The parallel full-approximation-scheme in space and time for electromagnetic transient simulation

Lisa Wimmer

University of Wuppertal

July 11, 2022

Contents

$v(t)$
x
$\frac{t}{dt}v(t) = f_{DC}(t, v(t))$
$\frac{t}{dt}v(t) = f_{DC}(t, v(t))$
DC/DC
DC/DC
$\frac{t}{dt}v(t) = f_{Piline}(t, v(t))$
t

Part I

Introduction

- Parallel-full-approximation scheme in space and time (PFASST)
- PinTSimE project

SDC - The problem[1]

Consider the initial value problem

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t))$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \underset{\text{over } t}{\overset{\text{integrate}}{\Longrightarrow}}$$

$$u(t_0) = u_0, \quad t \in [0, T]$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \overset{\text{integrate}}{\underset{\text{over } t}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \underset{\text{over } t}{\overset{\text{integrate}}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \quad \overset{\text{integrate}}{\underset{\text{over } t}{\Rightarrow}}$$

$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f\left(\tau, u(\tau)\right) d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \overset{\text{integrate}}{\underset{\text{over } t}{\Rightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \quad \overset{\text{integrate}}{\underset{\text{over } t}{\Rightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \tag{1}$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \quad \underset{\text{over } t}{\overset{\text{integrate}}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f\left(\tau, u(\tau)\right) d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$\left(\mathbf{I_M} - \Delta t \mathbf{QF}\right)\left(\mathbf{u}_{l+1}\right) = \mathbf{u}_l \tag{1}$$

Collocation problem for time domain $[0, T]$:

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \overset{\text{integrate}}{\underset{\text{over } t}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f\left(\tau, u(\tau)\right) d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$\left(\mathbf{I_M} - \Delta t \mathbf{QF}\right)\left(\mathbf{u}_{l+1}\right) = \mathbf{u}_l \qquad (1)$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}\left(\vec{\mathbf{u}}\right) := \left(\qquad \right)\left(\quad \right) = \left(\quad \right) \qquad (2)$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \quad \underset{\text{over } t}{\overset{\text{integrate}}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \tag{1}$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}(\vec{\mathbf{u}}) := \begin{pmatrix} \mathbf{I_M} - \Delta t \mathbf{QF} & & \\ & & \\ & & \end{pmatrix} \begin{pmatrix} \mathbf{u_1} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \mathbf{u_0} \\ \\ \\ \end{pmatrix} \tag{2}$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \underset{\text{over } t}{\overset{\text{integrate}}{\Rightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \qquad (1)$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}(\vec{\mathbf{u}}) := \begin{pmatrix} \mathbf{I_M} - \Delta t \mathbf{QF} & & \\ & \mathbf{I_M} - \Delta t \mathbf{QF} & \\ & & \\ & & \end{pmatrix} \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \mathbf{u_0} \\ \mathbf{0} \\ \\ \\ \end{pmatrix} \qquad (2)$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \underset{\text{over } t}{\overset{\text{integrate}}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \tag{1}$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}(\vec{\mathbf{u}}) := \begin{pmatrix} \mathbf{I_M} - \Delta t \mathbf{QF} & & \\ -\mathbf{H} & \mathbf{I_M} - \Delta t \mathbf{QF} & \\ & & \\ & & \end{pmatrix} \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \mathbf{u_0} \\ \mathbf{0} \\ \\ \\ \end{pmatrix} \tag{2}$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \quad \overset{\text{integrate}}{\underset{\text{over } t}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \tag{1}$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}(\vec{\mathbf{u}}) := \begin{pmatrix} \mathbf{I_M} - \Delta t \mathbf{QF} & & & \\ -\mathbf{H} & \mathbf{I_M} - \Delta t \mathbf{QF} & & \\ & \ddots & \ddots & \\ & & & \end{pmatrix} \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{u_0} \\ \mathbf{0} \\ \vdots \end{pmatrix} \tag{2}$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \qquad \overset{\text{integrate}}{\underset{\text{over } t}{\Longrightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \tag{1}$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}(\vec{\mathbf{u}}) := \begin{pmatrix} \mathbf{I_M} - \Delta t\mathbf{QF} & & & \\ -\mathbf{H} & \mathbf{I_M} - \Delta t\mathbf{QF} & & \\ & \ddots & \ddots & \\ & & -\mathbf{H} & \mathbf{I_M} - \Delta t\mathbf{QF} \end{pmatrix} \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \vdots \\ \mathbf{u_L} \end{pmatrix} = \begin{pmatrix} \mathbf{u_0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \tag{2}$$

SDC - The problem[1]

Consider the initial value problem

$$\frac{d}{dt}u(t) = f(t, u(t)) \quad \overset{\text{integrate}}{\underset{\text{over } t}{\Rightarrow}}$$
$$u(t_0) = u_0, \quad t \in [0, T]$$

Picard form over $[t_l, t_{l+1}]$, $l = 0..L-1$:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\, d\tau$$

for partition $0 = t_0 < .. < t_L = T$.

Discretization using quadrature weights leads to collocation problem over $[t_l, t_{l+1}]$:

$$(\mathbf{I_M} - \Delta t \mathbf{QF})(\mathbf{u}_{l+1}) = \mathbf{u}_l \tag{1}$$

Collocation problem for time domain $[0, T]$:

$$\mathbf{C}(\vec{\mathbf{u}}) := \begin{pmatrix} \mathbf{I_M} - \Delta t\mathbf{QF} & & & \\ -\mathbf{H} & \mathbf{I_M} - \Delta t\mathbf{QF} & & \\ & \ddots & \ddots & \\ & & -\mathbf{H} & \mathbf{I_M} - \Delta t\mathbf{QF} \end{pmatrix} \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \vdots \\ \mathbf{u_L} \end{pmatrix} = \begin{pmatrix} \mathbf{u_0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \tag{2}$$

[1] *M. Emmett, M. L. Minion, Toward an efficient parallel in time method for partial differential equations, Commun. Appl. Math. Comp. Sc. 7, no. 1, 105.132, 2012*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial):

Collocation problem - The iterative solvers[1,2]

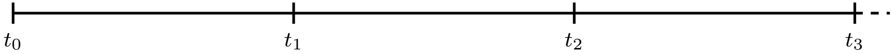
Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$

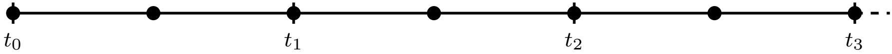
Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_{L} \otimes \Delta t \mathbf{Q}_{\Delta}\mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



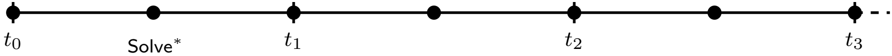
Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$

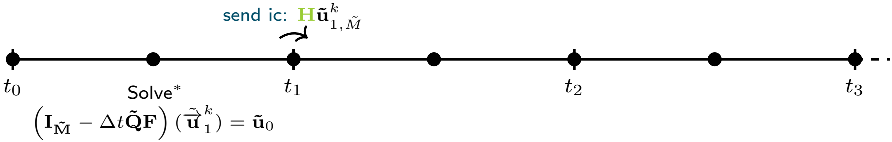


$$\left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}_1^{\,k}\right) = \tilde{\mathbf{u}}_0$$

