Simulation of Robotic Systems, Sensors, Environment and Processes

Ng Phuc Duc

1, Introduction:

- Simulation as a Digital Twin of the Real Model
- Simulation accross different levels

  Process ⇒ Module ⇒ Plant ⇒ Factory

- +) Interdisciplinary:
  - Kinematics / dynamics
  - Communication Infrastructure
  - Data Processing Systems
  - Workers / Environment
  - Sensors / Actuators
  - User Interfaces
  - Superordinate Control Systems
  - Neighboring cells
- +) Aspects: must be taken into account
  - Able to calculate energy, cost..
  - Able to vary parameters, configuration..
  - Coordinate planning, layout..

⊛ "Such Multiformalism modelling capability is important since the world does not usually lend itself to using on form of abstraction at a time" [Zeigler 2000]

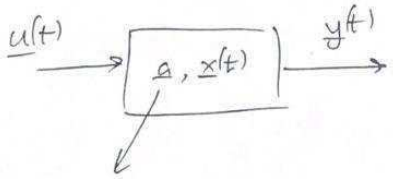
- Simulation approaches: choose appropriate modelling approaches & tools
- Digital Twins: take care to reflect the physical structure
- FMI: assure interfaces can be used
- Simulation processes, V& V (Verification & Validation): assure simulation is correct
- Digital Twins, XiL, MMI: Use simulation multiple times

L2) ⊛ Basic terms & definitions

- 3 pillars of science & engineering { theory, experiment, simulation

↳ Definition:

- Experimental simulation: a process simulated by a real process.
- Theoretical simulation: a system is mathematically modelled, simulated
- Computer simulation: computer as the basis
- System: set of inter-related elements, seperated from their environment
- System of system (SoS): set of systems for a greater task
- System is characterized by: { input, output, state variables; borders; subsystem, sequence structure..



system params

$a_i, x_i, u_i, y_i \notin = (w_a, d_a, l_a) \in \mathbb{V}_a$

value ($\in \mathbb{W}_a$) | type ($\in \mathbb{D}_a$) | unit ($\in \mathbb{L}_a$) (physical quantity)

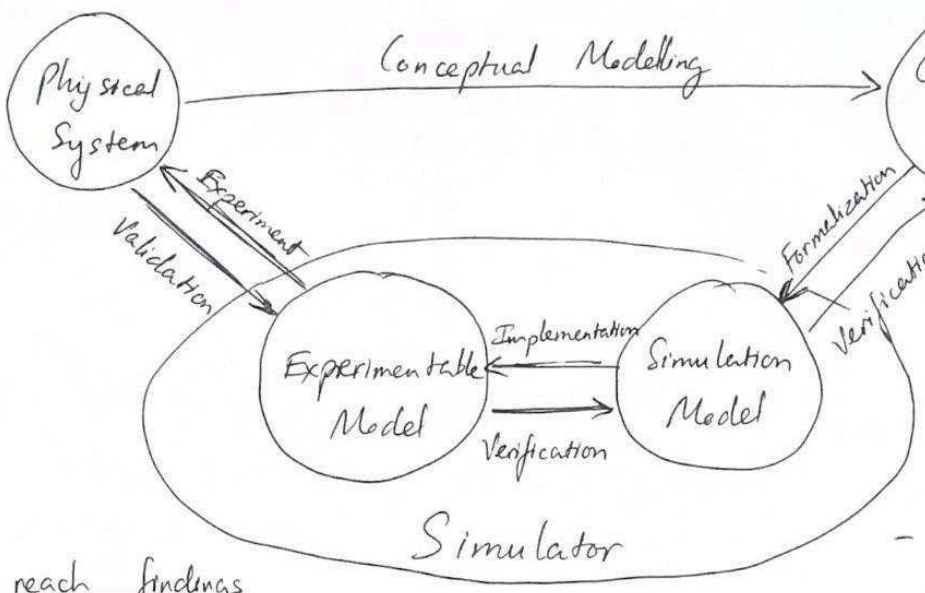
- Model: simplified reproduction of { existing, planned } system

whether in { conceptual, physical } system

3 main features { mapping (represents a real system); abstraction (reduced / simplified compared to real model); pragmatic (to be used, fullfill some tasks / interest)

- Simulator: simulation tool (software..)

Tasks: { create a model; execute



- Set of concepts (abstract description or specification) & relationship among those
- not executable
- has to be formalized

aims to reach findings which are transferable to reality

executed intergrating both { params, algorithms

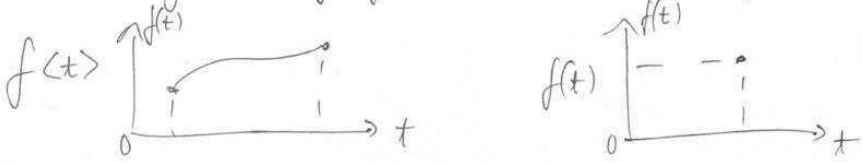
formal
- Simulation-specific reproduction
- Built from { simulation languages, tools
- Required detailed data

$M := (a, A, U, Y)$

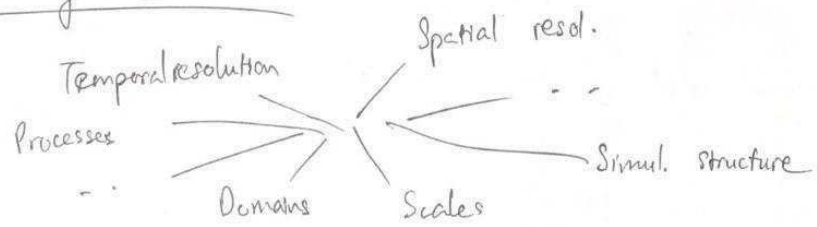
a: params, A: algorithms, U: in, Y: out

- Simulation state $\underline{s}(t) = \begin{bmatrix} \underline{x}(t) \\ \underline{a}(t) \\ \underline{A}(t) \end{bmatrix}$
- Time interval $T = \langle t_{min}, t_{max} \rangle$
- Input segment $u\langle t \rangle$: $u(t)$ in $0 \leq t' \leq t$

⇒ Function segment: a segment of func values, not a point value



+) Classification of simulations:



+) Simulation process:

Determine problem → Conceptual Modelling → Formalization → Implementation → Execute → Post processing → Interpretation

System → Conceptual Model → Simulation Model → Exp.. Model → Simulation Data → Result

⊗ Domain-specific simulation algorithms for robotics

+) Finite Element Method (FEM): numerical method for solving PDEs
- space is discretized ⇒ mesh
- deal with problems of deformations, temperature, pressure distributions

+) Kinematics & dynamics

⊗ Domain-independent simulation algorithms

- Equation-based Simulation: describe system with DAE
  scripting language (MATLAB, ..)
- Block-oriented Simulation (Signal oriented): functional diagram (includes functional blocks)
  ex: Simulink
- Object-oriented Simulation: inp, out not defined before hand
  but derived from connections
  ex: Modelica
- Discrete Event Simulation: events + actions + transitions

+) System Engineering: [interdisciplinary approach
enable the realization of successful systems

- Tasks: explore, document, synthesize, verify ..
- Model-based Systems Engineering: the formalized application of modelling

⊗ Requirements for simulation technology:

+) Functional requirements

+) Non-functional requirements: close to reality — performance
real time — scalability
multi-platform support — automatability
..

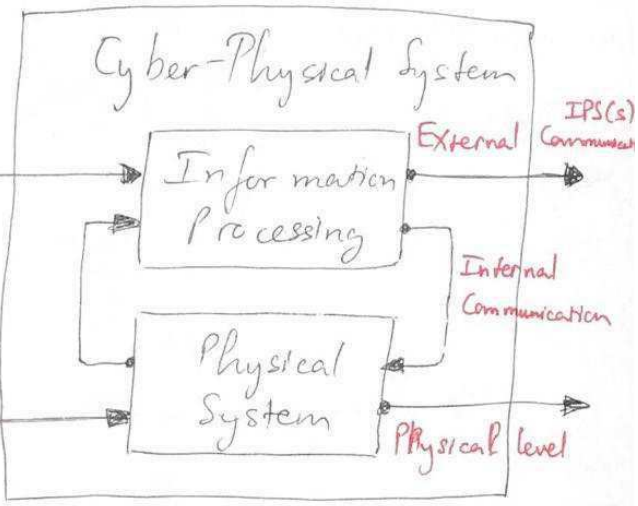
The Digital Twin

L3)

1) Cyber - Physical System:

- Definition: links real | objects with virtual ones (processes)
- Asset: an entity [owned by organ. / has value
- Industry 4.0 component = Asset + Asset Administration shell (AAS)

(Physical System) + (Info. Processing System)



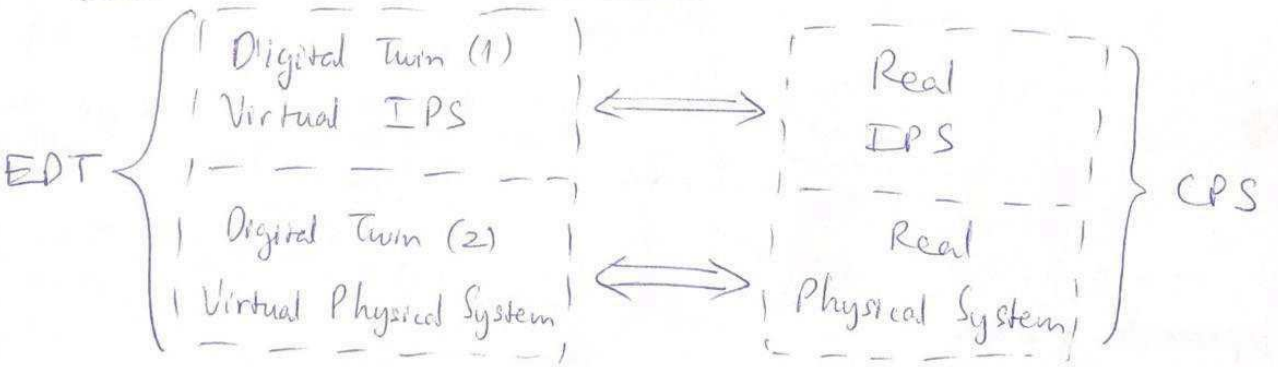
2) Digital Twin:

- Digitization vs Digitalization

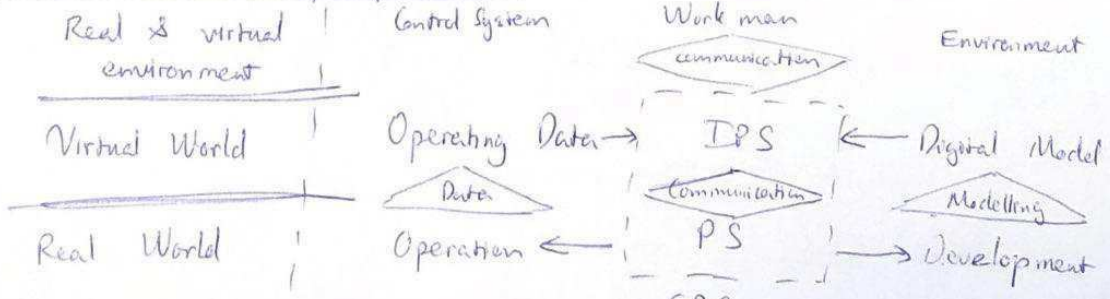
convert analog → digital

use digital artifacts in [products / processes

+) Experimental Digital Twin (EDT)



+) Typical simulation perspectives:



3) The Digital Twin History:

have the idea but not the tools, data yet

1960 — 2003 — 2010 — 2015 — 2016 — 2018

1960 NASA
an identical prototype for flight preparation

2003 "Information Mirroring Model"

2010

2015 Industry 4.0 & Simulation (Siemens)

2016 Industry 4.0 & Networking (Bosch, SAP, GE ..)
The methodology of EDT

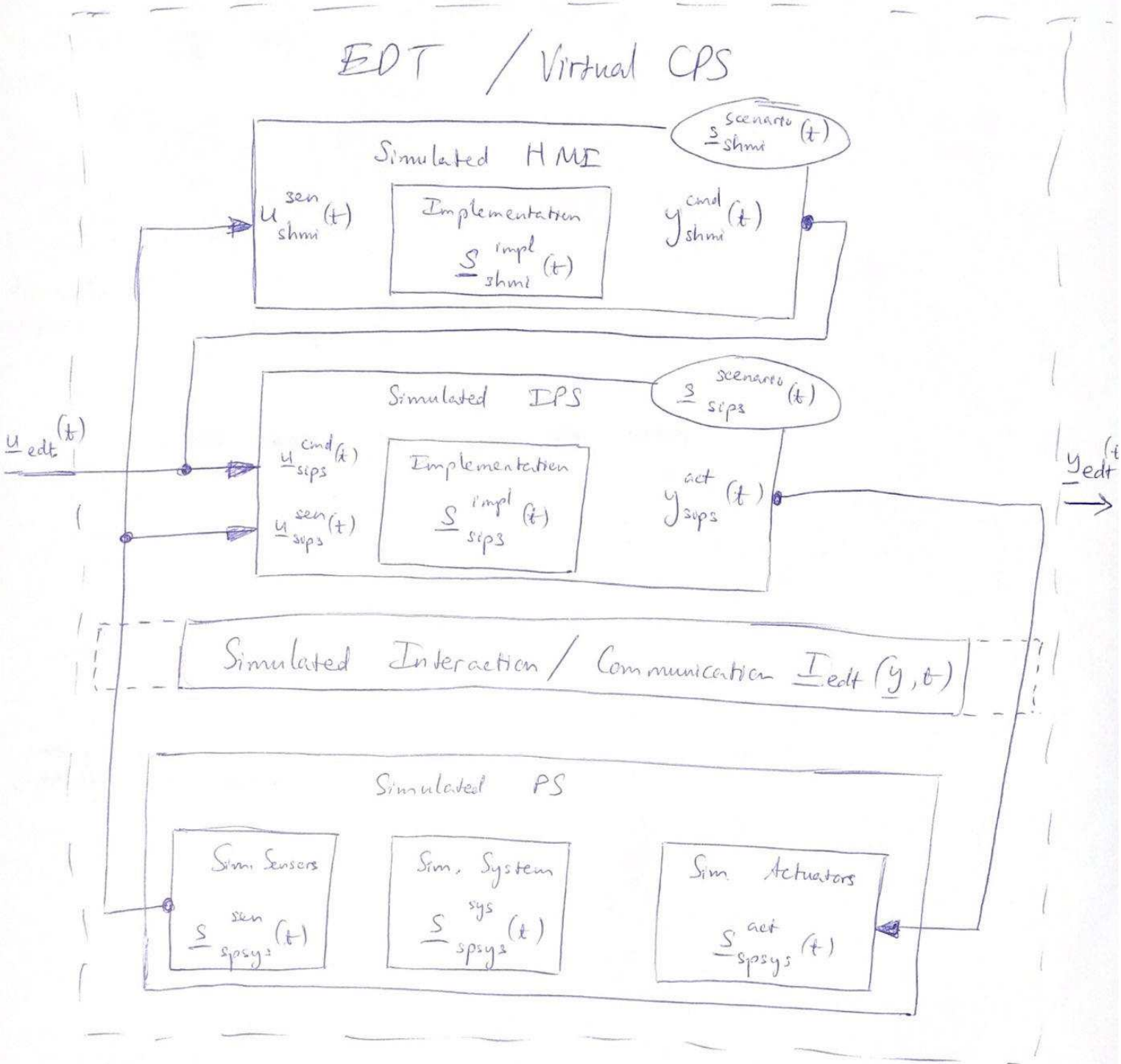
2018 DTs for MBSE (Model-based System Engineering)

4) Defining the term Digital Twin (DT)

- Digital Twin: virtual digital representation .. sufficient { requirements .. / allow interaction ..
- A Twin Type: exemplary description of a Digital / Real Twin
  Ex: Length of a car is what is being considered
- Hybrid Twin = Digital Twin + Real Twin
- Aspects of DT:
  - Object of Interest (what is the Real Twin) — a Component, IoT, system, commissing
  - Digital Artifacts (the elements)
  - Technical Realization: able to use & get benefit from DT
  - Tasks (purposes ..)

- EDT = $DT_{cps} = DT_{psys} + DT_{ips}$
  representation of a CPS in experimentable model
  $RT_{cps} = RT_{psys} + RT_{ips}$
- Typical Digital Artifacts: Structure & Semantic, Metadata, Data, Models, Functions, Interaction, Prognosis
- 4 Main Tasks of DT: Single-Source-of-Truth, Prognosis, Conversation partner, Mediator

5) CPS and EDT in detail:



$\hat{=}$

Real Twin / Real CPS

- Actuator: device → influences → process
- Sensor: ——— → measure variables → standardized output signals

State vector

$$\underline{s}_{edt}(t) = \begin{bmatrix} \underline{x}_{edt}(t) \\ \underline{a}_{edt}(t) \\ \underline{A}_{edt}(t) \\ \underline{c}_{edt}(t) \\ \underline{I}_{edt}(t) \end{bmatrix} = \begin{bmatrix} \underline{s}_{spsys}(t) \\ \underline{s}_{sips}(t) \\ \underline{s}_{shmi}(t) \end{bmatrix}, \quad s_i = (v_i, d_i, l_i) \in \mathbb{V}$$

$$\mathbb{V} := \mathbb{W} \times \mathbb{D} \times \mathbb{L}_a$$

value type unit

connections

Interaction infrastructure

- Initial state $\underline{s}_{edt}(0)$
- Simulation: $\underline{s}_{edt}(t) = \Gamma(\underline{s}_{edt}(0), \underline{u}_{edt}\langle t \rangle, t)$
- Output: $\underline{y}_{edt}(t) = \Phi(\underline{s}_{edt}(t), \underline{u}_{edt}(t), t)$
- Model: $\mathcal{M}_{comp} = (\underline{a}_{edt}, \underline{A}_{edt}, \underline{U}_{edt}, \underline{Y}_{edt}, \underline{c}_{edt}, \underline{I}_{edt}, \underline{M}_{comp})$

sub-module (component)

params, algor, Inp, out, connections, Interaction, Infrastr.

$$\mathcal{M}_{comp} \in \mathbb{M}_{comp} := \mathbb{V}_a^{n_a} \times \mathbb{V}_A^{n_A} \times \mathbb{P}^{n_u} \times \mathbb{P}^{n_y} \times \mathbb{V}_c^{n_c} \times \mathbb{V}_I^{n_I} \times \mathbb{M}_{comp}^{n_{comp}}$$

$$\mathcal{M}_{edt} \in \mathbb{M}_{comp}$$

- A Port models an interaction point between system & its environment
- A Connection allow | info / material | flow exchange between ports (compatible)
  (classify: direction, unit, info param, disciplines, sys. architecture..)
- the interface specifies | structure / behaviors that system [provides / requires] its environment

- EDT scenarios: network of EDT connected to replicate an application scenario

Emergence: when an entity is observed to have extra properties (its parts do not have on their own)

6, EDT-based methods: . . .

Model-based Systems Engineering

2q,

1, Definitions

- MBSE: formalized application of modelling
  to support
  - system requirements
  - design
  - analysis
  - validation, verification

- 3 major aspects:
  - System Architecture
  - System Behavior
  - System Requirements

- System Architecture: elements & relationships

Functional ("logical")	Structural ("physical")
- With which devices this func is actually achieved - Virtual / Conceptual ports	- How are things physically connected? - Concrete ports.

