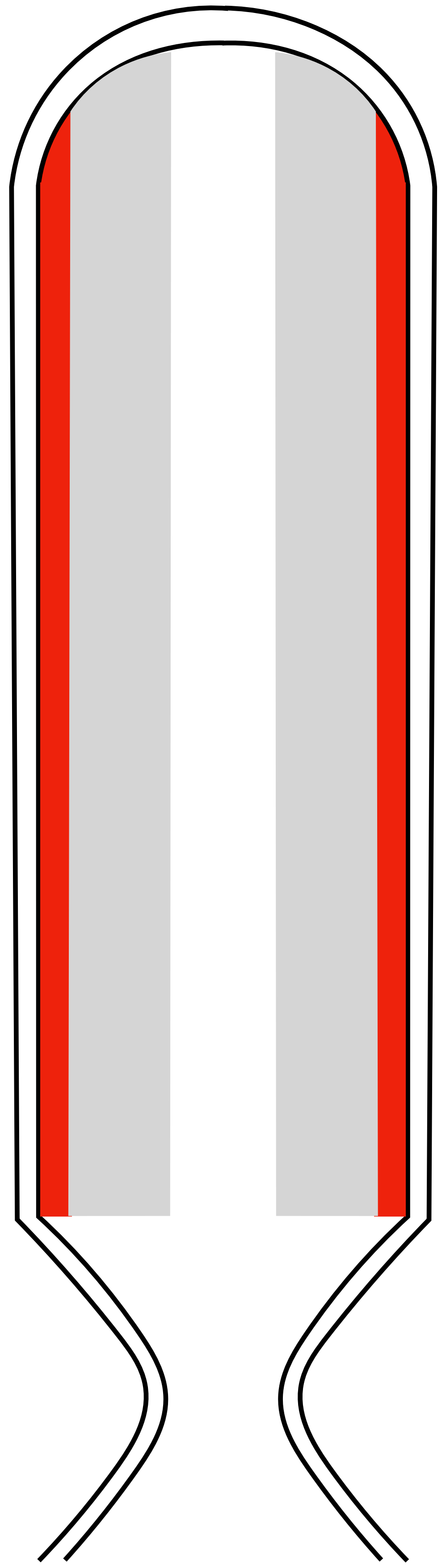
Sliding Basis Optimization for Heterogeneous Material Design

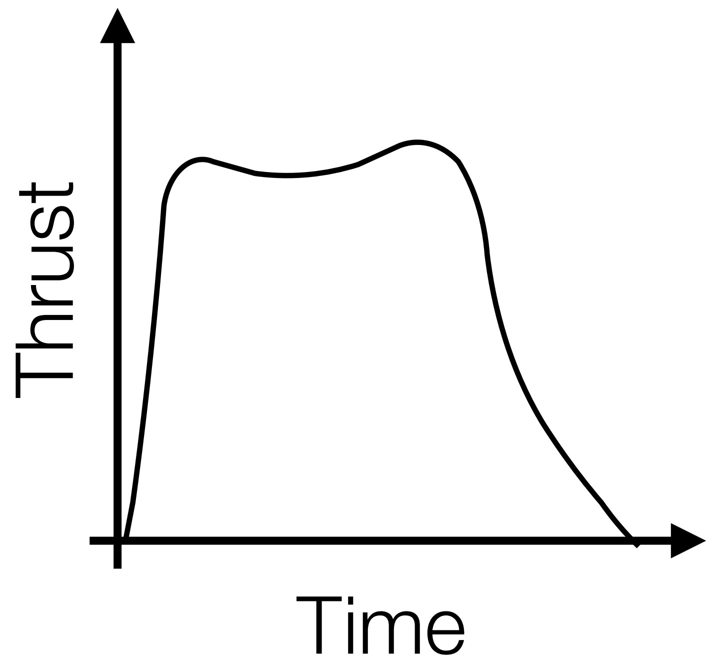
Nurcan Gecer Ulu, Svyatoslav Korneev, Erva Ulu, Saigopal Nelaturi

Palo Alto Research Center

Space Shuttle launch image courtesy **NASA.gov**

Solid Rocket Propellant Design

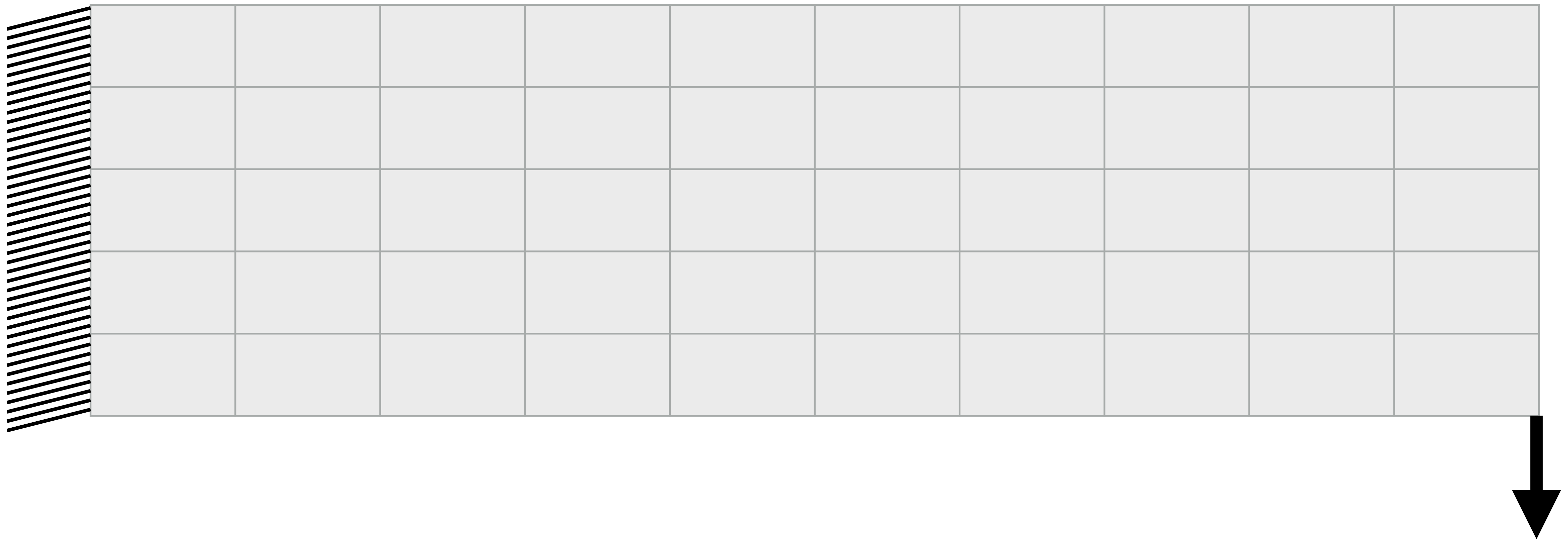




Geometry design with single material is not enough to achieve both desired thrust and eliminate insulation !!

General Material Design Optimization Pipeline

$$\begin{aligned} \min_{\boldsymbol{\mathcal{F}}} \quad & f(\boldsymbol{\mathcal{F}}) \\ \text{s.t.} \quad & g_i(\boldsymbol{\mathcal{F}}) \leq 0 \end{aligned}$$



$$\Phi(\boldsymbol{\mathcal{F}}) = 0$$

Costly analysis!

Complex physics!

Maybe black box!

Difficult to derive analytical gradients

Numerical gradients are not practical for large scale problems

Model reduction approach to reduce number of optimization variables

$$\begin{aligned}\min_{\boldsymbol{\mathcal{F}}} \quad & f(\boldsymbol{\mathcal{F}})\\ \text{s.t.} \quad & g_i(\boldsymbol{\mathcal{F}}) \leq 0\end{aligned}$$

$$\Phi(\boldsymbol{\mathcal{F}}) = 0$$

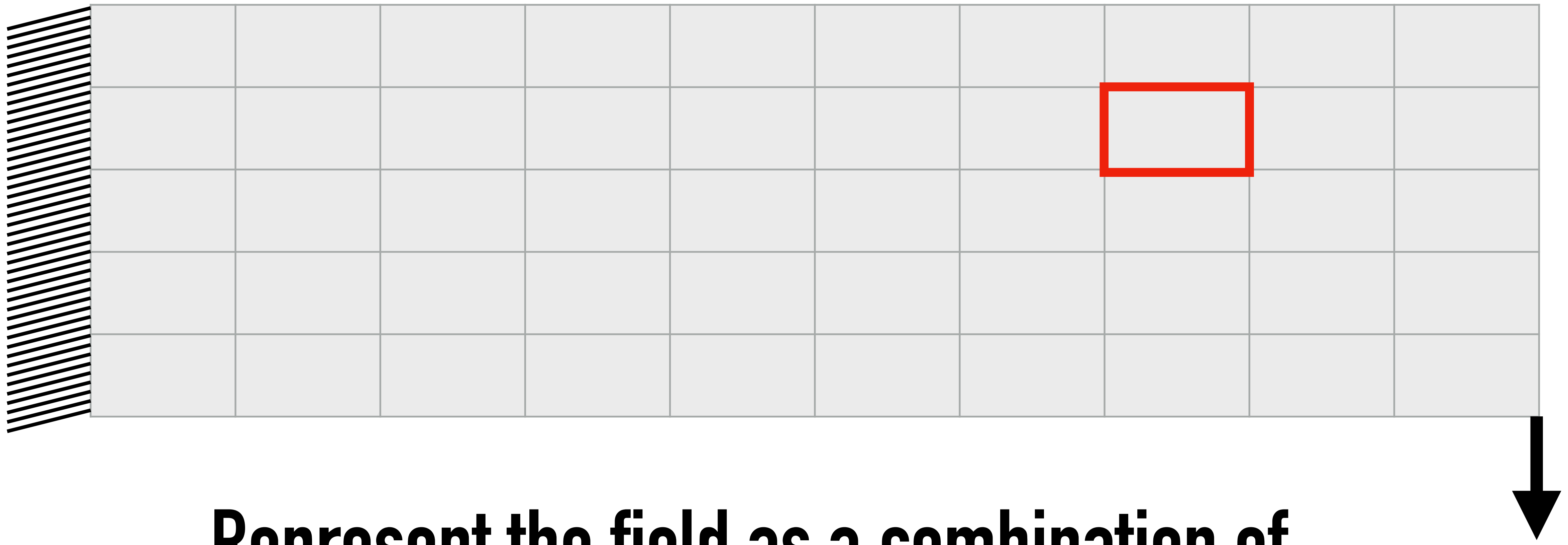
thousand-millions

➡

$$\begin{aligned}\min_{\boldsymbol{w}} \quad & f(\boldsymbol{w})\\ \text{s.t.} \quad & g_i(\boldsymbol{w}) \leq 0\end{aligned}$$

$$\Phi(\boldsymbol{w}) = 0$$

tens-hundreds



Represent the field as a combination of small set of basis functions!

$$\boldsymbol{\mathcal{F}} = \boldsymbol{B}\boldsymbol{w}$$

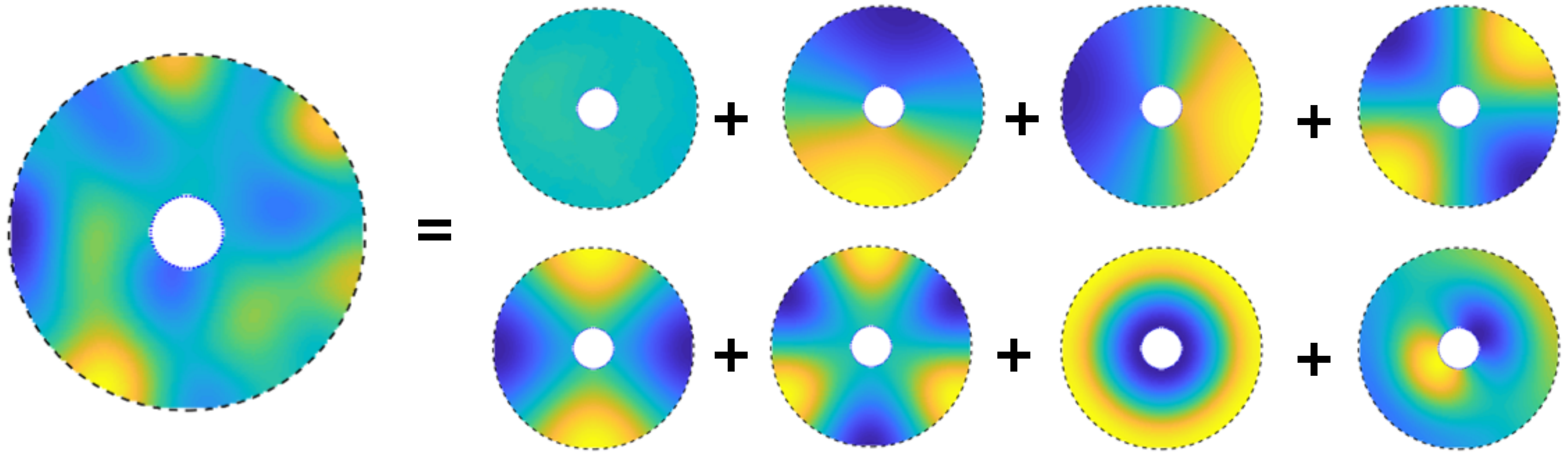
Reduce number of design variables without compromising analysis quality!

Representing Material Distributions Using Shape Harmonics

- Fourier expansion of signal into harmonics (sin + cos functions) can be applied to manifold decomposition
- Fourier bases are eigenfunctions of the Laplacian on the unit interval
- Spherical harmonics are eigenfunctions of the Laplacian on the sphere

$$\lambda_i \boldsymbol{e_i} = \boldsymbol{\mathcal{L} e_i} \quad \boldsymbol{B} = [\boldsymbol{e_1}, \boldsymbol{e_2}, ..., \boldsymbol{e_k}]$$

- Idea can be generalized to harmonics over any manifold
- Weighted sum of manifold harmonic basis can be used to describe shape and material in a ‘frequency domain’



$$\mathcal{F} = \boldsymbol{B w}$$

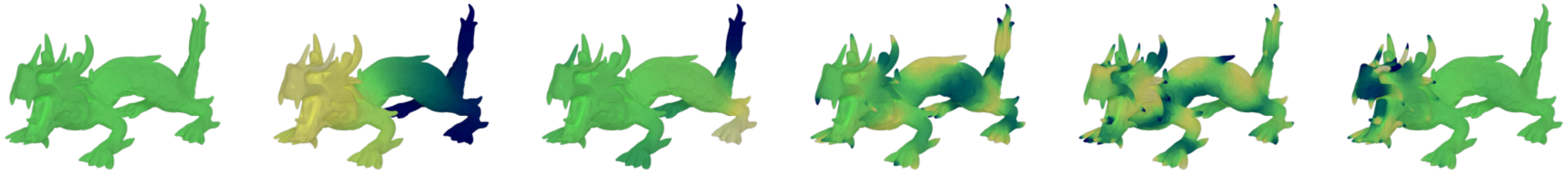
10k quad mesh **8 weights**

We represent a field with fewer parameters using Laplacian basis

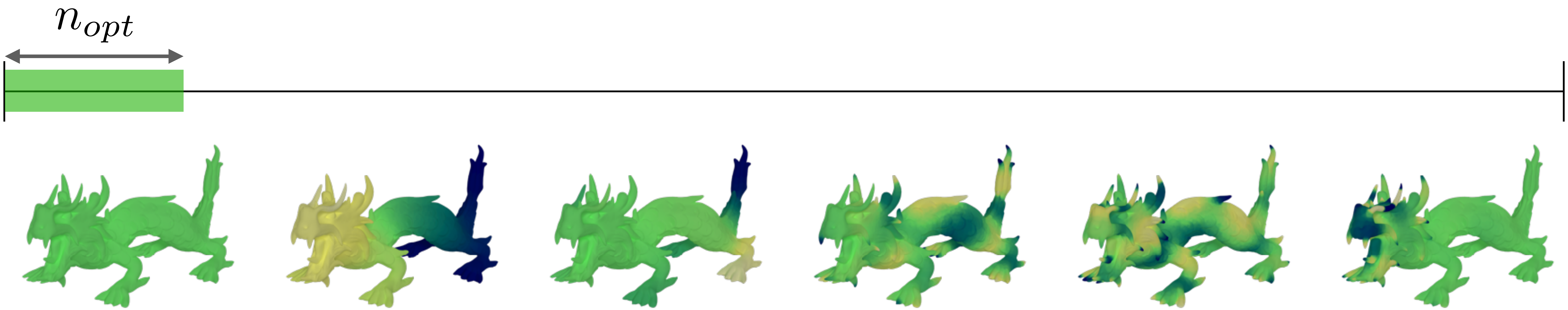
Key Observation

low frequency *high frequency*

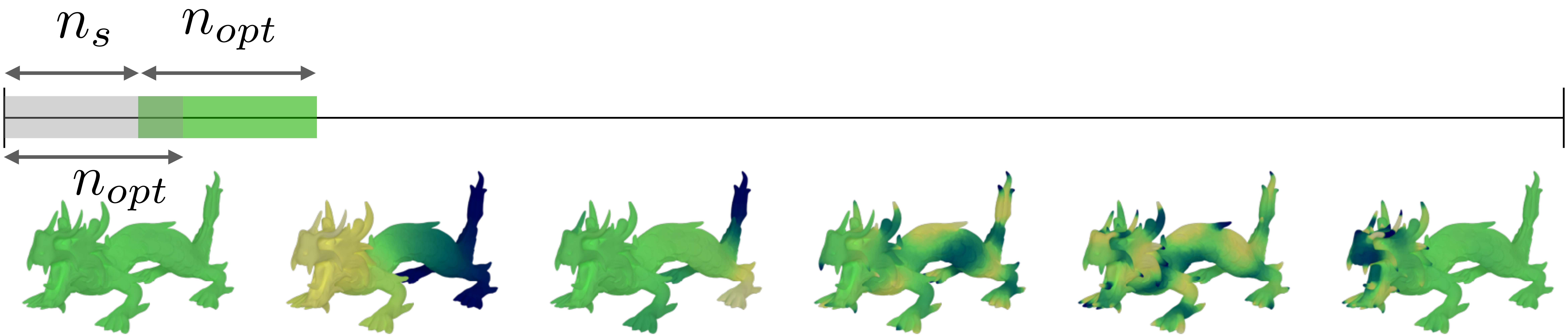
basis functions



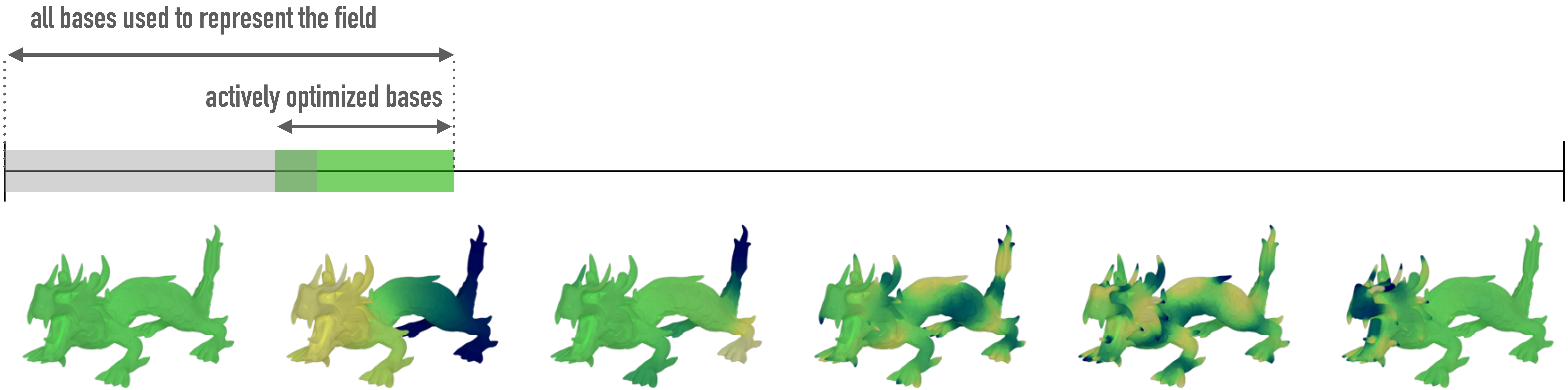
Sliding Basis Optimization



Sliding Basis Optimization

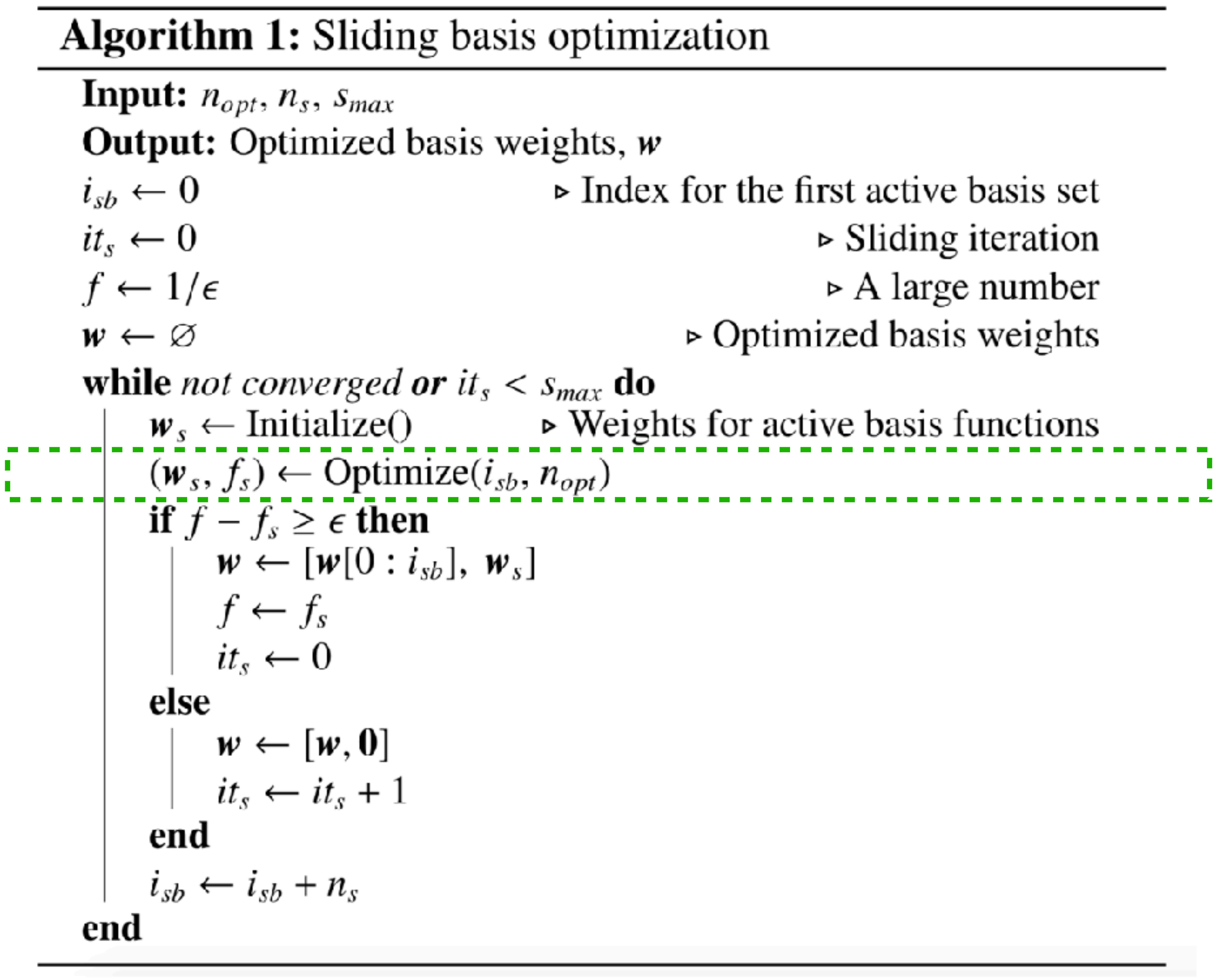


Sliding Basis Optimization



We usually get convergence using only a small set of basis functions!

