Vehicle Propulsion Systems
Lecture 5
Deterministic Dynamic Programming and Some Examples

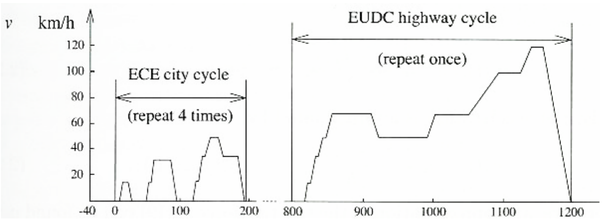
Lars Eriksson
Professor

Vehicular Systems
Linköping University

April 6, 2020

Outline

Energy consumption for cycles



Numerical values for MVEG-95, ECE, EUDC

$$\text{air drag} = \frac{1}{x_{tot}} \sum_{i \in trac} \bar{v}_i^3\, h = \quad \{319, 82.9, 455\}$$

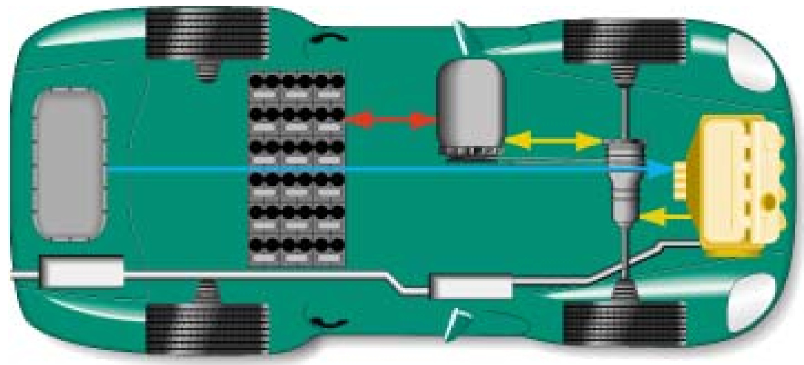
$$\text{rolling resistance} = \frac{1}{x_{tot}} \sum_{i \in trac} \bar{v}_i\, h = \quad \{.856, 0.81, 0.88\}$$

$$\text{kinetic energy} = \frac{1}{x_{tot}} \sum_{i \in trac} \bar{a}_i\, \bar{v}_i\, h = \quad \{0.101, 0.126, 0.086\}$$

$$\bar{E}_{\text{MVEG-95}} \approx A_f\, c_d\, 1.9 \cdot 10^4 + m_v\, c_r\, 8.4 \cdot 10^2 + m_v\, 10 \quad kJ/100km$$

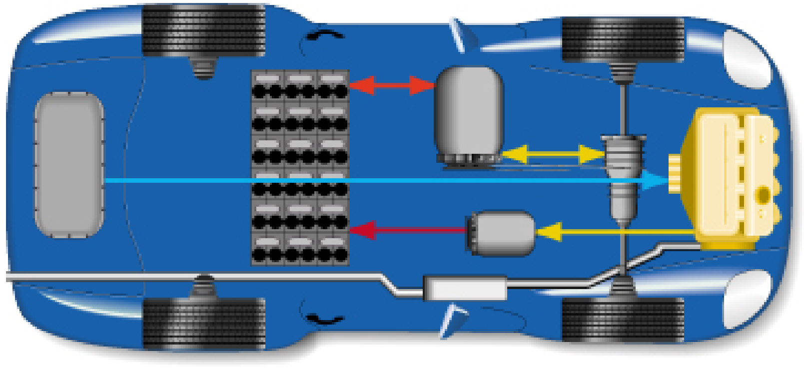
Hybrid Electrical Vehicles – Parallel

- Two parallel energy paths



Hybrid Electrical Vehicles – Serial

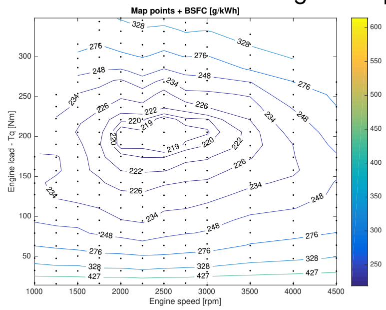
- Two paths working in series
- Decoupled through the battery