*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

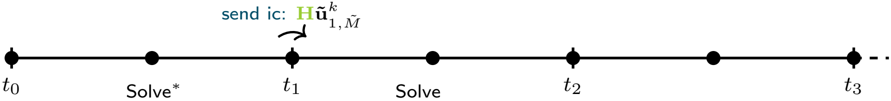
Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \color{green}{\mathbf{H}}\right)\left(\vec{\mathbf{u}}\right)$

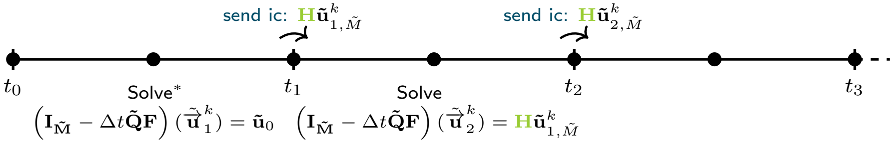


$$\left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\vec{\tilde{\mathbf{u}}}^k_1\right) = \tilde{\mathbf{u}}_0 \quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\vec{\tilde{\mathbf{u}}}^k_2\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{1,\tilde{M}}$$

*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

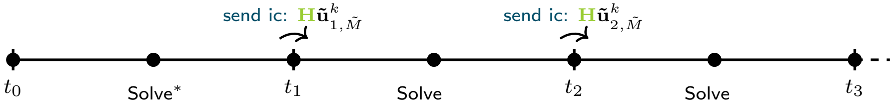
Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$

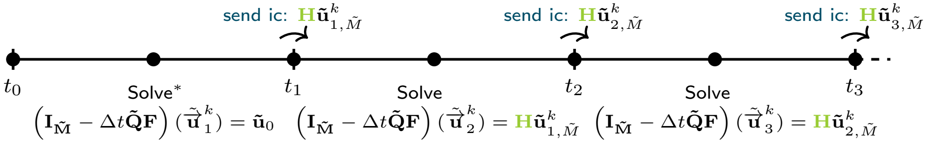


$$\left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}^k_1\right) = \tilde{\mathbf{u}}_0 \quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}^k_2\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{1,\tilde{M}} \quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}^k_3\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{2,\tilde{M}}$$

*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

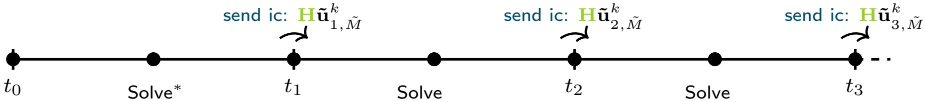
Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



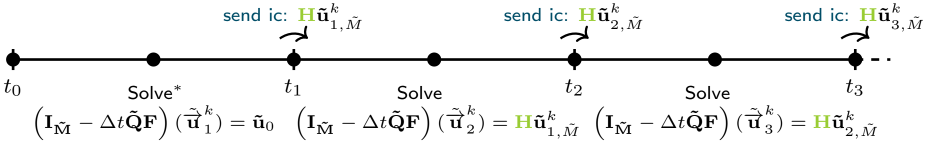
$$\left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}^k_1\right) = \tilde{\mathbf{u}}_0 \quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}^k_2\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{1,\tilde{M}} \quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\tilde{\vec{\mathbf{u}}}^k_3\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{2,\tilde{M}}$$

Jacobi on fine level (parallel):

*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$

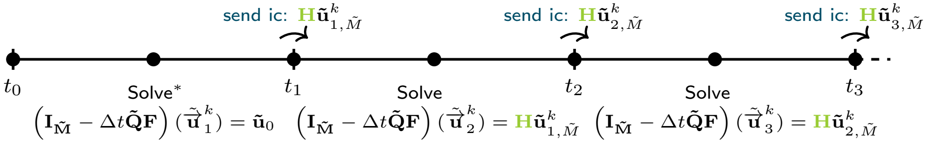


Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$

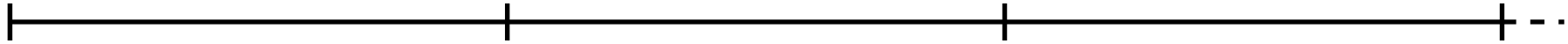
*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



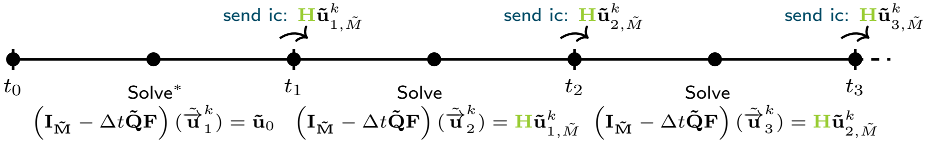
Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$



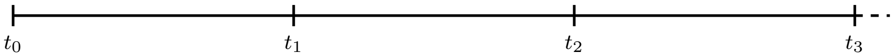
*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



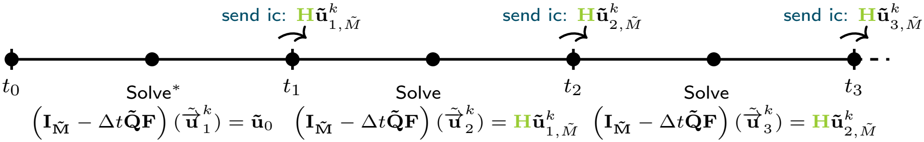
Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$



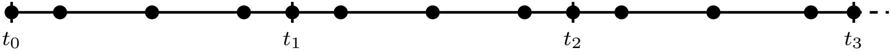
**: a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



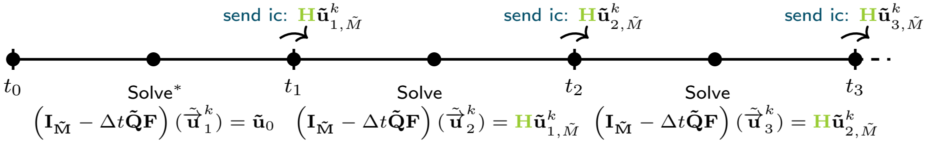
Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$



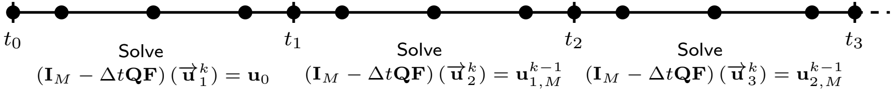
**: a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



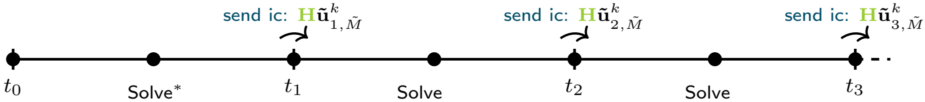
Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$