Sliding basis optimization is a top level framework that works with existing optimization methods



Algorithm 1: Sliding basis optimization

```
Input: $n_{opt}$, $n_s$, $s_{max}$
Output: Optimized basis weights, $\boldsymbol{w}$
$i_{sb} \leftarrow 0$                                  ▹ Index for the first active basis set
$it_s \leftarrow 0$                                    ▹ Sliding iteration
$f \leftarrow 1/\epsilon$                              ▹ A large number
$\boldsymbol{w} \leftarrow \varnothing$                ▹ Optimized basis weights
while not converged or $it_s < s_{max}$ do
    $\boldsymbol{w}_s \leftarrow$ Initialize()         ▹ Weights for active basis functions
    $(\boldsymbol{w}_s, f_s) \leftarrow$ Optimize($i_{sb}$, $n_{opt}$)
    if $f - f_s \geq \epsilon$ then
        $\boldsymbol{w} \leftarrow [\boldsymbol{w}[0 : i_{sb}],\ \boldsymbol{w}_s]$
        $f \leftarrow f_s$
        $it_s \leftarrow 0$
    else
        $\boldsymbol{w} \leftarrow [\boldsymbol{w}, \boldsymbol{0}]$
        $it_s \leftarrow it_s + 1$
    end
    $i_{sb} \leftarrow i_{sb} + n_s$
end
```

This optimize step can be implemented using general nonlinear optimizers

Sliding basis optimization speeds up differentiable problems, too

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & f(\boldsymbol{w}) \\ \text{s.t.} \quad & g_i(\boldsymbol{w}) \le 0 \end{aligned}$$

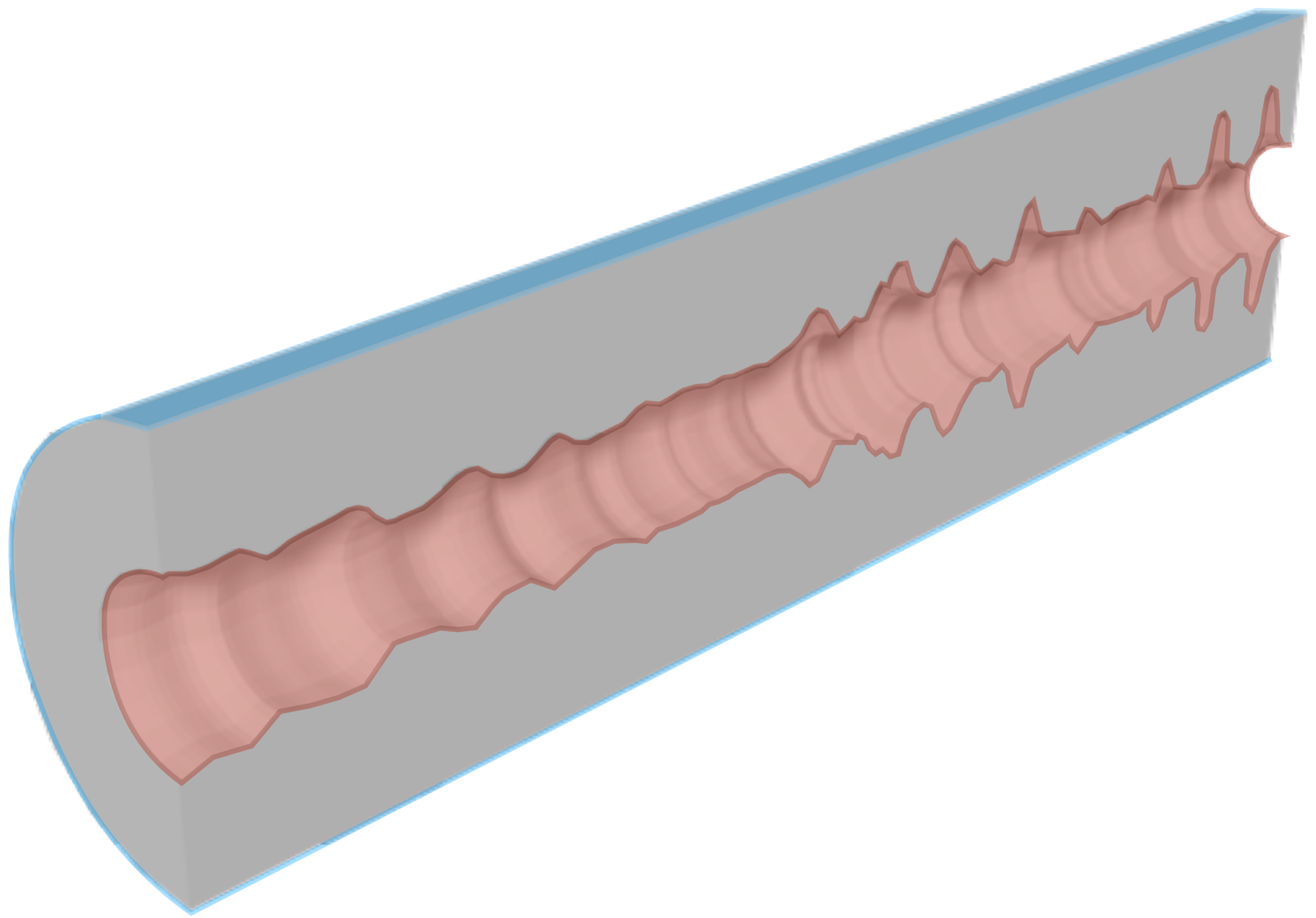
If gradients are derived
for the full resolution

$$\frac{\partial f}{\partial \boldsymbol{w}} = \boxed{\frac{\partial f}{\partial \boldsymbol{\mathcal{F}}}} \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \boldsymbol{w}}$$

Gradients w.r.t. basis weights can be found
through simple matrix multiplication with B

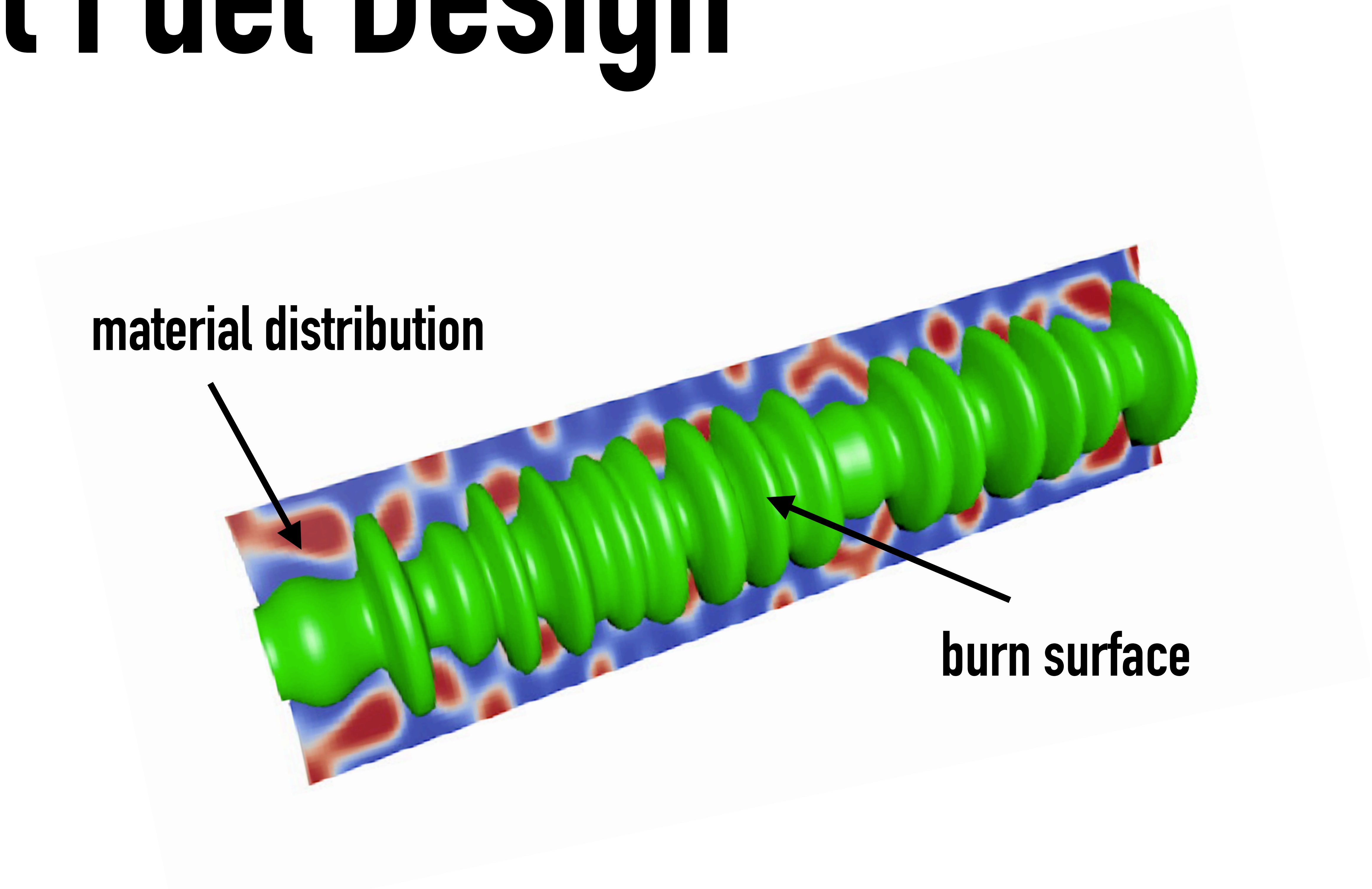
$$\boldsymbol{\mathcal{F}} = \boldsymbol{B}\boldsymbol{w}$$

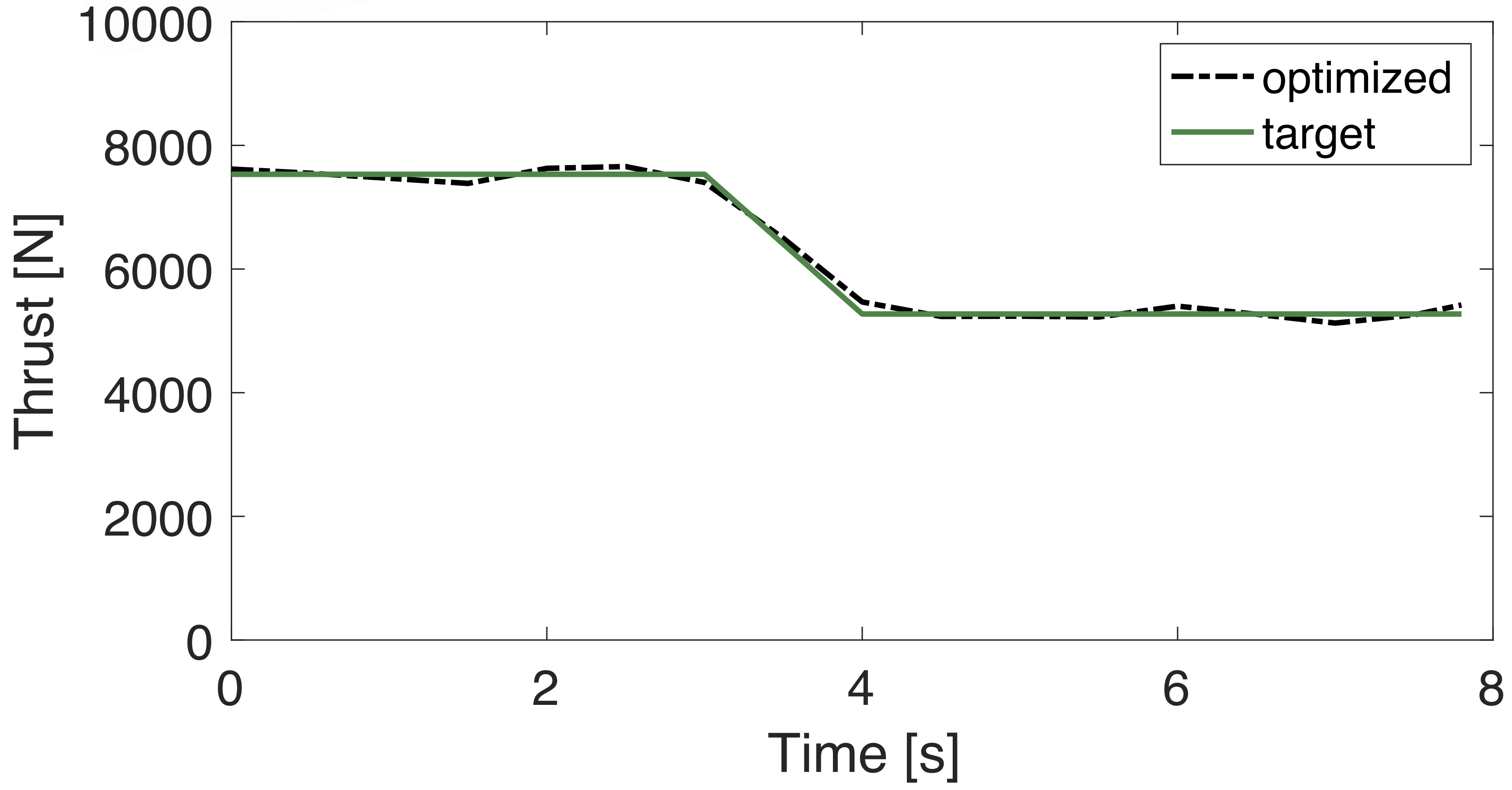
Solid Rocket Fuel Design



$$\min_{\boldsymbol{w}} \quad \sum_t \left(th(\boldsymbol{w}) - th_{target}\right)^2$$

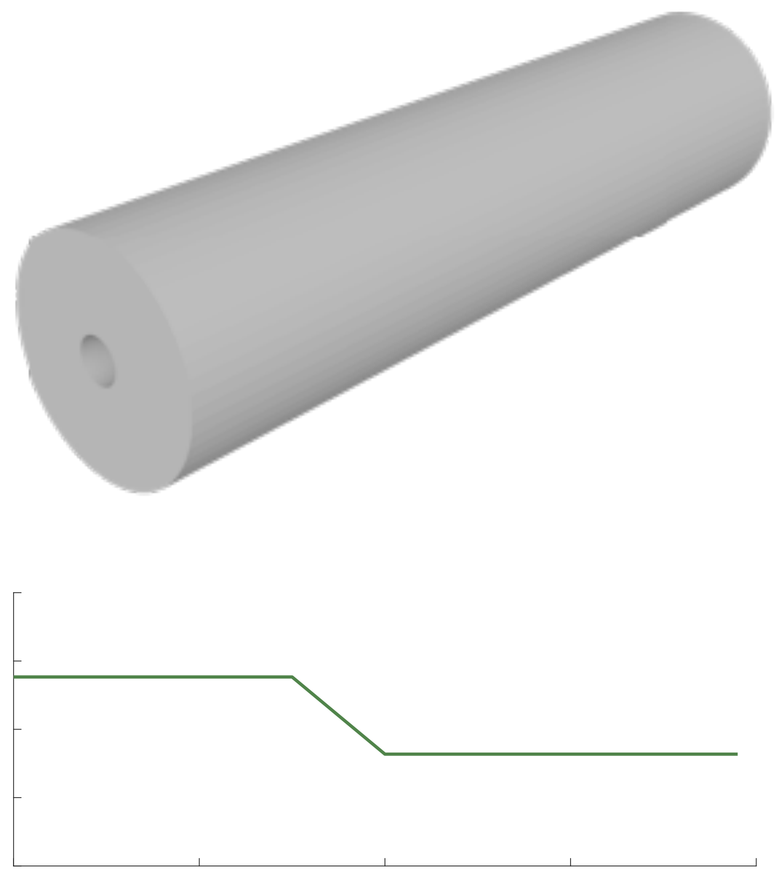
$$\text{s.t.} \quad r_b(\boldsymbol{w})^i > r_{in}$$



