Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

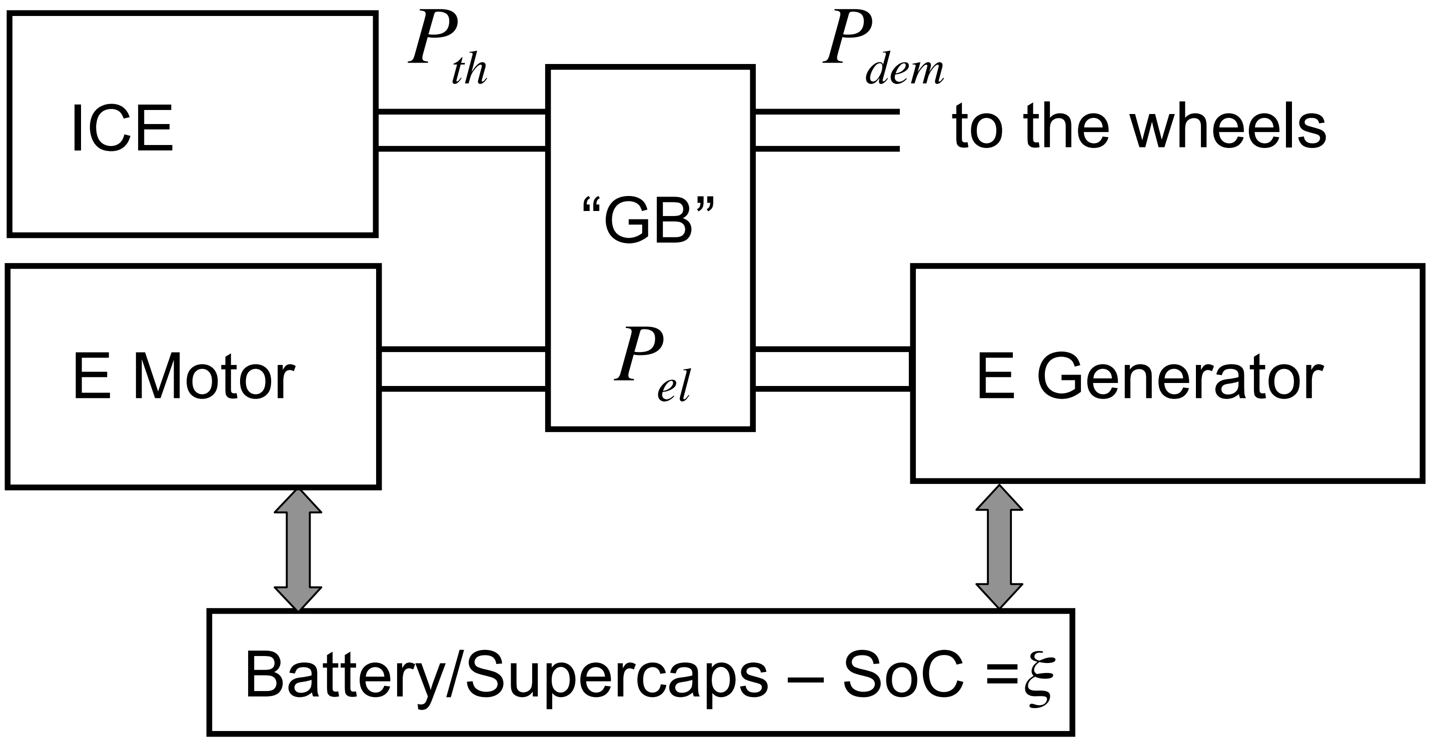
# The Role of System Theory in Reducing Energy Losses in Hybrids

Lino Guzzella
http://www.imrt.ethz.ch

20 minutes

June 12, 2008

# Hybrid-Electric Powertrains



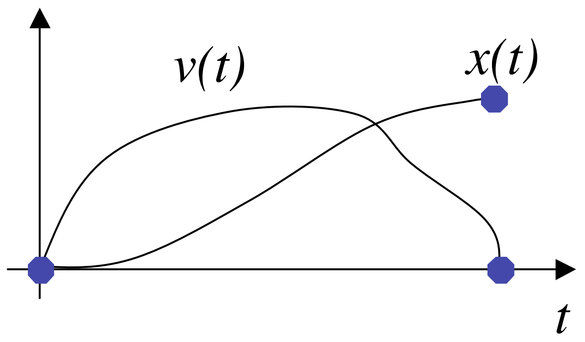

power-split factor $P_{dem} = u \cdot P_{el} + (1 - u) \cdot P_{th}$

# Pro Memoria: HEP Fuel-Economy Options

- Reduce idling losses (“friction”), i.e., engine shut-down at idle speed or zero torque
- Recuperate kinetic energy while braking
- Use “two engines”: one for acceleration, one for cruising (optimized part-load efficiency)
- Operate powertrain in duty-cycle and load-shift mode

» Load distribution not easy!

# Why "Two-Engines"




Vehicle $$\frac{d}{dt}x(t) = v(t)$$

Forward model $$\frac{d}{dt}v(t) = c_u \cdot u(t) - c_0 - c_2 \cdot v^2(t)$$

Control input $u \in [0, u_{\max}]$ (no braking assumed)

Objective function

$$J(u(t)) = \int_0^{t_f} \left(u(t) + u_0\right) \cdot v(t) \;\; dt$$

i.e., fuel consumption (using Willans approximation)

# Co-States and Hamiltonian

Costate
$$\frac{d}{dt}\lambda_1(t) = 0$$

$$\frac{d}{dt}\lambda_2(t) = -u(t) - \lambda_1(t) + 2 \cdot c_2 \cdot v(t) \cdot \lambda_2(t)$$

Hamiltonian

$$H = (u + u_0) \cdot v + \lambda_1 \cdot v + \lambda_2 \cdot \left(c_u \cdot u - c_0 - c_2 \cdot v^2\right)$$
$$= h_0(v, \lambda_1, \lambda_2) + \left(v + \lambda_2 \cdot c_u\right) \cdot u$$

Affine in the control! Minimum Principle yields “bang-bang” control with possible singular arcs.

# Results

Resulting optimal control law

$$u_{\text{opt}} = \begin{cases} 0 & \text{if } v + c_u \cdot \lambda_2 > 0 \\ u_{\text{sing}} & \text{if } v + c_u \cdot \lambda_2 \equiv 0 \\ u_{\max} & \text{if } v + c_u \cdot \lambda_2 < 0 \end{cases}$$

Singular arc solution yields constant speed $v_0$ with

$$u_{\text{sing}} = \frac{1}{c_u} \cdot \left( c_0 + c_2 \cdot v_0^2 \right)$$

# Main Points

Use two mechanical power sources:

- One to sustain the desired speed and optimized for efficiency
- One to provide the desired drivability (acceleration), optimized for high torque
- Of course during accelerations both motors are used in parallel mode
- When coasting both motors are shut down; complete separation from wheels (minimize friction)

# Optimal Energy Management in HEP

- When full information is available (perfect models, full driving profile data, …) the optimal energy control can be computed
- This solution is useful because it defines the benchmark that no other control algorithm can beat
- Some problems are close to this setup (hybrid buses, …), in other cases the chosen control algorithm can be compared to the benchmark

