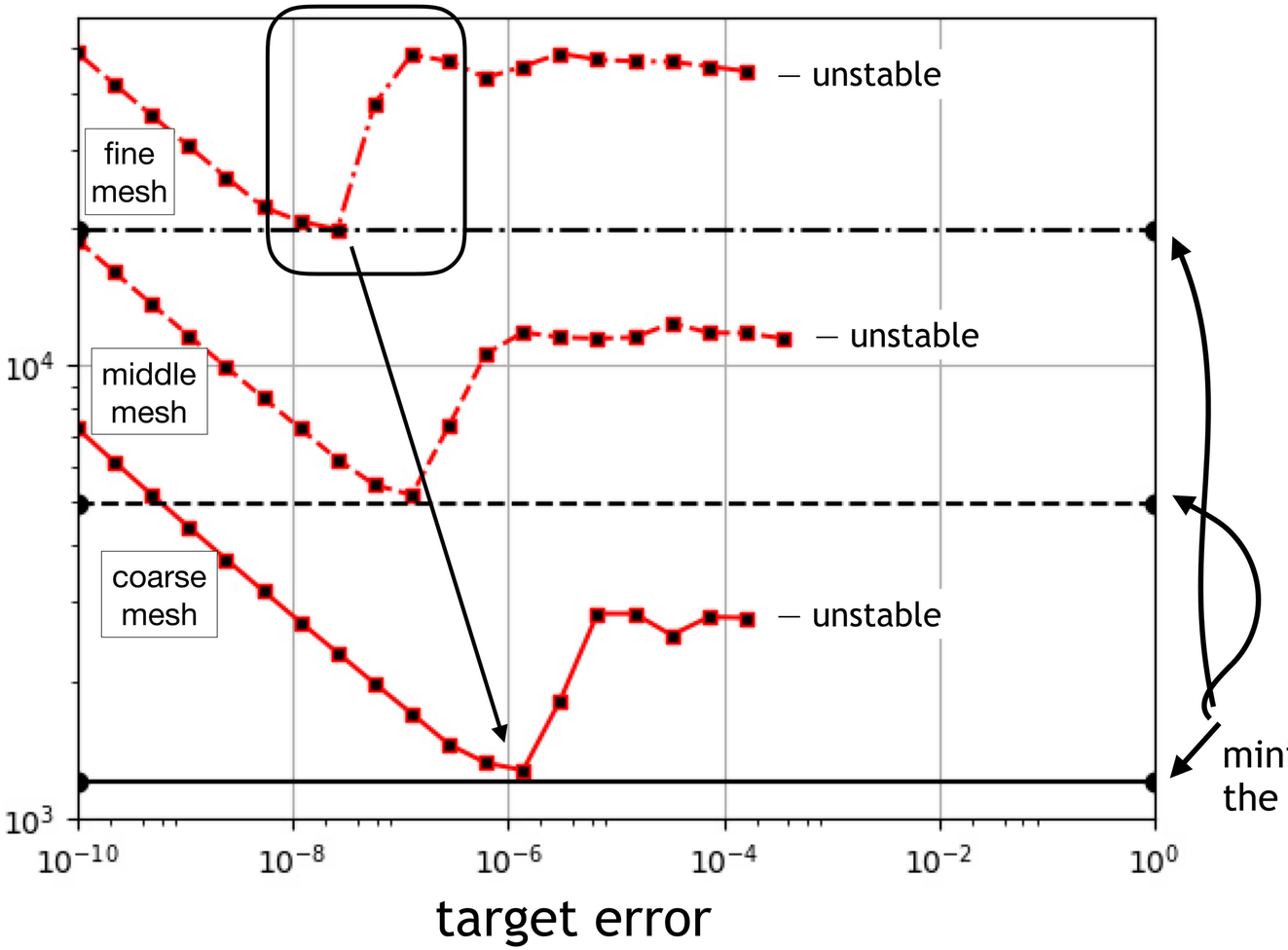


minimal work for each mesh, found by bisecting the maximum fixed time step size possible



Adaptive time-stepping is capable of running at the stability boundary, but only for a tiny window of the target error.

minimal work for each mesh, found by bisecting the maximum fixed time step size possible

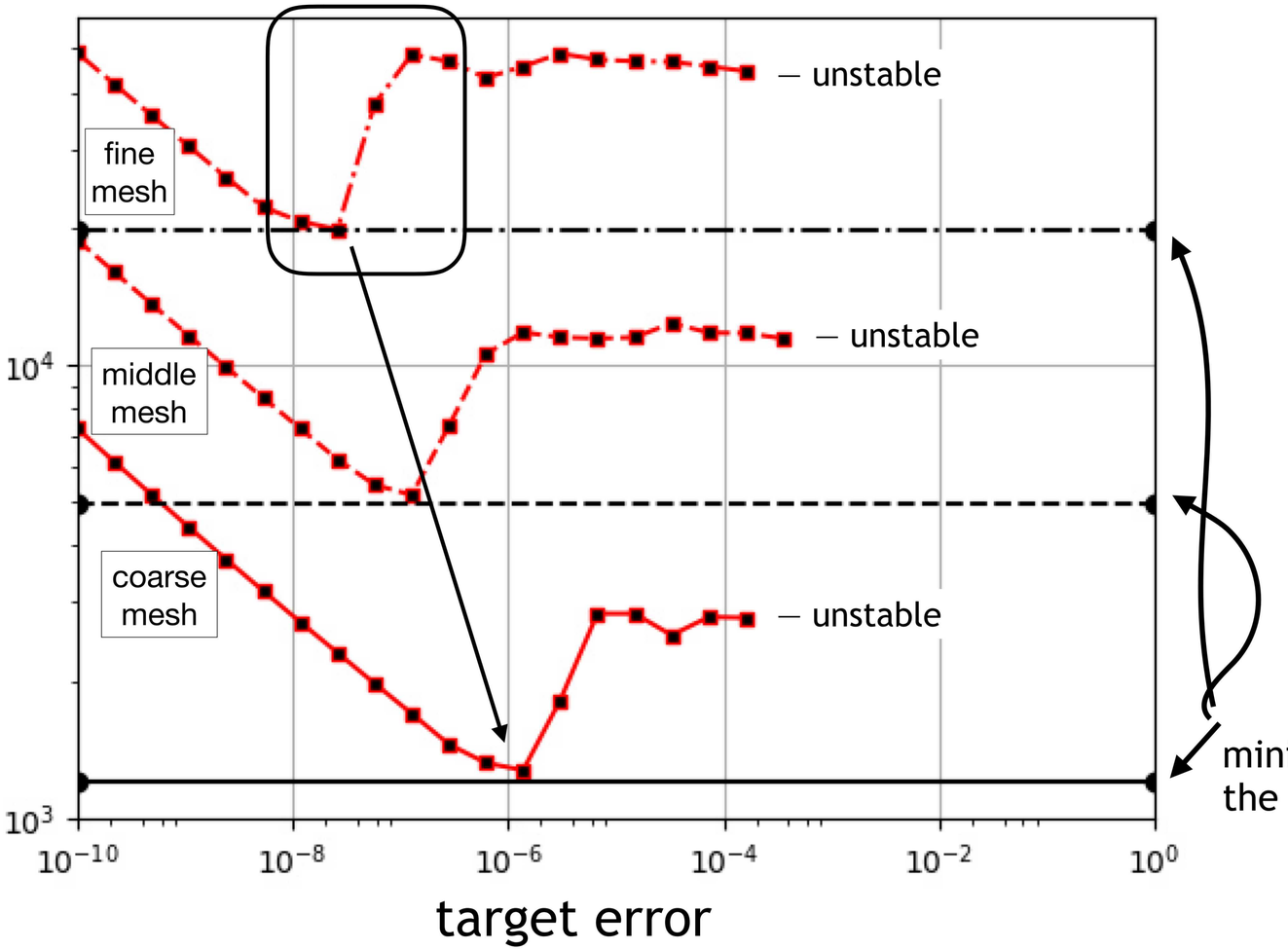


Adaptive time-stepping is capable of running at the stability boundary, but only for a tiny window of the target error.

This window depends on the mesh resolution and specific problem. Guess-and-check!

minimal work for each mesh, found by bisecting the maximum fixed time step size possible

Error-control offers a problem-independent ***approach*** but current methods need to be ‘tuned’ per problem



Adaptive time-stepping is capable of running at the stability boundary, but only for a tiny window of the target error.

This window depends on the mesh resolution and specific problem. Guess-and-check!

minimal work for each mesh, found by bisecting the maximum fixed time step size possible

Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

Constraints on time integrator

- Don’t want it to blow up
- Don’t want ‘too much’ time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!

Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

Several sources of error

- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

dominant sources

Constraints on time integrator

- Don’t want it to blow up
- Don’t want ‘too much’ time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!

Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

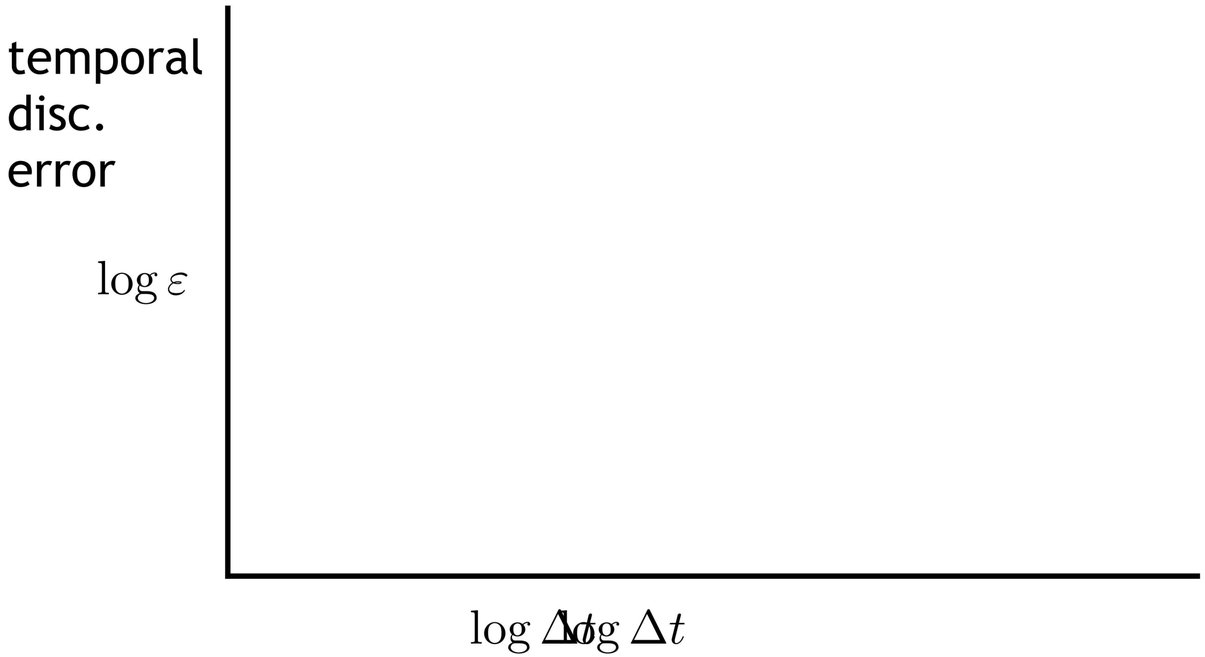
Several sources of error

- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

dominant sources

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!



Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

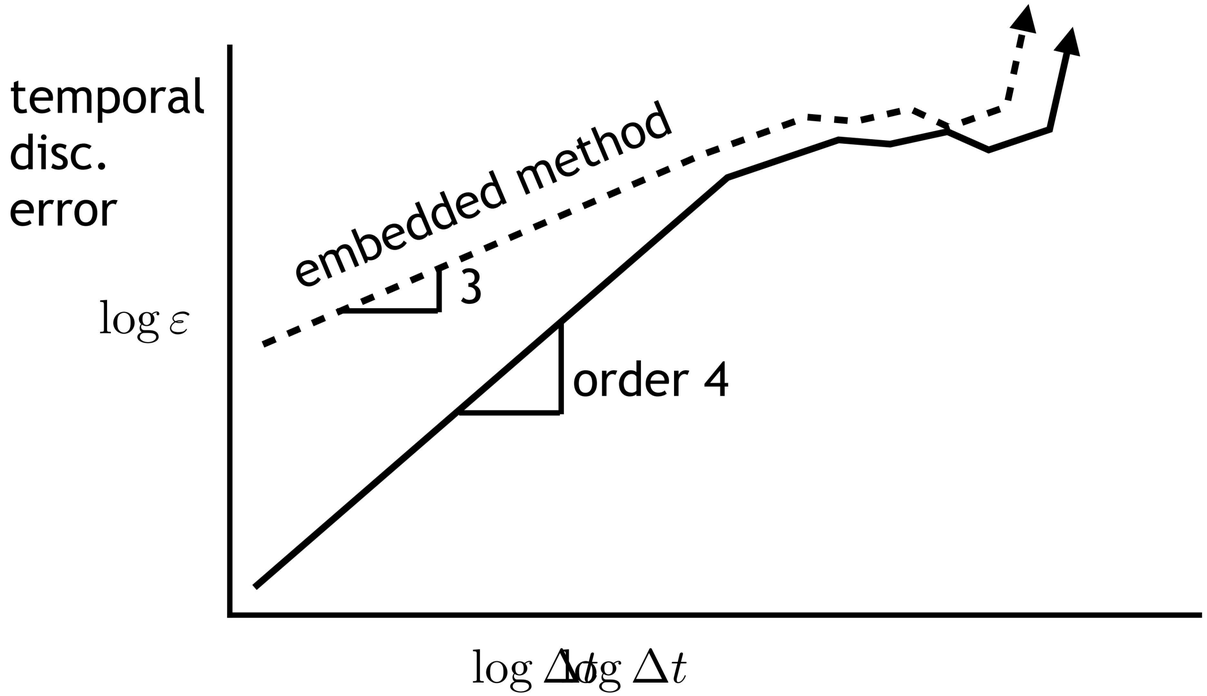
Several sources of error

- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

dominant sources

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!



Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

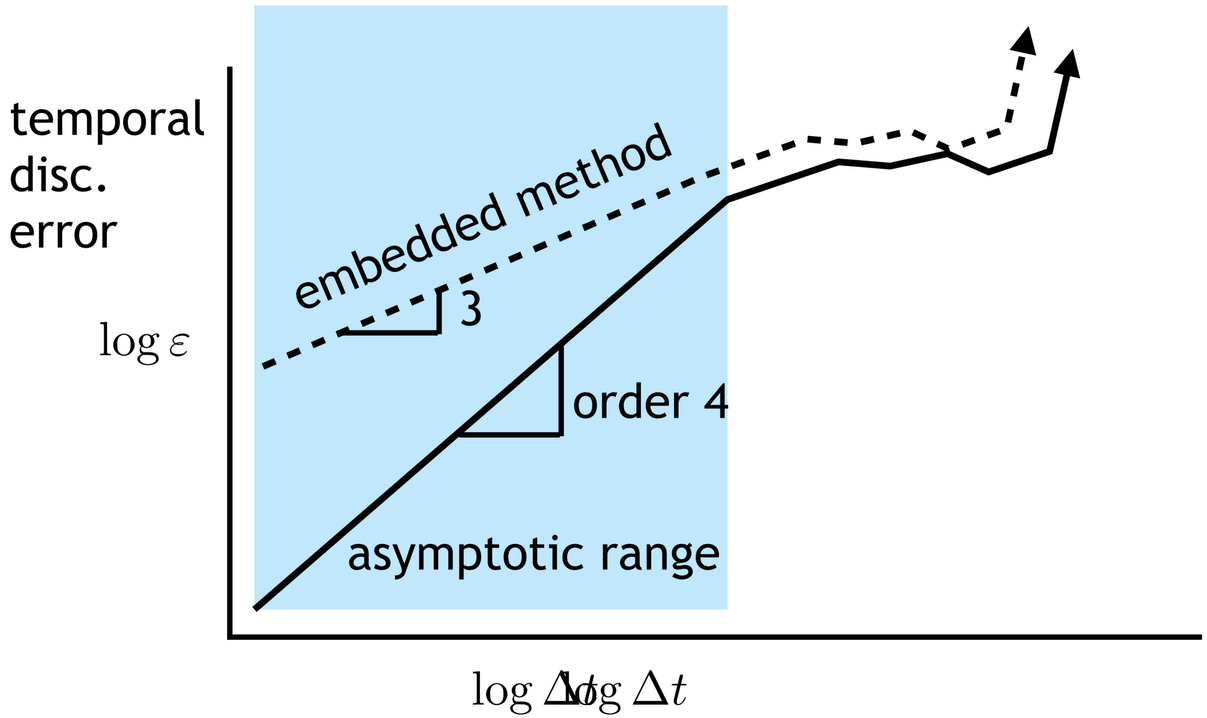
Several sources of error

- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

dominant sources

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!



Manipulate the time step to keep
the error estimate at a target value

Sophisticated PID controllers can do this!

But what do I use for the target error?!

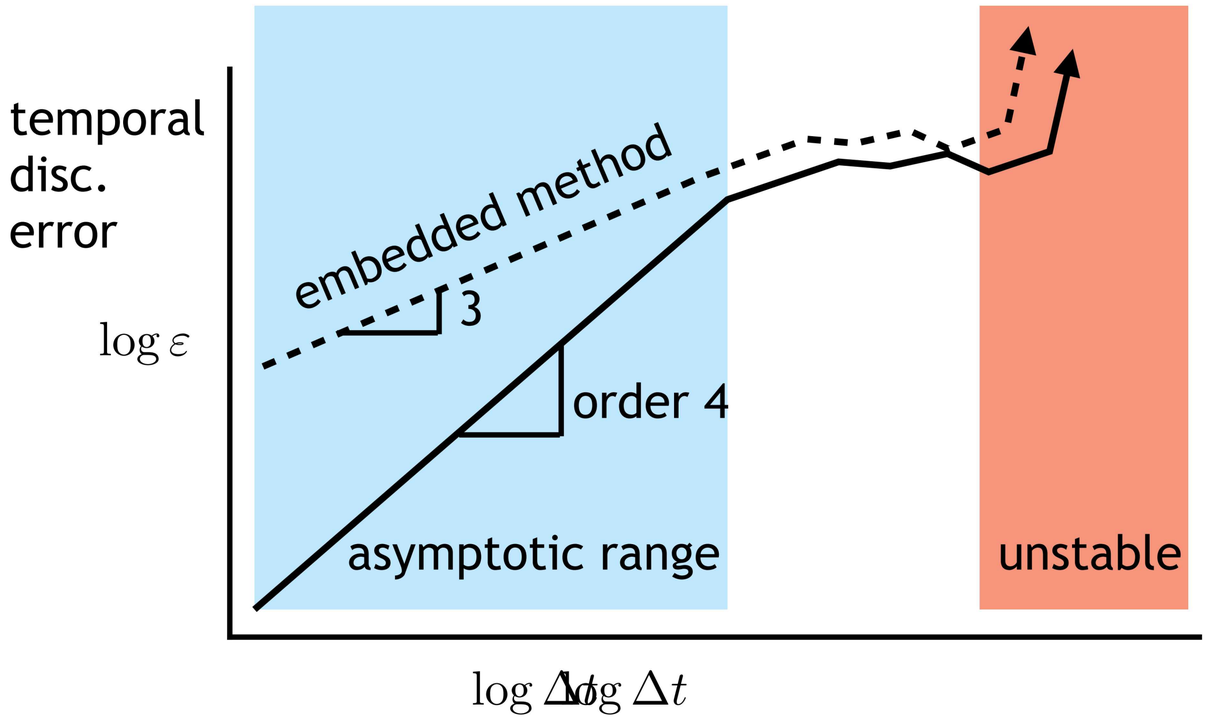
Several sources of error

- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

dominant sources

Constraints on time integrator

- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!



Manipulate the time step to keep
the error estimate at a target value

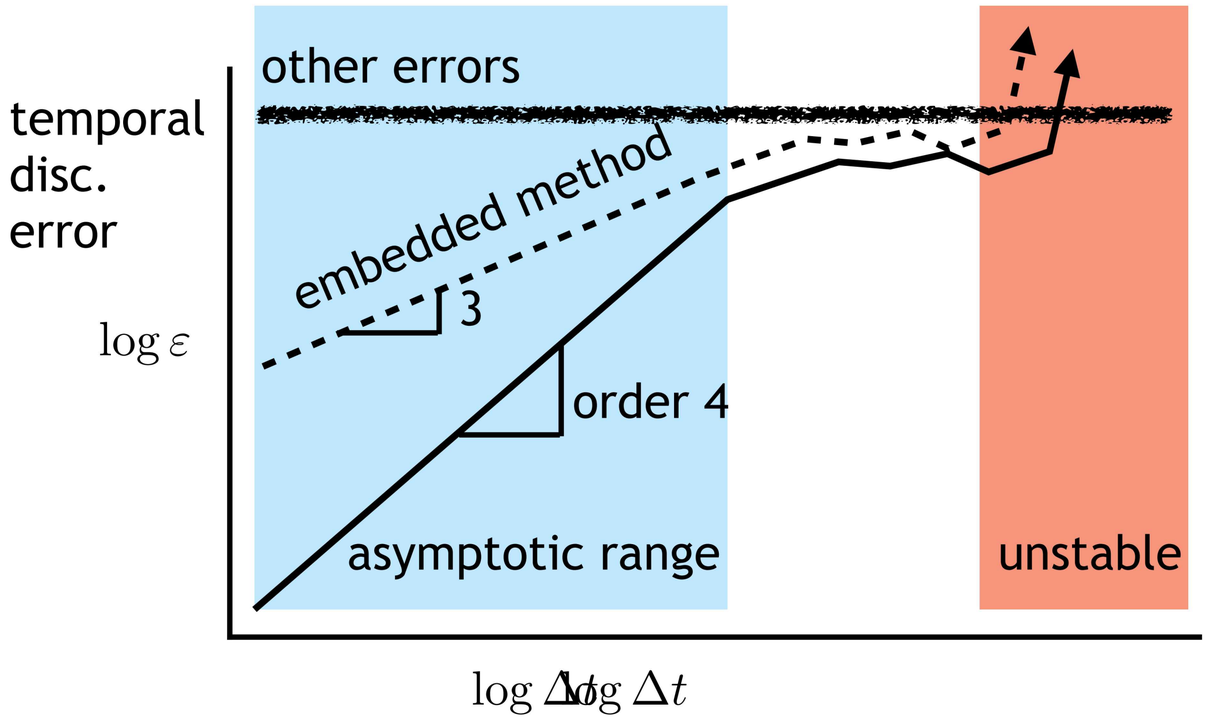
Sophisticated PID controllers can do this!

But what do I use for the target error?!

Several sources of error
- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

dominant sources

Constraints on time integrator
- Don't want it to blow up
- Don't want 'too much' time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!



We don’t want to control the temporal error, we only want to keep the simulation stable

Manipulate the time step to keep
the error estimate at a target value

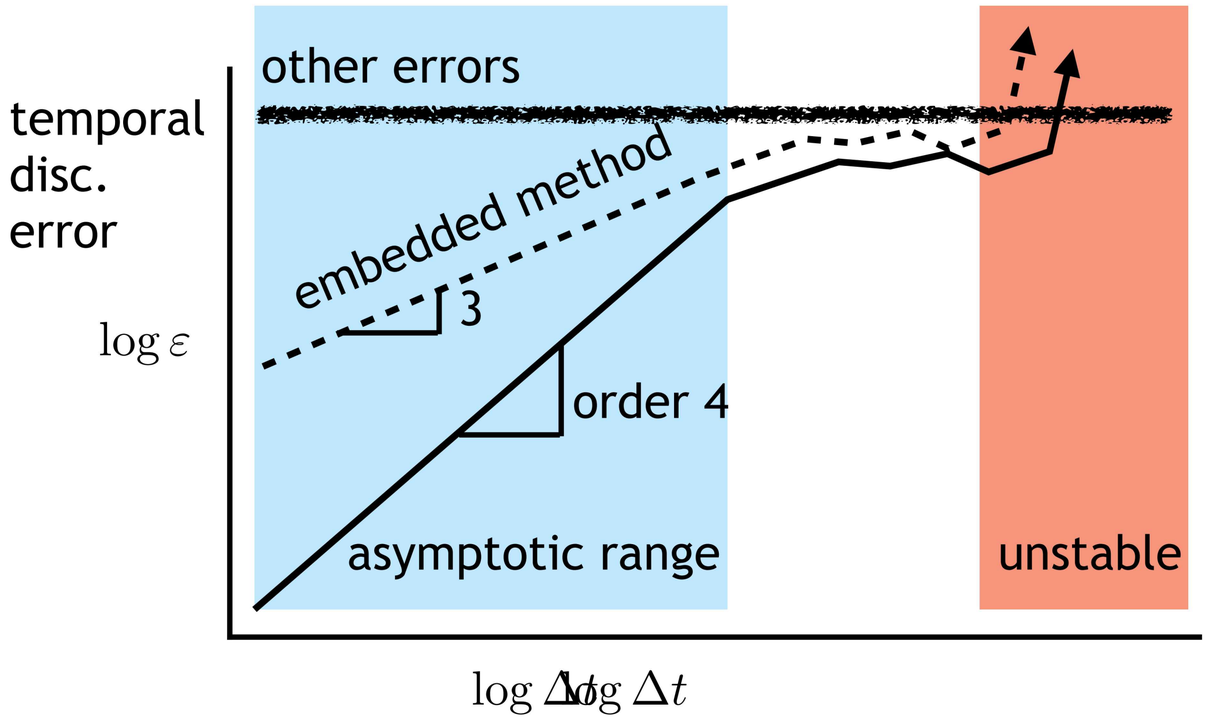
Sophisticated PID controllers can do this!

But what do I use for the target error?!

Several sources of error
- Temporal disc. error
- Spatial disc. error
- Model error / uncertainty

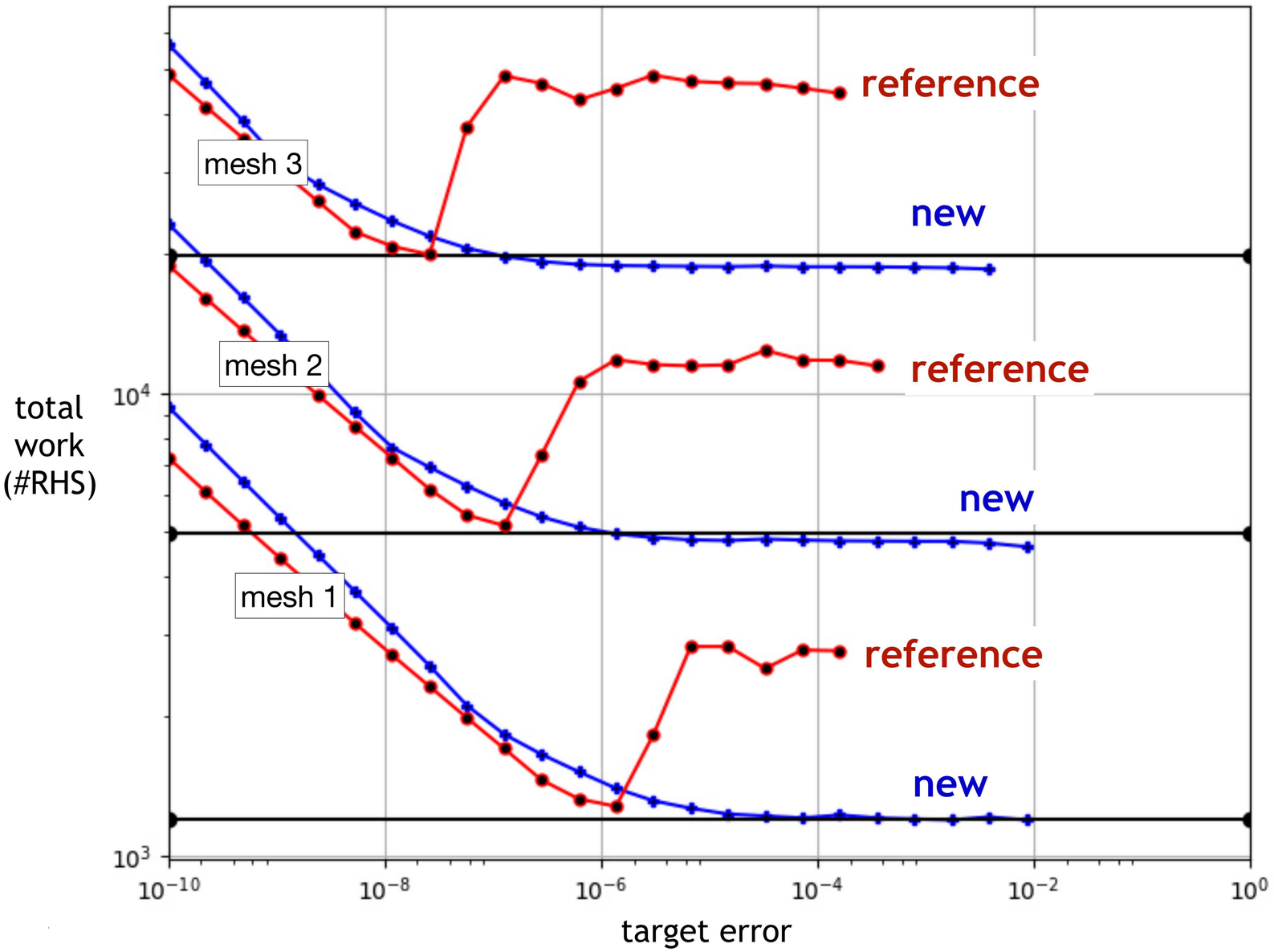
dominant sources

Constraints on time integrator
- Don’t want it to blow up
- Don’t want ‘too much’ time discretization error
- Minimal cost = fastest simulation = aggressive stepping
- Guess-and-check *for every variety* is not acceptable!

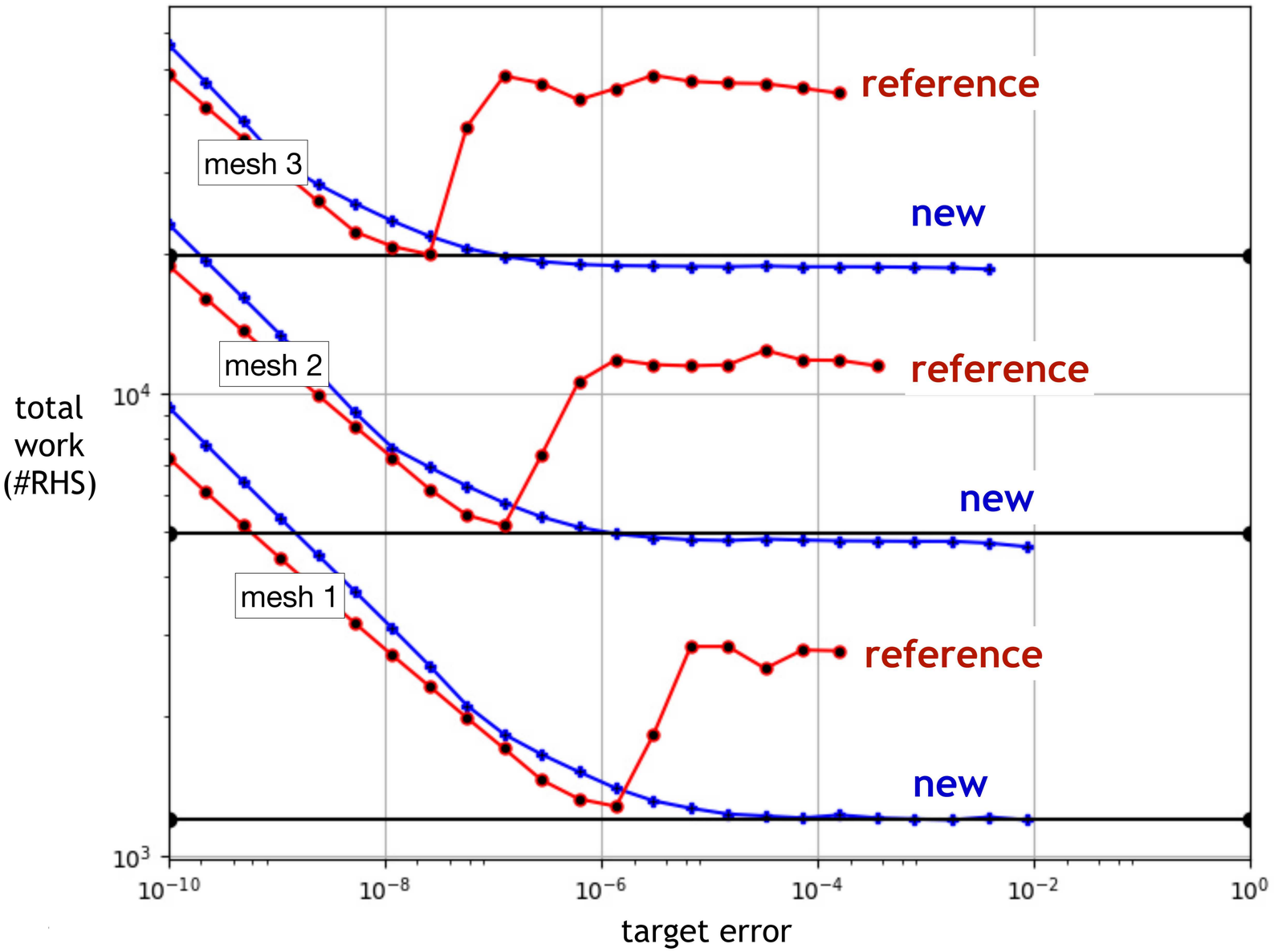


Can we do error control better, knowing that we only want stability?