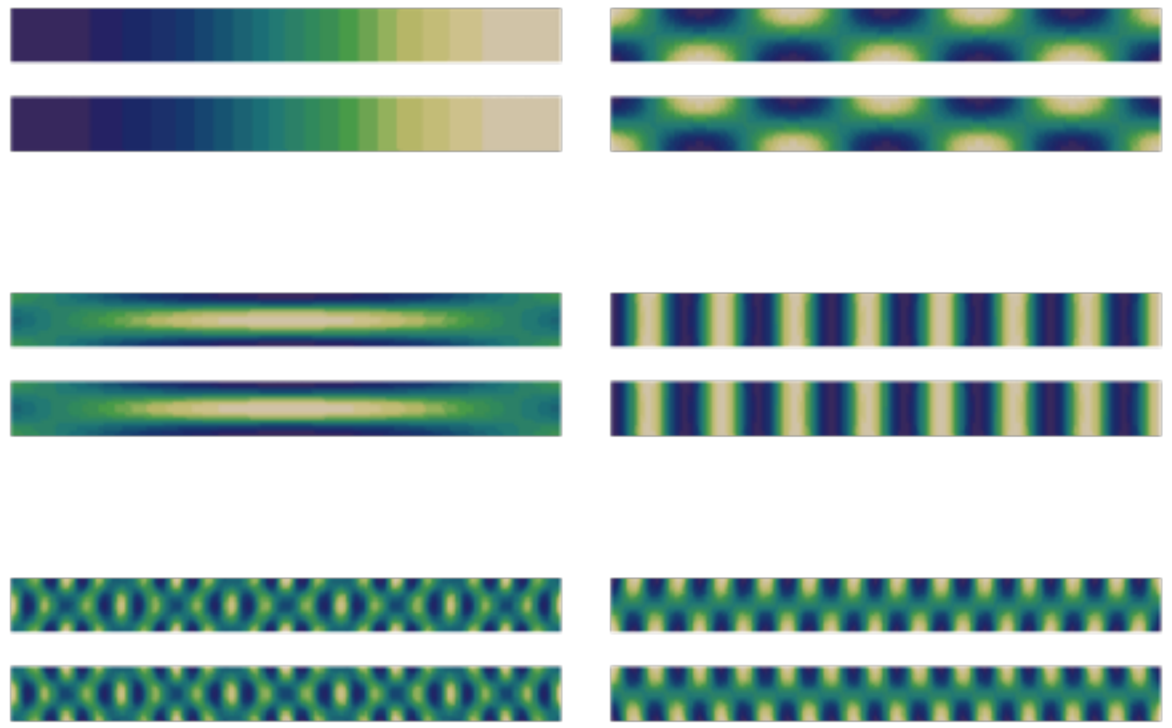


Sliding Basis Optimization

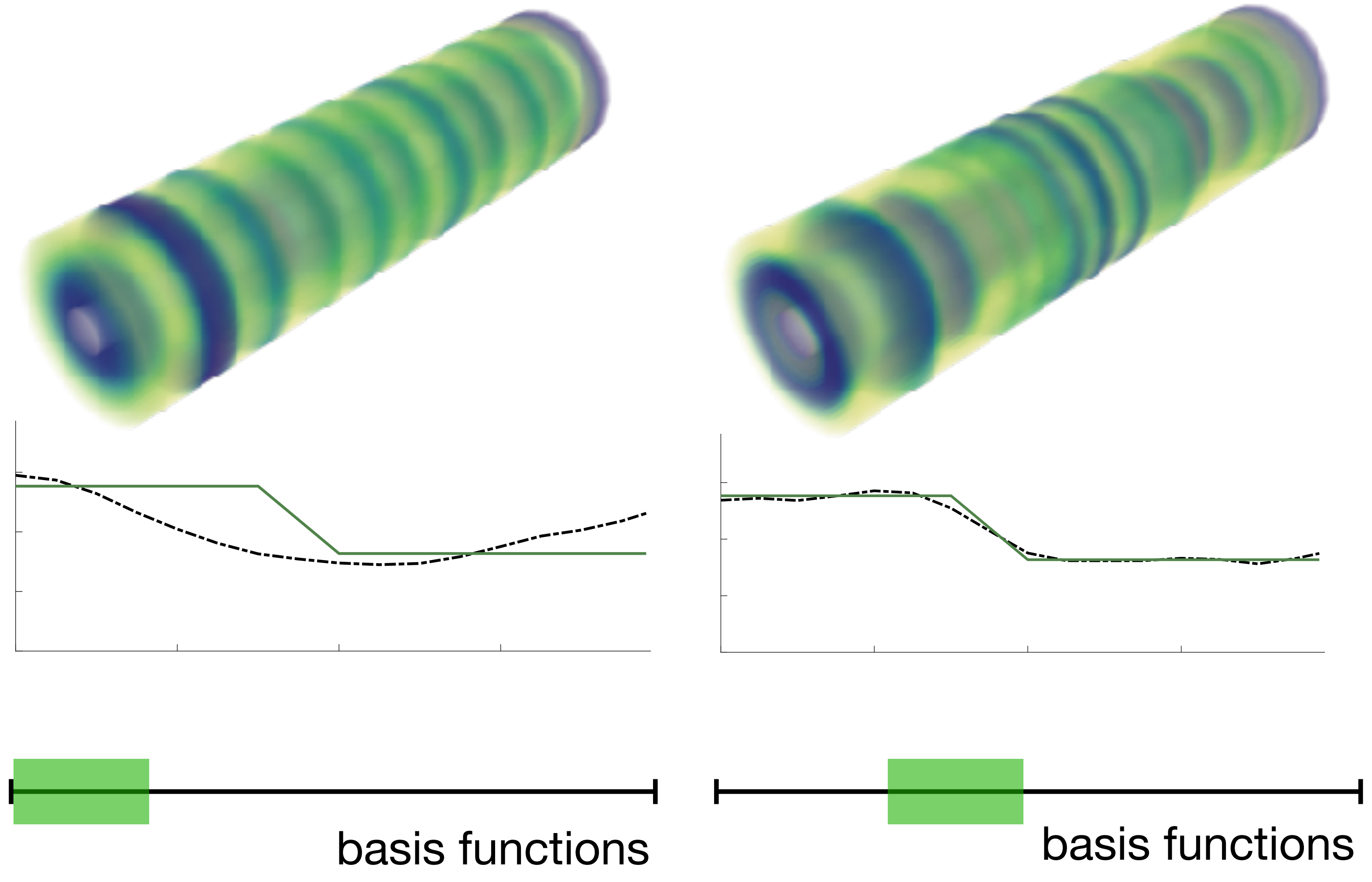
Inputs



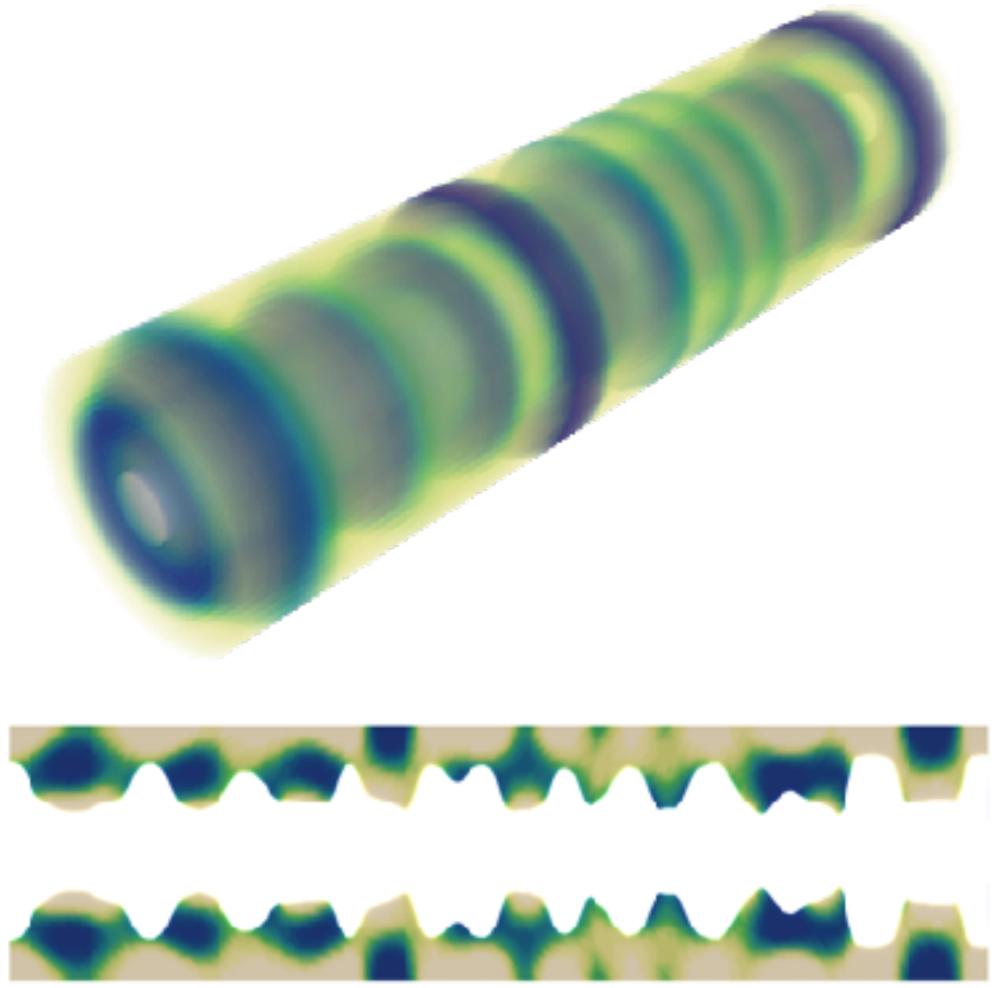
Compute Basis Functions



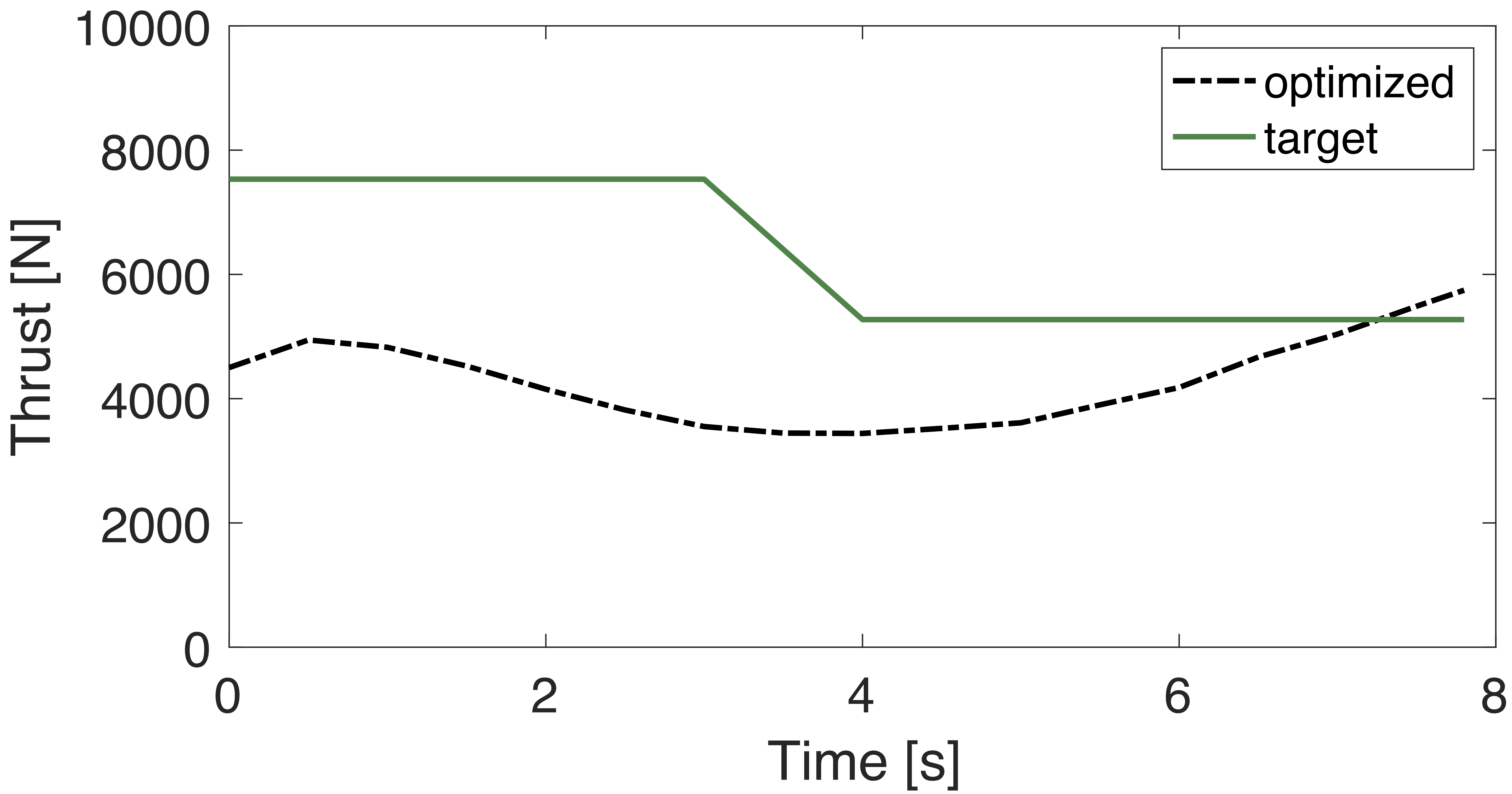
Optimization



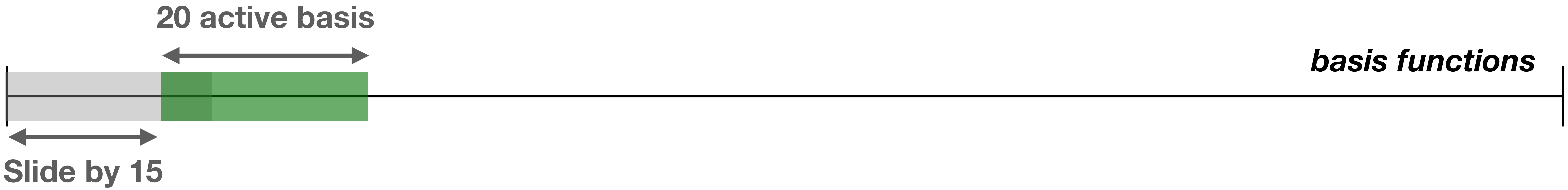
Output

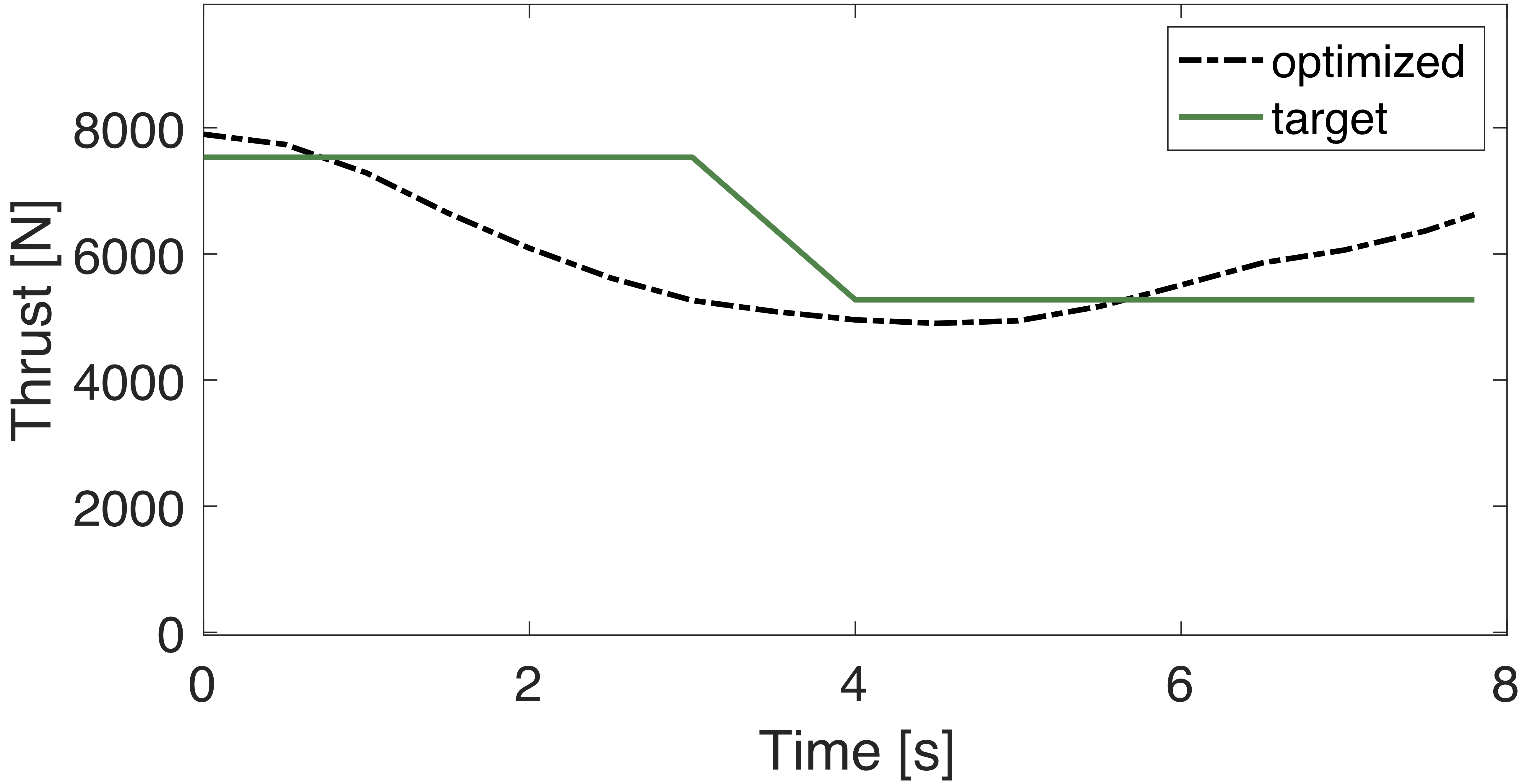


Progression of thrust profile match through the sliding basis optimization

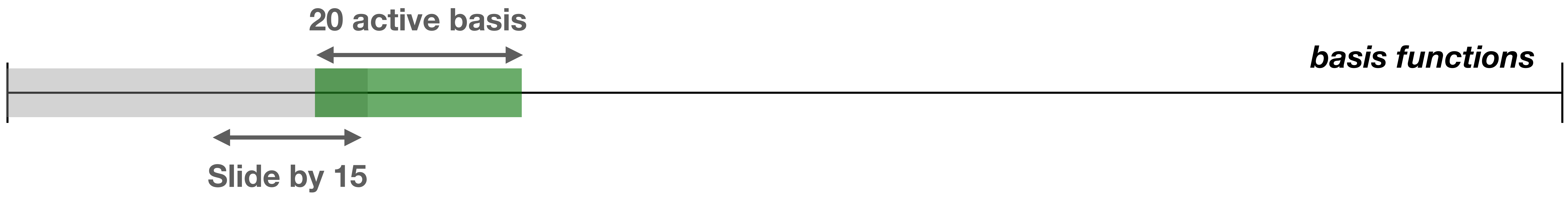


Progression of thrust profile match through the sliding basis optimization





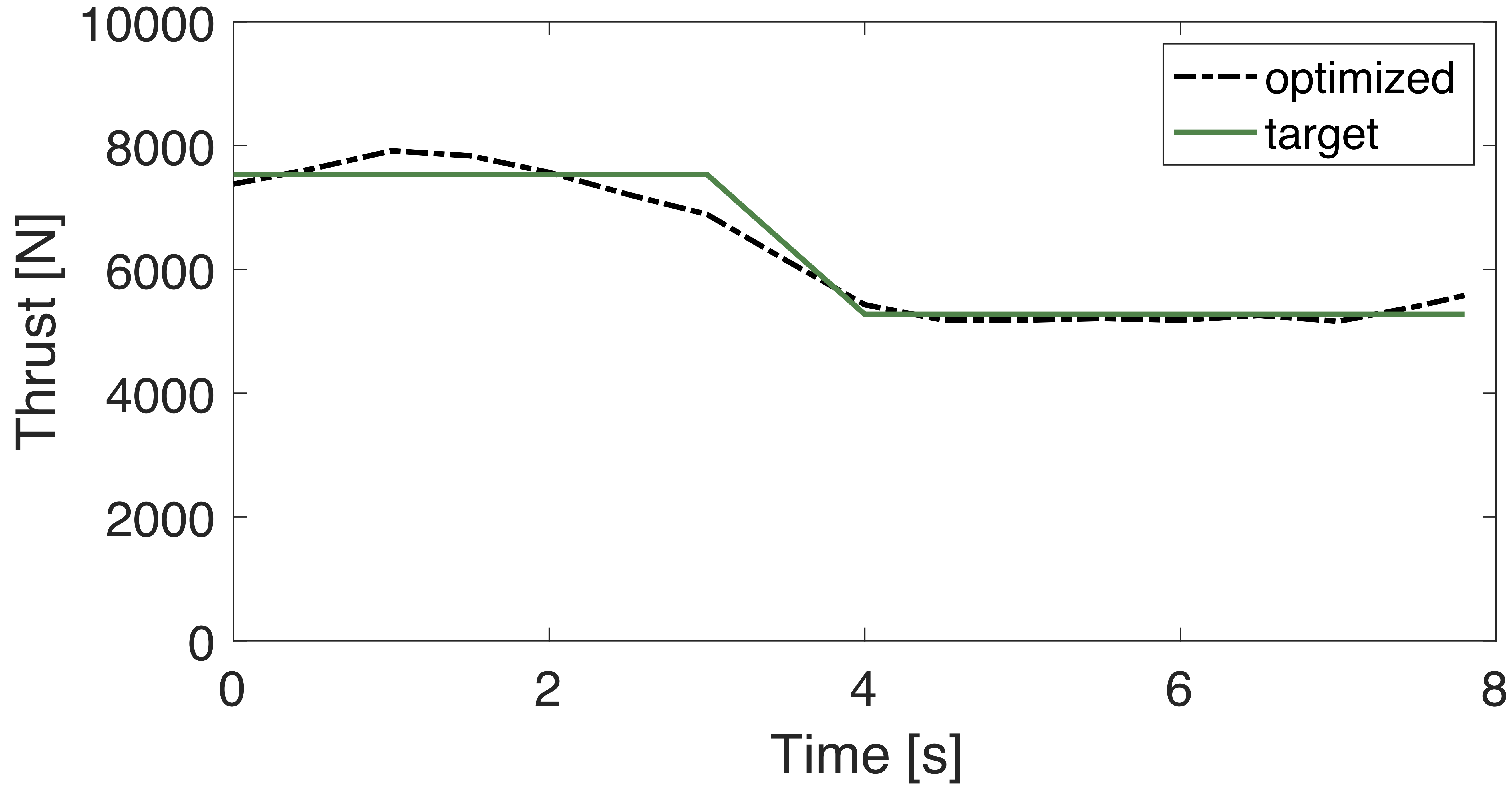
Progression of thrust profile match through the sliding basis optimization



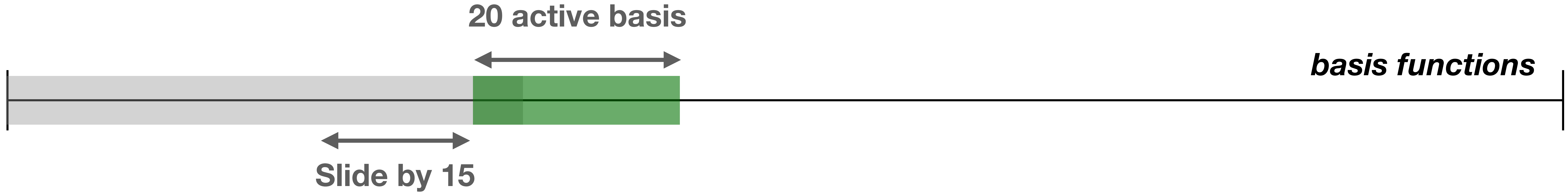
20 active basis

basis functions

Slide by 15



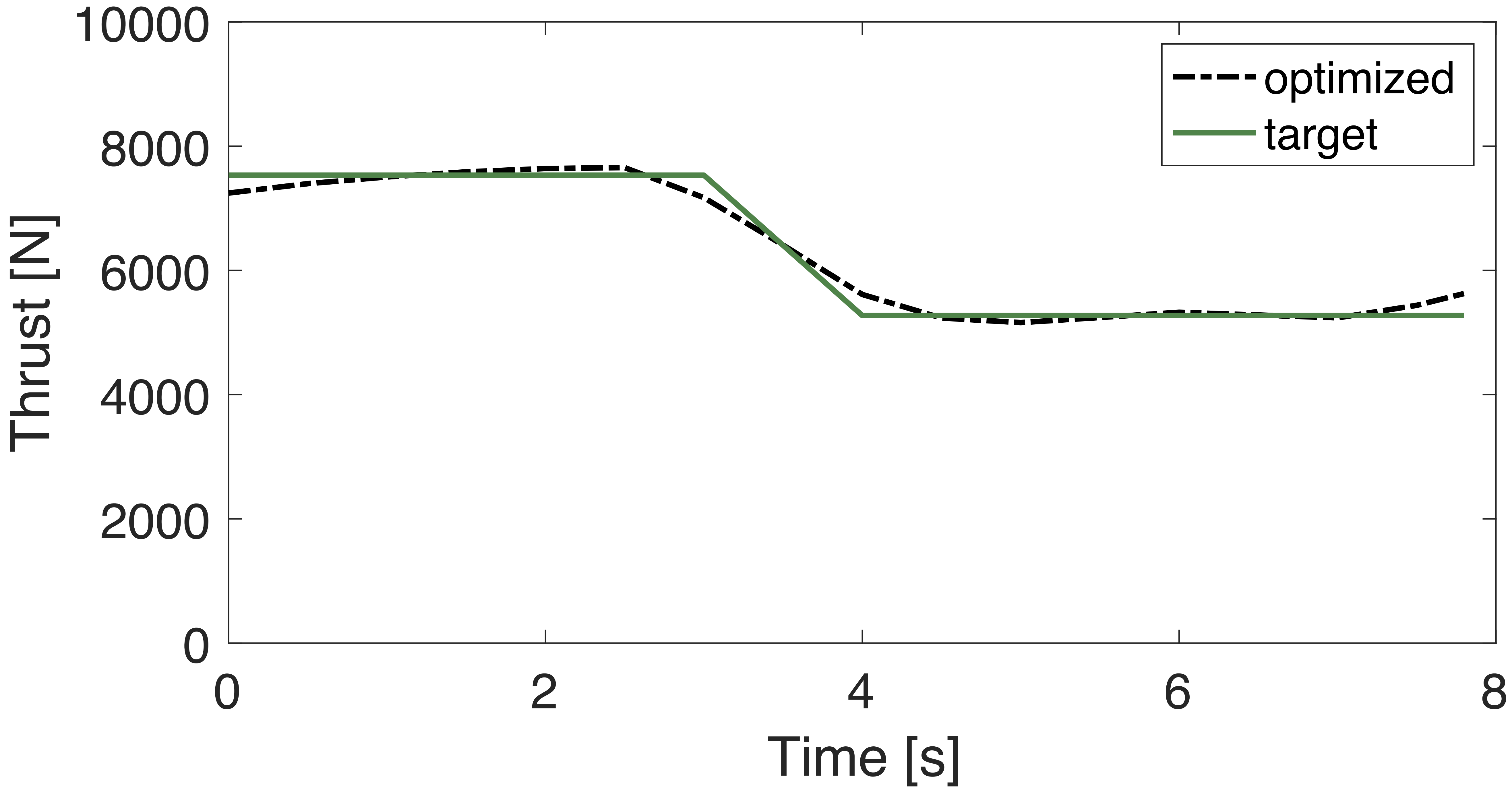
Progression of thrust profile match through the sliding basis optimization