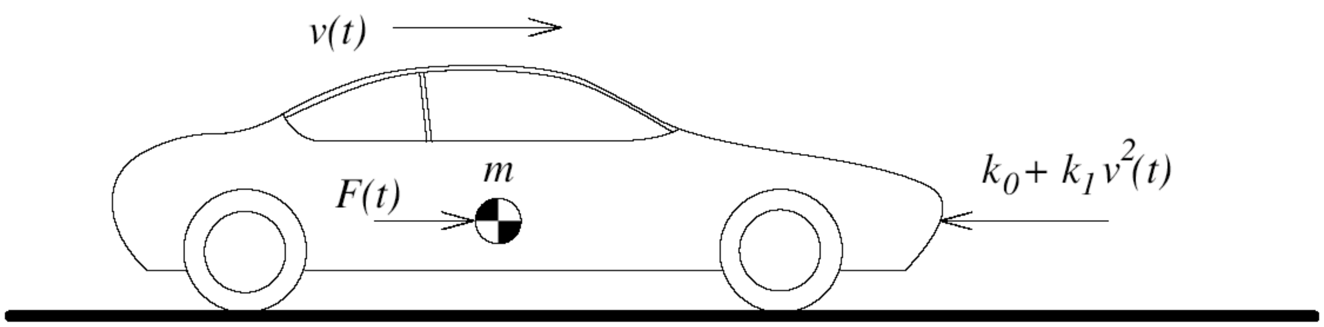
# Modeling Paradigms

"Forward," physics-based, causal, …

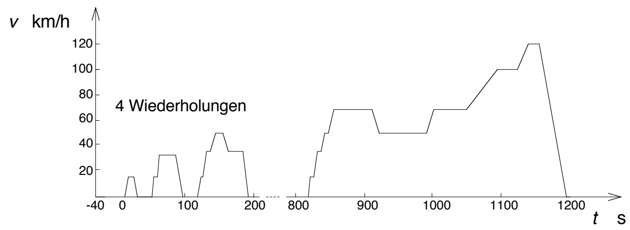
$$m\frac{d}{dt}v(t) = -\{k_0 + k_1 v(t)^2\} + F(t)$$

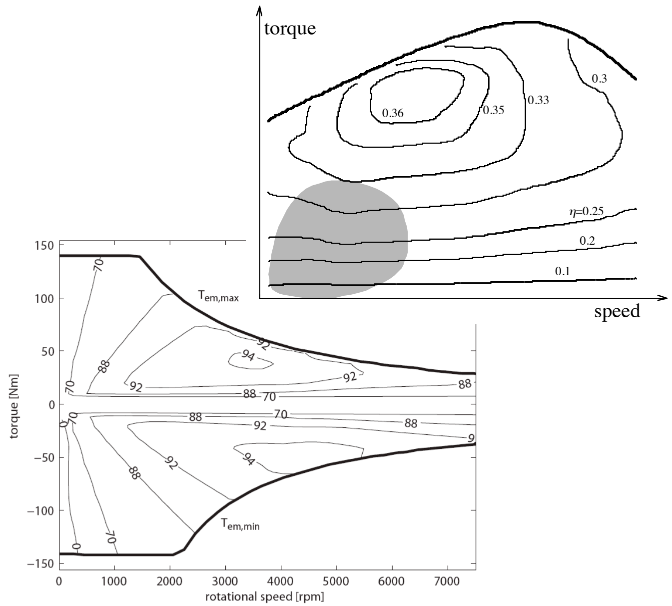
"Backward," inverted causality, …

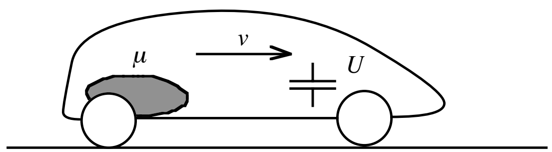
$$F(t_i) \approx m\frac{v(t_i) - v(t_{i-1})}{\delta} + k_0 + k_1\left(\frac{v(t_i) + v(t_{i-1})}{2}\right)^2$$



# Problem Setup




Driving cycle fixed a priori (speed and elevation as functions of either time or vehicle position)

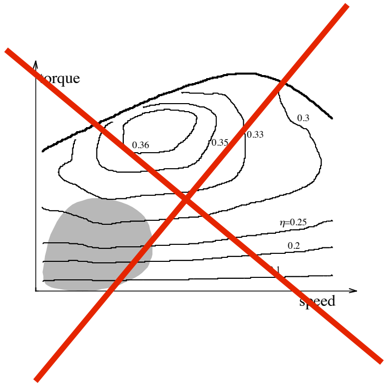
ICE & EM described by quasi-static models (“map”)

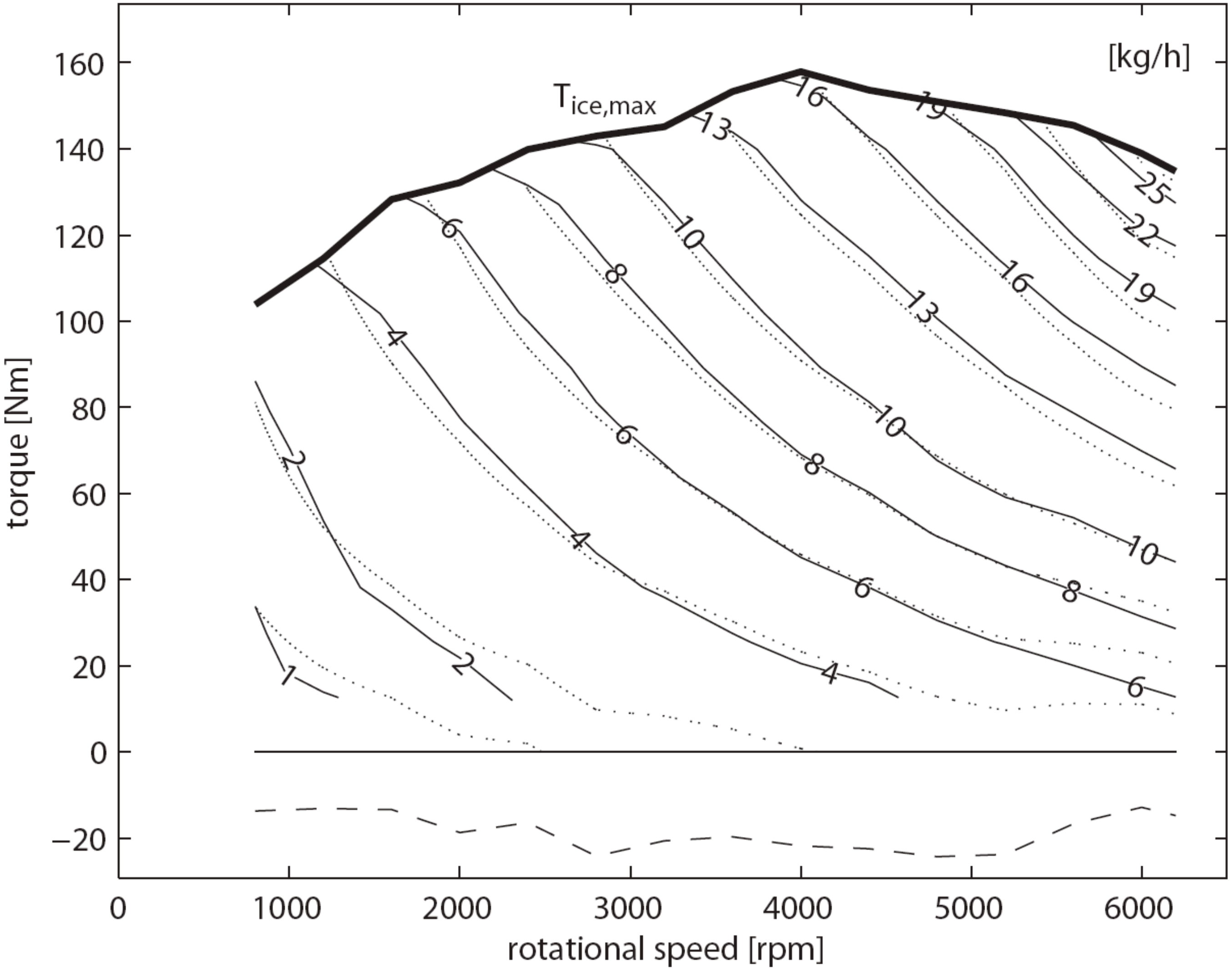
$$\eta(n, T) \qquad \text{(speed, torque)}$$