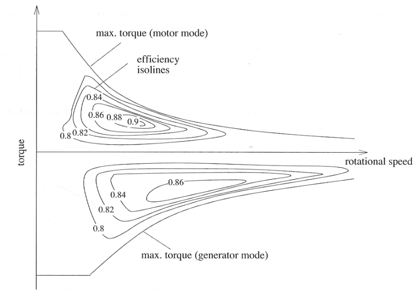
Component modeling

- Model energy (power) transfer and losses
- Using maps $\eta = f(T, \omega)$

Combustion engine map



Electric motor map



- Using parameterized (scalable) models
  –Willans approach

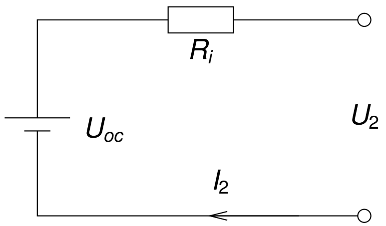
Battery – Standard model in this course

Simple model for the battery

–Open circuit voltage $U_{oc}(SOC)$
–State of charge SOC, (Q/Q_{max})

Output voltage

$$U_2 = U_{oc}(SOC) - R_i I_2 \qquad \frac{dQ}{dt} = -I_2$$



C-rate

How fast is the battery (pack) charged.

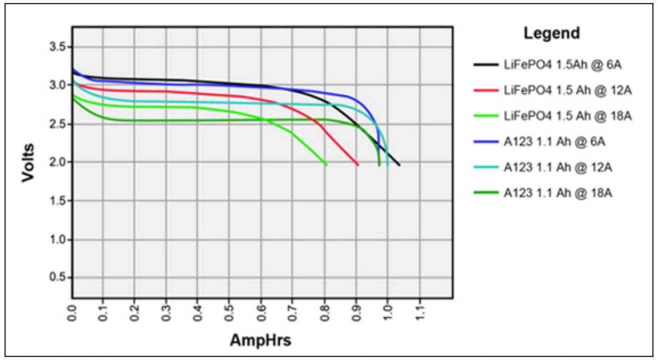
- C=1, full capacity in 1 hour.

To protect the battery we need to:

- impose limits on the current.
- avoid emptying the battery completely
- avoid over filling the battery

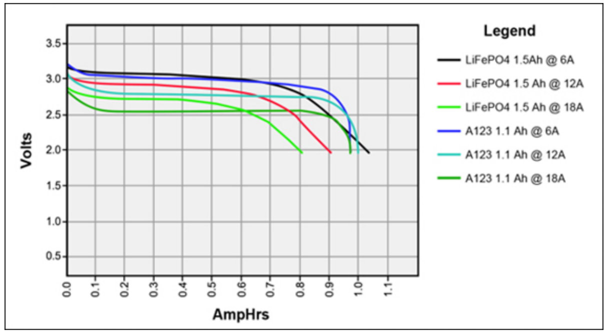
New lecture on batteries planned after easter – Stay tuned

Voltage and SOC



Typical characteristics. Can extract inner resistance, and capacity.
(Image source: batteryuniversity.com)

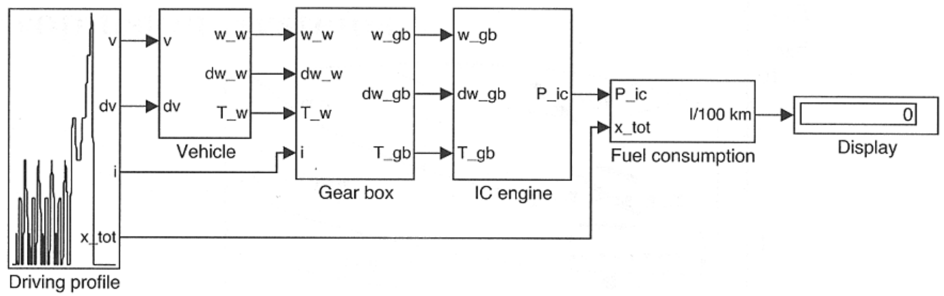
Two important battery estimation problems