*: *a tilde marks values on coarse level*

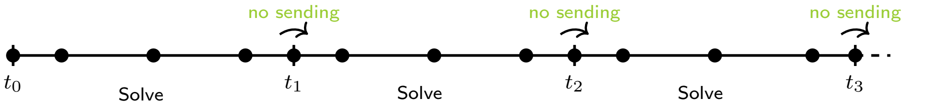
Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



$\left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\vec{\tilde{\mathbf{u}}}^k_1\right) = \tilde{\mathbf{u}}_0$ $\quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\vec{\tilde{\mathbf{u}}}^k_2\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{1,\tilde{M}}$ $\quad \left(\mathbf{I}_{\tilde{\mathbf{M}}} - \Delta t \tilde{\mathbf{Q}}\mathbf{F}\right)\left(\vec{\tilde{\mathbf{u}}}^k_3\right) = \mathbf{H}\tilde{\mathbf{u}}^k_{2,\tilde{M}}$

Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$

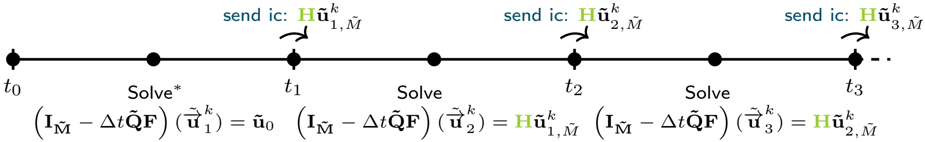


$\left(\mathbf{I}_M - \Delta t \mathbf{Q}\mathbf{F}\right)\left(\vec{\mathbf{u}}^k_1\right) = \mathbf{u}_0$ $\quad \left(\mathbf{I}_M - \Delta t \mathbf{Q}\mathbf{F}\right)\left(\vec{\mathbf{u}}^k_2\right) = \mathbf{u}^{k-1}_{1,M}$ $\quad \left(\mathbf{I}_M - \Delta t \mathbf{Q}\mathbf{F}\right)\left(\vec{\mathbf{u}}^k_3\right) = \mathbf{u}^{k-1}_{2,M}$

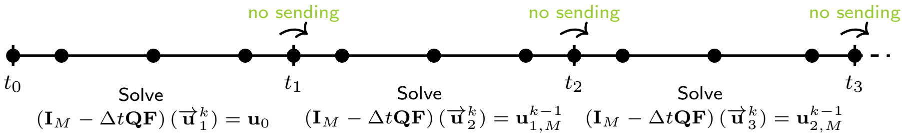
*: *a tilde marks values on coarse level*

Collocation problem - The iterative solvers[1,2]

Gauß-Seidel on coarse level (serial): $P_{GS}\left(\vec{\mathbf{u}}\right) = \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F} - \mathbf{E} \otimes \mathbf{H}\right)\left(\vec{\mathbf{u}}\right)$



Jacobi on fine level (parallel): $P_J\left(\vec{\mathbf{u}}\right) := \left(\mathbf{I}_{LM} - \mathbf{I}_L \otimes \Delta t \mathbf{Q}_\Delta \mathbf{F}\right)\left(\vec{\mathbf{u}}\right)$



*: *a tilde marks values on coarse level*

[2]: *M. Bolten, D. Moser, R. Speck, A multigrid perspective on the parallel full approximation scheme in space and time, Numerical Linear Algebra with Applications 24, 6, 2017*

PFASST Algorithm[1,2]

1. Execute Jacobi solver on fine level in parallel using P_J

PFASST Algorithm[1,2]

1. Execute Jacobi solver on fine level in parallel using P_J
2. Compute τ-correction

PFASST Algorithm[1,2]

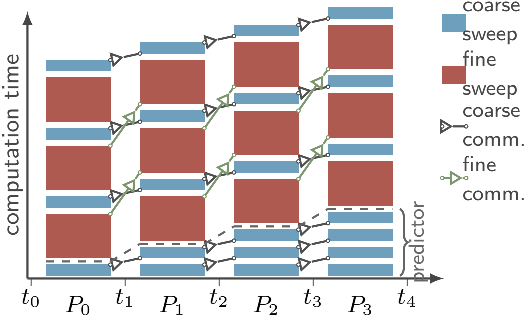
1. Execute Jacobi solver on fine level in parallel using P_J
2. Compute τ-correction
3. Execute Gauß-Seidel solver on coarse level in serial using P_{GS}

PFASST Algorithm[1,2]

1. Execute Jacobi solver on fine level in parallel using P_J
2. Compute τ-correction
3. Execute Gauß-Seidel solver on coarse level in serial using P_{GS}
4. Correct solution

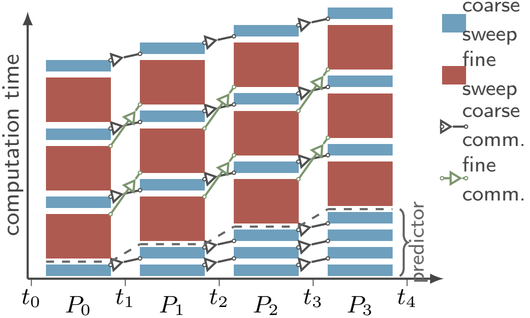
PFASST Algorithm[1,2]

1. Execute Jacobi solver on fine level in parallel using P_J
2. Compute τ-correction
3. Execute Gauß-Seidel solver on coarse level in serial using P_{GS}
4. Correct solution



PFASST Algorithm[1,2]

1. Execute Jacobi solver on fine level in parallel using P_J
2. Compute τ-correction
3. Execute Gauß-Seidel solver on coarse level in serial using P_{GS}
4. Correct solution



[3]: *Fabian Koehler, PFASST tikz, https://github.com/Parallel-in-Time/pfasst-tikz, 2015*

DFG Project - PinTSimE

Members:

Prof. Dr.
Matthias Bolten

(BUW)

Dr.
Robert Speck

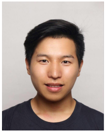
(JSC)

Prof. Dr.-Ing.
Andrea Benigni

(FZJ IEK-10)

M. Sc.
Julius Strake

(FZJ IEK-10)



M. Sc.
Junjie Zhang

(FZJ IEK-10)

DFG Project - PinTSimE

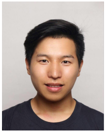
Members:

Prof. Dr.
Matthias Bolten
(BUW)

Dr.
Robert Speck
(JSC)

Prof. Dr.-Ing.
Andrea Benigni
(FZJ IEK-10)

M. Sc.
Julius Strake
(FZJ IEK-10)



M. Sc.
Junjie Zhang
(FZJ IEK-10)

- PinTSimE - Parallel-in-Time Simulation in multimodal Energy systems

DFG Project - PinTSimE

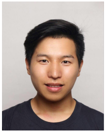
Members:

Prof. Dr.
Matthias Bolten
(BUW)

Dr.
Robert Speck
(JSC)

Prof. Dr.-Ing.
Andrea Benigni
(FZJ IEK-10)

M. Sc.
Julius Strake
(FZJ IEK-10)



M. Sc.
Junjie Zhang
(FZJ IEK-10)

- PinTSimE - Parallel-in-Time Simulation in multimodal Energy systems
- Multimodal - Energy system in which different energy grids *are connected* such as power grid, heat grid, gas grid,..

