Time parallelised Waveform Relaxation for Field/Circuit Coupled Systems

I. Cortes Garcia, H. De Gersem, P. Förster, I. Kulchytska-Ruchka, A. Moskalew, F. Quetscher, S. Schöps

TECHNISCHE UNIVERSITÄT DARMSTADT

Institut für Theorie Elektromagnetischer Felder
Computational Electromagnetics Research Group at GSCE

Outline

Motivation

- Monolithic time domain simulations are expensive due to ...
  - complicated (high-fidelity) models
  - different time scales
  - data transfer between simulators

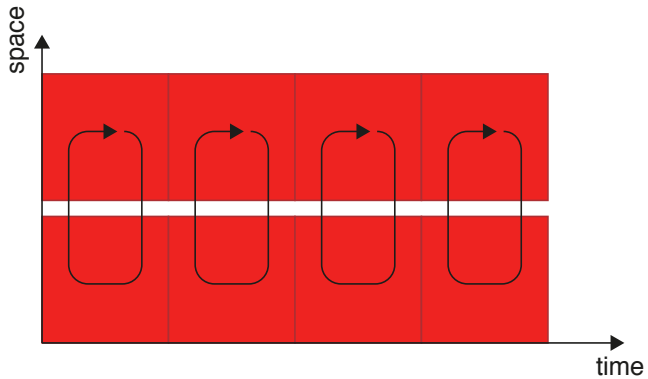
Motivation

- Monolithic time domain simulations are expensive due to ...
  - complicated (high-fidelity) models
  - different time scales
  - data transfer between simulators

- Modern computer hardware is based on **multiple processor** architectures that favors parallelization

Motivation

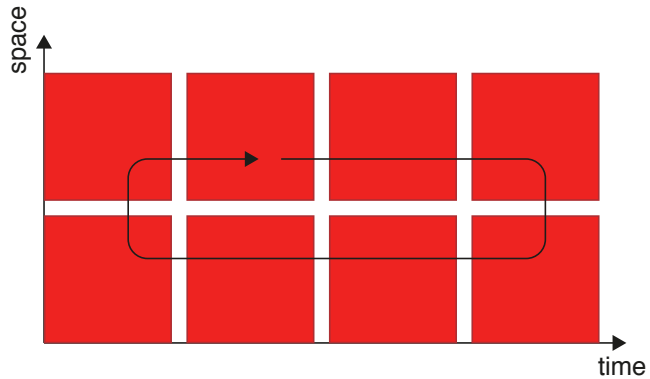
- Monolithic time domain simulations are expensive due to ...
  - complicated (high-fidelity) models
  - different time scales
  - data transfer between simulators



- Modern computer hardware is based on **multiple processor** architectures that favors parallelization
- Parallelization in ‘space’ or among models **already exploited**, e.g. when using waveform-relaxation

Motivation

- Monolithic time domain simulations are expensive due to ...
  - complicated (high-fidelity) models
  - different time scales
  - data transfer between simulators



- Modern computer hardware is based on **multiple processor** architectures that favors parallelization
- Parallelization in ‘space’ or among models **already exploited**, e.g. when using waveform-relaxation

$\Rightarrow$ **Time-parallel** algorithms have to be developed

Outline

The time domain is sequential

- If you want to solve a time domain problem for $t \in \mathcal{I} = (t_0, t_{\text{end}}]$

$$\mathbf{M}\mathrm{d}_t\mathbf{u}(t) = \underbrace{\mathbf{K}\mathbf{u}(t) + \mathbf{g}(t)}_{=:\mathbf{f}(\mathbf{u},t)} \qquad \text{with} \quad \mathbf{u}(0) = \mathbf{u}_0,$$

  then you need the solution operator $\mathbf{u}(t) = \mathcal{F}(t, t_0, \mathbf{u}_0)$.

The time domain is sequential

- If you want to solve a time domain problem for $t \in \mathcal{I} = (t_0, t_{\text{end}}]$

$$\mathbf{M}\mathrm{d}_t\mathbf{u}(t) = \underbrace{\mathbf{K}\mathbf{u}(t) + \mathbf{g}(t)}_{=:\mathbf{f}(\mathbf{u},t)} \qquad \text{with} \quad \mathbf{u}(0) = \mathbf{u}_0,$$

  then you need the solution operator $\mathbf{u}(t) = \mathcal{F}(t, t_0, \mathbf{u}_0)$.

- The Backward Euler Method $i = 1, \ldots$

$$\left(\frac{1}{\delta t}\mathbf{M} + \mathbf{K}\right)\mathbf{u}_i = \frac{1}{\delta t}\mathbf{M}\mathbf{u}_{i-1} + \mathbf{g}(t_i)$$

  is a numerical approximation to $\mathcal{F}$. It propagates $\mathbf{u}_0$ through time.

The time domain is sequential

- If you want to solve a time domain problem for $t \in \mathcal{I} = (t_0, t_{\text{end}}]$

$$\mathbf{M}\mathrm{d}_t\mathbf{u}(t) = \underbrace{\mathbf{K}\mathbf{u}(t) + \mathbf{g}(t)}_{=:\mathbf{f}(\mathbf{u},t)} \qquad \text{with} \quad \mathbf{u}(0) = \mathbf{u}_0,$$

  then you need the solution operator $\mathbf{u}(t) = \mathcal{F}(t, t_0, \mathbf{u}_0)$.

- The Backward Euler Method $i = 1, \ldots$

$$\left(\frac{1}{\delta t}\mathbf{M} + \mathbf{K}\right)\mathbf{u}_i = \frac{1}{\delta t}\mathbf{M}\mathbf{u}_{i-1} + \mathbf{g}(t_i)$$

  is a numerical approximation to $\mathcal{F}$. It propagates $\mathbf{u}_0$ through time.

- Let us call $\mathcal{F}$ the **fine propagator** and a less accurate $\mathcal{G}$ **coarse propagator**, e.g., Euler using $\Delta t \gg \delta t$.

Parareal: Splitting of the time interval

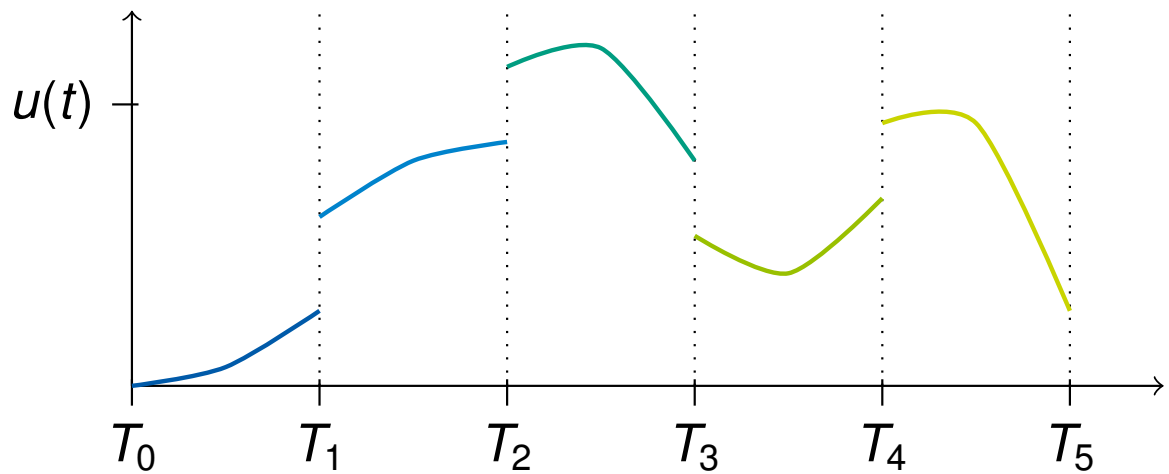
Partitioning $\mathcal{I}$ into N windows (e.g. one per core) yields

$$
\begin{cases}
\mathbf{M}\mathrm{d}_t\mathbf{u}_0 = \mathbf{f}(t,\mathbf{u}_0), & \mathbf{u}_0(t_0) = \mathbf{U}_0, & t \in (t_0, T_1], \\
\mathbf{M}\mathrm{d}_t\mathbf{u}_1 = \mathbf{f}(t,\mathbf{u}_1), & \mathbf{u}_1(T_1) = \mathbf{U}_1, & t \in (T_1, T_2], \\
\qquad\vdots & & \\
\mathbf{M}\mathrm{d}_t\mathbf{u}_{N-1} = \mathbf{f}(t,\mathbf{u}_{N-1}), & \mathbf{u}_{N-1}(T_{N-1}) = \mathbf{U}_{N-1}, & t \in (T_{N-1}, T_N],
\end{cases}
$$

Parareal: Splitting of the time interval

Partitioning $\mathcal{I}$ into N windows (e.g. one per core) yields

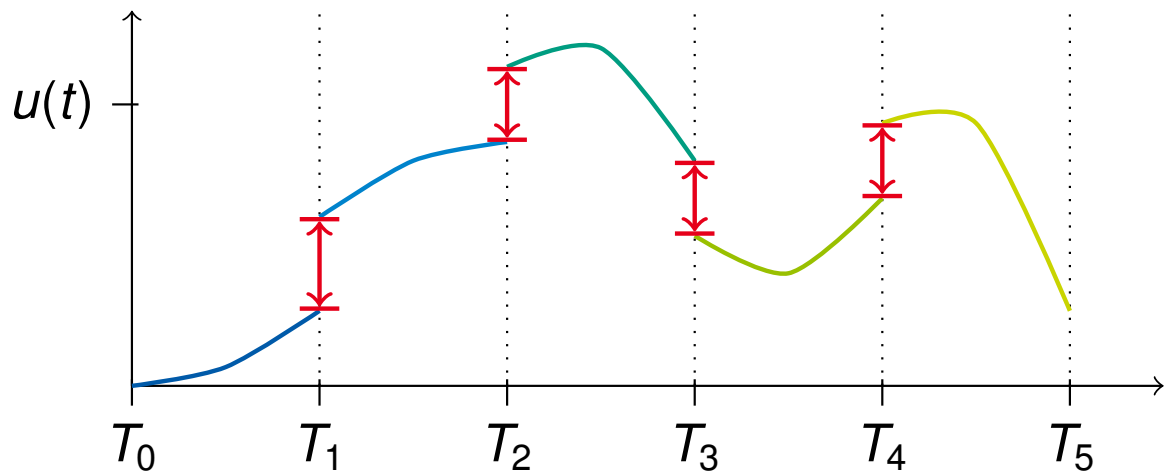
$$
\begin{cases}
\mathbf{M}\mathrm{d}_t\mathbf{u}_0 = \mathbf{f}(t,\mathbf{u}_0), & \mathbf{u}_0(t_0) = \mathbf{U}_0, & t \in (t_0, T_1], \\
\mathbf{M}\mathrm{d}_t\mathbf{u}_1 = \mathbf{f}(t,\mathbf{u}_1), & \mathbf{u}_1(T_1) = \mathbf{U}_1, & t \in (T_1, T_2], \\
\quad\vdots & & \\
\mathbf{M}\mathrm{d}_t\mathbf{u}_{N-1} = \mathbf{f}(t,\mathbf{u}_{N-1}), & \mathbf{u}_{N-1}(T_{N-1}) = \mathbf{U}_{N-1}, & t \in (T_{N-1}, T_N],
\end{cases}
$$



Parareal: Splitting of the time interval

Partitioning $\mathcal{I}$ into N windows (e.g. one per core) yields

$$
\begin{cases}
\mathbf{M}\mathrm{d}_t\mathbf{u}_0 = \mathbf{f}(t,\mathbf{u}_0), & \mathbf{u}_0(t_0) = \mathbf{U}_0, & t \in (t_0, T_1], \\
\mathbf{M}\mathrm{d}_t\mathbf{u}_1 = \mathbf{f}(t,\mathbf{u}_1), & \mathbf{u}_1(T_1) = \mathbf{U}_1, & t \in (T_1, T_2], \\
\qquad\vdots & & \\
\mathbf{M}\mathrm{d}_t\mathbf{u}_{N-1} = \mathbf{f}(t,\mathbf{u}_{N-1}), & \mathbf{u}_{N-1}(T_{N-1}) = \mathbf{U}_{N-1}, & t \in (T_{N-1}, T_N],
\end{cases}
$$



Parareal as a Newton-Raphson method (I)

The matching conditions

$$\begin{cases} \mathbf{U}_0 - \mathbf{u}_0 = 0, \\ \mathbf{U}_1 - \mathcal{F}(T_1, T_0, \mathbf{U}_0) = 0, \\ \vdots \\ \mathbf{U}_{N-1} - \mathcal{F}(T_{N-2}, T_{N-1}, \mathbf{U}_{N-2}) = 0 \end{cases}$$

can be written as unknown of a nonlinear equation

$$\mathbf{F}(\mathbf{U}) = 0, \quad \text{with} \quad \mathbf{U} = (\mathbf{U}_0, \mathbf{U}_1, ..., \mathbf{U}_i, ..., \mathbf{U}_{N-1}).$$

[1] M. J. Gander and E. Hairer, Nonlinear convergence analysis for the parareal algorithm. Springer Berlin Heidelberg, 2008.

