Introduction to Assignment 2 and Debugger

TSFS03 Lesson

Mahdi Morsali

Vehicular Systems
Department of Electrical Engineering
Linköping University

April 6, 2020

LINKÖPING UNIVERSITY

LINKÖPING
UNIVERSITY

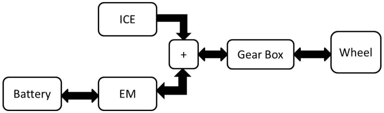
Hand-in 2 Introduction

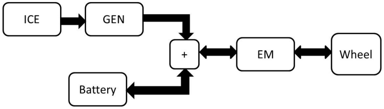
For a known drive cycle, how to optimally operate the electric motor/combustion engine?

Hand-in 2 Introduction

For a known drive cycle, how to optimally operate the electric motor/combustion engine?

- Two hybrid architectures: parallel and series.

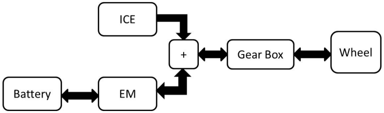


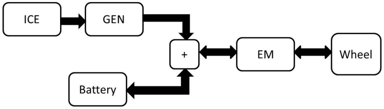


Hand-in 2 Introduction

For a known drive cycle, how to optimally operate the electric motor/combustion engine?

- Two hybrid architectures: parallel and series.

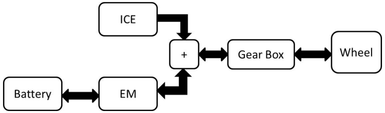


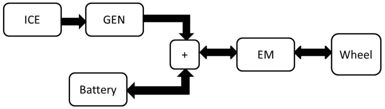


Hand-in 2 Introduction

For a known drive cycle, how to optimally operate the electric motor/combustion engine?

- Two hybrid architectures: parallel and series.



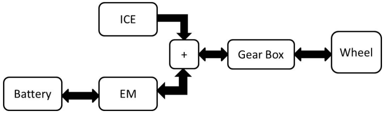


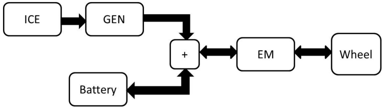
- Model the vehicles following the quasi-static approach.

Hand-in 2 Introduction

For a known drive cycle, how to optimally operate the electric motor/combustion engine?

- Two hybrid architectures: parallel and series.



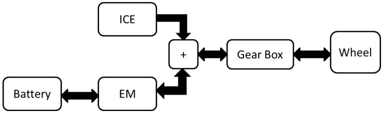


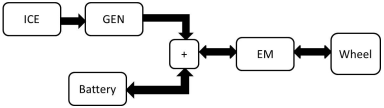
- Model the vehicles following the quasi-static approach.
- Solve the problem with Deterministic Dynamic Programming.

Hand-in 2 Introduction

For a known drive cycle, how to optimally operate the electric motor/combustion engine?

- Two hybrid architectures: parallel and series.





- Model the vehicles following the quasi-static approach.
- Solve the problem with Deterministic Dynamic Programming.

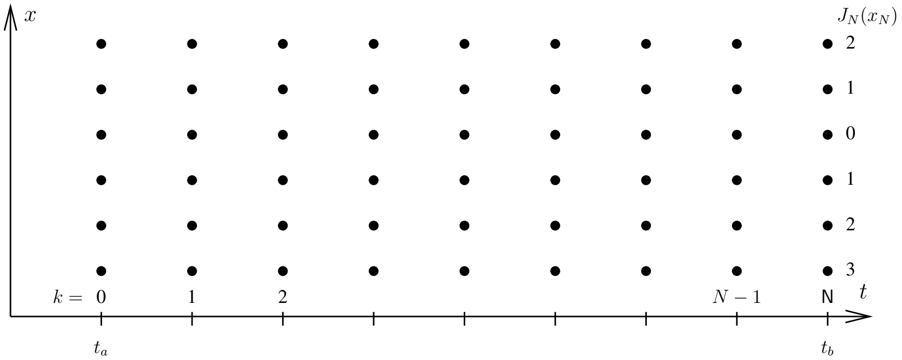
Hand-in Goals:

- Acquire knowledge and experience with DDP.
- Acquire knowledge about the properties and differences between parallel and series architectures.