DFG Project - PinTSimE

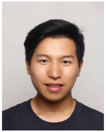
Members:


Prof. Dr.
Matthias Bolten
(BUW)


Dr.
Robert Speck
(JSC)


Prof. Dr.-Ing.
Andrea Benigni
(FZJ IEK-10)


M. Sc.
Julius Strake
(FZJ IEK-10)


M. Sc.
Junjie Zhang
(FZJ IEK-10)

- PinTSimE - Parallel-in-Time Simulation in multimodal Energy systems
- Multimodal - Energy system in which different energy grids *are connected* such as power grid, heat grid, gas grid,..
- First focus lies on DC power microgrid

DFG Project - PinTSimE

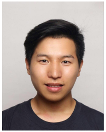
Members:

Prof. Dr.
Matthias Bolten
(BUW)

Dr.
Robert Speck
(JSC)

Prof. Dr.-Ing.
Andrea Benigni
(FZJ IEK-10)

M. Sc.
Julius Strake
(FZJ IEK-10)



M. Sc.
Junjie Zhang
(FZJ IEK-10)

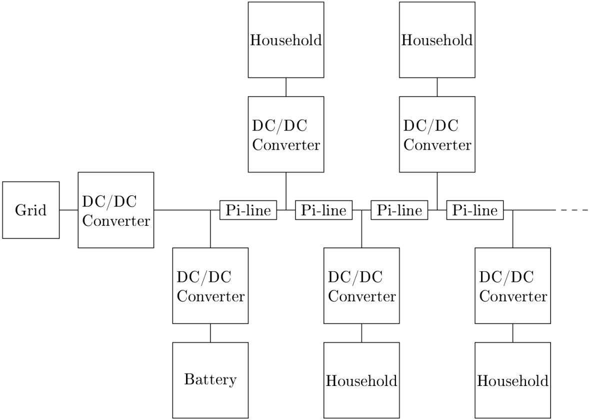
- PinTSimE - Parallel-in-Time Simulation in multimodal Energy systems
- Multimodal - Energy system in which different energy grids *are connected* such as power grid, heat grid, gas grid,..
- First focus lies on DC power microgrid
- Goal: Simulation faster than real-time

Part II

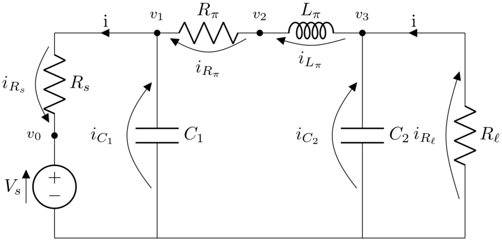
DC microgrid model

- Components
- Switching processes

DC microgrid

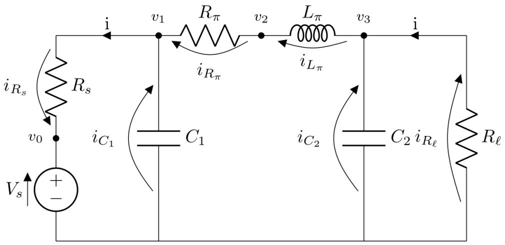


Pi-line model



Pi-line model

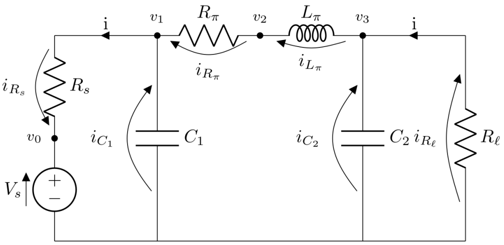
Pi-line model



Pi-line model

Model can be described as
ODE System:

Pi-line model

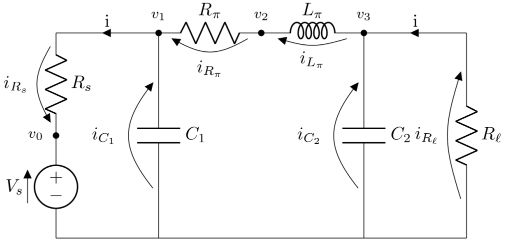


Pi-line model

Model can be described as ODE System:

$$\frac{d}{dt}u(t) = Au(t) + f(t)$$

Pi-line model



Pi-line model

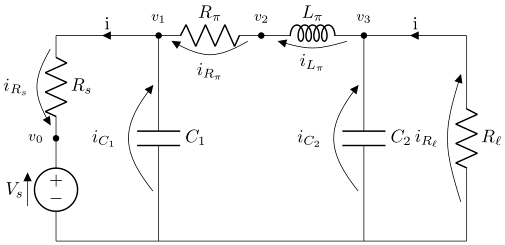
Model can be described as

ODE System:

$$\frac{d}{dt}u(t) = Au(t) + f(t)$$

Properties:

Pi-line model



Pi-line model

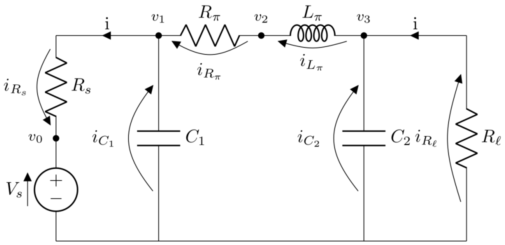
Model can be described as ODE System:

$$\frac{d}{dt}u(t) = Au(t) + f(t)$$

Properties:

- serves as transmission line in DC microgrid

Pi-line model



Pi-line model

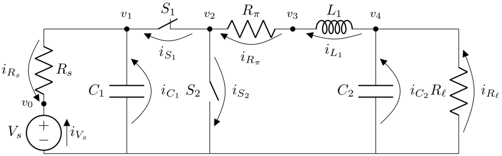
Model can be described as ODE System:

$$\frac{d}{dt}u(t) = Au(t) + f(t)$$

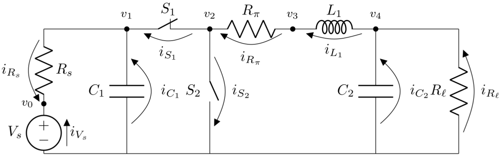
Properties:

- serves as transmission line in DC microgrid
- no special events in the simulation

Buck converter model

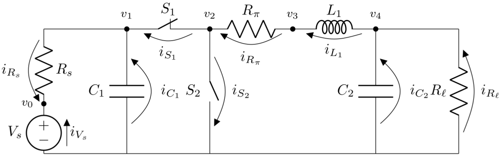


Buck converter model



Buck converter model with the states

Buck converter model

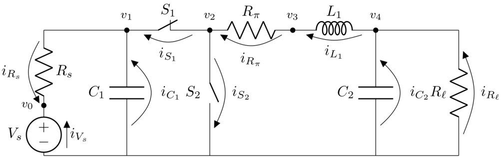


Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

Buck converter model



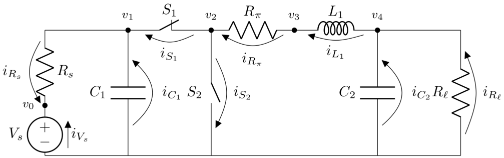
Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

$$SS = \{S_1 = 0, S_2 = 1\}$$

Buck converter model



Each state can be described as ODE System:

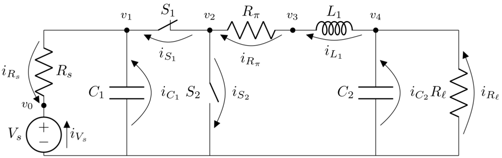
Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

$$SS = \{S_1 = 0, S_2 = 1\}$$

Buck converter model



Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

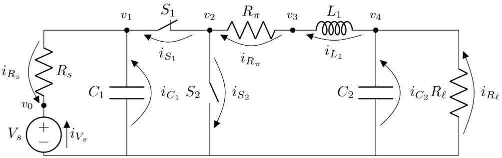
$$SS = \{S_1 = 0, S_2 = 1\}$$

Each state can be described as ODE System:

FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

Buck converter model



Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

$$SS = \{S_1 = 0, S_2 = 1\}$$

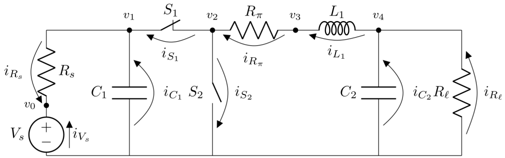
Each state can be described as ODE System:

FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

Properties:

Buck converter model



Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

$$SS = \{S_1 = 0, S_2 = 1\}$$

Each state can be described as ODE System:

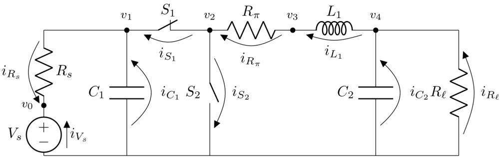
$$\text{FS: } \frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$$

$$\text{SS: } \frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$$

Properties:

- converts large input voltage in a smaller output voltage

Buck converter model



Buck converter model with the states

$$FS = \{S_1 = 1, S_2 = 0\}$$

and

$$SS = \{S_1 = 0, S_2 = 1\}$$

Each state can be described as ODE System:

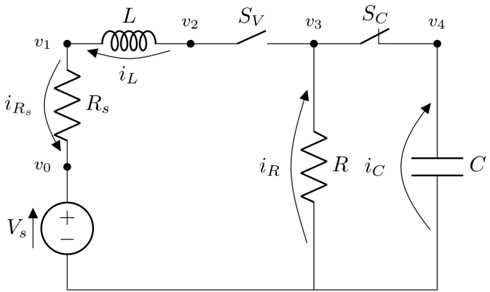
FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

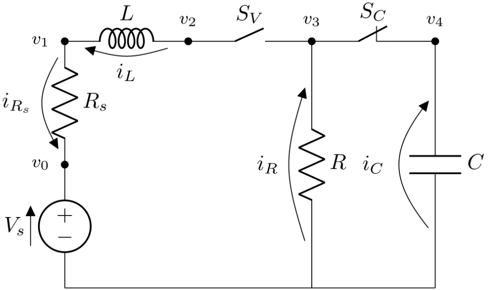
Properties:

- converts large input voltage in a smaller output voltage
- switching produces many discrete events

Battery drain model

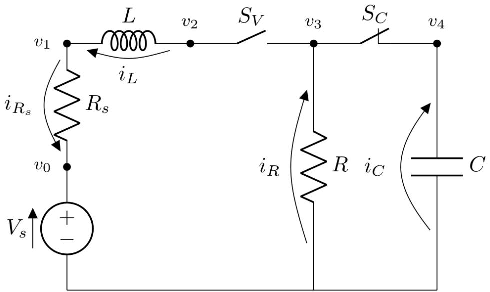


Battery drain model



Battery model with the states

Battery drain model

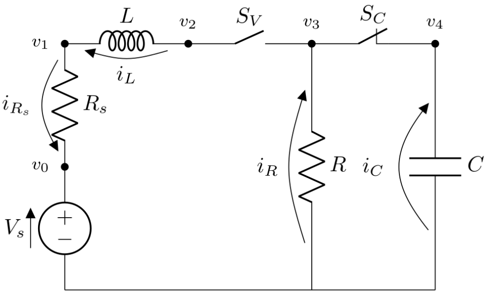


Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

Battery drain model



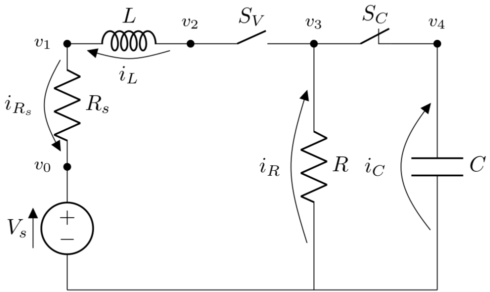
Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

Battery drain model



Each state can be described as ODE System:

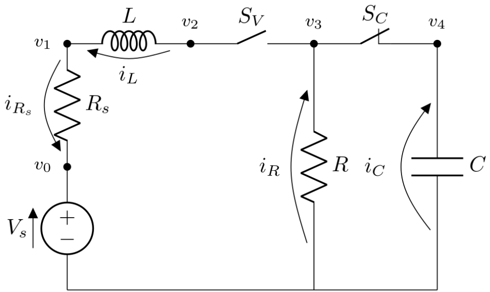
Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

Battery drain model



Each state can be described as ODE System:

$$\text{FS: } \frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$$

$$\text{SS: } \frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$$

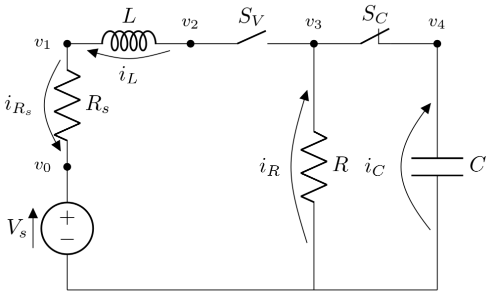
Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

Battery drain model



Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

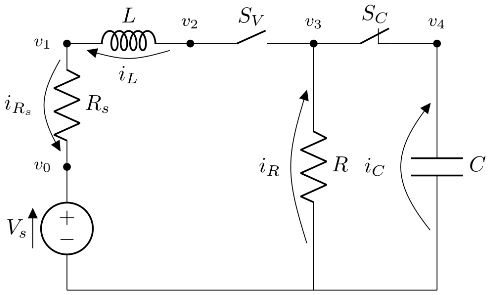
Each state can be described as ODE System:

FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

Properties:

Battery drain model



Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

Each state can be described as ODE System:

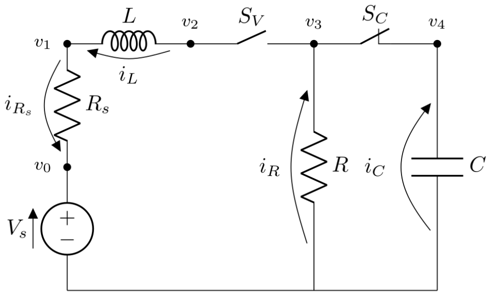
FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

Properties:

- capacitor supplies energy

Battery drain model



Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

Each state can be described as ODE System:

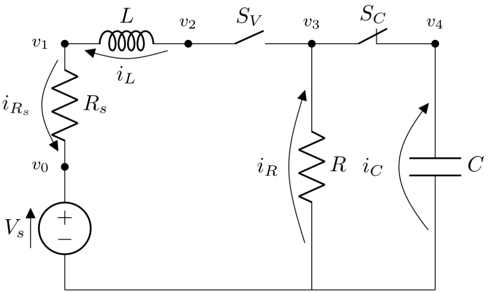
FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

Properties:

- capacitor supplies energy
- when energy drops below a reference voltage, switch to voltage source

Battery drain model



Battery model with the states

$$FS = \{S_C = 1, S_V = 0\}$$

and

$$SS = \{S_C = 0, S_V = 1\}$$

Each state can be described as ODE System:

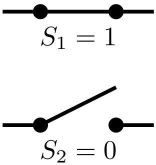
FS: $\frac{d}{dt}u(t) = A_1 u(t) + f_1(t)$

SS: $\frac{d}{dt}u(t) = A_2 u(t) + f_2(t)$

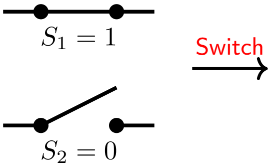
Properties:

- capacitor supplies energy
- when energy drops below a reference voltage, switch to voltage source
- only one single discrete event

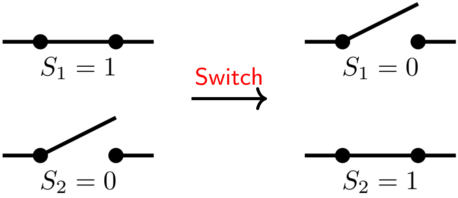
Switching processes



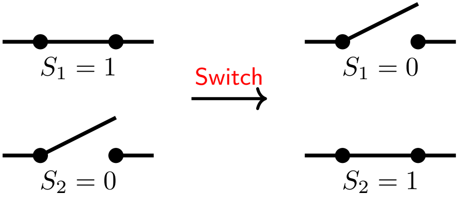
Switching processes



Switching processes

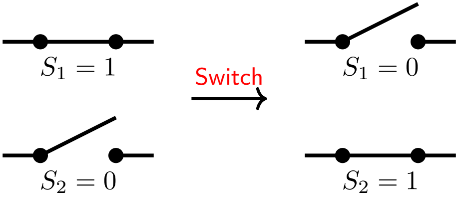


Switching processes



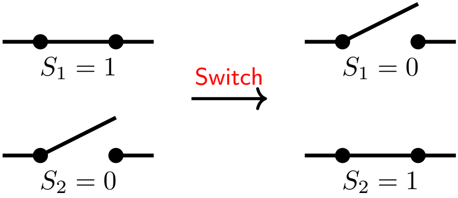
- discrete events

Switching processes



- discrete events
- for models: switching from one ODE system to another

Switching processes



- discrete events
- for models: switching from one ODE system to another
- needs to investigate their behavior in SDC (and PFASST)

Occurence of a switch

Two possibilities, where a switch can occur:

Occurence of a switch

Two possibilities, where a switch can occur:

- on a time step

Occurence of a switch

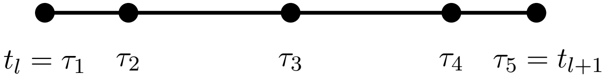
Two possibilities, where a switch can occur:

- on a time step
- between collocation nodes

Occurence of a switch

Two possibilities, where a switch can occur:

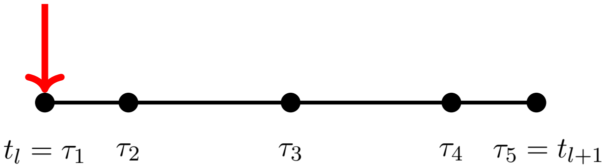
- on a time step
- between collocation nodes



Occurence of a switch

Two possibilities, where a switch can occur:

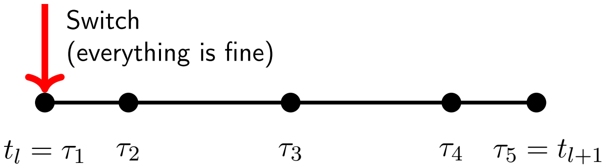
- on a time step
- between collocation nodes



Occurence of a switch

Two possibilities, where a switch can occur:

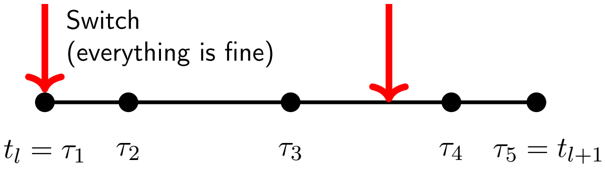
- on a time step
- between collocation nodes



Occurence of a switch

Two possibilities, where a switch can occur:

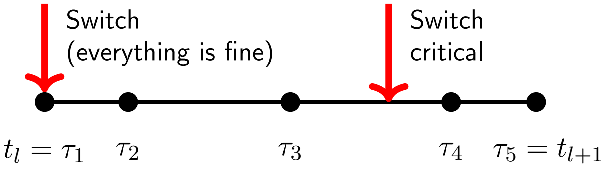
- on a time step
- between collocation nodes



Occurence of a switch

Two possibilities, where a switch can occur:

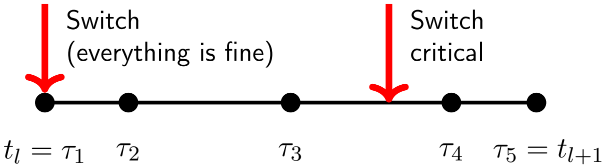
- on a time step
- between collocation nodes



Occurence of a switch

Two possibilities, where a switch can occur:

- on a time step
- between collocation nodes

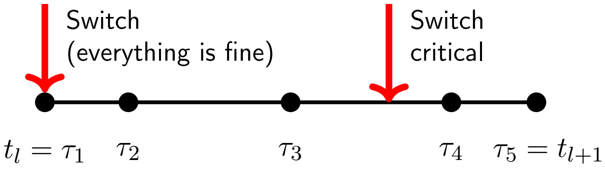


Challenges:

Occurence of a switch

Two possibilities, where a switch can occur:

- on a time step
- between collocation nodes



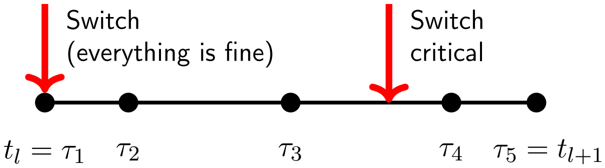
Challenges:

- Occuring of switch between collocation nodes leads to later resolution of behavior

Occurence of a switch

Two possibilities, where a switch can occur:

- on a time step
- between collocation nodes



Challenges:

- Occuring of switch between collocation nodes leads to later resolution of behavior
- Accuracy is worse: energy monitoring is delayed

Part III

Results

Pi-line modelling - SDC

Settings:

Pi-line modelling - SDC

Settings:

- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$

Pi-line modelling - SDC

Settings:

- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$
- IMEX SDC with 5 *Gauss-Lobatto (GL)* nodes used for simulation

Pi-line modelling - SDC

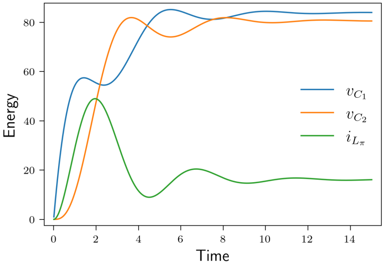
Settings:

- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$
- IMEX SDC with 5 *Gauss-Lobatto (GL)* nodes used for simulation
- local (per subinterval) $restol = 10^{-13}$

Pi-line modelling - SDC

Settings:

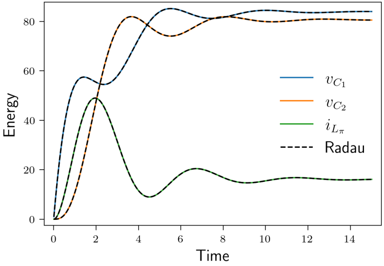
- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$
- IMEX SDC with 5 *Gauss-Lobatto (GL)* nodes used for simulation
- local (per subinterval) $restol = 10^{-13}$



Pi-line modelling - SDC

Settings:

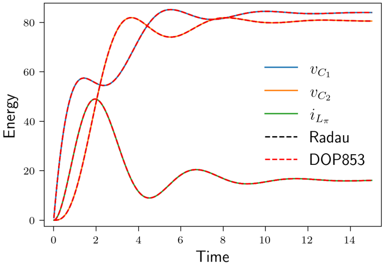
- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$
- IMEX SDC with 5 *Gauss-Lobatto (GL)* nodes used for simulation
- local (per subinterval) $restol = 10^{-13}$



Pi-line modelling - SDC

Settings:

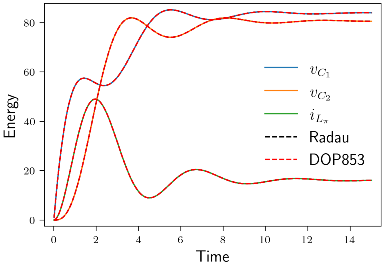
- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$
- IMEX SDC with 5 *Gauss-Lobatto (GL)* nodes used for simulation
- local (per subinterval) $restol = 10^{-13}$

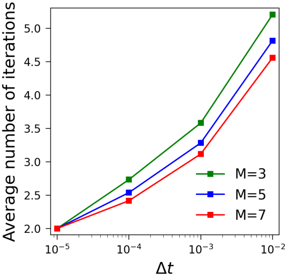


Pi-line modelling - SDC

Settings:

- $t_0 = 0$, $\Delta t = 10^{-2}$, $T = 15$
- IMEX SDC with 5 *Gauss-Lobatto (GL)* nodes used for simulation
- local (per subinterval) $restol = 10^{-13}$





Pi-line modelling - PFASST

Settings:

Pi-line modelling - PFASST

Settings:

- Same as for SDC, num_procs $= 8$

Pi-line modelling - PFASST

Settings:

- Same as for SDC,
  num_procs $= 8$
- Coarse level: 3 GL nodes
  Fine level: 5 GL nodes

Pi-line modelling - PFASST

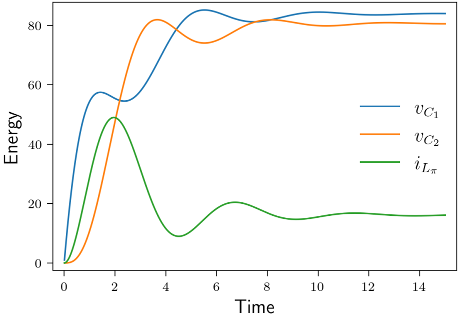
Settings:

- Same as for SDC,
  num_procs $= 8$
- Coarse level: 3 GL nodes
  Fine level: 5 GL nodes
- local $restol = 10^{-13}$
  (per subinterval)

Pi-line modelling - PFASST

Settings:

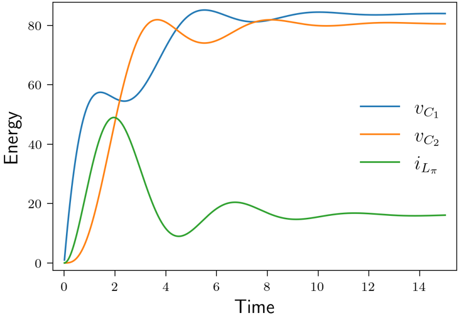
- Same as for SDC, num_procs $= 8$
- Coarse level: 3 GL nodes
  Fine level: 5 GL nodes
- local $restol = 10^{-13}$ (per subinterval)

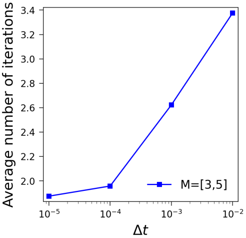


Pi-line modelling - PFASST

Settings:

- Same as for SDC, num_procs $= 8$
- Coarse level: 3 GL nodes
  Fine level: 5 GL nodes
- local $restol = 10^{-13}$ (per subinterval)

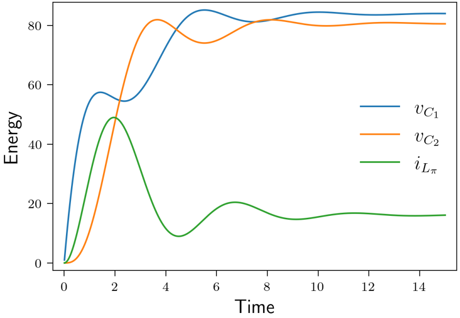


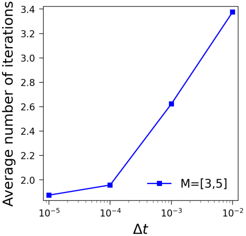


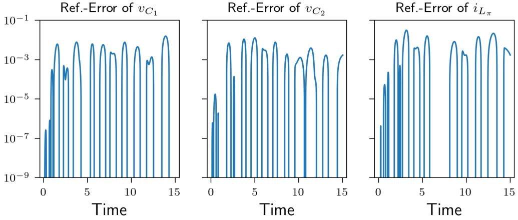
Pi-line modelling - PFASST

Settings:

- Same as for SDC, num_procs $= 8$
- Coarse level: 3 GL nodes
  Fine level: 5 GL nodes
- local $restol = 10^{-13}$ (per subinterval)







Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

- $t_0 = 0$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

- $t_0 = 0$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Setting:

- $t_0 = 0.0001$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

- $t_0 = 0$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Setting:

- $t_0 = 0.0001$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Setting:

- $t_0 = 0.00009$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

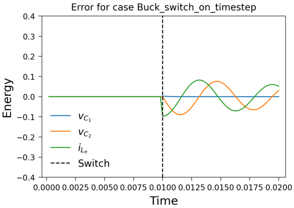
- $t_0 = 0$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Setting:

- $t_0 = 0.0001$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$

Setting:

- $t_0 = 0.00009$,
  $\Delta t = 2 \cdot 10^{-4}$,
  $T = 0.02$



Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

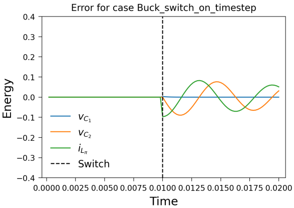
- $t_0 = 0$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$

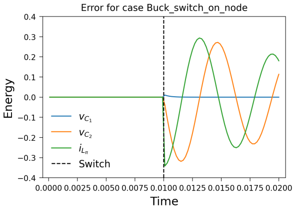
Setting:

- $t_0 = 0.0001$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$

Setting:

- $t_0 = 0.00009$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$





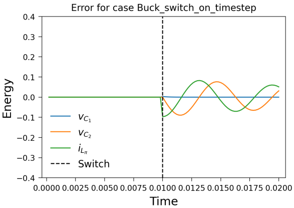
Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

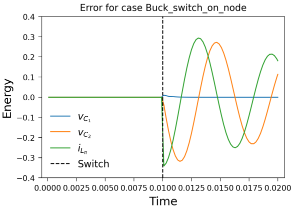
Setting:

- $t_0 = 0$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$



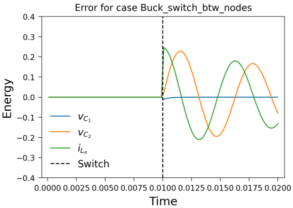
Setting:

- $t_0 = 0.0001$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$



Setting:

- $t_0 = 0.00009$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$



Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

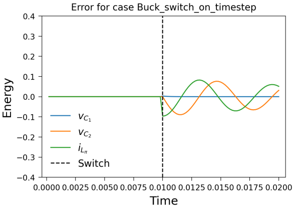
- $t_0 = 0$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$

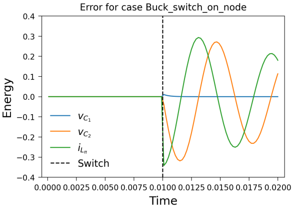
Setting:

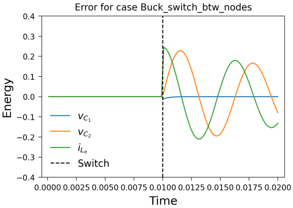
- $t_0 = 0.0001$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$

Setting:

- $t_0 = 0.00009$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$







- Error for case *Switch on timestep* relatively small

Buck converter modelling - SDC

Buck converter manipulation: Implementation of *only one switch*!

Reference solution via (IMEX) SDC: $t_0 = 0$, $\Delta t = 10^{-8}$, $T = 0.02$
Switch falls on time $t_{switch} = 0.01$
IMEX SDC with 5 *GL* nodes used

Setting:

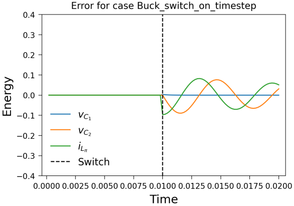
- $t_0 = 0$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$

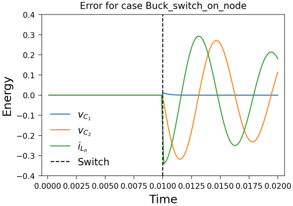
Setting:

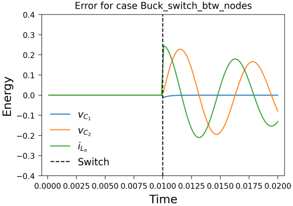
- $t_0 = 0.0001$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$

Setting:

- $t_0 = 0.00009$, $\Delta t = 2 \cdot 10^{-4}$, $T = 0.02$







- Error for case *Switch on timestep* relatively small
- Error for *other cases* show larger error

Battery drain modelling - SDC

Battery drain model has *internal switching*: Switching depends on system dynamics

Battery drain modelling - SDC

Battery drain model has *internal switching*: Switching depends on system dynamics

Settings:

Battery drain modelling - SDC

Battery drain model has *internal switching*: Switching depends on system dynamics

Settings:

- $t_0 = 0$, $\Delta t = 10^{-3}$, $T = 4$

Battery drain modelling - SDC

Battery drain model has *internal switching*: Switching depends on system dynamics

Settings:

- $t_0 = 0$, $\Delta t = 10^{-3}$, $T = 4$
- Reference value for *switch*: $V_{ref} = 1$

Battery drain modelling - SDC

Battery drain model has *internal switching*: Switching depends on system dynamics

Settings:

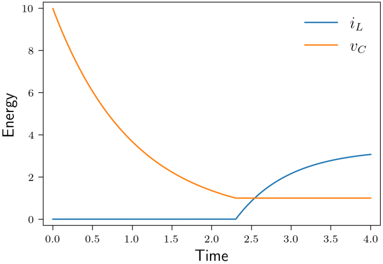
- $t_0 = 0$, $\Delta t = 10^{-3}$, $T = 4$
- Reference value for *switch*: $V_{ref} = 1$
- IMEX SDC with 5 *GL* nodes used

Battery drain modelling - SDC

Battery drain model has *internal switching*: Switching depends on system dynamics

Settings:

- $t_0 = 0$, $\Delta t = 10^{-3}$, $T = 4$
- Reference value for *switch*: $V_{ref} = 1$
- IMEX SDC with 5 *GL* nodes used



Part IV

Future work

Future work

- Development of switch estimator (first, for SDC - then, for PFASST)

Future work

- Development of switch estimator (first, for SDC - then, for PFASST)
- Further investigation of each microgrid component

Future work

- Development of switch estimator (first, for SDC - then, for PFASST)
- Further investigation of each microgrid component
- Parallel simulation of whole DC microgrid

Future work

- Development of switch estimator (first, for SDC - then, for PFASST)
- Further investigation of each microgrid component
- Parallel simulation of whole DC microgrid
  - Runtime tests for PFASST for each component

The end - pySDC & contact

- PintSimE project is already implemented in pySDC

The end - pySDC & contact

- PintSimE project is already implemented in pySDC
- Tutorial page is under construction

The end - pySDC & contact

- PintSimE project is already implemented in pySDC
- Tutorial page is under construction

If you have any (deeper) questions, contact me on
wimmer@math.uni-wuppertal.de

The end - pySDC & contact

- PintSimE project is already implemented in pySDC
- Tutorial page is under construction

If you have any (deeper) questions, contact me on
wimmer@math.uni-wuppertal.de

Any questions or comments to the talk?