Parareal as a Newton-Raphson method (I)

The matching conditions

$$\begin{cases} \mathbf{U}_0 - \mathbf{u}_0 = 0, \\ \mathbf{U}_1 - \mathcal{F}(T_1, T_0, \mathbf{U}_0) = 0, \\ \vdots \\ \mathbf{U}_{N-1} - \mathcal{F}(T_{N-2}, T_{N-1}, \mathbf{U}_{N-2}) = 0 \end{cases}$$

can be written as unknown of a nonlinear equation

$$\mathbf{F}(\mathbf{U}) = 0, \quad \text{with} \quad \mathbf{U} = (\mathbf{U}_0, \mathbf{U}_1, ..., \mathbf{U}_i, ..., \mathbf{U}_{N-1}).$$

This problem can be solved using Newton-Raphson[1]

$$\mathbf{U}_n^{(k+1)} = \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)}) + \frac{\partial \mathcal{F}(T_n, T_{n-1}, \mathbf{U})}{\partial \mathbf{U}} (\mathbf{U}_{n-1}^{(k+1)} - \mathbf{U}_{n-1}^{(k)})$$

[1] M. J. Gander and E. Hairer, Nonlinear convergence analysis for the parareal algorithm. Springer Berlin Heidelberg, 2008.

Parareal as a Newton-Raphson method (II)

Parareal solves for the matching conditions using Newton-Raphson.
It uses the operators:

- $\mathcal{F}$ used for computing accurate solution for each sub-interval

$$\tilde{\mathbf{U}}_n^{(k)} = \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)}),$$

- $\mathcal{G}$ used for propagating information between sub-intervals

$$\frac{\partial \mathcal{F}(T_n, T_{n-1}, \mathbf{U})}{\partial \mathbf{U}} (\mathbf{U}_{n-1}^{(k+1)} - \mathbf{U}_{n-1}^{(k)}) \approx \bar{\mathbf{U}}_n^{(k+1)} - \bar{\mathbf{U}}_n^{(k)}$$

$$= \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k+1)}) - \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)}).$$

Parareal as a Newton-Raphson method (II)

Parareal solves for the matching conditions using Newton-Raphson.
It uses the operators:

- $\mathcal{F}$ used for computing accurate solution for each sub-interval

$$\tilde{\mathbf{U}}_n^{(k)} = \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)}),$$

- $\mathcal{G}$ used for propagating information between sub-intervals

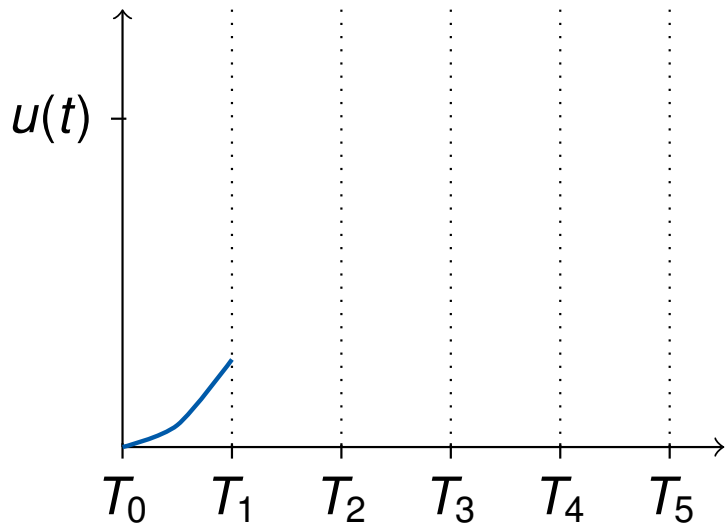
$$\frac{\partial \mathcal{F}(T_n, T_{n-1}, \mathbf{U})}{\partial \mathbf{U}}(\mathbf{U}_{n-1}^{(k+1)} - \mathbf{U}_{n-1}^{(k)}) \approx \bar{\mathbf{U}}_n^{(k+1)} - \bar{\mathbf{U}}_n^{(k)}$$
$$= \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k+1)}) - \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)}).$$

$\mathcal{F}$ uses only previous solution $\rightarrow$ can be run in parallel

$\mathcal{G}$ uses current solution $\rightarrow$ cannot be run in parallel but is cheap

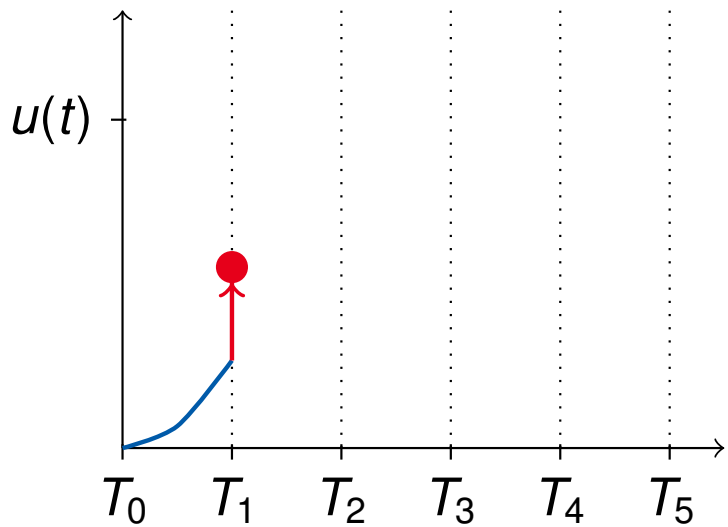
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



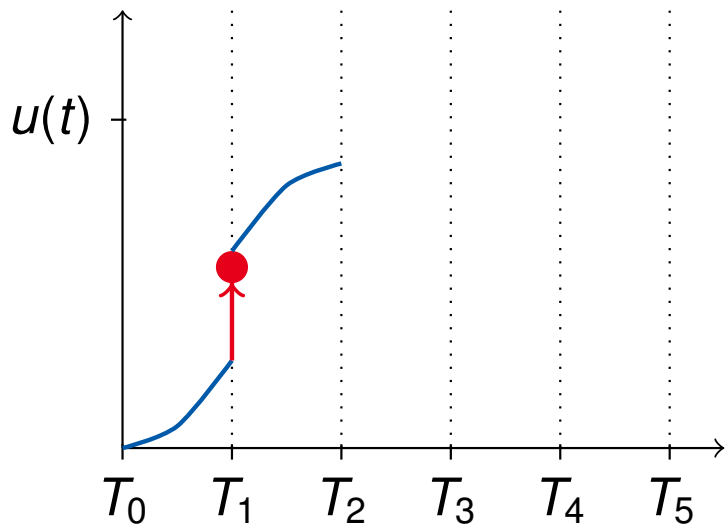
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



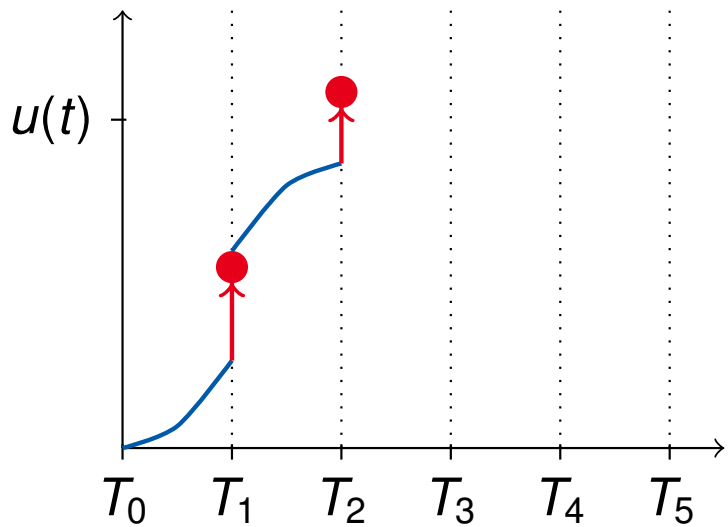
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



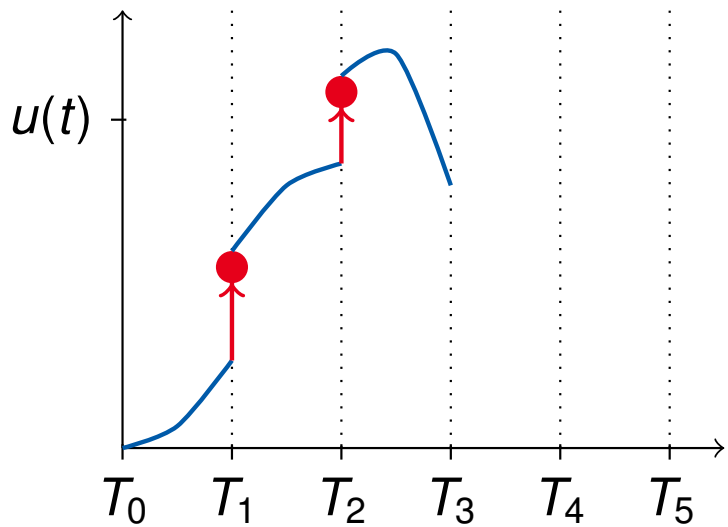
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



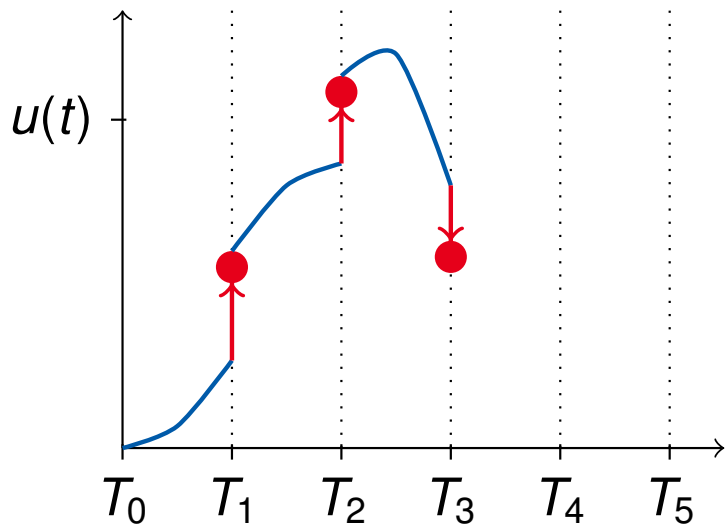
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