Can we do error control better, knowing that we only want stability?



Can we do error control better, knowing that we only want stability?

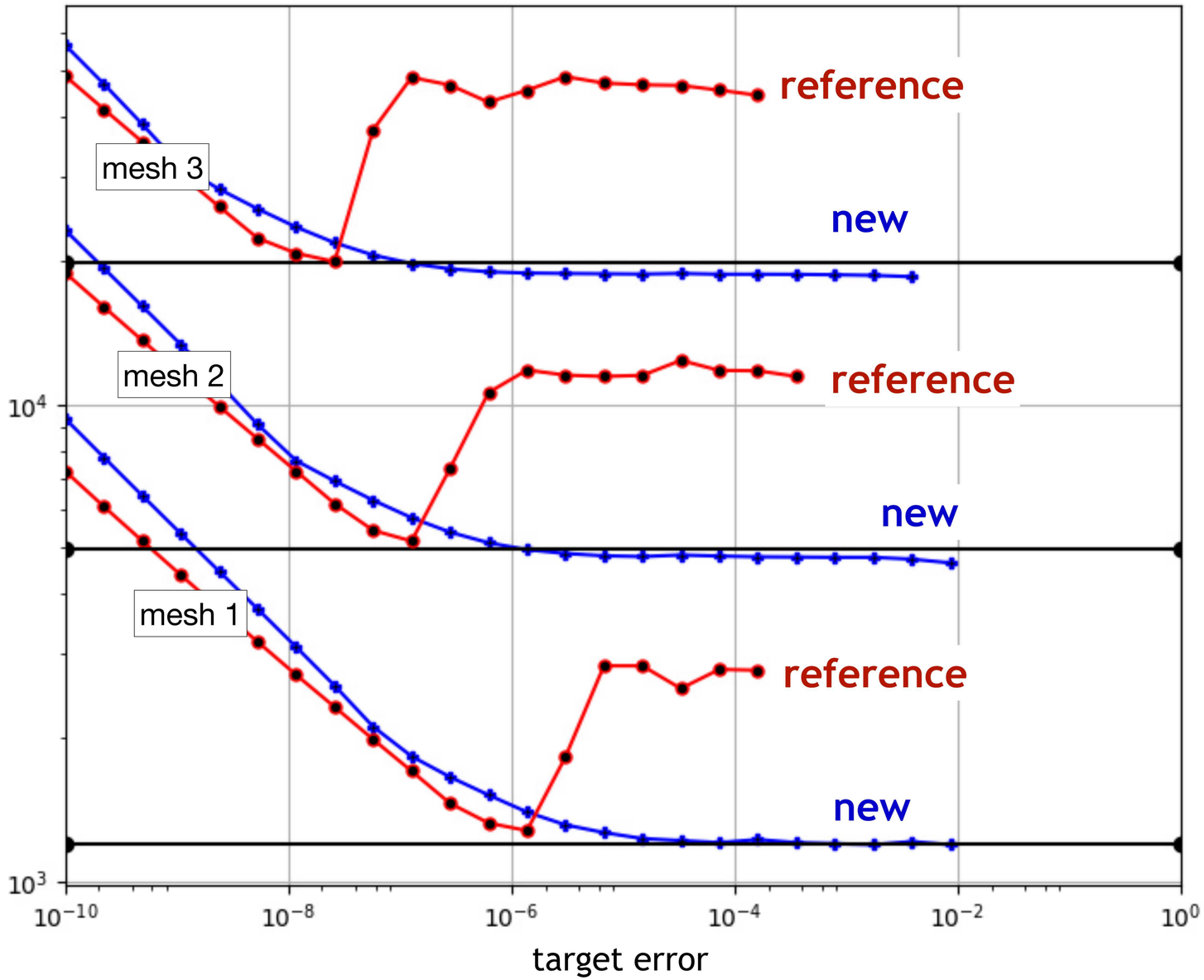


We can do much better!

A newly-designed method shows far less sensitivity to the target error or mesh

It runs efficiently at the stability boundary, nicely minimizing computational cost

Can we do error control better, knowing that we only want stability?



total work (#RHS)

We can do much better!

A newly-designed method shows far less sensitivity to the target error or mesh

It runs efficiently at the stability boundary, nicely minimizing computational cost

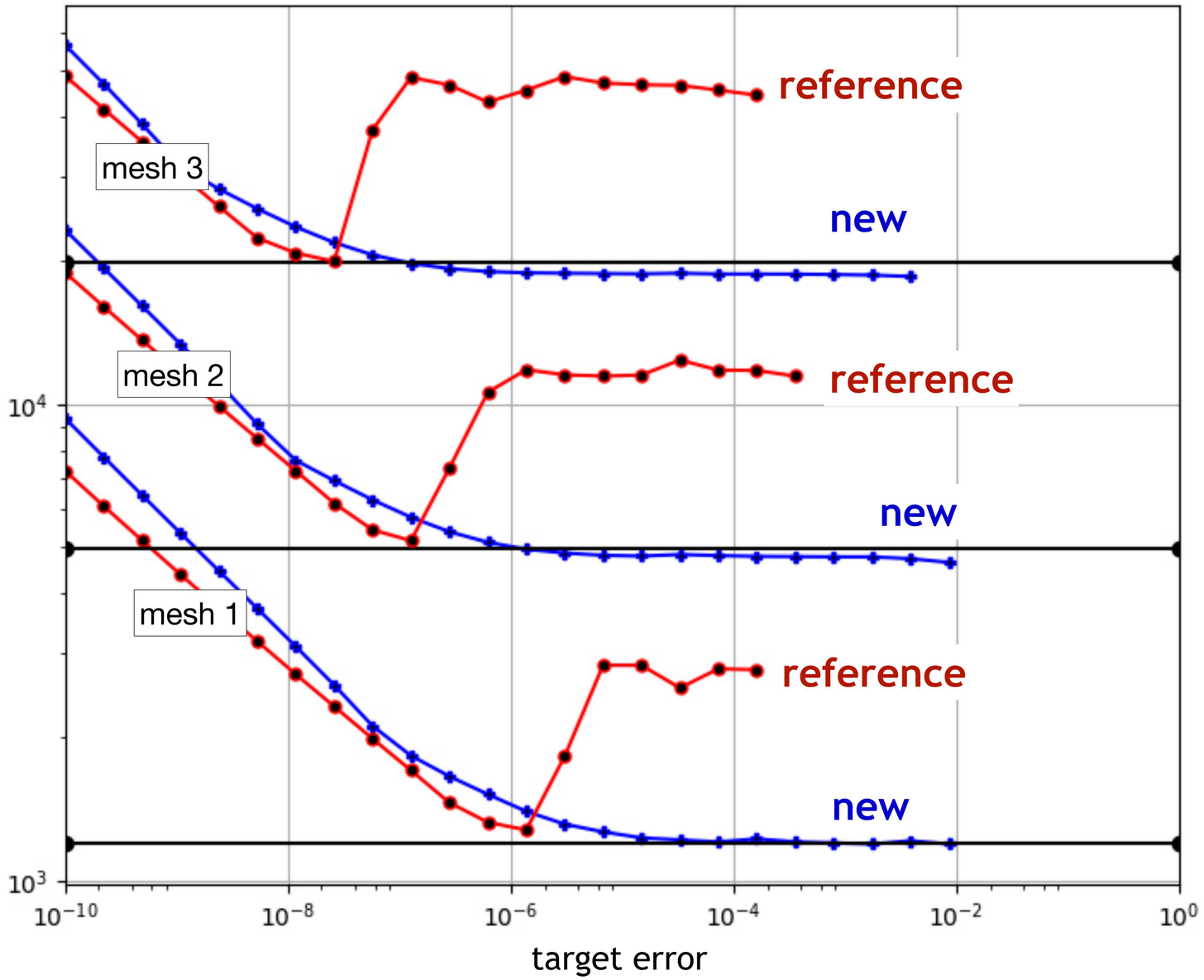
$$e_e = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) r_i(u_i)$$

order (p-1) error estimate

$\{b_i\}$ quadrature coeffs for order p (e.g. 4)

$\{\hat{b}_i\}$ quadrature coeffs for order p-1

Can we do error control better, knowing that we only want stability?



total work (#RHS)

We can do much better!

A newly-designed method shows far less sensitivity to the target error or mesh

It runs efficiently at the stability boundary, nicely minimizing computational cost

$$\hat{b}_i \text{ vs } \hat{b}_i$$

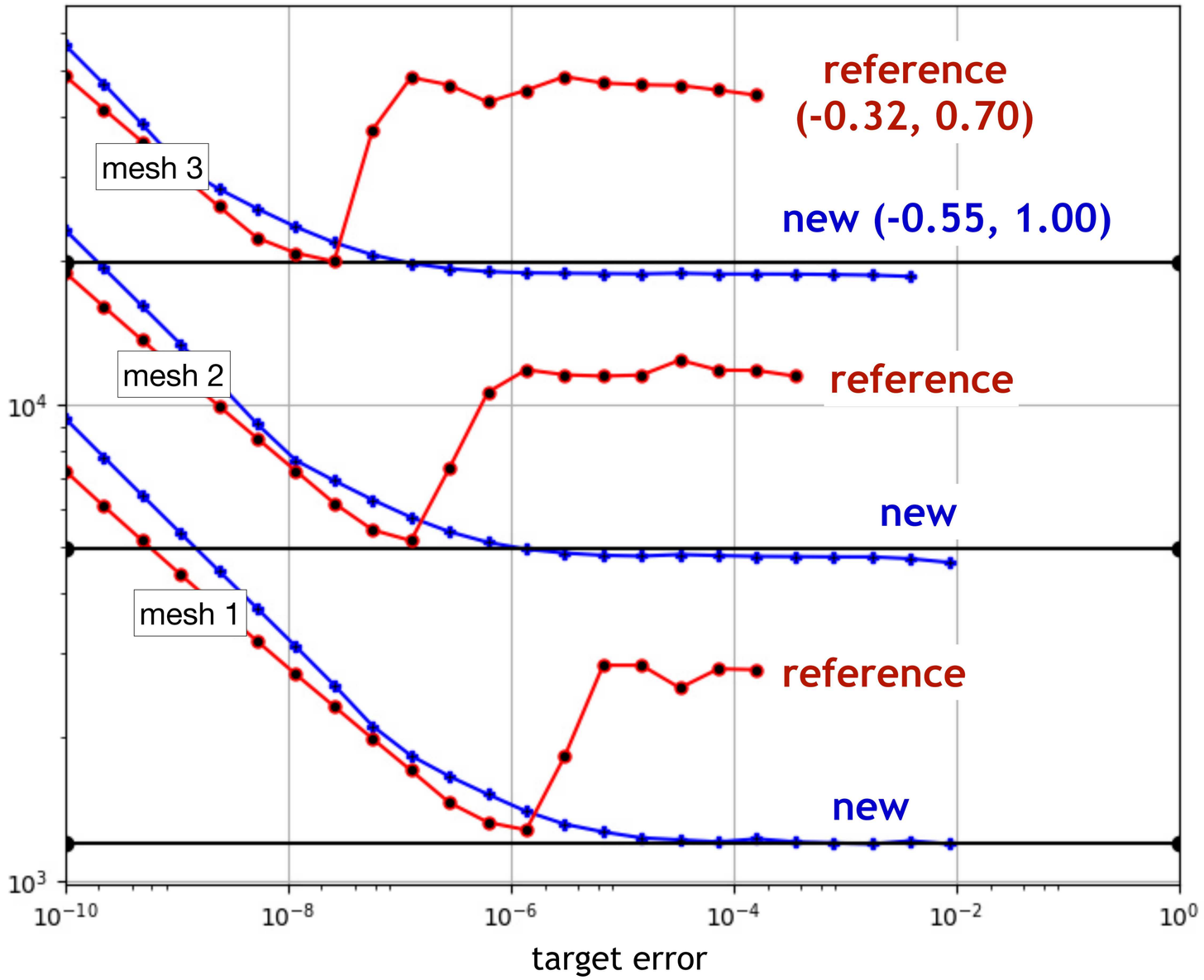
$$e = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) r(u_i)$$

order (p-1) error estimate

$\{b_i\}$ quadrature coeffs for order p (e.g. 4)

$\{\hat{b}_i\}$ quadrature coeffs for order p-1

Design of the embedded error estimator has a major impact on performance



$\hat{b}_i$ vs $\hat{b}_i$

Six stages: 6 b values

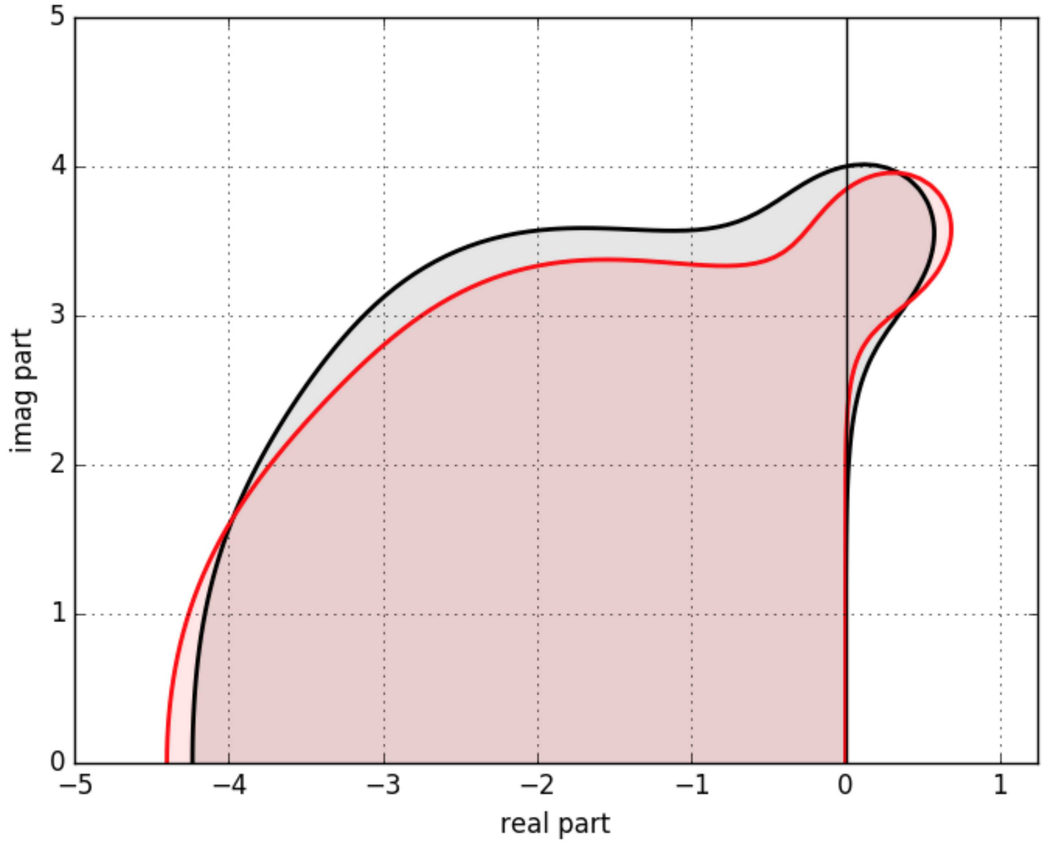
Third-order: 4 conditions

Two degrees of freedom

2D parameter space!

Linear stability properties dramatically impact embedded estimator performance

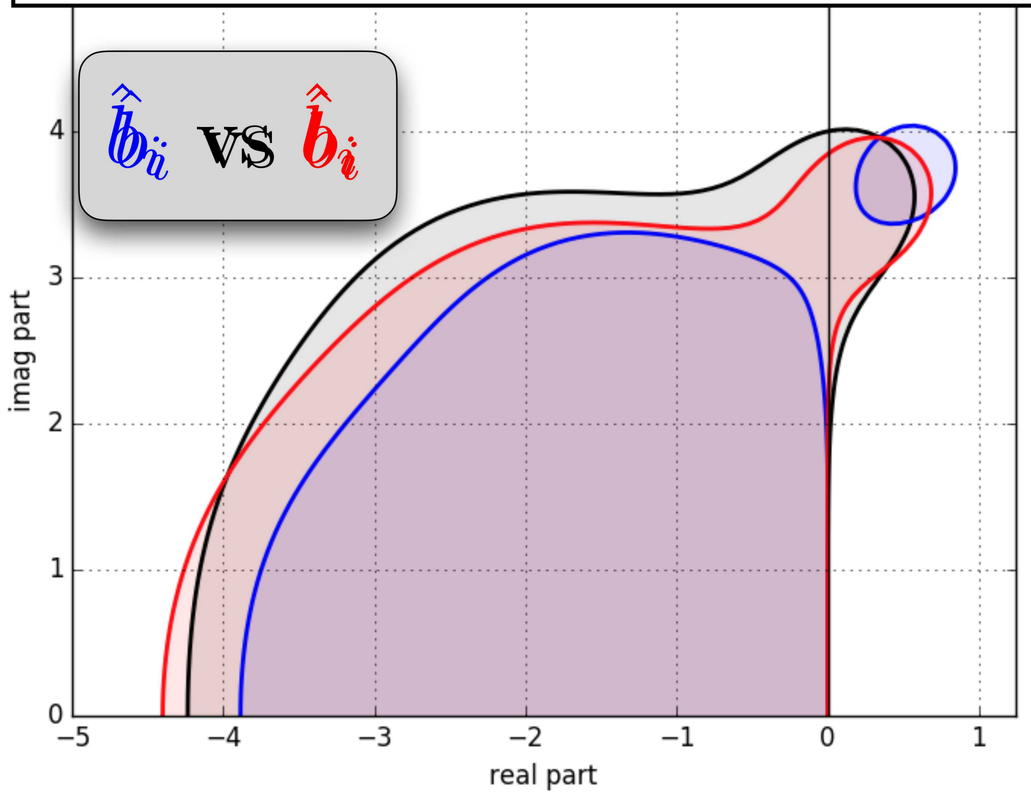
Reference: estimated error remains bounded until solver blows up,
the target error must match this bound precisely!



Linear stability properties dramatically impact embedded estimator performance

Reference: estimated error remains bounded until solver blows up,
the target error must match this bound precisely!

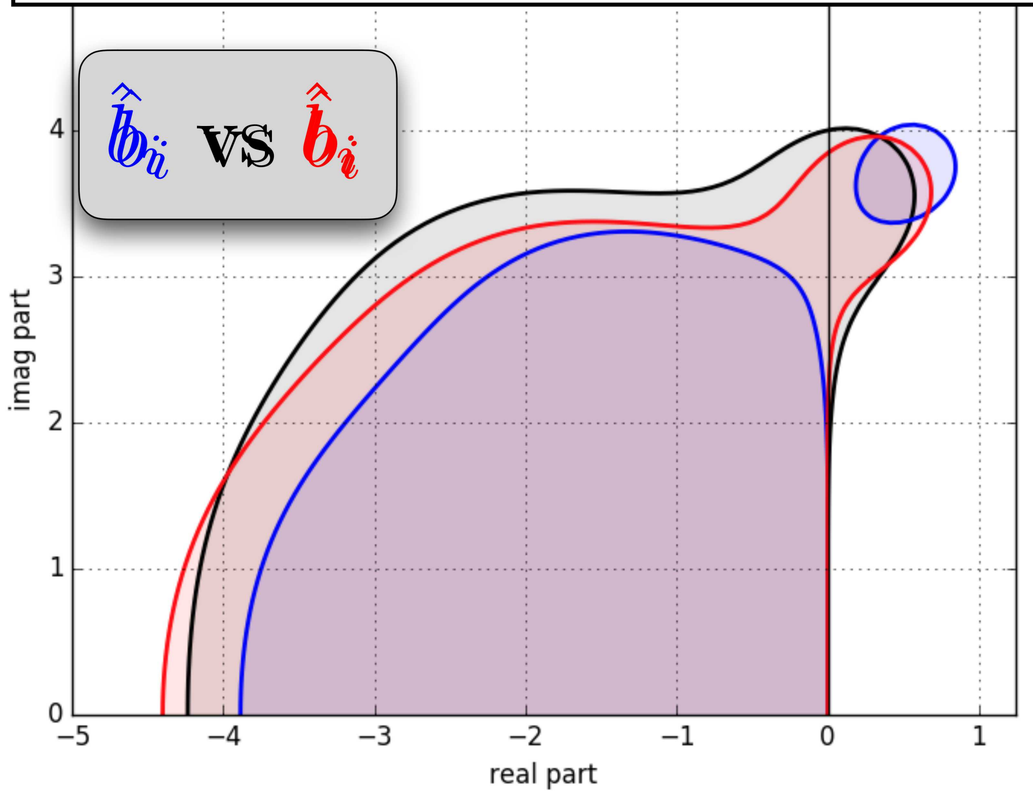
New: estimated error blows up before the solver
the target error is less important

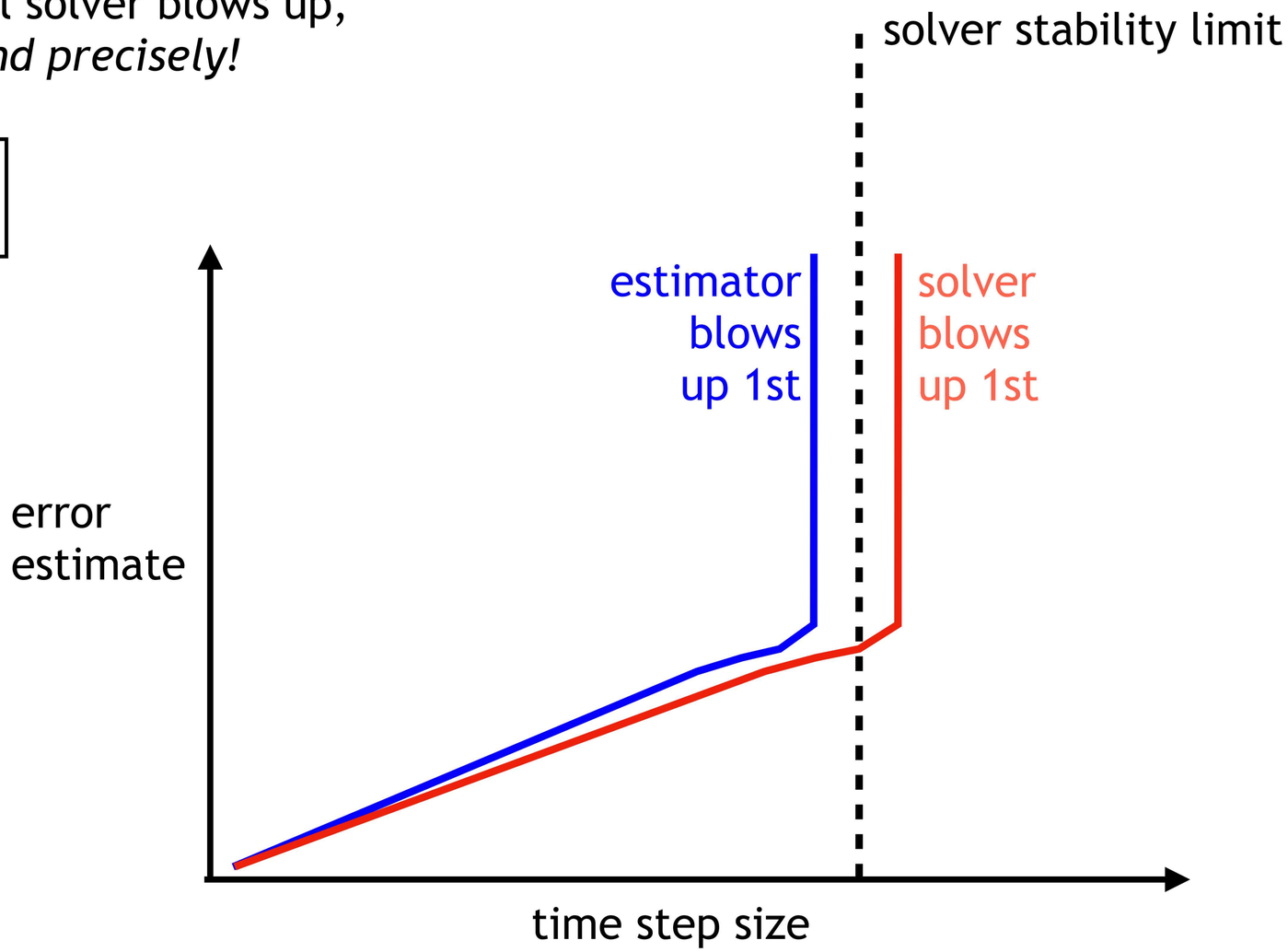


Linear stability properties dramatically impact embedded estimator performance

Reference: estimated error remains bounded until solver blows up,
the target error must match this bound precisely!

New: estimated error blows up before the solver
the target error is less important

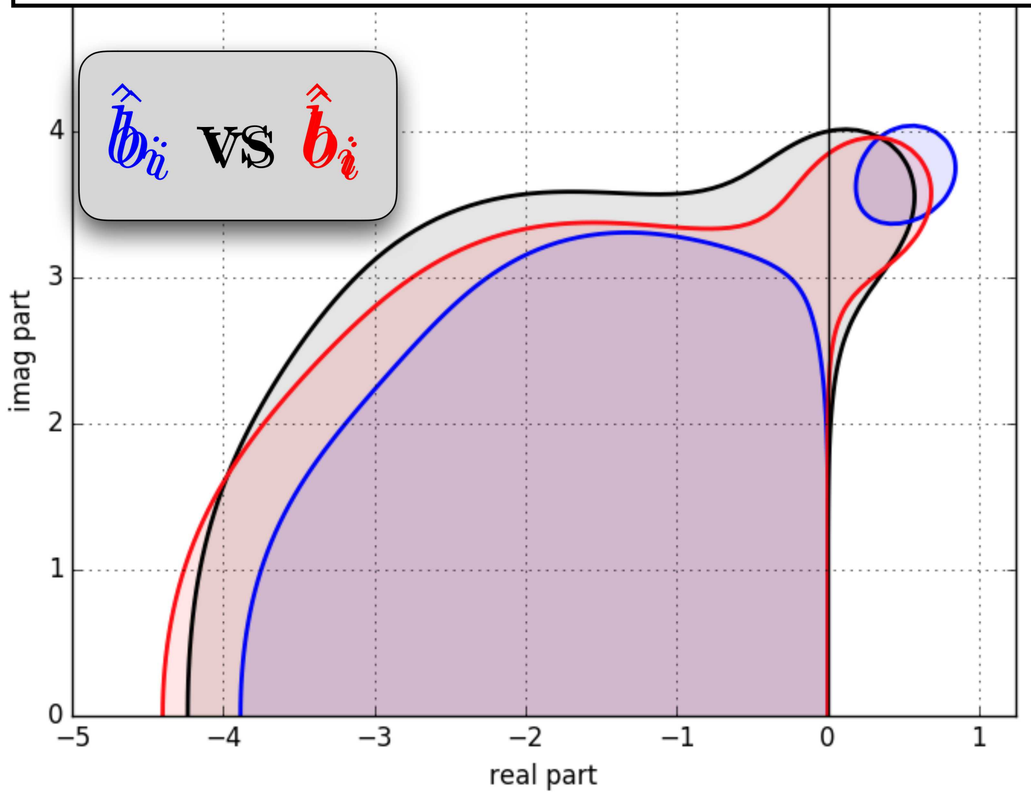


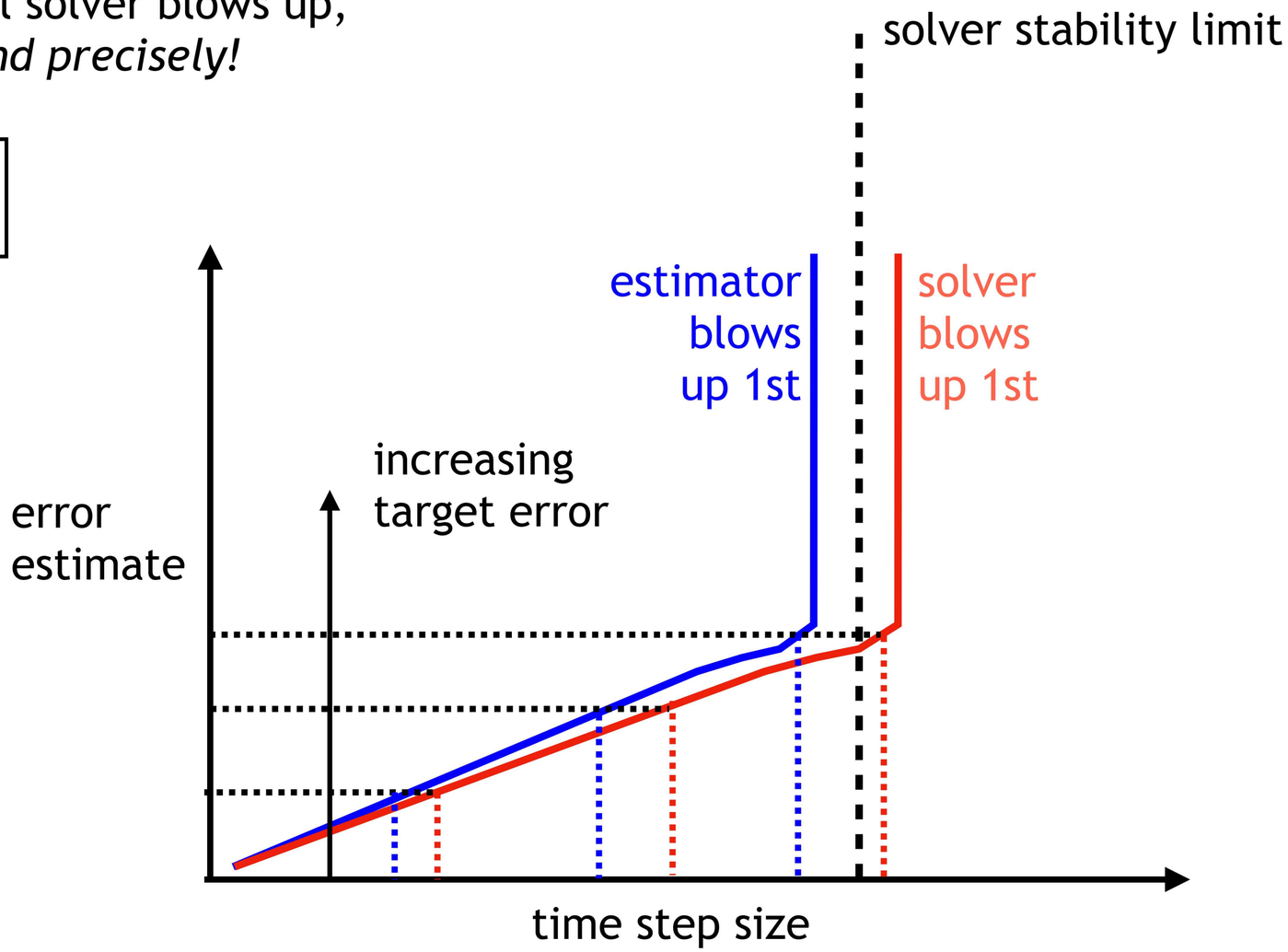


Linear stability properties dramatically impact embedded estimator performance

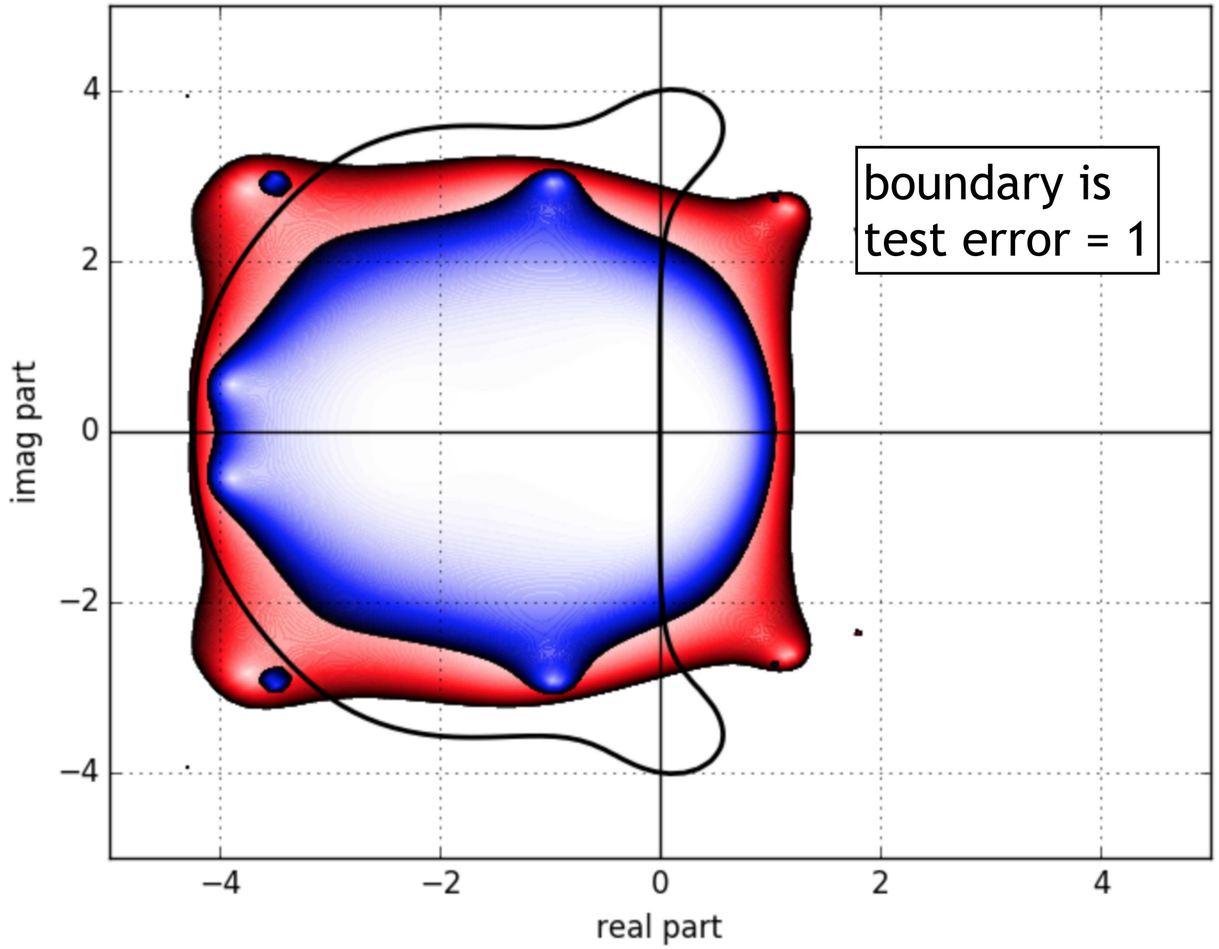
Reference: estimated error remains bounded until solver blows up,
the target error must match this bound precisely!

