



# Altair

Innovation Intelligence®

## Integrated Optimization in System Models

Ramine Nikoukhah

Sim@SL October 2015

# Outline

---



- Applications of Optimization in System Modeling
- Standard method for Optimization
- Script Based Batch Simulation
- Example in VisSim SIMULATE (VSS)
- Limitations of the Classical approach
- Reverse Communication Optimization
- Implementation in VSS
- Examples
- Conclusion

# Applications of Optimization in System Modeling

---



**Optimization required at some point in many/most system applications**

**Optimization used mainly for**

- Parameter identification
- Feedback control design
- Open-loop control
- Filter parameter tuning, in particular for diagnosis

**Optimization cost and constraints expressed in terms of simulation results**

**Multiple simulations required to solve the optimization problem**

**Any optimization method may be used**

# Standard method for Optimization

## Master-Slave configuration

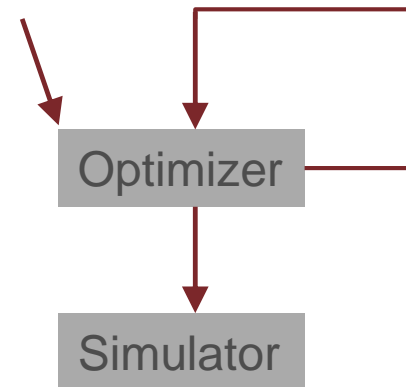
- Optimizer is the master
- Simulator is the slave

## Problem formulation

Find  $x$  which minimizes  $f(x)$

subject to

- $g_i(x) \leq 0, i=1, \dots, n$
- $h_j(x) = 0, j=1, \dots, p$



The functions  $f$ ,  $g$  and  $h$  evaluated by the simulator

The simulator evaluates the values using the results of System simulation

# Script Based Batch Simulation

---

## System Modeling tools with supporting scripting environments:

- NSP/Scicos (Scilab is the scripting language)
- VisSim SIMULATE (HyperMath is the scripting language)
- Matlab/Simulink

## Optimizer developed as a script

- Taking advantage of available optimization routines
- Cost function calls simulator in batch mode

## Requirements

- Availability of an optimization library in the scripting language environment
- Possibility to run system simulations from the scripting environment
- Ability to exchange data between the scripting and simulation environments

## Example in VisSim SIMULATE (VSS)

### Simple model of a 5 speed vehicle

- $T(\omega)$  : engine torque (table lookup)
- $g_i$  :  $i^{\text{th}}$  gear ratio
- $x$  : travelled distance