⇒ Typical models generated:
- behavior
- temporal
- structural
- mass
- layout
- network

(model)

- SysML (OMG Systems Modelling Language): a language not tool

international standard

- 3 diagram classes:

Structure Diagram		Behavior Diagram	Requirements Diagram
bdd Block Definition Diagram	ibd Internal Block Diagram	act Activity Diagram	req Requirements Diagram
Define components & their relationship	Internal structure of a block	Transformation of inputs ⇒ outputs through sequence of actions	Define requirements (& their relationships) & testcases
+) Block: describe structure of elements or system Represent a components	+) Elements: - Blocks - Ports	+) Elements: - Action	+) Elements - Requirement - Testcase
+) Associations: - Generic rela. between blocks - Aggregation/ Composition	+) Associations: - binding Connector - itemFlow	+) Typical representations (Nodes) - Initial Node - Act. Final Node - Fork Node - Join Node - Merge Node - Decision Node - Flow Final Node	+) Associations: - Derive - Contain - Satisfy - Copy - Verify - Refine - Trace

Simulate an EDT-Scenario

From models to simulation

LS,

F_i, Φ_i

EDT-Scenario → EDT ⇒ EDT Components → Simulation Entities ⇒ Simulation Tasks

$M_{scenario}$ includes many EDTs

M_{edt} each EDT with some components

M_{comp}

M_{entity} (Simulators in 1 domain)

M_{task} (for extra combination & sequence)

1) Simulation Entities, Continuous time vs Discrete Event

+) Mathematical description for continuous-time & discrete-event simulation approaches

	Continuous-time	Discrete-Event
transition func	$\dot{s}_c(t) = f_c(s_c(t), u_{entity}(t), t)$	$s_d(t^+) = f_d(s_d(t), u_{entity}(t), t)$
2ndary cond. func	$0 = h_c(s_c(t), \dot{s}_c(t), u_{entity}(t), t)$	$0 = h_d(s_d(t), u_{entity}(t), t)$
output func	$y_{entity}(t) = g(s_c(t), u_{entity}(t), t)$	$y_{entity}(t) = g(s_d(t), u_{entity}(t), t)$
time advance func.	$t_{R,next}(t) = T(s_c(t), u_{entity}(t), t)$	$t_{R,next}(t) = T(s_d(t), u_{entity}(t), t)$

$t = (t_R, t_I) \in \mathbb{T} = \mathbb{R} \times \mathbb{N}$

t_R: simulation time; t_I: event count

+) Super-dense time: the time-representation for both { continuous time, Discrete event } to distinguish multiple events at the same time

Previous state & event:

$$^{\bullet}s = s(^{\bullet}t)$$

$$^{\bullet}t = {}^{\bullet}(t_R, t_I) \Leftrightarrow \begin{cases} t^- & \text{if } t_I = 0 \\ (t_R, t_I - 1) & \text{if } t_I > 0 \end{cases}$$

$$t^- = (t_R, t_I)^- \Leftrightarrow \left(\lim_{\varepsilon \to 0} (t_R - \varepsilon), 0 \right)$$

Generic Simulation Entities:

Merging continuous time & discrete-event simulation

Sim. model Mentity = $(\underline{a}, \underline{A}, \underline{U}, \underline{Y}, \underline{c}, \underline{I}, \underline{M}^{comp})$ — Implement simu. func. Γ & Φ from ↓ , based on Sim. model Mentity

Simulator

Simulation Entity

Time:	$t \in \mathbb{T}$
State:	$\underline{s}(t) = [\underline{s}_c(t), \underline{s}_d(t)]^T \in \mathbb{V}_s^{n_s}$
Transition func s.	f_c , f_d
Secondary functions.	h_c , h_d
Output func:	g
Time advance funct	T

$\underline{u}(t) \in \mathbb{V}^{n_u}$; $\underline{u} \in \mathbb{P}^{n_u}$ → Simulation Entity → $\underline{y}(t) \in \mathbb{V}^{n_y}$; $\underline{Y} \in \mathbb{P}^{n_y}$

$t_0 \in \mathbb{T}$, $\underline{s}(t_0) \in \mathbb{V}^{n_s}$ ↑ Simulation Entity ↓ $\underline{s}(t) \in \mathbb{V}^{n_s}$; $\underline{\dot{s}}_c(t) \in \mathbb{V}^{n_{s_c}}$

Executing continuous time Simulation Entities

- $\underline{s}_c(t) = \underline{s}_c(0) + \int_0^t \underline{\dot{s}}_c(\tau)\, d\tau = \underline{s}_c(0) + \int_0^t f_c(\underline{s}_c(\tau), \underline{u}_{entity}(\tau), \tau)\, d\tau$

- $\underline{s}_c(t_{R,next}(t)) = \underline{s}_c(t) + \int_t^{t_{R,next}(t)} f_c(\underline{s}_c(\tau), \underline{u}_{entity}(\tau), \tau)\, d\tau$

$\Rightarrow$ $\underline{s}_c(t+h) = \underline{s}_c(t) + h \cdot f_c(\underline{s}_c(t), \underline{u}_{entity}(t), t)$; $h = t_{R,next}(t) - t$

Hybrid Simulation Entities

2) Coupling of Simulation Entities, Tasks & Simulators

still a hard problem (not fully & perfectly solved)

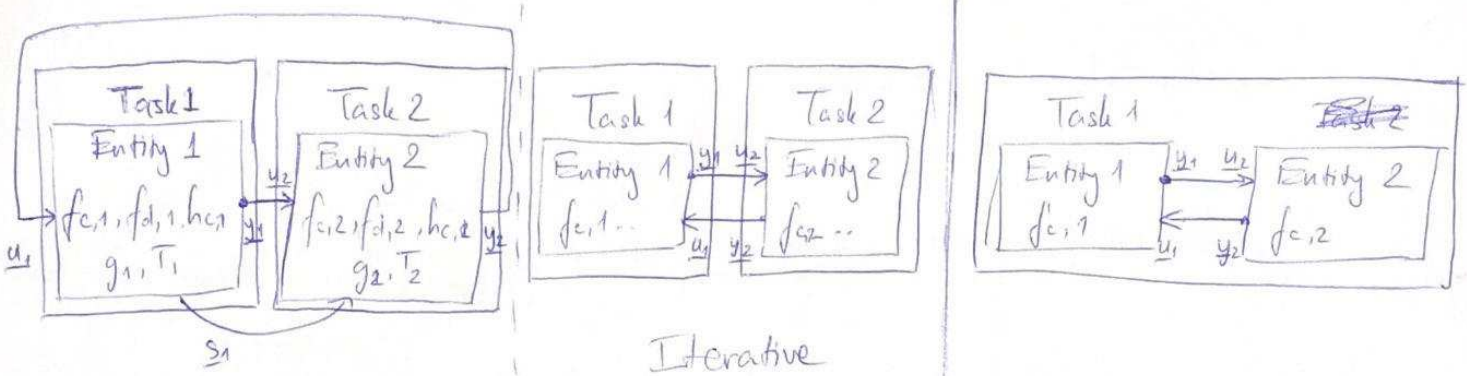
this problem arise as we have many simulation entities

- State types: $\underline{s}_c$ & $\underline{s}_d$

3 event types

- time event step → integration step times (for continuous)
- discrete time event → discrete event
- State event → when continuous states cross threshold..

+) Weak & strong coupling of continuous-time Sim Entities



Propagation

Iterative

Weak coupling

only exchange outputs

Strong coupling

need more exchanges

every entities are closely related

Prerequisites

- output $\underline{y}_{task}(t)$ can be read
- input $\underline{u}_{task}(t)$ can be specified at times t_j
- Evaluation of $\underline{y}_{task}(t)$ strictly monotonously with know increment h

Sim. Entities work according to the standard form

$\underline{s}(t)$, $\dot{\underline{s}}(t)$ are known

optional:
- $s(t)$ can be init any time
- free to choose increment h
- possible to step back time

~~+) Cross-domain system modelling~~

+) Comparison of ↗

- Both: - interface & exchanged quantities must be "semantically compatible"
  - API to control sim. entities necessary

- step times can vary but have to smaller than com. step time
- Work quasi always

all simulation entities must be calculated at the same times

Internal structures must be the same

+) Temporal course of system behavior

- Phy/Math. Model Description
  - one integrator
  - 2/more integrators → strong coupling
- Behavior Models
  - 1 integrators → strong coupling
  - 2/more integrators → weak coupling

3) Scheduling of Simulation Tasks:

+) Scheduler: coordinate simulation process & time control
calls sim. tasks (either cyclically
or to asynchronously occuring events

+) Scheduling: the temporal assignment of (activities / tasks) to limited resources

+) Simulation time vs real-time / wall-time

- Hard real-time: Sim. time matchs wall time
- Soft real-time: ——— (almost / most of time) match wall time
- Sim. time is faster than wall time

+) Cross-domain system modelling vs. simulator coupling.

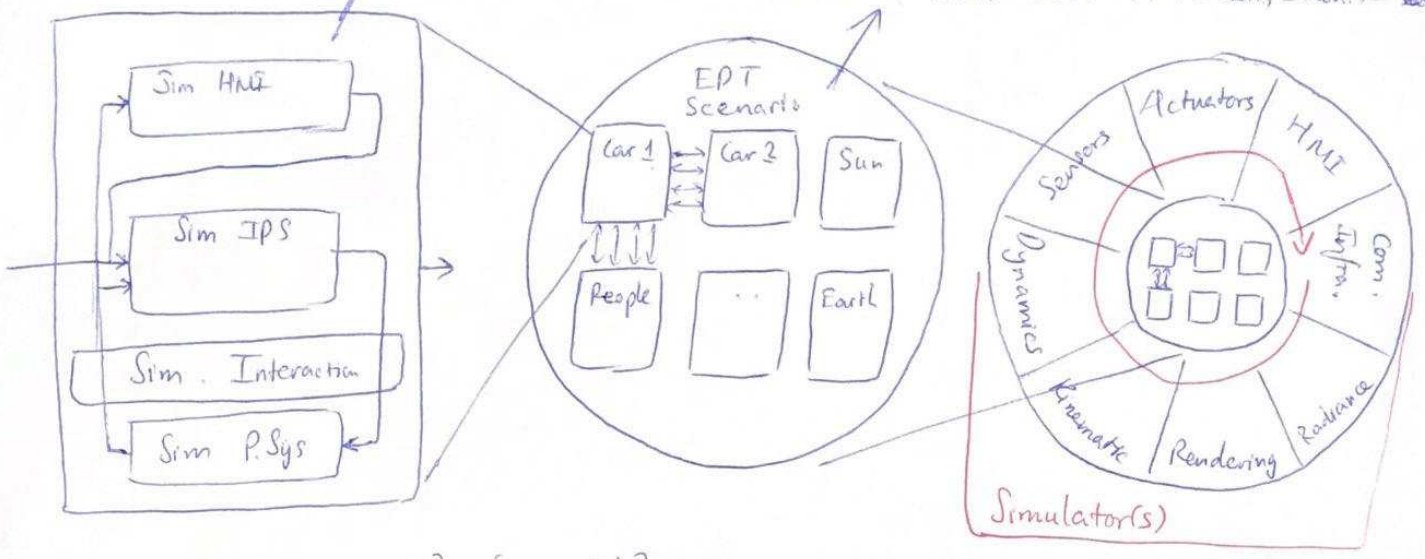
- Possible via different
  - development tools
  - descriptive models
  - Key Performance Indicators
  - Temporal Course of System Behavior

+) Communication step time (only for weak coupling)

cause each Simulation Entities may use their own integration time scheme
& only communicate with each others at communication step times

$$M_{edt} = [\underline{a}_{edt}, \underline{A}_{edt}, \underline{U}_{edt}, \underline{Y}_{edt}, \underline{c}_{edt}, \underline{I}_{edt}, \underline{M}_{comp}]$$

$$M_{scenario} = (\underline{a}_{scen.}, \underline{A}_{scen.}, \emptyset, \emptyset, \underline{c}_{scen.}, \underline{I}_{scen.}, \underline{M}_{edt})$$



- State vector $\underline{s}_{edt}(t) = \begin{bmatrix} \underline{x}_{edt}(t) \\ \underline{a}_{edt}(t) \\ \underline{A}_{edt}(t) \\ \underline{c}_{edt}(t) \\ \underline{I}_{edt}(t) \end{bmatrix} = \begin{bmatrix} \underline{s}_{spsys}(t) \\ \underline{s}_{sips}(t) \\ \underline{s}_{shmi}(t) \end{bmatrix}$

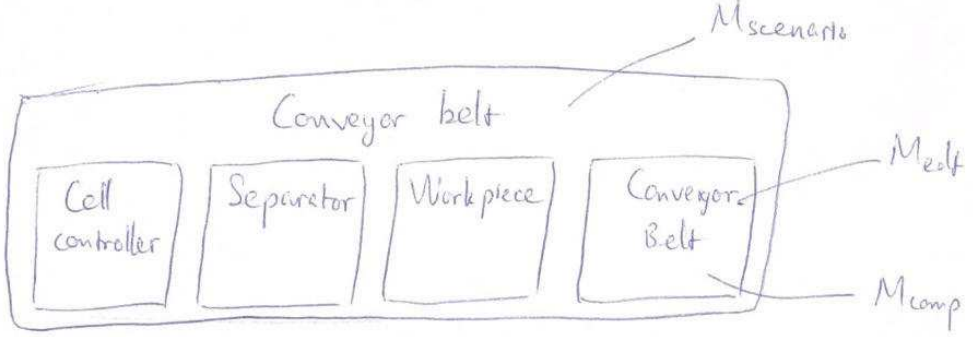
- Initial state $\underline{s}_{edt}(0)$
- Simulation $\underline{s}_{edt}(t) = \Gamma(\underline{s}_{edt}(0), \underline{u}_{edt}(t), t)$
- Output $\underline{y}_{edt}(t) = \phi(\underline{s}_{edt}(t), \underline{u}_{edt}(t), t)$

- each part with different simulation function

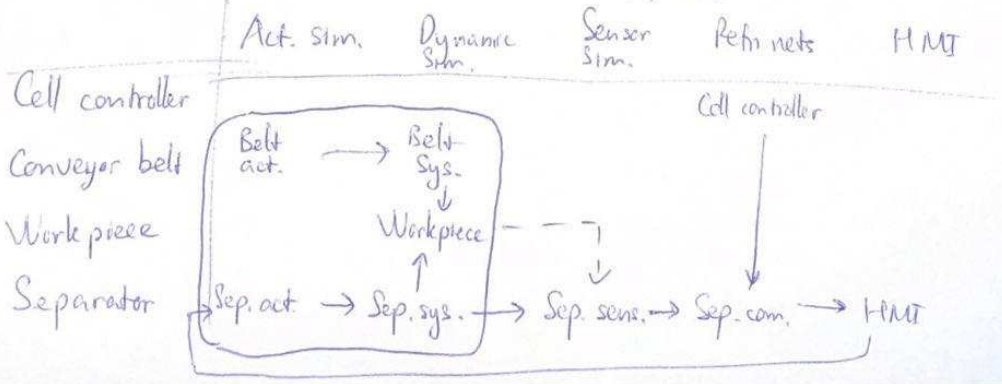
$$\underline{s}_i(t) = \Gamma_i(\underline{s}_i(0), \underline{u}_i(t), t)$$
$$\underline{y}_i(t) = \Phi_i(\underline{s}_i(t), \underline{u}_i(t), t)$$

- the input, output, state are connected / related

+) Ex:



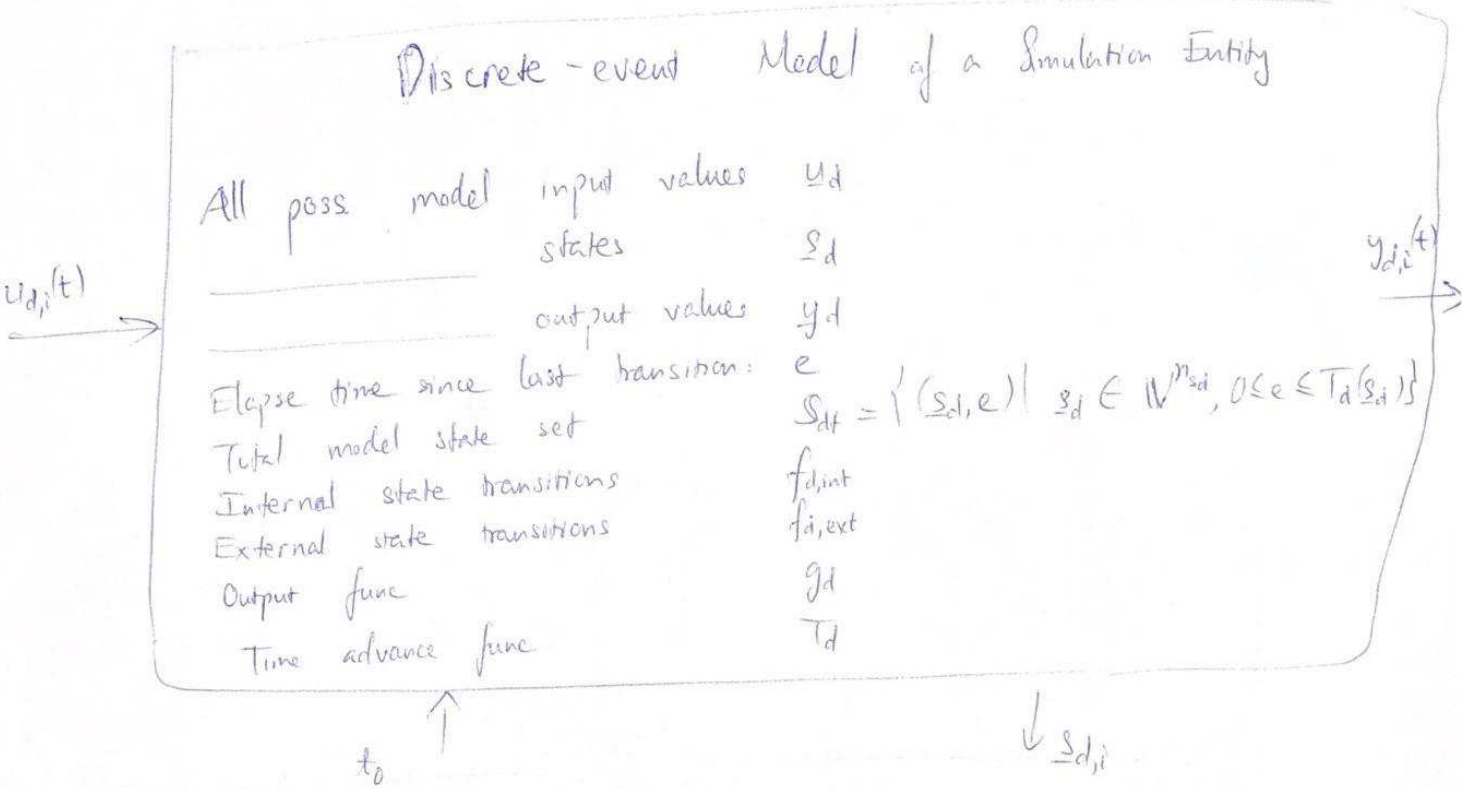
Simulation Func. Γ_i, Φ_i



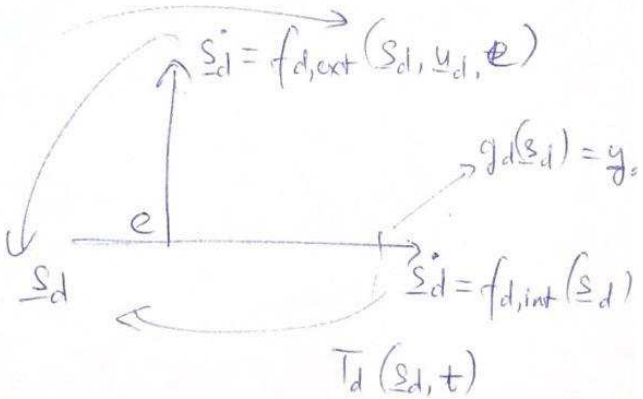
Discrete Event Systems

L6/

1, Formal Model Definition:



+/ When there is no external event $\underline{u}_d$, system stay at state $\underline{s}_d$ for time $T_d(\underline{s}_d, t) - t = \begin{cases} 0 & \text{transitory state} \\ \infty & \text{passive state} \end{cases}$



- "Passive" System

+) Modelling durations

- Use distribution func to model duration

ex: $T_d(S_d, t) = t + \text{normal}(10s, 2s)$

SRSP

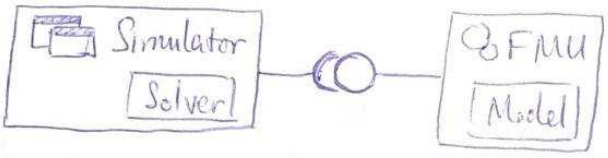
Functional Mockup Interface

1, FMI 2.0 Standard. FMI is a standard

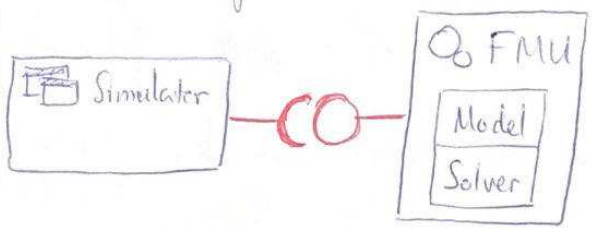
+, Motivation: to model & simulate large intergrated system
⇒ need to combine / coupling different simulators

⇒ 2 main approaches: export models from 1 simulator to another simulator
co-simulation of models in different Simulators

+, Basic Concepts:
- Each FMU is a Slave, Simulator is the Master
- FMI for Model Exchange



FMI for Co-Simulation

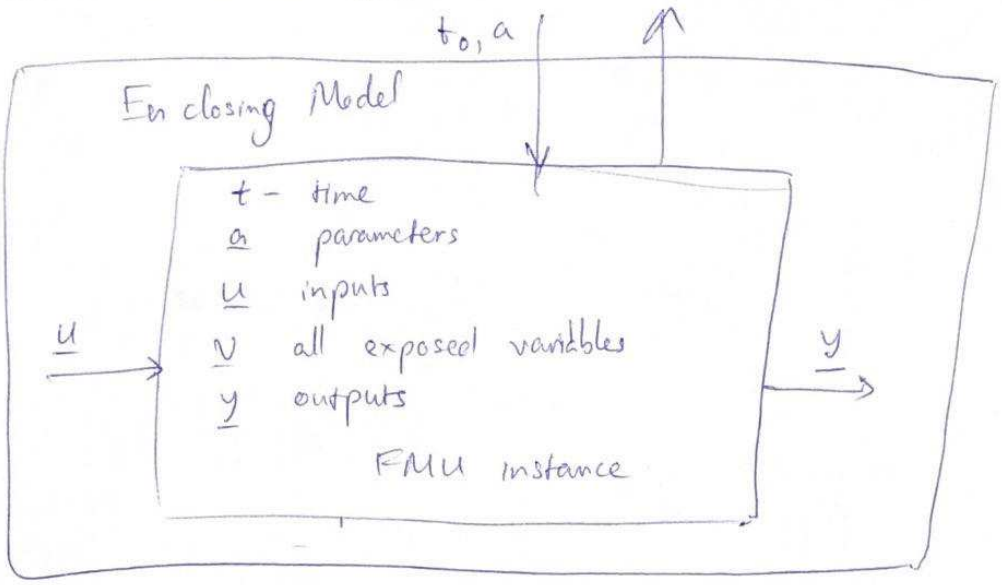


2, The Functional Mockup Unit (FMU) - the basic building block

- FMU implements a Simulation Entity
  x for state variables (not s)

+, FMU Structure
- Description of interface data: XML file of static model info
- Functionality: set of C-Functions .dll

+) Mathematical description:



⊛ Not only take in & process data
But also provide descriptions of inputs, outputs

⇒ Handling of algebraic loops (dependency information)
(description on input/output)

3) Model description: describe structure of an FMU

- FMI XML Schema

- +) Model variables:
  - name
  - valueReference
  - description
  - causality: "local", "output", "parameter", ... (what/how will it cause others effect ⇒)
  - variability: "constant", "fixed", "tunable", "discrete", "continuous" (time dependency)
  - initial: "exact", "approx", "calculated" (how it is initialized)
  - — (Super dense time option)
  - dataType IDa

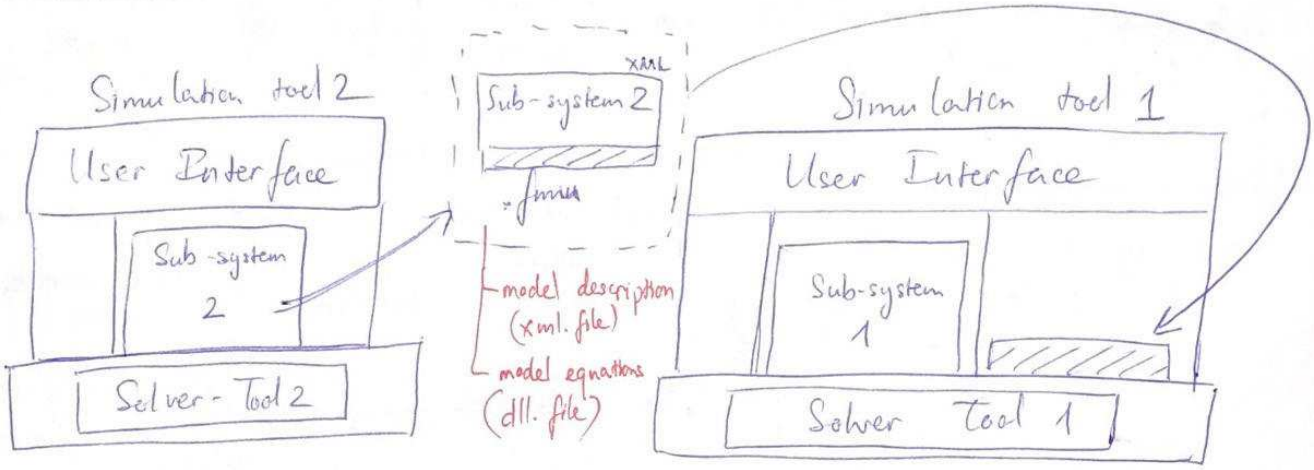
- +) Model structure (≤): define ordering of
  - outputs
  - (exposed) derivatives
  - Initial Unknowns

In each: define as: Unknown
- Index (of output)
- dependencies (index of input(s))
- dependenciesKind (between in- & output)

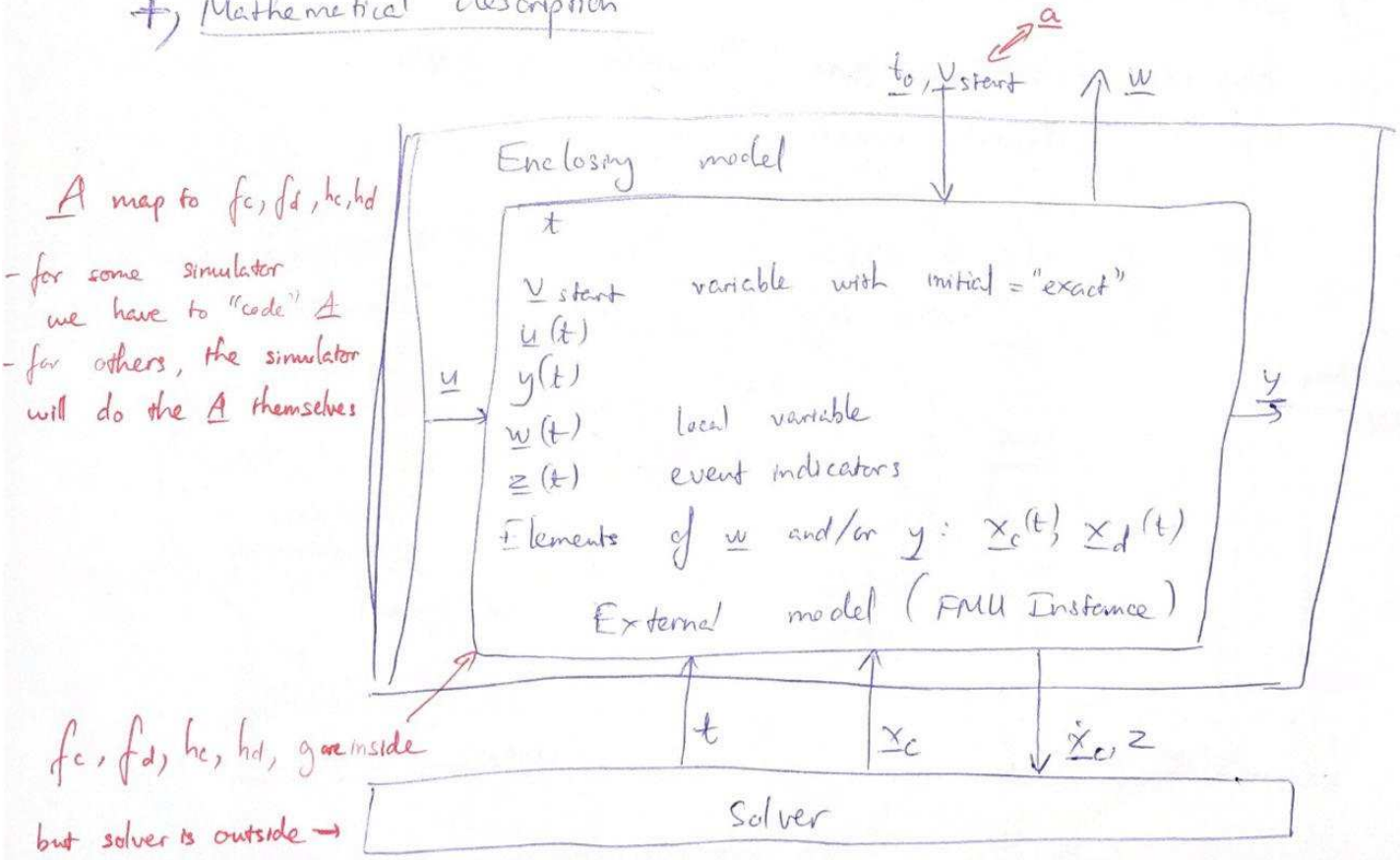
4) Implementation basics: The C Interface

5) FMI for Model-Exchange, Simulators with Integrators / "Master algorithm" for Model-exchange

Model Exchange FMU



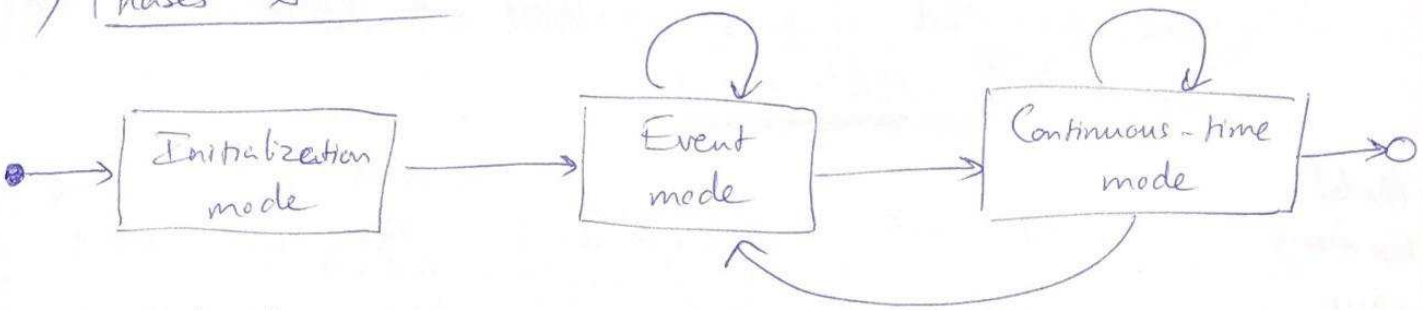
+) Mathematical description



+) Events

- Time step events: for integrator
- State events: when event indicator z changes
- (Discrete) Time events
- External events: when changes params values, effect { discrete time / cont

+) Phases vs modes.



Initial values of state, params, time $\underline{x}_c, \underline{x}_d, \underline{a}, t_0$

all / cont, time equa. are active | discrete

only cont. time equati. active | discrete variables remain fixed

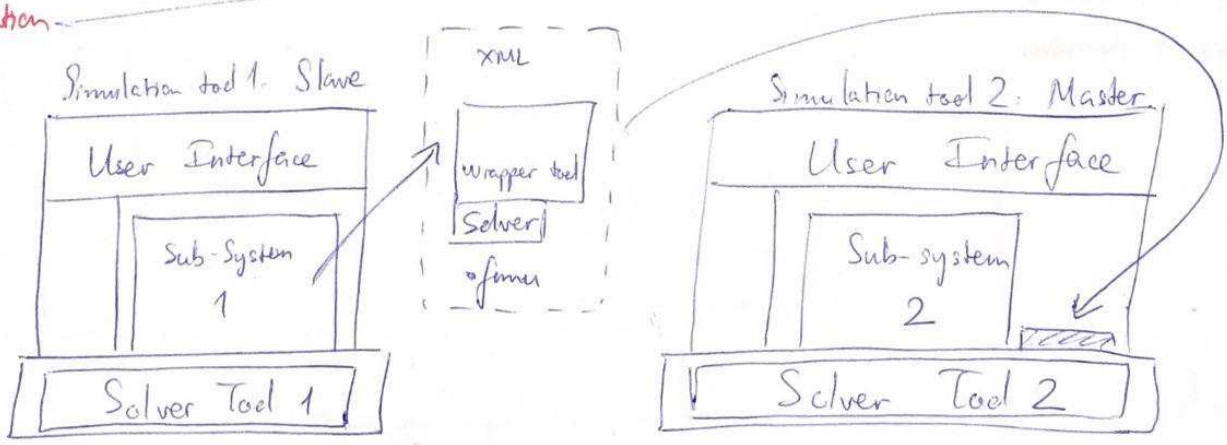
+) Master: FMU?

- Executing { continuous - time / discrete - event } Simulation Entities

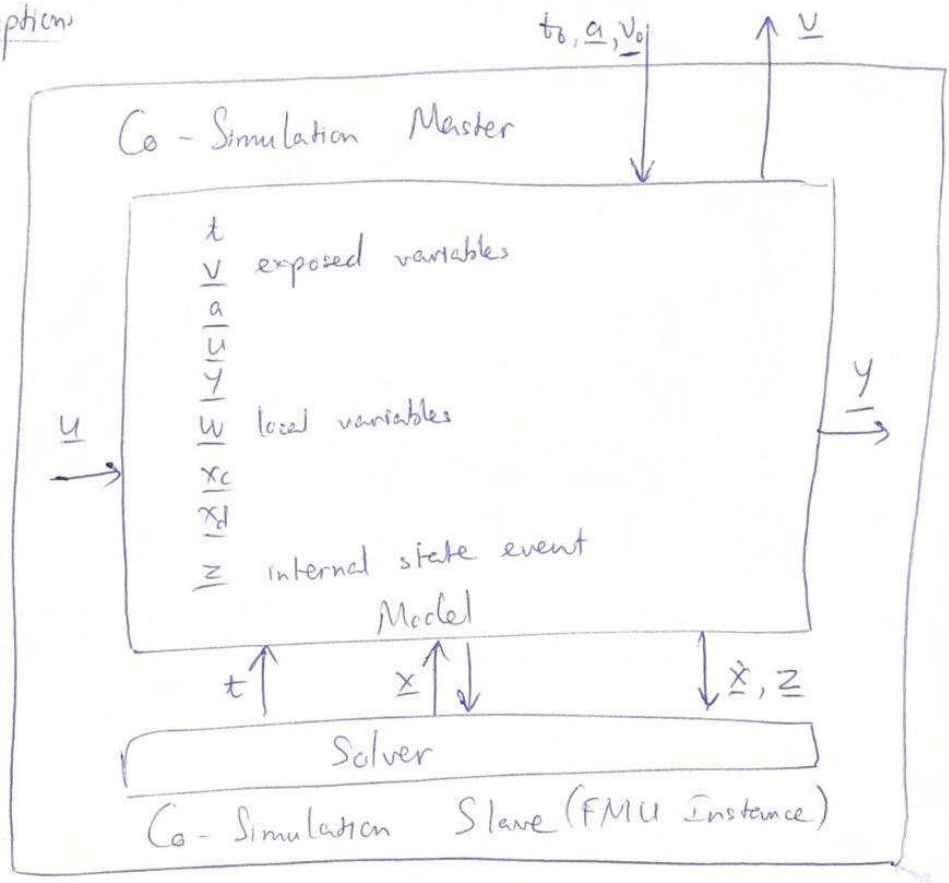
6) FMI for Co-Simulation

Simulators / "Master algorithm" for Co-Simulation

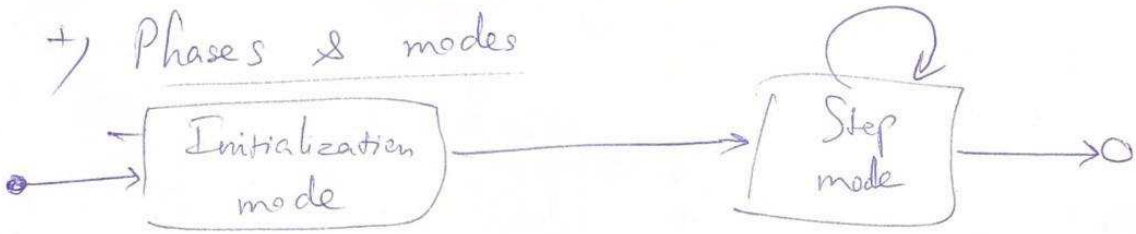
Co-Simulation FMU



+/ Mathematical description



+/ Phases & modes

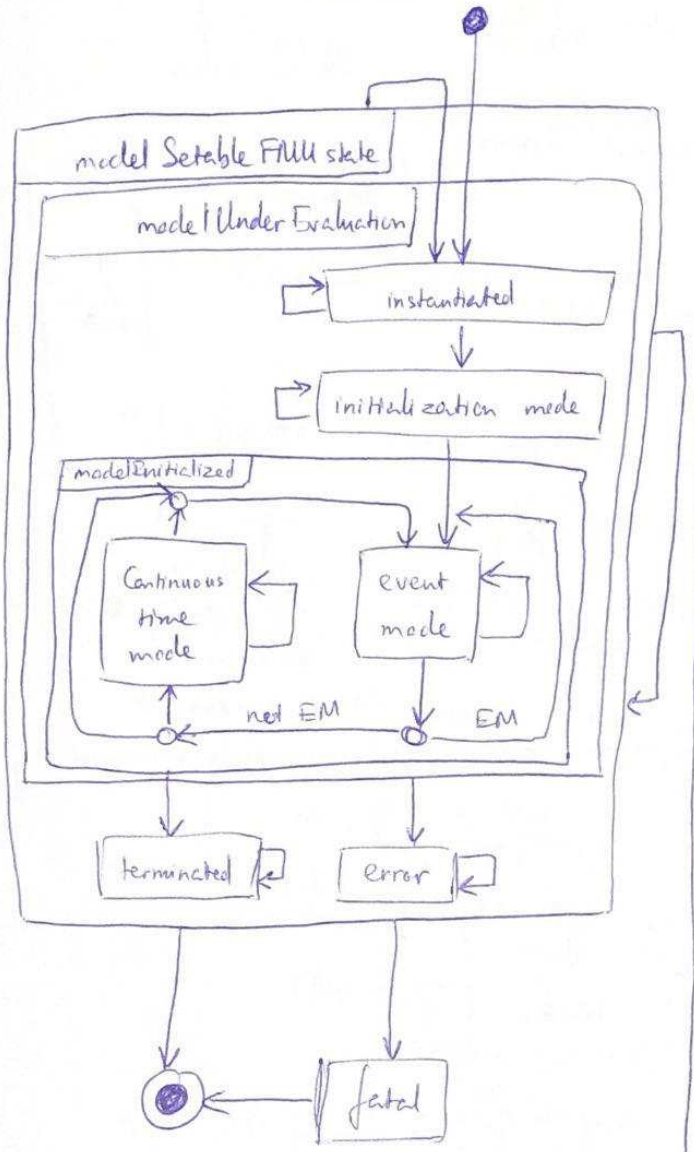


Initial state, params, time
$\underline{x}_c, \dot{\underline{x}}_d, \underline{a}, t_0$