- SOC – State of Charge. Current and voltage sensing.
- SOH – State of Health. Cycle monitoring, current and voltage sensing.
- Prolonging life: Temperature monitoring and current limits important.

Model implemented in QSS

Conventional powertrain



Efficient computations are important
–For example if we want to do optimization and sensitivity studies.

Outline

Optimization – Linear Programming

- Linear problem

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & A x = b \\ & x \geq 0 \end{aligned}$$

- Convex problem
- Much analyzed: existence, uniqueness, sensitivity
- Many algorithms: Simplex the most famous

- About the word *Programming*
  –The solution to a problem was called a program

Optimization – Non-Linear Programming

- Non-linear problem

$$\begin{array}{rlcl} \min_x & f(x) & & \\ \text{s.t.} & g(x) & = & 0 \\ & x & \geq & 0 \end{array}$$

- For convex problems
  –Much analyzed: existence, uniqueness, sensitivity.
  –Many (fast) algorithms.
- For non-convex problems
  –Some special problems have solutions
  –Local optimum is not necessarily a global optimum
- As engineers you need a methodology to ensure that you get a good solution.

Industry is not always interested in **The Optimal** solution
–more often a **Good Solution** is enough.

Mixed Integer and Combinatorial Optimziation

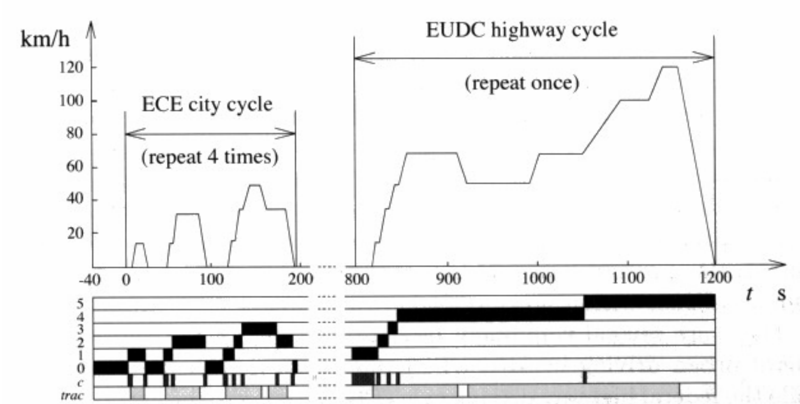
- Problem

$$\begin{array}{rlcl} \min_x & f(x,y) & & \\ \text{s.t.} & g(x,y) & = & 0 \\ & x & \geq & 0 \\ & y & \in & Z^+ \end{array}$$

- Inherently non-convex y
  Generally hard problems to solve.
- Much analyzed
  –Existence, uniqueness, sensitivity
  –Many types of problems
  –Many algorithms are available

An Example Problem – With Interesting Properties

What gear ratios give the lowest fuel consumption for a given drivingcycle?
–Problem presented in appendix 8.1



Problem characteristics

- Countable number of free variables, $i_{g,j}$, $j \in [1, 5]$
- A "computable" cost, $m_f(\cdots)$
- A "computable" set of constraints, model and cycle
- The formulated problem

Some comments on practical optimiztion

General process

- Find the "right" problem formulation
  - Model of the system
  - Important properties, and your goal
  - Constraints: What do you want to aviod
- Find and use the right solver for the problem
- Analyze the solution and (perhaps) reconsider the problem and iterate

Fundamental Issues that you Should be Aware Of

- All optimal solutions are **extreme points**
- The optimizer (solver) will **shamelessly exploit** all weaknesses of your model and problem formulation
- That's why you often need to reconsider the problem formulation

Outline

Optimal Control – Problem Motivation

Car with gas pedal $u(t)$ as control input:
How to drive from A to B on a given time with minimum fuel consumption?