$$\dot{\omega} = g_i^2 T(\omega) q / c - a\omega - bv^2$$

$$v = c\omega / g_i$$

$$\dot{x} = v.$$

**Objective** find the “best” gear ratios: minimize time to reach 1km

### Optimization parameters:

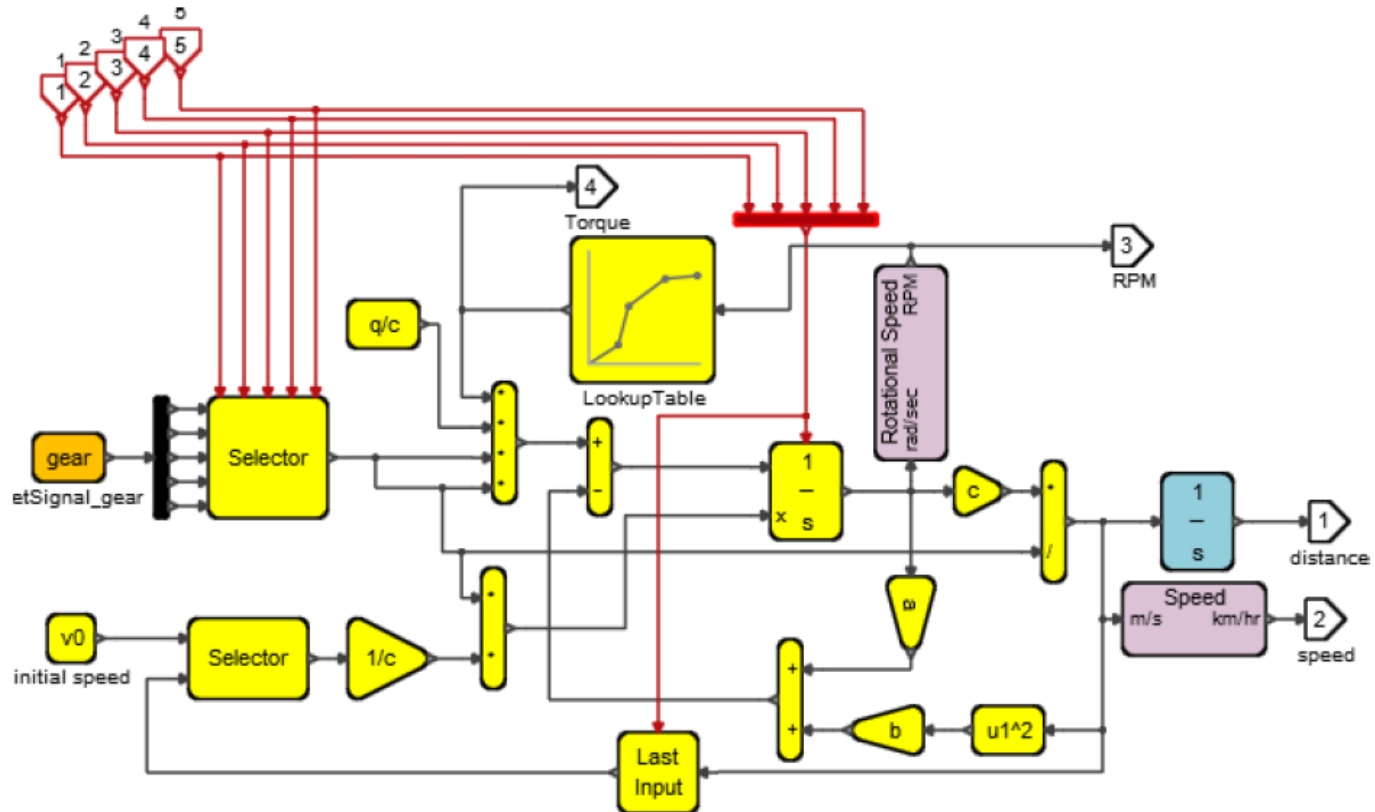
- $g_i$  : gear ratios (5 parameters)
- $T_i$  : times of gear shift (4 parameters)

### Basic assumptions

- Gear shifting is instantaneous
- Vehicle speed is a continuous time function
- Simulation is ended when 1km reached