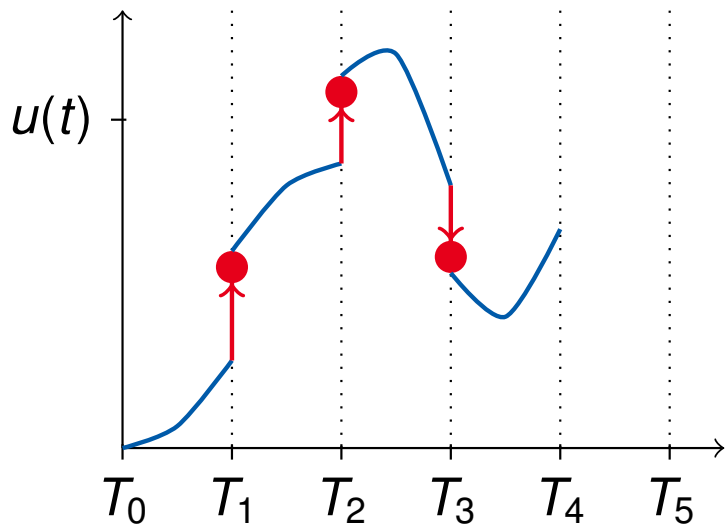
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



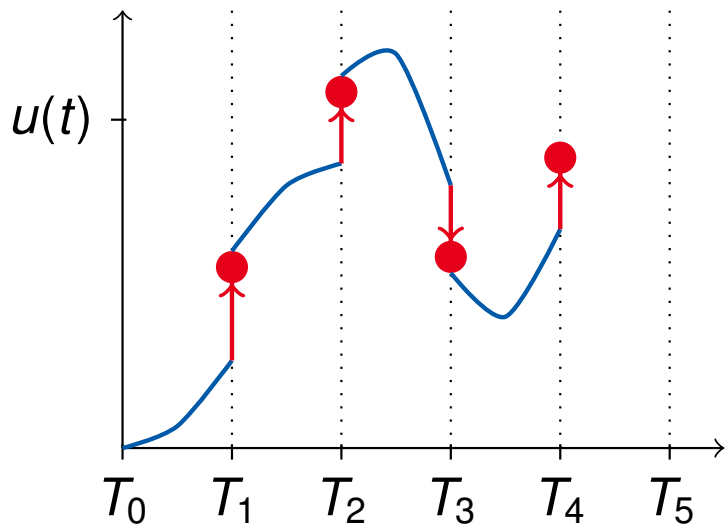
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



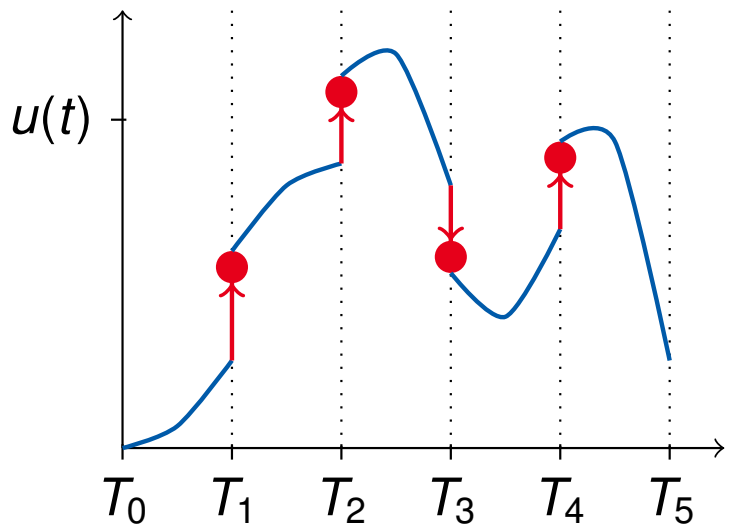
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



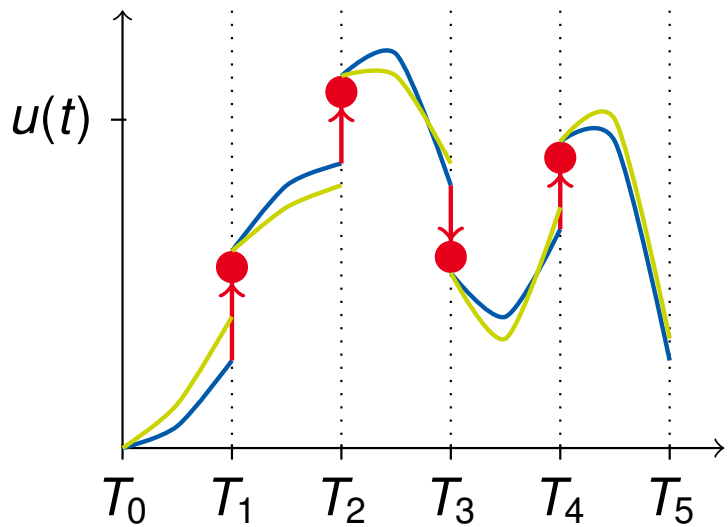
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



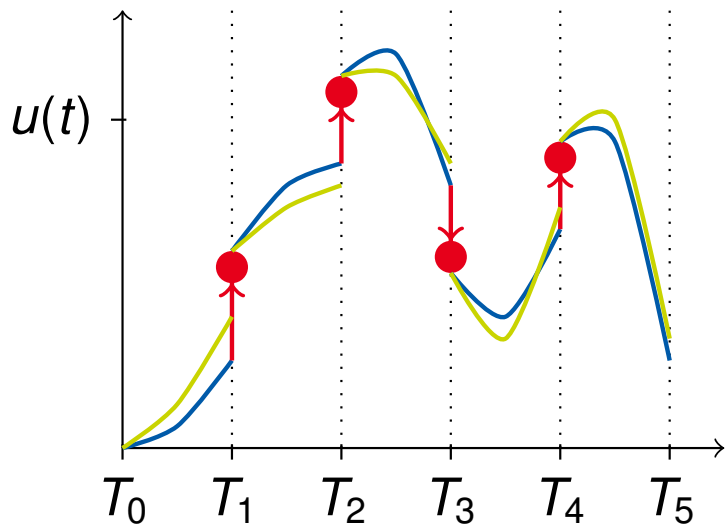
Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



Parareal algorithm with Animation

initialize: $\mathbf{U}_0^{(k)} \leftarrow \mathbf{U}_0$ and $\bar{\mathbf{U}}_n^{(0)}, \tilde{\mathbf{U}}_n^{(0)} \leftarrow \mathbf{0}$
counter: $k \leftarrow 1$;
while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    coarse: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{G}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    fine: $\tilde{\mathbf{U}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
  end
  increment: $k \leftarrow k + 1$;
end



Terminates at latest after iteration $k = n$.

Outline

Application to Field/Circuit Coupled Problems

- Field/circuit coupled problem consists of the following subproblems
  - an electric network problem
  - (multiple) field problems, coupled via currents i_m and voltages v_m
- During Waveform-Relaxation (WR) each field model is represented by

$$v_m = L_m \, \mathrm{d}_t i_\mathrm{m} + \text{correction.}$$

- Idea: use a simplified coarse propagation $\tilde{\mathcal{G}}$ that disregards corrections, i.e. solve the circuit with (linearized) inductances

$$\tilde{v}_m = L_m \, \mathrm{d}_t \tilde{i}_\mathrm{m}$$

  while $\mathcal{F}$ still uses WR to solve the fully coupled problem
- if field problems are **fast** then this is similar to Micro-Macro Parareal[2]

[2] F. Legoll, T. Lelièvre, and G. Samaey. A Micro-Macro Parareal Algorithm: Application to Singularly Perturbed Ordinary Differential Equations. SIAM Journal on Scientific Computing 35.4 (2013).

A Micro-Macro Parareal algorithm

while $\max_n \|\mathbf{U}_n^{(k)} - \mathbf{U}_n^{(k-1)}\| > tol$ **do**
  for $n \leftarrow 1$ **to** n_{cpu} **do**
    circuit solver: $\bar{\mathbf{U}}_n^{(k)} \leftarrow \tilde{\mathcal{G}}(T_n, T_{n-1}, \mathbf{U}_{n-1}^{(k)})$;
    Newton: $\mathbf{U}_n^{(k)} \leftarrow \bar{\mathbf{U}}_n^{(k)} + \tilde{\mathbf{U}}_n^{(k-1)} - \bar{\mathbf{U}}_n^{(k-1)}$;
  end
  parfor $n \leftarrow 1$ **to** n_{cpu} **do**
    upscale: solve a magnetostatic problem to obtain $\bar{\mathbf{Y}}_n^{(k)}$ from $\bar{\mathbf{U}}_n^{(k)}$;
    consistent Newton: $\mathbf{Y}_n^{(k)} \leftarrow \bar{\mathbf{Y}}_n^{(k)} + \tilde{\mathbf{Y}}_n^{(k-1)} - \bar{\mathbf{Y}}_n^{(k-1)}$;
    WR solver: $\tilde{\mathbf{Y}}_n^{(k)} \leftarrow \mathcal{F}(T_n, T_{n-1}, \mathbf{Y}_{n-1}^{(k)})$;
    downscale: extract circuit variables $\tilde{\mathbf{U}}_n^{(k)}$ from $\tilde{\mathbf{Y}}_n^{(k)}$;
  end
  increment: $k \leftarrow k + 1$;
end

How fast are the field problems?

- An ordinary autonomous differential equation

$$\mathbf{M}\mathrm{d}_t\mathbf{u} = \mathbf{K}\mathbf{u} + \mathbf{r}(t)$$

or equivalently

$$\mathrm{d}_t\mathbf{u} = \mathbf{M}^{-1}\mathbf{K}\mathbf{u} + \mathbf{M}^{-1}\mathbf{r}(t)$$

is called **fast** (or stiff) if the system matrix has large negative eigenvalues

$$\mathrm{eig}\left(\mathbf{M}^{-1}\mathbf{K}\right) \ll 0.$$

[3] Warning: the CERN problem might behave differently since there is no excitation in non-conductive domain, i.e., the nullspace of **M**.

How fast are the field problems?

- An ordinary autonomous differential equation

$$\mathbf{M}\mathrm{d}_t\mathbf{u} = \mathbf{K}\mathbf{u} + \mathbf{r}(t)$$

  or equivalently

$$\mathrm{d}_t\mathbf{u} = \mathbf{M}^{-1}\mathbf{K}\mathbf{u} + \mathbf{M}^{-1}\mathbf{r}(t)$$

  is called **fast** (or stiff) if the system matrix has large negative eigenvalues

$$\mathrm{eig}\left(\mathbf{M}^{-1}\mathbf{K}\right) \ll 0.$$

- Hand-waving argument: in our case is **K** negative definite and **M** not invertible, i.e., $\mathrm{eig}\left(\mathbf{M}^{-1}\right) \to \infty$, therefore the system is **very fast**

[3] Warning: the CERN problem might behave differently since there is no excitation in non-conductive domain, i.e., the nullspace of **M**.

How fast are the field problems?