Battery as reversible energy reservoir (charge integrator)

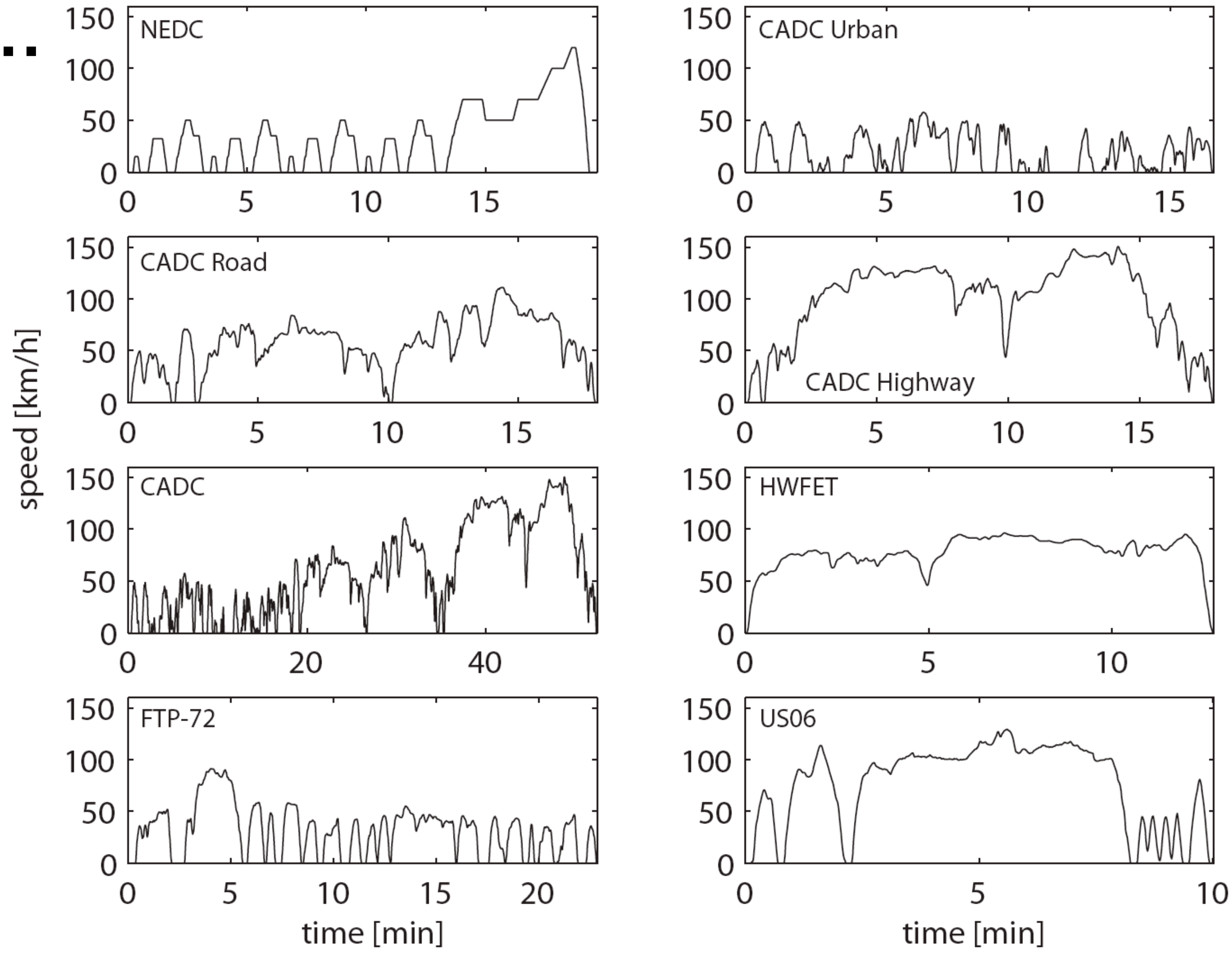
$$\eta_{BT}(U, I) \qquad \text{(voltage, current)}$$

# By the Way …


torque
0.36
0.35
0.33
0.3
η=0.25
0.2
0.1
speed


[kg/h]
T ice,max
160
140
120
100
80
60
40
20
0
−20
torque [Nm]
0
1000
2000
3000
4000
5000
6000
rotational speed [rpm]
1
2
4
6
8
10
13
16
19
22
25

# and …


NEDC
CADC Urban
CADC Road
CADC Highway
CADC
HWFET
FTP-72
US06
speed [km/h]
time [min]
time [min]
150
100
50
0
0
5
10
15
20
40

# Problem Formulation

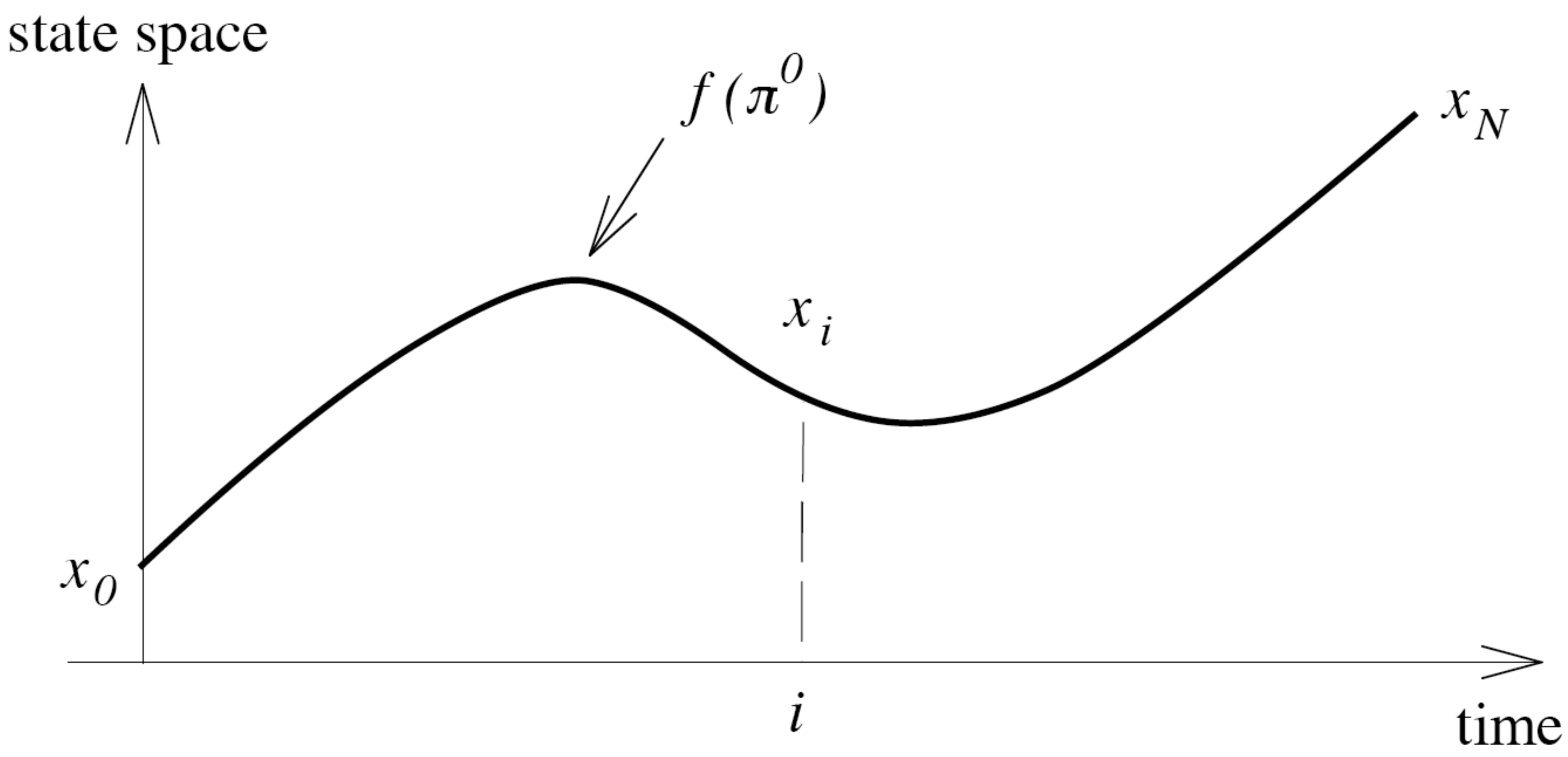
Find that control sequence ("policy") $u(k)$