## Example in VisSim SIMULATE (VSS)

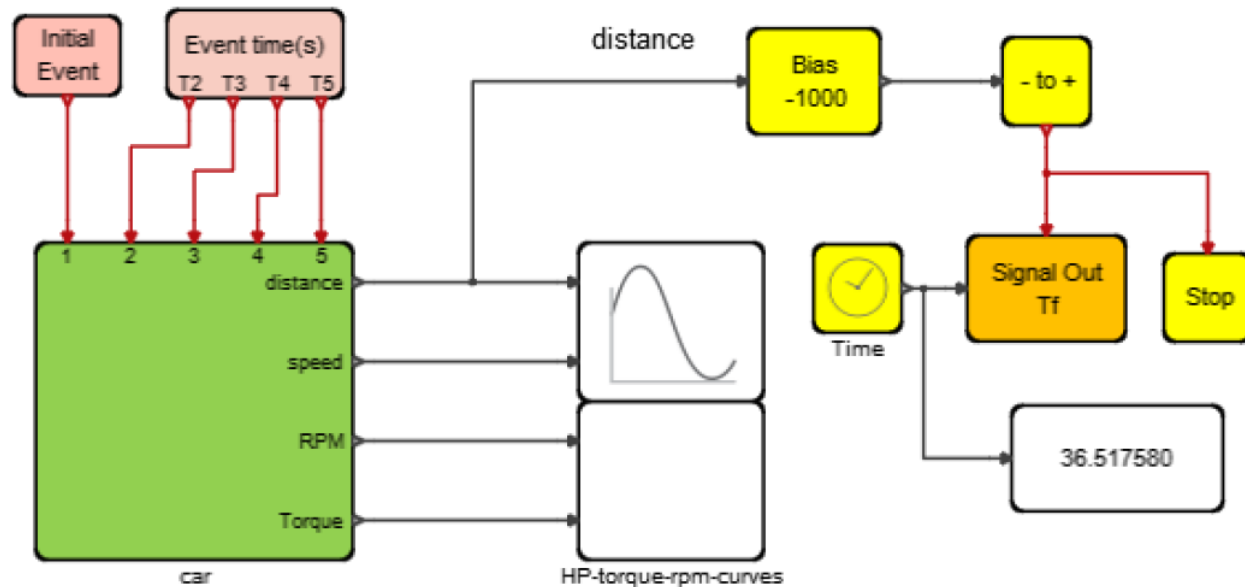
### Car model in VSS



Note that  $\omega$  jumps when gear is shifted

## Example in VisSim SIMULATE (VSS)

### Optimization model in VSS



Events are generated at gear shifting times (starting with first gear)

Travelled distance crossing 1km generates an event ending the simulation

Final simulation time returned to HyperMath (in Tf) via the Signal Out block



## Example in VisSim SIMULATE (VSS)

### Optimization script in HyperMath

```
global simobj,ev;  
model = "car_min_time_distance_1000_batch.scm";  
T0 = 4*ones(4,1);gear0=[3;2;1.5;1;0.5];  
p0 = [T0;gear0];  
Simobj = _vss.API::CreateSimulationObject(model);  
ev = _vss.API::VssCreateEvaluator(model,simobj);  
p,tf = FMinUncon('eval_cost',p0,[250],[1.e-4]);  
T = p([1:4]); gear = p([5:9]);
```

### Cost evaluation function called by FMinUncon

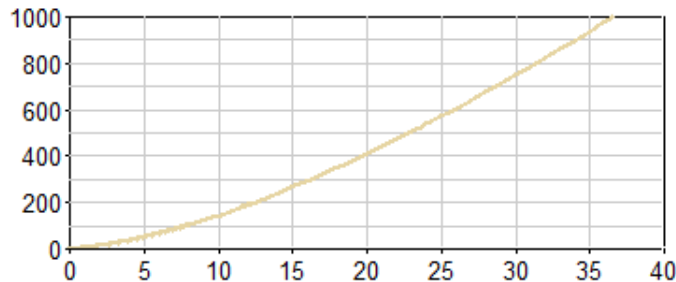
```
function eval_cost(p,d)  
    global simobj,ev;  
    vss_context = {}; vss_context.T = p([1:4]); vss_context.gear = p([5:9]);  
    _vss.API::VSSBatchRun(ev,_fenv_,vss_context,simobj)  
    return Tf.ch(1).data;  
end
```

## Example in VisSim SIMULATE (VSS)

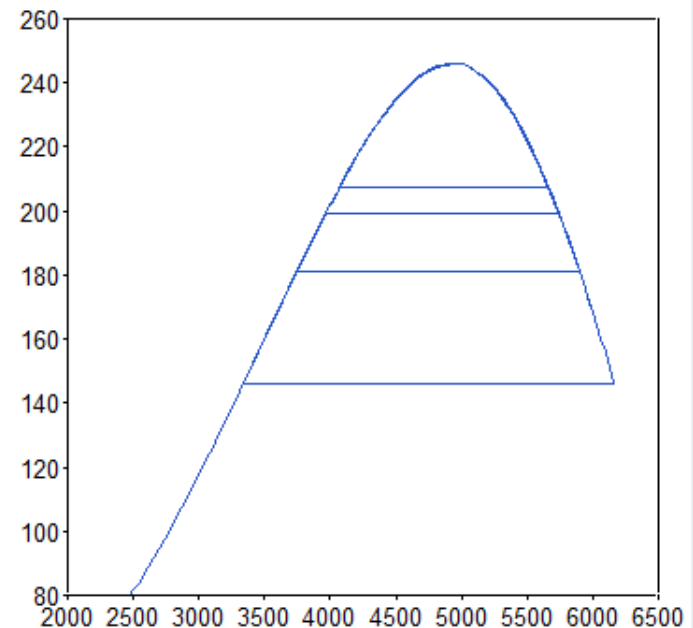
### Optimization result

- Stay closes to max power (and not max torque) RPM
- Change gear when no jump in power occurs