- An ordinary autonomous differential equation

$$\mathbf{M}\mathrm{d}_t\mathbf{u} = \mathbf{K}\mathbf{u} + \mathbf{r}(t)$$

  or equivalently

$$\mathrm{d}_t\mathbf{u} = \mathbf{M}^{-1}\mathbf{K}\mathbf{u} + \mathbf{M}^{-1}\mathbf{r}(t)$$

  is called **fast** (or stiff) if the system matrix has large negative eigenvalues

$$\mathrm{eig}\left(\mathbf{M}^{-1}\mathbf{K}\right) \ll 0.$$

- Hand-waving argument: in our case is $\mathbf{K}$ negative definite and $\mathbf{M}$ not invertible, i.e., $\mathrm{eig}\left(\mathbf{M}^{-1}\right) \to \infty$, therefore the system is **very fast**
- In the limit case ($\to \infty$) the problem is purely algebraic (static)

$$0 = \mathbf{K}\mathbf{u} + \mathbf{r}(t)$$

  which to represent field problems by inductances $L = -\mathbf{X}^\top\mathbf{K}^{-1}\mathbf{X}$.[3]

[3] Warning: the CERN problem might behave differently since there is no excitation in non-conductive domain, i.e., the nullspace of $\mathbf{M}$.

Trying to be less hand-waving

- Split the equation into parts

$$\begin{aligned} \mathrm{d}_t \mathbf{u}_1 &= \mathbf{M}_1^{-1}\mathbf{K}_{12}\mathbf{u}_1 + \mathbf{M}_1^{-1}\mathbf{K}_{12}\mathbf{u}_2 + \mathbf{M}_1^{-1}\mathbf{X}_1 i \\ \mathbf{0} &= \mathbf{K}_{21}\mathbf{u}_1 + \mathbf{K}_{22}\mathbf{u}_2 + \mathbf{X}_2 i \\ \Phi &= \mathbf{X}_1^\top \mathbf{u}_1 + \mathbf{X}_2^\top \mathbf{u}_2 \end{aligned}$$

- Let's exploit the Schur complement to have finite stiffness

$$\mathrm{d}_t \mathbf{u}_1 = \mathbf{M}_1^{-1}\bar{\mathbf{K}}_{11}\mathbf{u}_1 + \mathbf{M}_1^{-1}\bar{\mathbf{X}}_1 i$$

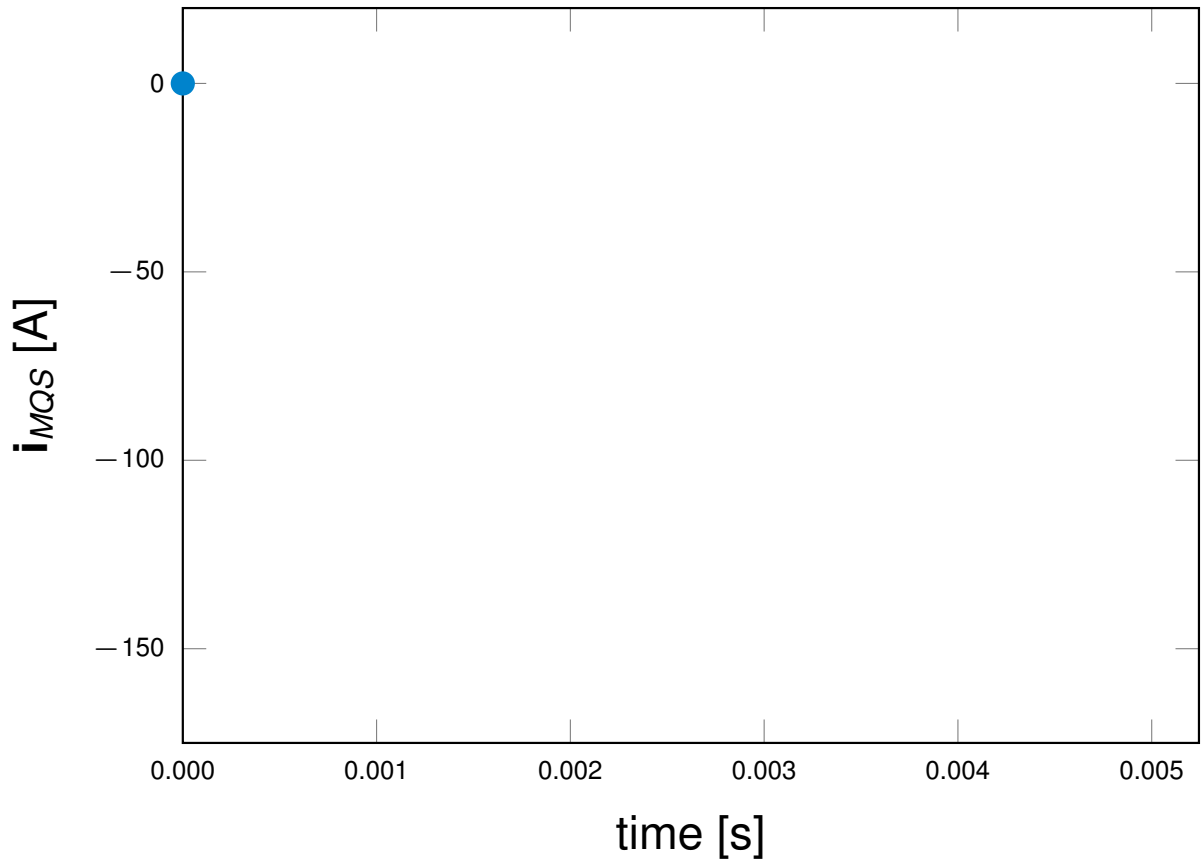
with $\bar{\mathbf{K}}_{11} = \mathbf{K}_{11} - \mathbf{K}_{12}\mathbf{K}_{22}^{-1}\mathbf{K}_{21}$ and $\bar{\mathbf{X}}_1 := \mathbf{X}_1^\top - \mathbf{X}_2^\top \mathbf{K}_{22}^{-1}\mathbf{K}_{21}$ and coupling equation

$$\Phi = \bar{\mathbf{X}}_1^\top \mathbf{u}_1 + L_2 i$$

and 'free space' inductance $L_2 := -\mathbf{X}_2^\top \mathbf{K}_{22}^{-1}\mathbf{X}_2$.

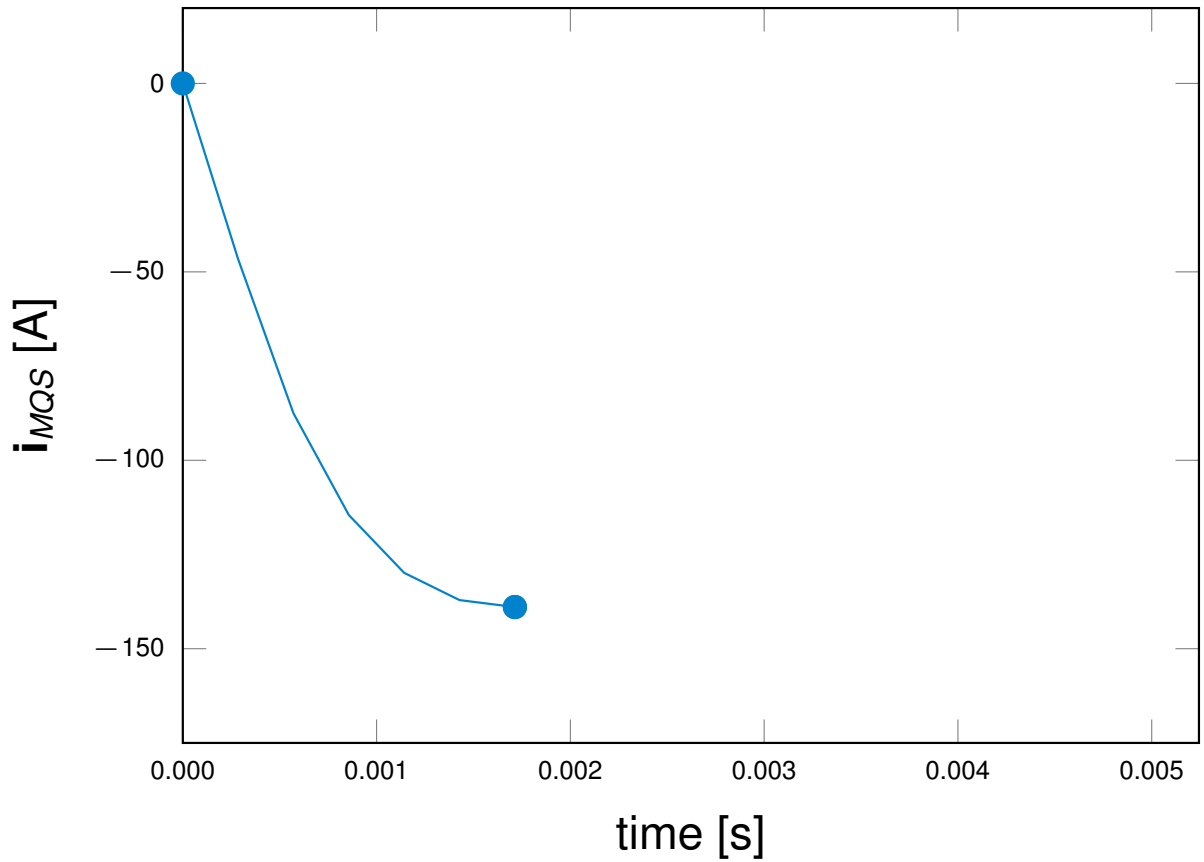
Animation of Micro-Macro Parareal using a Toy Example

Original initial value



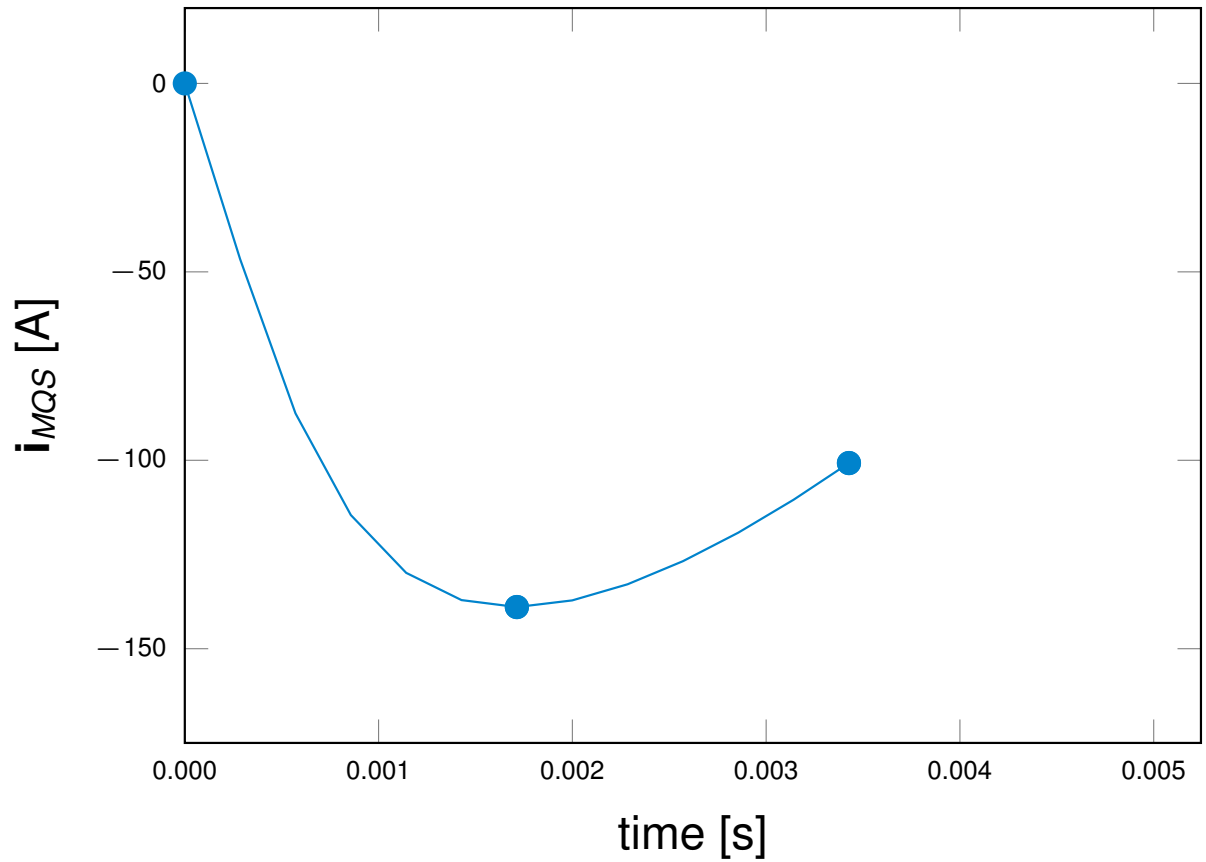
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