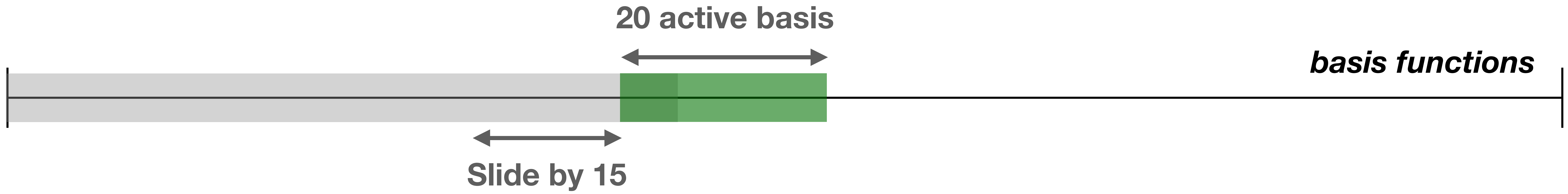
20 active basis

basis functions

Slide by 15



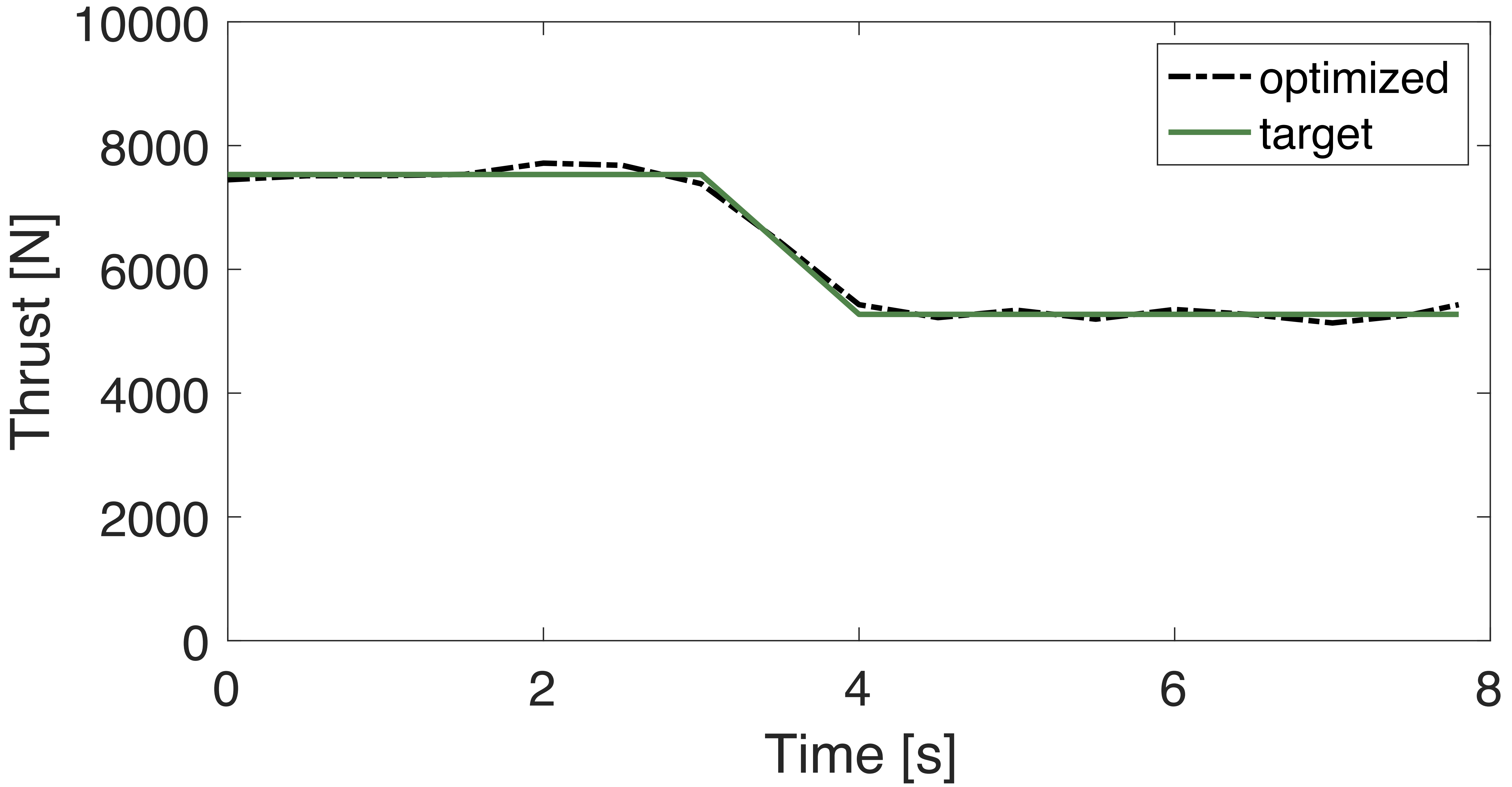
Progression of thrust profile match through the sliding basis optimization



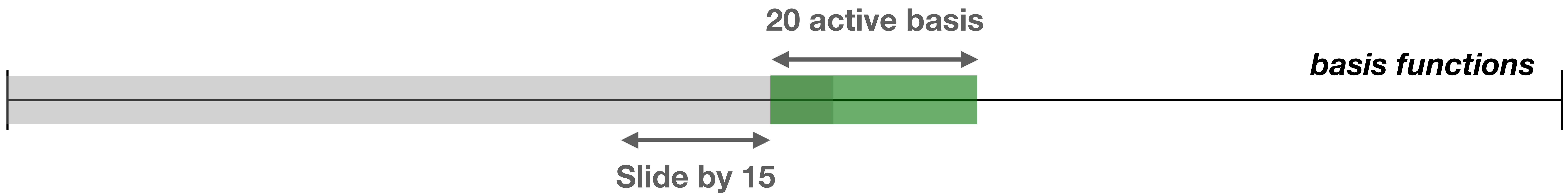
20 active basis

basis functions

Slide by 15



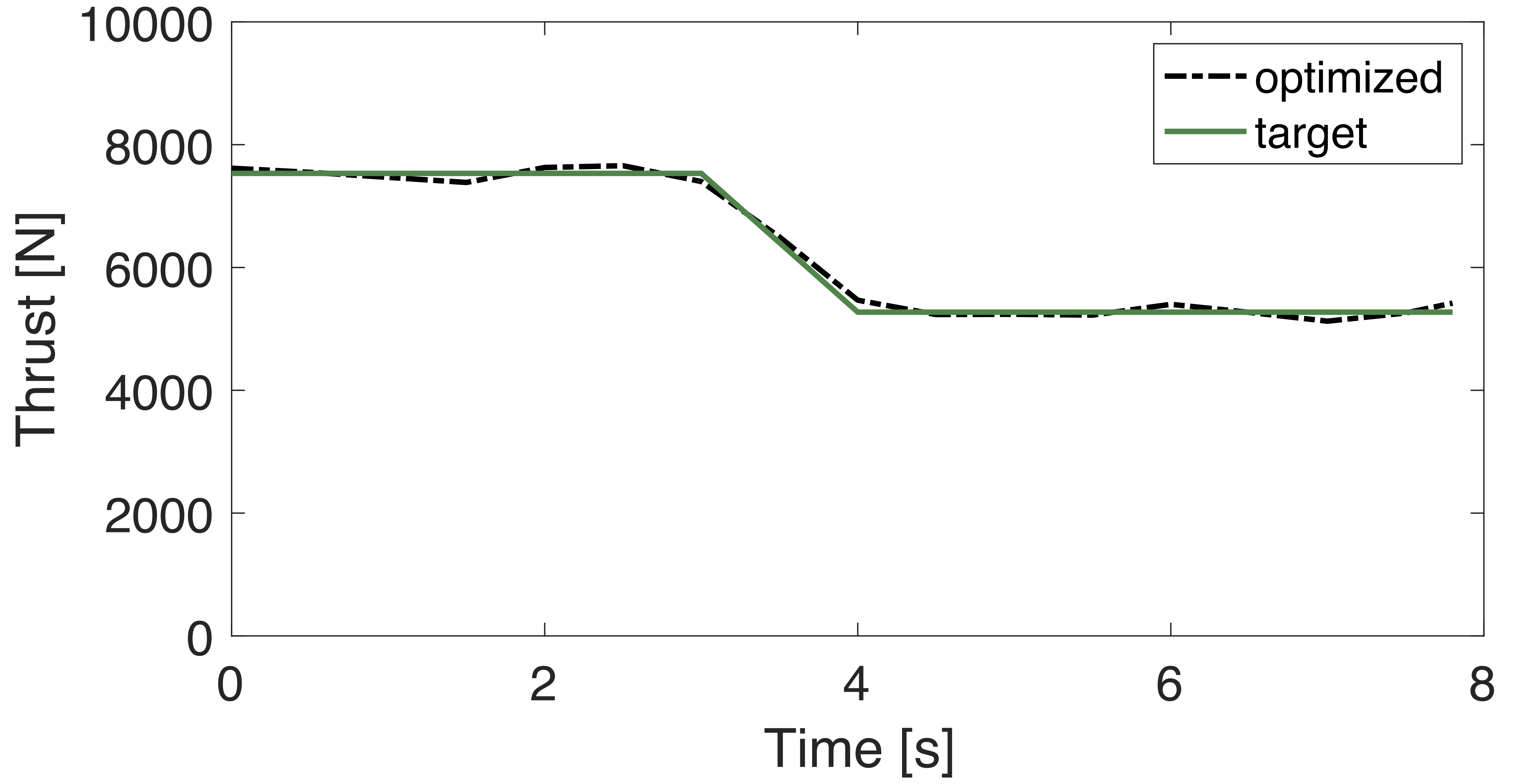
Progression of thrust profile match through the sliding basis optimization