that minimizes the total fuel consumption while satisfying all constraints imposed on the SoC and on the control signal

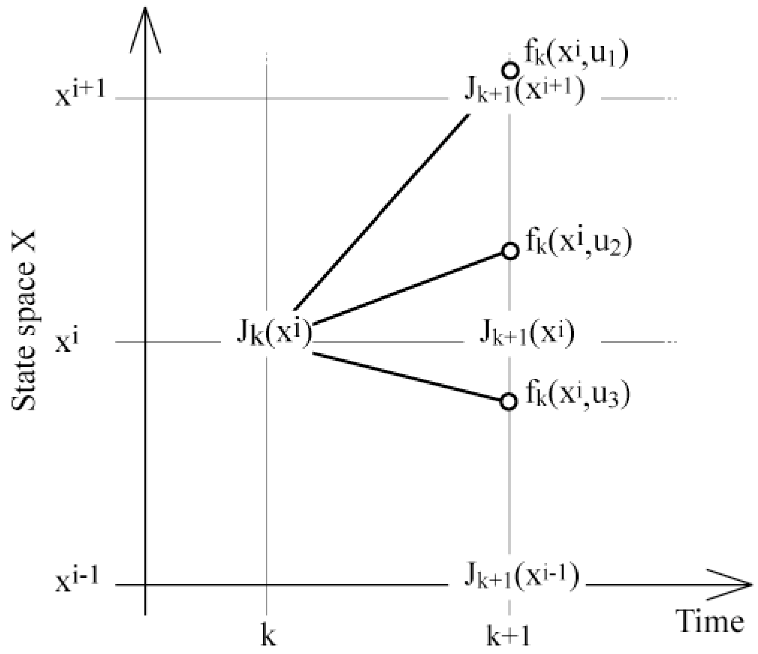
Equivalence factor $s(.)$ modified power split ratio

# Abstract Problem Formulation

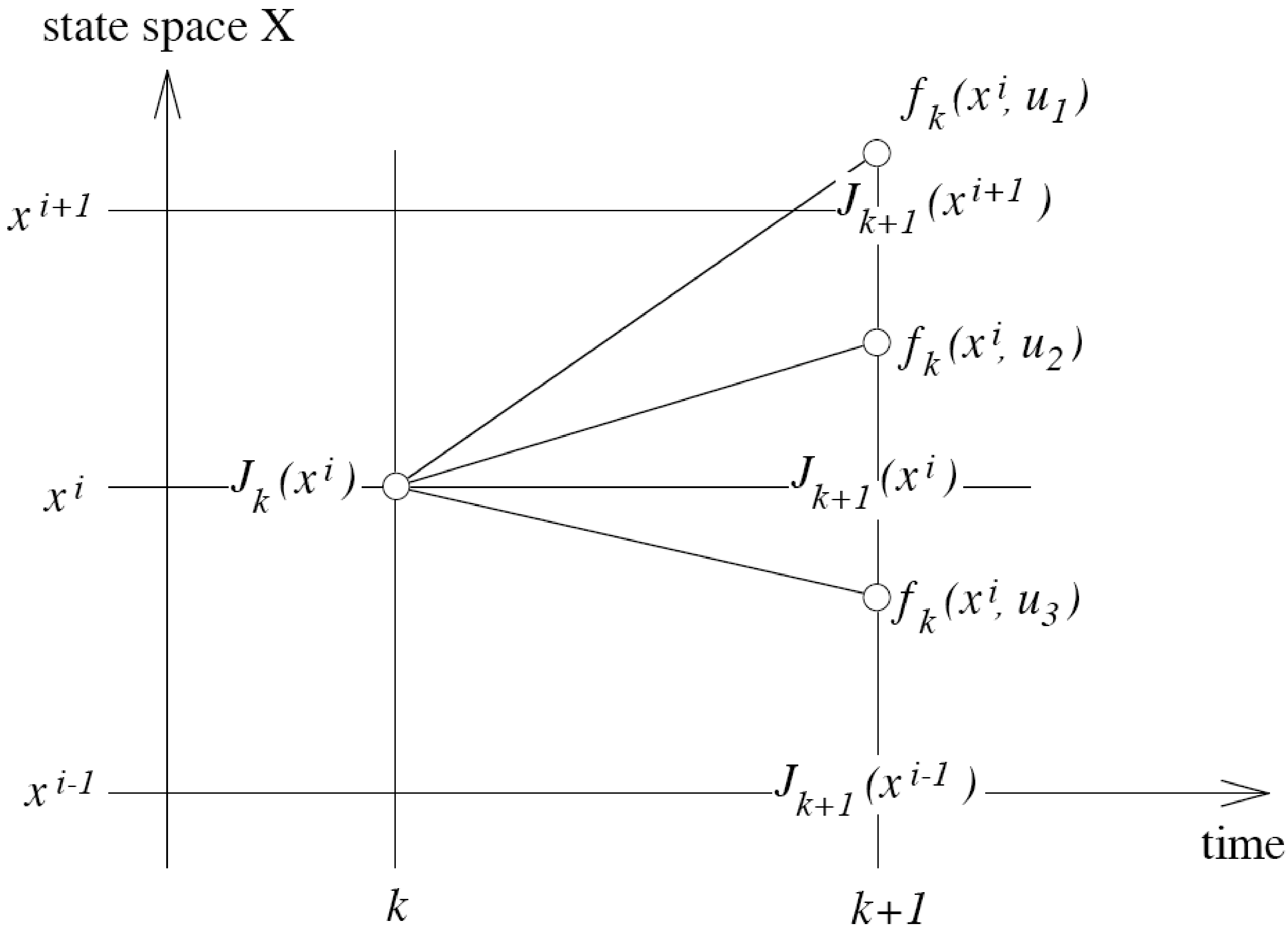
# Principle of Optimality



# Deterministic Dynamic Programming



# Discrete State Space Requires Interpolation



# Nearest Neighbor …

$$J_k(x_i) = \min \begin{bmatrix} g_k(x_i, u_1) + J_{k+1}(x^{i+1}) \\ g_k(x_i, u_2) + J_{k+1}(x^{i}) \\ g_k(x_i, u_3) + J_{k+1}(x^{i}) \end{bmatrix}$$