- Infinite dimensional decision variable $u(t)$.
- Cost function $\int_0^{t_f} \dot{m}_f(t)dt$
- Constraints:
  - Model of the car (the vehicle motion equation)

$$\begin{aligned} m_v \tfrac{d}{dt} v(t) &= F_t(v(t), u(t)) \quad -(F_a(v(t)) + F_r(v(t)) + F_g(x(t))) \\ \tfrac{d}{dt} x(t) &= v(t) \\ \dot{m}_f &= f(v(t), u(t)) \end{aligned}$$

  - Starting point $x(0) = A$
  - End point $x(t_f) = B$
  - Speed limits $v(t) \leq g(x(t))$
  - Limited control action $0 \leq u(t) \leq 1$
- Difficult problem to solve analytically, only some special cases are solvable.

General problem formulation

- Performance index

$$J(u) = \phi(x(t_b), t_b) + \int_{t_a}^{t_b} L(x(t), u(t), t)dt$$

- System model (constraints)

$$\frac{d}{dt}x = f(x(t), u(t), t), \qquad x(t_a) = x_a$$

- State and control constraints

$$u(t) \in U(t)$$

$$x(t) \in X(t)$$

Optimal Control – Historical Perspective

- Old subject
- Rich theory
  - Old theory from calculus of variations
  - Much theory and many methods were developed during 50's-70's
  - Theory and methods are still being actively developed
- Dynamic programming, Richard Bellman, 50's.
- A modern success story:
  –Model predictive control (MPC)
- Now a new interest for collocation methods:
  –A few during 1990's
  –Much interest 2000–

Separate Course ⇒ TSRT08 Optimal Control

Outline

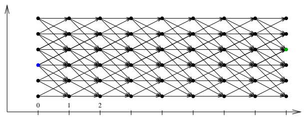
Dynamic programming – Problem Formulation

- Optimal control problem

$$\begin{aligned} \min J(u) &= \phi(x(t_b), t_b) + \int_{t_a}^{t_b} L(x(t), u(t), t)dt \\ s.t. \frac{d}{dt}x &= f(x(t), u(t), t) \\ x(t_a) &= x_a \\ u(t) &\in U(t) \\ x(t) &\in X(t) \end{aligned}$$

- $x(t)$, $u(t)$ functions on $t \in [t_a, t_b]$
- Search an approximation to the solution by discretizing
  - the state space $x(t)$
  - and maybe the control signal $u(t)$

  in both amplitude and time.
- The result is a combinatorial (network) problem



Dynamic Programming (DP) – Problem Formulation

- Find the optimal control sequence $\pi^0(x_0) = \{u_0, u_1, \ldots, u_{N-1}\}$ minimizing:

$$J(x_0) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)$$

- subject to:

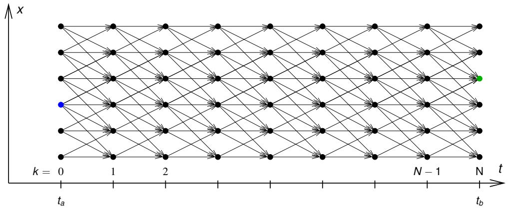
$$\begin{aligned} x_{k+1} &= f_k(x_k, u_k, w_k) \\ x_0 &= x(t = 0) \\ x_k &\in X_k \\ u_k &\in U_k \end{aligned}$$

- Disturbance w_k
- Stochastic vs Deterministic DP

DDP – Basic Algorithm

$$\begin{aligned} J(x_0) &= g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \\ x_{k+1} &= f_k(x_k, u_k) \end{aligned}$$