New: estimated error blows up before the solver
the target error is less important

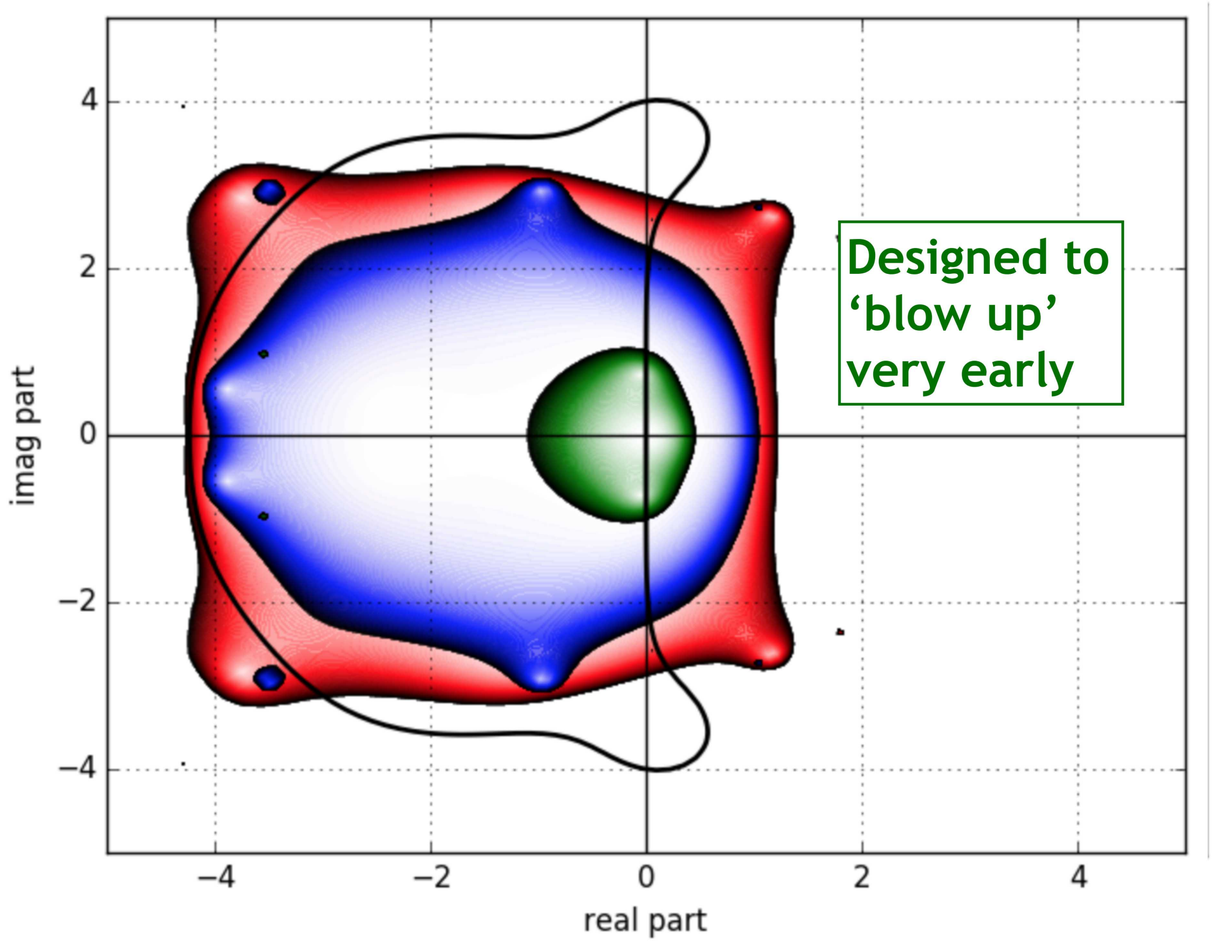




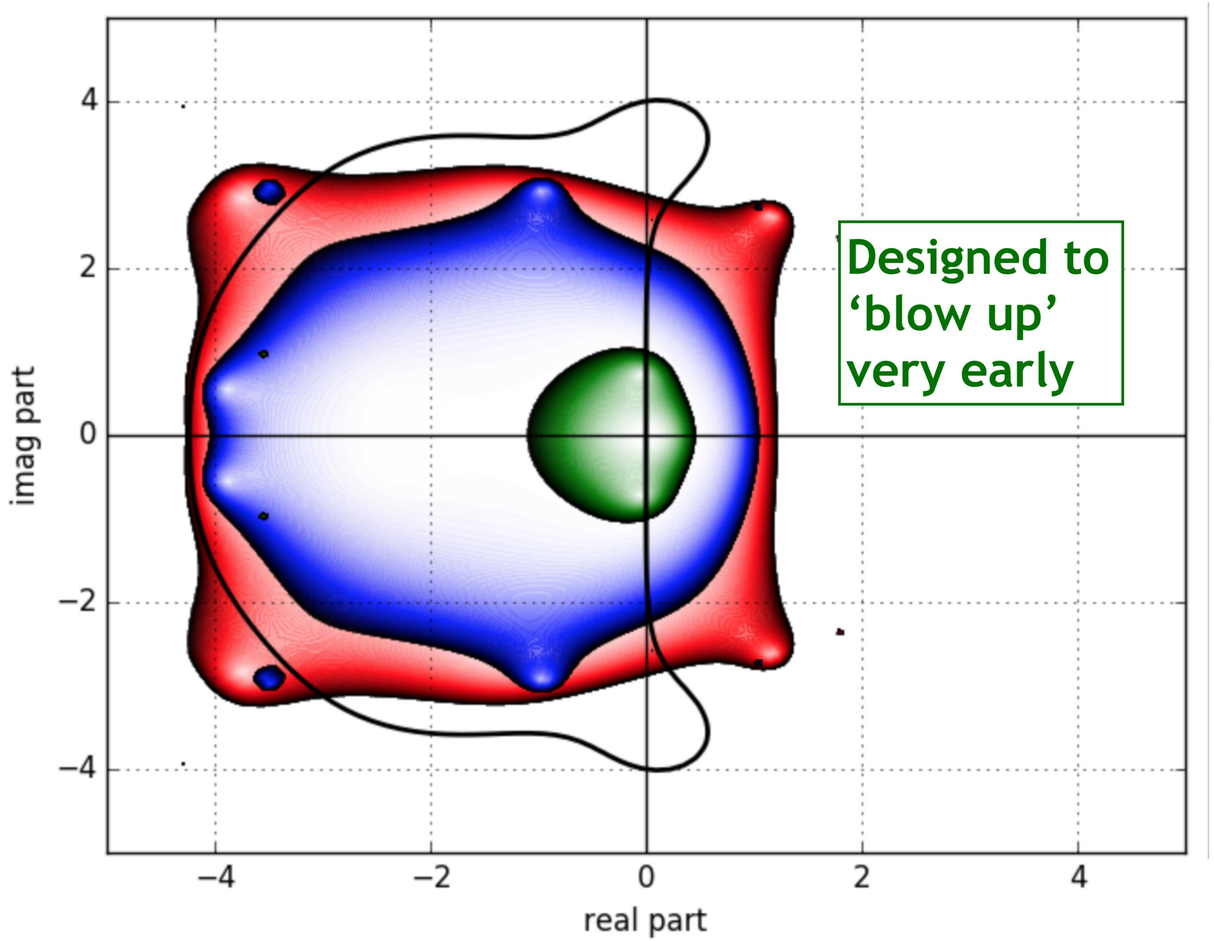
Direct calculation of the error on a nonlinear test equation also shows the distinction between estimators near the stability boundary

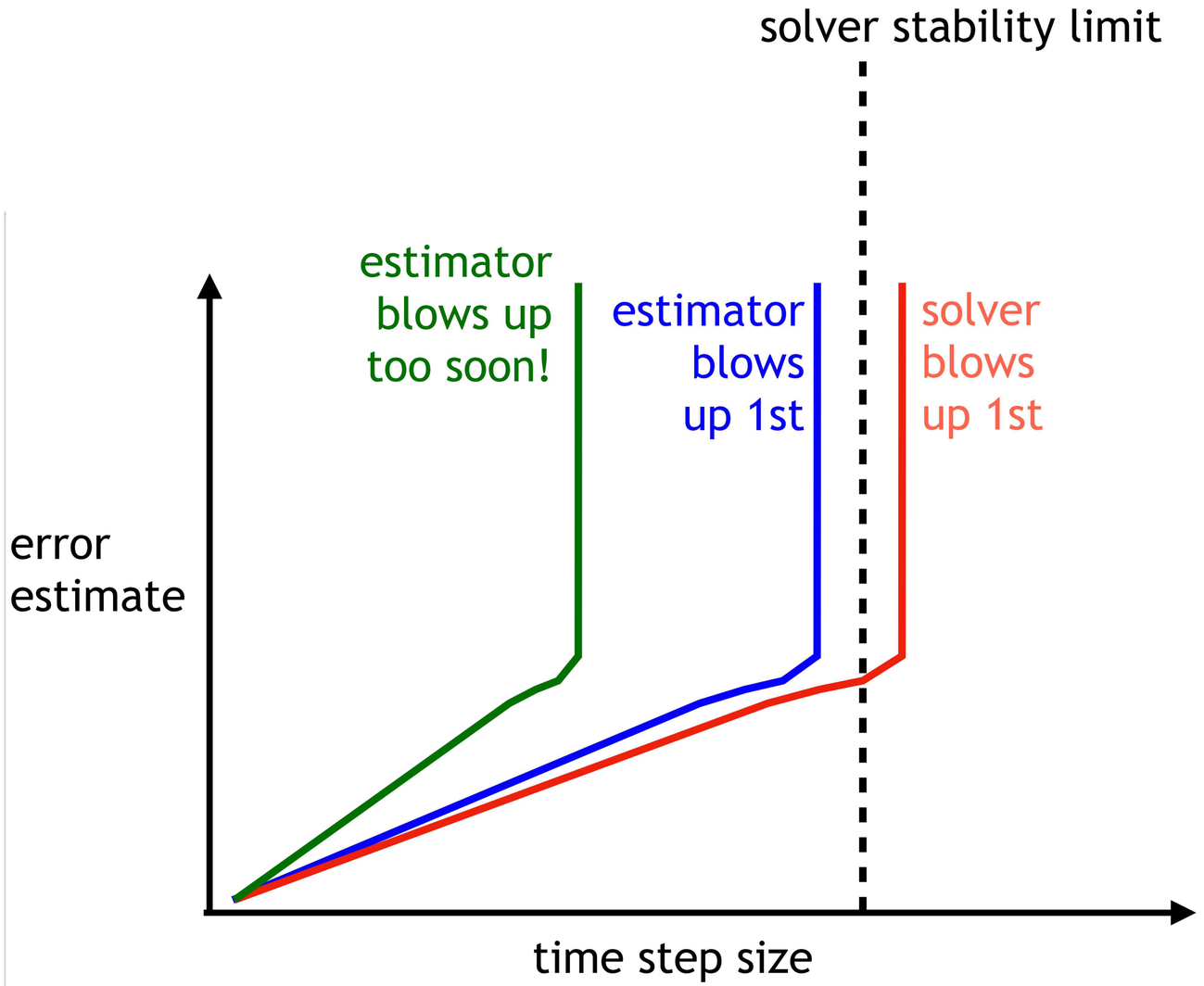


Direct calculation of the error on a nonlinear test equation also shows the distinction between estimators near the stability boundary

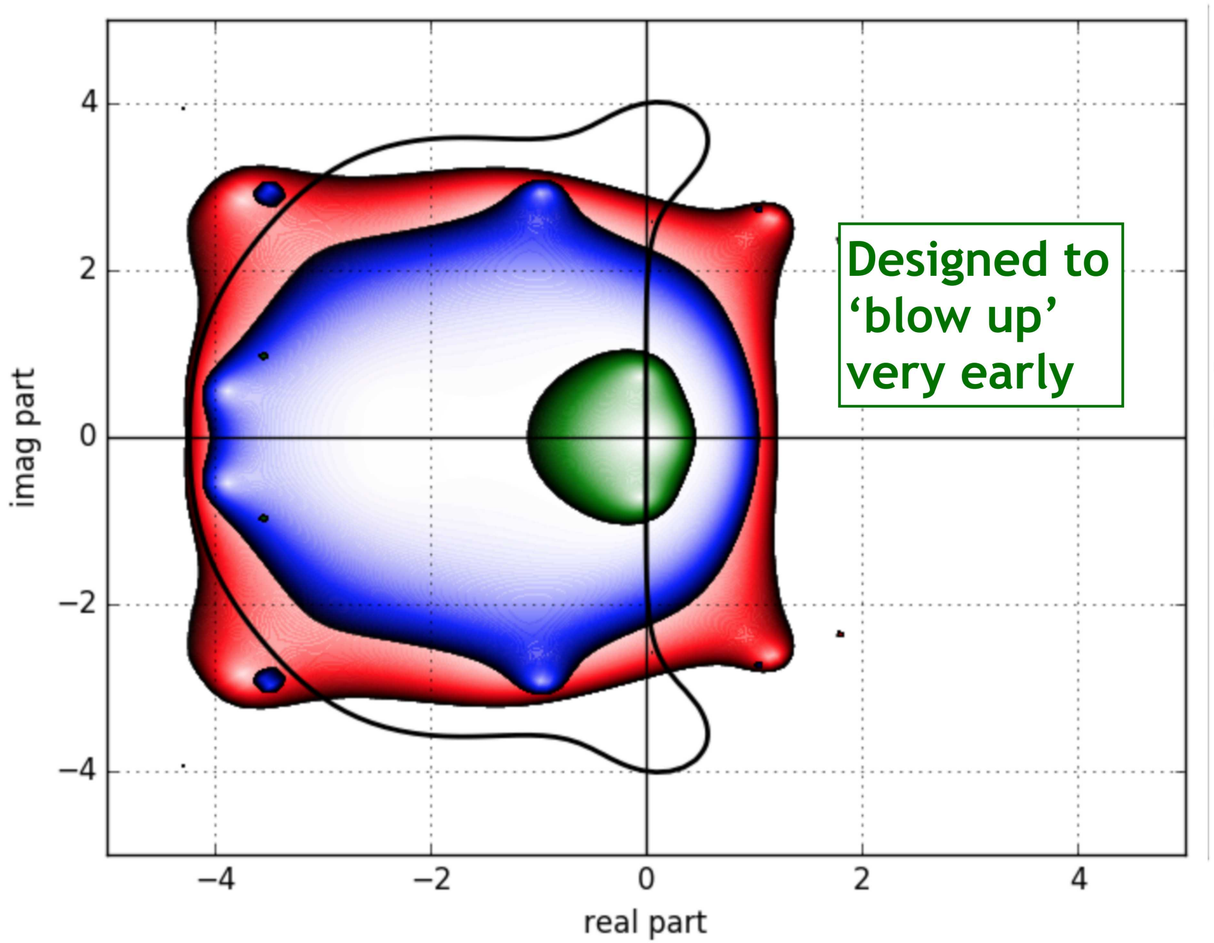


Direct calculation of the error on a nonlinear test equation also shows the distinction between estimators near the stability boundary

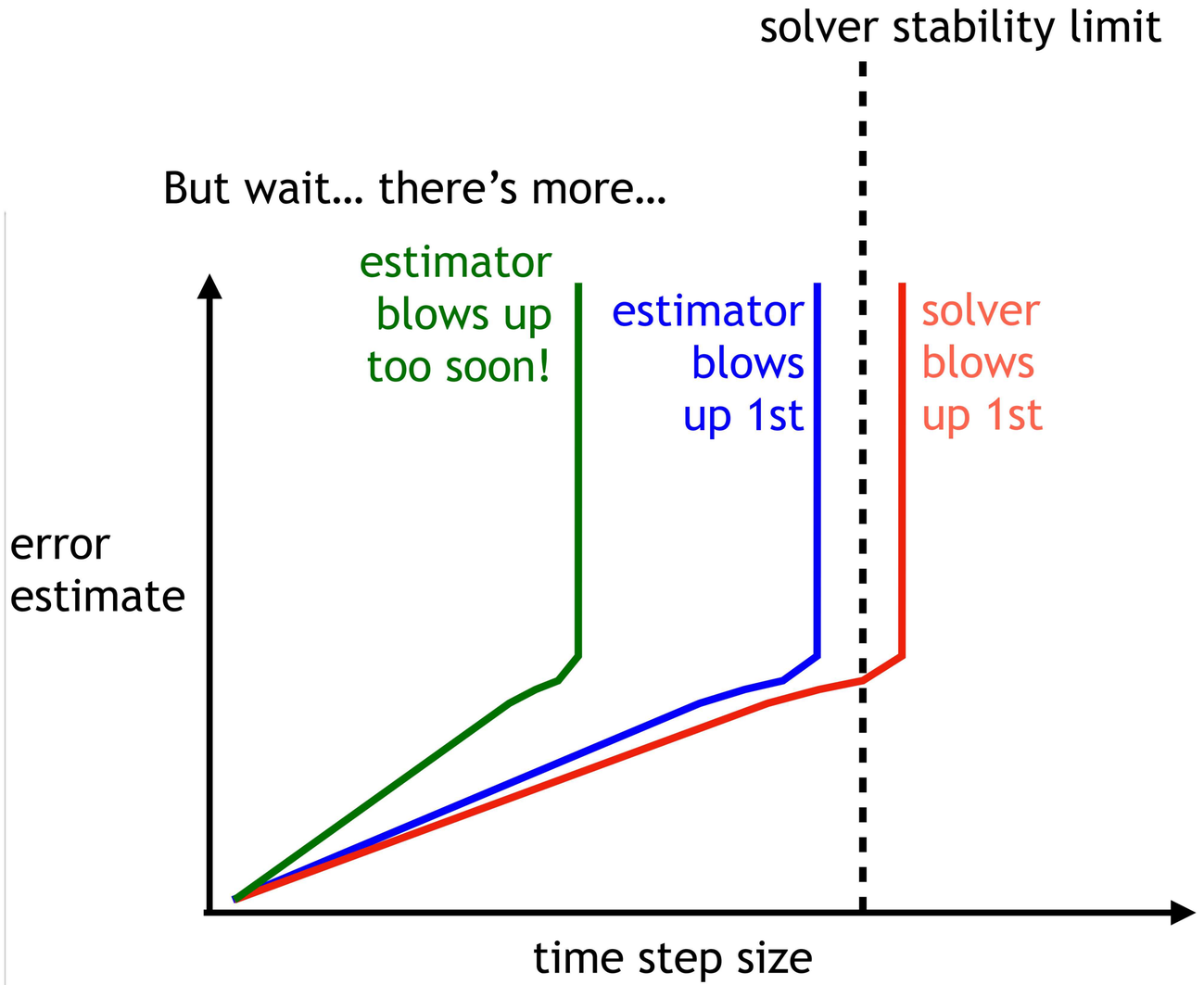




Direct calculation of the error on a nonlinear test equation also shows the distinction between estimators near the stability boundary

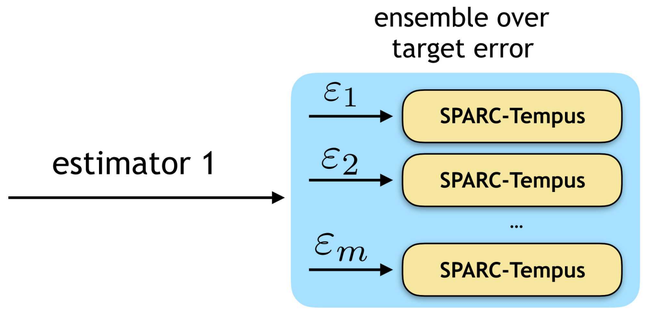


Designed to ‘blow up’ very early

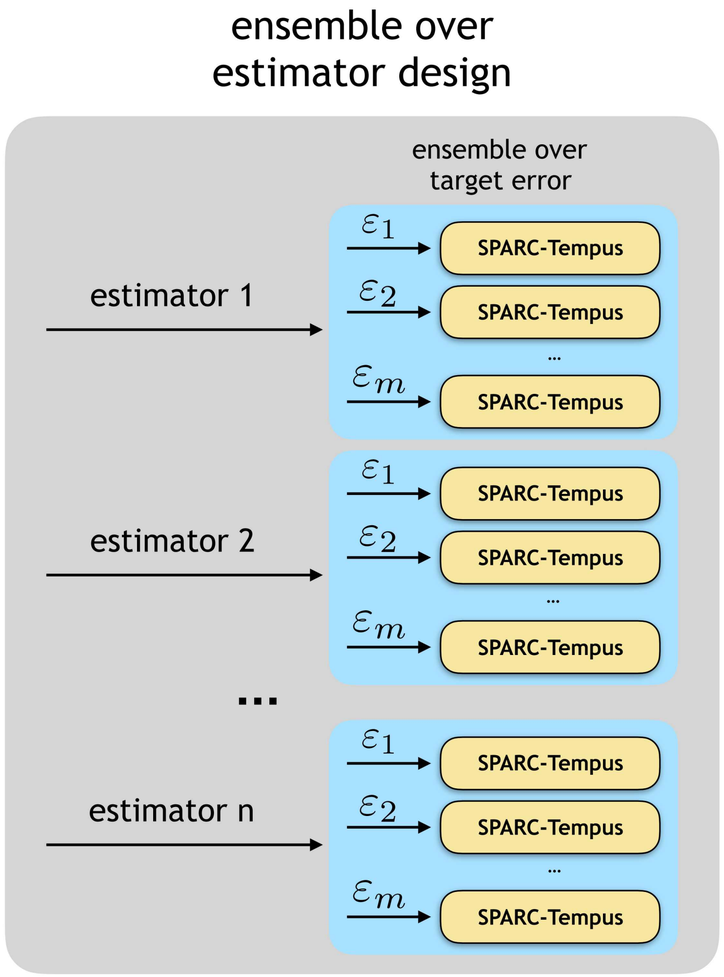


Extensive testing has shown the superiority of relatively unstable embedded estimators

Extensive testing has shown the superiority of relatively unstable embedded estimators

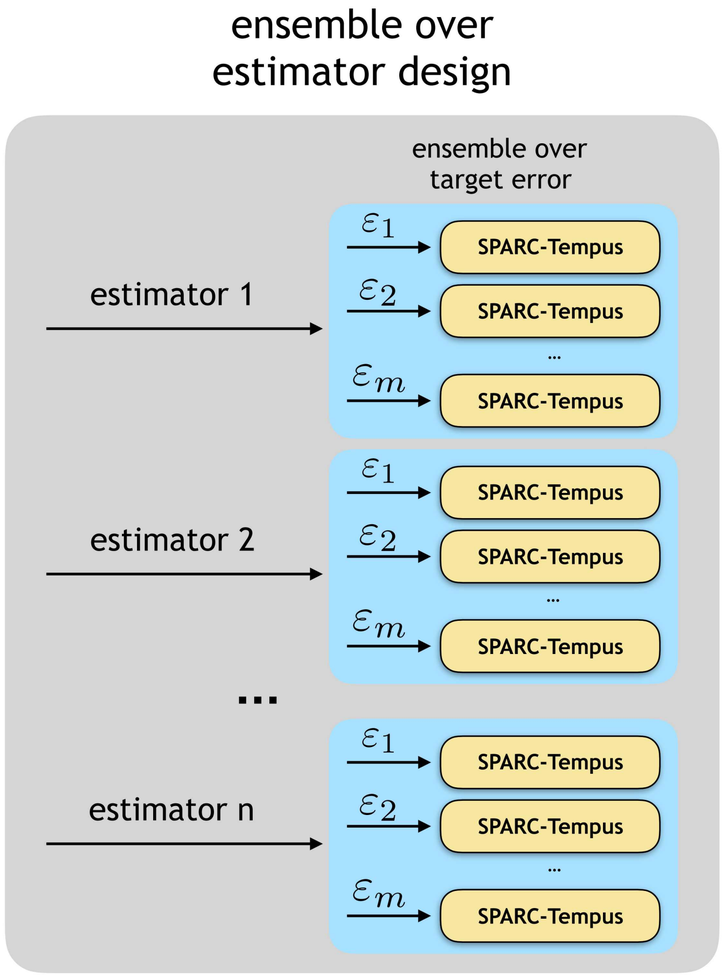


Extensive testing has shown the superiority of relatively unstable embedded estimators

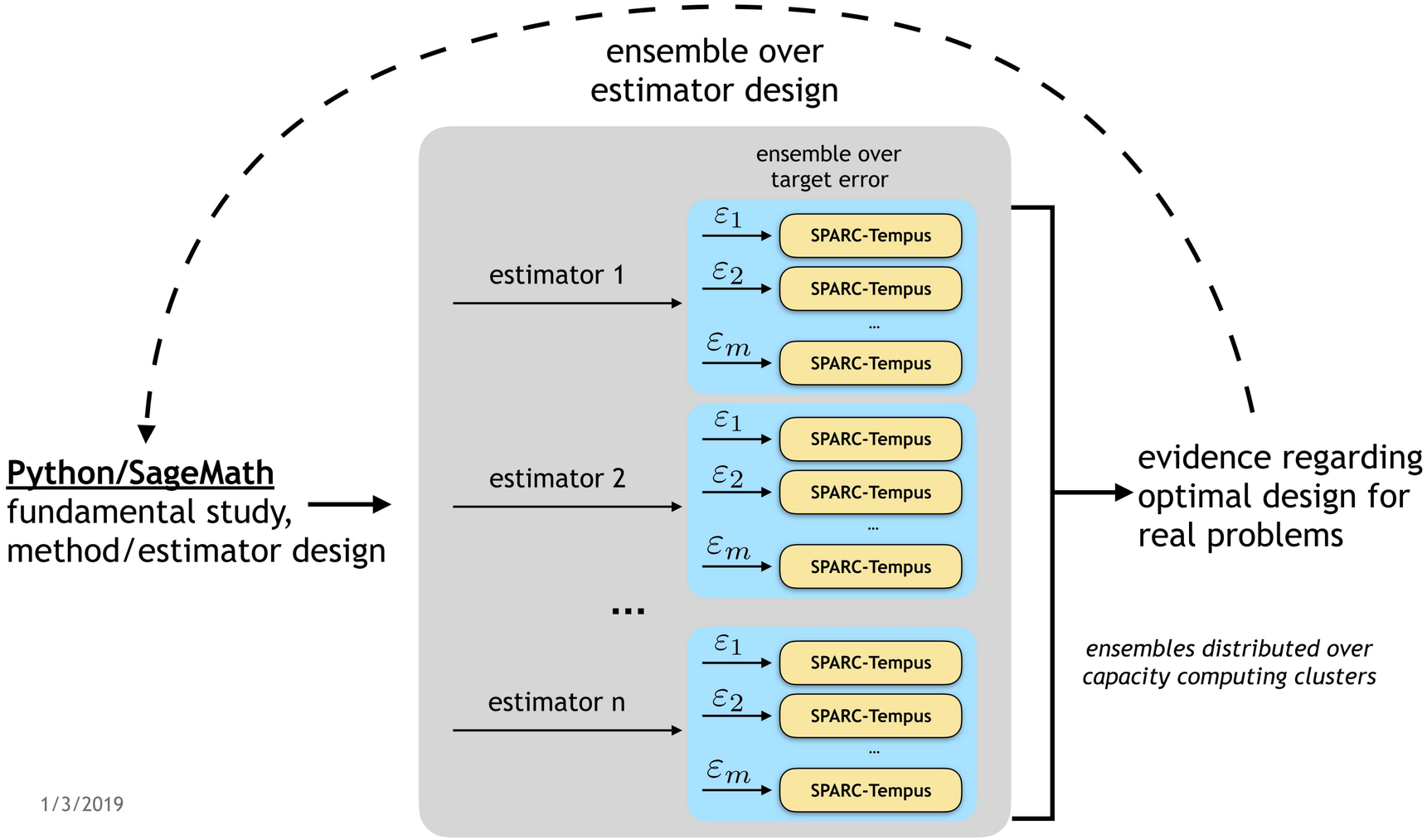


Extensive testing has shown the superiority of relatively unstable embedded estimators

Python/SageMath
fundamental study,
method/estimator design



Extensive testing has shown the superiority of relatively unstable embedded estimators