# … or Linear Interpolation

$$J_k(x_i) = \min \begin{bmatrix} g_k(x_i, u_1) + (f_k(x_i, u_1) - x^{i+1}) \cdot \frac{J_{k+1}(x^{i+2}) - J_{k+1}(x^{i+1})}{x^{i+2} - x^{i+1}} \\ g_k(x_i, u_2) + (f_k(x_i, u_2) - x^{i}) \cdot \frac{J_{k+1}(x^{i+1}) - J_{k+1}(x^{i})}{x^{i+1} - x^{i}} \\ g_k(x_i, u_3) + (f_k(x_i, u_3) - x^{i-1}) \cdot \frac{J_{k+1}(x^{i}) - J_{k+1}(x^{i-1})}{x^{i} - x^{i-1}} \end{bmatrix}$$

# Computational Burden

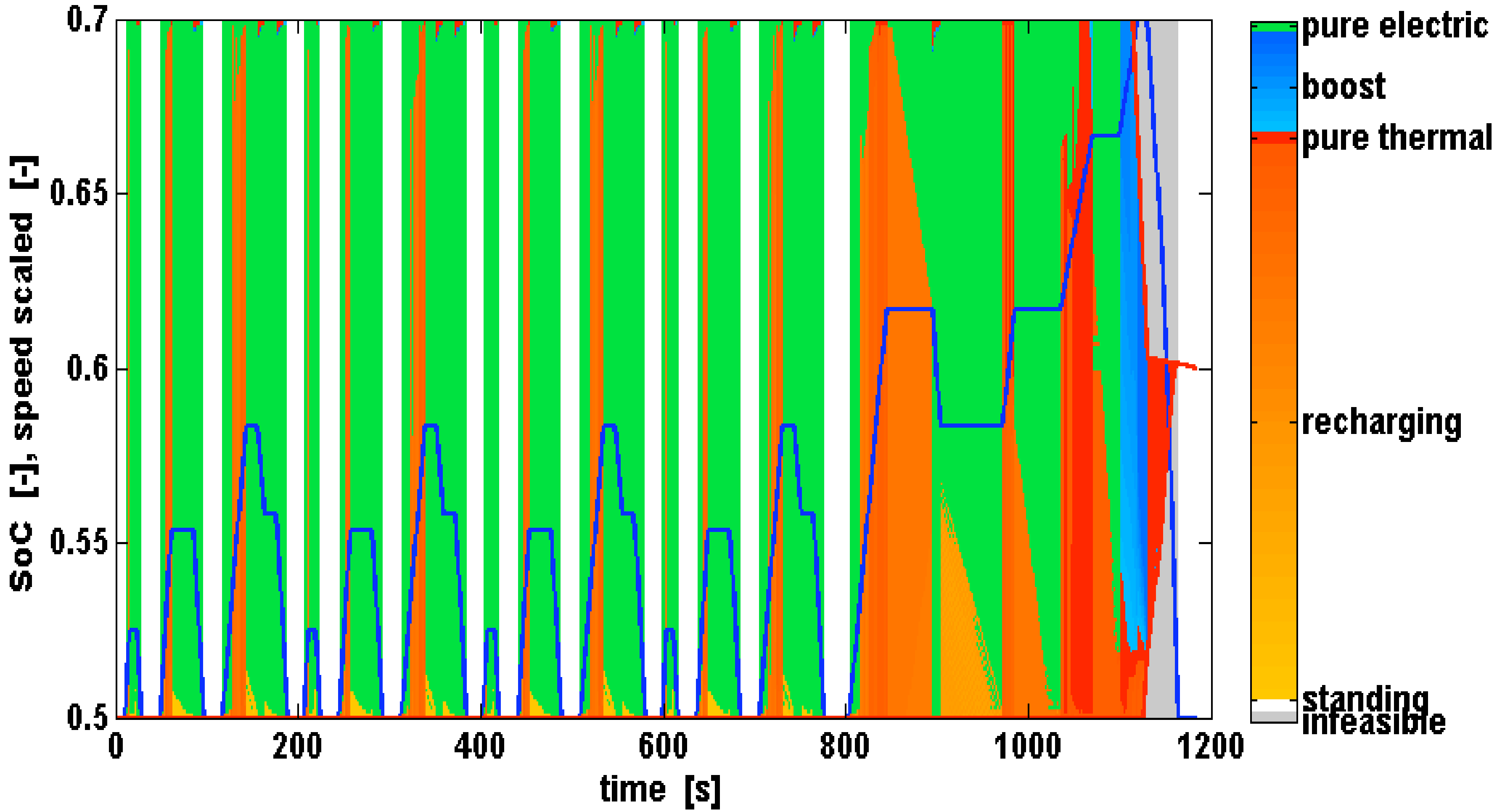
## Example

1. Problem of length 200s with a discretization of $\Delta t = 1$s
2. Control signal discretized with 10 points $(n_u = 10)$
3. State space discretized with 1000 points $(n_x = 1000)$
4. One model evaluation takes $t_{model} = 1\mu s$

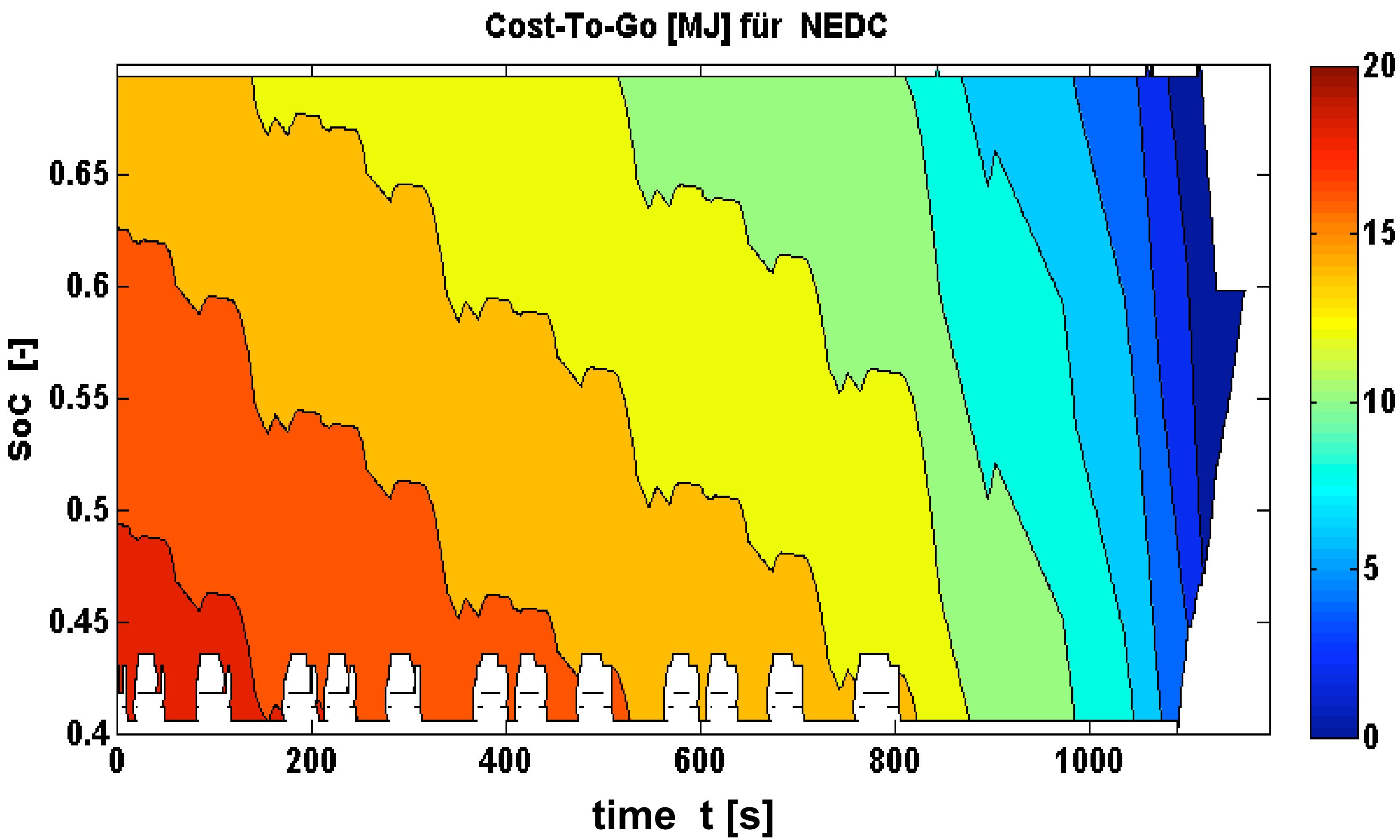
## Comparison "Brute Force" vs. "Dynamic Programming"