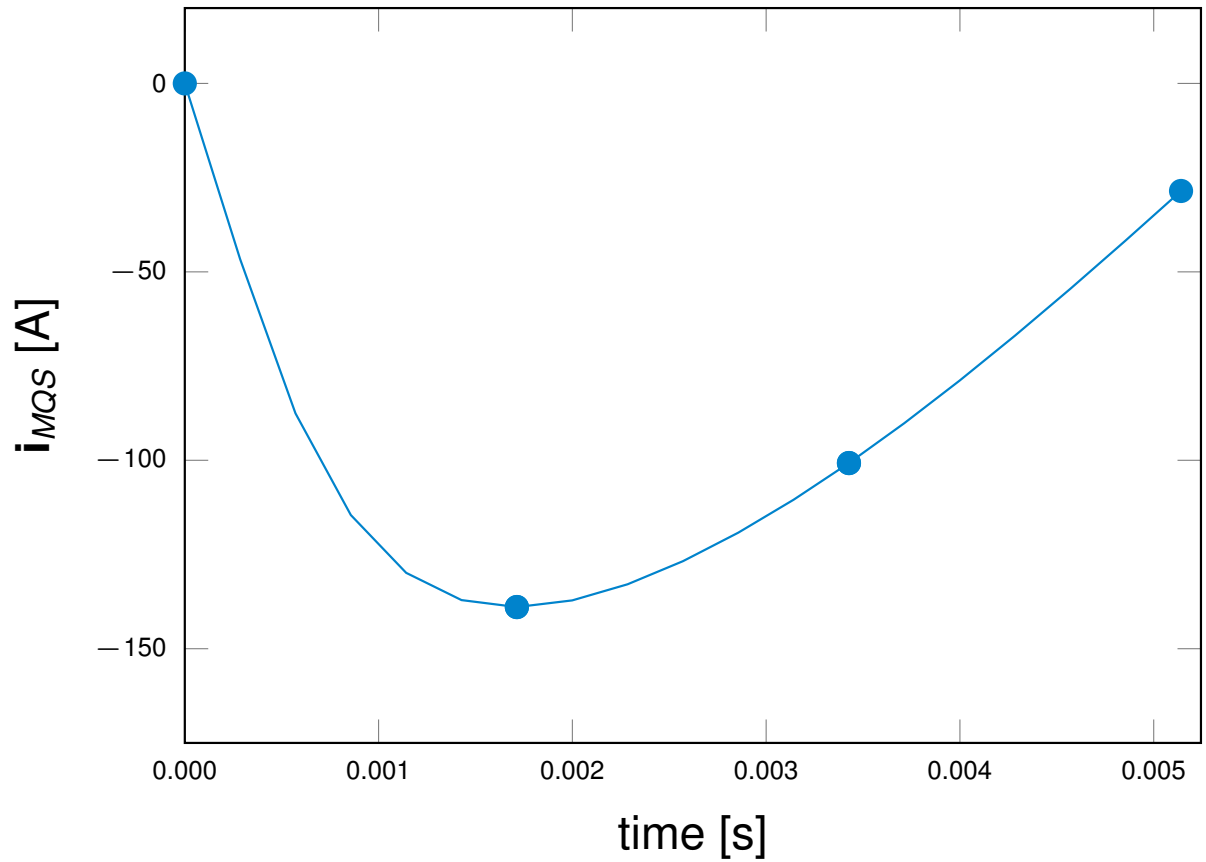
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



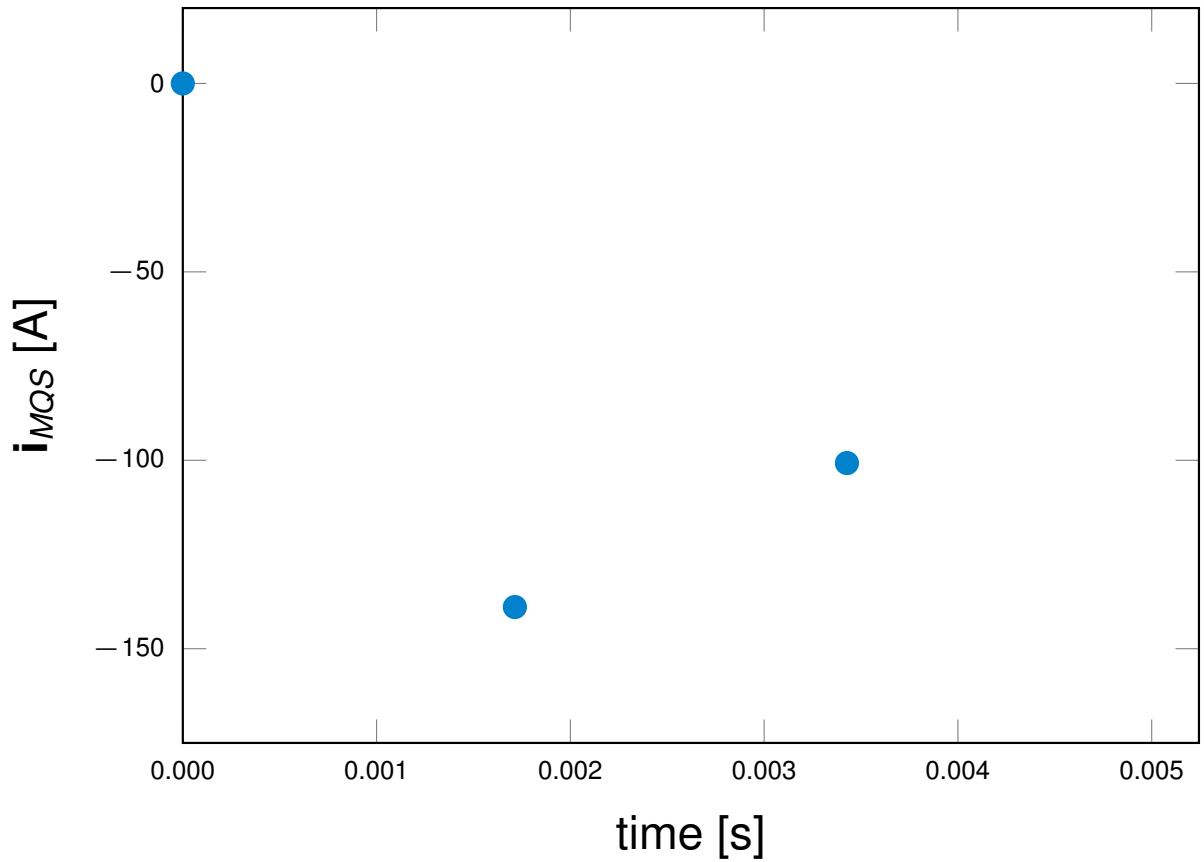
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



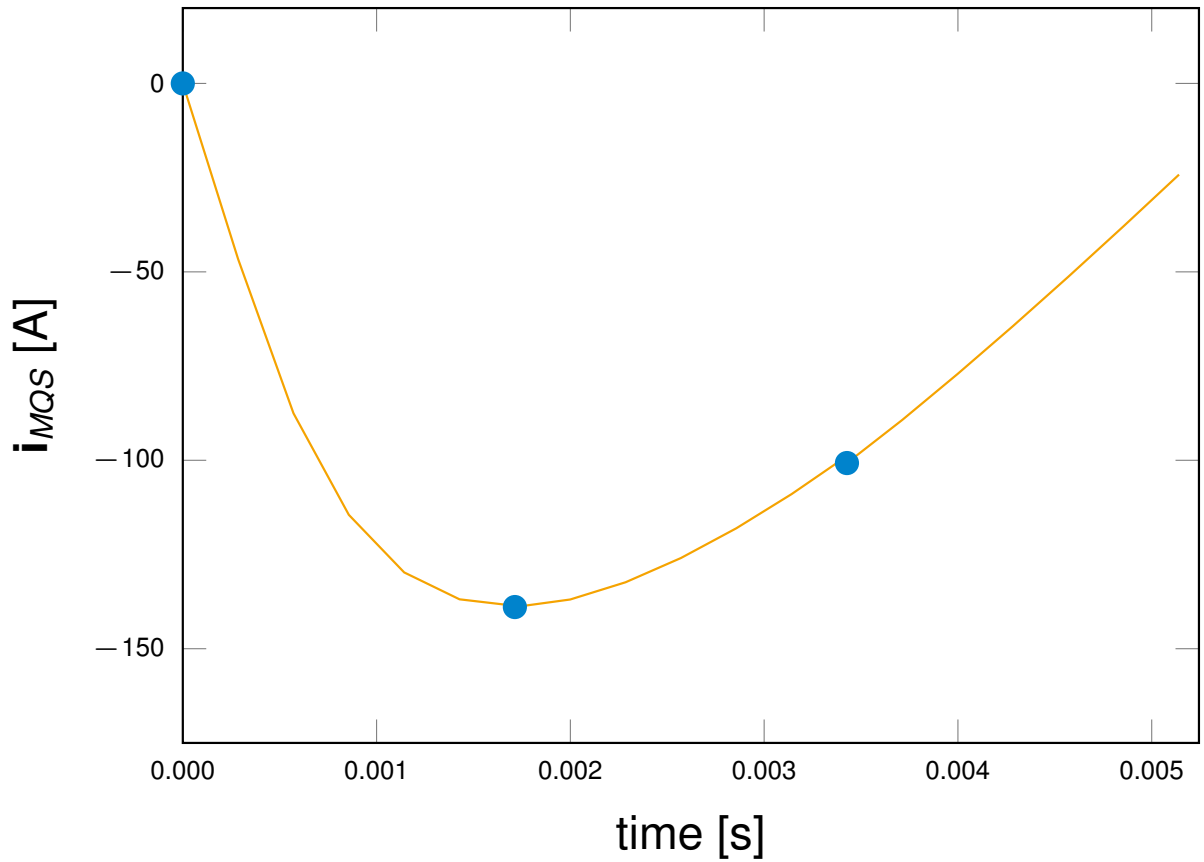
Animation of Micro-Macro Parareal using a Toy Example