LINKÖPING
UNIVERSITY

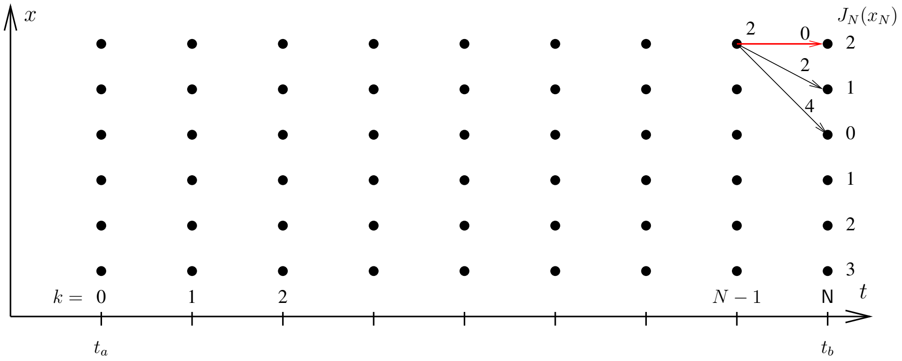
DDP Example

Graphical illustration of the solution procedure



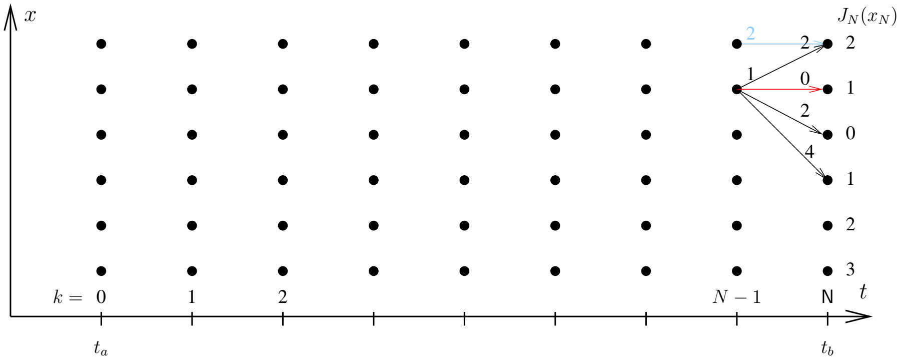
DDP Example

Graphical illustration of the solution procedure



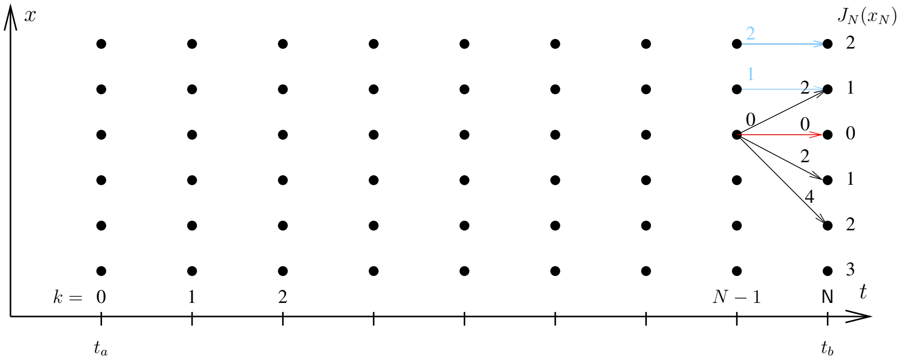
DDP Example

Graphical illustration of the solution procedure



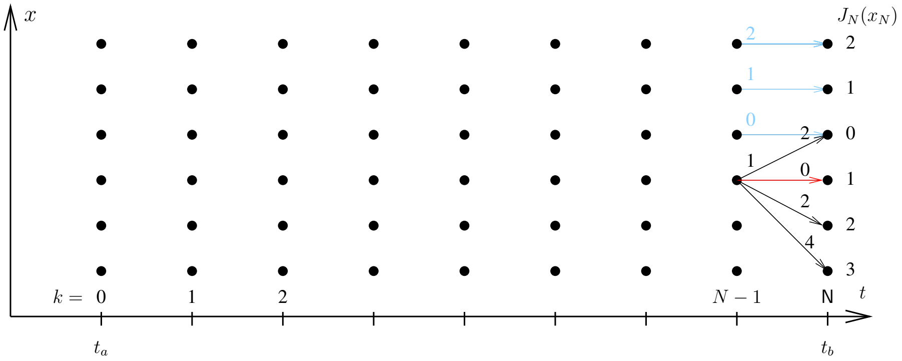
DDP Example

Graphical illustration of the solution procedure



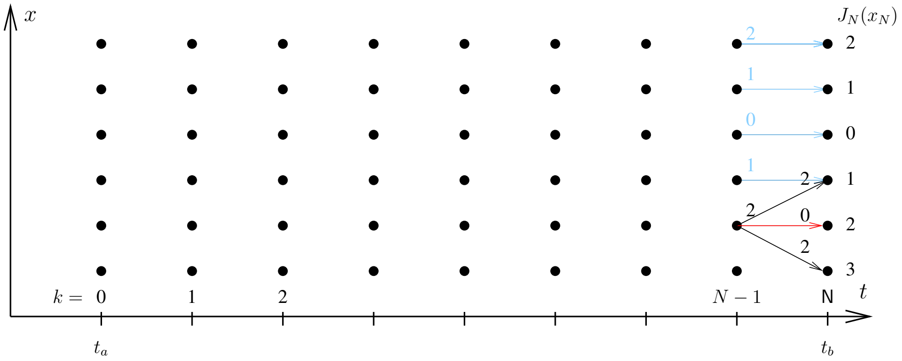