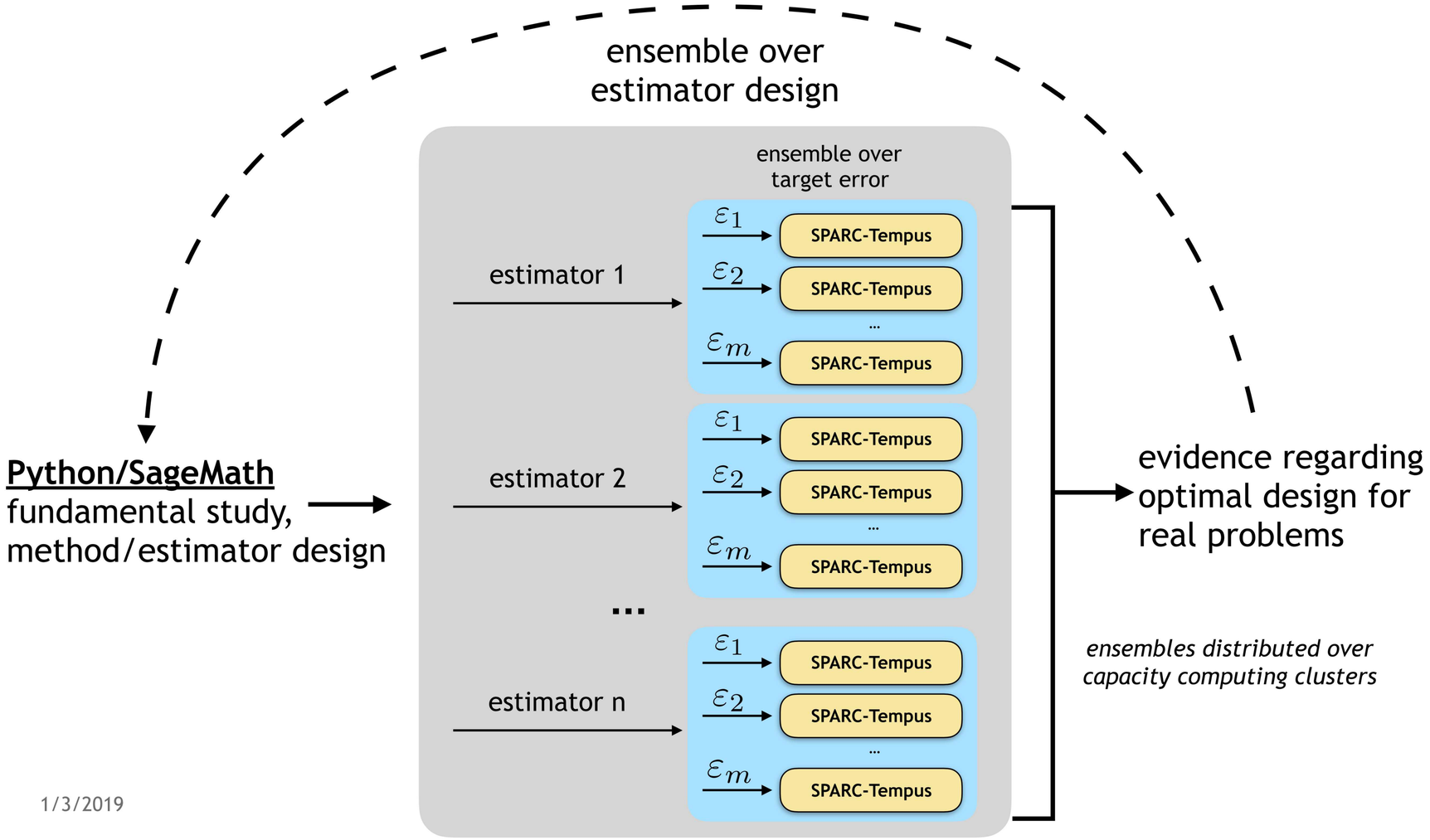


Extensive testing has shown the superiority of relatively unstable embedded estimators

x3 RK solvers (4,5,6 stages)

x3 canonical Navier-Stokes/Euler problems

x2-4 mesh refinements



Extensive testing has shown the superiority of relatively unstable embedded estimators

x3 RK solvers (4,5,6 stages)

x3 canonical Navier-Stokes/Euler problems

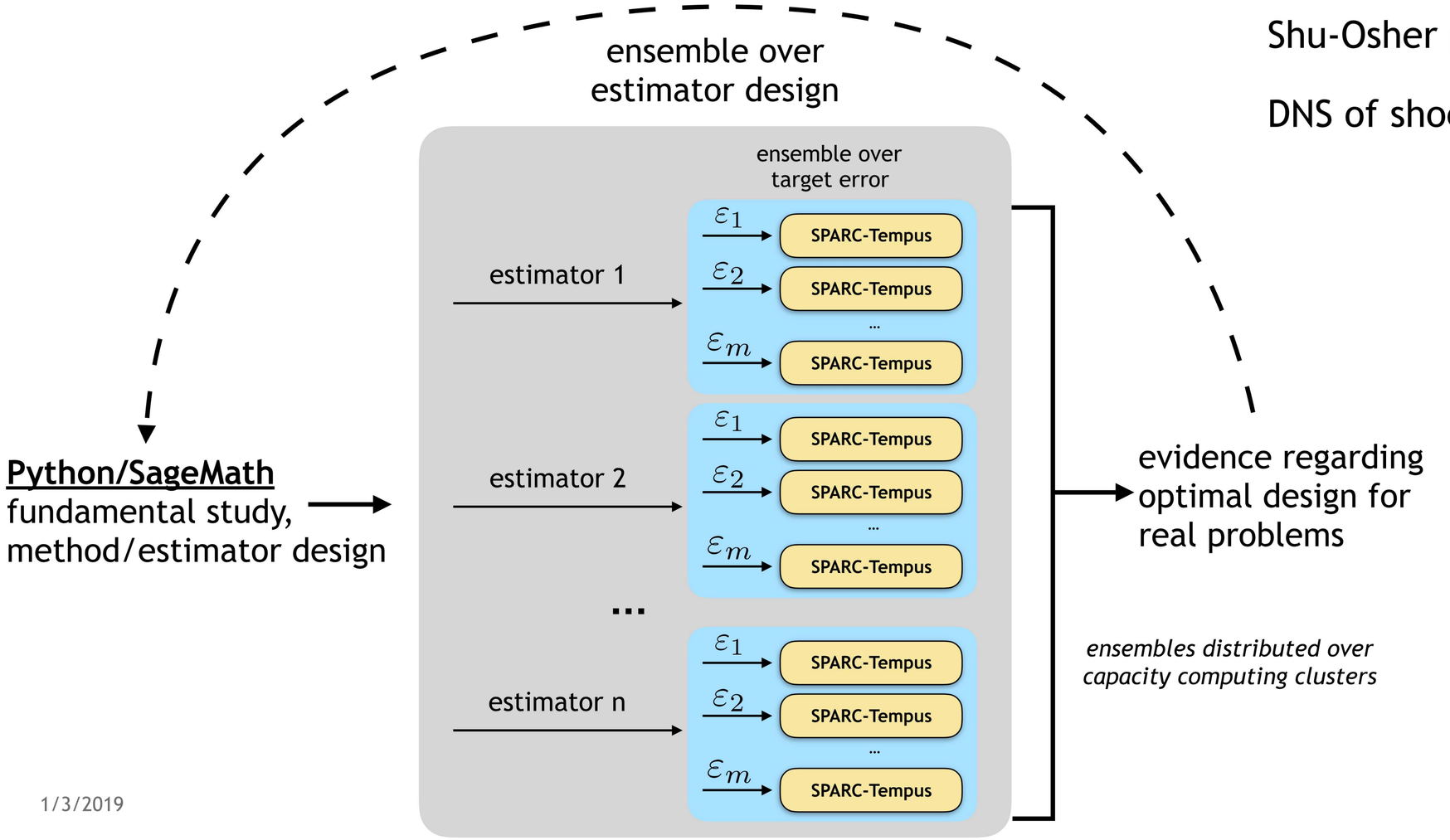
x2-4 mesh refinements

Improved embedded error design already provides improvement over existing workflow

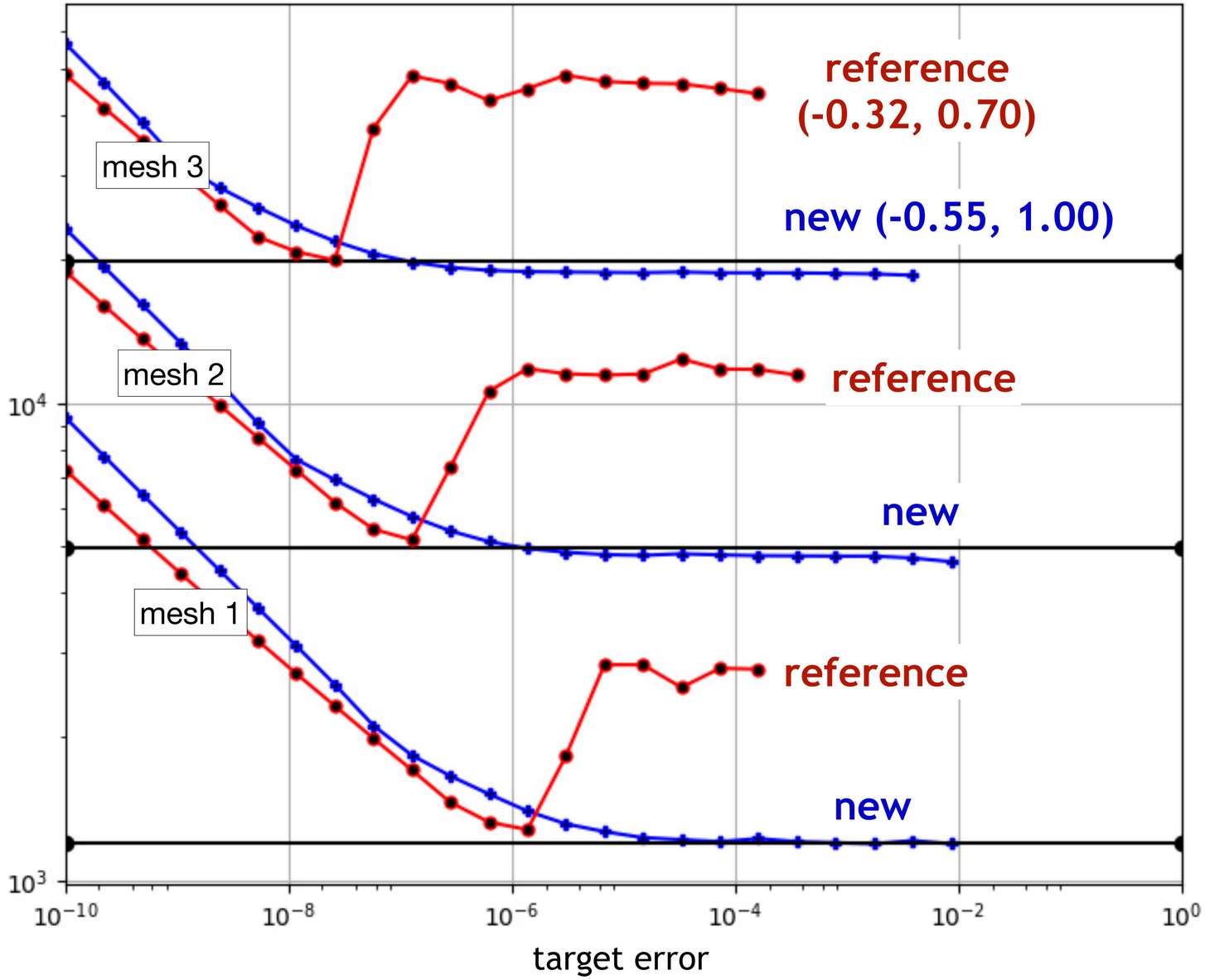
Taylor-Green vortex, 3x speedup

Shu-Osher problem, 6x speedup

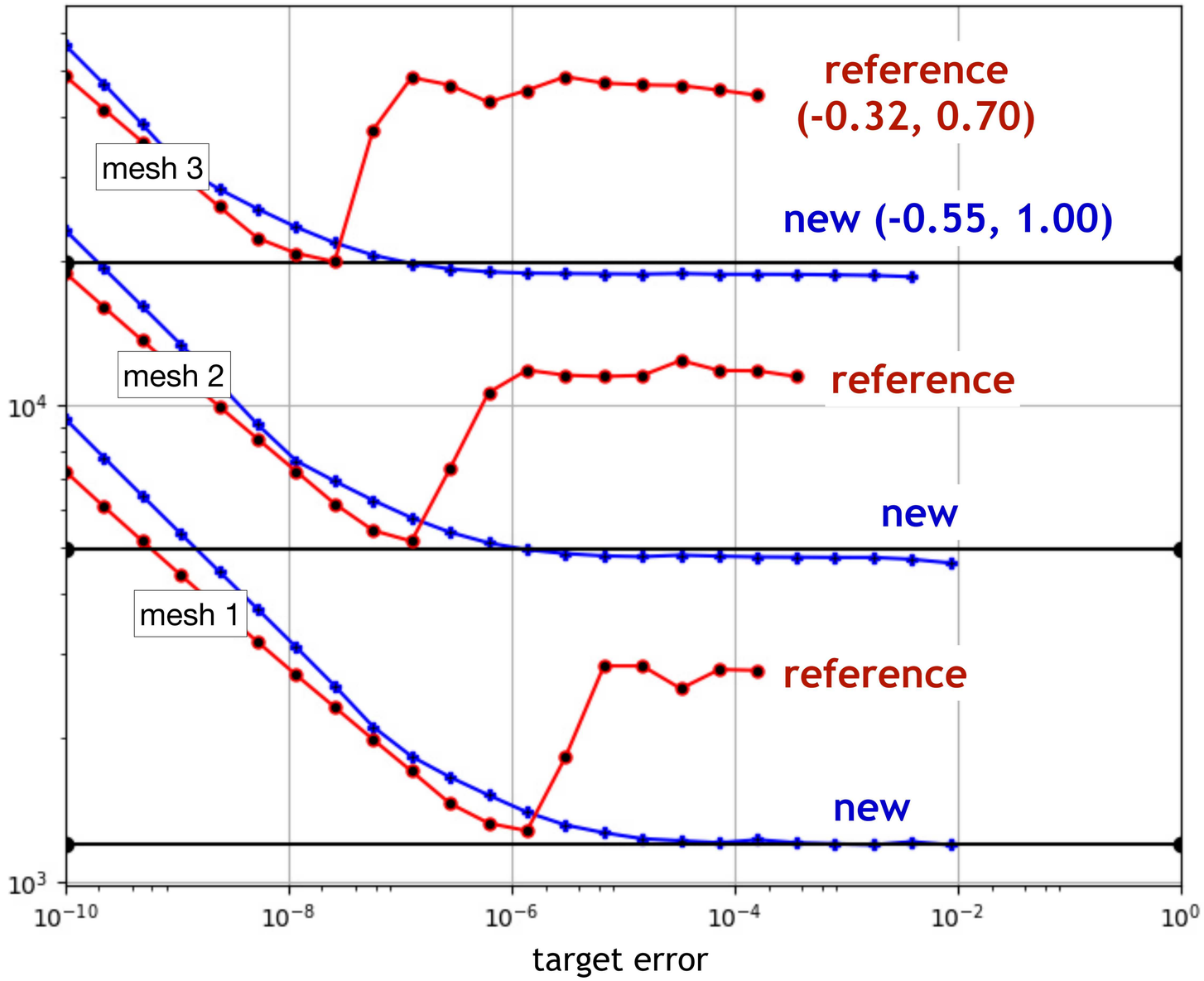
DNS of shock-BL interaction, mixing layer



Our new error estimators provide dramatic improvement over existing methods, but error control in this form still has shortcomings



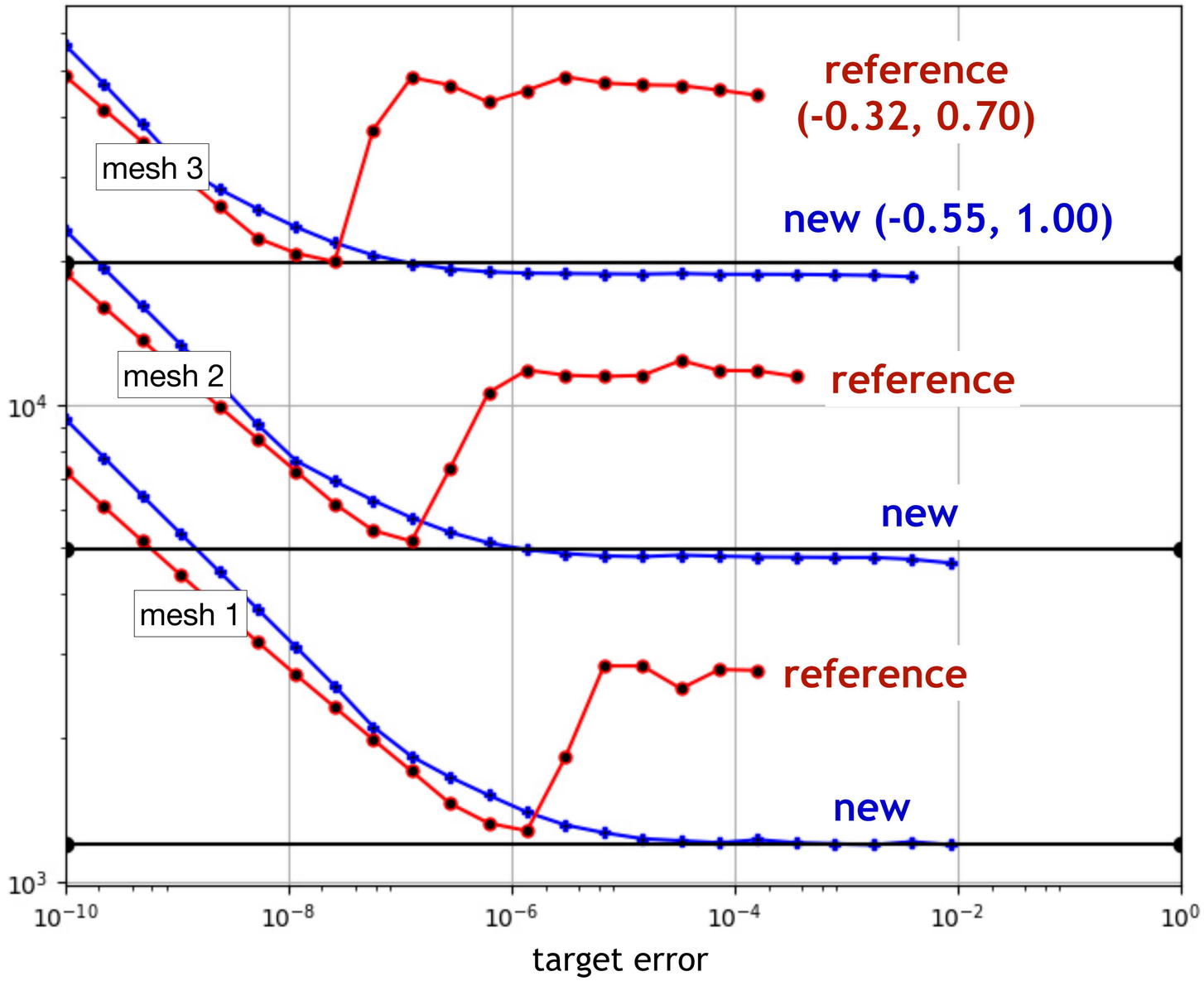
Our new error estimators provide dramatic improvement over existing methods, but error control in this form still has shortcomings



Still guess-and-check.
e.g. 1e-4 may work for one problem/mesh, it may blow up for another.

Still a dependency on mesh resolution

Our new error estimators provide dramatic improvement over existing methods, but error control in this form still has shortcomings



Still guess-and-check.
e.g. 1e-4 may work for one problem/mesh, it may blow up for another.

Still a dependency on mesh resolution

Bigger issue

Constant target error fails or is inefficient with shifts in dynamics

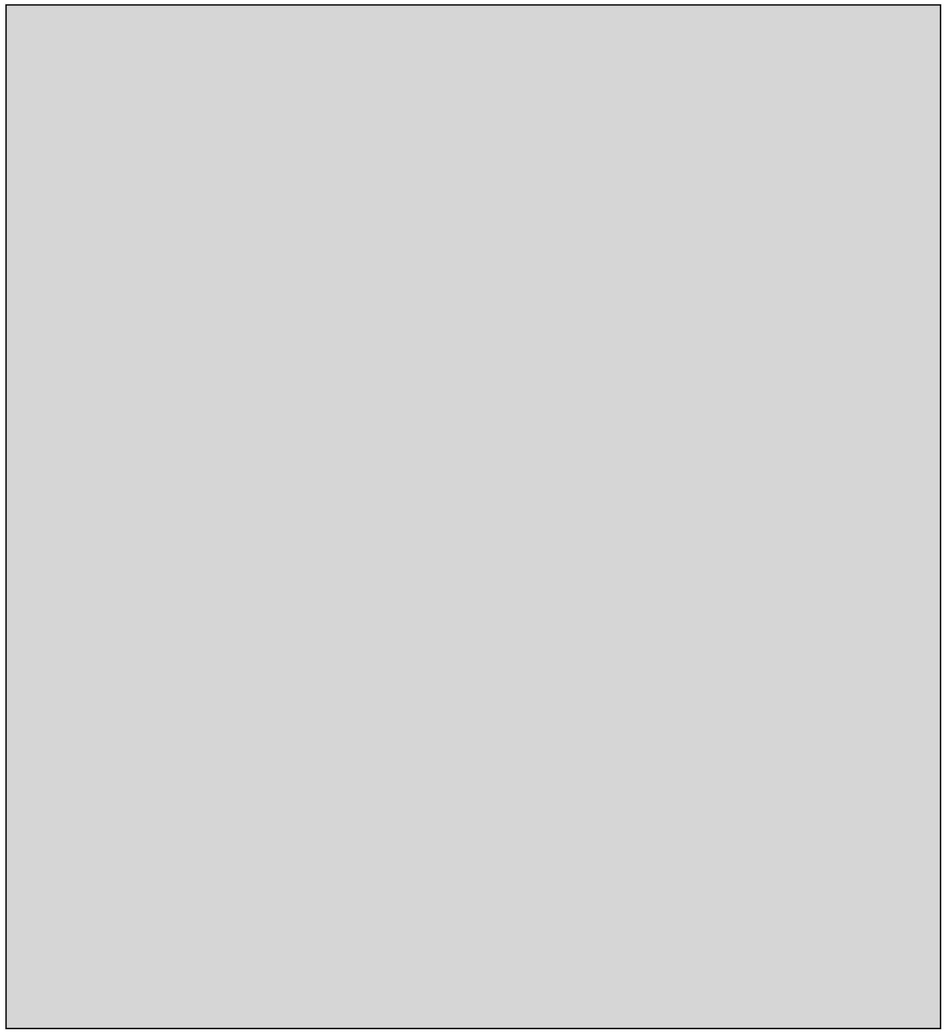
An optimal target error varies in time

Cascade control: predict instability and control the controller!

Design an “instability detector” and add a 2nd controller to manipulate the target error

Standard PID Step Control

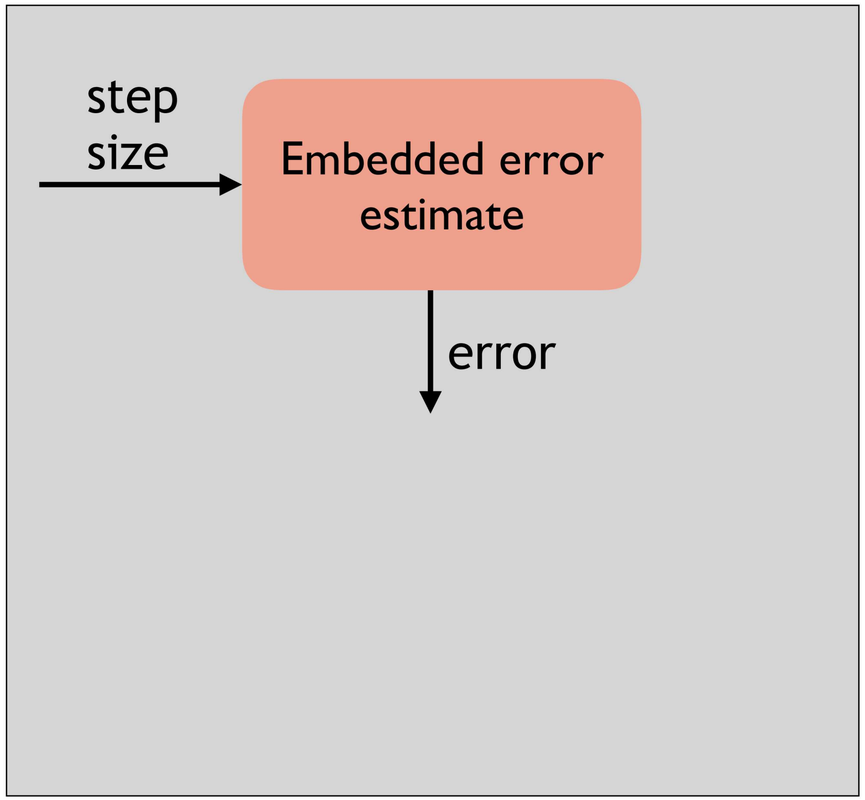
Cascade Control



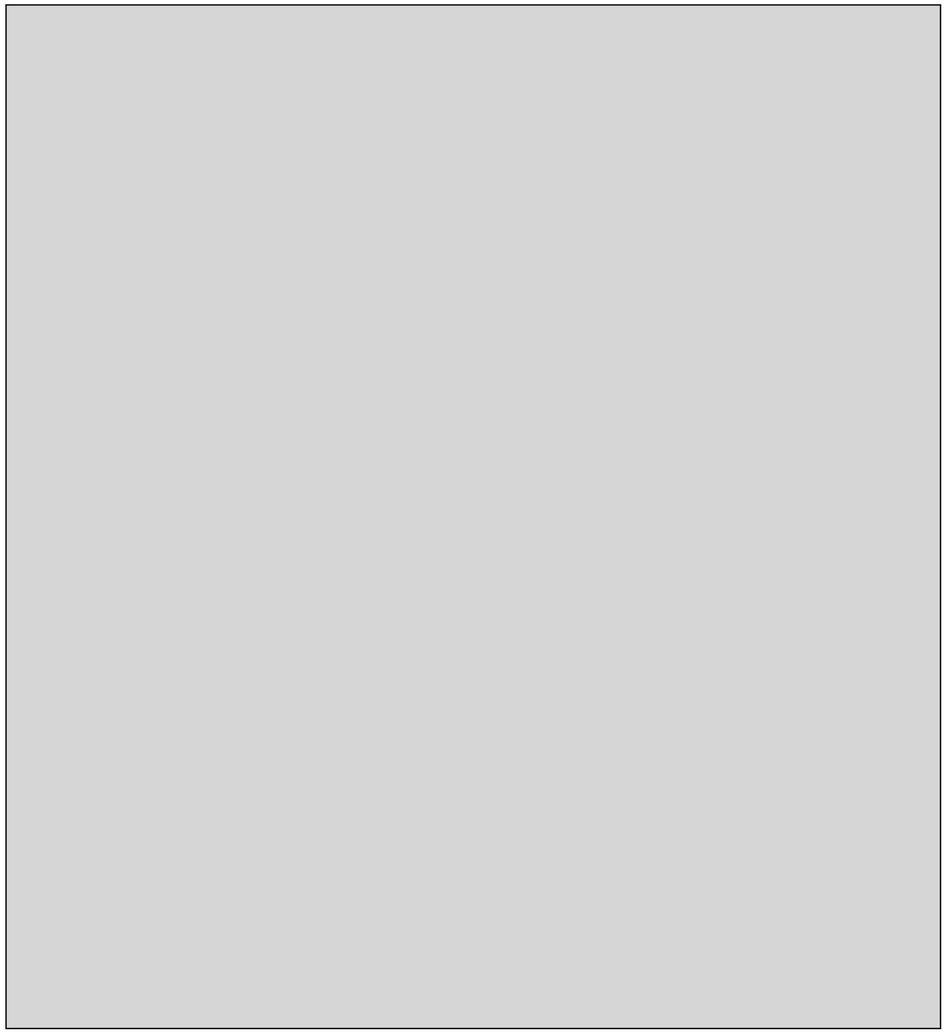
Cascade control: predict instability and control the controller!

Design an “instability detector” and add a 2nd controller to manipulate the target error

Standard PID Step Control



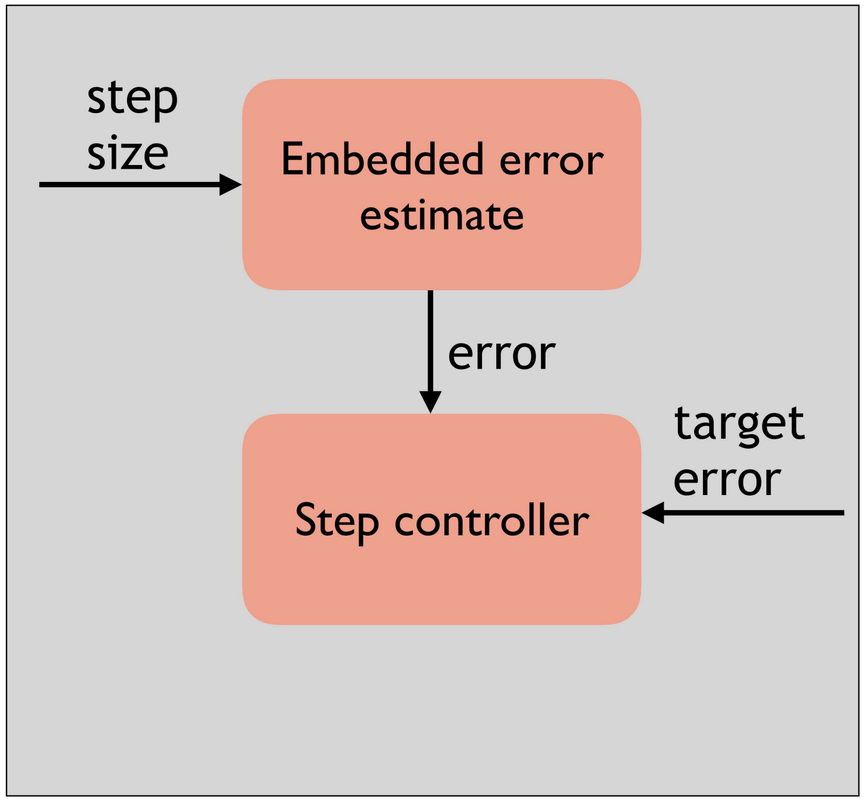
Cascade Control



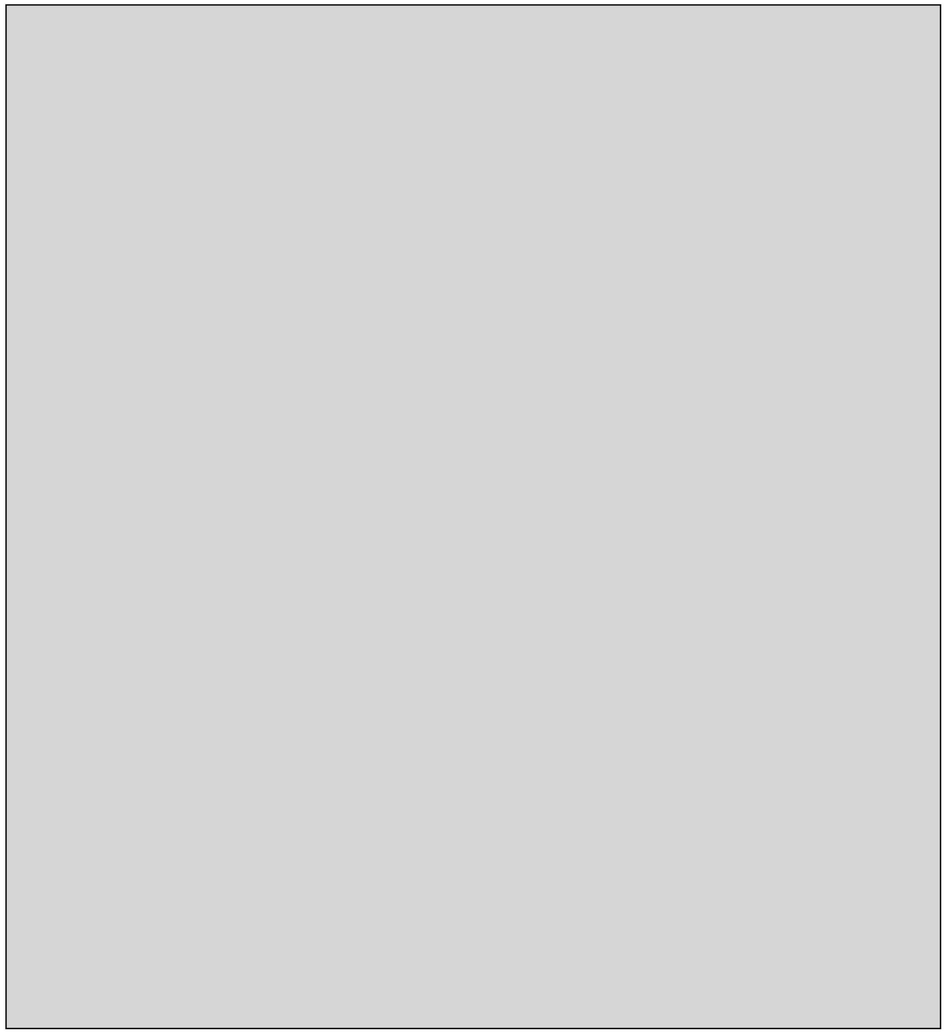
Cascade control: predict instability and control the controller!