| | "Brute Force" | "Dynamic Programming |
|---|---|---|
| Idea: | evaluation of all possible control sequences $\Pi(x_0)$ | "intelligent" evaluation as shown before |
| Number of evaluations: | $n_{evalBF} = n_u^N = 10^{200}$ | $n_{evalBF} = N \cdot n_u \cdot n_x = 2 \cdot 10^6$ |
| Calculation time: | $t_{calc} = n_{evalBF} \cdot t_{model} \cong 3 \cdot 10^{186}$ years | $t_{calc} = n_{evalDP} \cdot t_{model} = 2\ s$ |

# NEDC Results I – Trajectories

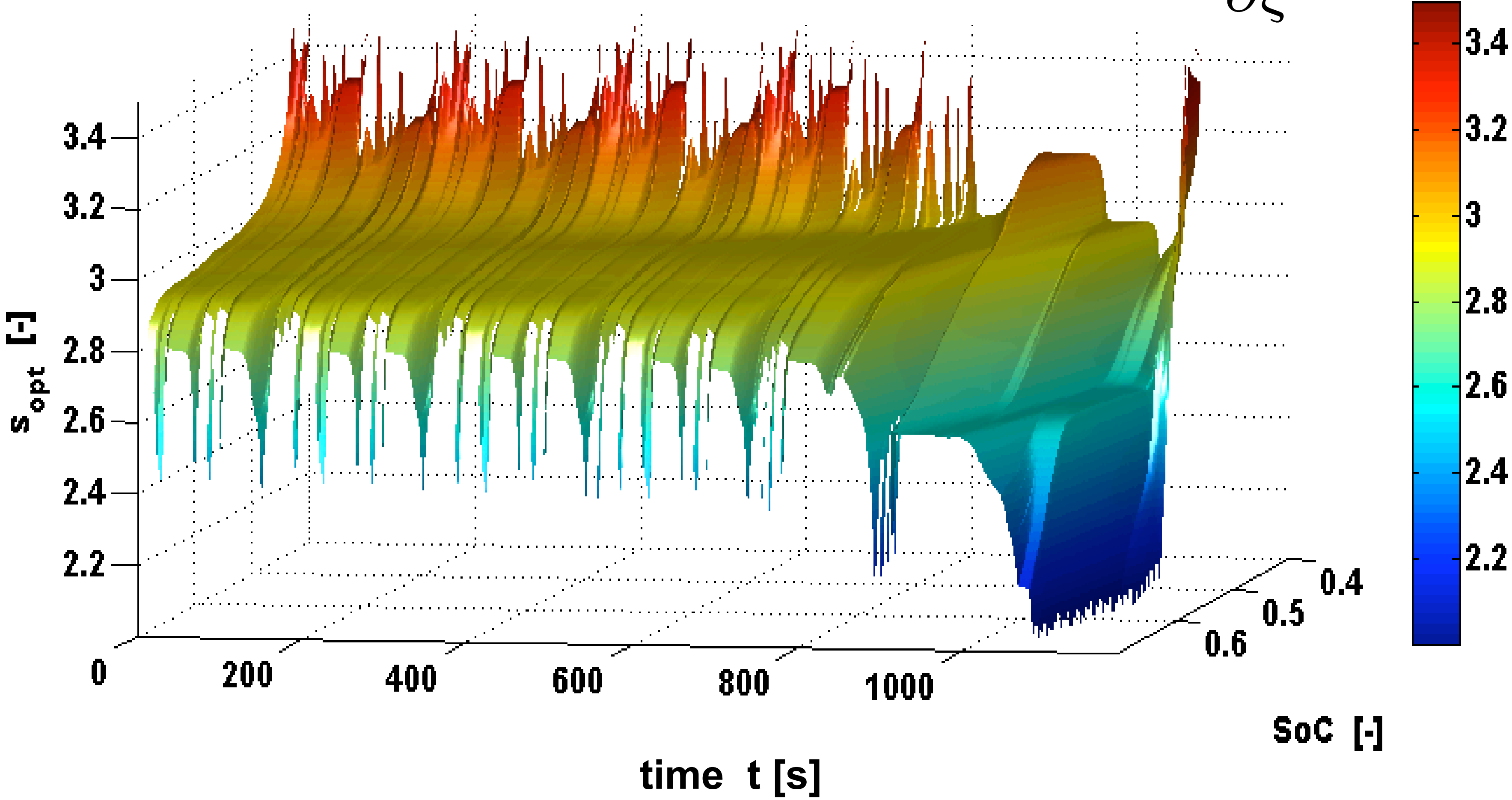


# NEDC Results II – “Cost to Go” $\mathcal{J}^o(\xi, t)$

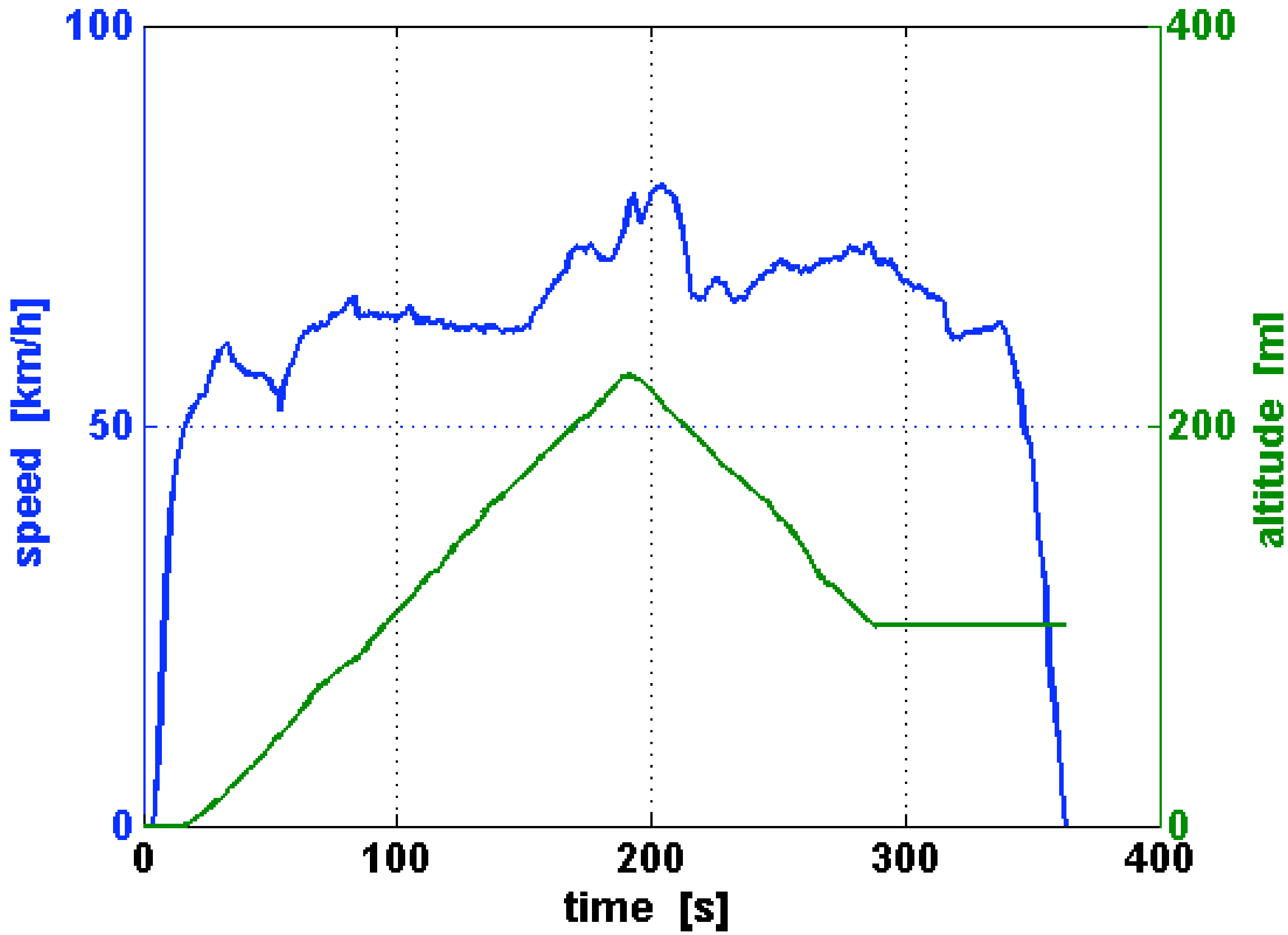


# NEDC Results III – Equivalence Factor

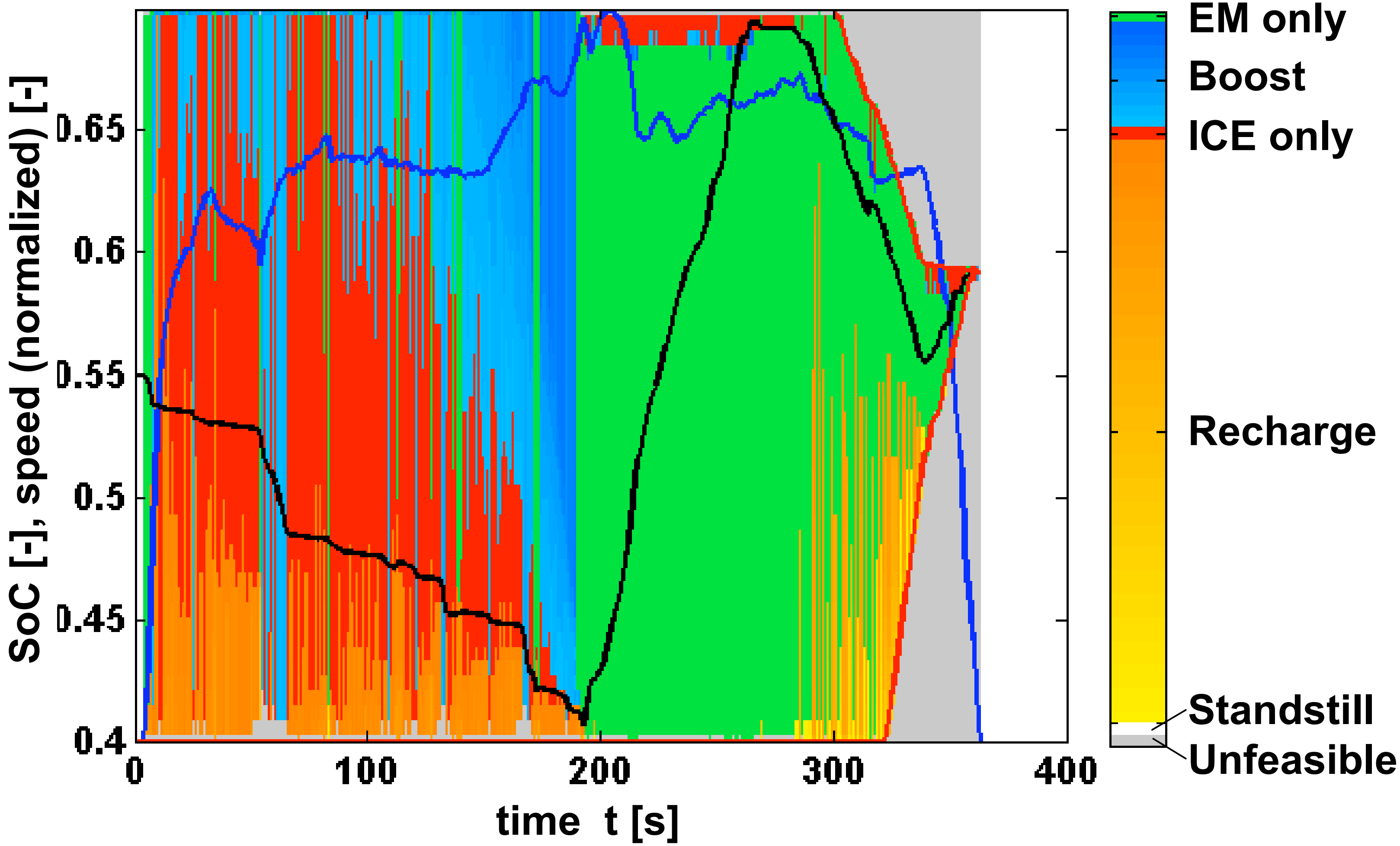
NEDC $s^o(\xi,t) = \frac{\partial \mathcal{J}^o(\xi,t)}{\partial \xi}$