DDP Example

Graphical illustration of the solution procedure



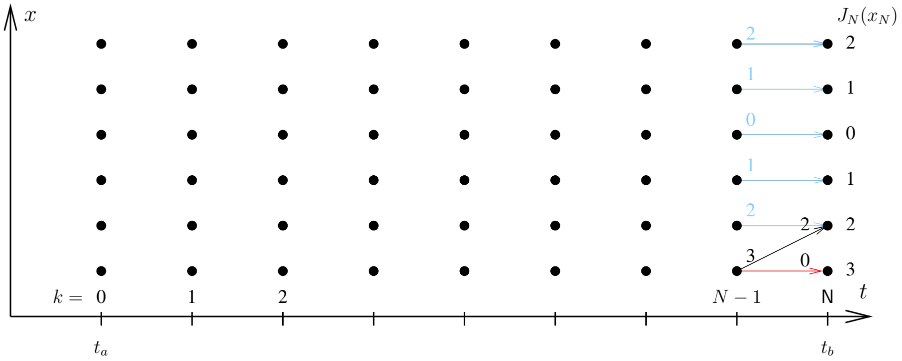
DDP Example

Graphical illustration of the solution procedure



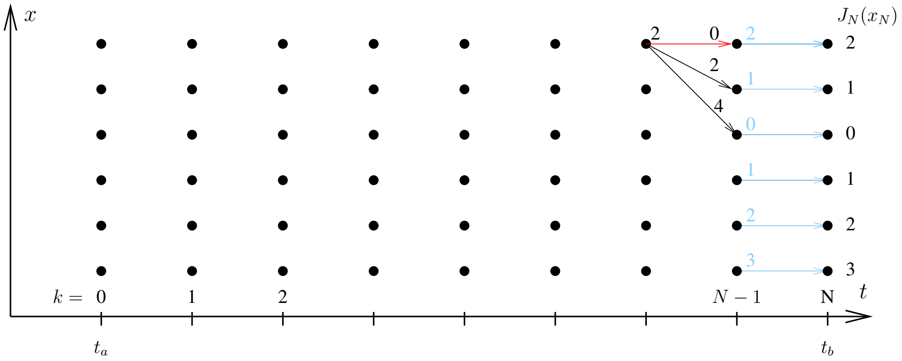
DDP Example

Graphical illustration of the solution procedure



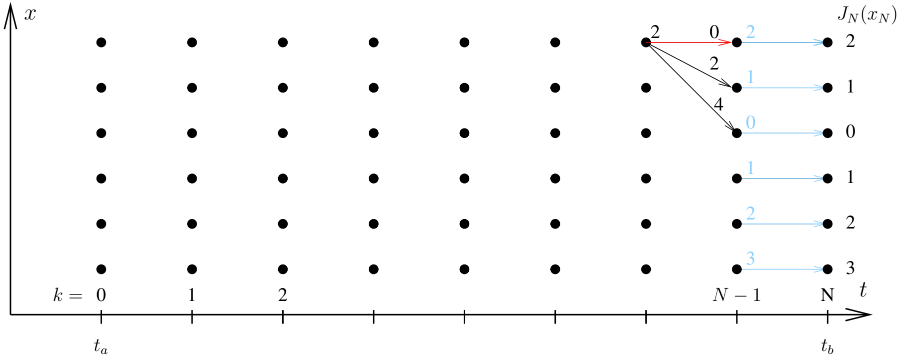
DDP Example

Graphical illustration of the solution procedure



DDP Example

Graphical illustration of the solution procedure



Matrix formulation benefits:

Each iteration compute all arcs using a vector/matrix

LINKÖPING UNIVERSITY

Algorithm Implementation

Generic DDP functions:

- `dynProg1D.m` – Solves a dynamic programming problem in one variable
- `dynProg2D.m` – Solves a dynamic programming problem in two variables

Algorithm Implementation

Generic DDP functions:

- `dynProg1D.m` – Solves a dynamic programming problem in one variable
- `dynProg2D.m` – Solves a dynamic programming problem in two variables

Problem specific scripts and functions that you need to complete:

- `testHybrids.m` – Template for setting up the problem
- `parallelHybrid.m` – Template for the parallel hybrid vehicle
- `seriesHybrid.m` – Template for the series hybrid vehicle

How to compute Arc Costs

Quasi-static approach (backward power flow)

How to compute Arc Costs

Quasi-static approach (backward power flow)

```
costVector = parallelHybrid(t_vec,SOC_start,SOC_final)
```

How to compute Arc Costs

Quasi-static approach (backward power flow)

```
costVector = parallelHybrid(t_vec,SOC_start,SOC_final)
```

- Calculate the whole bundle of arcs in one step.

How to compute Arc Costs

Quasi-static approach (backward power flow)

```
costVector = parallelHybrid(t_vec,SOC_start,SOC_final)
```

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

How to compute Arc Costs

Quasi-static approach (backward power flow)

```
costVector = parallelHybrid(t_vec,SOC_start,SOC_final)
```

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

How to compute Arc Costs

Quasi-static approach (backward power flow)

```
costVector = parallelHybrid(t_vec,SOC_start,SOC_final)
```

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

How to compute Arc Costs

Quasi-static approach (backward power flow)

`costVector = parallelHybrid(t_vec,SOC_start,SOC_final)`

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

- Computing time: `tic`, `toc`

How to compute Arc Costs

Quasi-static approach (backward power flow)

`costVector = parallelHybrid(t_vec,SOC_start,SOC_final)`

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

- Computing time: `tic`, `toc`
- Create matrices: `zeros`, `ones`

How to compute Arc Costs

Quasi-static approach (backward power flow)

`costVector = parallelHybrid(t_vec,SOC_start,SOC_final)`

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

- Computing time: `tic`, `toc`
- Create matrices: `zeros`, `ones`
- Modify matrices: `meshgrid`, `repmat`

How to compute Arc Costs

Quasi-static approach (backward power flow)

`costVector = parallelHybrid(t_vec,SOC_start,SOC_final)`

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

- Computing time: `tic`, `toc`
- Create matrices: `zeros`, `ones`
- Modify matrices: `meshgrid`, `repmat`
- Element-wise operations: `./` or `.*`

How to compute Arc Costs

Quasi-static approach (backward power flow)

`costVector = parallelHybrid(t_vec,SOC_start,SOC_final)`

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

- Computing time: `tic`, `toc`
- Create matrices: `zeros`, `ones`
- Modify matrices: `meshgrid`, `repmat`
- Element-wise operations: `./` or `.*`
- Logical indexing: `m_f(m_f<0)`

How to compute Arc Costs

Quasi-static approach (backward power flow)

`costVector = parallelHybrid(t_vec,SOC_start,SOC_final)`

- Calculate the whole bundle of arcs in one step.
- Add boundary and constraint checks.
- Be careful with: $\bar{v} = 0$, $Gear = 0$ and sign convention.