20 active basis

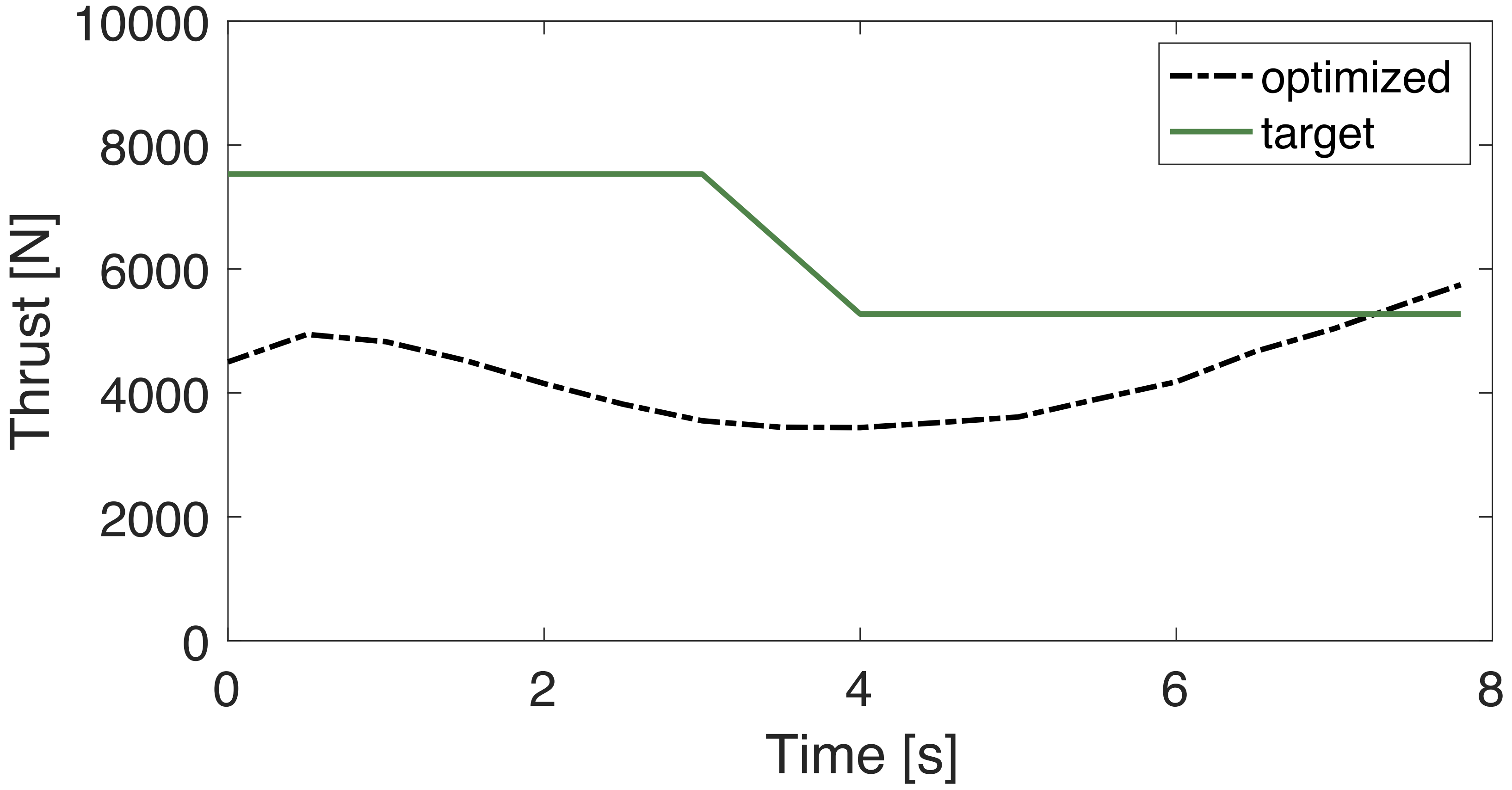
basis functions

Slide by 15

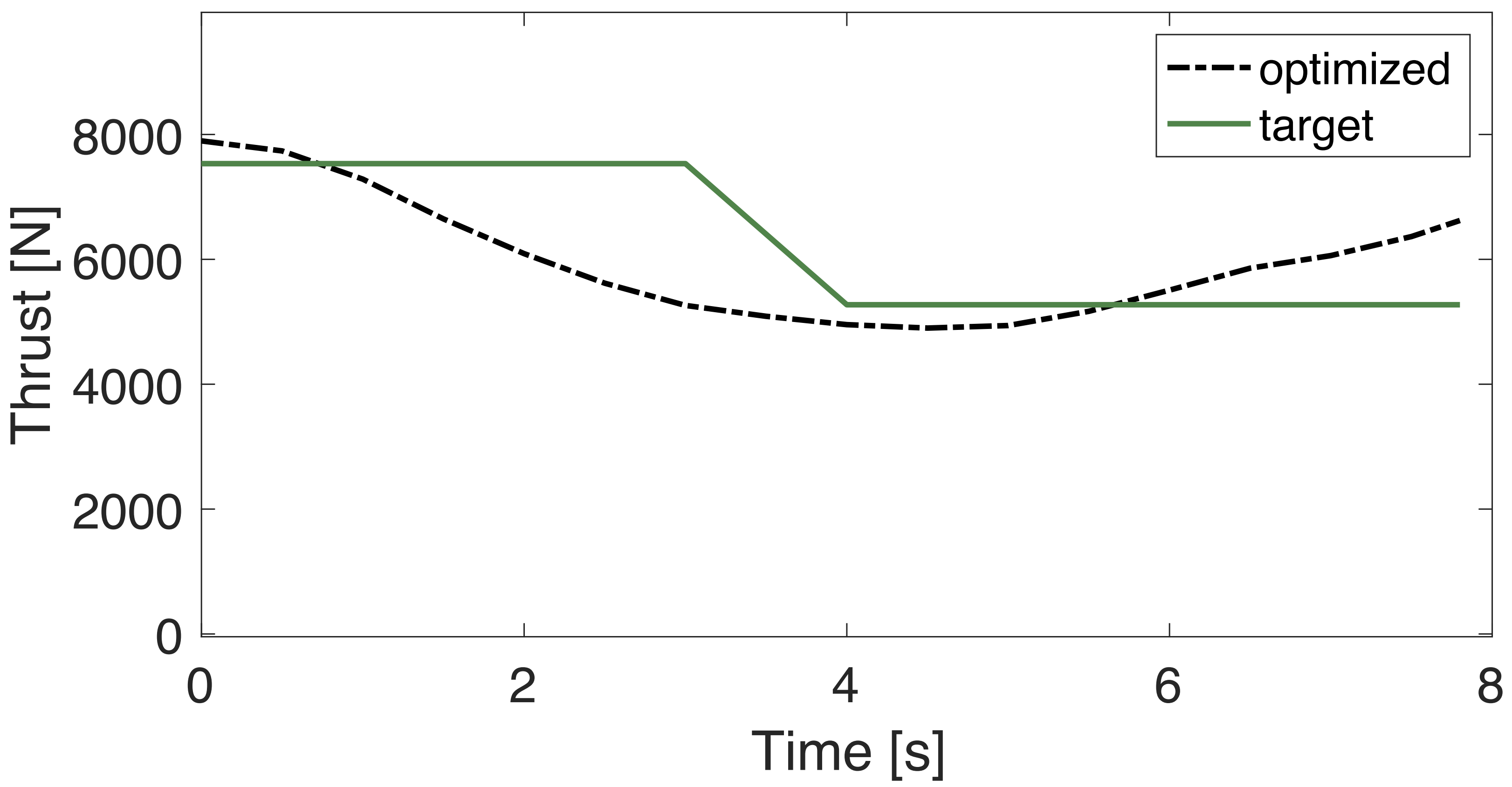


Progression of thrust profile match through the sliding basis optimization

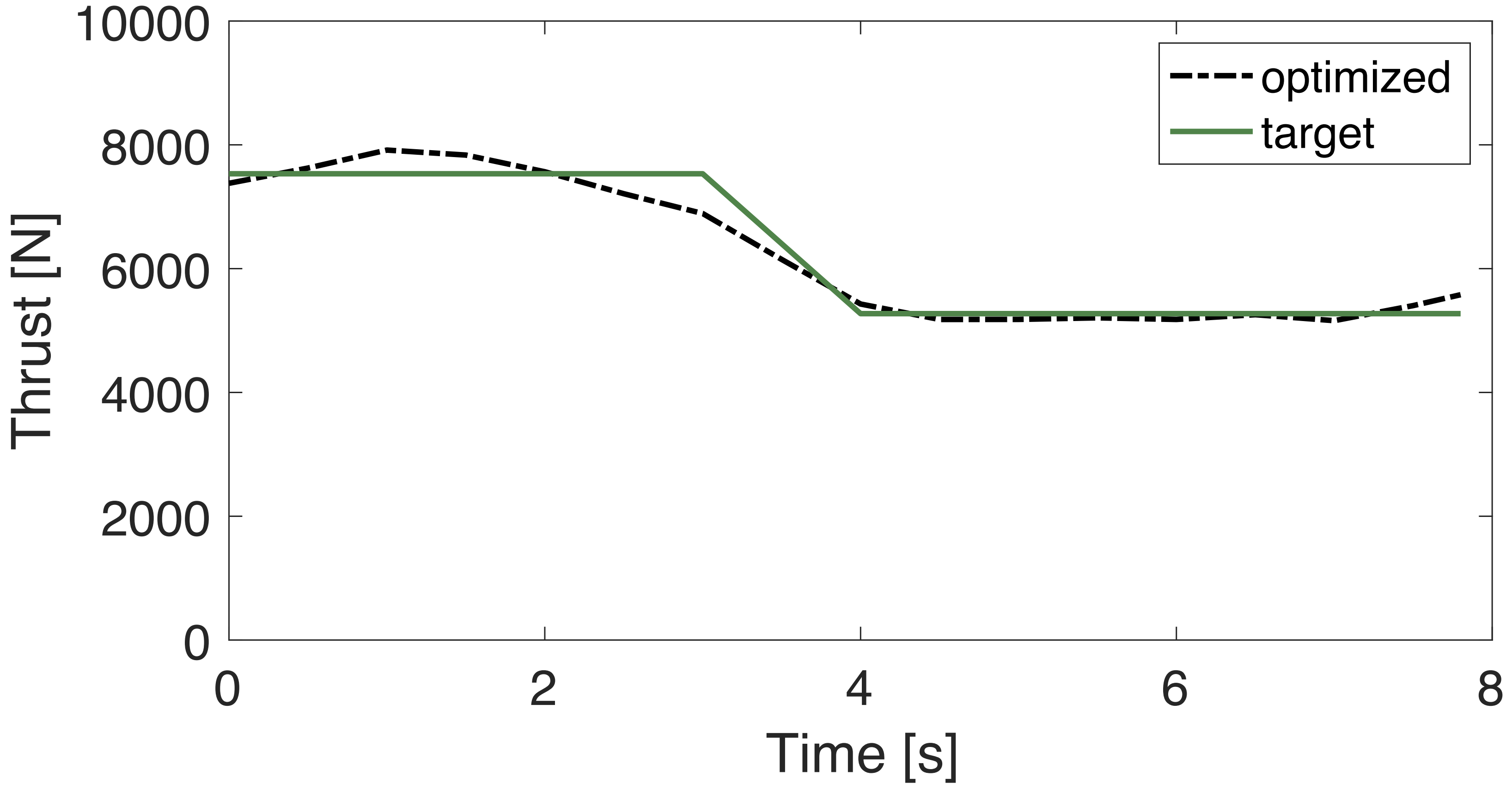
First 20 basis



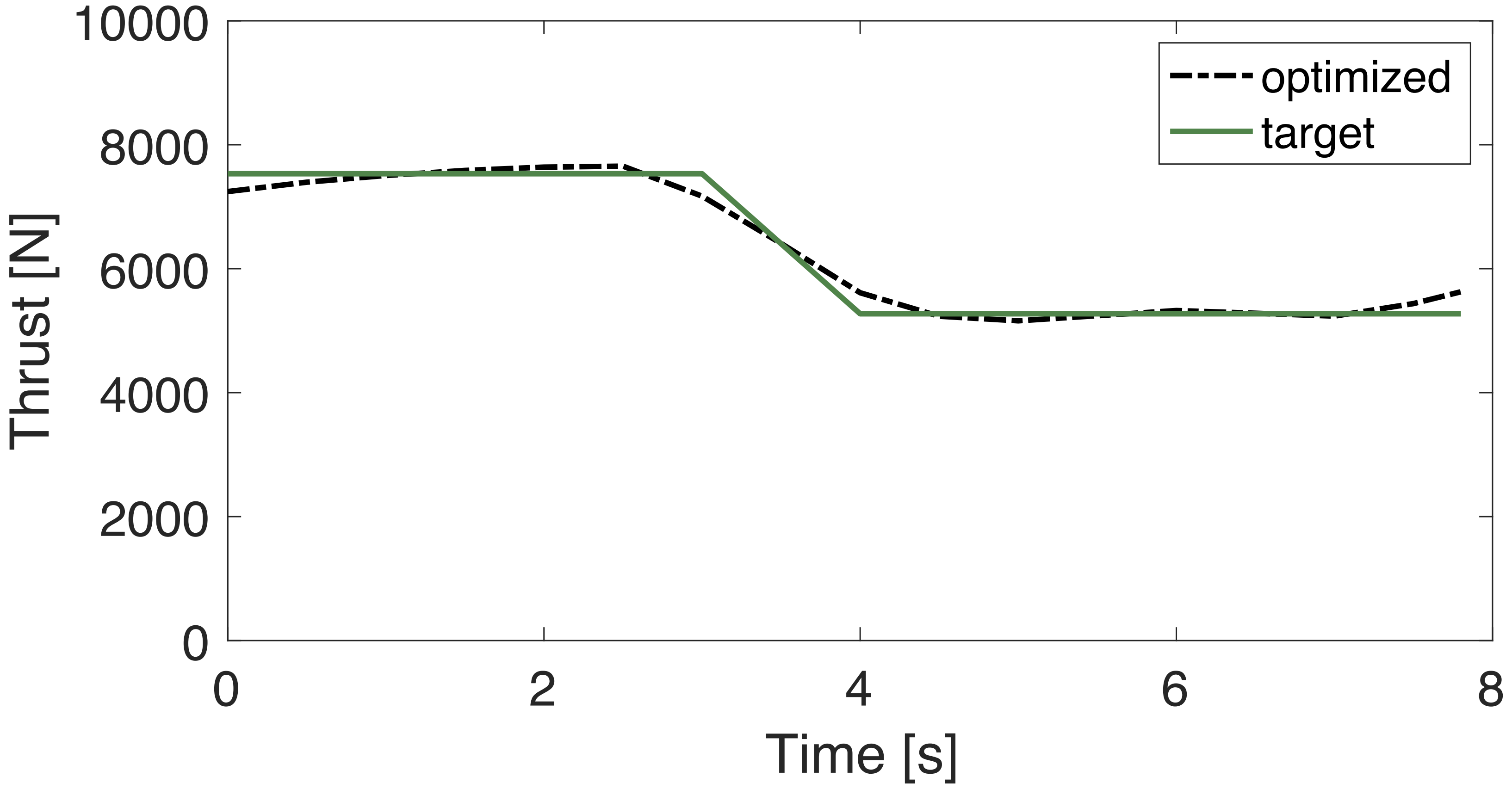
Slide by 15



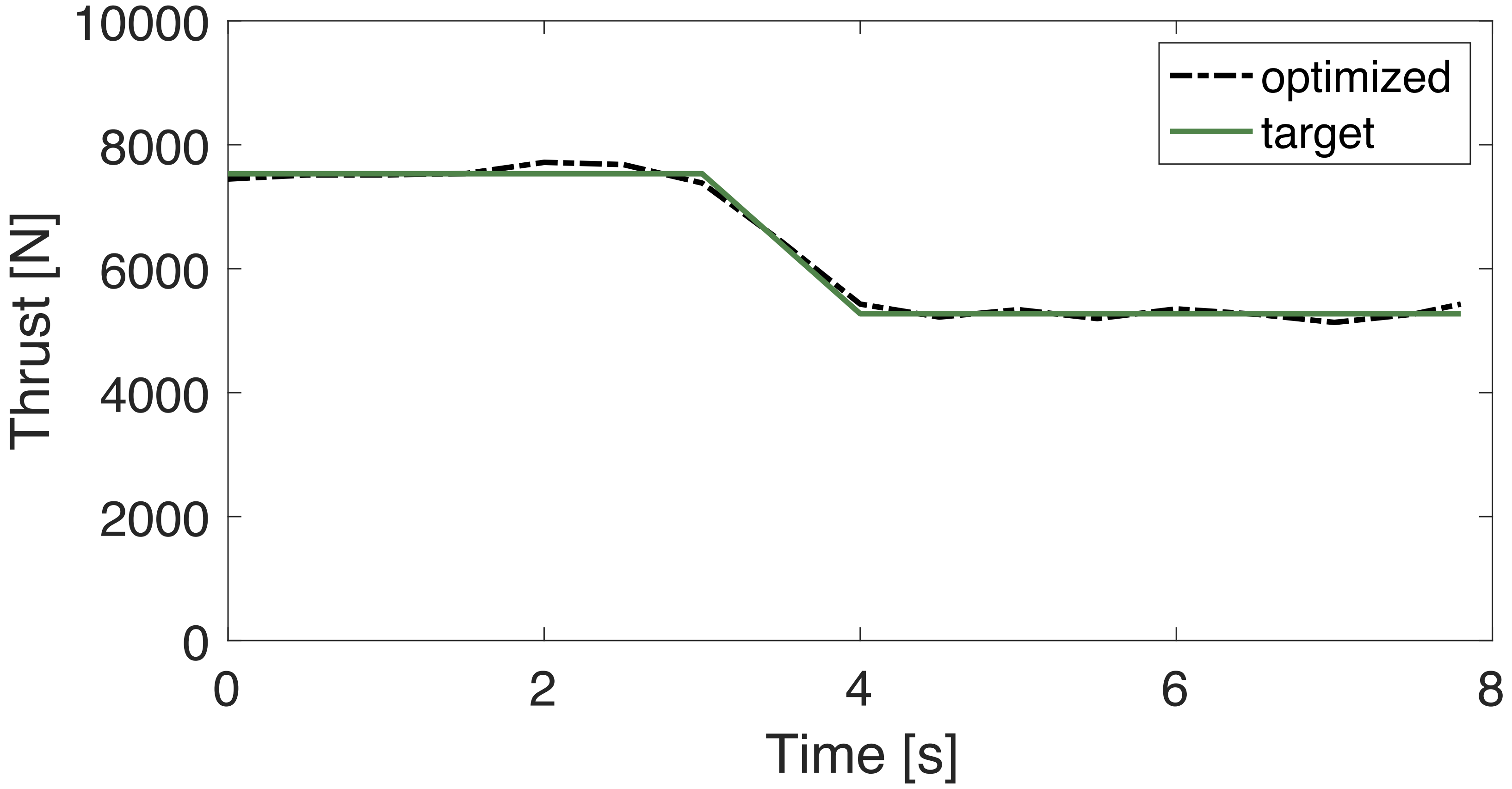
Slide by 15



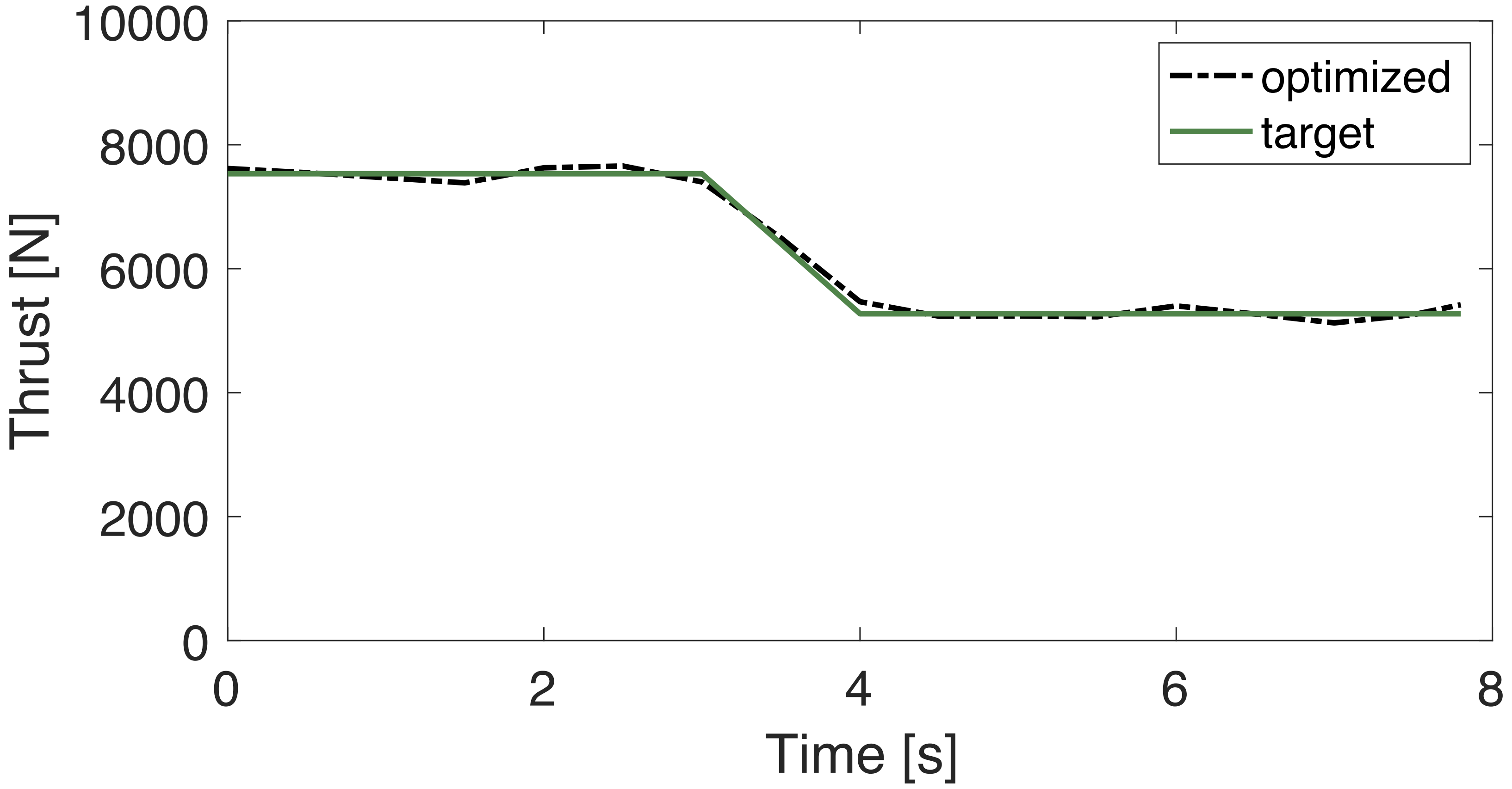
Slide by 15



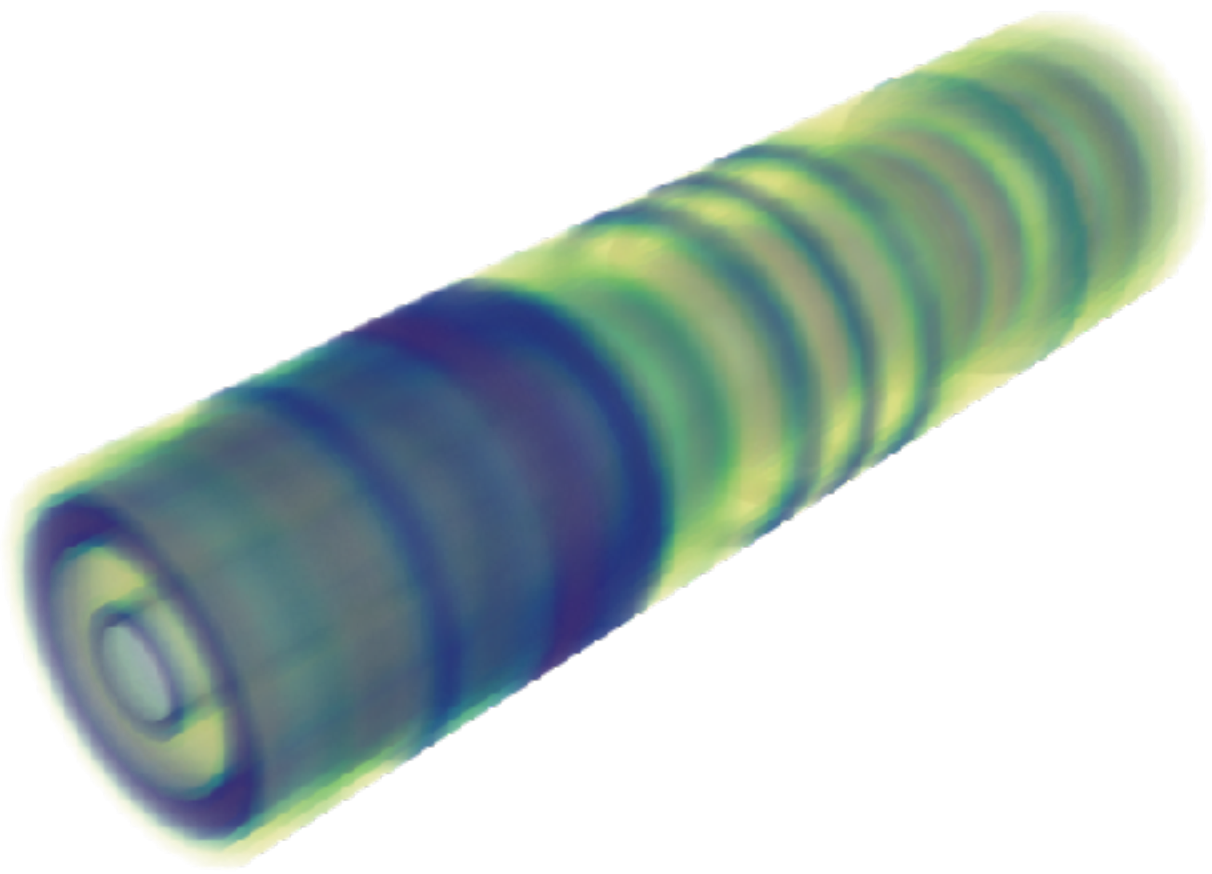
Slide by 15

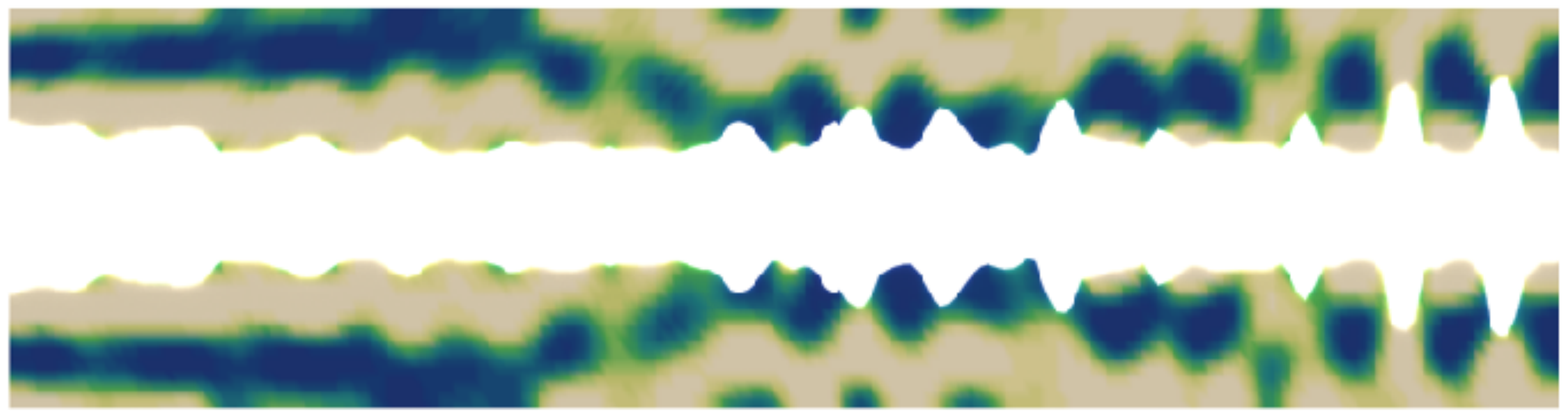


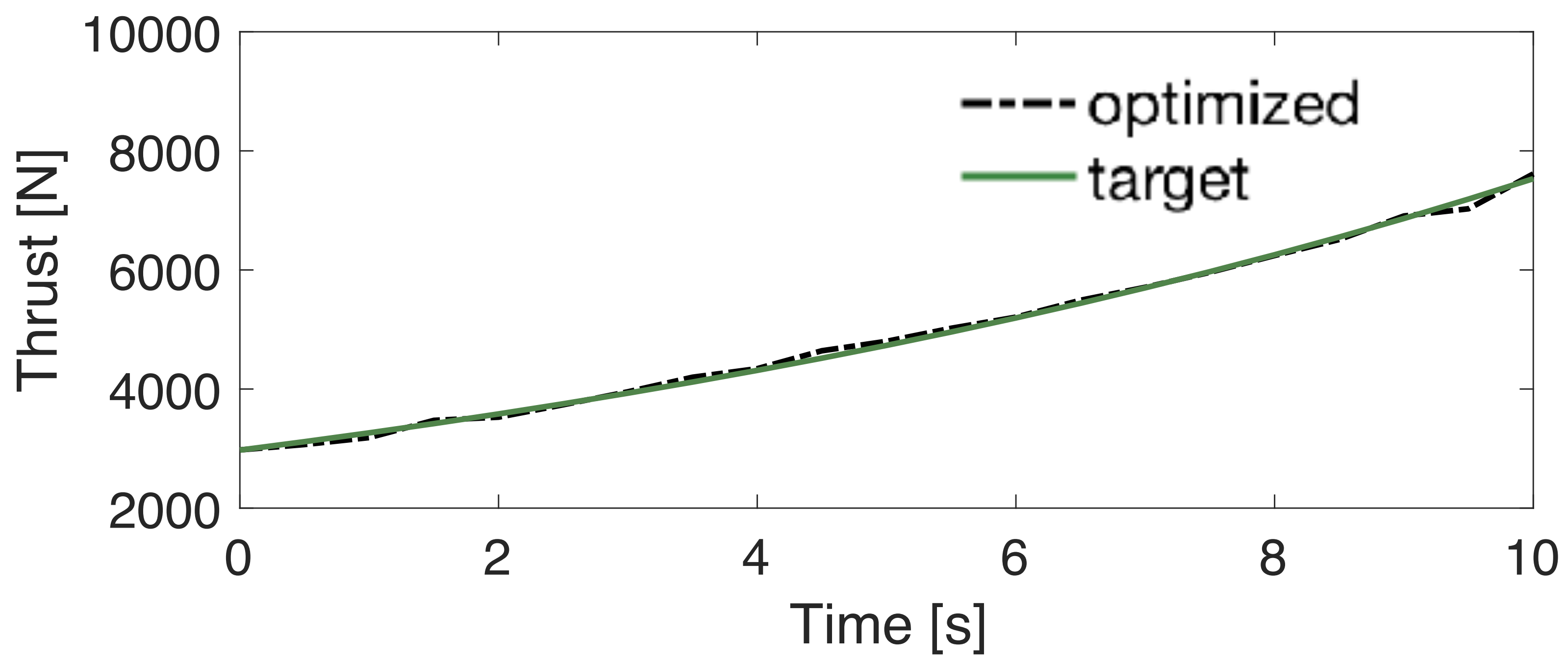
Slide by 15