Fine:
initial values



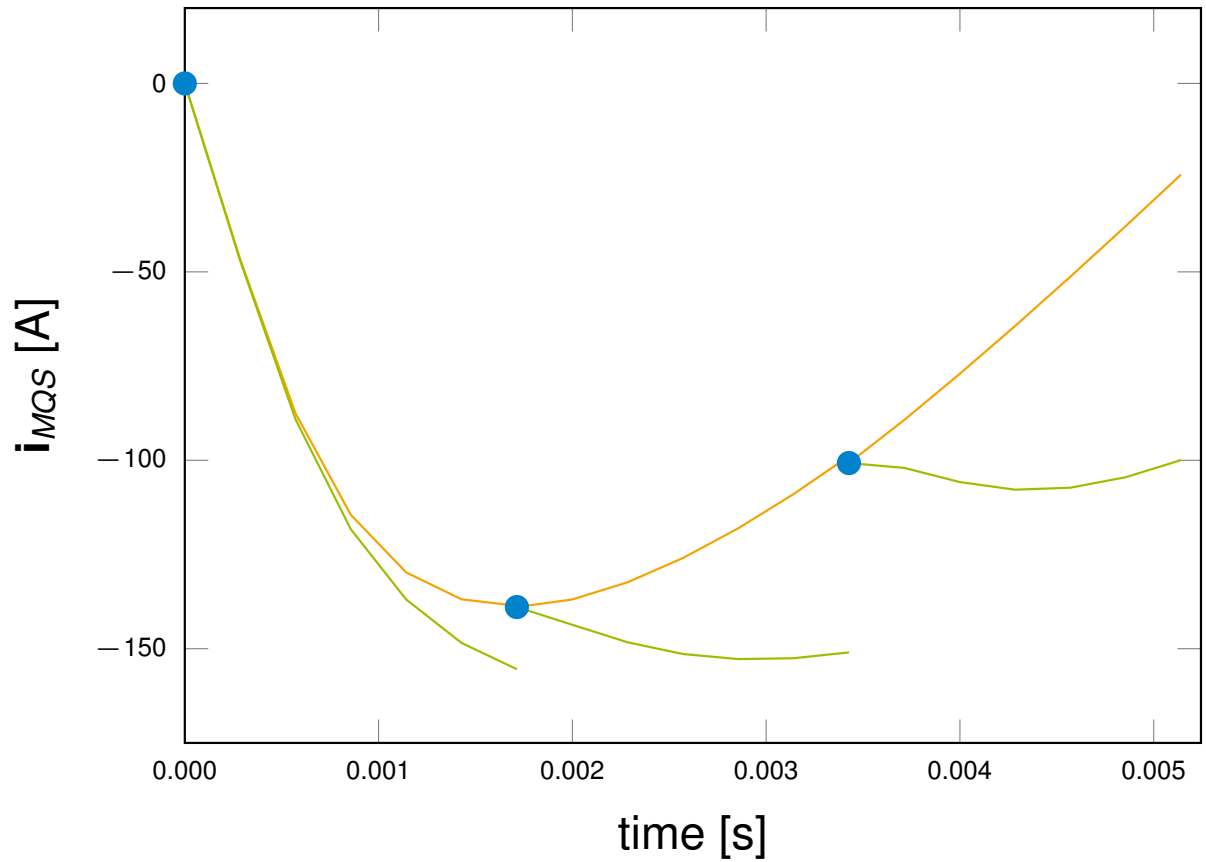
Animation of Micro-Macro Parareal using a Toy Example

Fine:
extrapolation from coarse



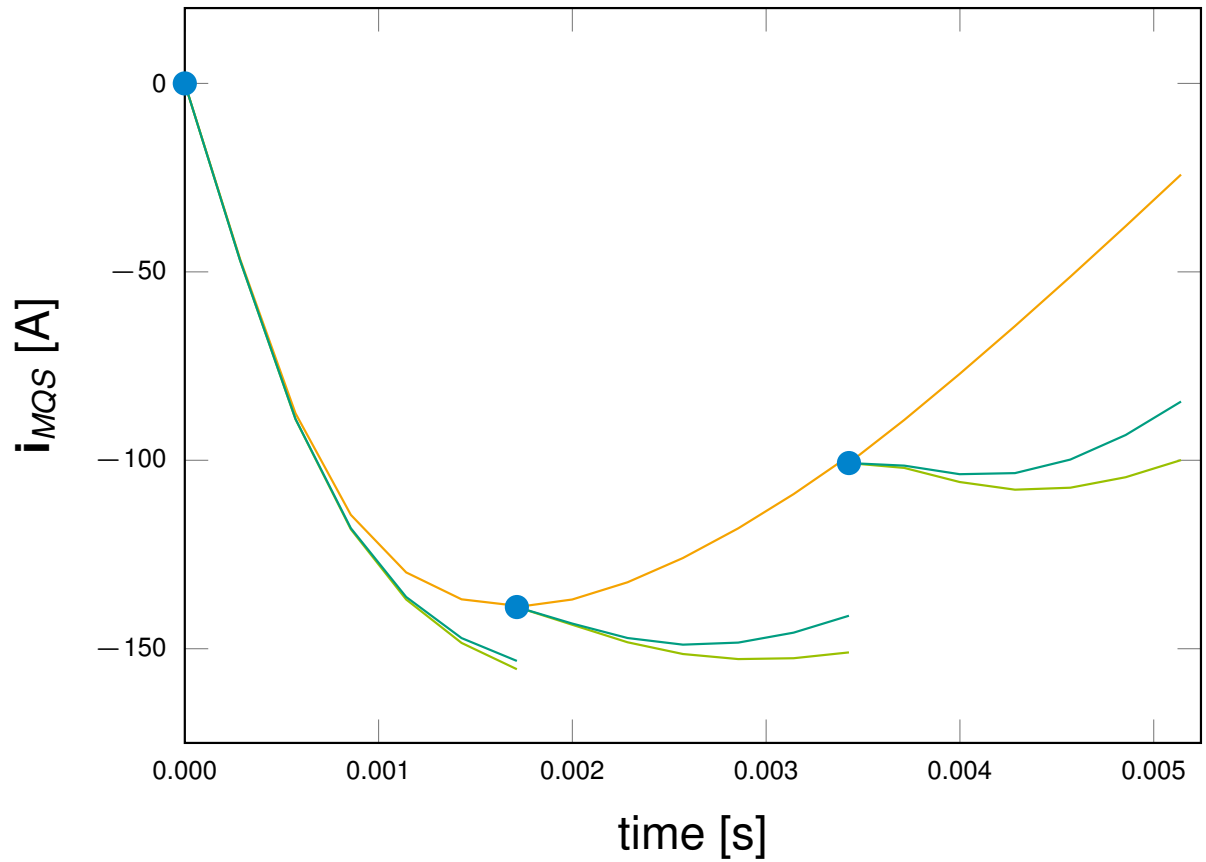
Animation of Micro-Macro Parareal using a Toy Example

Fine:
WR iteration 1



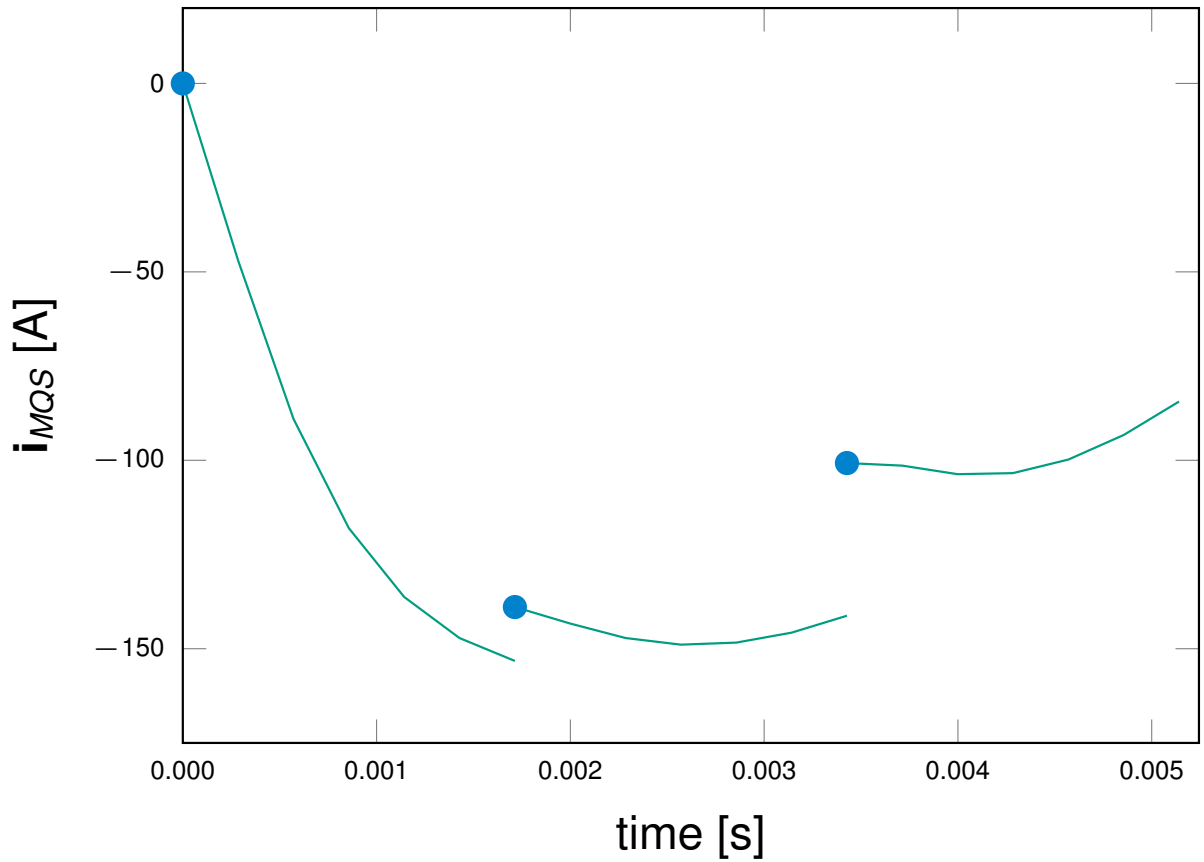
Animation of Micro-Macro Parareal using a Toy Example