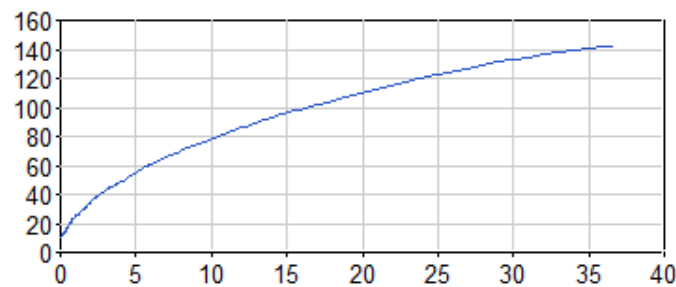
Travelled distance



Horse Power



speed



# Limitations of the Classical approach

---

**System model not self-contained**

**Requires development in two different environments**

**Complexity:**

- Data exchange between scripting environment and simulation environment
- Error reporting and debugging

**In some tools, batch mode simulation is blocking**

# Reverse Communication Optimization

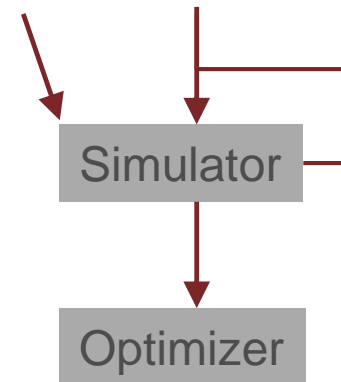
## Master-Slave configuration

- Optimizer is the slave
- Simulator is the master

**Well-know in the optimization community**

**Supported in some optimization packages**

**Not much used, in particular in system modeling**



## Unlike in standard mode:

- Simulator controls the main loop and calls the optimizer giving cost of parameter
- Optimizer returns a new parameter value and indicates whether it is the solution or additional iteration is required

# Implementation in VSS

---

## Reverse communication optimization:

- Essential for implementing optimization in System Modeling environments

## Other requirements: being able to

- Restart or Stop a simulation by a simulation event. This is implemented in VSS using a Restart/Stop block, which may be activated by an event.
- Keep values from one simulation run to the next. This is achieved by blocks *To* and *From Base*. Variables may be written and read and their values are kept in the base environment.

**The actual implementation is done as a regular block in VSS**

## Implementation in VSS

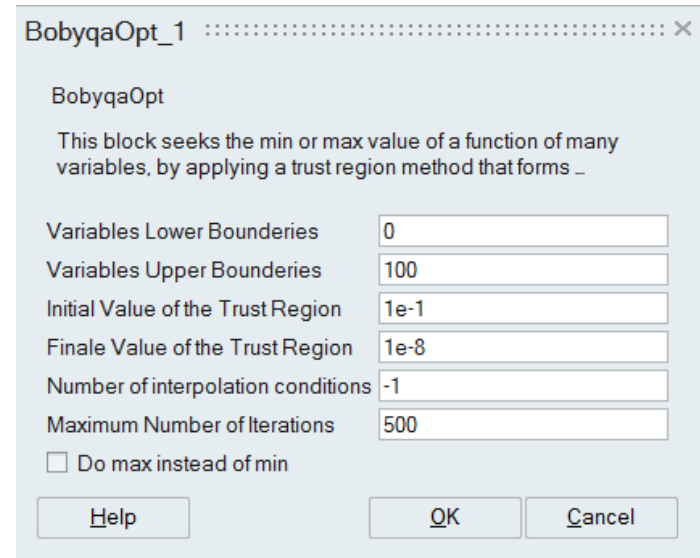
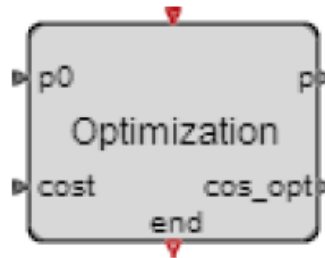
### Optimization block

#### Inputs:

- $p_0$  : initial/current value of parameter
- $cost$  : cost associated with  $p_0$