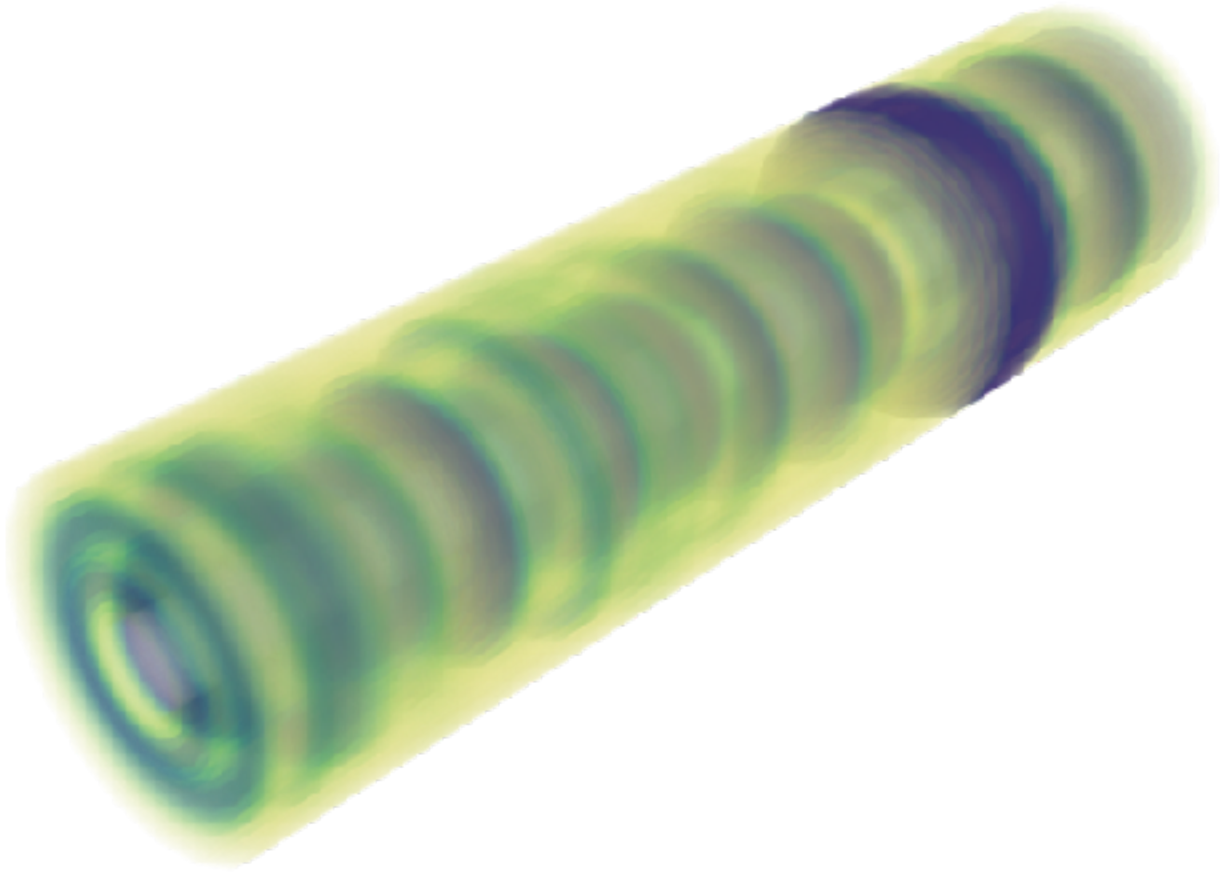


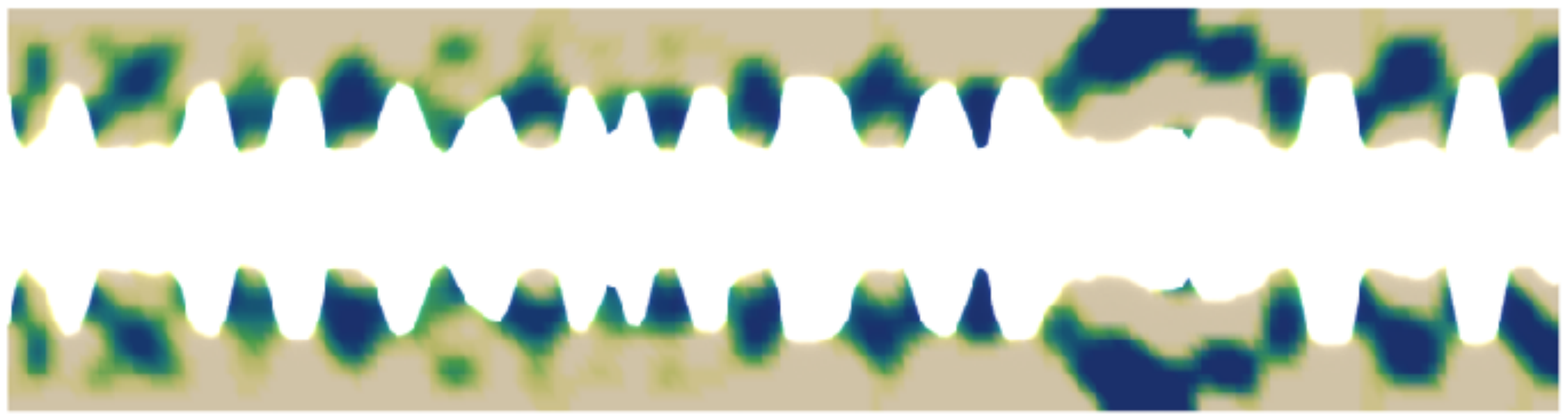
Optimized Solid Rocket Fuel Designs

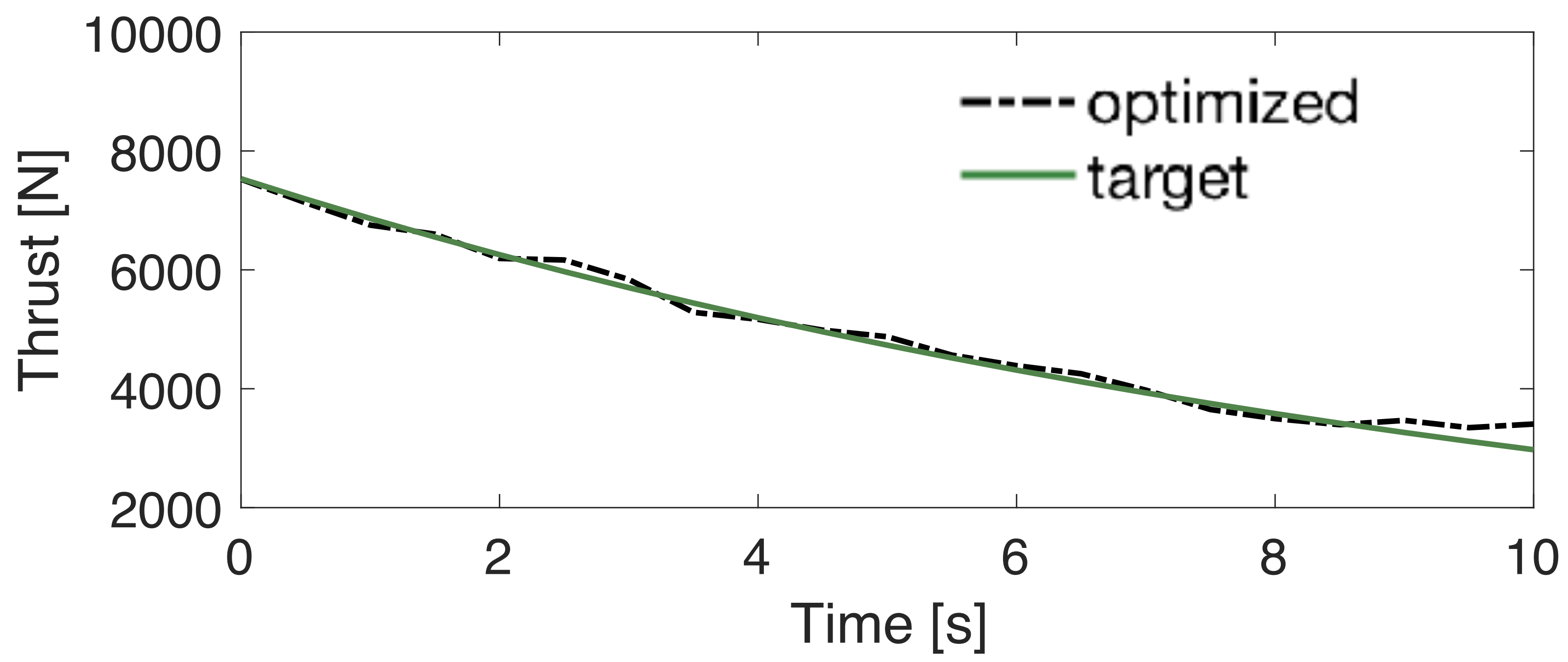




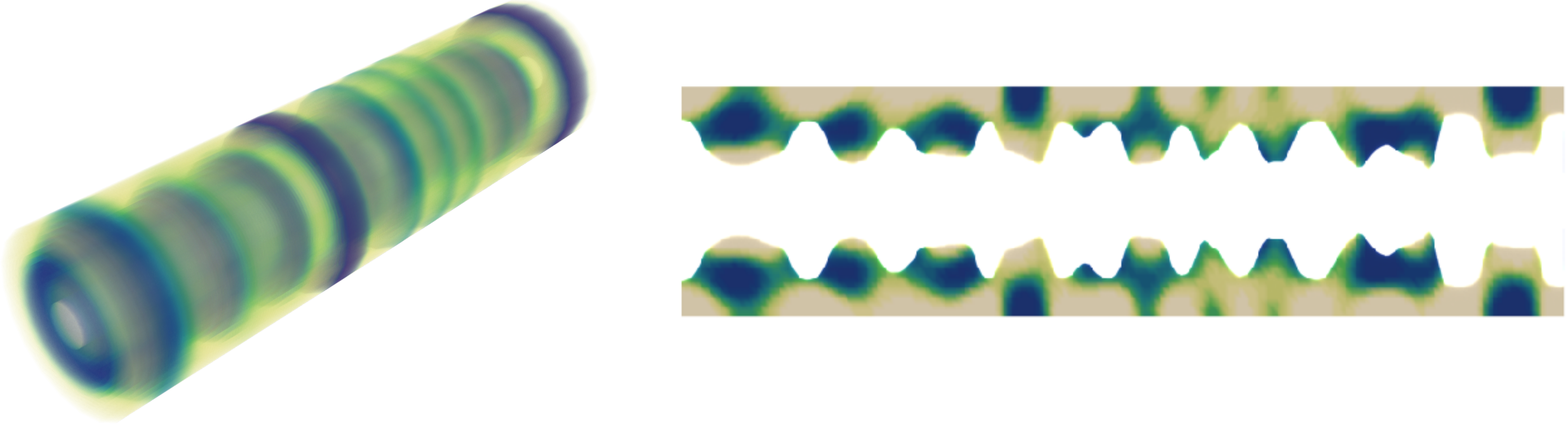


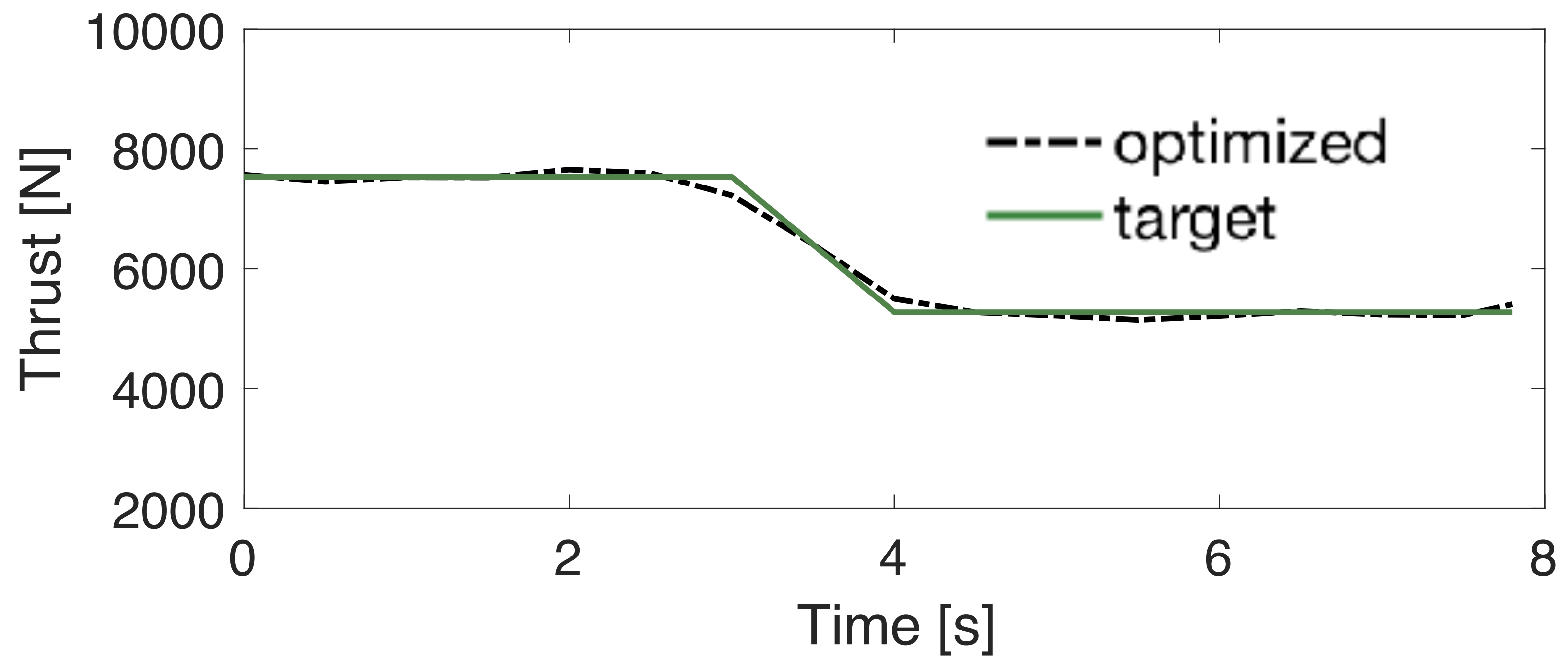


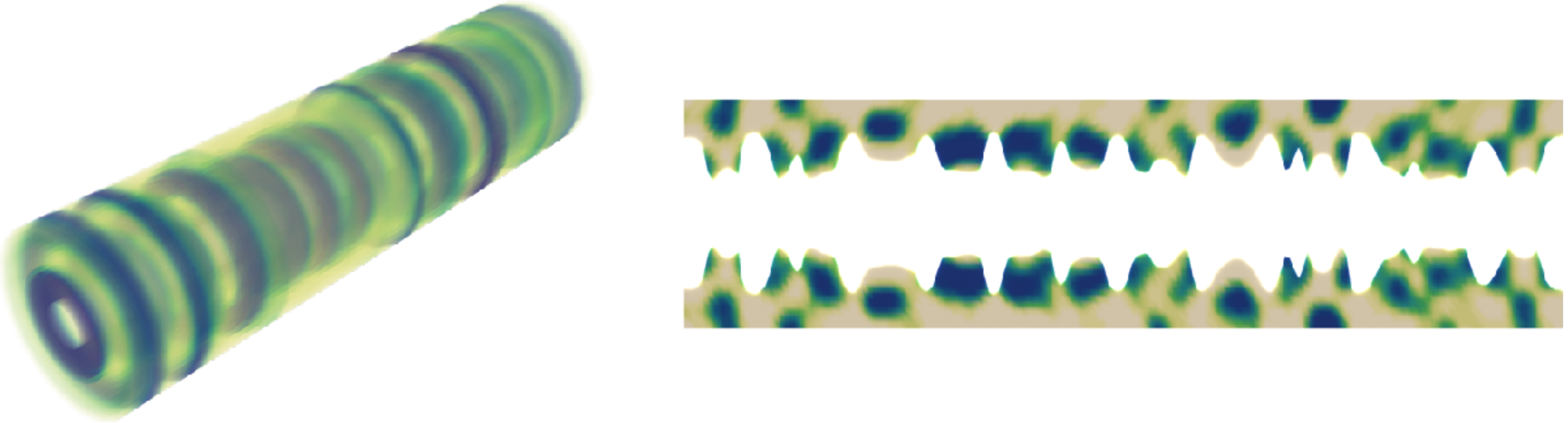


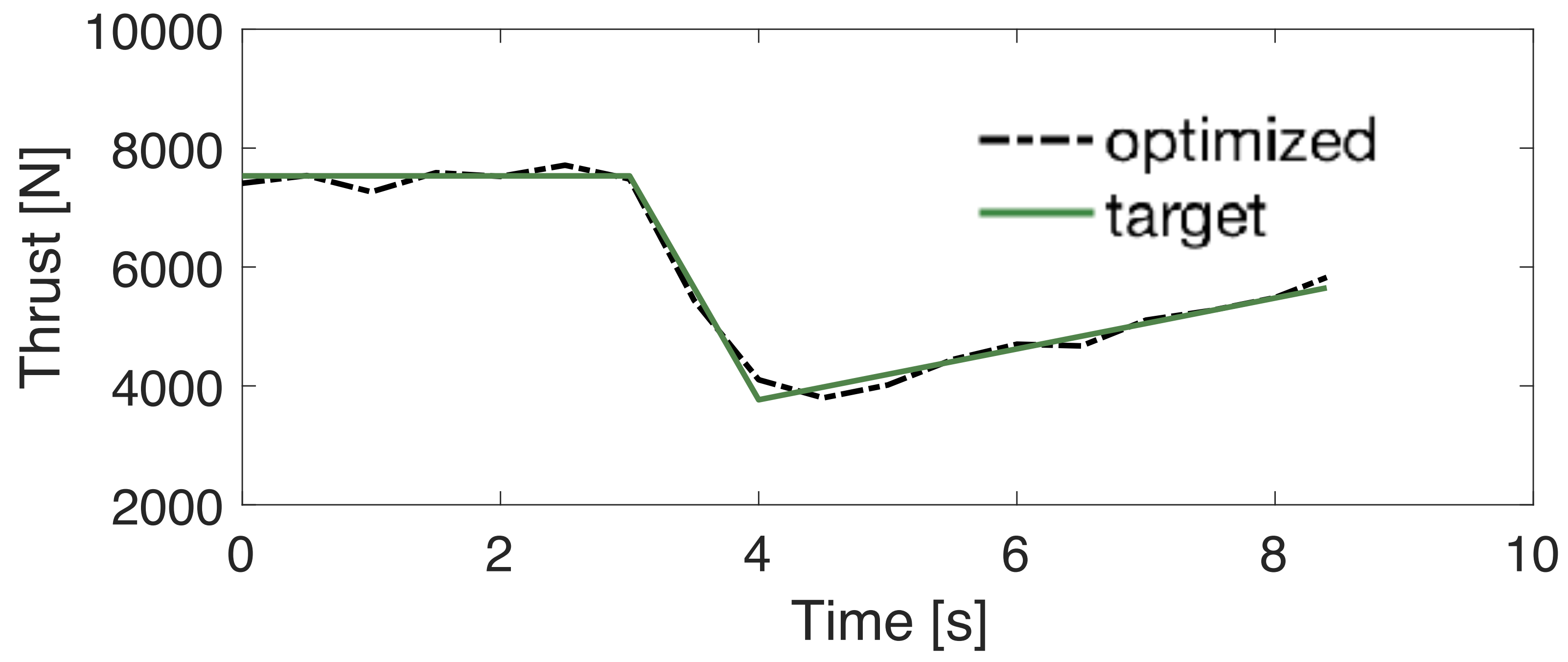


Optimized Solid Rocket Fuel Designs



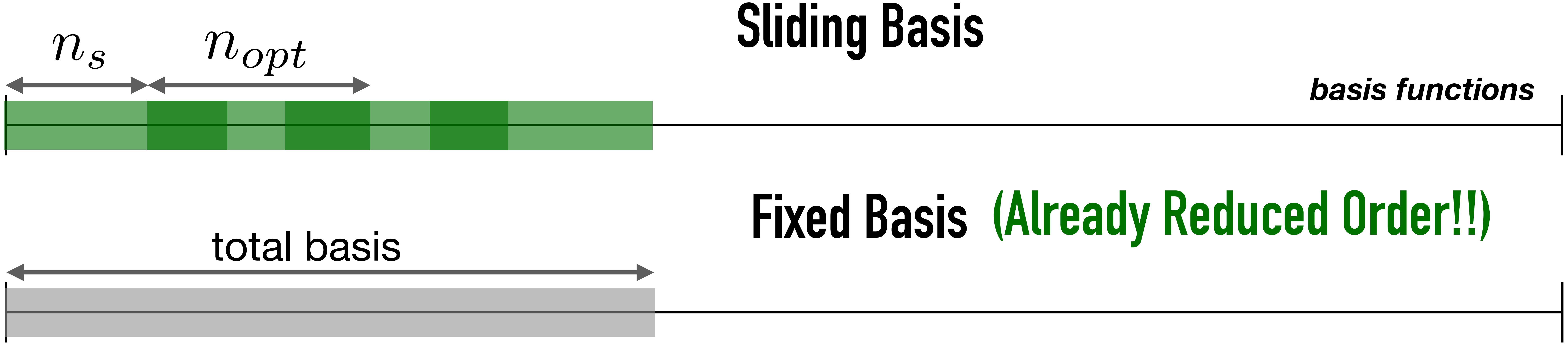






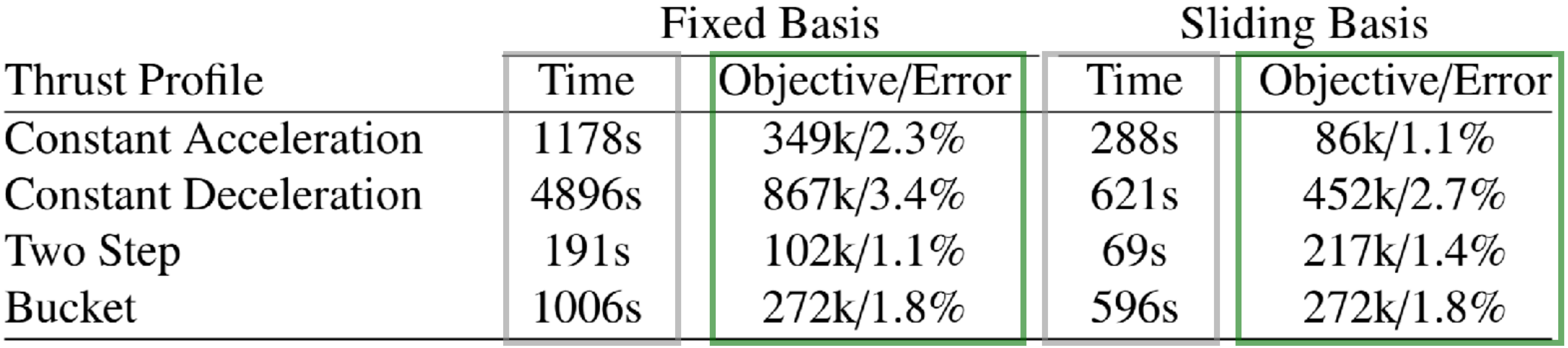
Graded material fields!

Performance



Sliding Basis

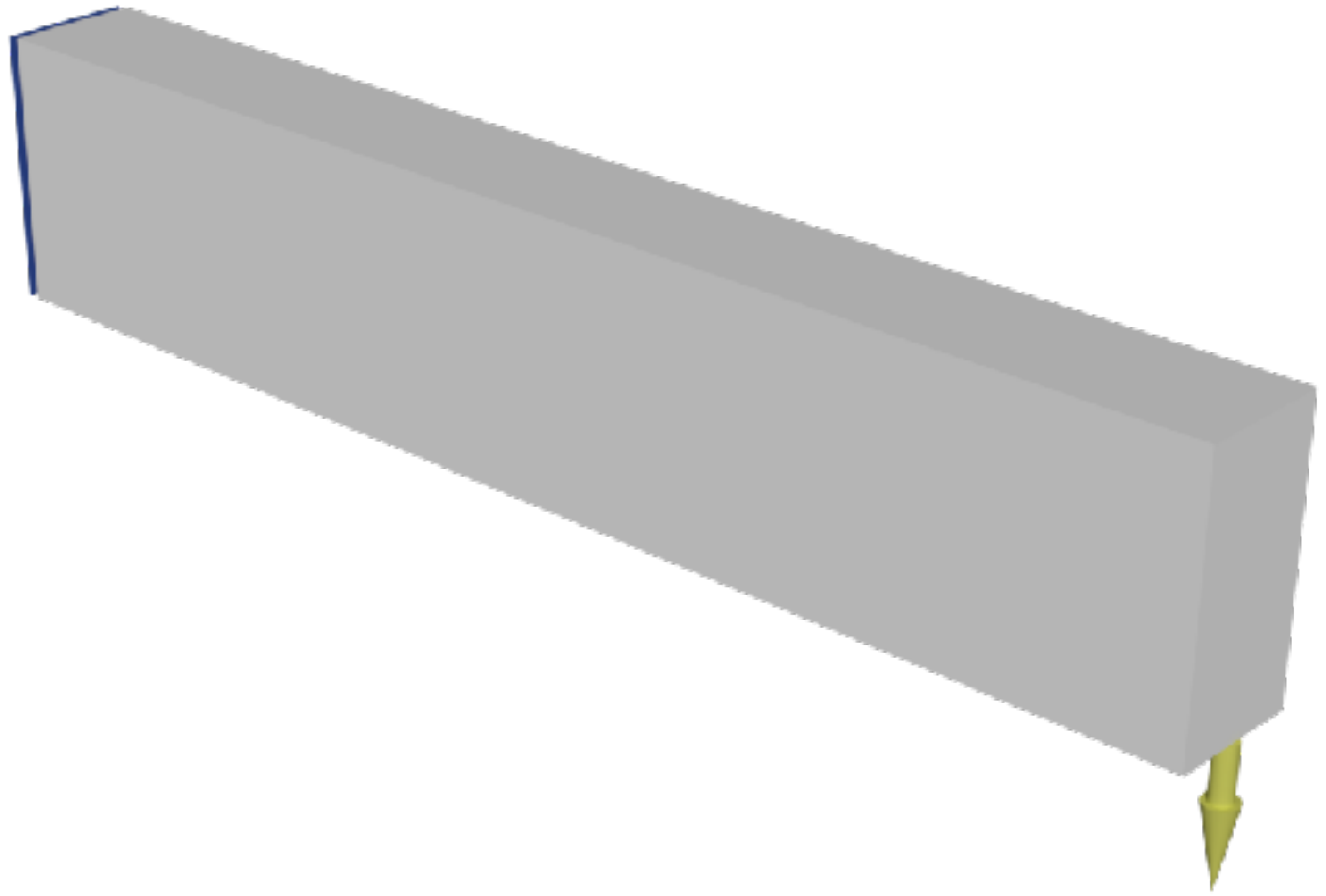
Fixed Basis (Already Reduced Order!!)