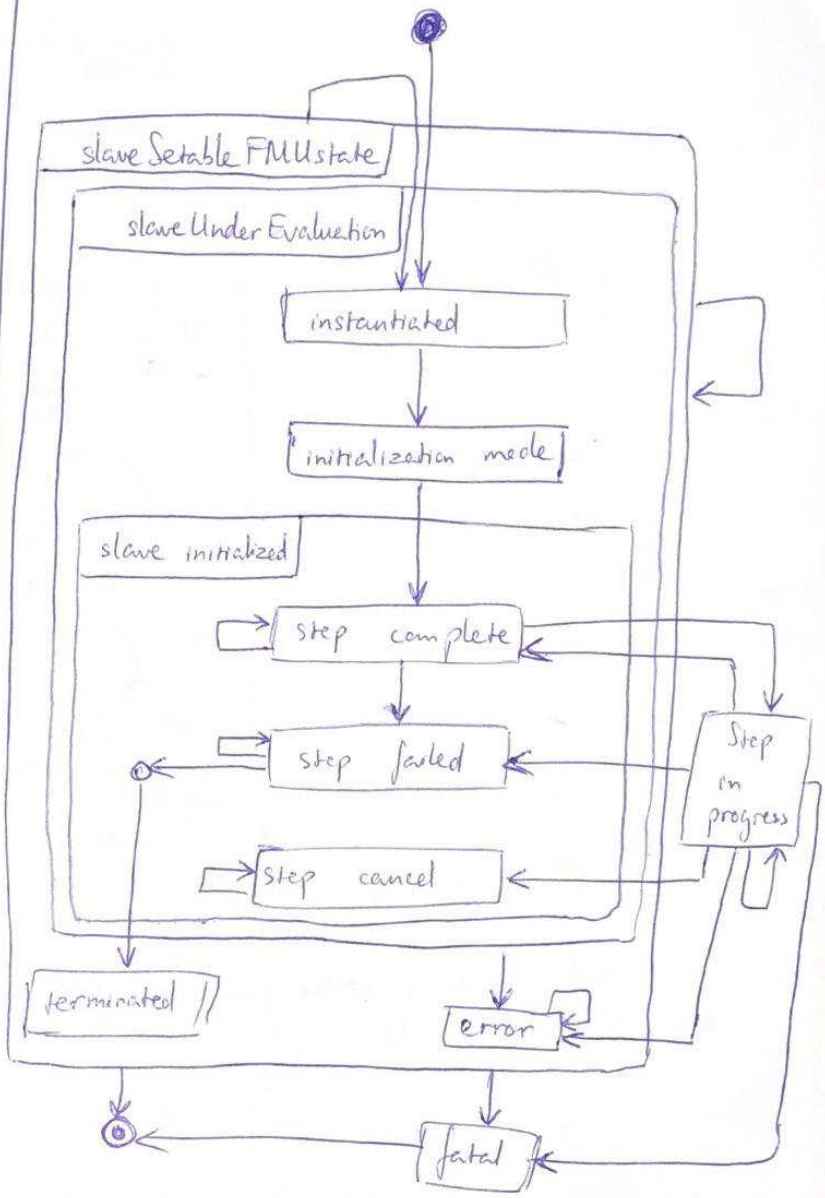
Compute | con. time var. $\underline{x}_c$
dis. $\underline{x}_d$
outputs
& their derivatives
(to copy to connected inputs)

⊗ Calling sequence

② Calling sequence:



Model Exchange



Co-Simulation C funcs

Equation-based / Signal-oriented Simulation

Matlab

L8,

1, Equation-based Simulation

+) Continuous-time Simulation Entity

$$\dot{\underline{s}}_c(t) = f_c\left(\underline{s}_c(t), \underline{u}_{entity}(t), t\right)$$
$$\underline{0} = h_c\left(\underline{s}_c(t), \dot{\underline{s}}_c(t), \underline{u}_{entity}(t), t\right)$$
$$\underline{y}_{entity} = g\left(\underline{s}_c(t), \underline{u}_{entity}(t), t\right)$$
$$t_{R,next}(t) = T\left(\underline{s}_c(t), \underline{u}_{entity}(t), t\right)$$

⇒ Need to intergrate $\underline{s}_c$

$\ddot{x} - \ddot{x}x + 1 = 0$

+) DE (Differential Equation)
ODE (Ordinary Differential Equation) : DE with 1 variable
DAEs (Differential Algebraic Equation)

+) MatLab
- function handles
- vectorization
- logicals

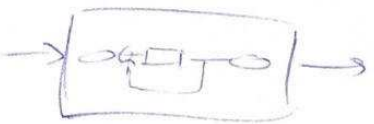
integrator ODE

2, Signal-oriented Simulation

+) Elements:
- Functional blocks
- Summing points / Branching points
- Action lines

+) Functional block

input, output, param, state



Sub system Block

+) Phases after "Run": Model-Compilation → Link phase → Init → Simulation Loop phase

3) Intergrating external simulators

Generic simulation language to model the entire system	Using as "Simulation Master"	Using a "Simulation Slave"
	FMU, S-Function	Matlab engine

Physical object-oriented Simulation

Modelica
(a language for multi-domain modelling)
Not a TOOL

29,

1, Introduction:

- Causality in Functional Diagram as arrow

2, Bond graph { nodes
bonds { effort e Power P = e·f
flow f

3, Physical object-oriented simulation with Modelica

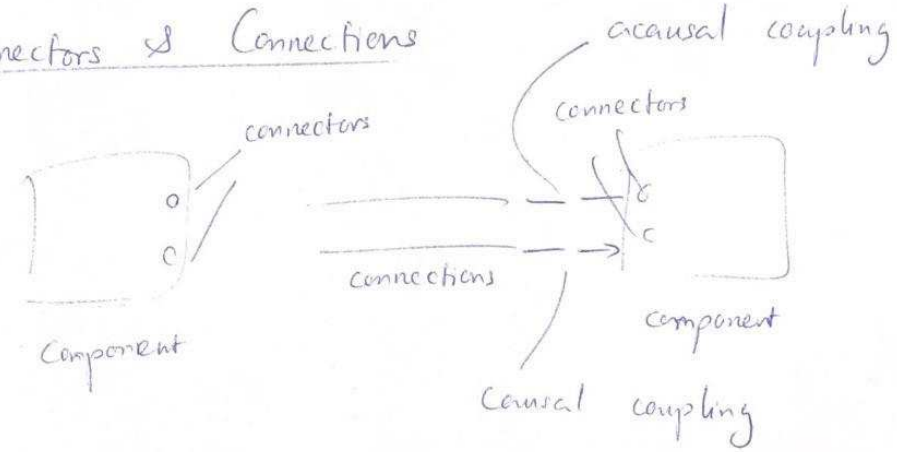
- Modelica is a multi-domain modelling language
of complex Cyber-Physical Systems
- Multi-domain: electrical, mechanical, hydraulic..
- Order of computations is not decided at modelling time (acausal)
≠ causal

4, Modelica Language concept

5) Hybrid modeling: continuous time + discrete-event modelig

- Event creation with if
  when

6) Components, Connectors & Connections



+) Two kinds of variables in connectors

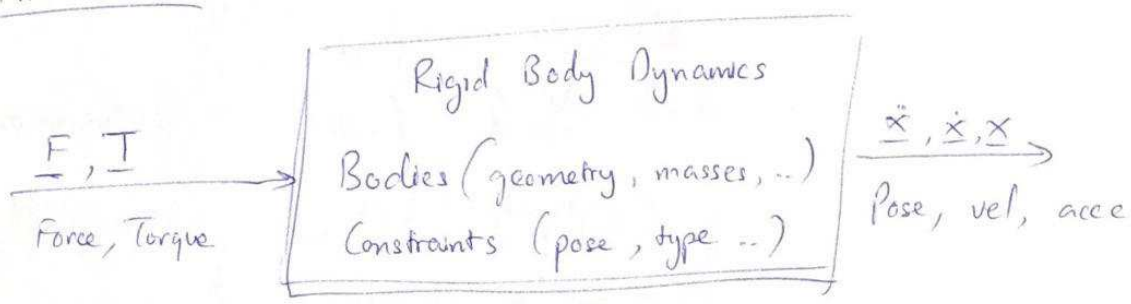
	coupling
- Non flow variables ⇒ potential or energy level	equality
- Flow variables ⇒ represent some kind of flow	sum-to-zero

Ex:		non flow	flow
	electrical	voltage	current

Rigid Body Dynamics Simulation

L10/

1) Introduction



+) Based on Simulation Models.
- describe the properties of each individual Rigid Body
  kinematic coupling between Rigid Bodies
- can contain additional actors / sensors

→) Can be integrated into a Simulation Entity

2) Overview:

+) Generalized coordinates vs Maximal Coordinates

Joint angles / vel / acc

Bodies pose / orient / vel..
(Cartesian coord)

can handle fixed/simple situation only as individual solution

when # constraints change
⇒ need comprehensive approach to handle parallel kinematics chains

q_1 $\dot{q}_1$ $\ddot{q}_1$
q_2 $\dot{q}_2$ $\ddot{q}_2$
q_n $\dot{q}_n$ $\ddot{q}_n$

x_1 x_2 .. $\dot{x}_1$ .. $\ddot{x}_1$
y_1 y_2 $\dot{y}_1$ $\ddot{y}_1$
z_1 z_2 $\dot{z}_1$ $\ddot{z}_1$

$$\frac{d}{dt}\begin{cases} p_i = m_i \cdot v_i \\ L_i = \Theta_i \cdot \omega_i \end{cases} \Rightarrow \begin{cases} \frac{dp_i}{dt} = f_{ext,i} = m_i \cdot \dot{v}_i \\ \frac{dL_i}{dt} = \tau_{ext,i} = \Theta_i \cdot \dot{\omega}_i + \omega_i \times \Theta_i \omega_i \end{cases} \Leftrightarrow \begin{cases} \dot{v}_i = \frac{1}{m_i} f_{ext,i} \\ \dot{\omega}_i = \Theta_i^{-1}(..) \end{cases}$$

$$\Rightarrow \begin{pmatrix} \dot{v}_1 \\ \dot{\omega}_1 \\ \vdots \\ \dot{v}_n \\ \dot{\omega}_n \end{pmatrix} = \begin{pmatrix} m_1^{-1} I & & & & \\ & \Theta_1^{-1} & & & \\ & & \ddots & & \\ & & & m_n^{-1} I & \\ & & & & \Theta_n^{-1} \end{pmatrix} \cdot \begin{pmatrix} f_{ext,1} \\ \vdots \\ f_{ext,n} \\ \vdots \end{pmatrix}$$

$$\ddot{x} = M^{-1} \cdot f_{ext}$$

$$\Rightarrow \dot{x}(t+dt) = \dot{x}(t) + dt \cdot M^{-1}(t) \cdot f_{ext}(t) \qquad \text{Integration}$$

+) Constraints: $C(x(t), t) = 0 \Leftrightarrow \frac{d}{dt} C(x(t), t) = 0$

$$\Leftrightarrow \frac{\partial C}{\partial x} \cdot \dot{x} + \frac{\partial C}{\partial t} = 0 \Leftrightarrow J \cdot \dot{x} + \underbrace{\frac{\partial C}{\partial t}}_{=0} = 0$$

Constraint Jacobian $J = \frac{\partial C}{\partial x}$

+) Principle of D'Alembert: Constraint forces f_c must apply no power

$$f_c = J^T \cdot \lambda \quad ; \quad P_c = f_c^T \cdot \dot{x} = (J^T \cdot \lambda)^T \cdot \dot{x} = \lambda^T \cdot \underbrace{J \cdot \dot{x}}_{=0} = 0$$

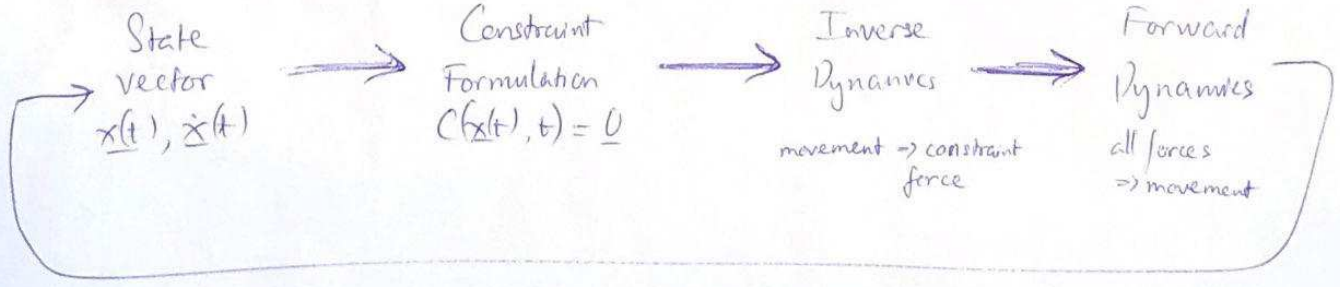
λ: magnitude of constraint forces; J^T: direction of the constraints

+) Inverse Dynamics: calculate constraint forces from current state / movements

$$\begin{cases} J \cdot \dot{x}(t+dt) = b \\ \dot{x}(t+dt) = \dot{x}(t) + dt \cdot M^{-1}(t)(f_{ext} + J^T \cdot \lambda) \end{cases}$$

$$\Rightarrow J \cdot M^{-1} \cdot J^T dt \cdot \lambda + J(\dot{x} + M^{-1} dt\, f_{ext}) - b = 0$$

+) Simulation loop of RBDS:



3) Properties & Kinematics of Rigid Bodies

+) **Rigid Body:** a solid body, deformation is 0

- Centre of gravity: $\underline{p}_{cog} = \frac{1}{m} \iiint_E \rho(\underline{p}) \cdot \underline{p} \, dV$

- Inertia tensors: symmetric ${}^{W}\underline{\Theta}_{cog} = ({}^{W}\underline{\Theta}_{cog})^T$

  positive definite

  To change ref frame: ${}^{W_2}\underline{\Theta}_{cog} = {}^{W_2}\underline{R}_{W_1} \cdot {}^{W_1}\underline{\Theta}_{cog} \cdot ({}^{W_2}\underline{R}_{W_1})^T$

+) **Kinematics of Rigid Bodies**

- Homogeneous transformation: ${}^{W}\underline{T}_{B} = \begin{bmatrix} {}^{W}\underline{R}_{B} & {}^{W}\underline{p}_{cog} \\ 0 & 1 \end{bmatrix}_{4\times4}$

⇒ - Alternative: quaternion $\begin{bmatrix} {}^{W}\underline{p}_{cog} & {}^{W}\underline{q}_{B} \end{bmatrix}^T$ (${}^{W}\underline{q}_{B}$ is the quaternion)

- Cross-product matrix: $\underline{\omega} \times \underline{v} = \underline{\Omega}_{\times} \cdot \underline{v}$

- Velocities of a body fixed point: $\underline{v}_p = \underline{v}_{cog} + \underline{\omega} \times \underline{r}$

  $\underline{\omega}_p = \underline{\omega}_{cog} = \underline{\omega}$

⇒ Accelerations ——— $\dot{\underline{v}}_p = \dot{\underline{v}}_{cog} + \underline{\omega} \times (\underline{\omega} \times \underline{r}) + \dot{\underline{\omega}} \times \underline{r}$

$\dot{\underline{\omega}}_p = \dot{\underline{\omega}}_{cog} = \dot{\underline{\omega}}$

${}^{W}\underline{f} = {}^{W}\underline{p}_{cog} + {}^{W}\underline{R}_{B} \cdot {}^{B}\underline{f} \Rightarrow {}^{W}\underline{v}_p = {}^{W}\underline{v}_{cog} + \underline{\omega} \times ({}^{W}\underline{R}_{B} \cdot {}^{B}\underline{f}) + {}^{W}\underline{R}_{B} \cdot {}^{B}\underline{v}_p$

$\Rightarrow {}^{W}\dot{\underline{v}}_p = {}^{W}\dot{\underline{v}}_{cog} + \dot{\underline{\omega}} \times ({}^{W}\underline{R}_{B} \cdot {}^{B}\underline{f}) + \underline{\omega} \times (\underline{\omega} \times ({}^{W}\underline{R}_{B} \cdot {}^{B}\underline{f})) + 2\underline{\omega} \times ({}^{W}\underline{R}_{B} \cdot {}^{B}\underline{v}_p) + {}^{W}\underline{R}_{B} \cdot {}^{B}\dot{\underline{v}}_p$

4) Simulation of free-floating Rigid Bodies:

- Principle of momentum conservation:

  The momentum of RB is constant if no external forces are applied

$$\begin{cases} \underline{p} = m \cdot \underline{v} \\ \underline{L} = \underline{\Theta} \cdot \underline{\omega} \end{cases} \Rightarrow \begin{cases} \sum_i \underline{p}_i = \text{const} & \text{Linear} \\ \sum_i \underline{L}_i = \text{const} & \text{Angular} \end{cases} \Big\} \text{momentum}$$

→ Derivative of momentum:

$$f_{ext} = \dot{p} = m \cdot \dot{v}$$

$$\tau_{ext} = \dot{L} = \Theta \cdot \dot{\omega} + \underbrace{\omega \times \Theta \cdot \omega}_{\text{Gyroscopic torque}}$$

⇒ → Dynamic equations:

$$\begin{bmatrix} \dot{v}_1 \\ \dot{\omega}_1 \\ \vdots \\ \dot{v}_n \\ \dot{\omega}_n \end{bmatrix} = \begin{bmatrix} M_1^{-1} & & \\ & {}^{W}\Theta_1^{-1} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} f_{ext,1} \\ \tau_{ext,1} - \omega_1 \times {}^{W}\Theta_1 \omega_1 \\ \vdots \end{bmatrix}$$

→ $$\begin{cases} v(t+dt) = v(t) + dt \cdot M^{-1} f_{ext} \\ p(t+dt) = p(t) + dt \cdot v(t+dt) \end{cases} \quad \begin{bmatrix} p_i(t+dt) \\ q_i(t+dt) \end{bmatrix} = \begin{bmatrix} p_i(t) \\ q_i(t) \end{bmatrix} + dt \begin{bmatrix} v_i(t+dt) \\ \frac{1}{2} \begin{bmatrix} 0 \\ \omega_i(t+dt) \end{bmatrix} q_i \end{bmatrix}$$

→ Explicit integration

Implicit ——

Semi-implicit integration:

→ Simulation loop:

- Current system state: $p_i(t)$ $q_i(t)$ $v_i(t)$ $\omega_i(t)$
- Calculation of derivative of system state: $v(t+dt) = \ldots \rightarrow p(t+dt)$
- Normalization of quaternions: $q_i(t+dt) = \frac{q_i(t+dt)}{|q_i(t+dt)|}$
- Re calculation of inertia tensor: ${}^{W}\Theta(t+dt) = {}^{W}R_{B_i}(t+dt)\, {}^{B_i}\Theta \cdot (\ldots)^T$

5) Simulation of constrained Rigid Bodies
Generalized vs maximal coordinates formalism

→ Generalized coordinates: $q_1, q_2, \ldots, q_n$

- Lagrange equation $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \left(\frac{\partial L}{\partial q}\right) = Q$ with $L = E_{kin} - E_{pot}$

⇒ Equation of motion $H(q)\,\ddot{q} + C(q, \dot{q}) = \tau$

- Forward dynamics and inverse dynamics

$$\ddot{q} = FD(q, \dot{q}, \tau) \qquad \tau = ID(q, \dot{q}, \ddot{q})$$

+) Maximal coordinates: describe each bodies individually

$$m \cdot \ddot{\underline{x}} = \sum \underline{f}_i \quad \Rightarrow \quad \ddot{\underline{x}} = f(\underline{x}, \dot{\underline{x}}, \underline{f}_{ext})$$

6) Constraints: restrict dof of system..