Design an “instability detector” and add a 2nd controller to manipulate the target error

Standard PID Step Control



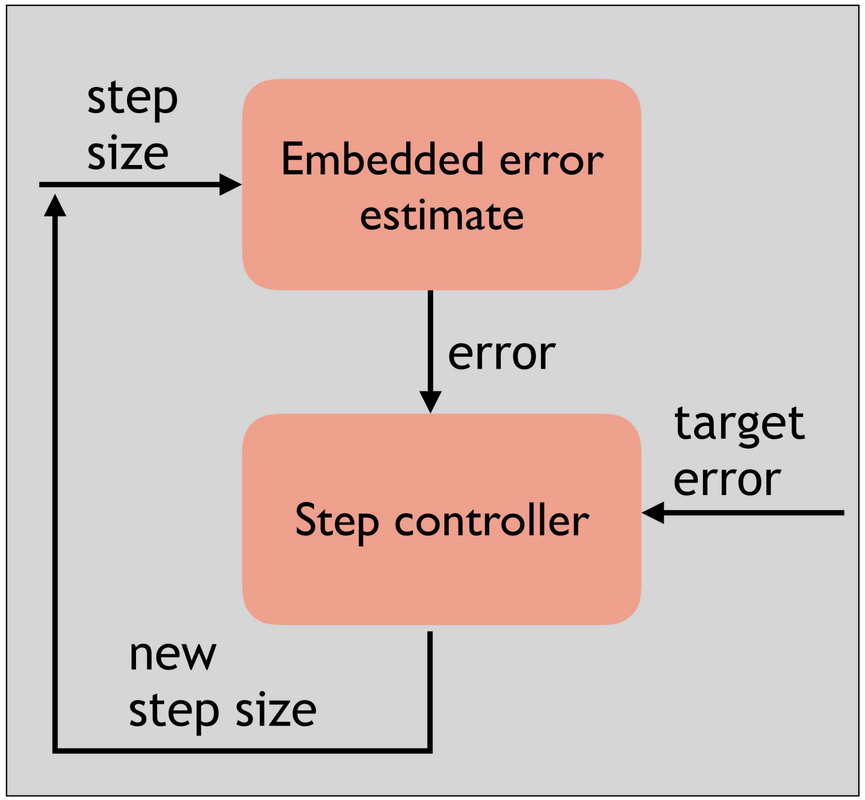
Cascade Control



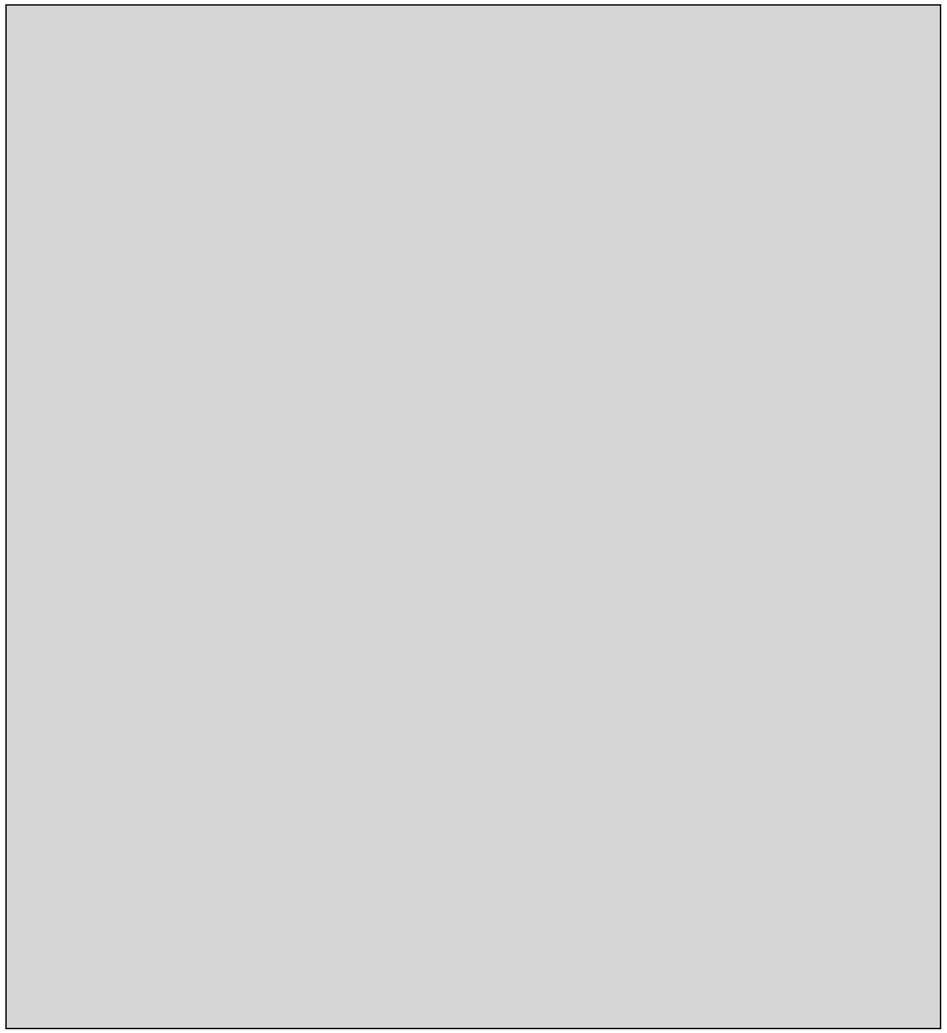
Cascade control: predict instability and control the controller!

Design an “instability detector” and add a 2nd controller
to manipulate the target error

Standard PID Step Control



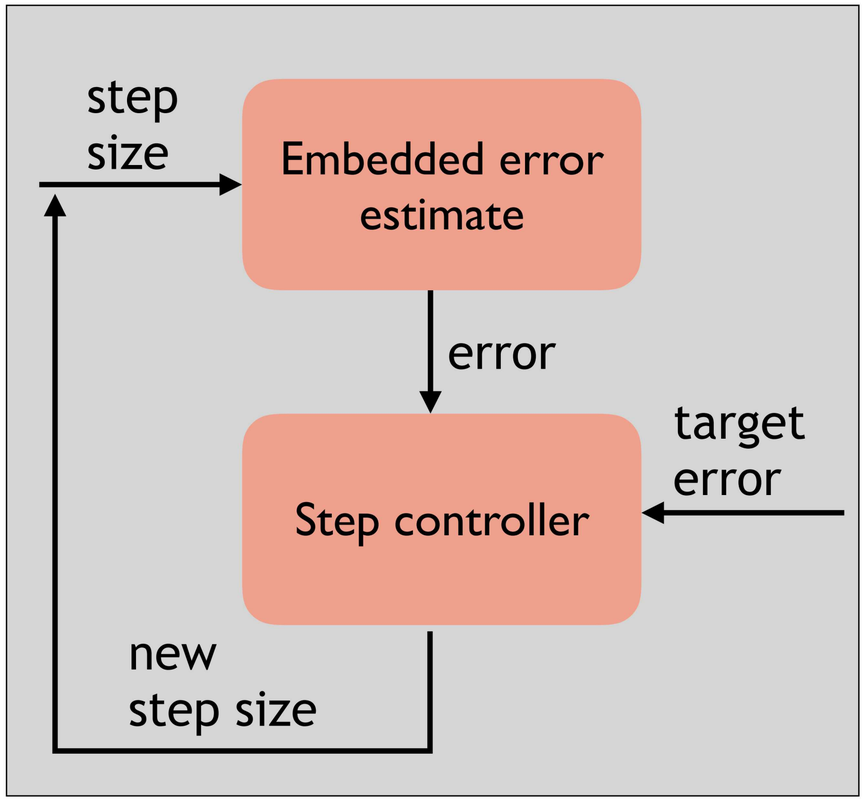
Cascade Control



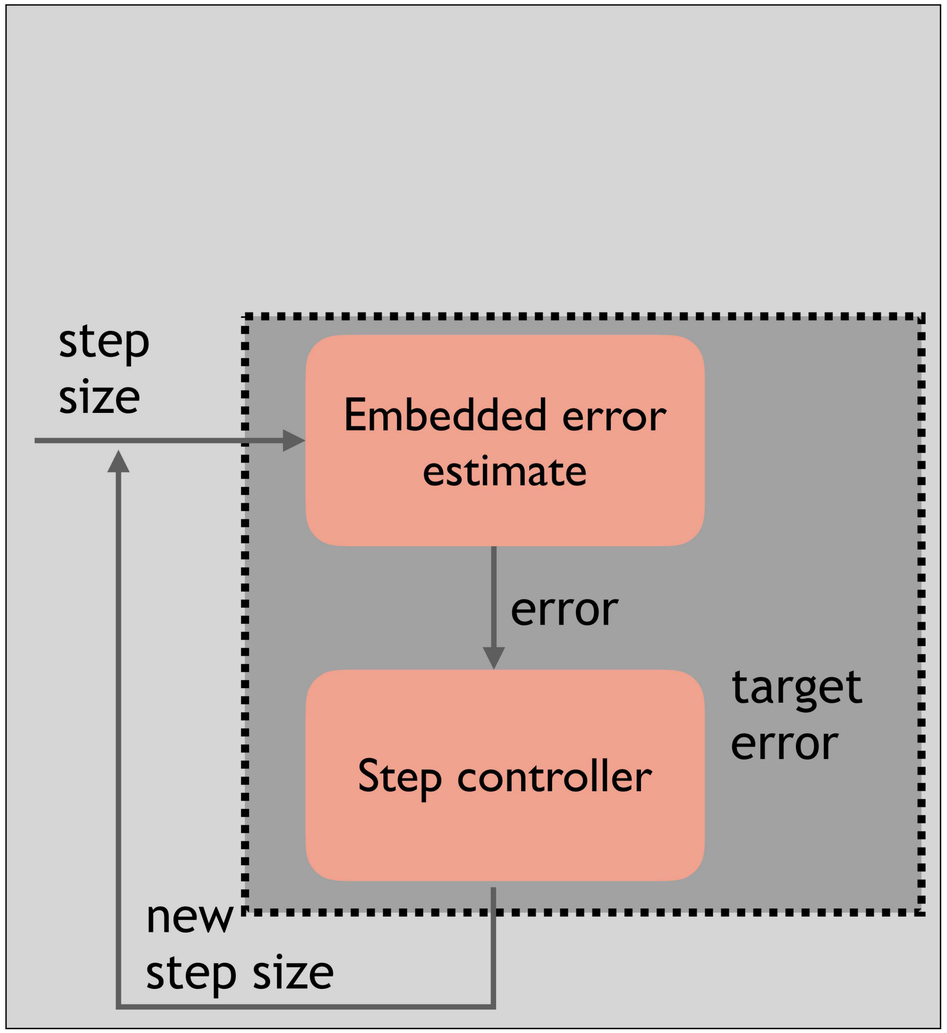
Cascade control: predict instability and control the controller!

Design an “instability detector” and add a 2nd controller
to manipulate the target error

Standard PID Step Control



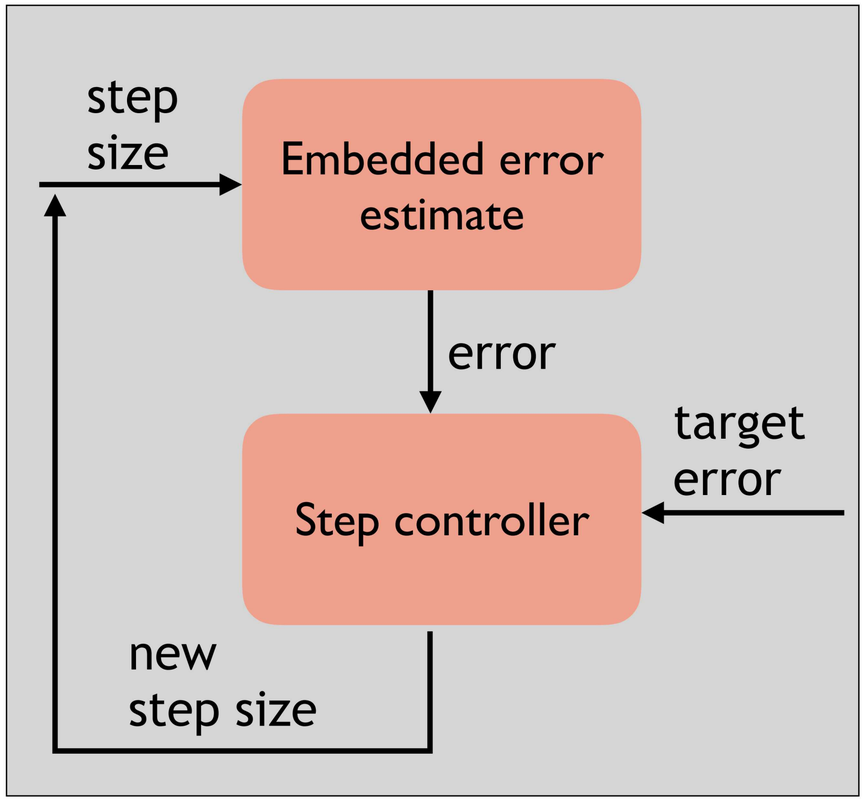
Cascade Control



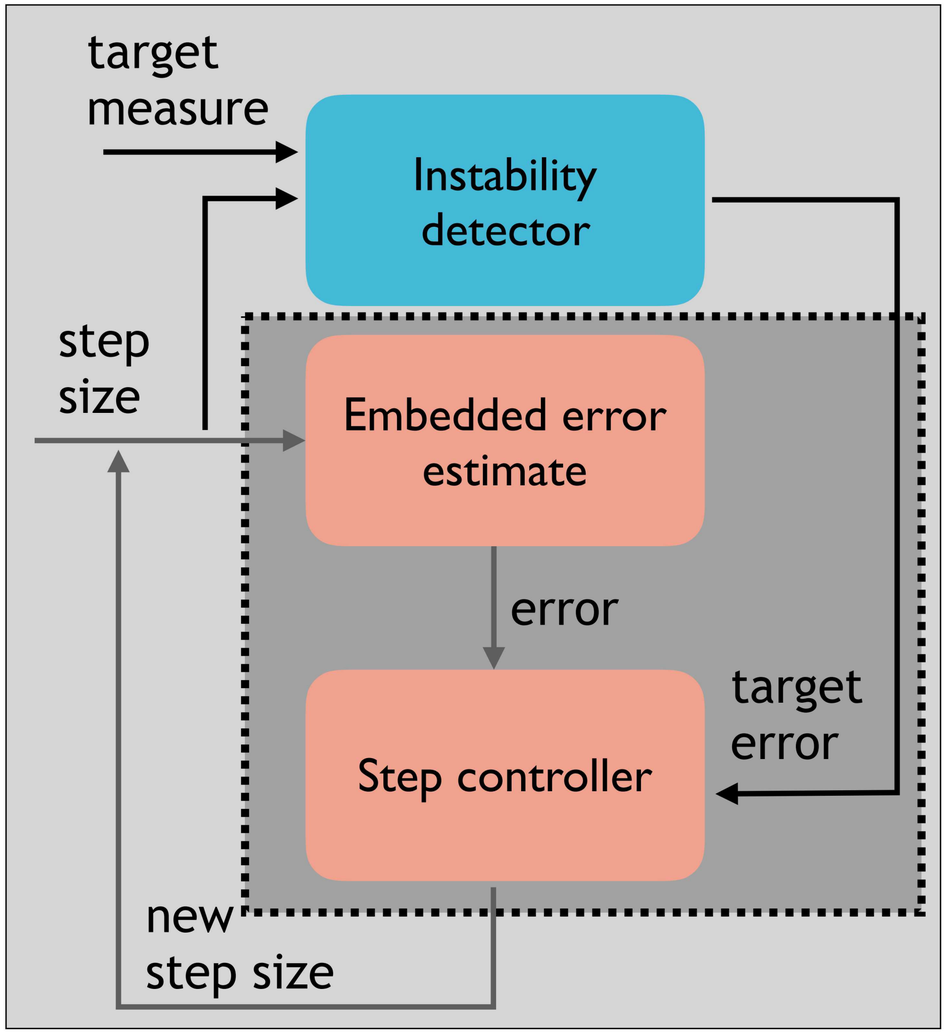
Cascade control: predict instability and control the controller!

Design an “instability detector” and add a 2nd controller
to manipulate the target error

Standard PID Step Control



Cascade Control



Instability measure

1. How do we detect imminent instability?
2. How do we avoid *even more* problem dependence?

Instability measure

1. **How do we detect imminent instability?**
2. **How do we avoid *even more* problem dependence?**

Use multiple estimators of the temporal error!

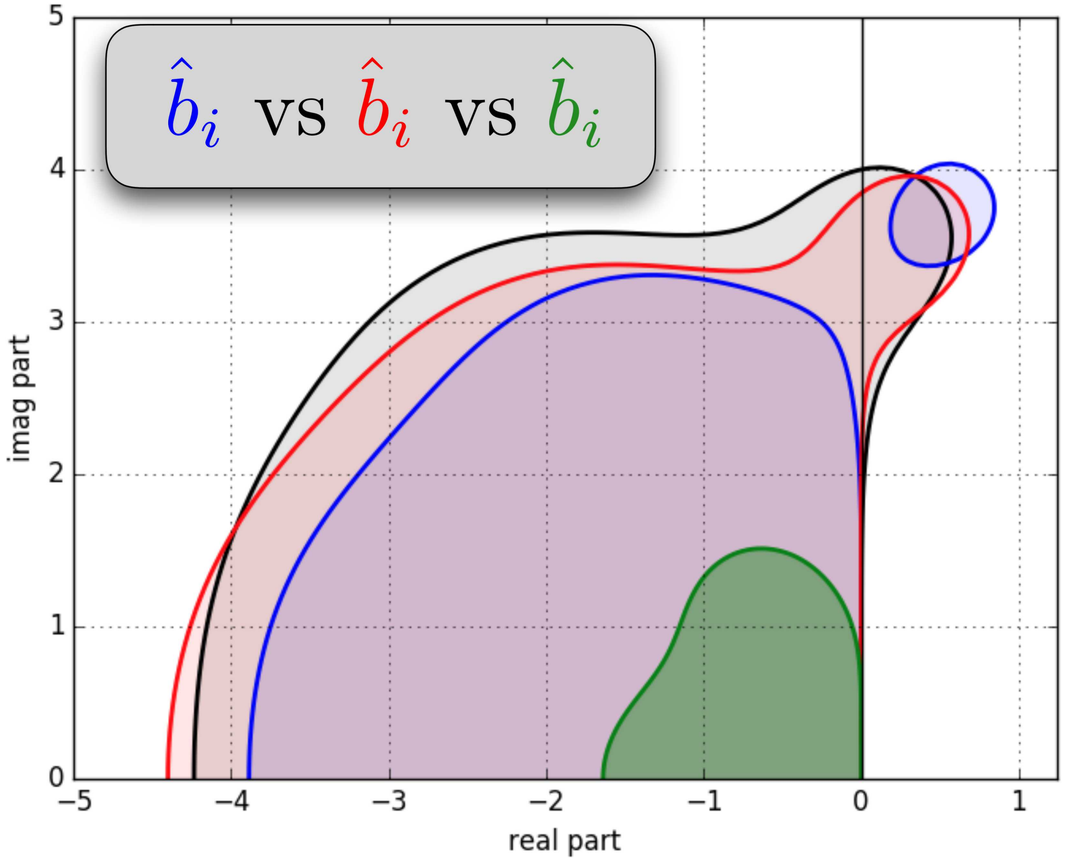
Compared to RHS evaluation (on HPC platforms especially), extra embedded error evaluation is nearly free

Instability measure

1. **How do we detect imminent instability?**
2. **How do we avoid *even more* problem dependence?**

Use multiple estimators of the temporal error!

Compared to RHS evaluation (on HPC platforms especially), extra embedded error evaluation is nearly free



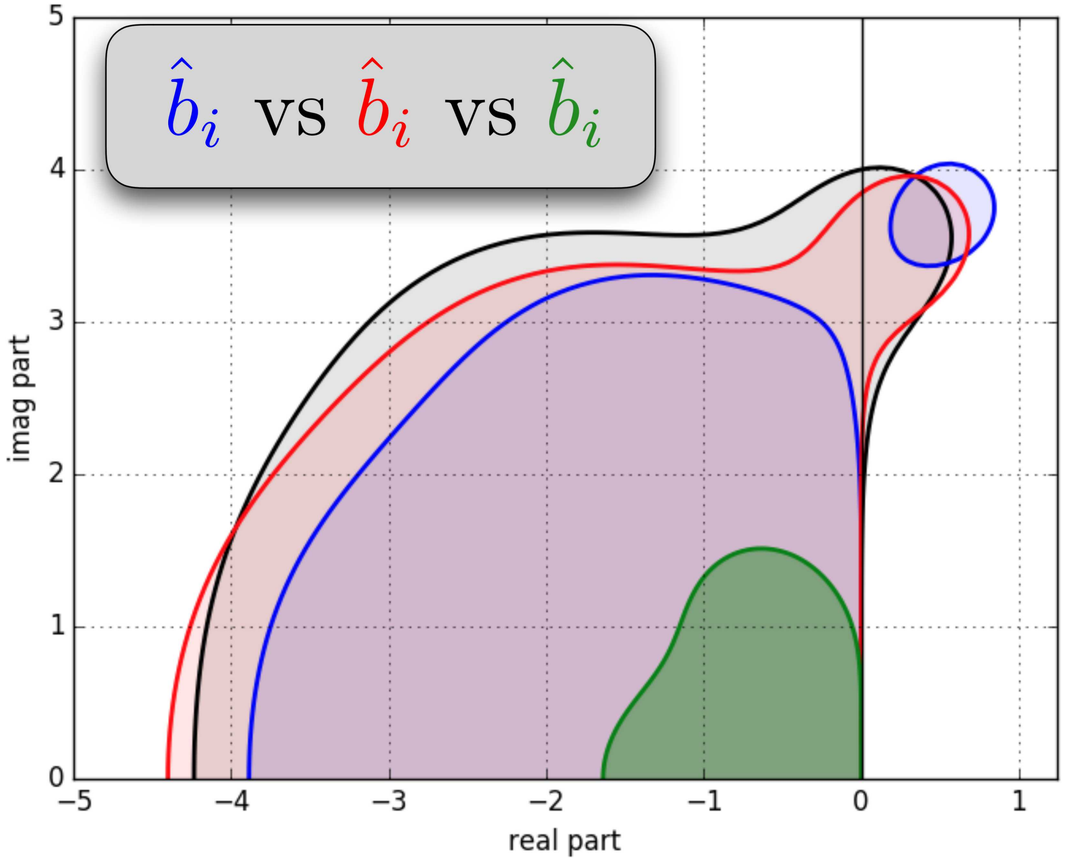
‘Lead’ instability with a second **more** unstable estimator

Instability measure

1. **How do we detect imminent instability?**
2. **How do we avoid *even more* problem dependence?**

Use multiple estimators of the temporal error!

Compared to RHS evaluation (on HPC platforms especially), extra embedded error evaluation is nearly free



‘Lead’ instability with a second **more** unstable estimator

Use the ratio to hopefully eliminate most problem dependence,

$$g = \frac{e_2}{e_1 + \epsilon}$$

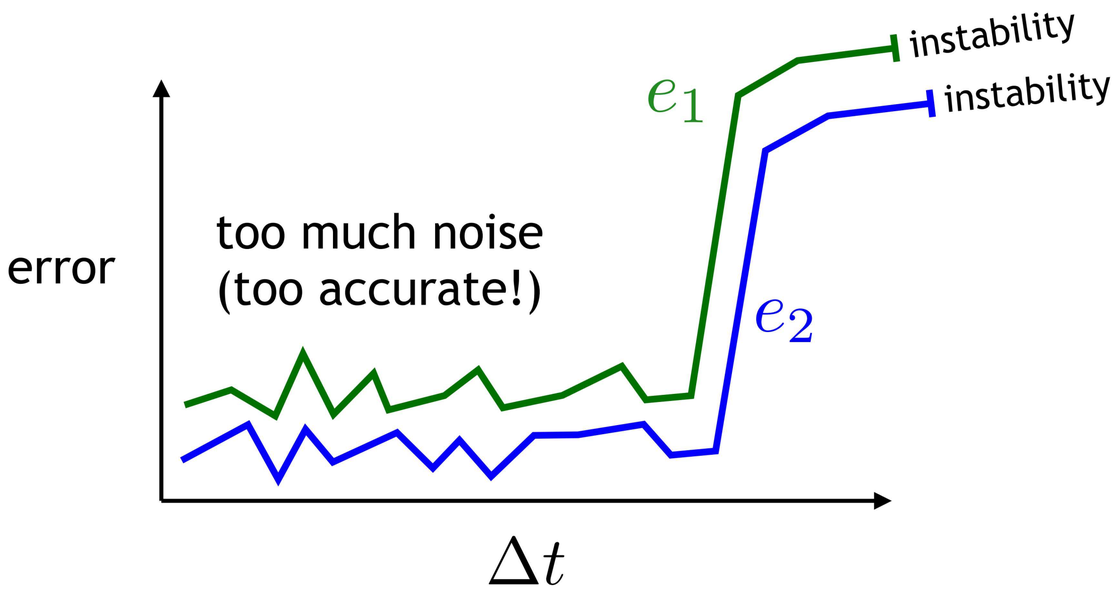
e.g. dimensionless vs dimensional

and use an L∞ norm to avoid mesh dependence

$$g = \frac{e_2}{e_1 + \epsilon}$$

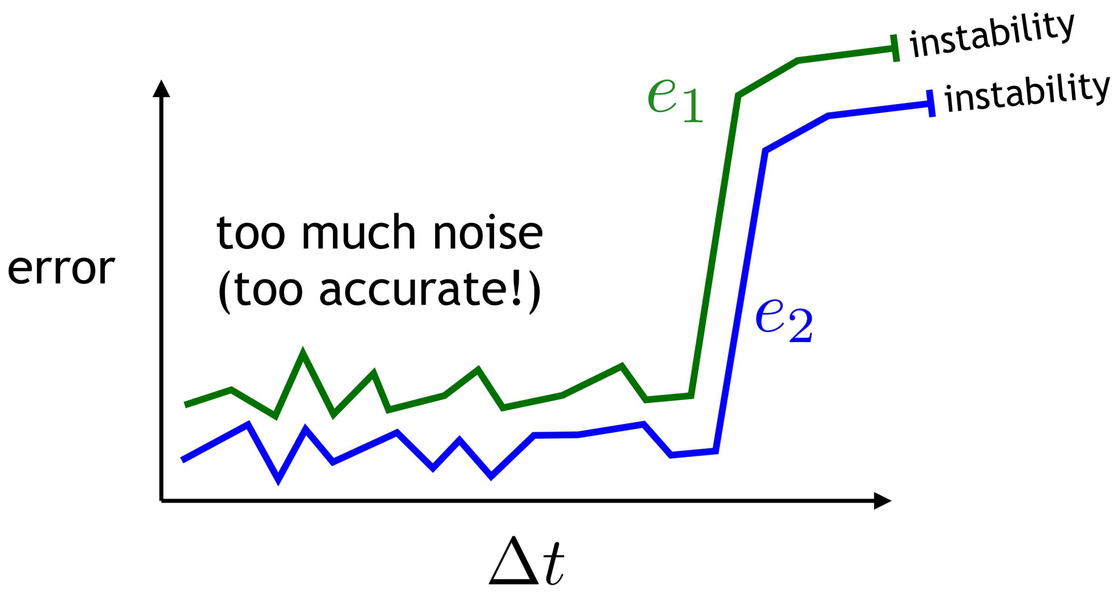
$$g = \frac{e_2}{e_1 + \epsilon}$$

Two third-order embedded estimators



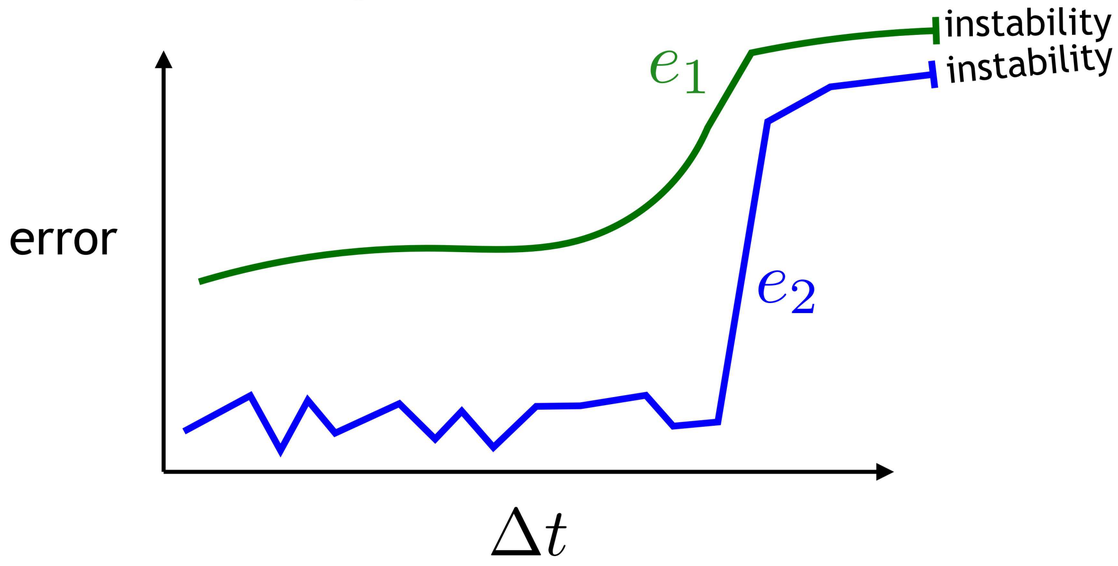
$$g = \frac{e_2}{e_1 + \epsilon}$$

Two third-order embedded estimators



3rd- vs 1st-order embedded estimators

smoother growth -> controllable!

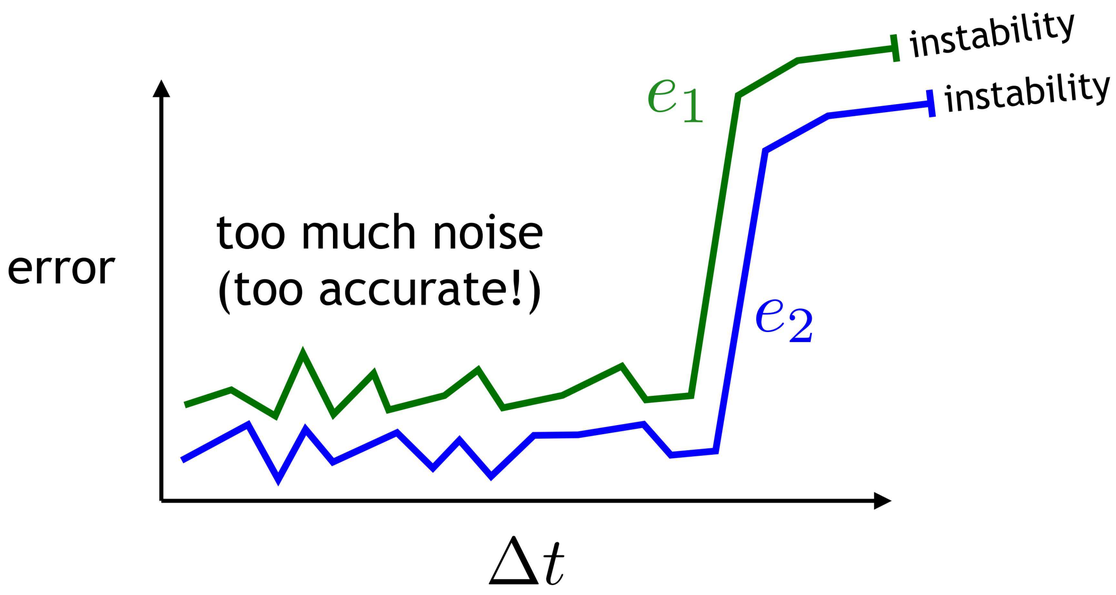


For the instability measure to be controllable, we need to use a lower-order second estimator

ratio is monotonic and one-to-one

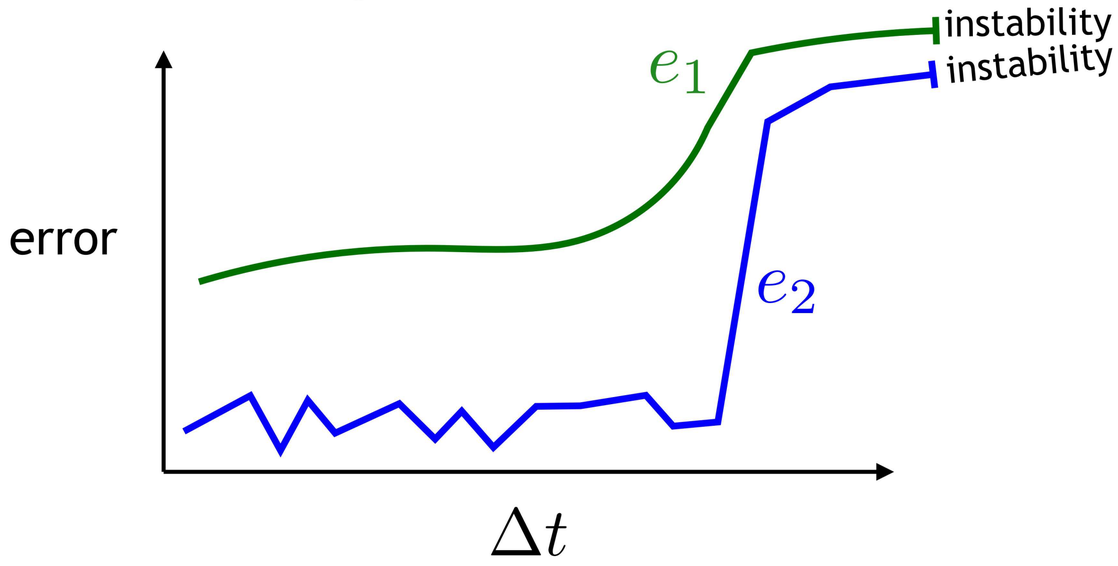
$$g = \frac{e_2}{e_1 + \epsilon}$$

Two third-order embedded estimators



3rd- vs 1st-order embedded estimators

smoother growth -> controllable!