#### Outputs:

- $p$  : new value of parameter proposed by optimizer
- $cost_{opt}$  : optimal cost found by the optimizer at the end of optimization
- $end$  : event generated when optimizer finds the solution



The image shows a dialog box titled "BobyqaOpt\_1". It contains the following fields and options:

- BobyqaOpt**  
This block seeks the min or max value of a function of many variables, by applying a trust region method that forms \_
- Variables Lower Boundaries**: 0
- Variables Upper Boundaries**: 100
- Initial Value of the Trust Region**: 1e-1
- Finale Value of the Trust Region**: 1e-8
- Number of interpolation conditions**: -1
- Maximum Number of Iterations**: 500
- ☐ Do max instead of min
- Buttons**: Help, OK, Cancel

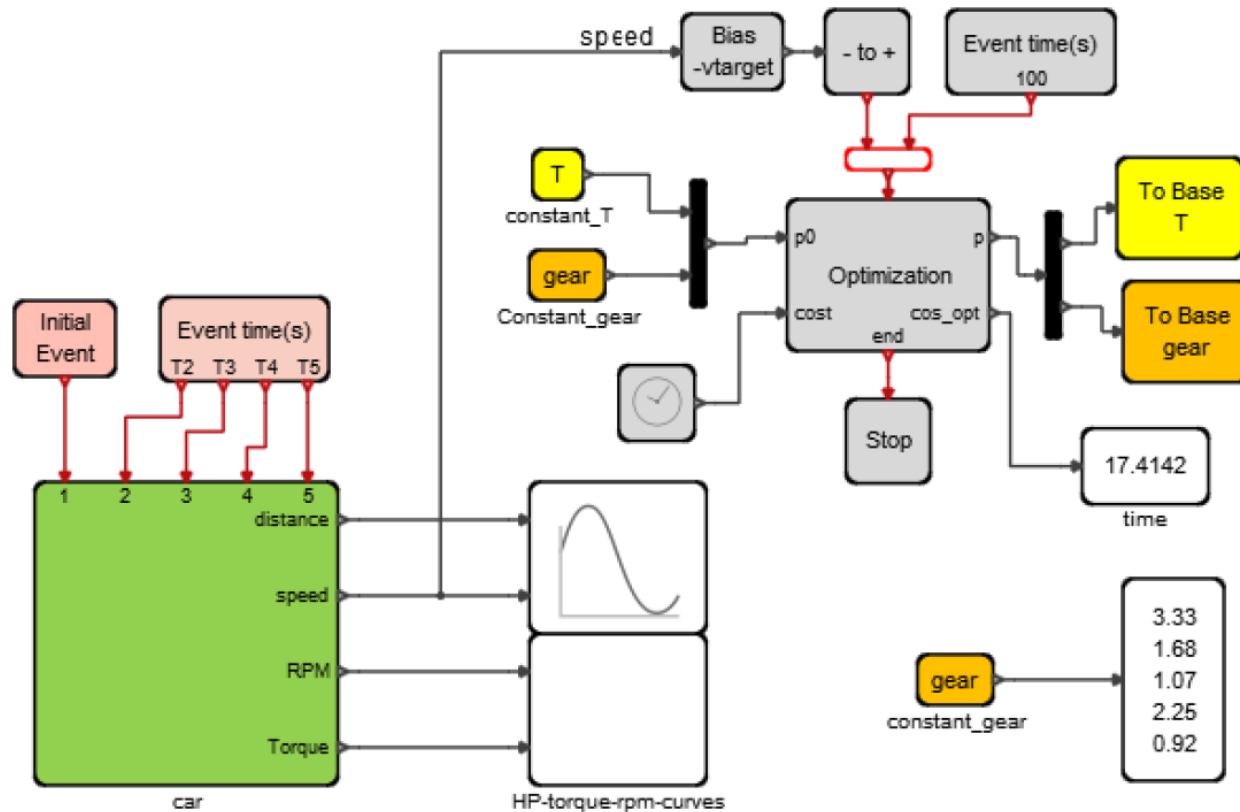
When activated, usually at the end of a simulation run, the Optimization block calls the optimizer providing  $p_0$  and  $cost$ .