- Holonomic constraints vs Non-holonomic constraints

describe by $\underline{p}_i, \underline{q}_i$

for joints..

$$C(\underline{x}(t), t) = 0$$

collisions

$$C(\underline{x}(t), t) \geq 0$$

+) Constraints Jacobi matrix

$$\frac{d}{dt} C(\underline{x}(t), t) = \frac{\partial C}{\partial \underline{x}} \cdot \dot{\underline{x}} + \frac{\partial C}{\partial t} = \underline{J} \cdot \dot{\underline{x}} + \frac{\partial C}{\partial t} = \underline{0}$$

$$\dot{\underline{x}} = [\underline{v}_1, \underline{w}_1, \ldots, \underline{v}_n, \underline{\omega}_n]^T$$

$$\frac{d^2}{dt^2} C(\underline{x}(t), t) = \underline{J} \cdot \ddot{\underline{x}} + \frac{\partial^2 C}{\partial x^2} \cdot \dot{x}^2 + 2 \frac{\partial^2 C}{\partial i \partial x} \cdot \dot{x} + \frac{\partial^2 C}{\partial t^2}$$

7) Simulation of constrained Rigid Bodies – Constraint forces & simulation

+) $\underline{M}\ddot{\underline{x}} - \underline{J}^T \underline{\lambda} - \underline{f}_{ext} = 0$

$\underline{J}\ddot{\underline{x}} - \underline{b} = 0$

$$\Rightarrow \begin{bmatrix} \underline{M} & -\underline{J}^T \\ \underline{J} & \underline{0} \end{bmatrix} \cdot \begin{bmatrix} \ddot{\underline{x}} \\ \underline{\lambda} \end{bmatrix} - \begin{bmatrix} \underline{f}_{ext} \\ \underline{b} \end{bmatrix} = \underline{0}$$

+) JMJT Approach: $\underline{J}\underline{M}^{-1}\underline{J}^T\underline{\lambda} + \underline{J}\underline{M}^{-1}\underline{f}_{ext} - \underline{b} = 0$

- Much smaller equation system
  ⇒ Ccan be calculated much more efficient
- Forward dynamics is calculated independently

+) Using Maximal Coordinates, we will have numerical drifts, which part over time ⇒ need stabilization techniques

⇒ 8) Stabilization of constrained Rigid Body Dynamics

+) Initially, the position-based constraints are included in the algorithm at acceleration level: $\underline{J}\,\ddot{\underline{x}} - \underline{b} = 0$

⇒ Now, we add both velocity- and position-based level:

⇒ Baumgarte stabilization:

$$\ddot{\underline{C}}(\underline{x},t) + \alpha \cdot \dot{\underline{C}}(\underline{x},t) + \beta \cdot \underline{C}(\underline{x},t) = \underline{0}$$

$$\Rightarrow \underline{J}\,\ddot{\underline{x}} - \underbrace{\underline{b} + \alpha\,\underline{J}\,\dot{\underline{x}} + \beta\,\underline{\gamma}_{rel}}_{\hat{\underline{b}}} = \underline{0}$$

$$\Rightarrow \underline{J}\,\ddot{\underline{x}} - \hat{\underline{b}} = \underline{0} \quad \Rightarrow \text{ no algorithmic complexity}$$

+) Same for JMJT approach:

$$\underline{J} \cdot \underline{M} \cdot \underline{J}^T \cdot \underline{\lambda} + \underline{J} \cdot \underline{M}^{-1} \cdot \underline{f}_{ext} - \hat{\underline{b}} = \underline{0}$$

9) Impulse-based simulation of constrained Rigid Bodies

⊛ Change everything from acceleration-level to velocity-level

+) Equation system approach

$$\begin{cases} \underline{M} \cdot \dfrac{\underline{v}(t+dt) - \underline{v}(t)}{dt} = \underline{f}_{ext} + \underline{J}^T \cdot \underline{\lambda} \\ \underline{J} \cdot \underline{v}(t+dt) = \underline{0} \end{cases} \qquad \text{(instead of } \underline{J}\,\ddot{\underline{x}} - \underline{b} = \underline{0}\text{)}$$

$$\begin{bmatrix} \underline{M} & -\underline{J}^T \\ \underline{J} & \underline{0} \end{bmatrix} \cdot \begin{bmatrix} \underline{v}(t+dt) \\ dt \cdot \underline{\lambda} \end{bmatrix} - \begin{bmatrix} \underline{M} \cdot \underline{v}(t) + dt \cdot \underline{f}_{ext} \\ \underline{0} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{0} \end{bmatrix}$$

$dt \cdot \underline{\lambda}$: constraint momentum (instead of forces before)

could also add Baumgarte stabilization here

+) JMJT approach:

$$\underline{M} \cdot \frac{\underline{v}(t+dt) - \underline{v}(t)}{dt} = \underline{f}_{ext} + \underline{J}^T \cdot \underline{\lambda}$$

$$\Rightarrow \underline{v}(t+dt) = \underline{v}(t) + \underline{M}^{-1} \cdot dt \left(\underline{f}_{ext} + \underline{J}^T \underline{\lambda} \right)$$

and with $\underline{J} \cdot \underline{v}(t+dt) = \underline{0}$

$$\Rightarrow \underline{J} \cdot \underline{M} \cdot \underline{J}^T \cdot dt \cdot \underline{\lambda} + \underline{J} \left(\underline{v}(t) + \underline{M}^{-1} \cdot dt \cdot \underline{f}_{ext} \right) = \underline{0}$$

+) Advantages:

- can use velocity-based constraints
- larger integration steps possible
- easier integration of friction
- no ambiguities if coulomb friction is modelled

10) Simulation of constrained Rigid Bodies with contacts

(A) Adding non-holonomic constraints $C(\underline{x}(t), t) \geq \underline{0}$

+ Equation system approach:

$$\begin{bmatrix} \underline{M} & -\underline{J}_e^T & -\underline{J}_n^T \\ \underline{J}_e & 0 & 0 \\ \underline{J}_n & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{v}(t+dt) \\ dt \cdot \underline{\lambda}_e \\ dt \cdot \underline{\lambda}_n \end{bmatrix} - \begin{bmatrix} \underline{M} \cdot \underline{v}(t) + dt \, \underline{f}_{ext} \\ \underline{0} \\ \underline{0} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{0} \\ \underline{a}_n \end{bmatrix}$$

non-holonomic constraints

$\underline{a}_n \geq \underline{0}$ slack variables

$$\underline{a}_n \geq \underline{0} \;, \quad \underline{\lambda}_n \geq \underline{0}, \quad \underline{a}_n^T \underline{\lambda}_n = \underline{0}$$

$\Rightarrow$ Mixed Linear Complementarity Problem

- Case 1: Bodies are seperating $\underline{a}_n \geq \underline{0}$, $\underline{\lambda}_n = \underline{0}$
- Case 2: Bodies are resting on others $\underline{a}_n = \underline{0}$, $\underline{\lambda}_n \geq \underline{0}$

  $\underline{a}_n$ relates to distance between bodies / or velocities

  $\underline{\lambda}_n$ ——— the forces (as it always has)

+ JMJT Approach

$$\underline{v}(t+dt) = \underline{v}(t) + dt \cdot \underline{M}^{-1} \left[\begin{pmatrix} \underline{J}_e \\ \underline{J}_n \end{pmatrix}^T \cdot \begin{pmatrix} \underline{\lambda}_e \\ \underline{\lambda}_n \end{pmatrix} + \underline{f}_{ext} \right] \quad \text{and} \quad \begin{bmatrix} \underline{J}_e \\ \underline{J}_n \end{bmatrix} \cdot \underline{v}(t+dt) = \begin{bmatrix} \underline{0} \\ \underline{a}_n \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \underline{J}_e \underline{M}^{-1} \underline{J}_e^T & \underline{J}_e \underline{M}^{-1} \underline{J}_n^T \\ \underline{J}_n \underline{M}^{-1} \underline{J}_e^T & \underline{J}_n \underline{M}^{-1} \underline{J}_n^T \end{bmatrix} \cdot \begin{bmatrix} dt \, \underline{\lambda}_e \\ dt \, \underline{\lambda}_n \end{bmatrix} + \begin{bmatrix} \underline{J}_e \\ \underline{J}_n \end{bmatrix} \left(\underline{v}(t) + \underline{M}^{-1} dt \, \underline{f}_{ext} \right) = \begin{bmatrix} 0 \\ \underline{a}_n \end{bmatrix}$$

With the same 2 cases as above

⊛ How to solve a Mixed Linear Complementarity Problem

+) Projected - Gauss - Seidel algorithm: for JMJT approach

$$\begin{bmatrix} \underline{A} & \underline{C} \\ \underline{D} & \underline{B} \end{bmatrix} \begin{bmatrix} \underline{\lambda}_e \\ \underline{\lambda}_n \end{bmatrix} = \begin{bmatrix} \underline{b}_e \\ \underline{b}_n \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{a}_n \end{bmatrix}$$

- Splitting in upper and lower triangular matrices

$$\underline{A} = \underline{L}_A + \underline{U}_A \quad , \quad \underline{B} = \underline{L}_B + \underline{U}_B$$

- Then derive:

$$\underline{\lambda}_e = \underline{L}_A^{-1} (\underline{b}_e + \underline{C}\,\underline{\lambda}_n - \underline{U}_A \underline{\lambda}_e)$$

$$\underline{\lambda}_n = \underline{L}_B^{-1} (\underline{b}_n - \underline{D}\,\underline{\lambda}_e - \underline{U}_B \underline{\lambda}_n)$$

⇒ A fixed point formulation for iterative calculation (we neglect a_n for now)

$$\Longrightarrow \begin{cases} \underline{\lambda}_e^{k+1} = \underline{L}_A^{-1} (\underline{b}_e \overset{?}{+} \underline{C}\,\underline{\lambda}_n^k - \underline{U}_A \underline{\lambda}_e^k) \\ \underline{\lambda}_n^{k+1} = \underline{L}_B^{-1} (\underline{b}_n - \underline{D}\,\underline{\lambda}_e^k - \underline{U}_B \underline{\lambda}_n^k) \end{cases}$$

+) Algorithm:

- Determine $\underline{A}, \underline{B}, \underline{C}, \underline{D}$
- ——— $\underline{L}_A, \underline{U}_A, \underline{L}_B, \underline{U}_B$
- Calculate $\underline{L}_A^{-1}, \underline{L}_B^{-1}$
- Guess initial $\underline{\lambda}_0$
- $\underline{\lambda}^k = \underline{\lambda}_0$
- error = ∞
- while error > errorLimit do — or also a maximal no of iter

$$\underline{\lambda}_e^{k+1} = \underline{L}_A^{-1} (\underline{b}_e + \underline{C}\,\underline{\lambda}_n^k - \underline{U}_A \underline{\lambda}_e^k)$$

$$\underline{\lambda}_n^{k+1} = \underline{L}_B^{-1} (\underline{b}_n - \underline{D}\,\underline{\lambda}_e^k - \underline{U}_B \underline{\lambda}_n^k)$$

$$\underline{\lambda}_n^{k+1} = \max (\underline{0}, \underline{\lambda}_n^{k+1})$$

$$\text{error} = | \underline{\lambda}^{k+1} - \underline{\lambda}^k |$$

$$\underline{\lambda}_k = \underline{\lambda}_{k+1}$$

end

+) Gauss - Seidel

+ Easy
+ Numerically Robust

- #No. of iteration depend on initial guess

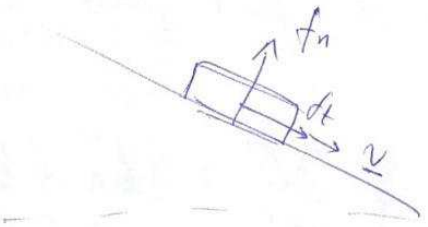
Dantzig Solver

+ Guaranteed solution in finite #no of steps

- Difficult
- Lexicography ordering required in order to avoid infinite loops

II) Simulation of constrained Rigid Bodies with friction

+) $f_t \leq \mu \cdot f_n$



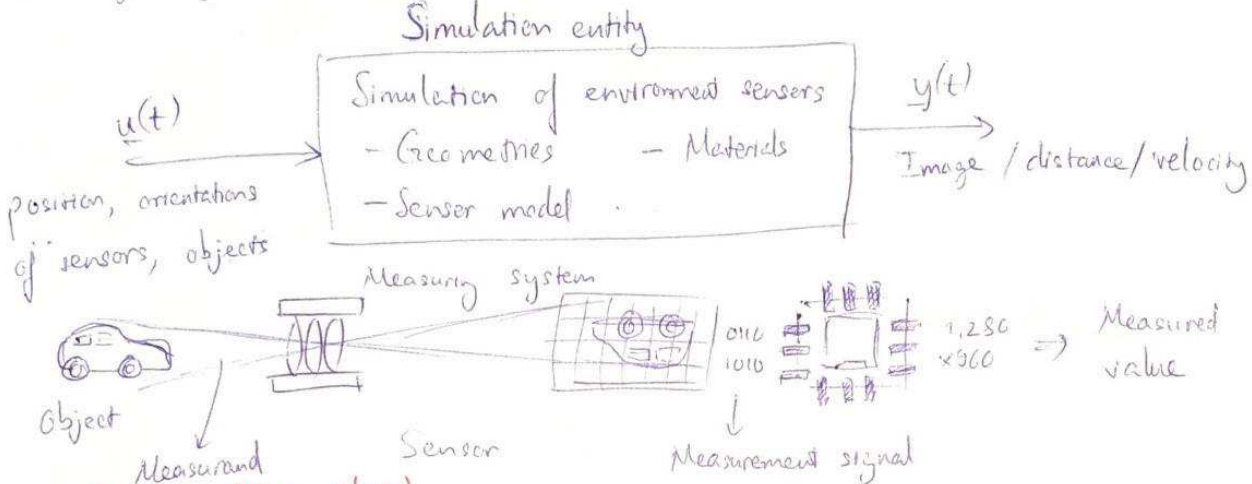
SRSP

Environment Sensors

41

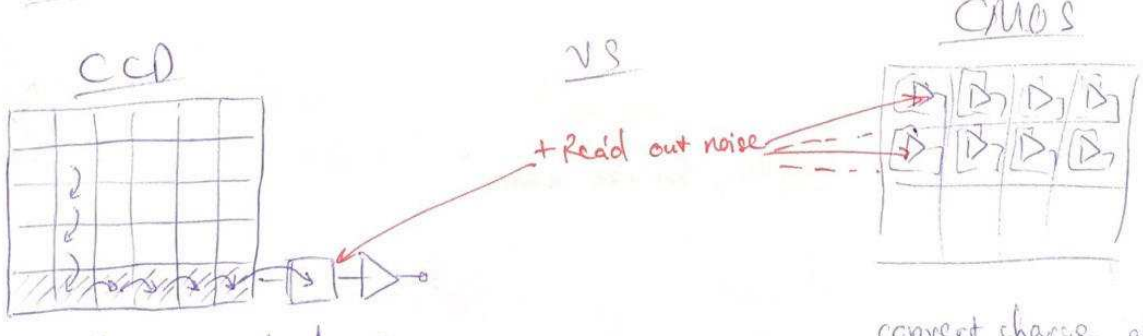
1, Introduction:

- Scenarios which it is difficult to get reference data (space, underwater..)
- Annotated data for ML (Autonomous Driving, RL..)
- Dangerous environment (Radiation, hazardous ..)