Useful MATLAB commands:

- Computing time: `tic`, `toc`
- Create matrices: `zeros`, `ones`
- Modify matrices: `meshgrid`, `repmat`
- Element-wise operations: `./` or `.*`
- Logical indexing: `m_f(m_f<0)`
- Trick: $P = T \quad {}^{sign(T)}$

Send parameters to parallelHybrid.m

The driving cycle used is set in testHybrids.m and is used in parallelHybrid.m. Two solutions to access T_z, V_z, and G_z

Send parameters to parallelHybrid.m

The driving cycle used is set in testHybrids.m and is used in parallelHybrid.m. Two solutions to access T_z, V_z, and G_z

1. Take the variables as inputs to dynProg1D and use them in the call for parallelHybrid.m

Send parameters to parallelHybrid.m

The driving cycle used is set in testHybrids.m and is used in parallelHybrid.m. Two solutions to access T_z, V_z, and G_z

1. Take the variables as inputs to dynProg1D and use them in the call for parallelHybrid.m
2. Use global variables.

testHybrids.m:

```
global V_z
load EUDC_MAN
```

parallelHybrid.m:

```
global V_z
```

V_z is available

Send parameters to parallelHybrid.m

The driving cycle used is set in testHybrids.m and is used in parallelHybrid.m. Two solutions to access T_z, V_z, and G_z

1. Take the variables as inputs to dynProg1D and use them in the call for parallelHybrid.m
2. Use global variables.

testHybrids.m:

```
global V_z
load EUDC_MAN
```

parallelHybrid.m:

```
global V_z
```

V_z is available

Easier solution since the scripts are prepared to use global variables.

Send parameters to parallelHybrid.m

The driving cycle used is set in testHybrids.m and is used in parallelHybrid.m. Two solutions to access T_z, V_z, and G_z

1. Take the variables as inputs to dynProg1D and use them in the call for parallelHybrid.m
2. Use global variables.

testHybrids.m:

```
global V_z
load EUDC_MAN
```

parallelHybrid.m:

```
global V_z
```

V_z is available

Easier solution since the scripts are prepared to use global variables.

First global variable definition, then assign value.

Send parameters to parallelHybrid.m

The driving cycle used is set in testHybrids.m and is used in parallelHybrid.m. Two solutions to access T_z, V_z, and G_z

1. Take the variables as inputs to dynProg1D and use them in the call for parallelHybrid.m
2. Use global variables.

testHybrids.m:

```
global V_z
load EUDC_MAN
```

parallelHybrid.m:

```
global V_z
```

V_z is available

Easier solution since the scripts are prepared to use global variables.

First global variable definition, then assign value.

Be cautious to not overwrite variables $\rightarrow$ Read only variables.

Solution Postprocess

How to retrieve the solution after calculating the `cost_to_go` matrix is not trivial.

Solution Postprocess

How to retrieve the solution after calculating the `cost_to_go` matrix is not trivial.

Helpful things to check:

- Start with the parallel hybrid.
- Check what are the outputs from `dynProg1D.m` and `dynProg2D.m`
- How are the costs stored in the DDP algorithm?
- How is the optimal path stored in the DDP algorithm?

LINKÖPING
UNIVERSITY

Debugging

- Is used to identify faults in the code
- It is possible to find and use the values of parameters and variables inside functions (that use local workspace).

Debugging

- Is used to identify faults in the code
- It is possible to find and use the values of parameters and variables inside functions (that use local workspace).

Matlab debugger

- Mouse click to the left in a m-file to add a debugger point. Possible to add conditional debugger point by right click (e.g. stop at a wanted index in a for loop).
- F10: Step forward (one line)
- F11: Step inside called function
- F5: Continue till next debugger point or till the end

Example in Matlab

Summary

- Understand how DDP works and how returns the solution.

Summary

- Understand how DDP works and how returns the solution.
- Implement your hybrid models in paper first.

Summary

- Understand how DDP works and how returns the solution.
- Implement your hybrid models in paper first.
- Use your notes to implement your code with the given templates.

Summary

- Understand how DDP works and how returns the solution.
- Implement your hybrid models in paper first.
- Use your notes to implement your code with the given templates.
- Make use of the debugging tool to polish your code.

Summary

- Understand how DDP works and how returns the solution.
- Implement your hybrid models in paper first.
- Use your notes to implement your code with the given templates.
- Make use of the debugging tool to polish your code.
- Answer the questions and discuss your results in the Hand-in report.

Thanks for your attention

Mahdi Morsali
mahdi.morsali@liu.se

www.liu.se

LINKÖPING UNIVERSITY