Bellman's Theory and Algorithm:
–Start at the end and proceed backward in time
–Determine the optimal cost-to-go
–Store the corresponding control signal



DDP – Basic algorithm

$$J(x_0) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$
$$x_{k+1} = f_k(x_k, u_k)$$

Algorithm:

1. Set $k = N$, and assign final cost $J_N(x_N) = g_N(x_N)$
2. Set $k = k - 1$
3. For all points in the state-space grid, find the optimal cost to go

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} g_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k))$$

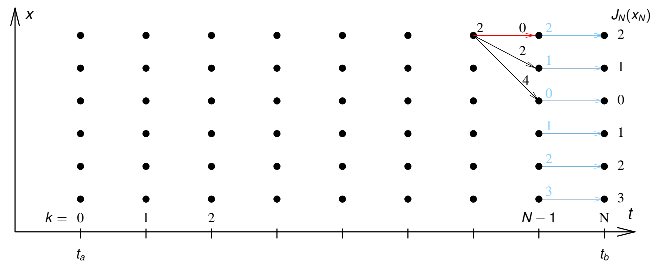
4. If $k = 0$ then return solution
5. Go to step 2

Deterministic Dynamic Programming – Basic Algorithm

Fundamental idea

Construct the Cost-to-go by solving small subproblems.

Graphical illustration of the solution procedure



Arc Cost Calculations

For an arc

- You know where you are
- also know all places you can go to

There are two ways for calculating the arc costs

- Calculate the exact control signal and cost for each arc
  –Quasi-static approach
- Make a grid over the control signal and interpolate the cost for each arc
  –Forward calculation approach

Matlab implementation – it is important to utilize matrix calculations

- Calculate the whole bundle of arcs in one step
- Add boundary and constraint checks

Pros and Cons with Dynamic Programming

Pros

- Globally optimal, for all initial conditions
- Can handle nonlinearities and constraints
- Time complexity grows linearly with horizon
- Use output and solution as reference for comparison

Cons

- Non causal
- Time complexity grows “exponentially” with number of states, curse of dimensionality
- 2-3 states are often at the limit

Calculation Example

- Problem 200s with discretization $\Delta t = 1$s.
- Control signal discretized with 10 points.
- Statespace discretized with 1000 points.
- One evaluation of the model takes 1μs
- Solution time:
  - Brute force:
    Evaluate all possible combinations of control sequences.
    Number of evaluations, 10^{200} gives $\approx 3 \cdot 10^{186}$ years. (Universe is $\approx 13.8 \cdot 10^{9}$ years.)
  - Dynamic programming:
    Number of evaluations: $200 \cdot 10 \cdot 1000$ gives 2 s.

(Example contributed by ETH)

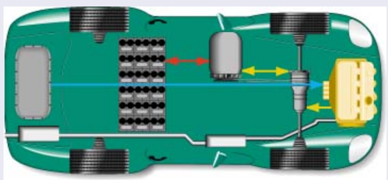
Outline

1. Repetition
2. “Traditional” Optimization
   - Different Classes of Problems
   - An Example Problem
3. Optimal Control
   - Problem Motivation
4. Deterministic Dynamic Programming
   - Problem setup and basic solution idea
   - Cost Calculation – Two Implementation Alternatives
5. Hand-In Task 2
   - The Provided Tools
   - Case Studies

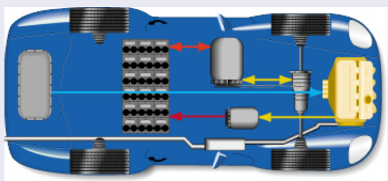
Hand-In Task 2 – Energy Management of Two Hybrids

Optimize the fuel consumption of 2 hybrids over driving cycles, using DDP

Parallel Hybrid



Series Hybrid



One degree of freedom

– SOC, main control variable
– Engine speed is **given** by the cycle

Two degrees of freedom

– SOC, main control variable
– Engine speed can be **freely selected**

The Provided Tools for Hand-in 2 and the Goals

Tasks and Tools

Investigate optimal control of one parallel and one series hybrid configuration in different driving profiles