The optimizer either proposes a new  $p$  or ends the optimization by generating  $end$  event and providing optimal  $p$  and  $cost$ .

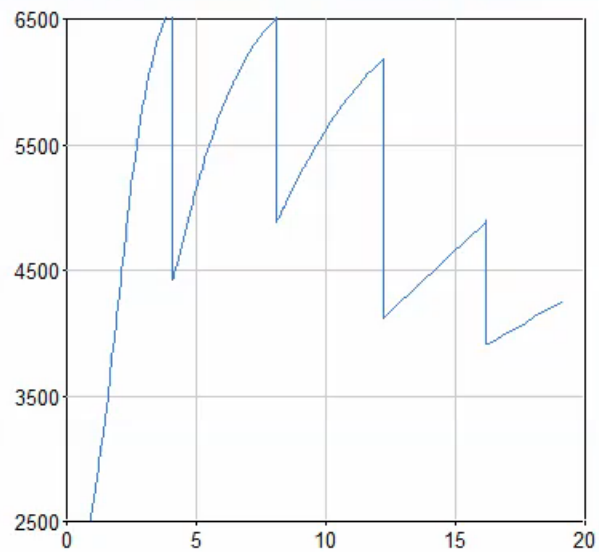
## Examples

### Gear optimization problem with different objective:

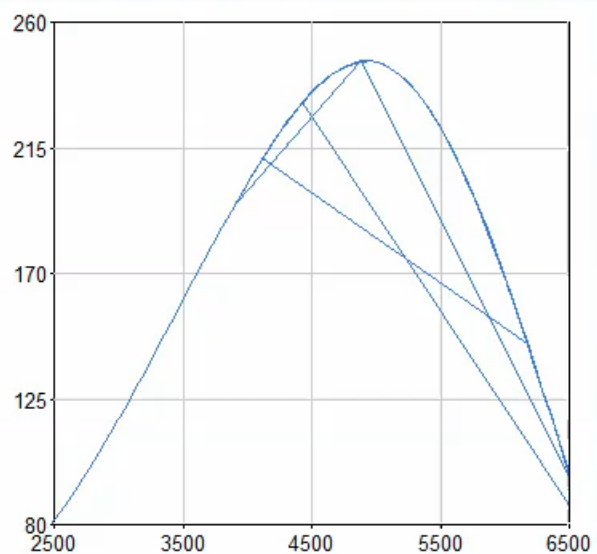
- Minimize the time to attain a given speed  $v_{\text{target}}$
- Optimization problem may be formulated within the model:



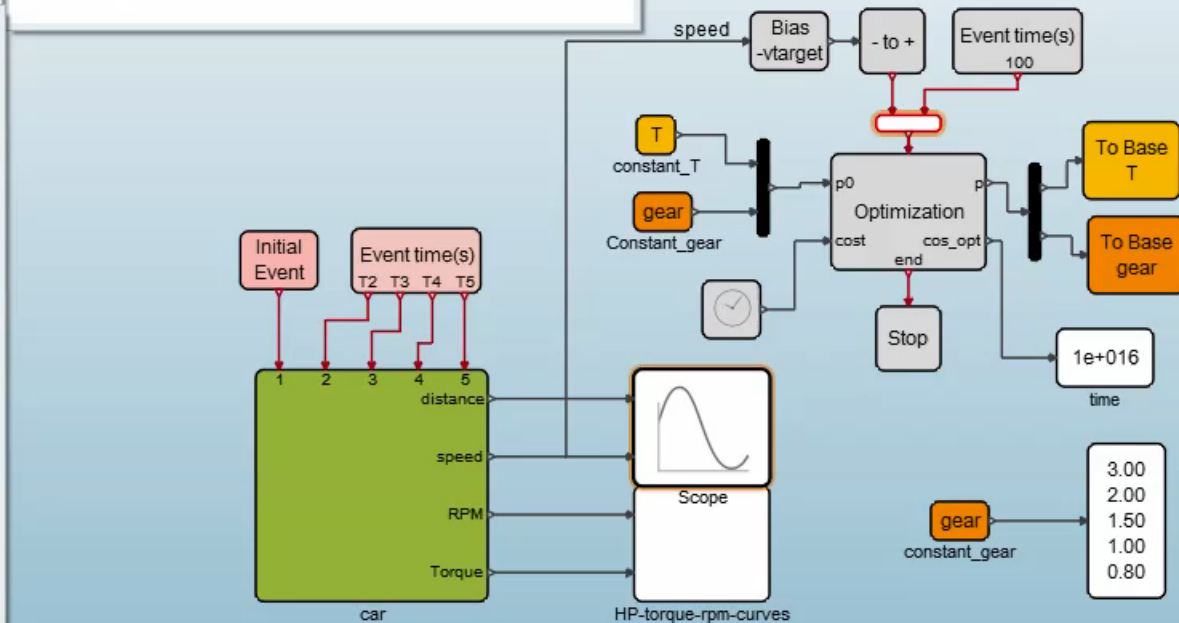
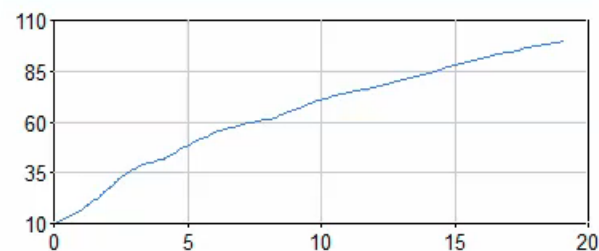
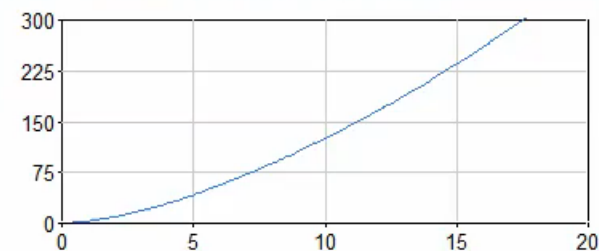
car\_min\_time\_speed\_vtarget\_2/HP-torque-r... :::: x



car\_min\_time\_speed\_vtarget\_2/HP-torque-r... :::: x



car\_min\_time\_speed\_vtarget\_2/Scope :::::::::: x





**Min Max problem: Gear optimization problem but assume the gear shifting time cannot be precise so the objective is:**

- 
- The schematic diagram illustrates a multi-objective optimization algorithm for a vehicle simulation. The process is divided into two main functional areas: simulation and optimization.
- Simulation Block (Left):**
- Inputs:** An **Initial Event** and four **Event Variable Delay** blocks are connected to a central **car** block.
  - Car Block:** A green block representing the vehicle model, which outputs **distance**, **speed**, **RPM**, and **Torque**.
  - Scope:** A block that receives the **distance** output from the car block and displays a graph of distance over time.
  - HP-torque-curves:** A block that receives the **Torque** output from the car block and displays a graph of torque over time.
  - gear Block:** An orange block that outputs a list of gear values: 4.67, 2.58, 1.78, 1.18, 0.86.
- Optimization Block (Right):**
- Top Optimization Block:**
    - Inputs:** **dT** (time step) and **Event time(s) Tf** (total time).
    - Outputs:** **p0**, **p**, **cost**, **cos\_opt**, and **end**.
    - Intermediate Outputs:** **To Base dT** and **1e+016 Distance**.
  - Bottom Optimization Block:**
    - Inputs:** **T** (torque) and **gear** (gear selection).
    - Outputs:** **p0**, **p**, **cost**, **cos\_opt**, and **end**.
    - Intermediate Outputs:** **To Base T**, **To Base gear**, and **-1e+016 MaxDistance**.
  - Stop Block:** A block that receives the **end** signal from both optimization blocks.



## Conclusion

---

### **A BobbyQA based Optimization block is available in VSS**

- User may create new optimization block based on other optimization routine (provided it works in reverse communication mode)

### **May be used as an alternative to script based optimization**

- May also be used as a complement: for example to formulate mixed Min Max optimization problems

### **Models including Optimization blocks may be parameterized to operate either in optimization or simple simulation modes**

- Often switch is made to simple simulation once optimal parameters found

### **Stochastic optimization (for example Monte Carlo methods) may be implemented using the Optimization block**