- Simulation Models describe | properties of each individual sensor & its measuring chain / measuring objects / movement of sensors, objects
- Intergrate Sensor Simulation algorithm into a Simulation Entity

2, Recap: Sensors technologies and fundamental measuring principles for environment sensors:



move photogenerated charge from pixel to pixel

Smear effect

convert charge at each pixel

most common

In some LIDAR as well

3, Rendering

Functional Requirements

- Camera

	Camera	LIDAR
Intrinsic	Sensor dim. S_x, S_y pixel count w, h focal length $f \cdot f_x, f_y$ principle point c_x, c_y distortion params: aperture size: a_x a_y	Beam divergence Θ Beam power P_L Pulse width τ wave length λ Velocity v
	+ Noise	
Extrinsic	${}^{WS}T_{ES}$	${}^{WS}T_{ES}$
Output	- Intensity (W/m^2) - Distance, velocity	Intensity Distance + Velocity

- Modelling Scenario: all relevant properties of corresponding real scenario

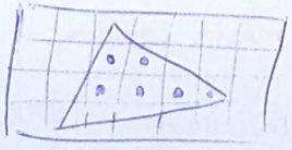
Non-Functional Requirements

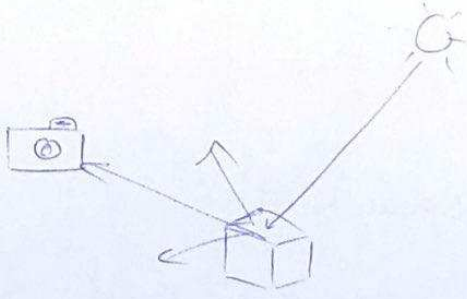
- Fast Simulation
- Real time (or faster)
- X-in-the-loop
- Simulation & visualization of internal sensor data
- Multiplatform - support

+, Render: Generate sensor output

- Classification:
  - Rasterization
    - Direct Illumination (real time)
    - Radiosity (offline, but more realistic)
  - Ray Tracing
    - Forward ray Tracing (emit too many rays)
    - Backward ray Tracing (almost always use this)

Vector graphics format → 2D raster image





Rasterization

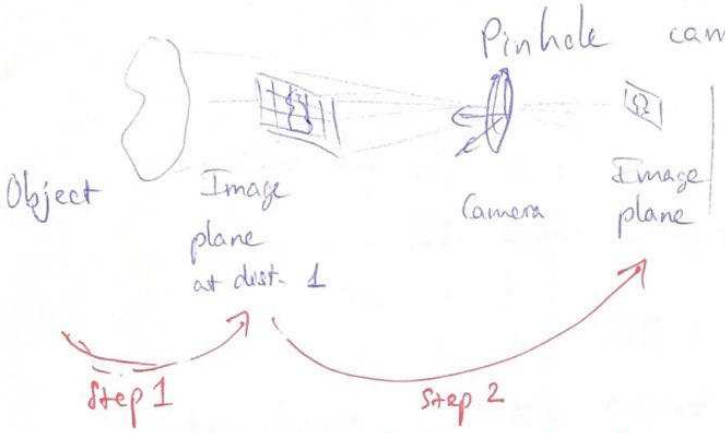
⊕
- Well-suited for current GPU
- Enabling virtual camera
- Allowing to account for occlusion ⇒ depth sensors

⊖
- Fixed sampling grid ⇒ limited capabilities to represent sensor optics
- Single bounce, single transformation ⇒ Simple lighting model
  No real intensity

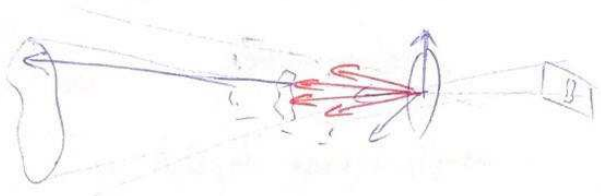
Ray Tracing

⊕ Allows reflection, transmissions

⊖ Multiple bounces, multiple transformation
"Real" intensity, distance attenuation
energy assorption



Final image created from rasterized image
Only vertices are transformed to raster image



Emit rays ⇒ find intersection point

4) Fundamentals of spatial scene description:

1) Geometry:

Hull / Surface Geometry

- Perfect for rasterization & ray tracing
- Is approximation (via triangle mesh) ↑ object surface

Volume Mesh

Only for ray tracing
- Approximate entire volume

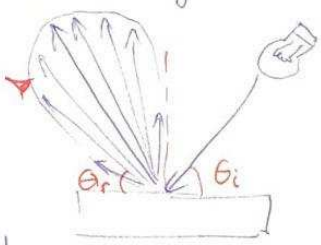
- Vertex, facet, facet type (triangle , lines, quads ..)
- Facet normals, vertex normals ... need normalization

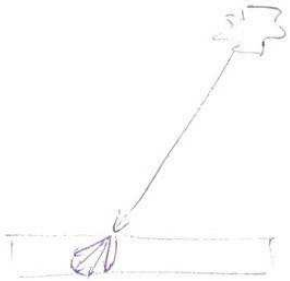
+) Highly detailed geometry : with textures
Basically is the height offset, added to the original surface

+) Materials

- Color : (but similar idea for other wavelength / frequency ..)
  How strong RGB rays are reflected

- Roughness :
  Angles matter





⇒ Brightness depends on viewing angle θ_r and lighting angle θ_i

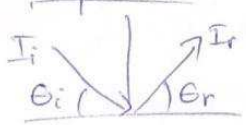
- BSDF = BRDF + BTDF
  Bidirectional [Scattering / Reflection / Transmission] Distribution function

- For single freq: $I_r = bsdf_\lambda(\theta_i, \theta_r, \Phi_r) \cdot I_i$ → Intensity input
  $bsdf_\lambda : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

- For multiple freq.s: (ex. color)
  $\underline{c}_r = bsdf_{rgb}(\theta_i, \theta_r, \Phi_r)^T \cdot \underline{c}_i \; ; \; \underline{c}_i, \underline{c}_r \in \mathbb{C}$

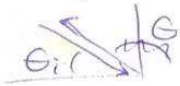
- Typical reflection characteristics:

Specular



$\theta_i = \theta_r$
$0 \le \frac{I_r}{I_i} \le 1$
Ideal reflection

Retroreflective

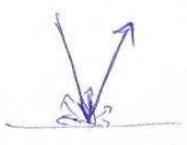


$\theta_i = \theta_r$ (same direction)
$I_i = I_r$

Diffuse
Roughness $\gg \lambda$
$\frac{I_r}{I_i} \approx \cos\theta_r$

≈ Mostly reflected in specular manner

Glossy
Roughness $\ll \lambda$
Minor diffuse



5) Rasterization - Fundamentals of rasterization-based camera & depth simulation

- Homogeneous coordinates $\underline{x} = \begin{bmatrix} a \\ b \\ c \\ w \end{bmatrix}$ describe translation and linear geometric transformations by 1 matrix

$${}^{0}\underline{T}_{1} = \begin{bmatrix} {}^{0}\underline{A}_{1} & {}^{0}\underline{d} \\ \underline{0} & 1 \end{bmatrix}$$

- Affine transformation: $T_{41} = T_{42} = T_{43} = 0, \quad T_{44} \neq 0$
- Multiple homo. transf.: ${}^{n}\underline{x} = {}^{n}\underline{T}_{n-1} \cdot {}^{n-1}\underline{T}_{n-2} \cdots {}^{1}\underline{T}_{0} \cdot {}^{0}\underline{x}$

$$= {}^{n}\underline{T}_{0} \cdot {}^{0}\underline{x}$$

- Rotation

$${}^{0}\underline{x} = \begin{bmatrix} {}^{0}\underline{R}_{1} & \underline{0} \\ \underline{0} & 1 \end{bmatrix} \cdot {}^{1}\underline{x}$$

Translation

$${}^{0}\underline{x} = \begin{bmatrix} \underline{I} & {}^{0}\underline{d} \\ \underline{0} & 1 \end{bmatrix} \cdot {}^{1}\underline{x}$$

Scale

$${}^{0}\underline{x} = \begin{bmatrix} \mathrm{diag}(\underline{s}) & \underline{0} \\ \underline{0} & 1 \end{bmatrix} \cdot {}^{1}\underline{x}$$

Shear

$$\begin{bmatrix} 1 & t_{12} & t_{13} & 0 \\ t_{21} & 1 & t_{23} & 0 \\ t_{31} & t_{32} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

+) Coordinate systems:

- Model C.S. (MS)
- World C.S: (WS)
- Tangent space C.S.: (TS)
- Eye/camera/senser (C.S.: (ES)
- Image C.S. (IS)

- Pinhole Camera Model: ${}^{IS}\underline{x}'$ in [m], ${}^{IS}\underline{x}$ in [pixel]

$${}^{IS}x = \frac{w}{s_x} f \frac{{}^{ES}x}{{}^{ES}z} + c_x \quad \Leftrightarrow \quad {}^{ES}z \cdot \begin{bmatrix} {}^{IS}x \\ {}^{IS}y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & & c_x & 0 \\ & f_y & c_y & 0 \\ & & 0 & 0 \\ & & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} {}^{ES}x \\ {}^{ES}y \\ {}^{ES}z \\ 1 \end{bmatrix}$$

$${}^{IS}y = \frac{h}{s_y} f \frac{{}^{ES}y}{{}^{ES}z} + c_y$$

$${}^{ES}z \cdot {}^{IS}\underline{x} = {}^{IS}\underline{T}_{ES} \cdot {}^{ES}\underline{x}$$

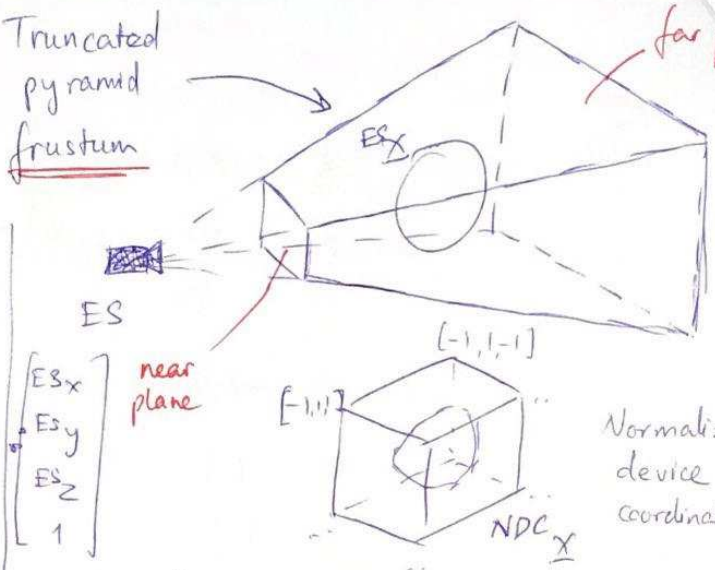
with f: focal length

$f_x = \frac{w}{s_x} f$, $f_y = \frac{h}{s_y} f$: focal length in image coord. [pixel]

c_x, c_y: principle point [pixel]

$$\begin{bmatrix} PC_x \\ PC_y \\ PC_z \\ PC_w \end{bmatrix} = PC_w \cdot \begin{bmatrix} NDC_x \\ NDC_y \\ NDC_z \\ 1 \end{bmatrix} = -ES_z \cdot \begin{bmatrix} NDC_x \\ NDC_y \\ NDC_z \\ 1 \end{bmatrix}$$

Truncated pyramid frustum



Normalized device coordinates

$$= \begin{bmatrix} \frac{2z_n}{r-l} & & \frac{r+l}{r-l} & \\ & \frac{2z_n}{t-b} & \frac{t+b}{t-b} & \\ & & -\frac{z_f+z_n}{z_f-z_n} & -\frac{2z_f z_n}{z_f-z_n} \\ & & -1 & 0 \end{bmatrix} \begin{bmatrix} ES_x \\ ES_y \\ ES_z \\ 1 \end{bmatrix}$$

$[l, r] \rightarrow [-1, 1]$

$[b, t] \rightarrow [-1, 1]$

Perspective projection for rasterization

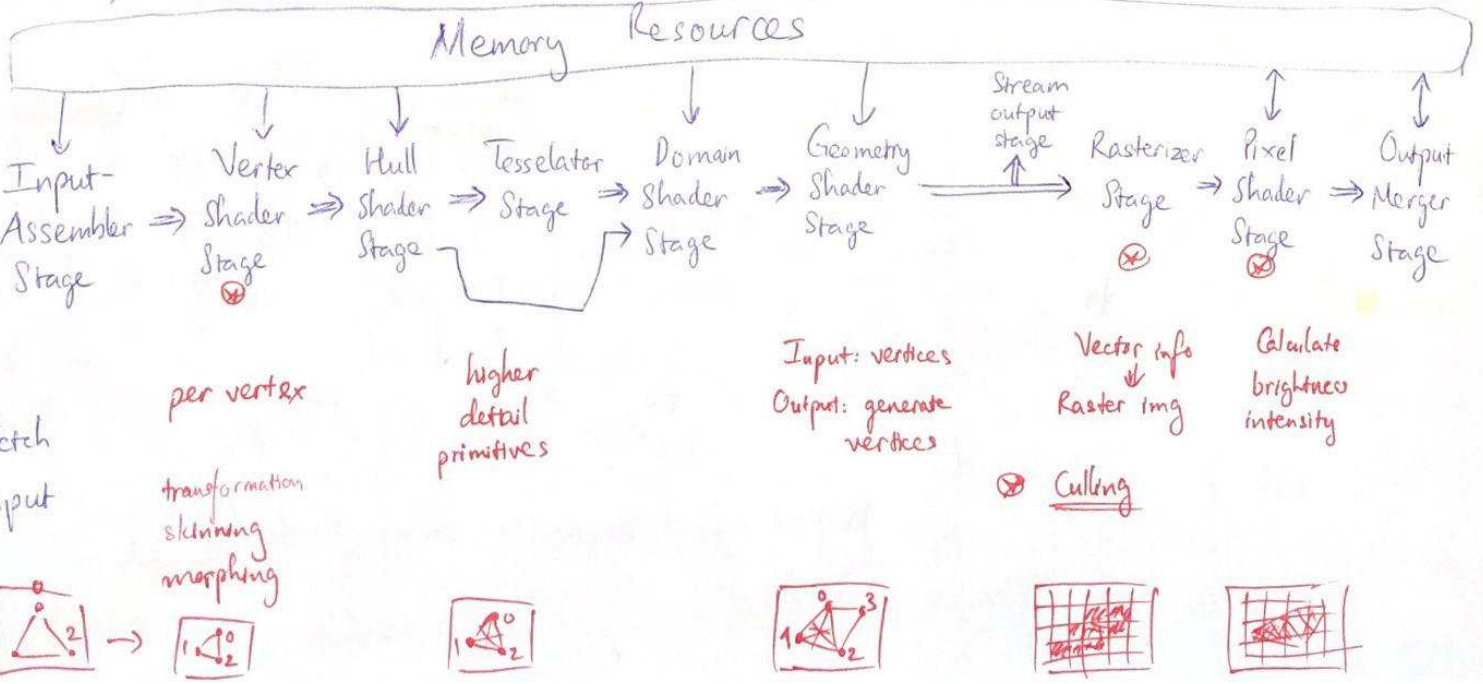
$$\Rightarrow -ES_z \cdot \begin{bmatrix} IS_x \\ IS_y \\ IS_z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w \cdot z_n}{r-l} & 0 & \frac{w}{r-l} \cdot \frac{r+l}{2} - \frac{w}{2} & 0 \\ 0 & \frac{h \cdot z_n}{t-b} & \frac{h}{t-b} \cdot \frac{t+b}{2} - \frac{h}{2} & 0 \\ 0 & 0 & -\frac{z_f+z_n}{z_f-z_n} & -\frac{2z_f z_n}{z_f-z_n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} ES_x \\ ES_y \\ ES_z \\ 1 \end{bmatrix}$$

+) World view projection matrix:

$$T_{WVP} = {}^{IS}T_{MS} = {}^{IS}T_{ES} \cdot {}^{WS}T_{ES}^{-1} \cdot {}^{WS}T_{MS}$$

compare with ${}^{IS}T_{ES}$ of pinhole camera

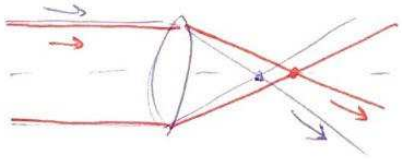
6) Rasterization - The Render Pipeline for rasterization



7) Rasterization - Post processing for the simulation of sensor effects

+) Chromatic Aberration

- Refraction index varies for diff. wavelength



⇒ +) Lens distortion:

- The further the distance from optical center, the stronger the distortion

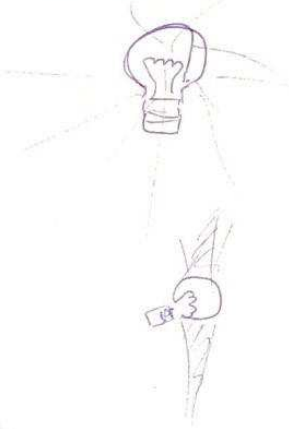
$$^{IS}\underline{x} = {}^{IS}\underline{T}_{ES} \cdot F_{distort}\left({}^{WS}\underline{T}_{ES}^{-1} \cdot {}^{WS}\underline{x}\right)$$

too complex to compensate

$F_{distort} : \mathbb{R}^4 \rightarrow \mathbb{R}^4$

Use camera calibration method to estimate distortion

+) Lens flare



+) Smear effect of CCD sensors

+) Depth of field Blur or in focus

+) Image noise

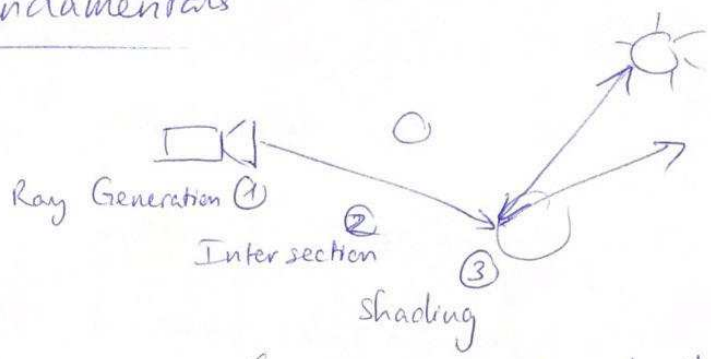
8) Rasterization - APIs, frameworks & applications

Open GL, Vulkan, unity, Direct X, Unreal Engine

Cry Engine

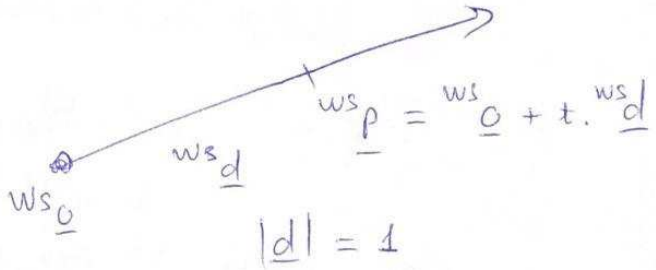
9) Ray Tracing - Fundamentals

+) Basic idea



+) Physically plausible : rasterization (misses some of real world effect
Physically correct : ray tracing

+)

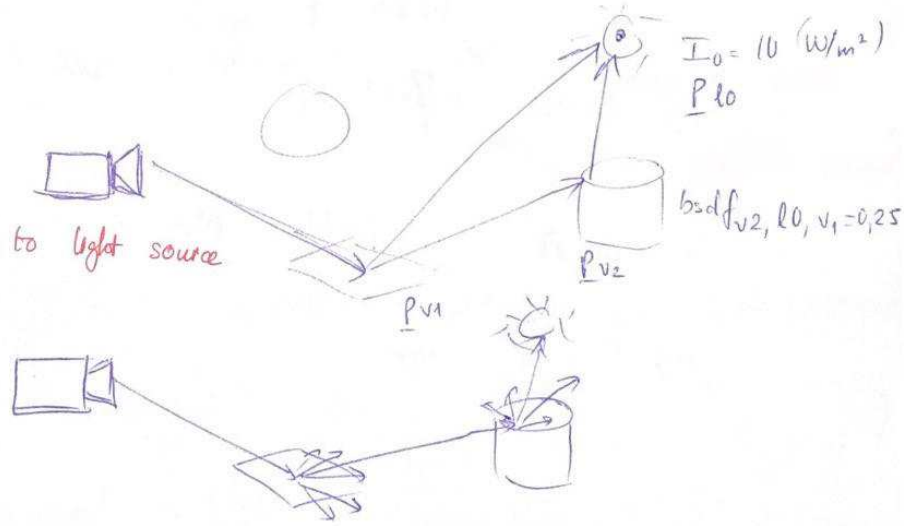


+) Forward ray tracing vs Backward ray tracing (Mostly used)