- Some Matlab-functions provided
  - Skeleton file for defining the problems
  - 2 DDP solvers, 1-dim and 2-dim.
  - 2 skeleton files for calculating the arc costs for parallel and serial hybrids

Solve the problems, analyze the solutions, see if they are generalizable

Learning Goals

- Knowledge about operation modes of different hybrid topologies
- Experience in modeling of hybrid electric vehicles
- Experience from working and solving an optimal control problem
- See the benefits of different hybrid topologies

Tools

Problem setup – `testHybrids.m`

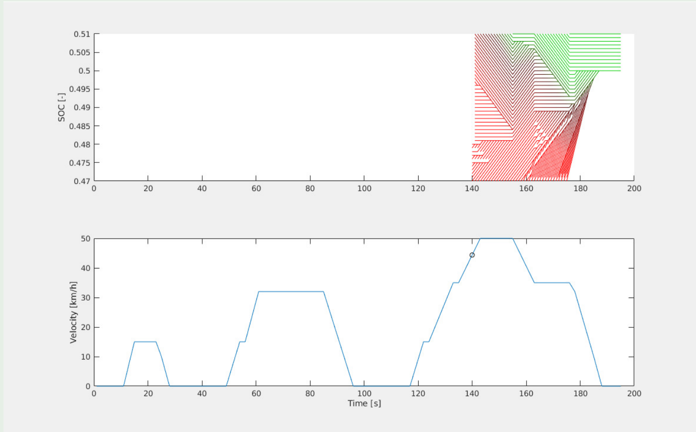
–Your Analysis Task

DDP Solver – `dynProg1D.m` –Given

Arc Calculator – `parallelHybrid.m` –Your main Implementation Task

Your Implementation Task 1 – The process of constructing a solution

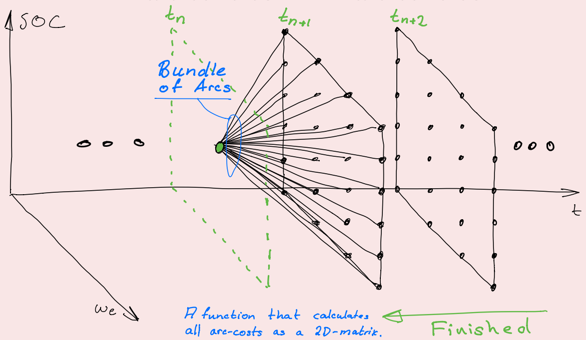
You will implement the arc cost calculations, for a bundle of arcs.



Your Implementation Task 1 – The process of constructing a solution

Upgrading to Series-Hybrid – 2 DoF

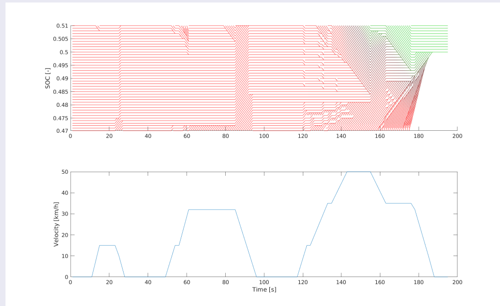
1D arc bundles → 2D arc bundles



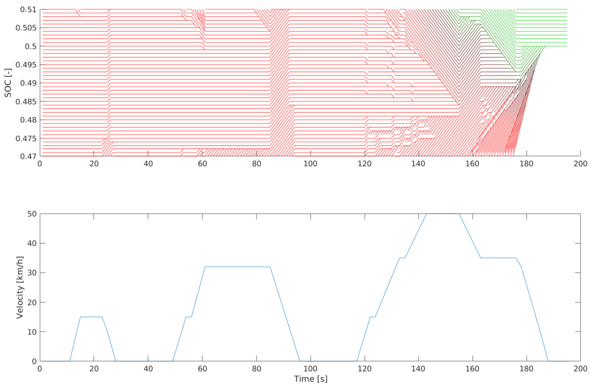
Your Implementation Task 2 – Unwiding the Solution