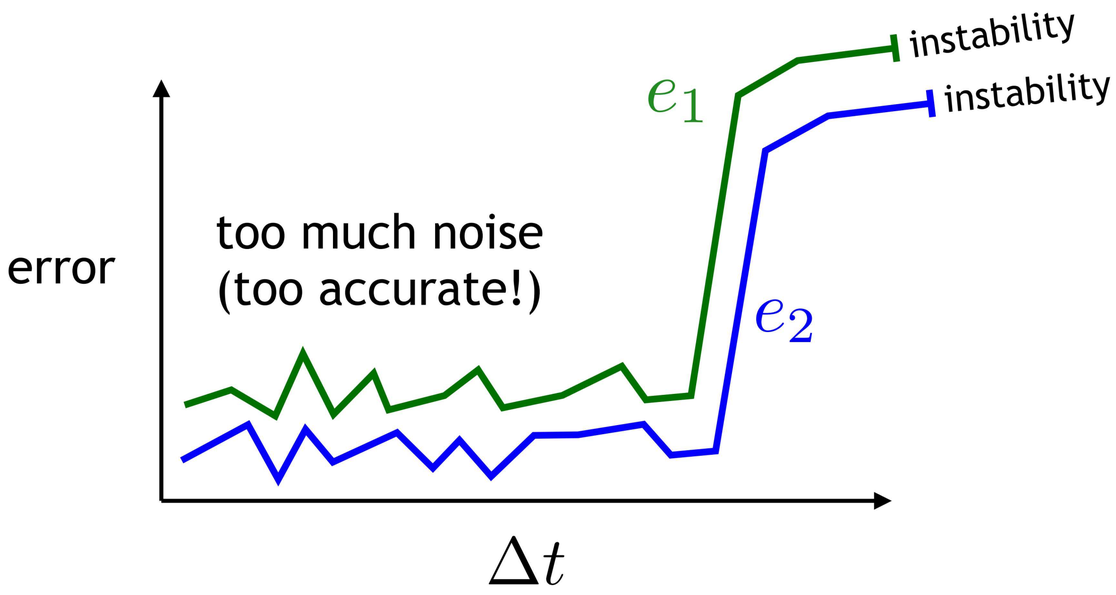
For the instability measure to be controllable, we need to use a lower-order second estimator

ratio is monotonic and one-to-one

$$g = \frac{e_2}{e_1 + \epsilon}$$

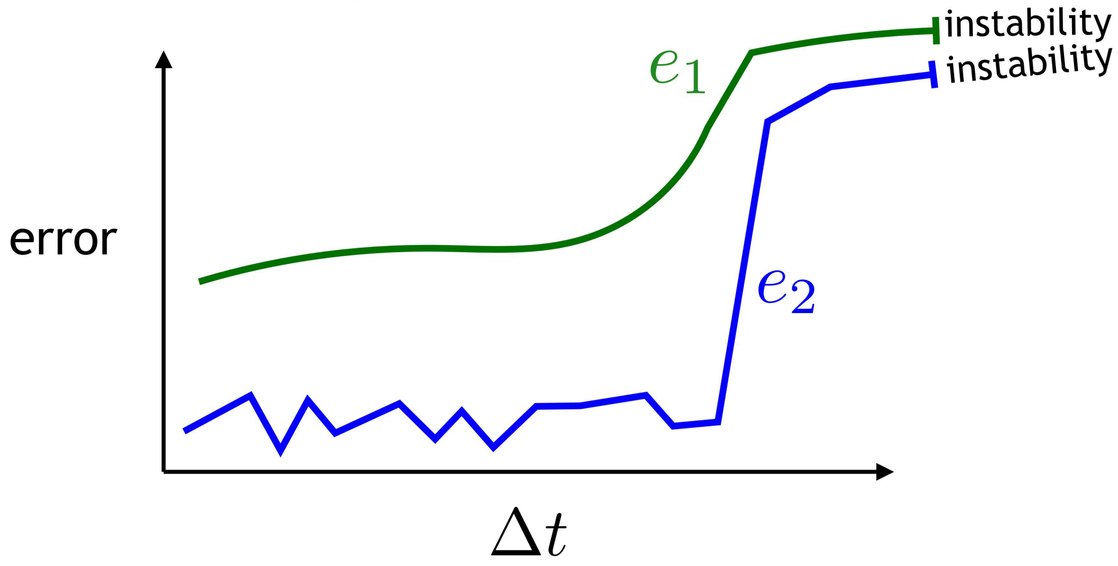
- What first order method to use? Only requires $\sum b_i = 1$
- Our RK method has a node at t^{n+1}, and **Backward Euler** has improved smoothness near instability

Two third-order embedded estimators



3rd- vs 1st-order embedded estimators

smoother growth -> controllable!



The cascade system runs as fast as optimally-tuned standard techniques, with no case-by-case tuning, no resolution dependence

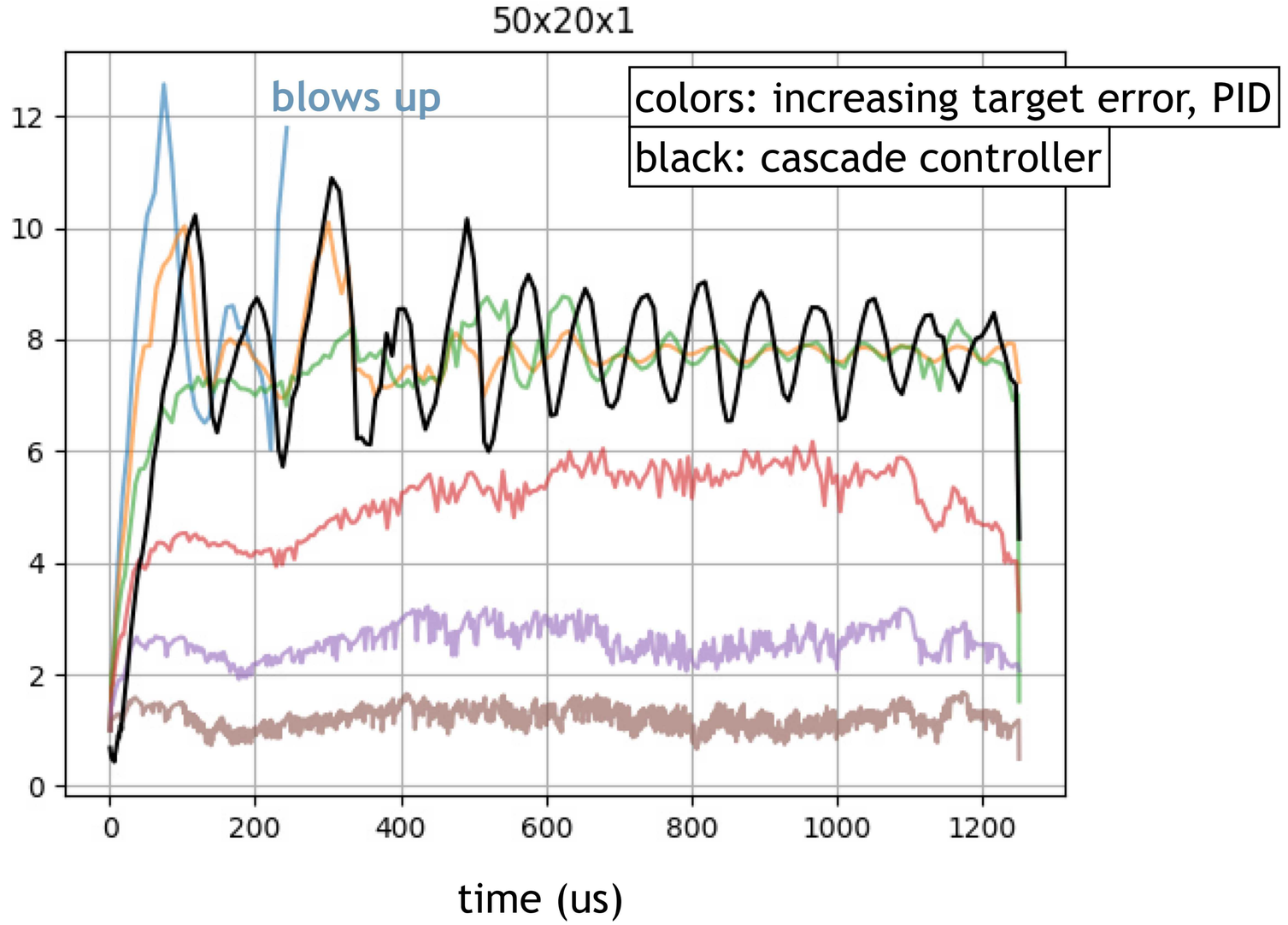
Method: 6-stage, 4th-order ERK subset of the IMEX of Kanevsky et al. (2007)

Canonical "isentropic vortex" problem
2-D inviscid, nonlinear Euler equations

Entropy-stable high-order finite difference formulation

Added oscillation can be reduced by improved controller design

balance responsiveness vs oscillation



The cascade system runs as fast as optimally-tuned standard techniques, with no case-by-case tuning, no resolution dependence

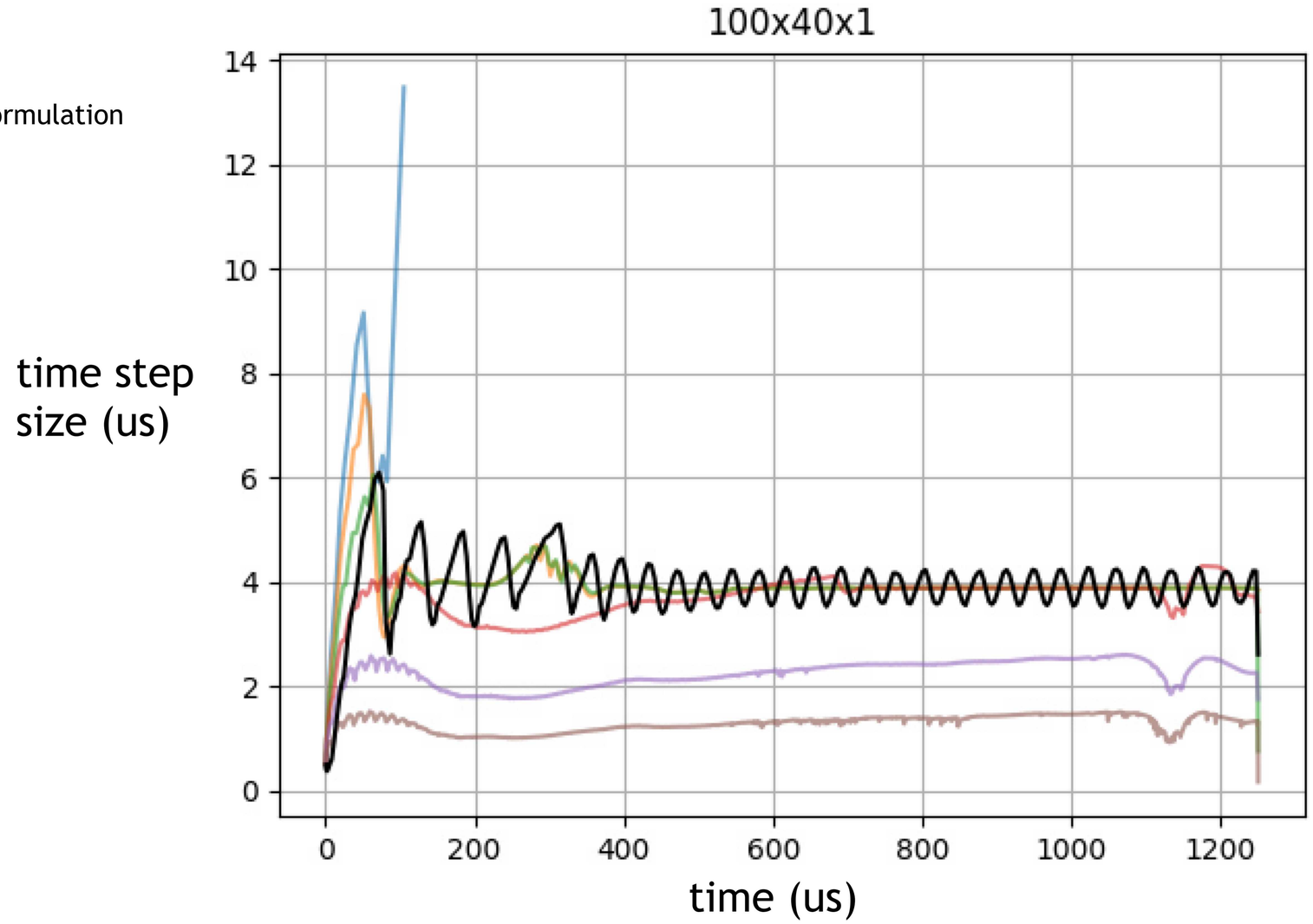
Added oscillation can be reduced by improved controller design

balance responsiveness vs oscillation

Method: 6-stage, 4th-order ERK subset of the IMEX of Kanevsky et al. (2007)

Canonical "isentropic vortex" problem
2-D inviscid, nonlinear Euler equations

Entropy-stable high-order finite difference formulation



The cascade system runs as fast as optimally-tuned standard techniques, with no case-by-case tuning, no resolution dependence

Added oscillation can be reduced by improved controller design

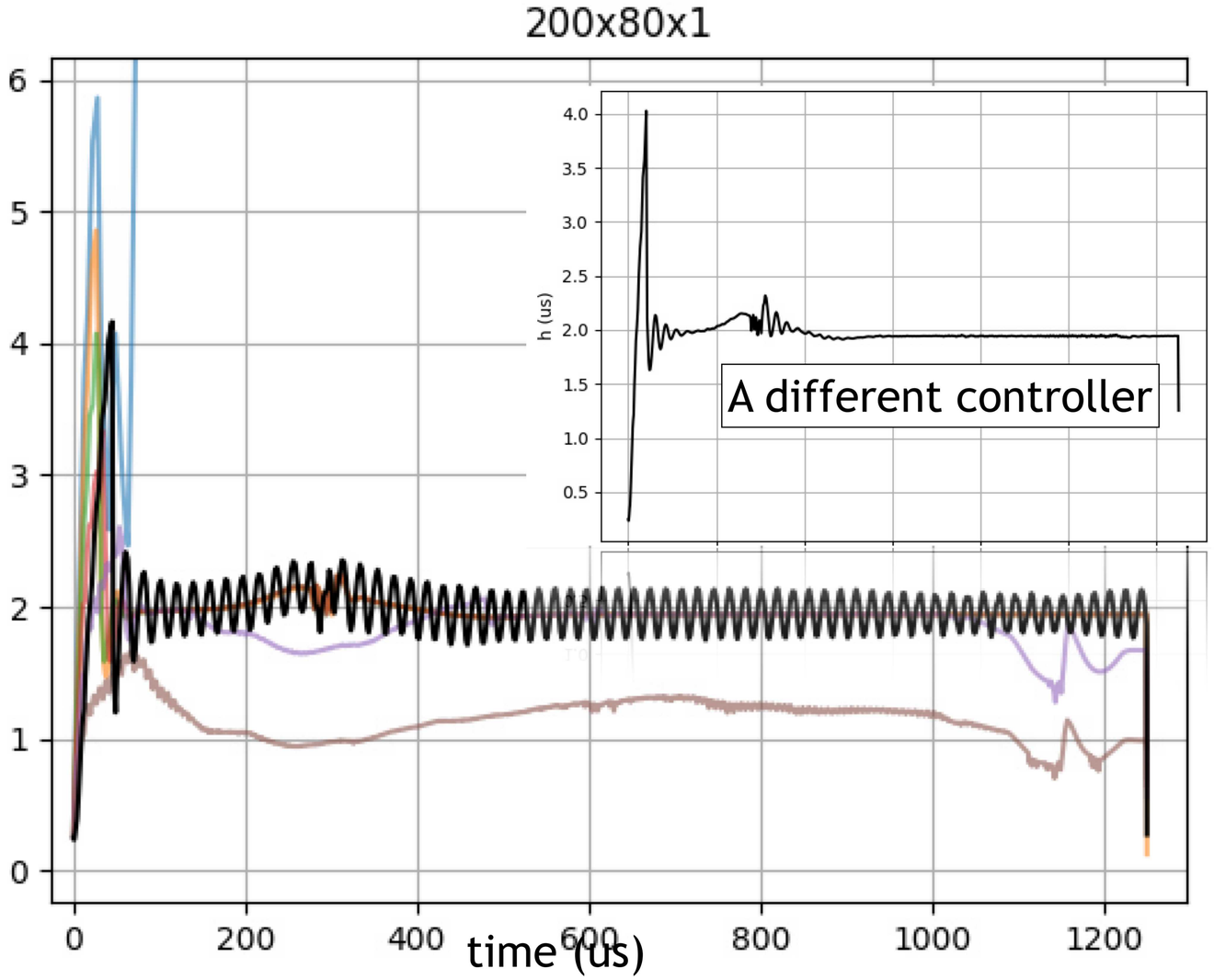
balance responsiveness vs oscillation

Method: 6-stage, 4th-order ERK subset of the IMEX of Kanevsky et al. (2007)

Canonical "isentropic vortex" problem
2-D inviscid, nonlinear Euler equations

Entropy-stable high-order finite difference formulation

time step size (us)



The cascade system runs as fast as optimally-tuned standard techniques, with no case-by-case tuning, no resolution dependence

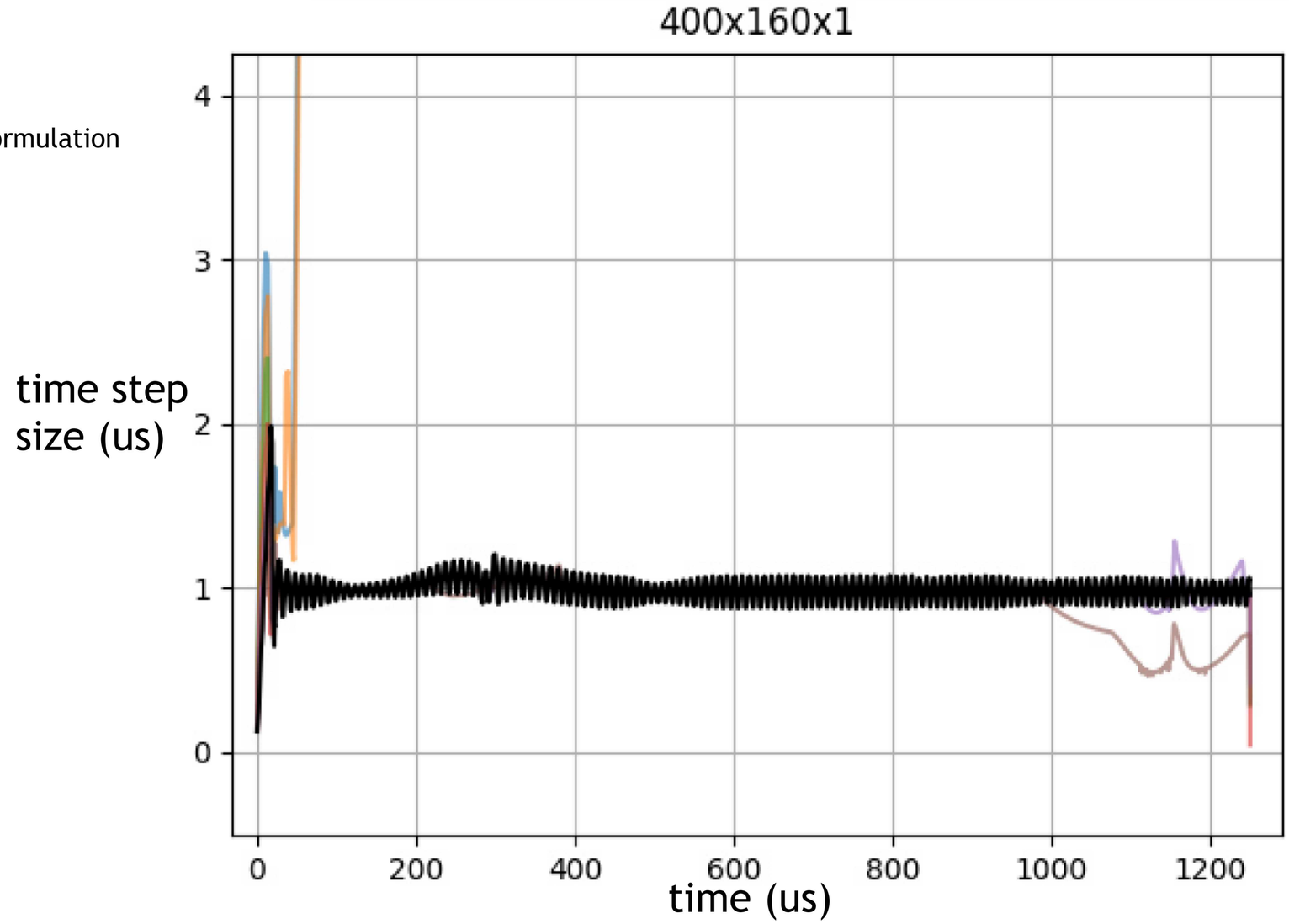
Method: 6-stage, 4th-order ERK subset of the IMEX of Kanevsky et al. (2007)

Canonical "isentropic vortex" problem
2-D inviscid, nonlinear Euler equations

Entropy-stable high-order finite difference formulation

Added oscillation can be reduced by improved controller design

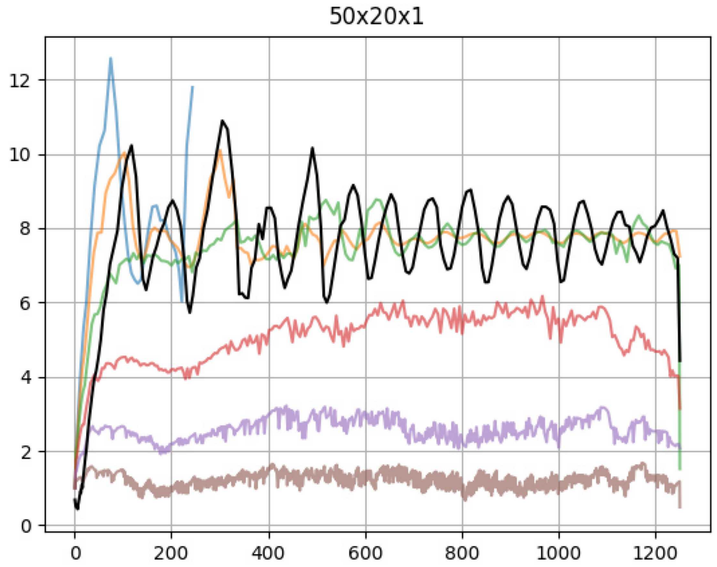
balance responsiveness vs oscillation

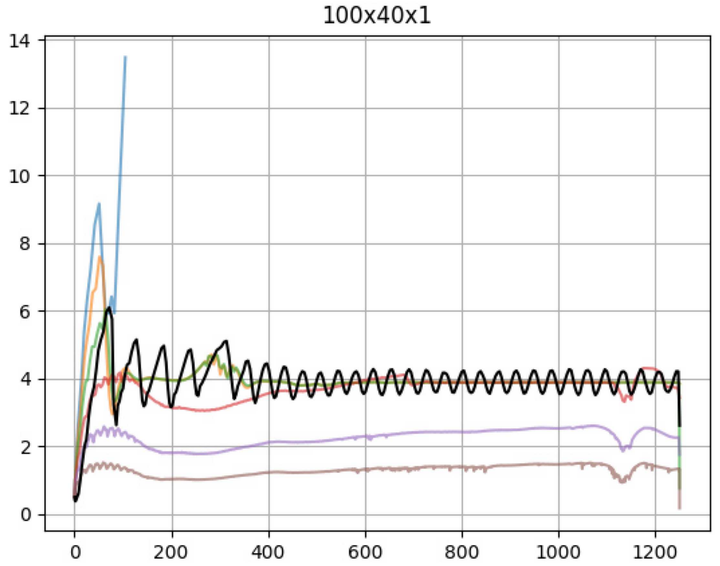


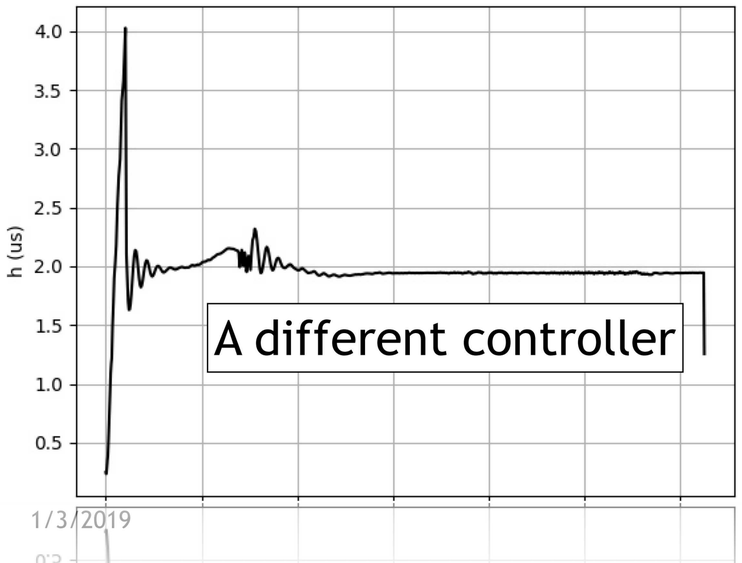
The cascade system runs as fast as optimally-tuned standard techniques, with no case-by-case tuning

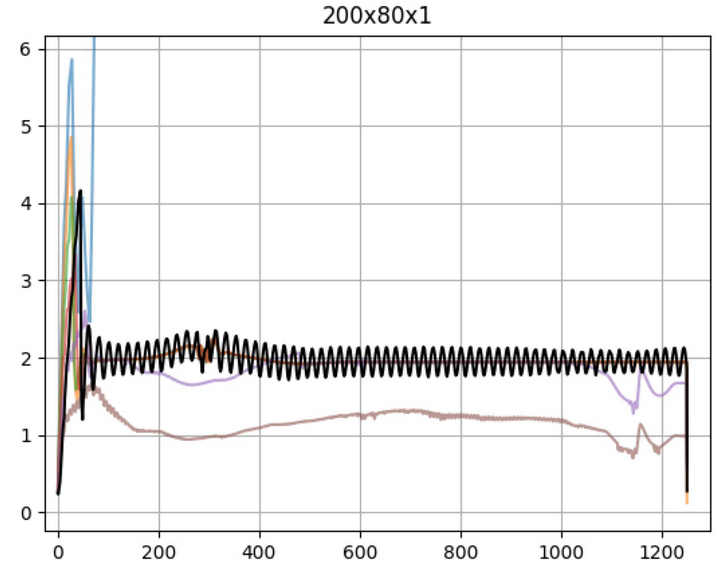
Added oscillation can be reduced by improved controller design

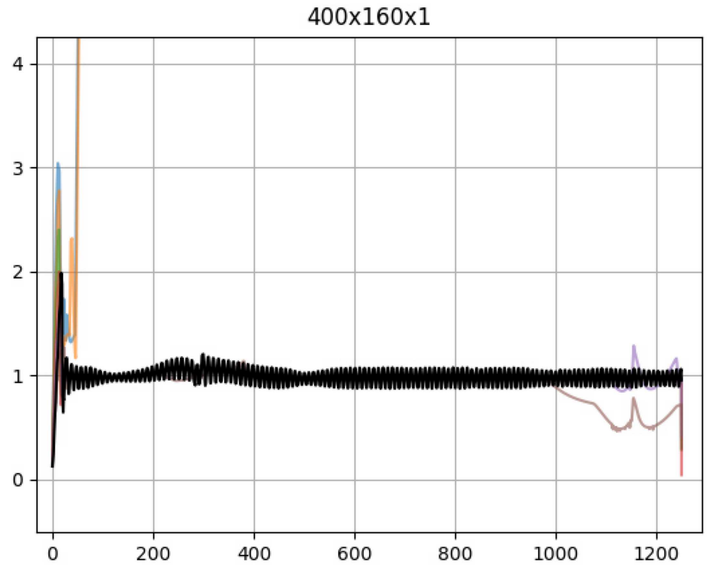
balance responsiveness vs oscillation







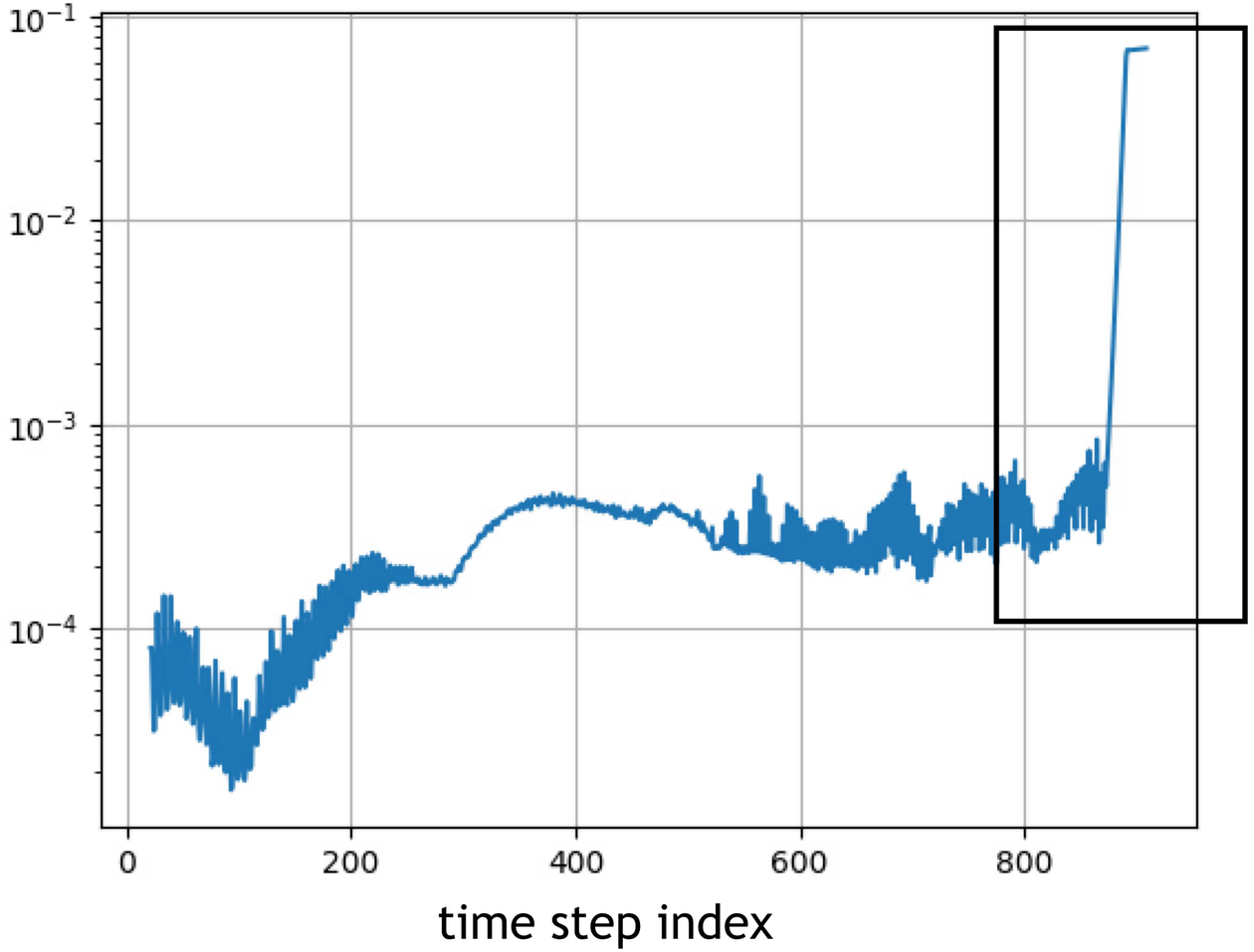




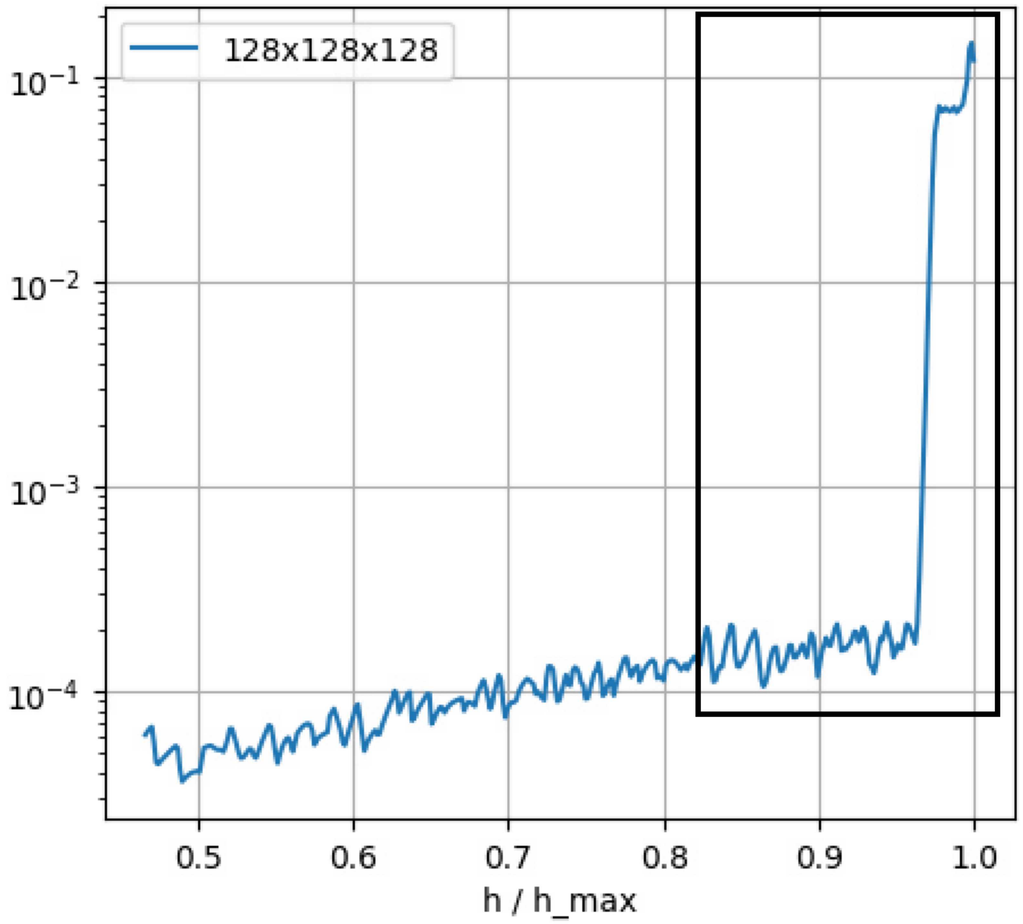
Empirical ramp tests showcase why cascade control works

a priori testing wherein the time step is slowly ramped up until instability occurs