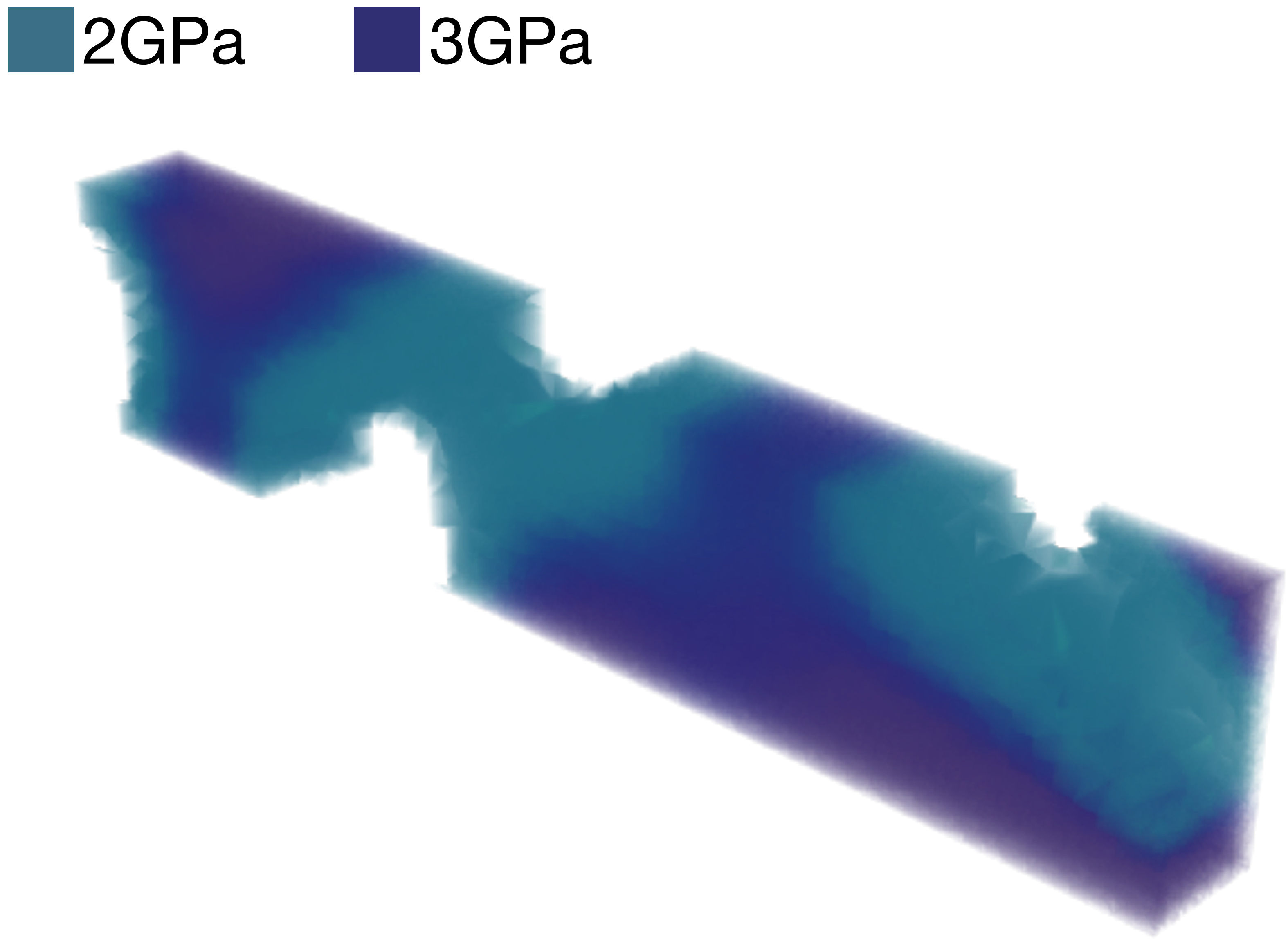
	Fixed Basis		Sliding Basis	
Thrust Profile	Time	Objective/Error	Time	Objective/Error
Constant Acceleration	1178s	349k/2.3%	288s	86k/1.1%
Constant Deceleration	4896s	867k/3.4%	621s	452k/2.7%
Two Step	191s	102k/1.1%	69s	217k/1.4%
Bucket	1006s	272k/1.8%	596s	272k/1.8%

Upto 8x speed up **Comparable objective values**

Multi-material Topology Optimization Through Sliding Optimization Steps



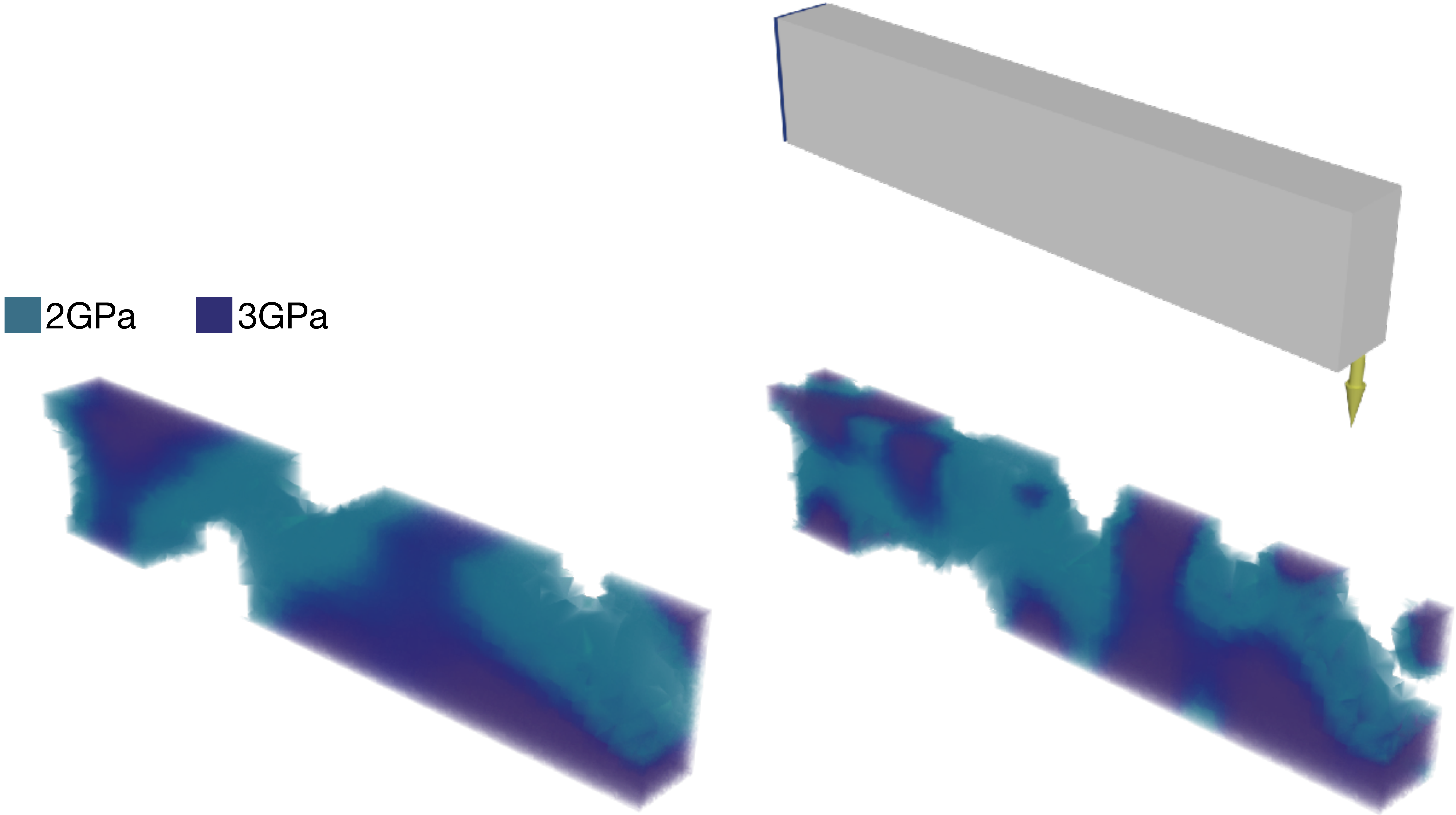
$$
\begin{aligned}
\min_{\boldsymbol{w}} \quad & \boldsymbol{u}^T \boldsymbol{K}(\boldsymbol{w})\boldsymbol{u} \\
\text{s.t.} \quad & m(\boldsymbol{w})/m_0 \leq m_{frac} \\
& \boldsymbol{K}(\boldsymbol{w})\boldsymbol{u} = \boldsymbol{F}
\end{aligned}
$$

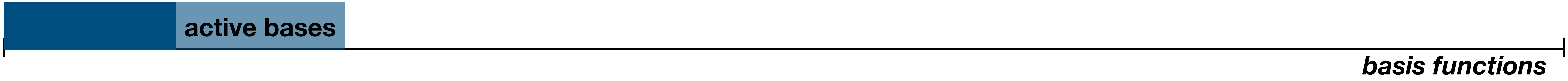


active bases

basis functions

Multi-material Topology Optimization Through Sliding Optimization Steps

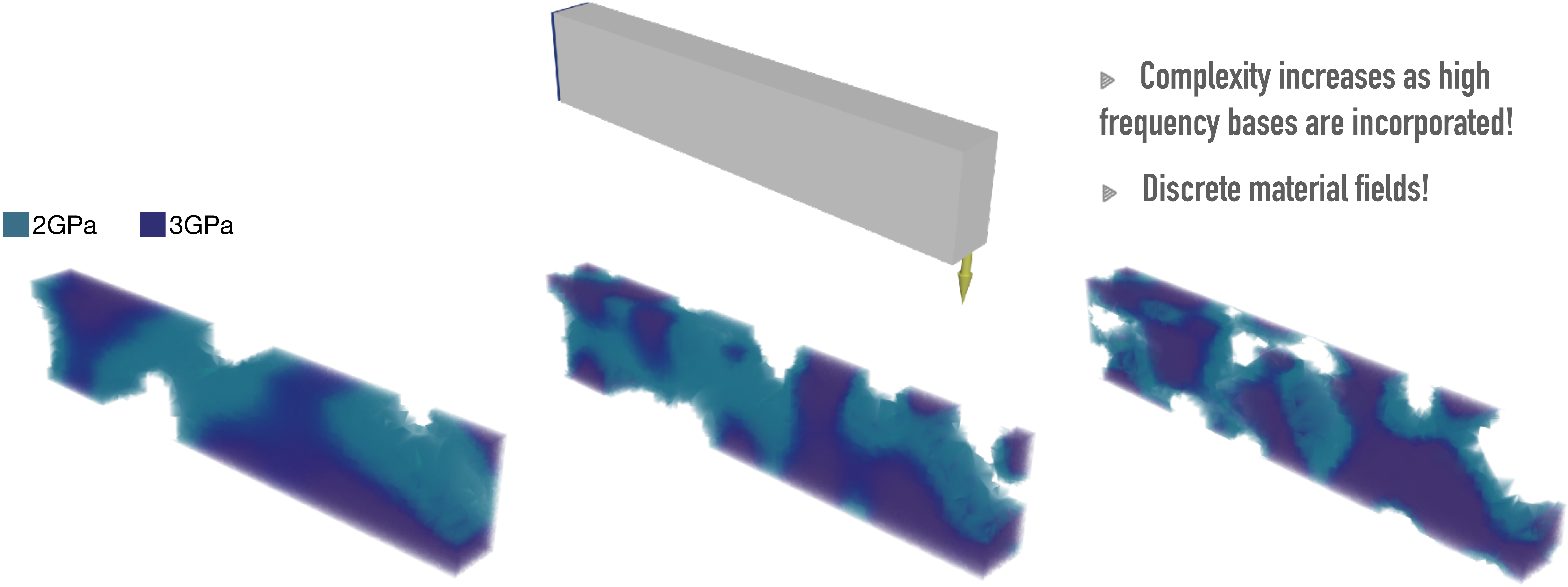




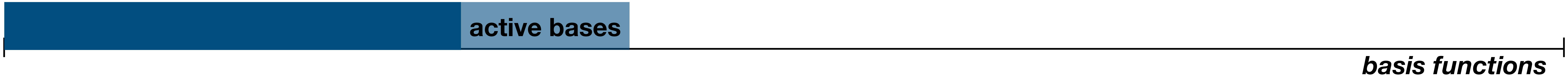
active bases

basis functions

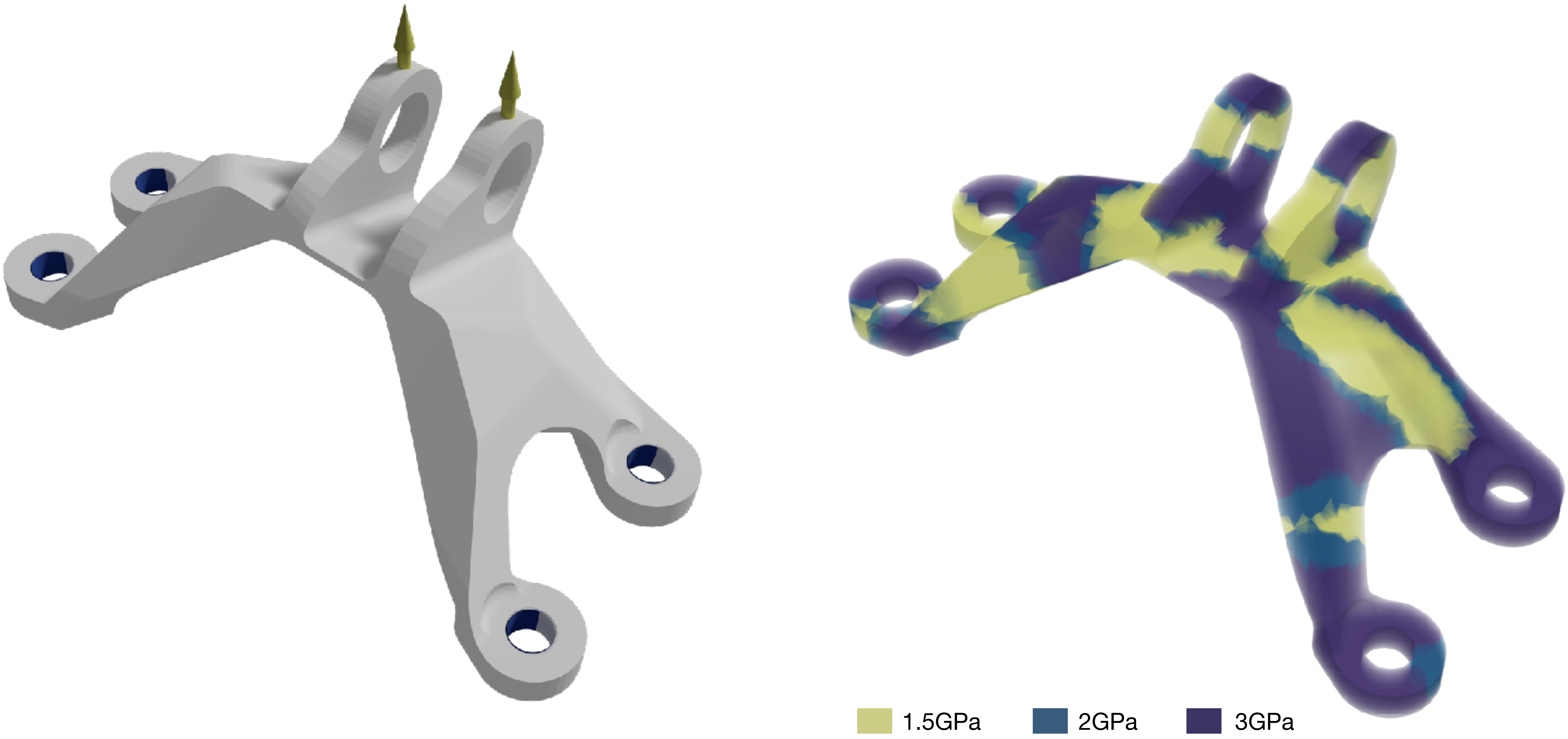
Multi-material Topology Optimization Through Sliding Optimization Steps



- Complexity increases as high frequency bases are incorporated!
- Discrete material fields!

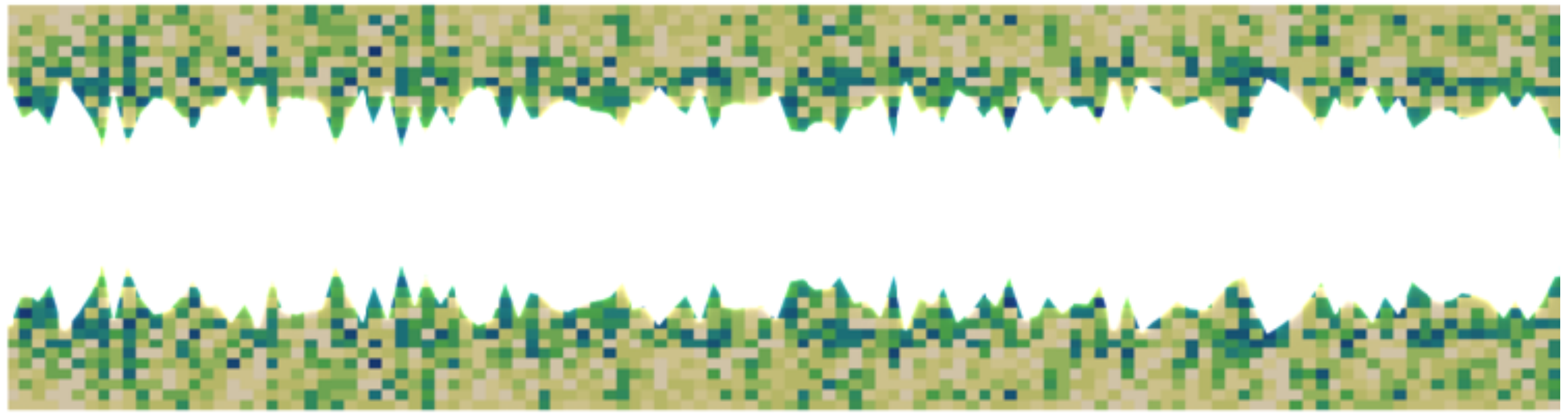


Material Distribution Optimization



Conventional VS Reduced Order

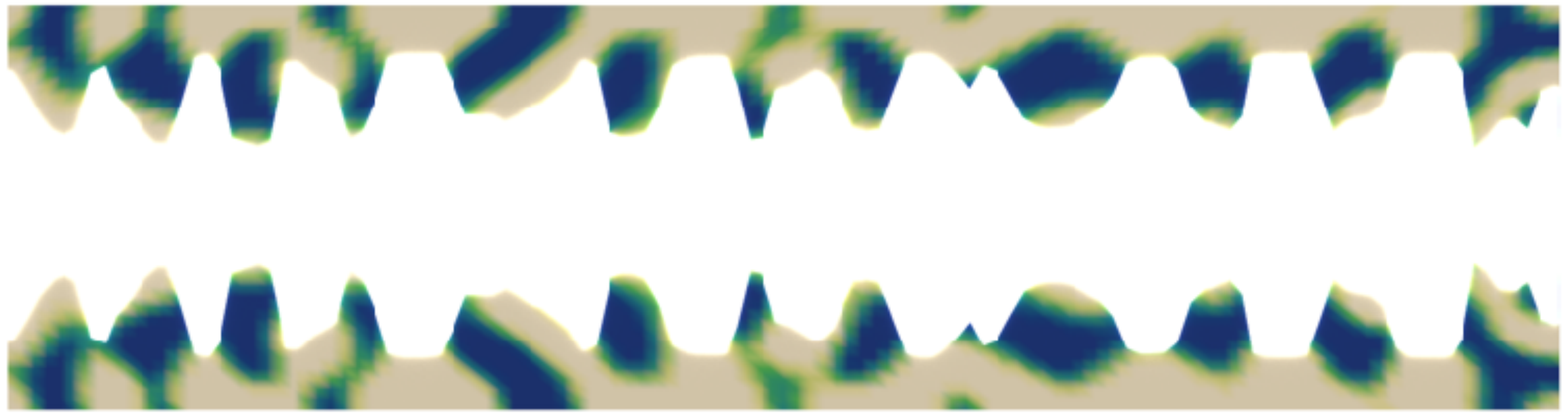
Optimize for each pixel



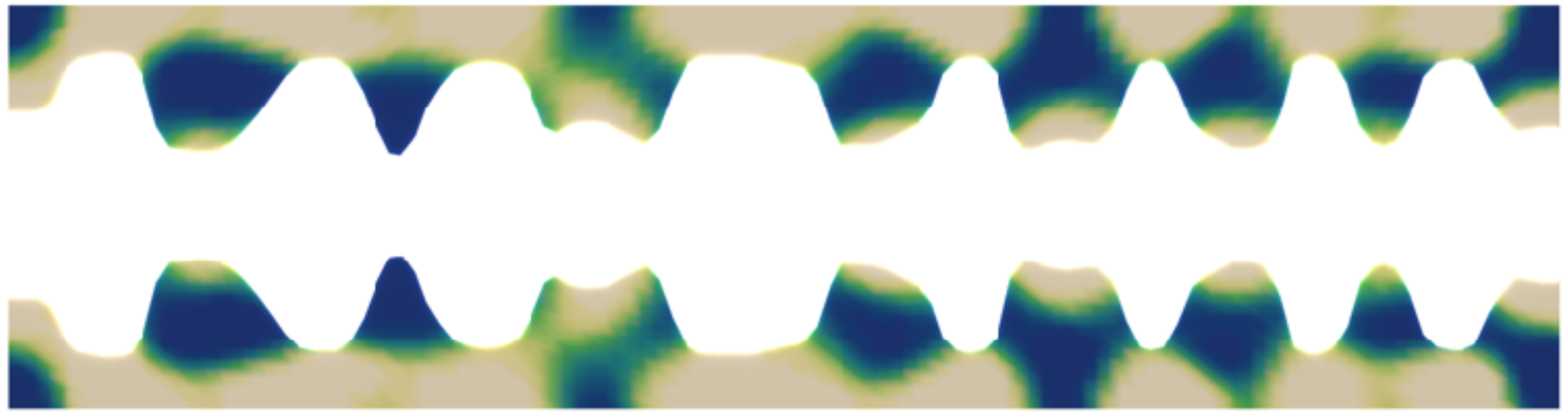
Objective: 252316 Time: 1945sec

Comparable Objective Values

Optimize for weights of the Laplacian basis (Ours)



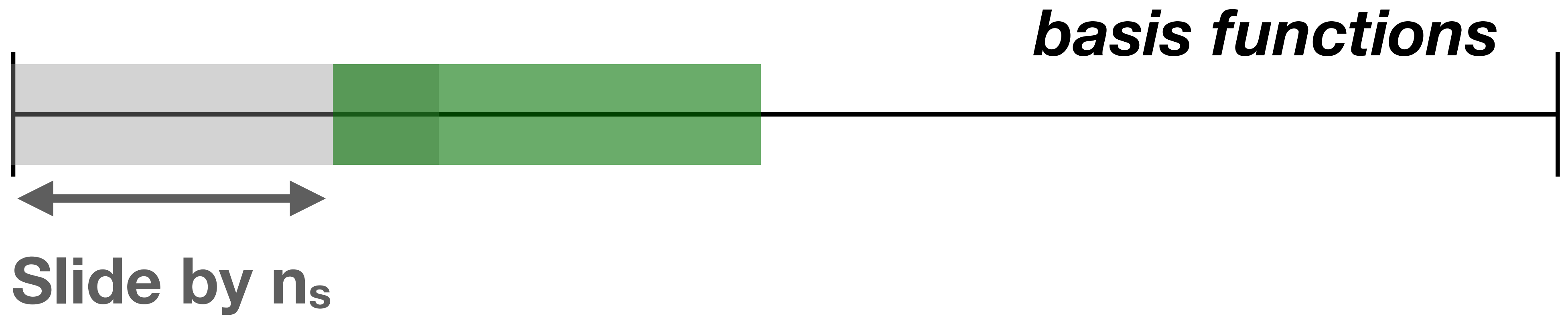
Sliding Basis — Objective: 253040 Time: 42sec