The functions dynProg1D and dynProg2D returns

- The cost to go function values and solution steps
- Solution: Information about the next step
- Unwind: Start from the initial value and follow the path to the end

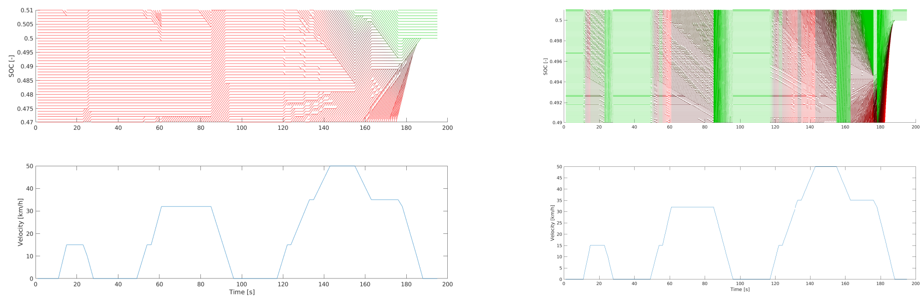


Unwiding the Solution - Video

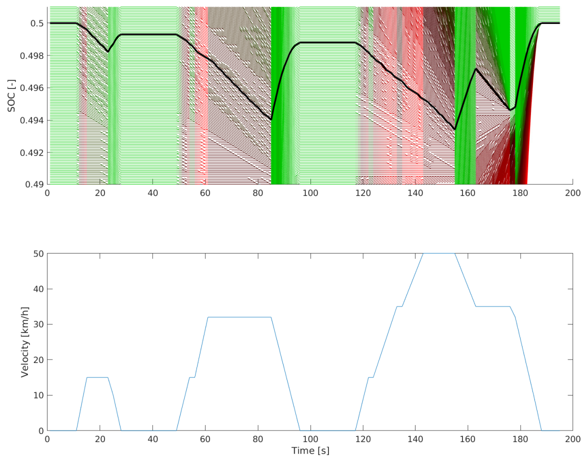


Numerical Accuracy

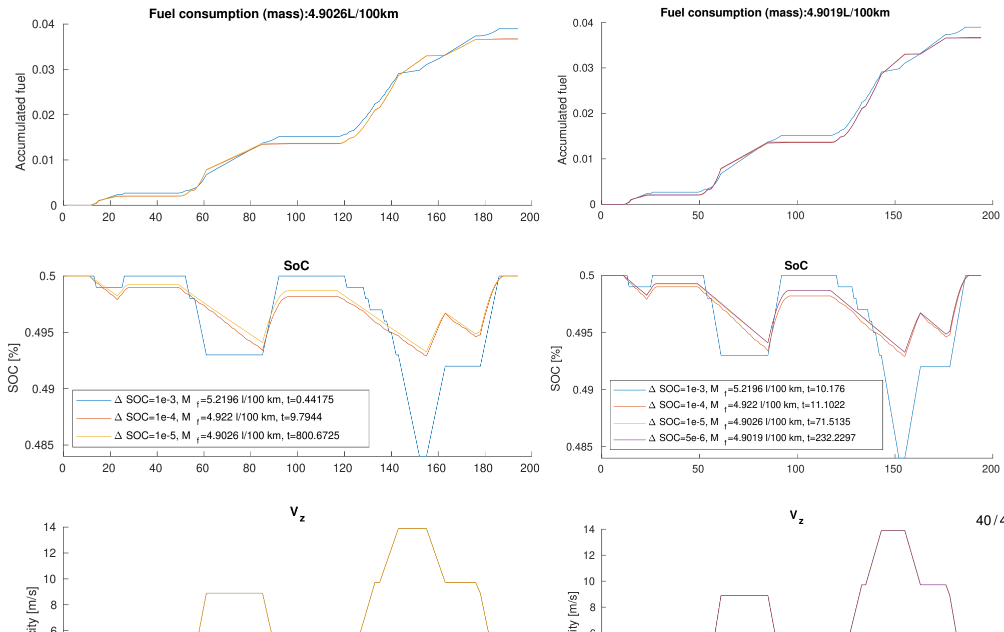
DDP guarantees a global solution – but only within the discretization
More accurate discretization might be needed to see the details in a solution