Lots of rays never reach senser

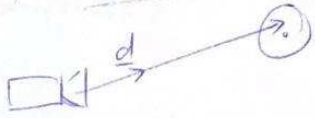
+) Shadow rays

the ray from hitpoint to light source



+) Scattering

+) Primitive intersections

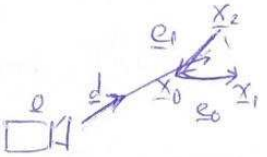


$\underline{p} = \underline{o} + t \cdot \underline{d}$ point on ray

$|\underline{p}_{sphere} - \underline{c}_{sphere}| = r_{sphere}$ point on sphere

=> Intersection:

$|\underline{o} + t \cdot \underline{d} - \underline{c}_{sphere}| = r_{sphere}$



$\underline{n} = \underline{e}_1 \times \underline{e}_0$

Normalized barycentric coordinates: (α, β, γ)

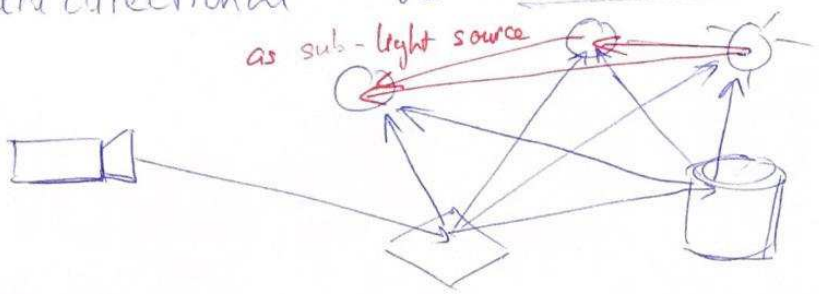
$\underline{p} = \alpha \underline{x}_0 + \beta \underline{x}_1 + \gamma \underline{x}_2$

Points within triangle: $\alpha + \beta + \gamma = 1$

- Solve t: $\left[(\underline{c} + t\,\underline{d}) - \underline{x}_0\right]^T \cdot \underline{n} = 0$
- Find α, β, γ: $\underline{o} + t \cdot \underline{d} = \alpha \underline{x}_0 + \beta \underline{x}_1 + \gamma \underline{x}_2$ such $\alpha + \beta + \gamma = 1$

10) Ray Tracing - Optimizations

+) Unidirectional vs bidirectional ray tracing



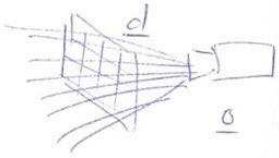
+) Since Ray tracing is just sampling of scattering
⇒ subject to lots of noise

⇒ +) Classical Denoising: { use random generators
linear interpolation { multiple iter > 10
accumulate results

+) AI Denoising: with Auto encoder
better, less iterations

11) Ray Tracing - Senser Parameters & Dynamic behavior in ray tracing

+) Pinhole camera



- with fodal length f_x, f_y [pixel] ⇒ Simply generate all rays to all pixels
principle point c_x c_y [pixel]

- Multiple rays per pixel based on uniform distribution

modified directions

$$\begin{bmatrix} {}^{ES}d'_x \\ {}^{ES}d'_y \\ {}^{ES}d'_z \end{bmatrix} = \begin{bmatrix} {}^{ES}d_x \\ {}^{ES}d_y \\ {}^{ES}d_z \end{bmatrix} + \left[\frac{S_x}{w}, \frac{S_y}{h}, 0 \right]^T \cdot \begin{bmatrix} rnd(seed) - 0,5 \\ rnd(seed) - 0,5 \\ 0 \end{bmatrix}$$

Then get ${}^{ES}\underline{d}'$: "distorted" ray direction

+) LIDAR Beam Divergence ⊖

Emission Distribution: $I(r,z) = I_0(z)\, e^{-2\frac{r^2}{w(z)}}$ r: distance from center

$$I_0(z) = P_L \cdot \frac{2}{\pi\, w(z)^2}$$

+) Importance Sampling:

Directions with higher intensity get more samples

+) Also consider Doppler effect:

As camera / object moves ⇒ affect the detected wavelength / intensity

⇒ different shade / color / intensity ...

$$\underline{v}_p = \underline{v}_{cg} + \underline{\omega} \times \underline{r}$$

$$\Delta f = 2\frac{v_{rad}}{\lambda} = 2 v_{rad} \frac{f_t}{c}$$

$$\underline{v}_{rad} = \underline{v}_s - \underline{v}_p$$

+) LIDAR Sensor Modelling:

$$\begin{bmatrix} \text{azimuth} \\ \text{elevation} \end{bmatrix} = \begin{bmatrix} \omega_{rot} \cdot t \\ el \end{bmatrix}$$

Verification & Validation

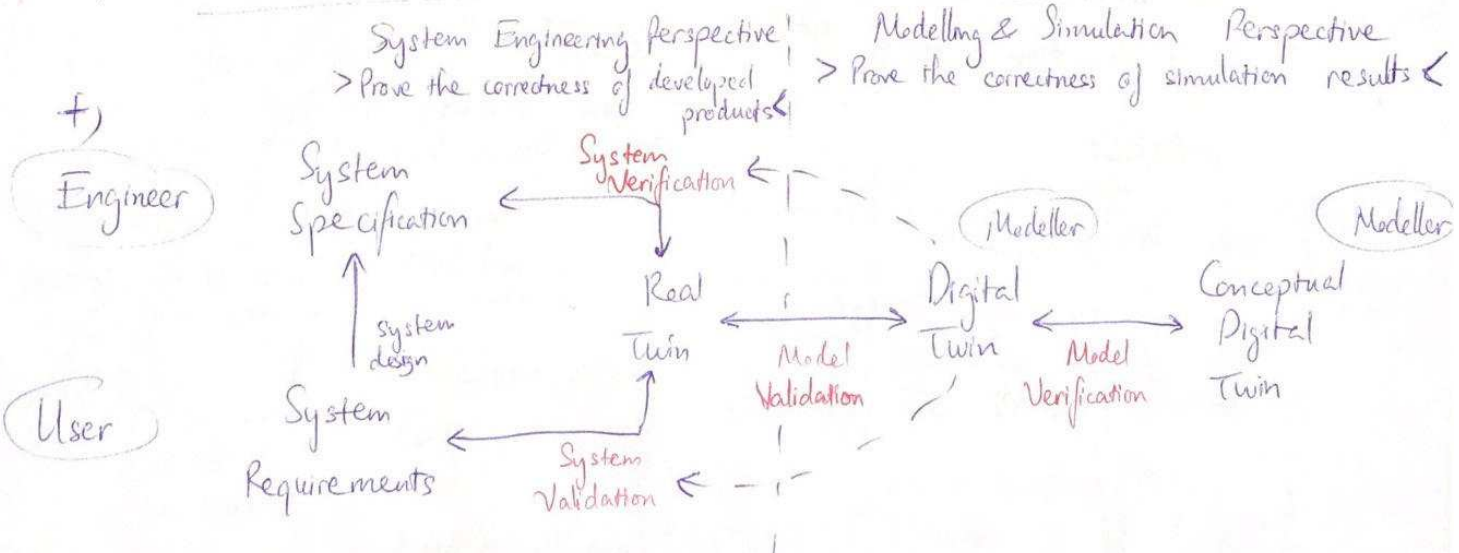
(42)

1, Introduction:

- Systems are getting more complex ⇒ higher risk of system failure
  ⇒ high costs of late detected
- "The rule of ten": the earlier an error is discovered and eliminated ⇒ less cost is lost
- Definition:

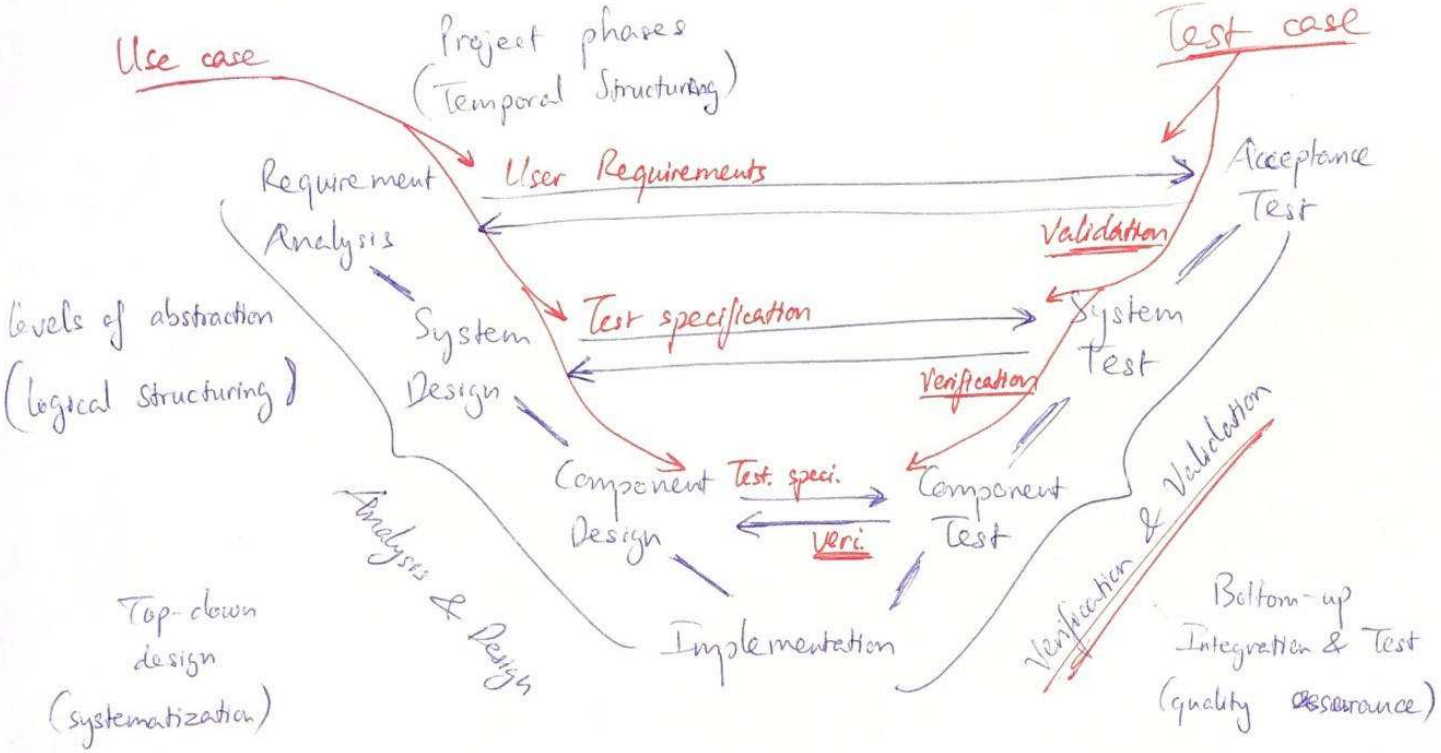
	Validation	Verification
	intended use	specification
	→ check specified requirements from users	(from engineering side)
Prove	it is the right thing	the thing is right

+)



2) Verification & validation of technical products

+ The traditional V-model (Verification & Validation in VDI 2206)



+ Verification & Validation could be done in

- Virtual experiment
- Real experiment
- Hardware-in-the-loop — real component in simulation envi.
- Software-in-the-loop — system models in modelled process

+ Verification methods in ECSS:

(European Cooperation for Space Standardization)

- Test
- Analysis
- Review of design
- Inspection

+) TRL (Technology Readiness Level)

- A systematic measurement system to assess the maturity of a tech
  compare

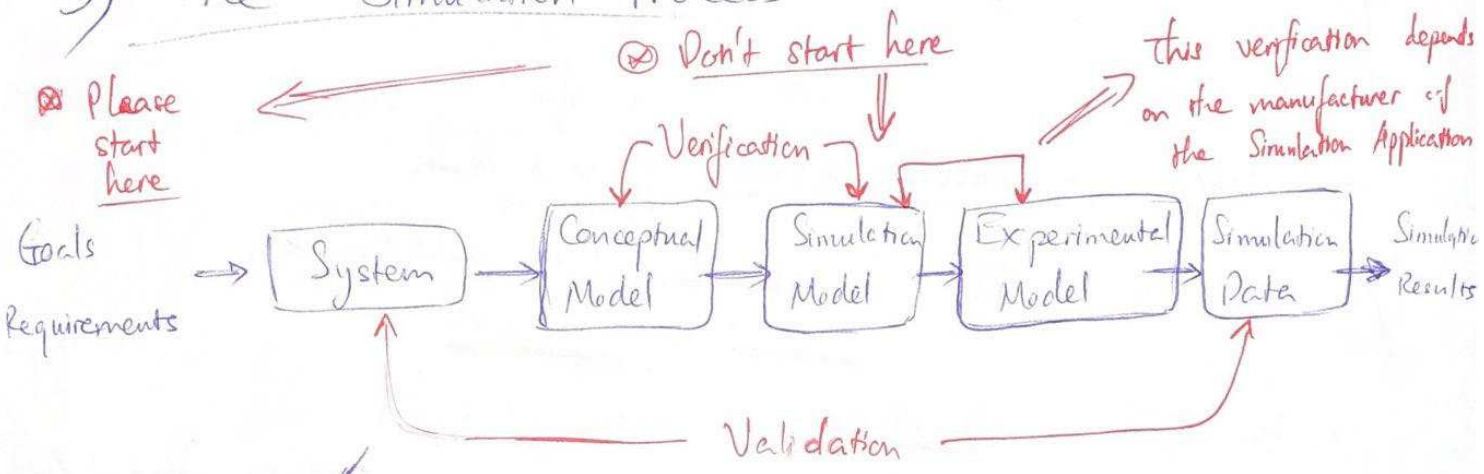
- TRL 1 → TRL 9
  Basic principles observed & detected — mature technology

+) Virtual Commissioning (VCOM): to test software controller ..

- Test configs & methods of VCOM:

  AT config → Model in the loop → SIL → HIL
  MIL

3) The Simulation Process



Determination of questions to be solved	Conceptual Modelling	Formalization	Implementation	Execution of simu. exp.	Post processing & interpretation
- MBSE .. - System context components param - ..	- Scenario defin. EDT - Avoid type III error	Create EDT & EDT scenario in specific language	Assemble Simu. Entities In a simulator software ..	Run the "Simulation" Logging data	

+) Errors types:

- Type III: does not completely contain the actual problem
  Ex: simulate an mobile robot / autonomous car with out LIDAR sensor
- Type I: simulation results are rejected even though they are sufficiently credible
- Type II: _____ are accepted not _____

4) Basic Concepts

+) Validity
- relative indication of appropriateness
- use fidelity info for specific purposes

+) Fidelity
- absolute indication of M&S results
- Def: the accuracy of the model / simulation compared to real world

+) Accuracy: - the degree .. conforms exactly to reality / chosen standard / referent
(articulation of abstraction)

+) Precision - level of resolution / granularity
⇒ places limit on accuracy

+) Error - differences between observed / measured value vs correct one

+) Timeliness [exact replication of time behavior
exact points in time of state changes, events ..

⊛ Note: – Simulations deliver "exact" results within defined scope
⇒ Level of fidelity required is one of the main cost drivers of a simulation

+) Fidelity descriptions: 3 basic categories:

- Short descriptions : qualitative
- Short hand descriptions } quantitative
- Long descriptions } describe in terms of multiple attributes
- Qualitative terms: accurate, emulated, exact, static, functionally, plausible, realistic, representative

5) Verification

+) Typical Verification actions:

- Estimate expected results.
- Calculate with simplified model
- Analyze code, structure
- ...
- Calculate with different methods
- Compare with physical sense..

X-in the loop

(43)

1, Introduction

- X could be
  - human
  - hardware
  - software
  - model

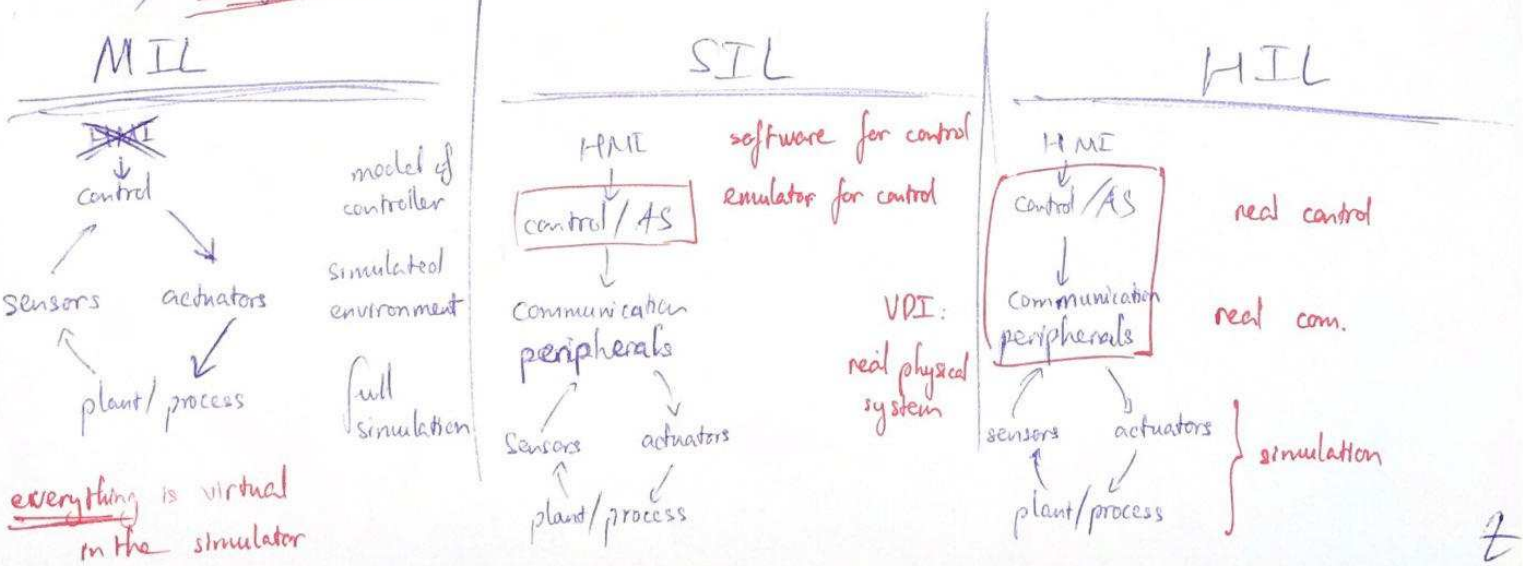
2, Virtual Commissioning

+) Definition: - commissioning : testing of | components / functions using | simulation method & model

⇒ virtual commissioning is overall test preceeding actual commissioning

+) Advantages
- Can be proponed in early stages
- Offer safety
- Reduced hardware complexity required

+) Configurations of the entire automation system



Virtual Commissioning Methods

- Forcing: manual override of signals : input, output..