Fine:
WR iteration 2



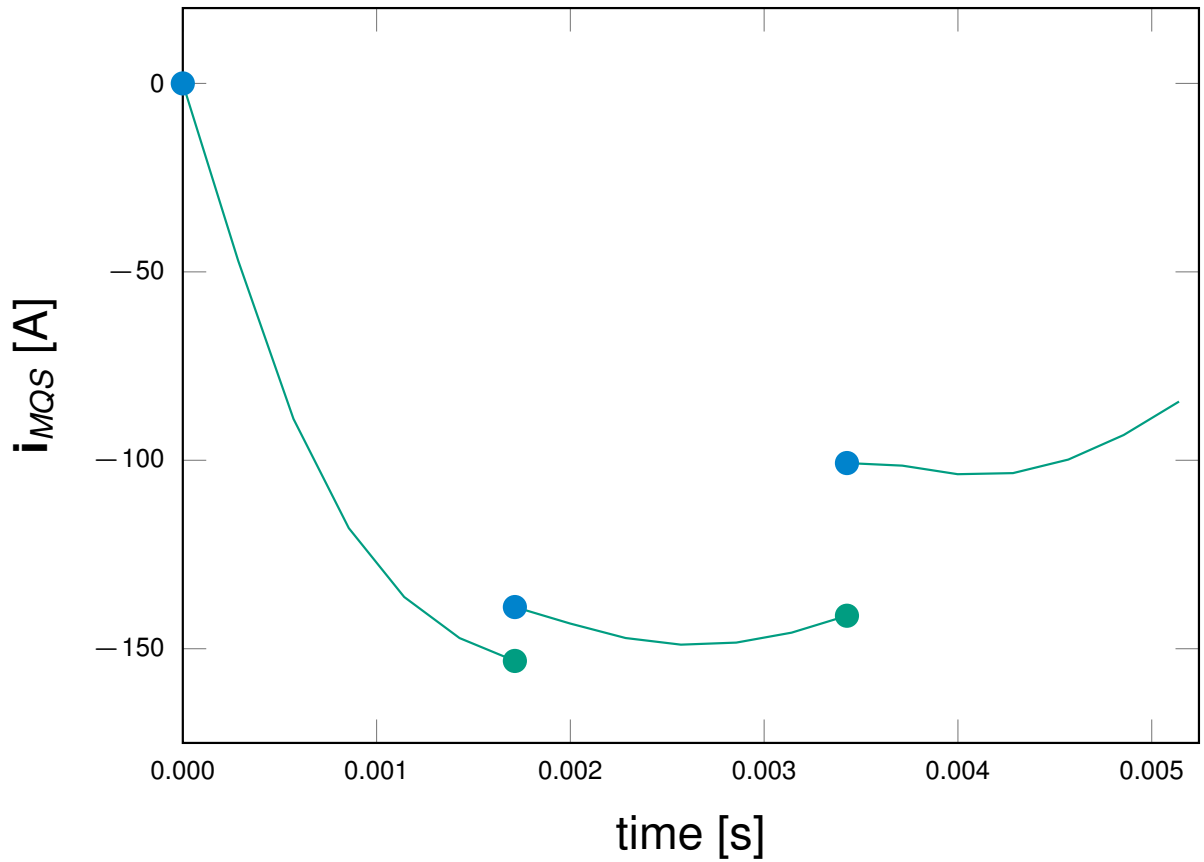
Animation of Micro-Macro Parareal using a Toy Example

Fine:
WR converged



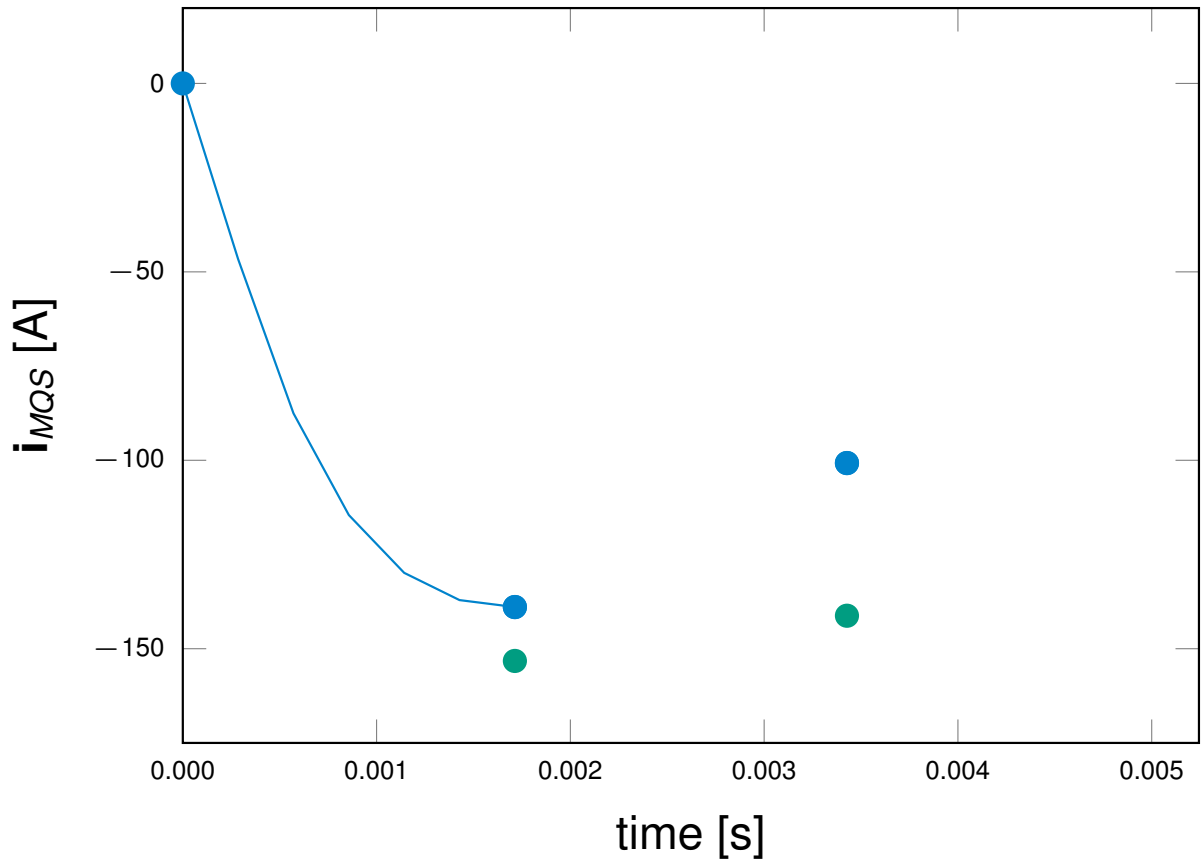
Animation of Micro-Macro Parareal using a Toy Example

Fine:
get circuit variables



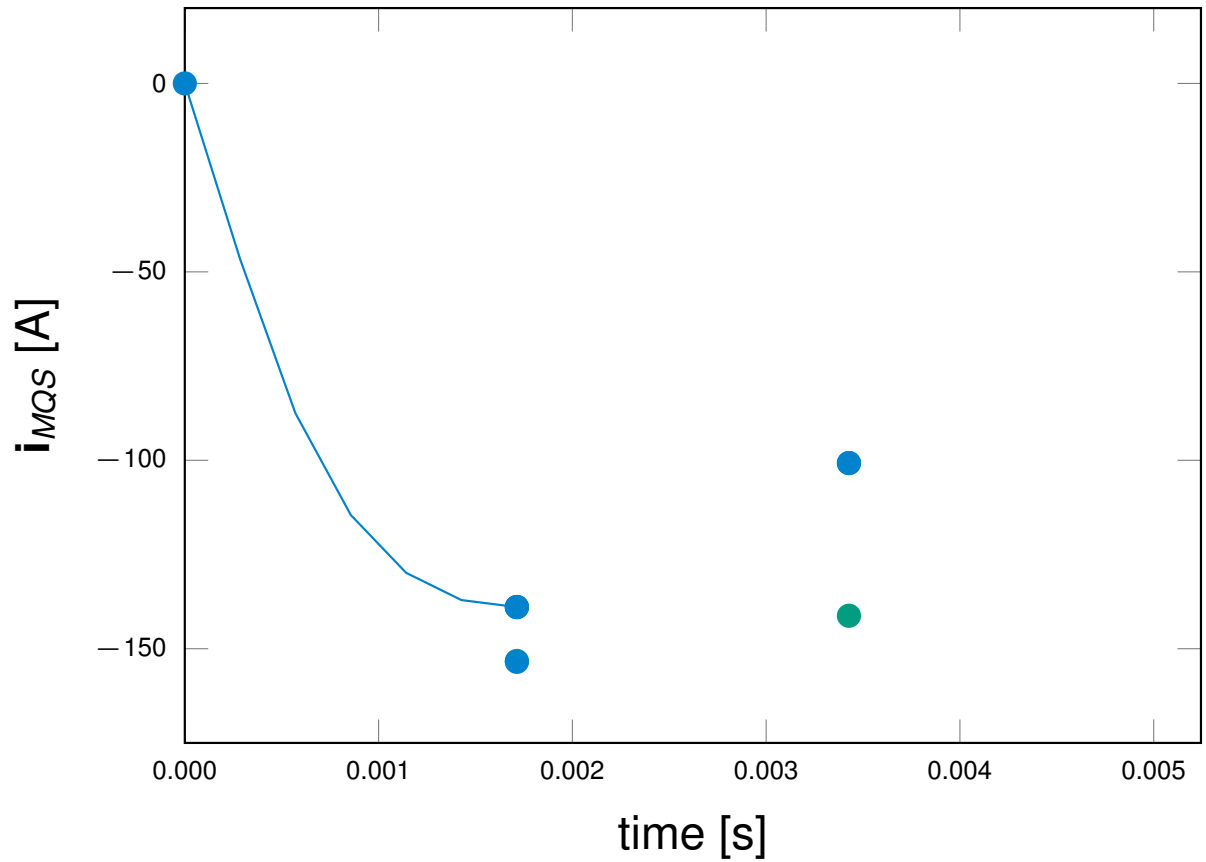
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



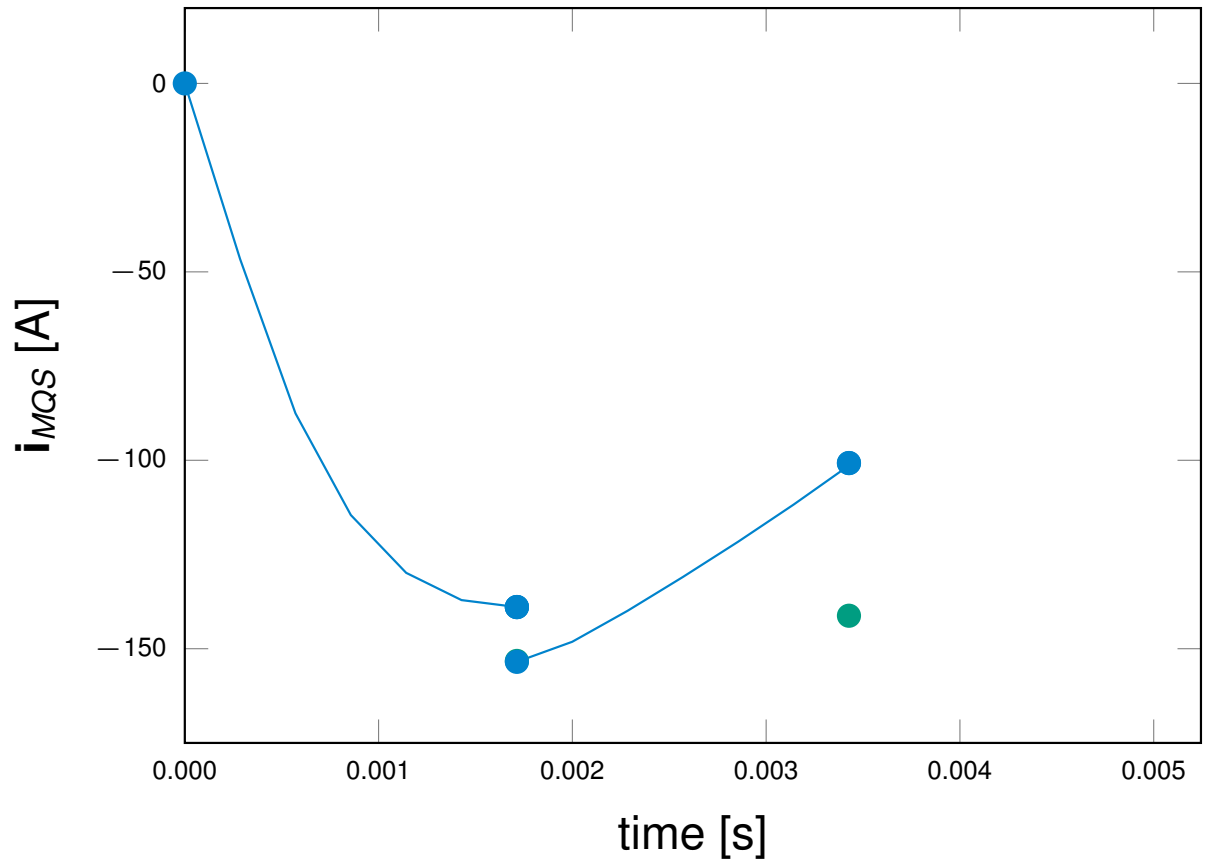
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