2-D isentropic vortex
Finest mesh (400x160x1)



3-D inviscid Taylor-Green vortex
128x128x128 mesh

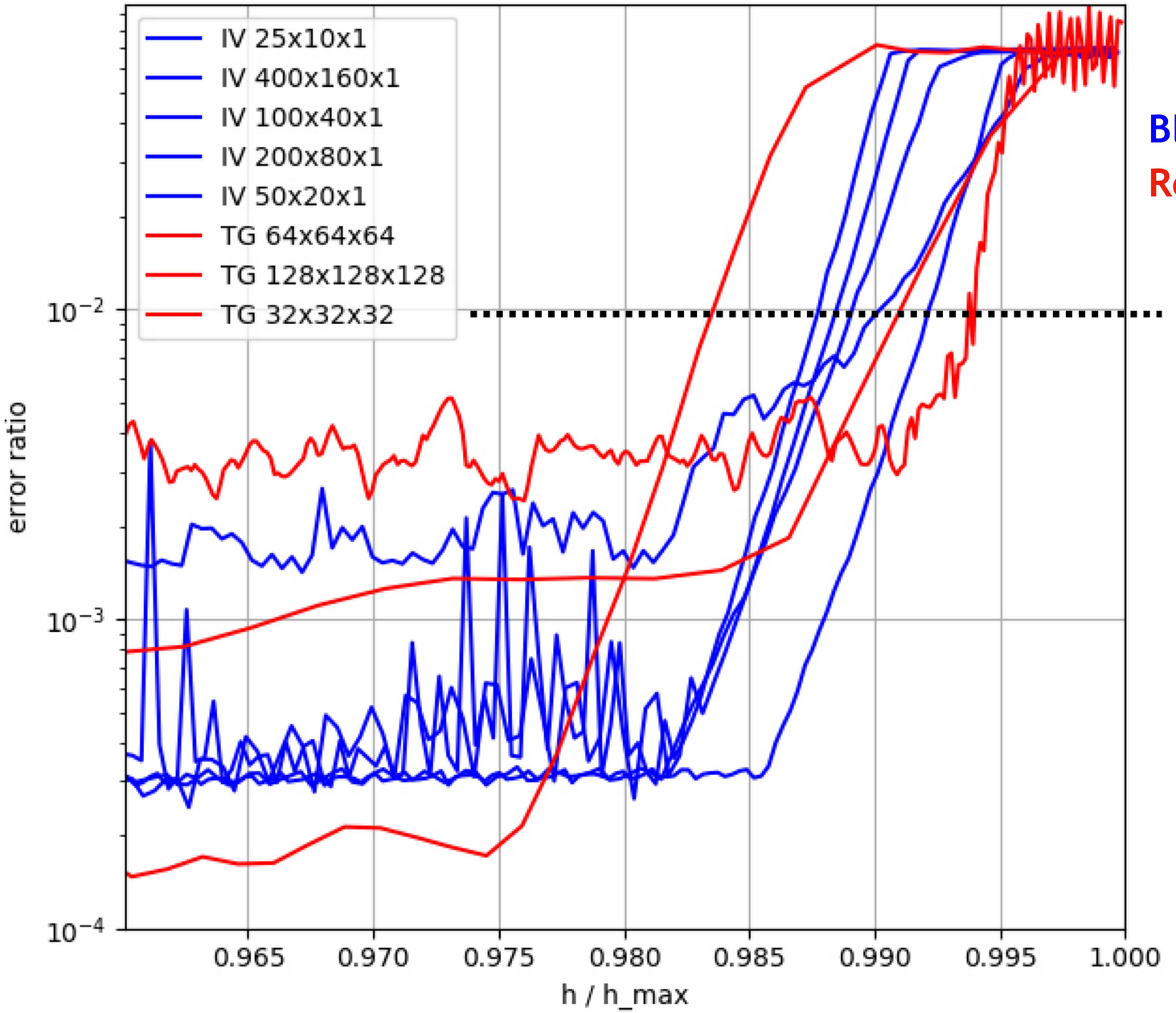


Empirical ramp tests showcase why cascade control works

2-D isentropic vortex ramp test results
4 levels of refinement

Same target ratio for all resolutions

Empirical ramp tests showcase why cascade control works



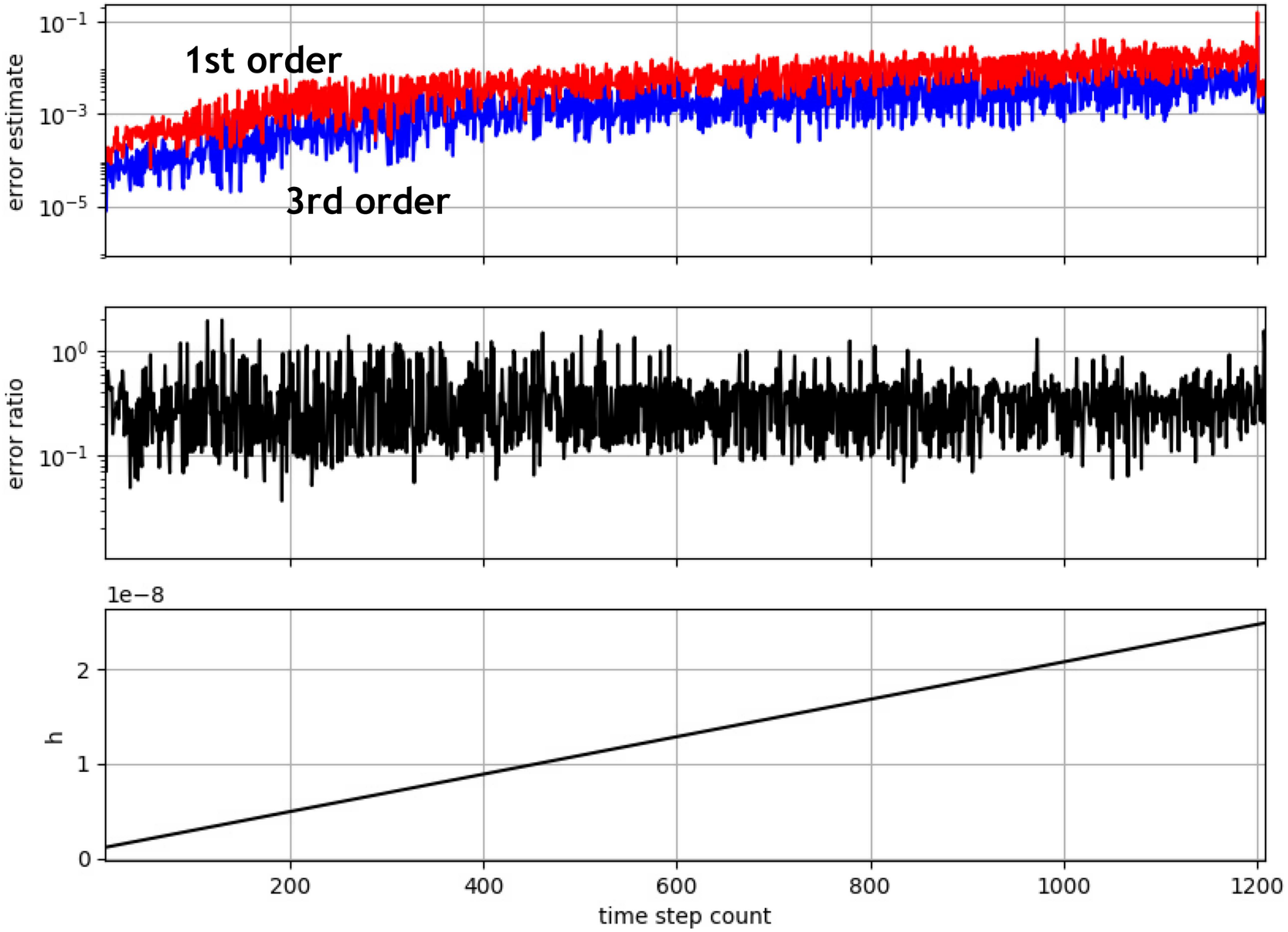
Blue: isentropic vortex, 2D

Red: invsicid Taylor-Green vortex, 3D

Both problems,
all resolutions,
same target ratio!

Same behavior and scales observed
on viscous-dominated problems

Empirical ramp tests showcase why cascade control works, and where it doesn’t…



Flat plate DNS (demo grid)

Transient startup for shock-BL interaction, high-speed flow

Initial condition is far from a physically realizable state

No change of the error ratio as we approach instability!

Higher-order embedded estimator is hardly more accurate than 1st-order…

Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

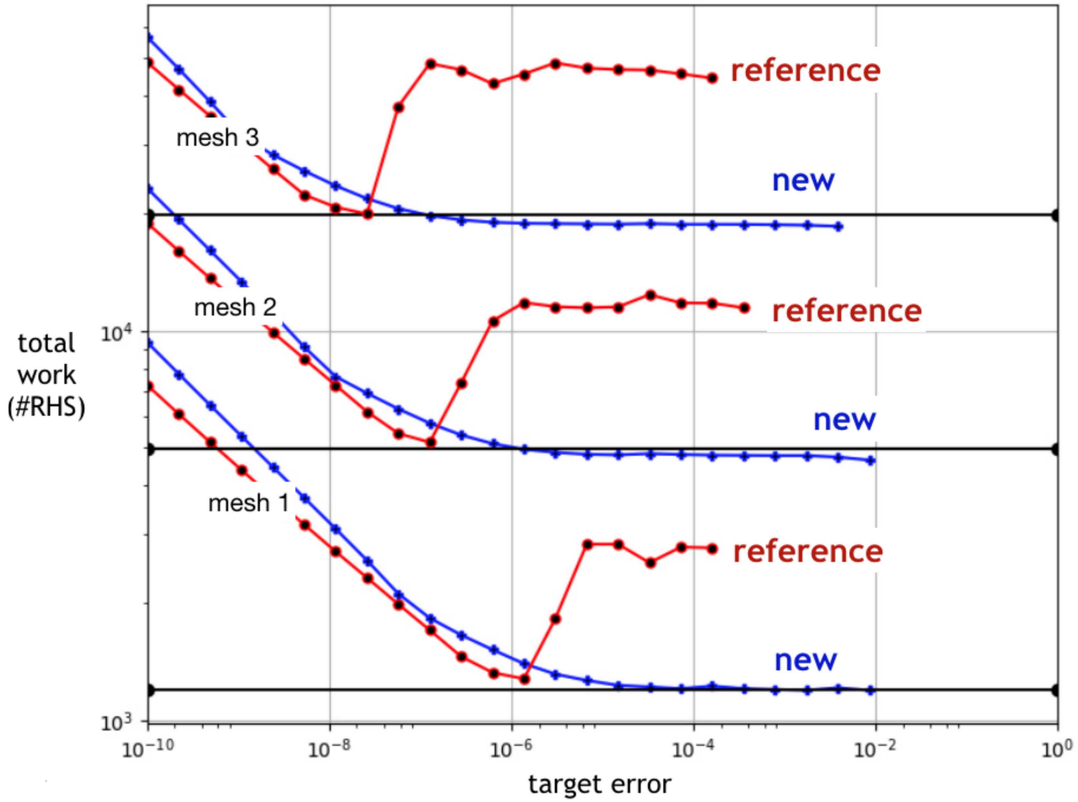
Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

Importance of the stability of embedded error estimators

Better estimators = better efficiency with less guess-and-check

- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup



Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

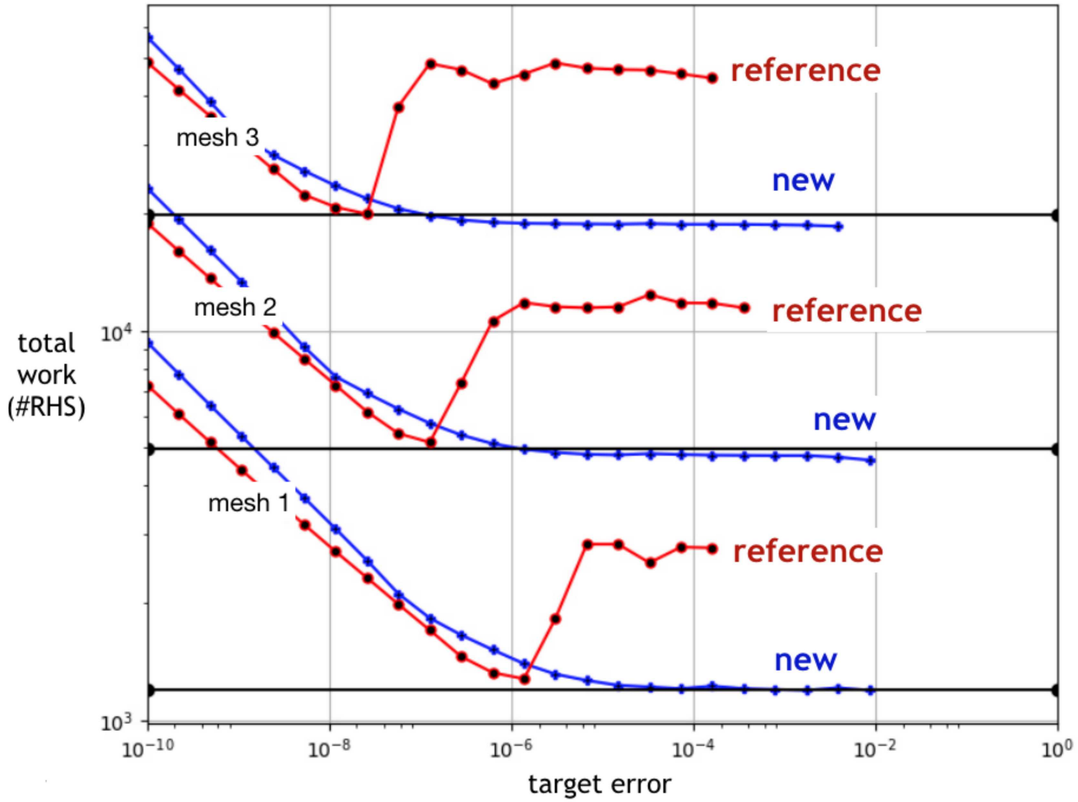
Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

Importance of the stability of embedded error estimators

Better estimators = better efficiency with less guess-and-check

- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup

Cascade control for greater generality and
problems with large temporal variations



Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

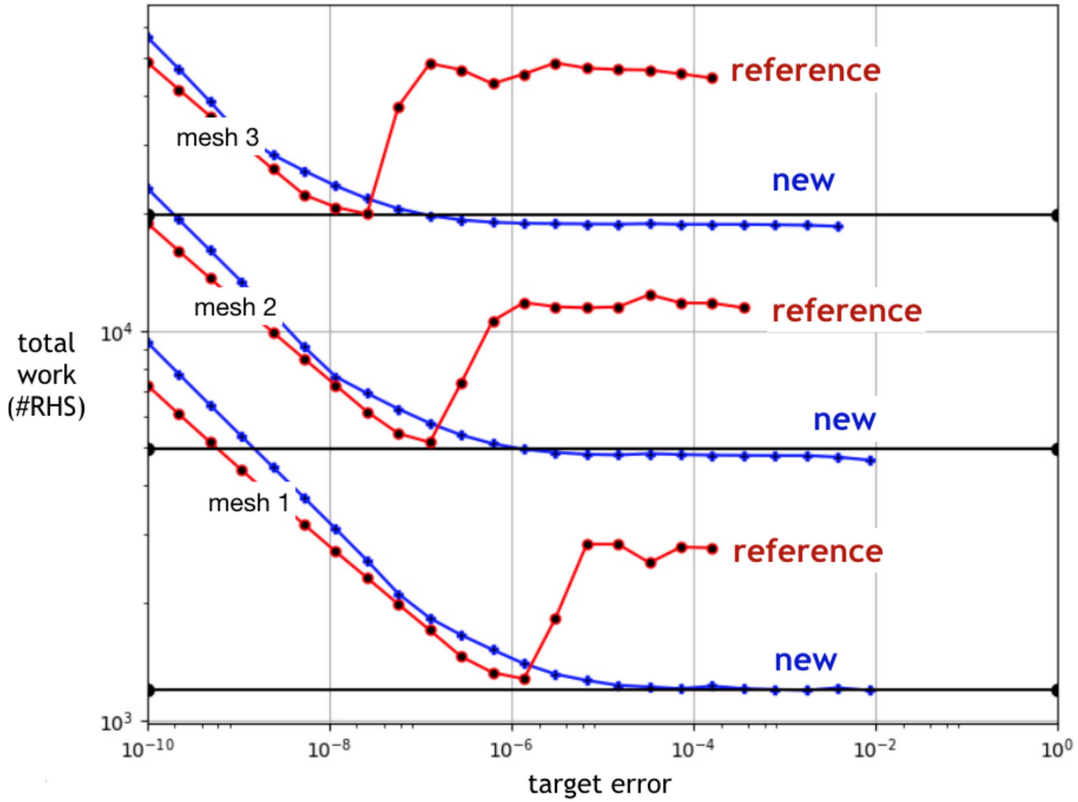
Importance of the stability of embedded error estimators

Better estimators = better efficiency with less guess-and-check

- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup

Cascade control for greater generality and
problems with large temporal variations

Efficient use of additional embedded Runge-Kutta methods



Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

Importance of the stability of embedded error estimators

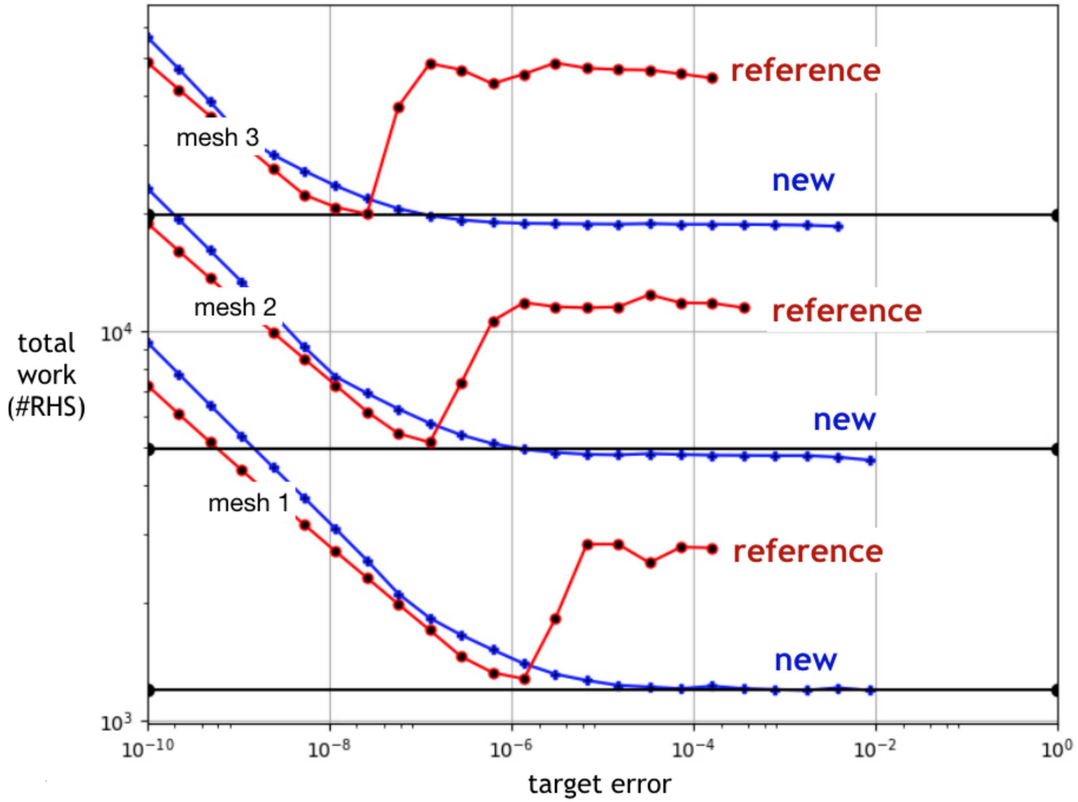
Better estimators = better efficiency with less guess-and-check

- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup

Cascade control for greater generality and problems with large temporal variations

Efficient use of additional embedded Runge-Kutta methods

Remarkable similarity across many problems, with correct error floor, 1st-order second estimator, and L-infinity norm



Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

Importance of the stability of embedded error estimators

Better estimators = better efficiency with less guess-and-check

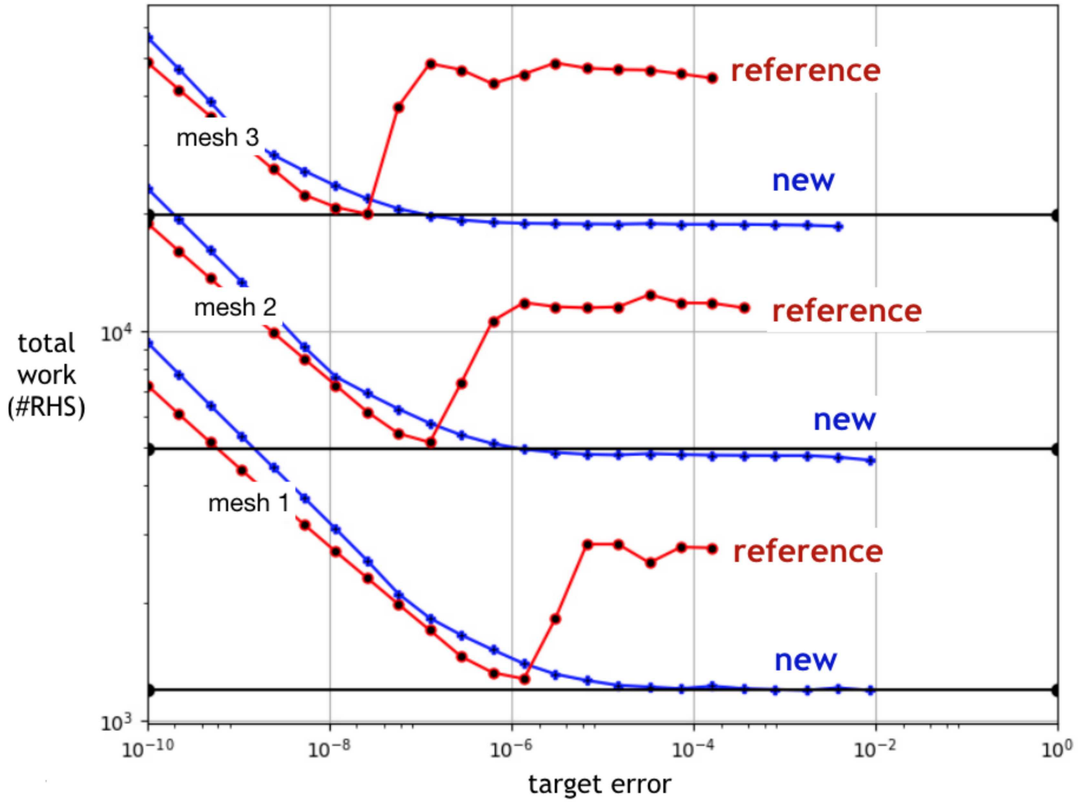
- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup

Cascade control for greater generality and problems with large temporal variations

Efficient use of additional embedded Runge-Kutta methods

Remarkable similarity across many problems, with correct error floor, 1st-order second estimator, and L-infinity norm

Inappropriate currently for problems with bad initial conditions



Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

Importance of the stability of embedded error estimators

Better estimators = better efficiency with less guess-and-check

- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup

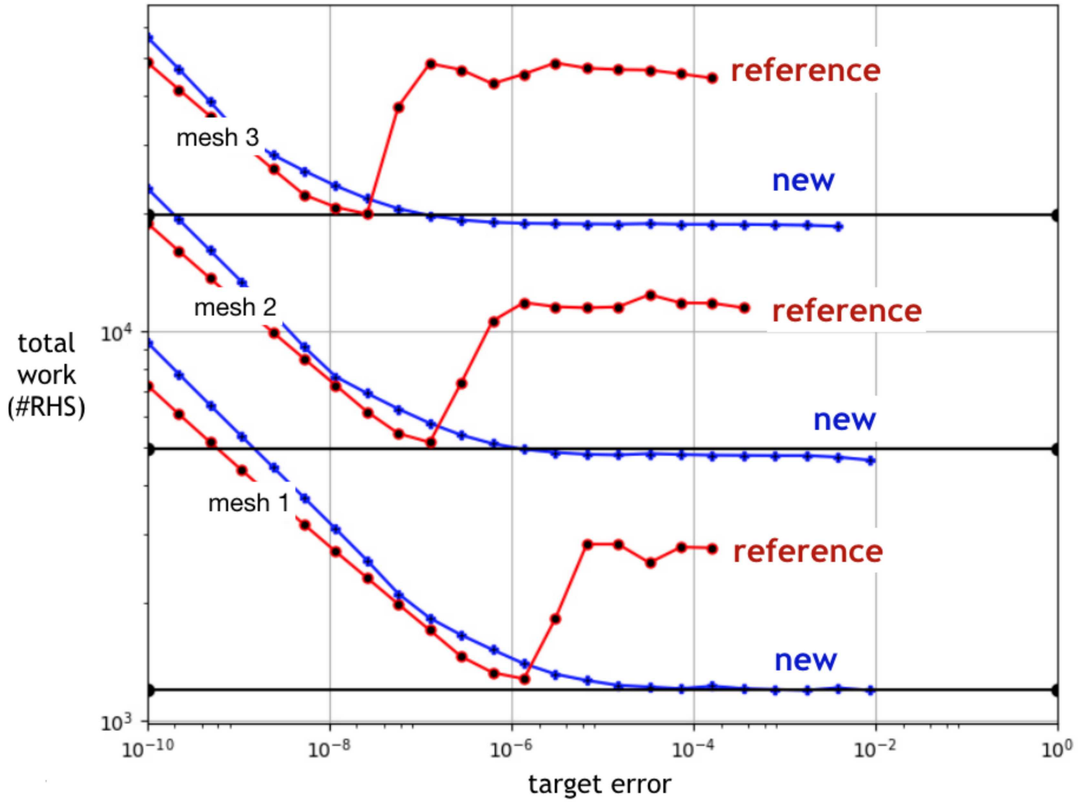
Cascade control for greater generality and
problems with large temporal variations

Efficient use of additional embedded Runge-Kutta methods

Remarkable similarity across many problems, with correct error floor, 1st-order second estimator, and L-infinity norm

Inappropriate currently for problems with bad initial conditions

Theory is matching results, but we want analytical predictions to better understand successes *and* failures



Improved embedded Runge-Kutta error estimators and a novel cascade control system are providing dramatic improvements in running fundamental reacting flow studies

Expensive and inefficient workflow for fundamental reacting flow studies,
due to time integration “guess-and-check” when stability limits performance

Importance of the stability of embedded error estimators

Better estimators = better efficiency with less guess-and-check

- Taylor-Green vortex 3x speedup
- Shu-Osher problem, 6x speedup

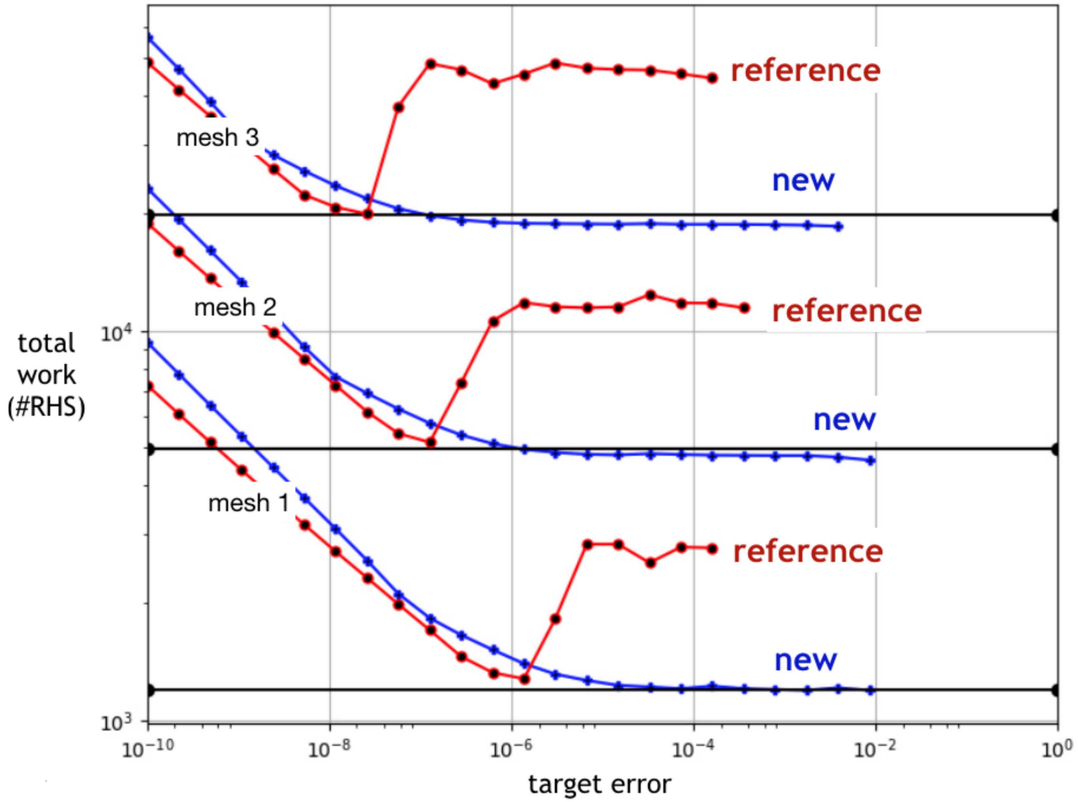
Cascade control for greater generality and
problems with large temporal variations

Efficient use of additional embedded Runge-Kutta methods

Remarkable similarity across many problems, with correct error floor, 1st-order second estimator, and L-infinity norm

Inappropriate currently for problems with bad initial conditions

Theory is matching results, but we want analytical predictions to better understand successes *and* failures



Direct ratio control?

Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Thanks to my collaborators
- Travis Fisher (SNL)
- Mark Carpenter (NASA)
- Brad Maeng (SNL)

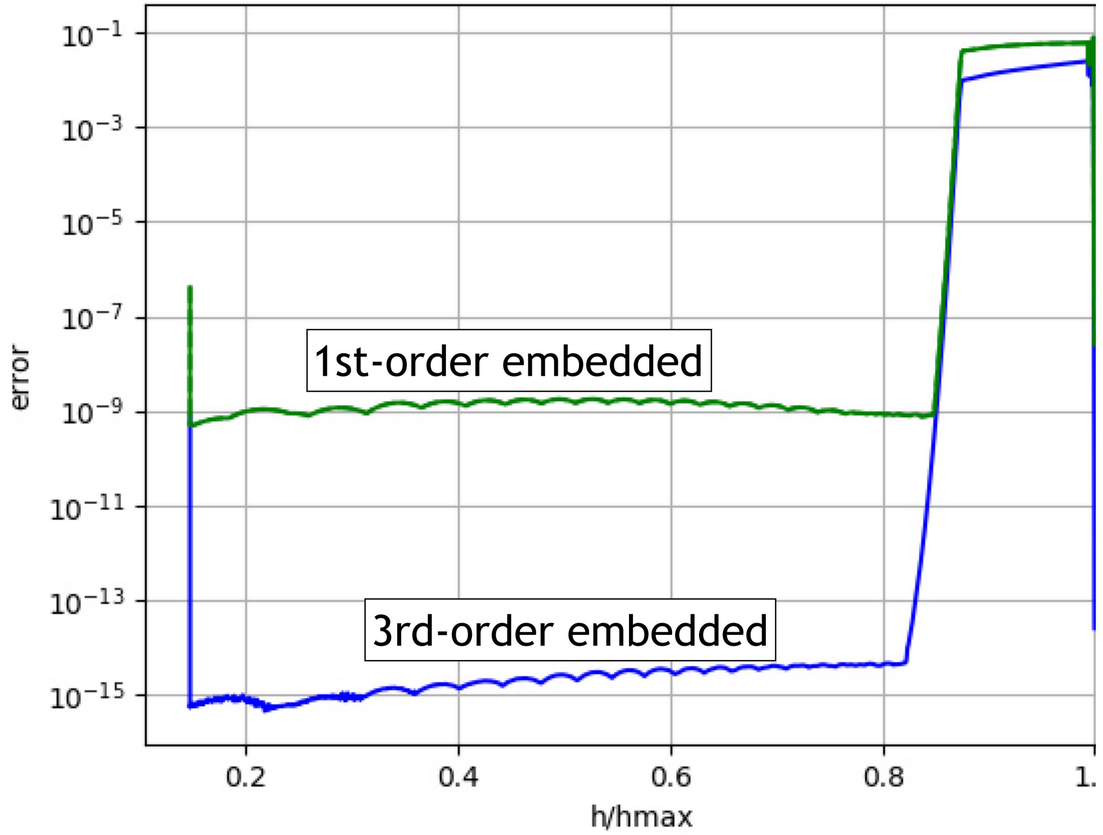
extras

Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate

Viscous shock problem - here we observe more accuracy of the lower-order estimator as mesh resolution increases

predominantly real eigenvalues, essentially a linear problem

50x1x1

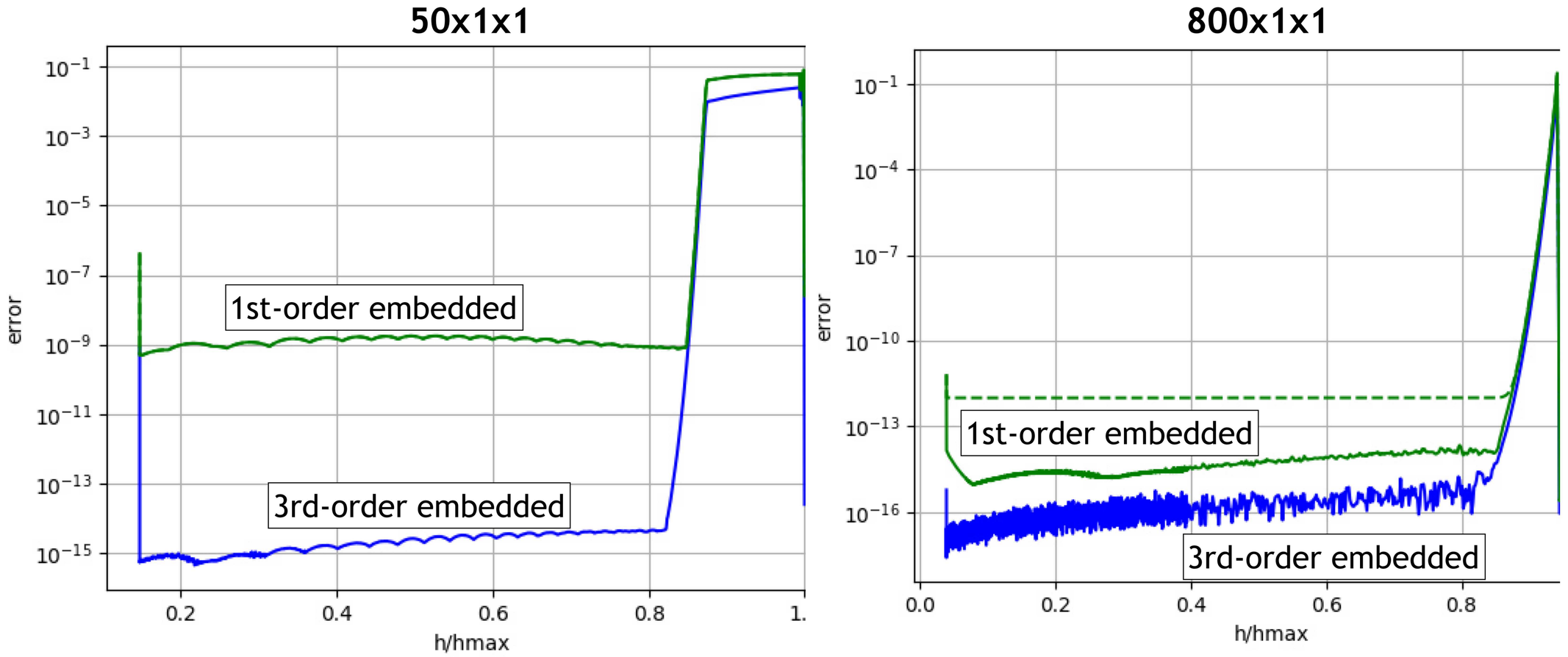


800x1x1

Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate

Viscous shock problem - here we observe more accuracy of the lower-order estimator as mesh resolution increases

predominantly real eigenvalues, essentially a linear problem

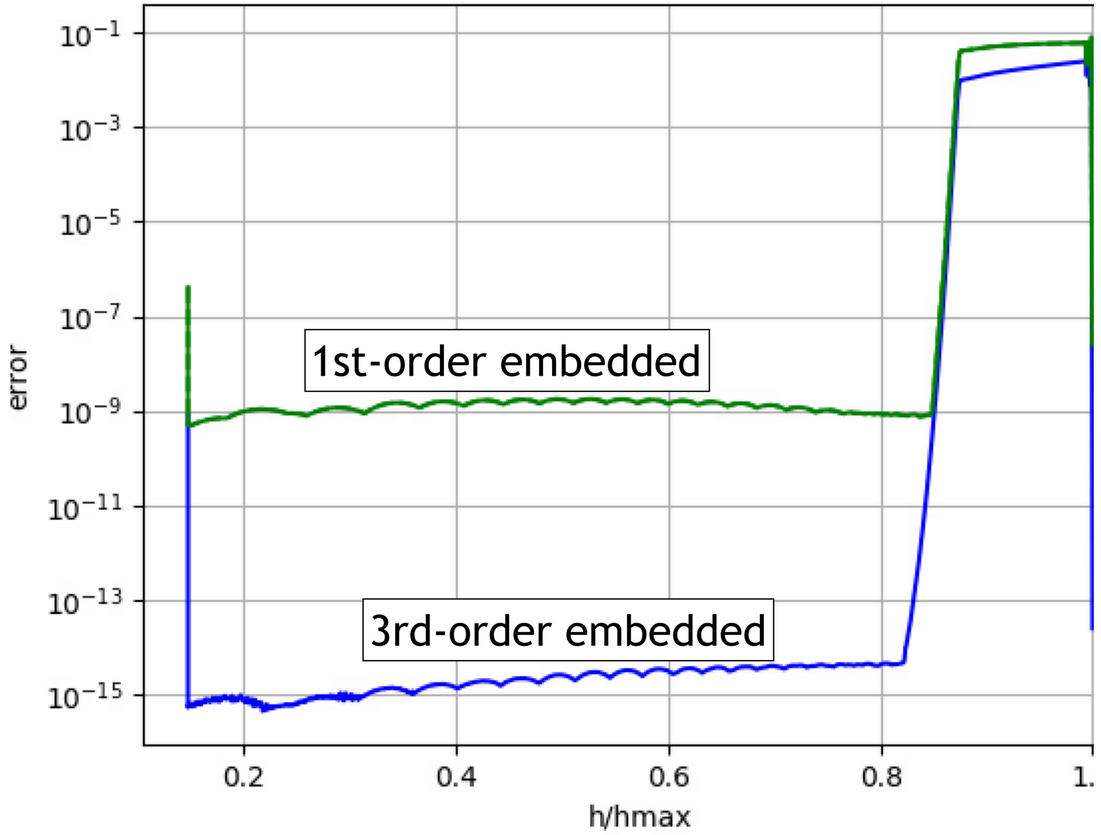


Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate

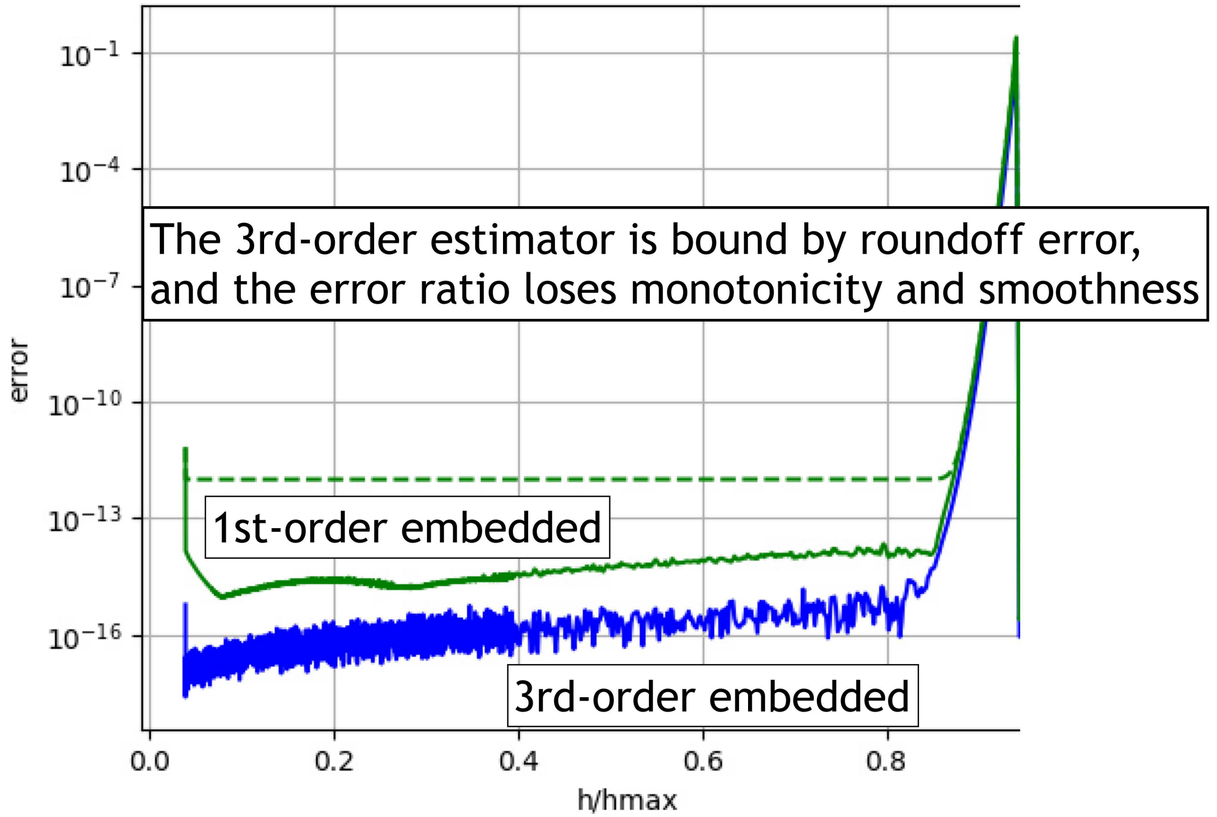
Viscous shock problem - here we observe more accuracy of the lower-order estimator as mesh resolution increases

predominantly real eigenvalues, essentially a linear problem

50x1x1



800x1x1

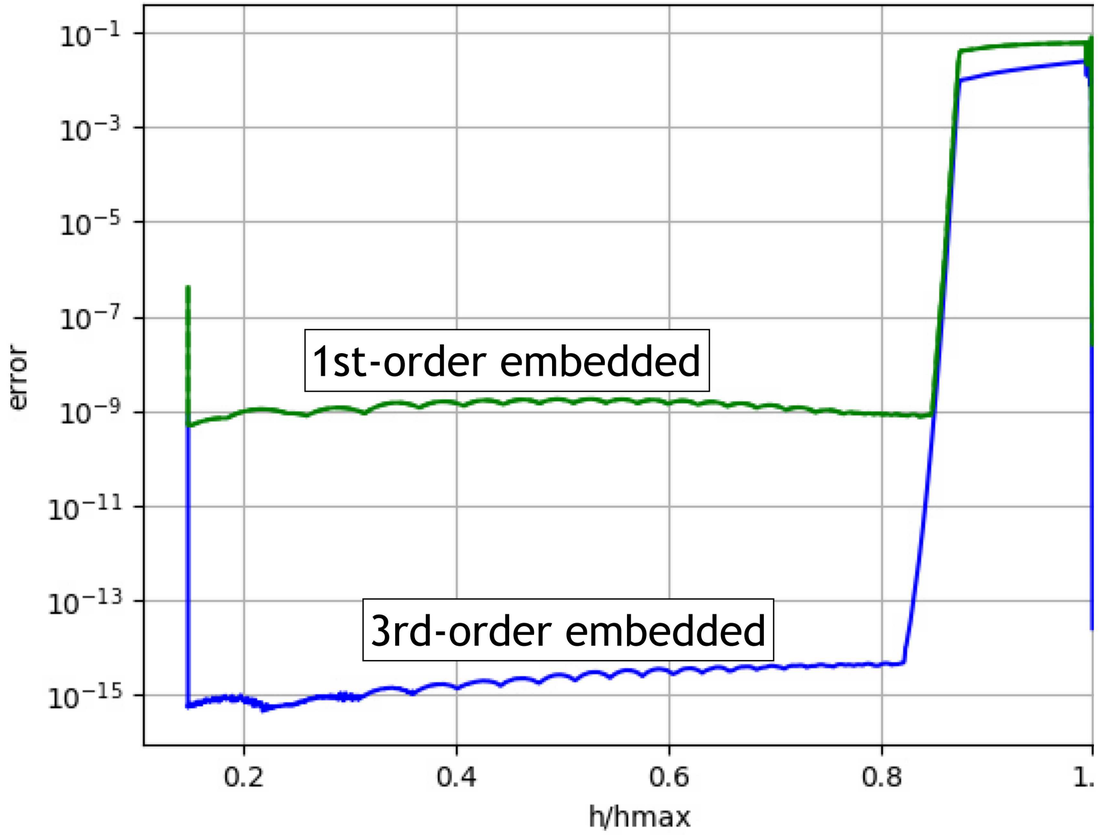


Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate

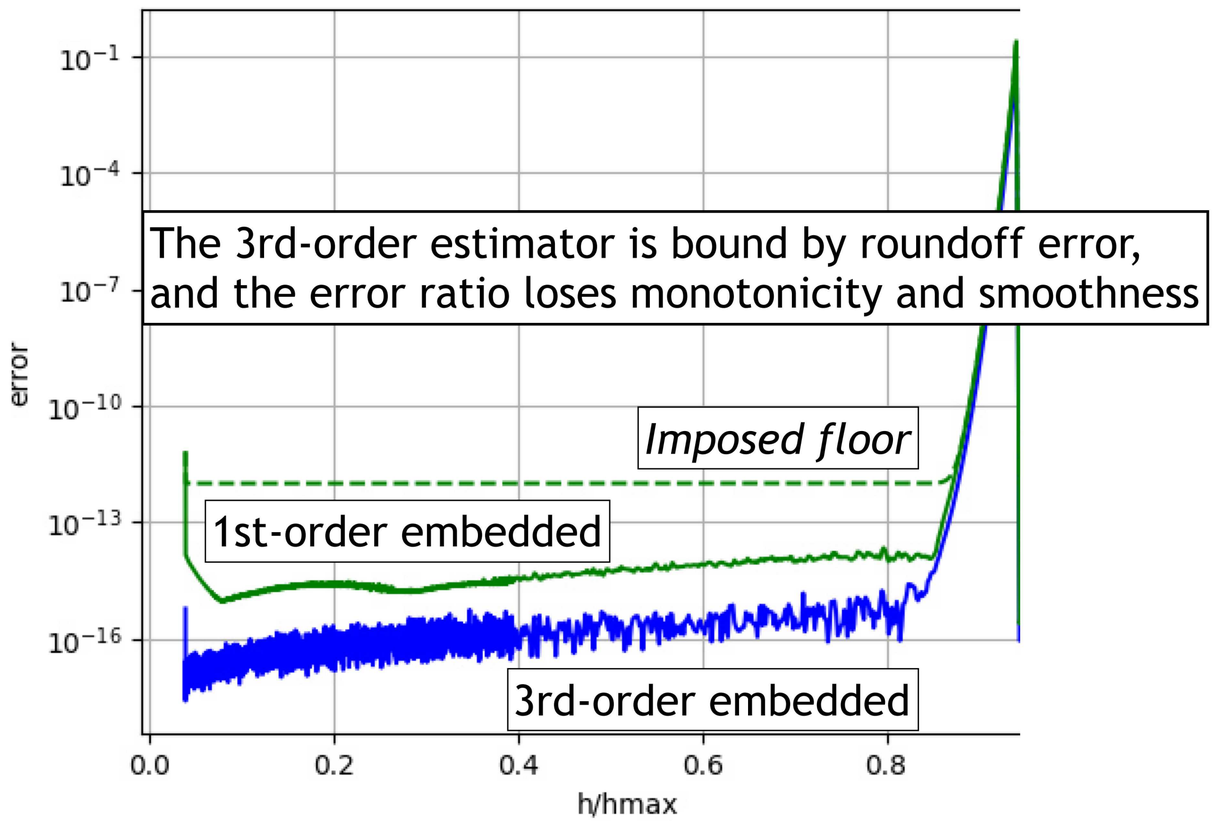
Viscous shock problem - here we observe more accuracy of the lower-order estimator as mesh resolution increases

predominantly real eigenvalues, essentially a linear problem

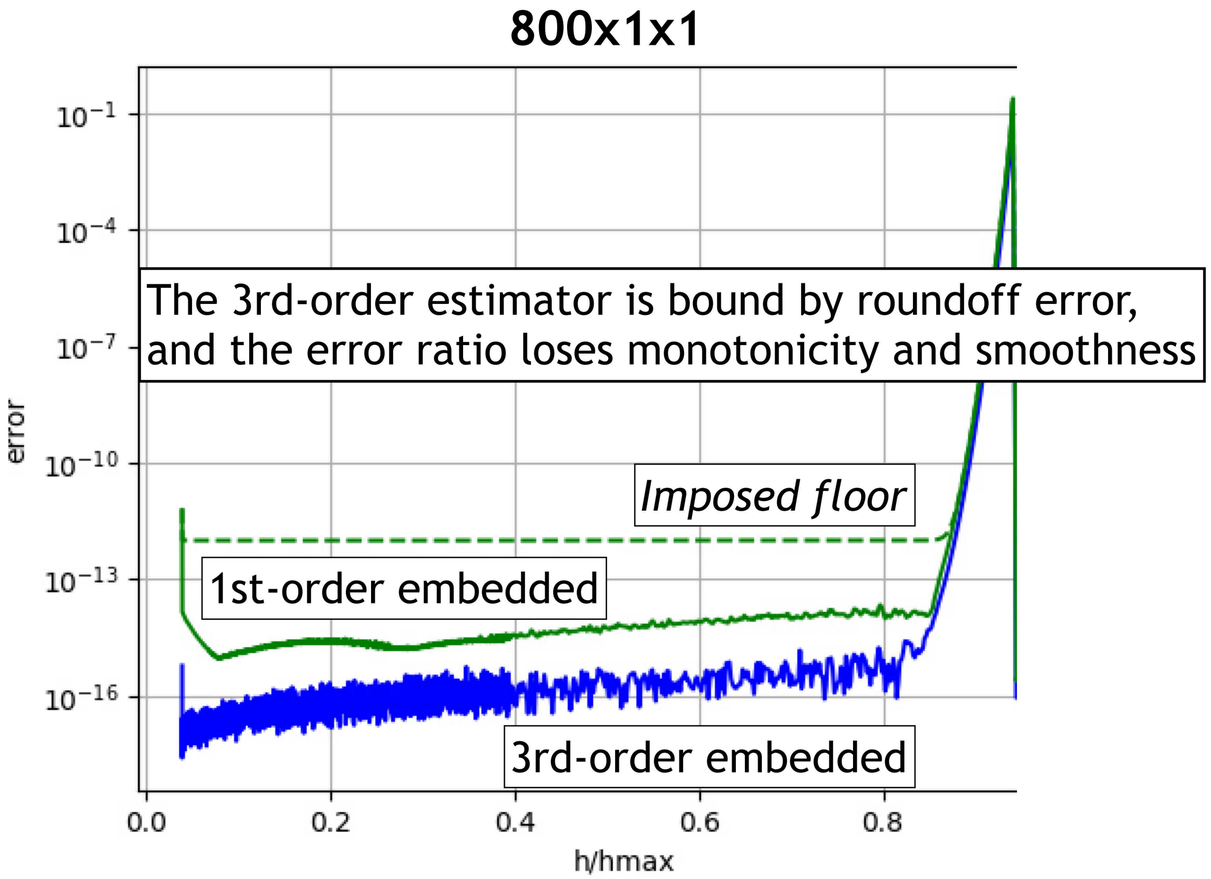
50x1x1



800x1x1

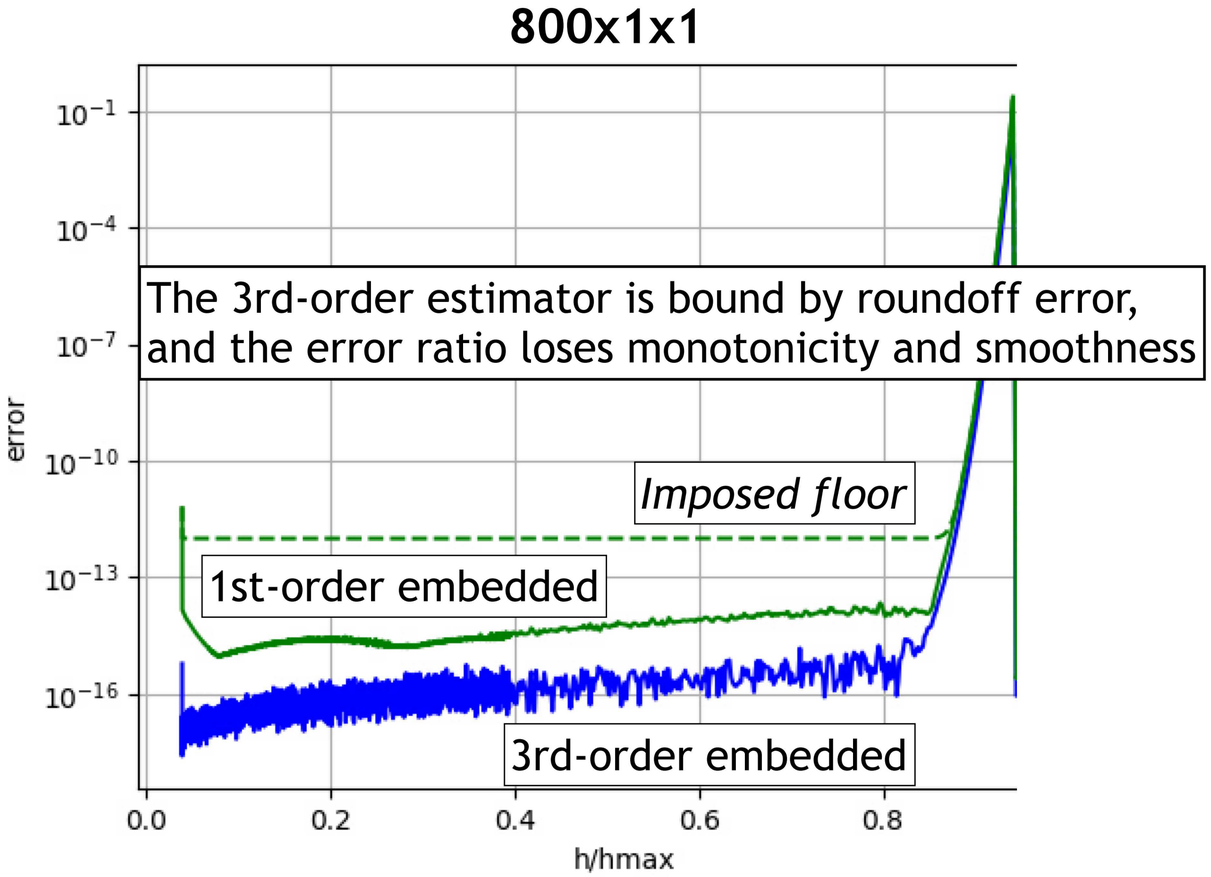


Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate



$$g = \frac{e_2}{e_1 + \sigma\epsilon_{\text{machine}}}$$

Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate

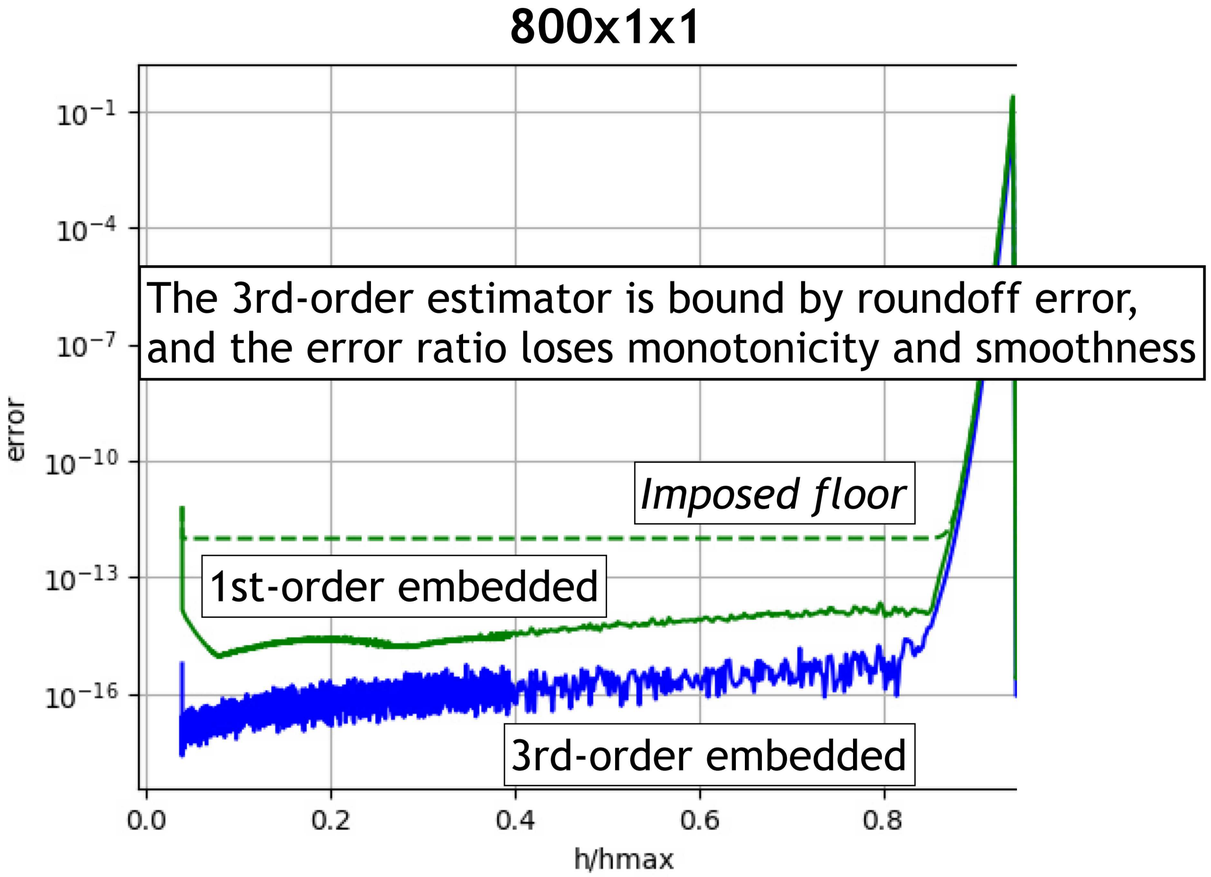


$$g = \frac{e_2}{e_1 + \sigma\epsilon_{\text{machine}}}$$

$$\lim_{\Delta t \to 0} g = \frac{1}{1+\sigma}$$

Choose σ = 10^4

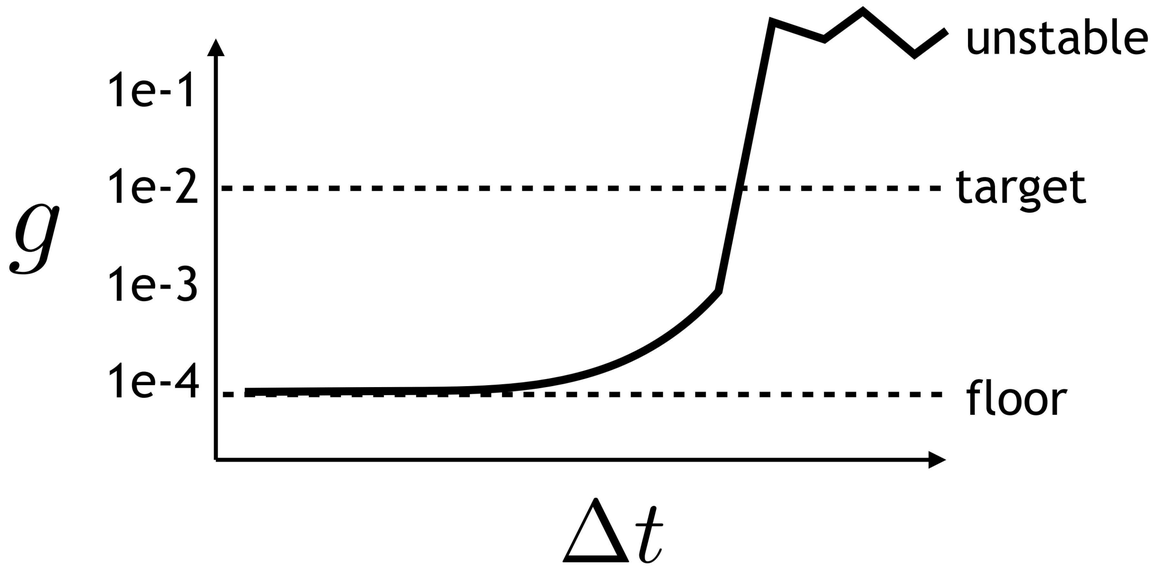
Empirical ramp tests showcase why cascade control works, and that a floor is necessary when the lower-order estimator is very accurate



$$g = \frac{e_2}{e_1 + \sigma\epsilon_{\text{machine}}}$$

$$\lim_{\Delta t \to 0} g = \frac{1}{1+\sigma}$$

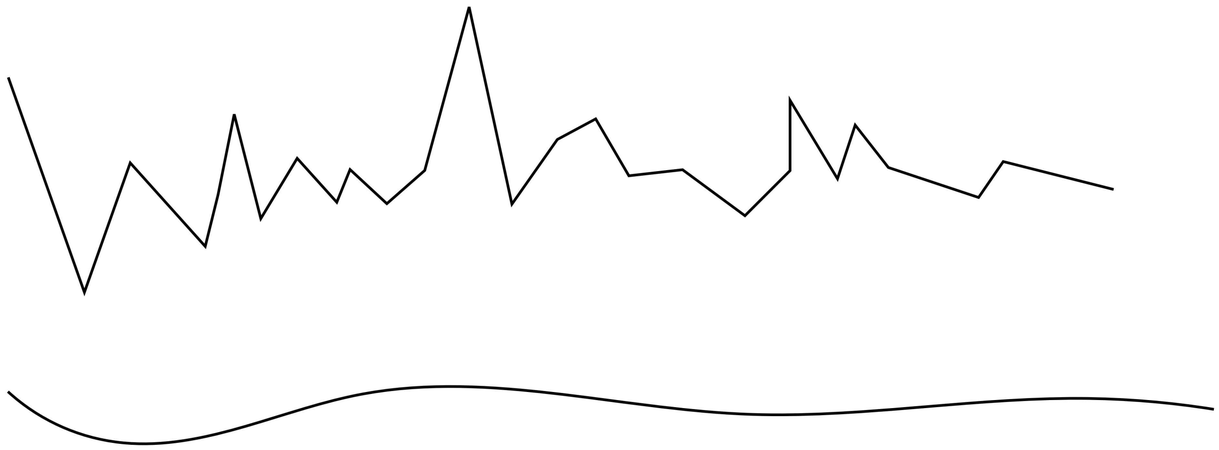
Choose σ = 10^4



- use first-order method to get smoothness and controllability (Backward Euler)
- show ramp test results, show result alongside ramp test g(h) plot, shows instability detection, show resolution independence on isentropic vortex problem
- isentropic vortex a posteriori tests show that the cascade controller (as tuned) added oscillation while matching the mean time step of optimal target errors in PID control, across resolution
- controller dynamics of the cascade controller: needs to be slower - common in cascade control, master controller must allow subservient controller to equilibrate
- it works and the goal of reducing problem dependence appears to succeed
- trying instability detection directly as a time step controller, easier in concept, harder in code as is
- problem with transient startup on awful IC (usually not done with high-order method anyway) one option here is to use a third estimator with less error than the existing 3rd-order one for step control, which has the error cliff behavior. Eliminate the cliff and the ratio may show better behavior here.
- problem with viscous problem seeing 1st-order get closer to 3rd-order at higher resolution. Unexpected and not observed in the inviscid Taylor-Green vortex where more resolution makes 1st-order worse, one option here is to try FE as the first-order method. Need to look at stability regions of the 1st-order methods, as well as theoretical predictions of the error ratio. This is essentially the same problem as the transient startup

Is there an argument for cascade control in general?
Why not always just use the detector as a sole controller?
Find argument from engineering for this

Low-pass filters are built into the controller



The offset of numerical instability is a high-frequency ‘sawtooth’

Filtering removes controller responsiveness precisely at the stability boundary!