Unwided Solution – Higher Accuracy



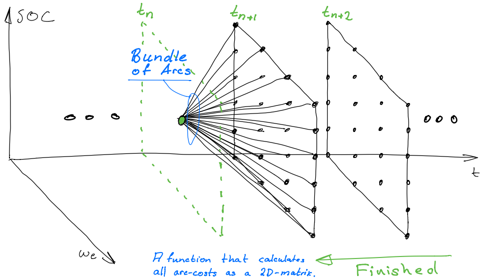
Numerical Accuracy – Solution time – Parallel Computing in Matlab



Your Implementation Task 1 – The process of constructing a solution

Analysis of complexity:
Consider a two dimensional problem that have N_x and N_y points in their grids and N_t time points.

- At each time step N_t we have to:
- evaluate all points $N_x\, N_y$ in the sheet and for each of them
- all their $N_x\, N_y$ following potential candidates



Resulting in a complexity of

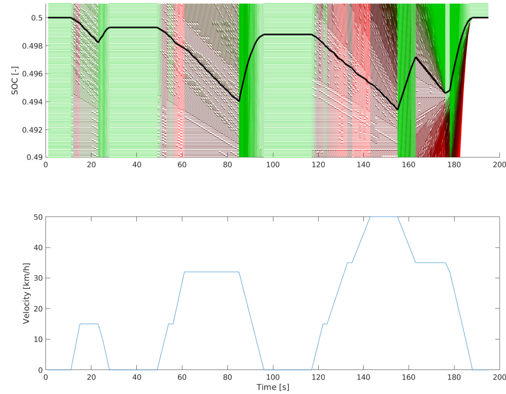
$$T = k\, N_t\, N_x^2 N_y^2$$

So it is quadratic in each dimension and linear in time

Exponential curse of dimensions (p-dim.)

$$T = k\, N^{2\,p}$$

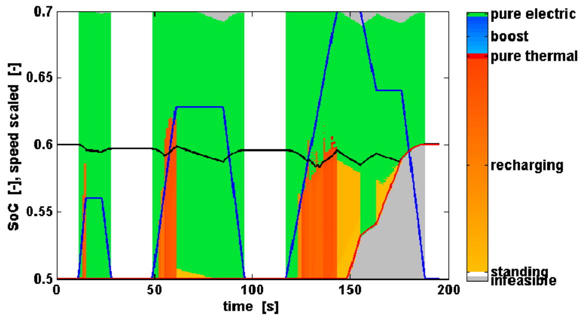
General Advice



- Work with arc costs and debug
- Use Matlab matrix math
- Start with a smaller problem to learn
- Start with a coarser grid and then refine
- When you are convinced that you have the solution ready then increase the problem size and level of detail
- Computation time for series hybrid $\sim$ 1 hour

Parallel Hybrid Example

- Fuel-optimal torque split factor $u(SOC, t) = \frac{T_{e-motor}}{T_{gearbox}}$
- ECE cycle
- Constraints $SOC(t = t_f) \geq 0.6$, $SOC \in [0.5, 0.7]$



Parallel Hybrid Example

- Fuel-optimal torque split factor $u(SOC, t) = \frac{T_{e-motor}}{T_{gearbox}}$
- NEDC cycle
- Constraints $SOC(t = t_f) = 0.6$, $SOC \in [0.5, 0.7]$