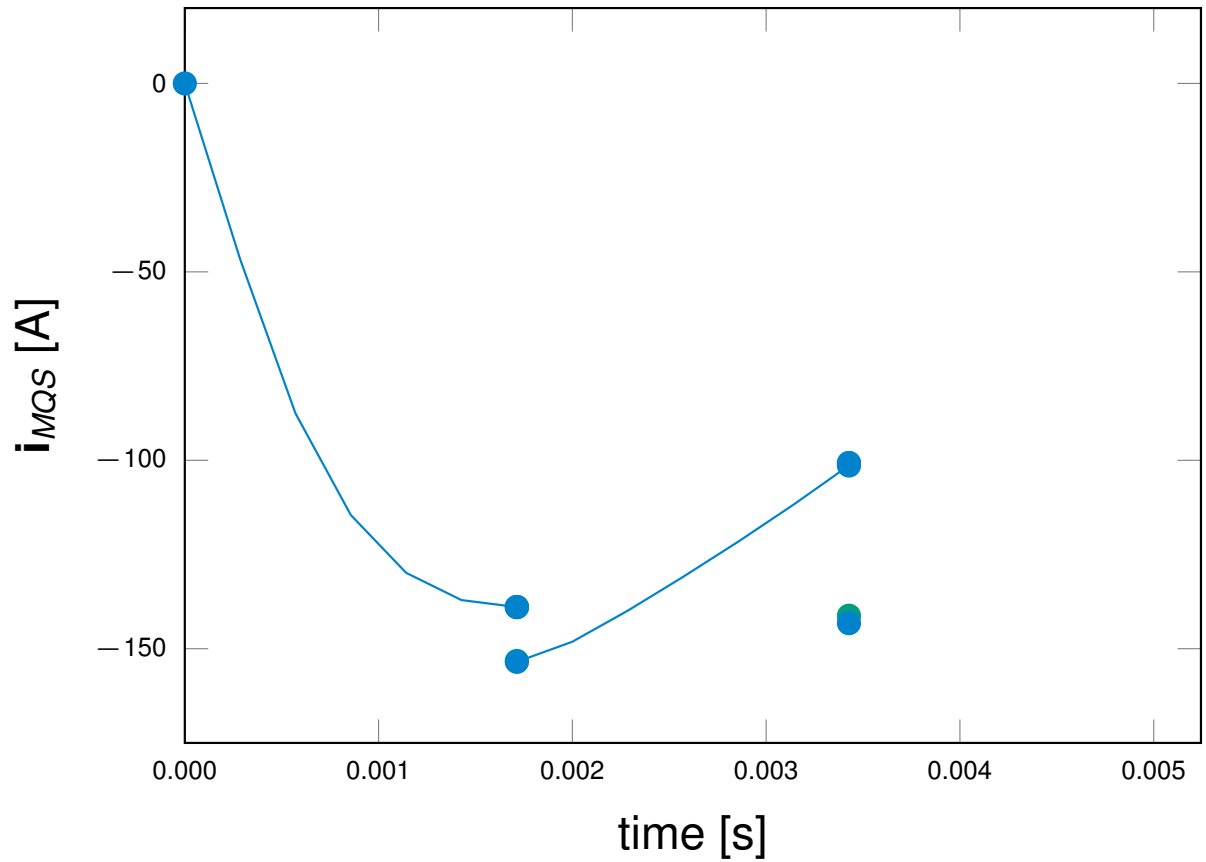
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



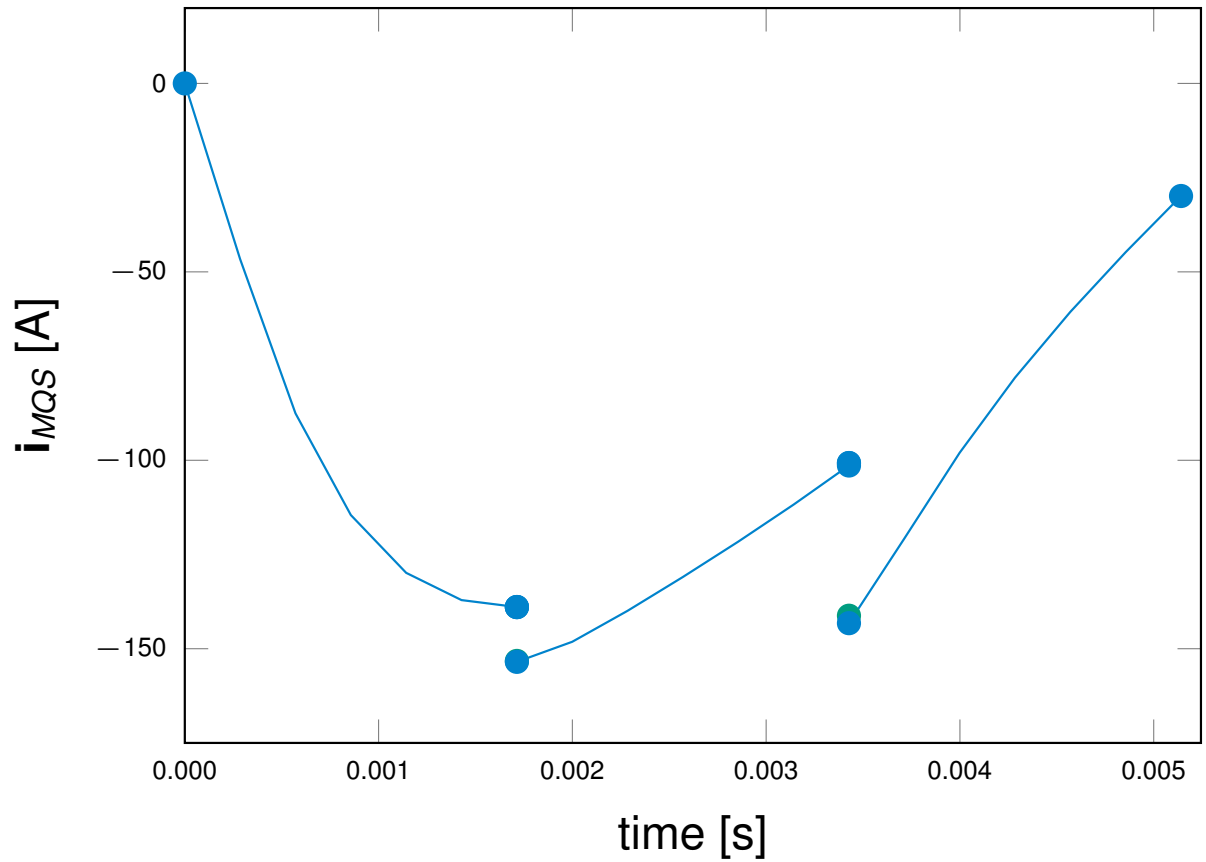
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



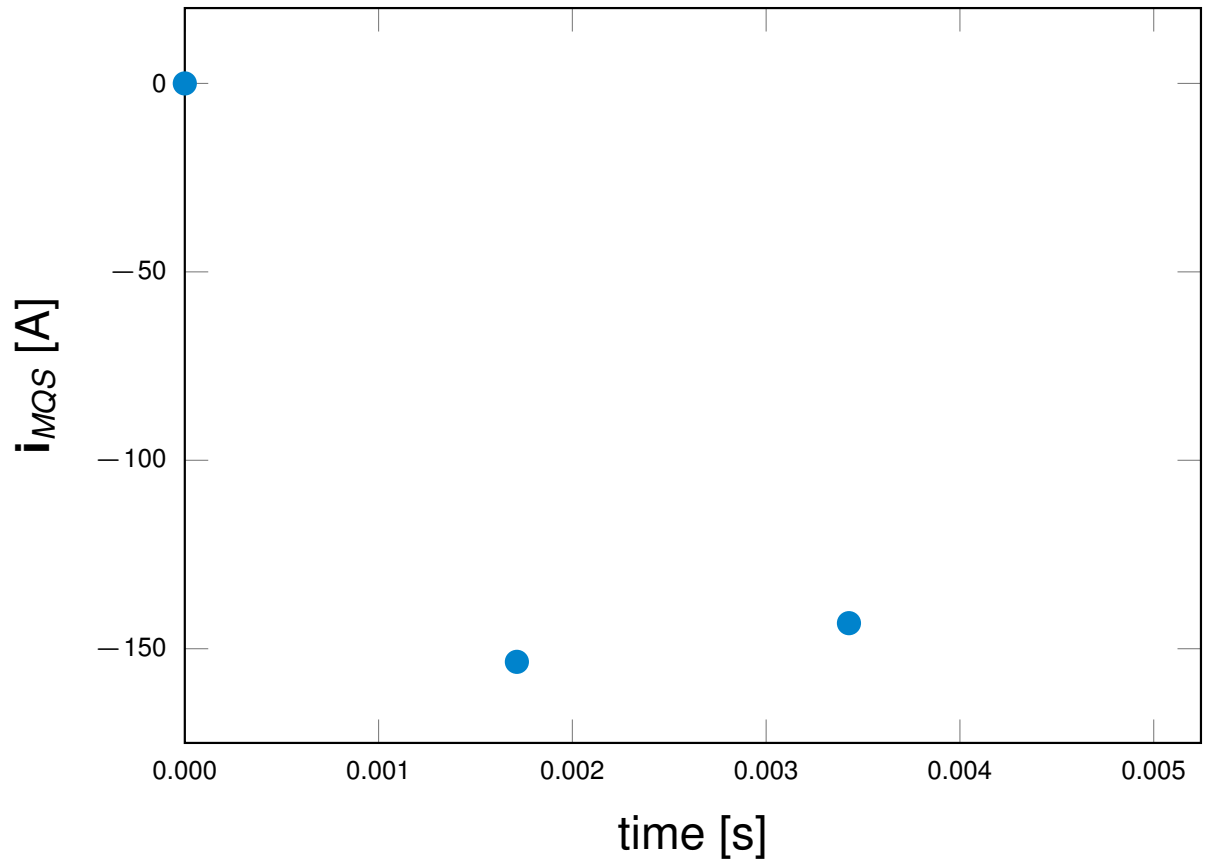
Animation of Micro-Macro Parareal using a Toy Example

Coarse:
circuit solver



Animation of Micro-Macro Parareal using a Toy Example

Fine:
new initial values
...start WR again...



Outline

Summary and Outlook

- Micro-Macro Parareal seems to work well for field/circuit coupled problems
- The only additional burden is the solution of static system for the initial values
  - this can be parallelized
  - the system matrix is the same as used for the inductance extraction
- there are currently two nested iterations for Parareal (outer loop) and waveform relaxation (inner loop)
- initial tests by the student group show that one can probably get rid of the inner loop, such that Parareal takes care of the convergence
- in this combined setting, convergence cannot be ensured anymore after n iterations (if n is the number of time windows)