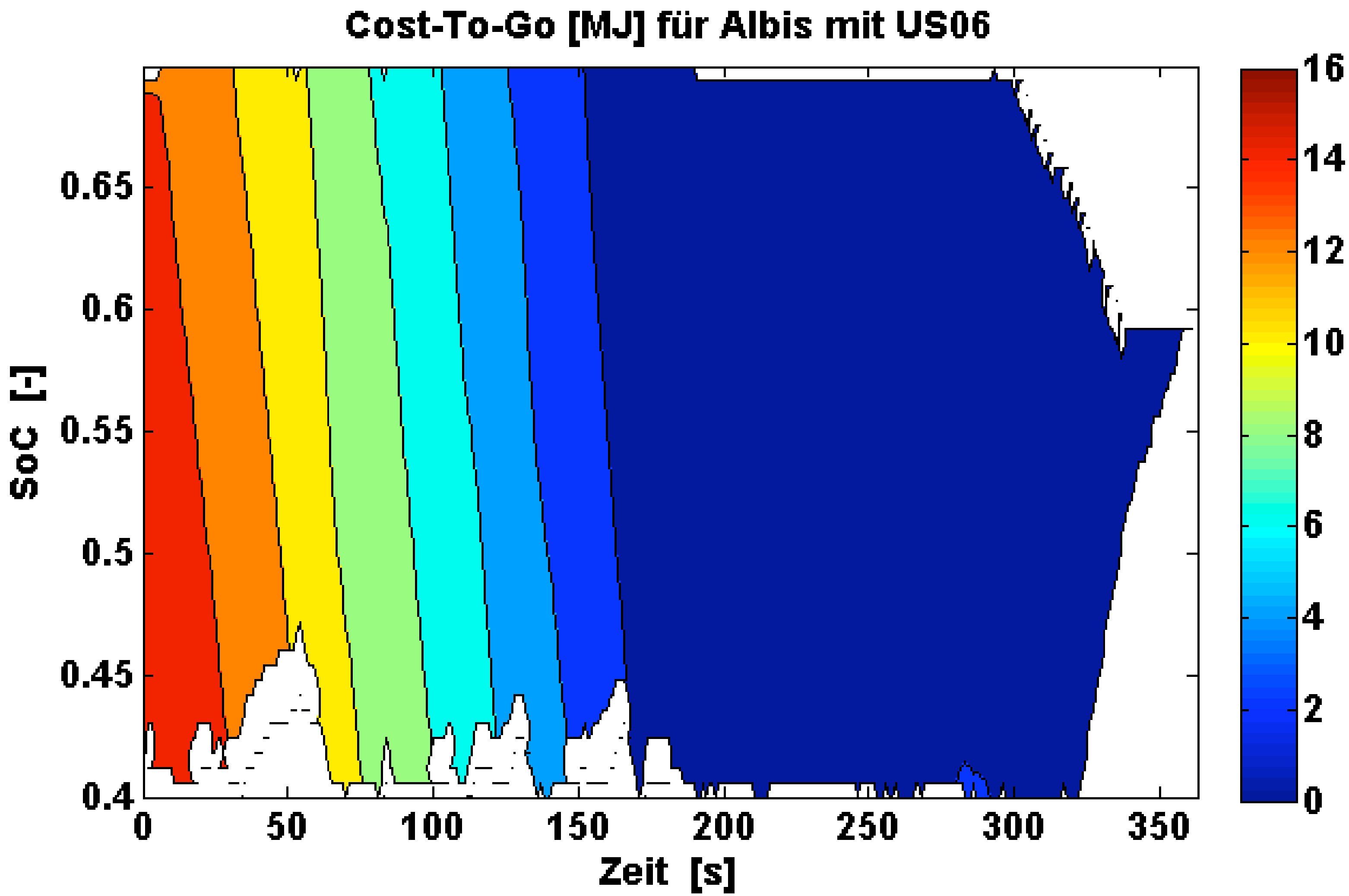
# Varying Altitude Profiles – I



# Varying Altitude Profiles – Control Signal

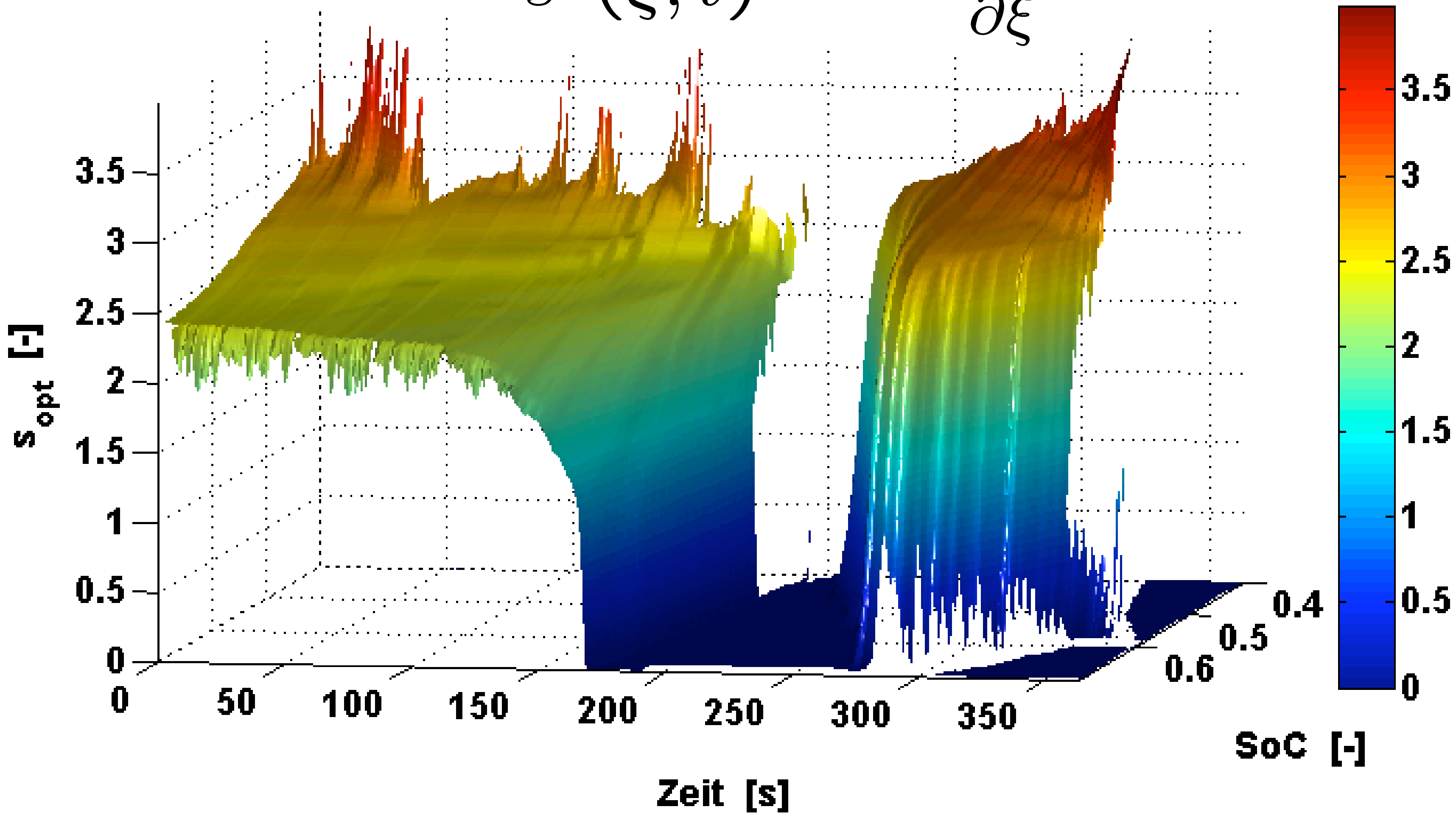


# Varying Altitude Profiles – Cost to Go $\mathcal{J}^o(\xi, t)$



# Varying Altitude Profiles – Equivalence Factor

$$s^o(\xi, t) = \frac{\partial \mathcal{J}^o(\xi,t)}{\partial \xi}$$



# By the Way …

```
for k = N-1 to 1
    [x_k+1 G] = F(X_grid,U_grid,k,...)
    Jk = G + V(x_k+1,k+1)
    J(i_x,k) = min Jk
    U(i_x,k) = argmin Jk
end
```

# Interesting Problems

Reduce numerical efforts (restricted search space, adaptive grid densities, combined “forward” – e.g. MPC – and backward approaches, etc.)

Computation “hardware,” on-board computations, …

Realization: integration of varying levels of information about the future driving profile (route, speed, altitude short and long prediction horizons, …)

Optimal system design (structures, parameters, and control algorithms combined)

Stochastic and mixed formulations