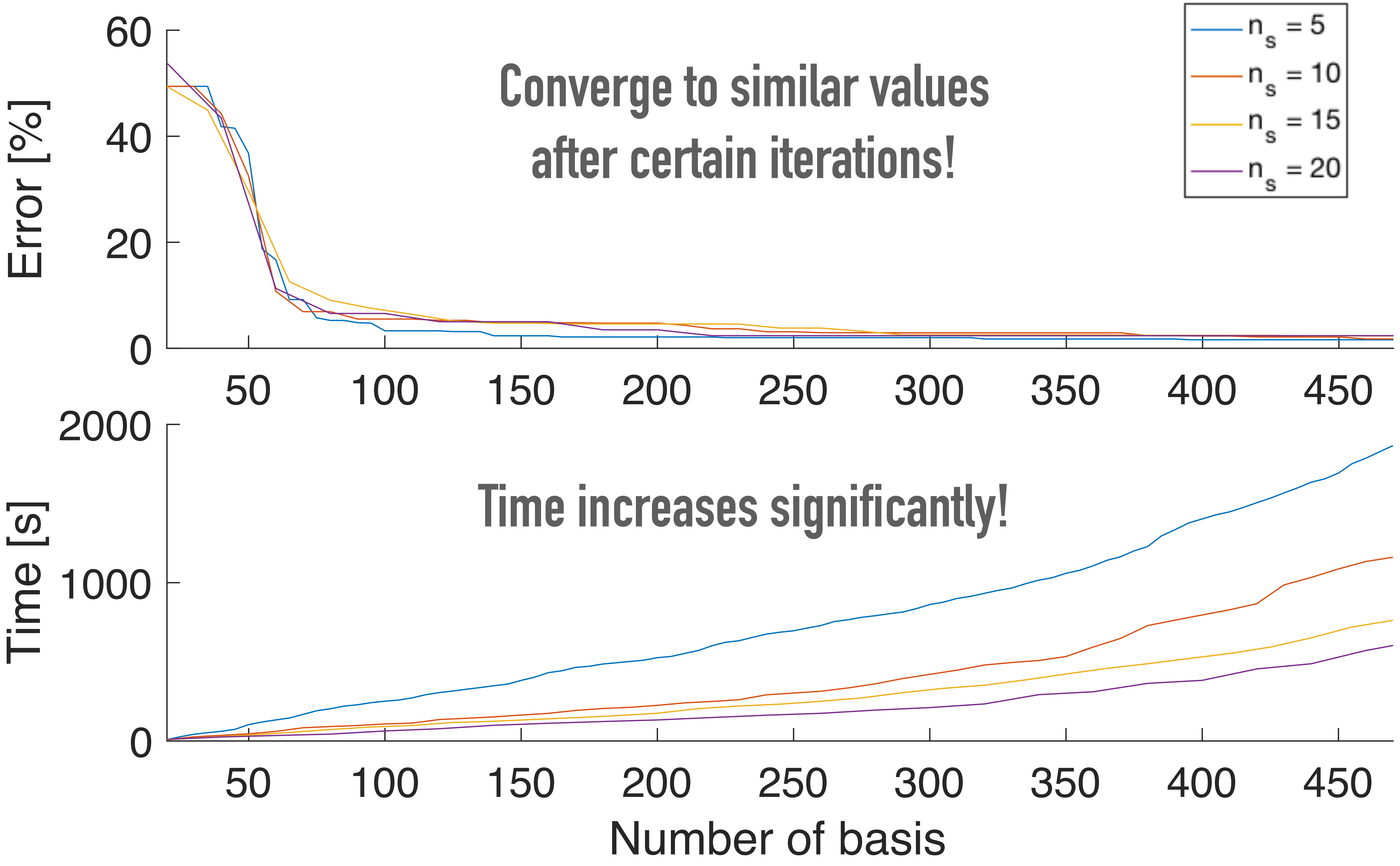


Fixed Basis — Objective: 287876 Time: 173sec

Reduced Order Faster

Effect of sliding amount, n_s





Converge to similar values after certain iterations!

Time increases significantly!

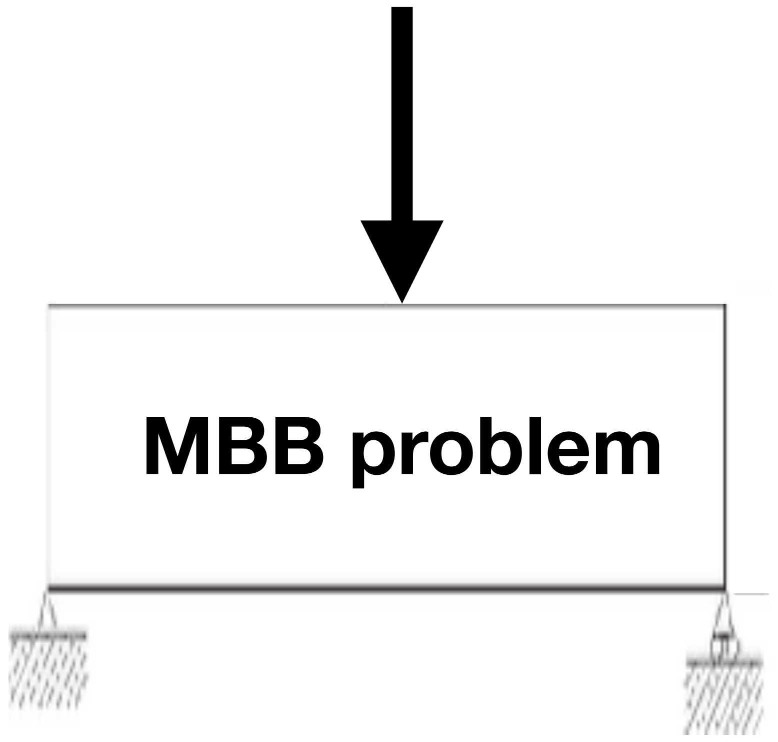
A Versatile Design Optimization Tool

The main **contributions** of the presented work:

- An optimization technique we call **sliding basis optimization** to **efficiently explore** parameterized design space
- **Practical material design method** with prescribed bounds using Laplacian basis
- **Enabling** optimization of material distributions for new applications coupled with **black-box analysis**

Recent Developments

Sliding basis topology optimization - a modular system



Sliding Basis Topology Optimization
Using **50 Bases**

Compliance objective: 302.2

Conventional Topology Optimization **(TOP88)**
Using **10k elements**

Compliance objective: 287.8

Goal is not to match the geometry but achieve comparable performance faster!!

Thank you!

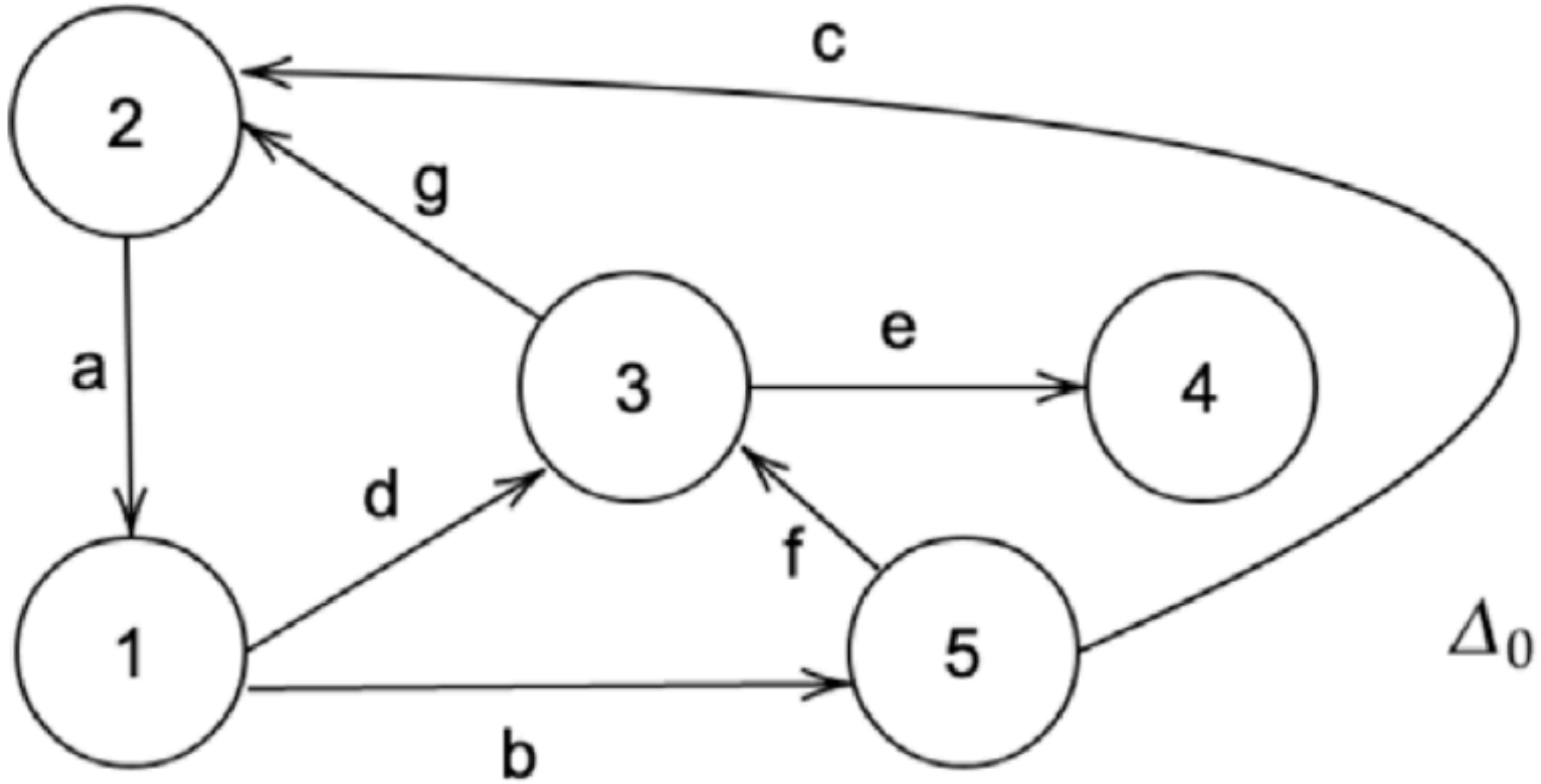
nulu@parc.com

Supplementary

Acknowledgements

The authors would like to thank NASA Jacobs Space Exploration Group for providing the solid rocket fuel design problem with the target thrust profile. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or U.S. Government. 3D models: dragon by XYZ RGB Inc and GE bracket by WilsonWong on GrabCAD.

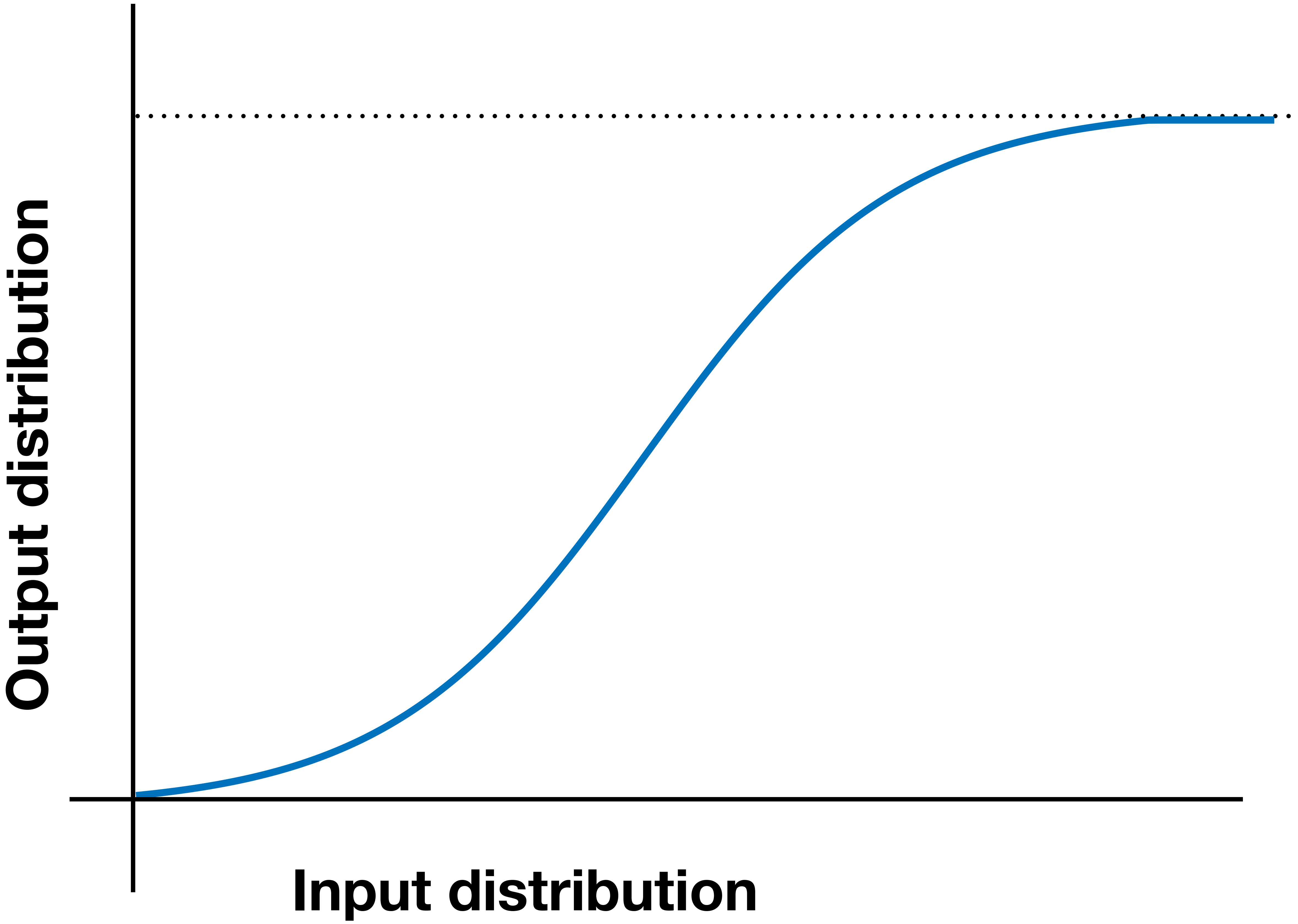
Graph Laplacian

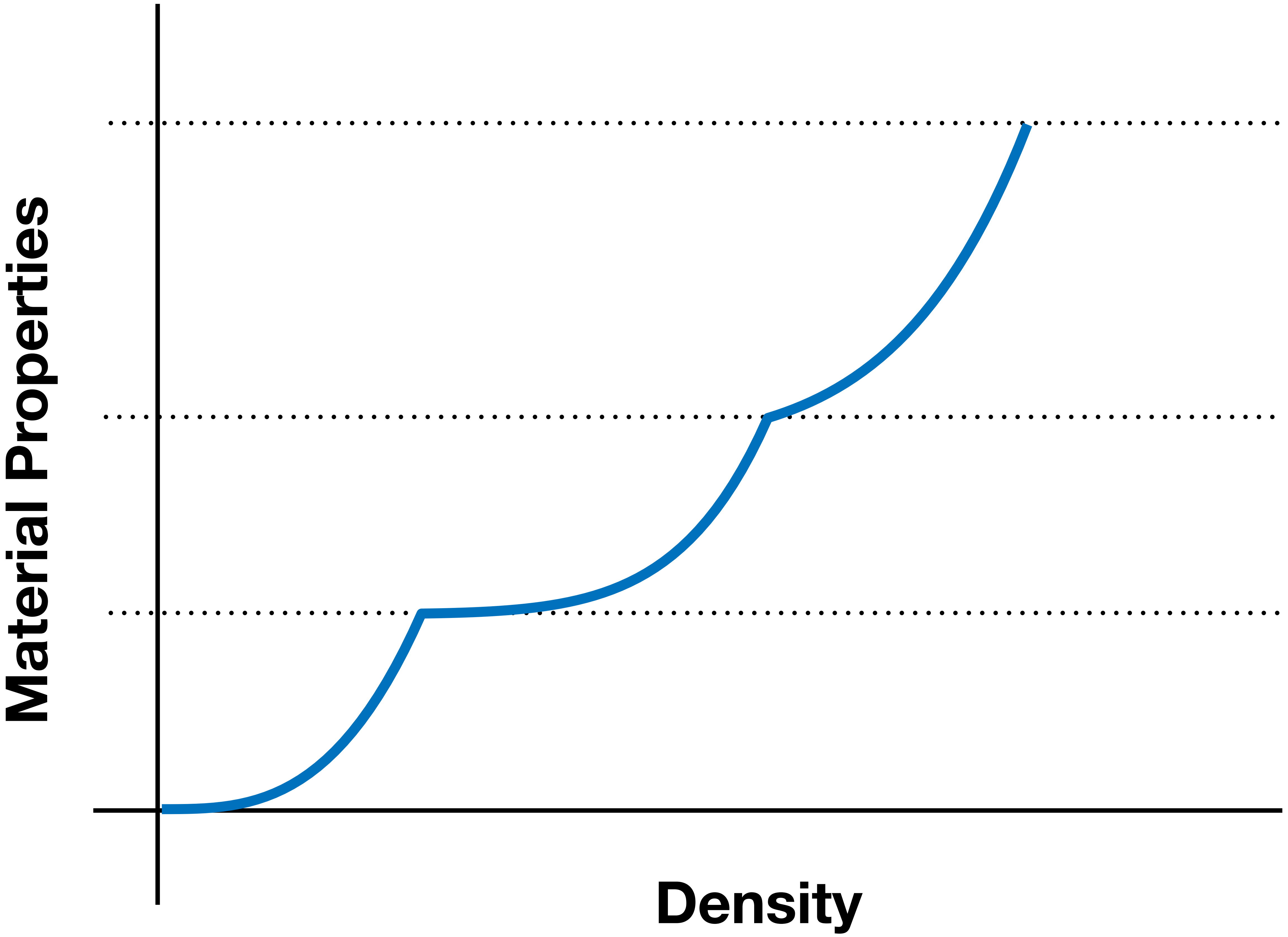


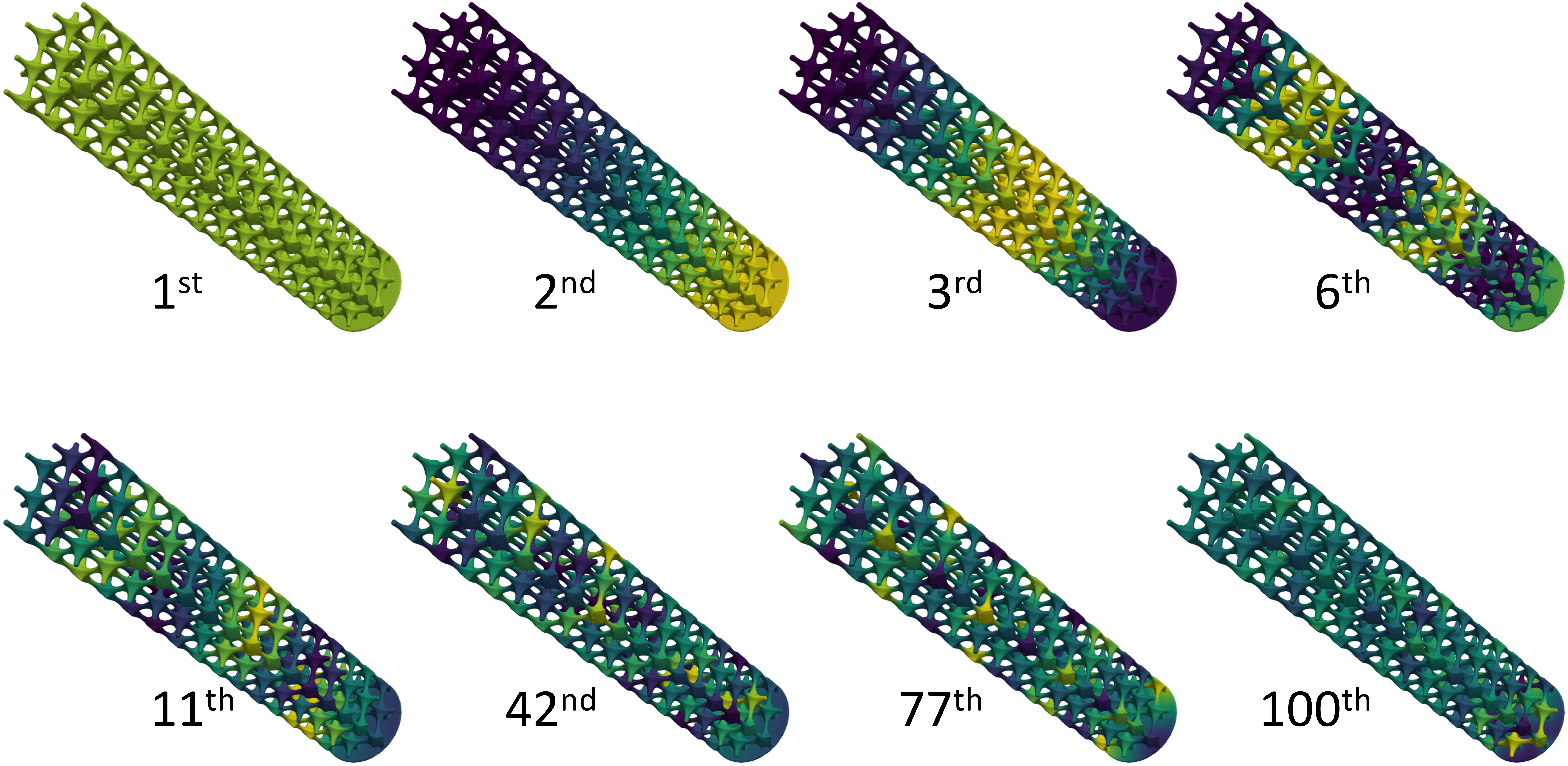
$$\partial_1 = \begin{pmatrix} -1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\Delta_0 = \partial_1 \, \partial_1^* = \begin{pmatrix} 3 & -1 & -1 & 0 & -1 \\ -1 & 3 & -1 & 0 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ -1 & -1 & -1 & 0 & 3 \end{pmatrix}$$

Filters


Output distribution
Input distribution


Material Properties
Density


1st
2nd
3rd
6th
11th
42nd
77th
100th


0.254x10-2
burn rate [m/sec]
1.52x10-2

					Fixed Basis		Sliding Basis	
Thrust Profile	n_{opt}	n_s	n_{slides}	Total Basis	Time	Objective/Error	Time	Objective/Error
Constant Acceleration	20	15	14	230	1178s	349k/2.3%	288s	86k/1.1%
Constant Deceleration	50	40	7	320	4896s	867k/3.4%	621s	452k/2.7%
Two Step	20	15	7	125	191s	102k/1.1%	69s	217k/1.4%
Bucket	20	15	24	380	1006s	272k/1.8%	596s	272k/1.8%

Why not automatic differentiation?

- Numerical differentiation is already implemented and default option in many optimization software.
- But our approach can also work with automatic differentiation.
- One disadvantage of automatic differentiation is that it cannot be used with truly black-box components.