- M1 System signal manipulation: MIL, SIL, HIL
- M2 Control-based system simulation SIL, HIL
- M3. System simulation : HIL, SIL
- M4. System simulation and control : HIL
- M5. Holistic system simulation & control : everything is in the Simulator
  MIL / SIL

SRSP

Man - Machine Interaction & Assistance Systems

(4)

1) Introduction

- DT as mediator (design, control, analysis, production, education)
- Different types
  - users
  - user levels
  - focus of user
  - assistance required
  - applications

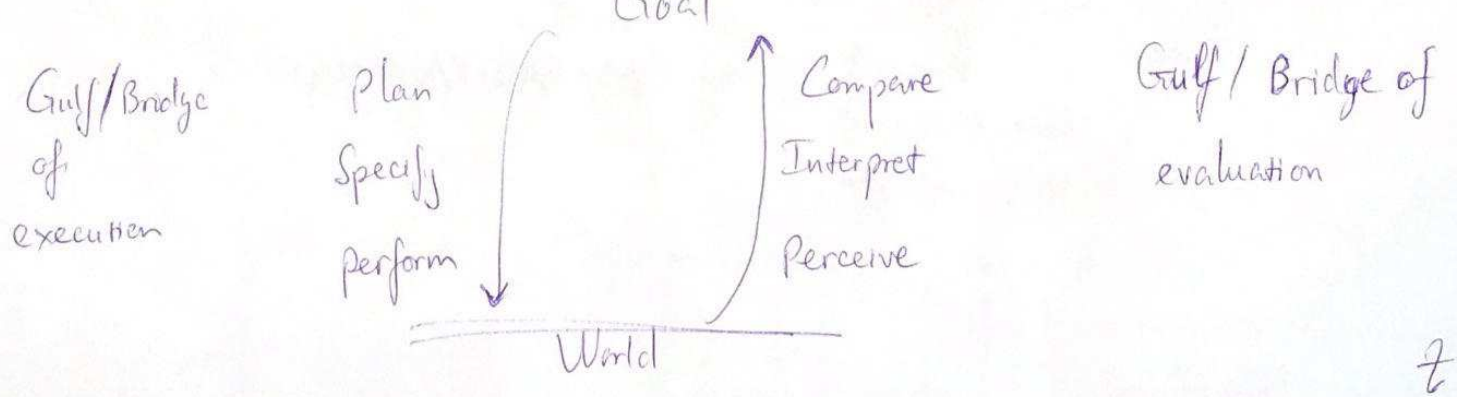
  ⇒ different scenarios
    - design, development
    - marketing
    - disaster scenarios
    - industrial
    - mobile robot
    - environment
    - education
    - ...

2) Fundamentals of Man - Machine Interaction:

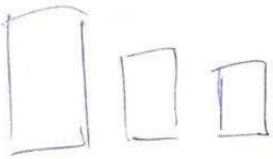
+) 7 interaction principles:

- suitability for the user's tasks
- Self-descriptiveness — present info understandable
- Conformity with user expectations — predictable behavior
- Learnability
- Controllability
- Use error robustness — avoid error, recover from ↑
- User engagement — capture attention

+) Norman: 7 stages of action:



+) Perception vs Cognition



easy to visualize
faster, clearer

889,56
566,55
376,02

+) Danger through depth : people do not see in 3D rather in 2,05 D
→ danger of concealment

+) Metaphor: concepts properties | that are familiar → similar | meaning function behavior structure

+) Flat design vs & Skeuomorphism

+) 3D User Interfaces & 3D Interaction
(User Interface with 3D interaction) (task in 3D spatial context)

3) Visualization, Interaction & Feedback

+) Input | output of Interfaces

Input:
- Keyboards
- Controller
- Optical
- Haptic

output:
- 2D Data
- 3D projection screen
- VR / AR / MR

+) Immersion → Presence → Perception → Cognition

physically present in a non-physical world | subjective sensation of being there | 5 senses must perceive | acquiring knowledge

← User's understanding of the behavior gets better

+) Virtual Reality (VR) | Assisted Reality (AsR) | Augmented Reality (AR) | Mixed Reality (MR)

+) Haptic

- Interaction involving touch
- Force & torque feedback

Kinesthetic

- sense of self movement
- Full body movement devices

4) EDT - based analysis & Optimization

+) Victor's Seeing Spaces:

- Key idea: the challenge is not building it, but understand it.
- 3 aspects / dimensions: seeing [inside, across time, across possibilities]
- 2 golden rules of information design [show the data, show comparisons]
- Despite the fascinating models & high stakes, interactive control / dynamic visualization are shockingly rare.

+) Analyzing EDTs

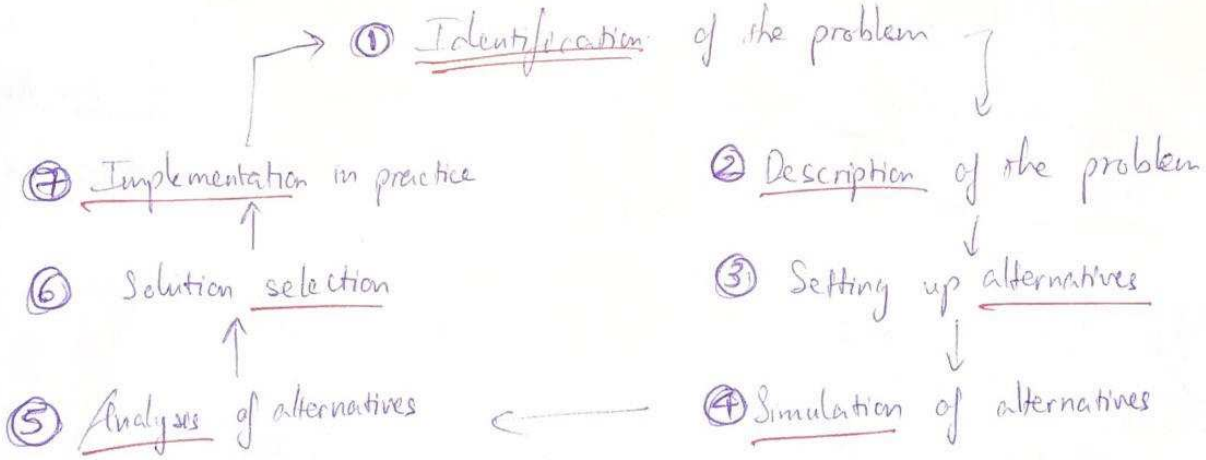
- Utilize Database : SQL or plain files (JSON, XML, CSV..)
- 3D displays: [frame tracks , replay
  isochrones , particle animations
  ghosts..

+) Optimizing EDTs

- Simulation-based optimization

5) EDT based User Interfaces Training & Decision Support

+) 7 steps of Decision Making Process



SRSP

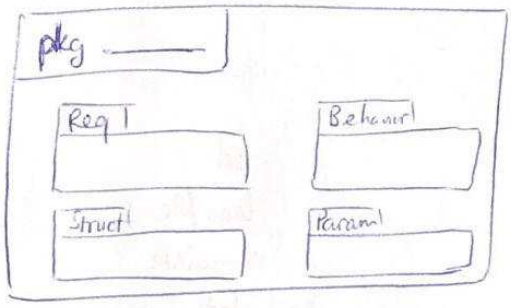
Exercise

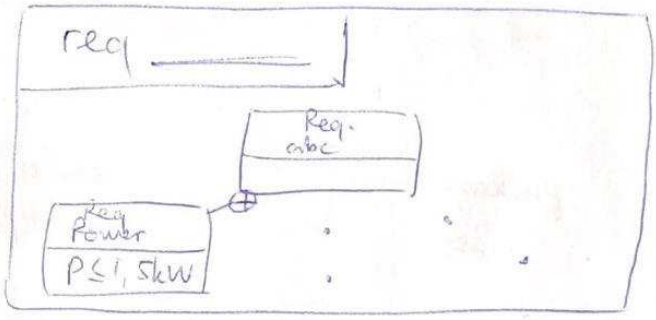
L4, None exercise

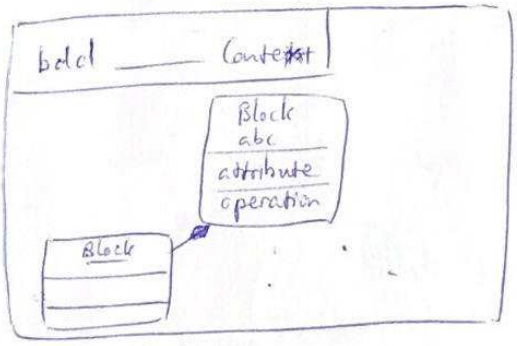
L2,

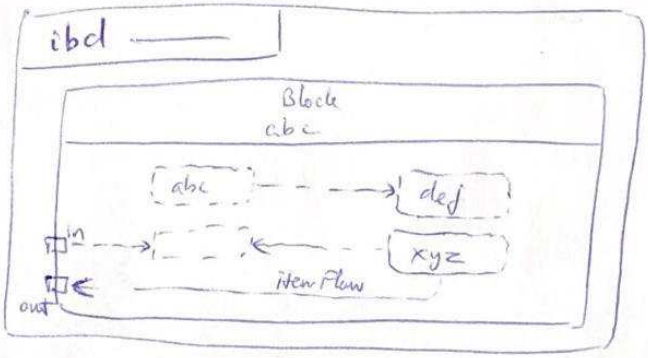
L3, No Exercise

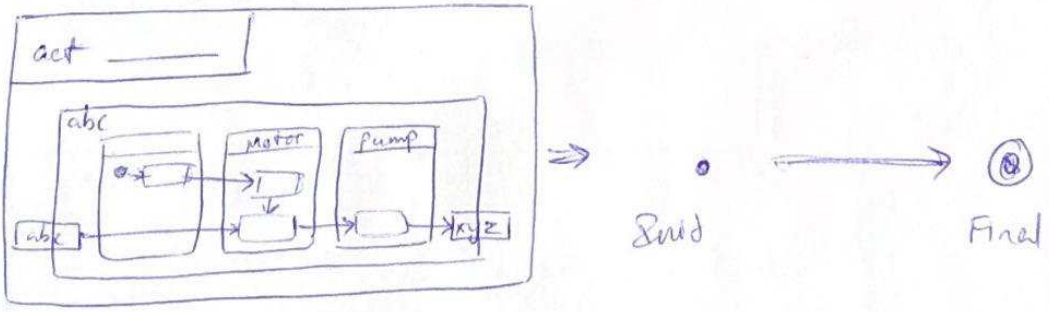
L4, MBSE











L8, MatLAB

```
(y = 2t)
tspan=[0 5];
y0 = 0;
[t, y] = ode45(@(t,y) 2*t, tspan, y0);
```

ode = @(t,x) func(..)

[t,x] = ode45(ode, —, —)

⇒ plot(t, y);

figure, subplot, hold all, plot, title, xlabel, legend, grid on, ..

L9, Modelica

```
model —
  Real _(start = _);
  parameter Real _ = _;
equation
  der(_) = _;
end —;
```

```
class —
  Real ..
  parameter ..
equation
  der ..
end —;
```

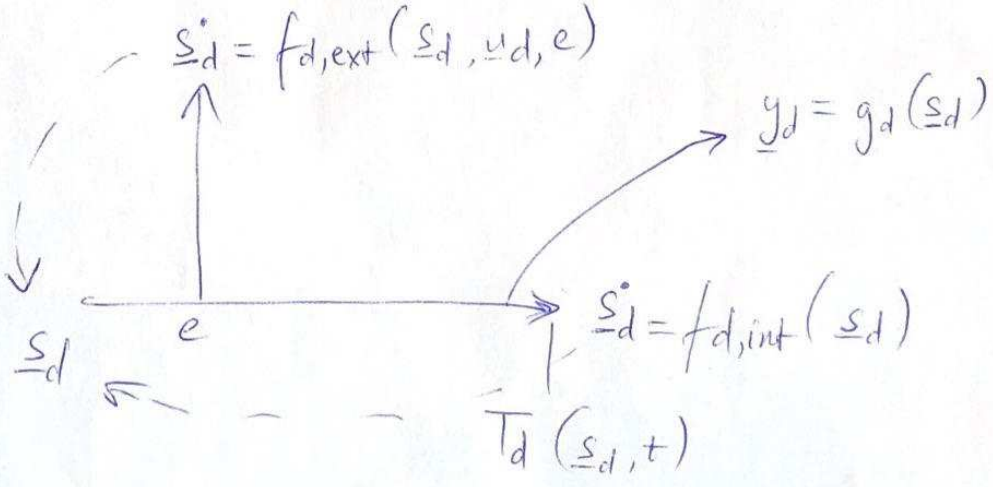
```
model —
  class1 name A;
  class2 name B;
equation
  connect(A.y, B.u);
end —;
```

```
package —
  class ..
  ..
end —;
```

```
model —
  import —;
```

```
cd(" .. ")
loadfile(" .. ")
simulate(—, stopTime = _)
plot({x1, x2, x3})
```

L6, DES



40, Rigid Body

+) Generalized Coordinates: $q_1 \dots q_n$

Maximal Coordinates: $\begin{bmatrix} x_1^T & \dots & x_n^T \end{bmatrix}^T$

$$x_i = \begin{bmatrix} x_i, & y_i, & z_i, & \alpha_i, & \beta_i, & \gamma_i \end{bmatrix}^T$$

+) Quaternion $\quad \dot{\underline{q}} = \frac{1}{2} \begin{bmatrix} 0 \\ \underline{\omega} \end{bmatrix}_{4\times1} \circ \underline{q}_{4\times1} \quad ; \quad \underline{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos \alpha/2 \\ \underline{a} \cdot \sin \alpha/2 \end{bmatrix}$

+) Rigid Bodies: $\quad {}^{w_2}\underline{\Theta}_{cog} = {}^{w_2}\underline{R}_{w_1} \cdot {}^{w_1}\underline{\Theta}_{cog} \cdot \left({}^{w_2}\underline{R}_{w_1}\right)^T$

+) Derivation of equation of motion:

$$\underline{f}_{ext,i} = \frac{d\underline{p}_i}{dt} = m_i \cdot \underline{\dot{v}}_i$$

$$\underline{\tau}_{ext,i} = \frac{d\underline{L}_i}{dt} = \underline{\Theta}_i \, \underline{\dot{\omega}}_i + \underline{\omega}_i \times \underline{\Theta}_i \, \underline{\omega}_i$$

$$\Leftrightarrow \quad \underline{\ddot{x}} = \underline{M}^{-1} \cdot \underline{f}_{ext} = \begin{bmatrix} m_1^{-1}\underline{I} & & \\ & \underline{\Theta}_1^{-1} & \\ & & \ddots \end{bmatrix} \cdot \begin{bmatrix} \underline{f}_{ext,1} \\ \underline{\tau}_{ext,1} - \underline{\omega}_1 \times \underline{\Theta}_1 \, \underline{\omega}_1 \\ \vdots \end{bmatrix}$$

+) Lagrangian: $\quad L = E_{kin} - E_{pot}$

$$\Rightarrow \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = Q$$

$$\Leftrightarrow \quad H(q) \cdot \ddot{q} + C(q, \dot{q}) = \tau$$

+) $\quad \underline{v}_p = \underline{v}_{cog} + \underline{\omega} \times \underline{r}$

$$\Rightarrow \quad \underline{a}_p = \underline{\dot{v}}_{cog} + \underline{\dot{\omega}} \times \underline{r} + \underline{\omega} \times (\underline{\omega} \times \underline{r})$$

Constraints

holonomic: $C(\underline{x}(t), t) = \underline{0}$

non-holonomic: $C(\underline{x}(t), t) \geq \underline{0}$

$$\Rightarrow \frac{d}{dt} C(\underline{x}(t), t) = \frac{\partial C}{\partial \underline{x}} \cdot \dot{\underline{x}} + \frac{\partial C}{\partial t} = \underline{J} \cdot \dot{\underline{x}} + \frac{\partial C}{\partial t} = \underline{0}$$

$$\Rightarrow \frac{d^2}{dt^2} C(\underline{x}(t), t) = \underline{J} \cdot \ddot{\underline{x}} + \frac{\partial^2 C}{\partial \underline{x}^2} \cdot \dot{\underline{x}}^2 + 2 \frac{\partial^2 C}{\partial t \partial \underline{x}} \dot{\underline{x}} + \frac{\partial^2 C}{\partial t^2} = \underline{0}$$

Constraint forces:

Principle D'Alembert's: $\underline{f}_c = \underline{J}^T \cdot \underline{\lambda}$

$$\Rightarrow P_c = \underline{f}_c^T \cdot \dot{\underline{x}} = (\underline{J}^T \underline{\lambda})^T \cdot \dot{\underline{x}} = \underline{\lambda}^T \cdot \underbrace{\underline{J} \dot{\underline{x}}}_{\underline{0}} = \underline{0}$$

$$\Rightarrow \underline{M} \ddot{\underline{x}} = \underline{f}_{ext} + \underline{f}_c = \underline{f}_{ext} + \underline{J}^T \underline{\lambda}$$

$$\underline{J} \ddot{\underline{x}} = \underline{b}$$

$$\Rightarrow \begin{bmatrix} \underline{M} & -\underline{J}^T \\ \underline{J} & \underline{0} \end{bmatrix} \cdot \begin{bmatrix} \ddot{\underline{x}} \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \underline{f}_{ext} \\ \underline{b} \end{bmatrix}$$

$$\Rightarrow \underline{M} \ddot{\underline{x}} = \underline{f}_{ext} + \underline{J}^T \underline{\lambda} \Rightarrow \ddot{\underline{x}} = \underline{M}^{-1} (\underline{f}_{ext} + \underline{J}^T \underline{\lambda})$$

$$\Rightarrow \underline{J} (\quad) = \underline{b} \Rightarrow \underline{J} \underline{M}^{-1} \underline{J}^T \underline{\lambda} + \underline{J} \underline{M}^{-1} \underline{f}_{ext} - \underline{b} = 0$$

+) Simulation Loop of Rigid Body Dynamics

State Vector $x(t),\ \dot{x}(t)$ → Constraints Formulation $C(x(t), t) = 0$ → Inverse Dynamics $J M^{-1} J^T dt\lambda + (J\dot{x} + M^{-1} dt\, f_{ext}) - b = 0$ (constraint forces) → Forward Dynamics $\dot{x}(t+dt) = \dot{x}(t) + dt \cdot M^{-1}(f_{ext} + J^T \lambda)$ (all forces ⇒ movement) → State Vector

+) Simulation Loop of free-floating Rigid Bodies:

Current State $\underline{p}_i(t)\ \underline{q}_i(t)\ \underline{v}_i(t)\ \underline{\omega}_i(t)$ → Calculate of derivative of system state $v_i(t+dt) = \ldots$ → $\begin{cases} p_i(t+dt) = \\ q_i(t+dt) = \end{cases}$ → Normalization of quaternions $q_i = \frac{q_i}{|q_i|}$ → Recalculation of inertia tenser ${}^{w}\Theta_i(t+dt)$ → Current State

+) Simulation Loop of constrained Rigid Bodies

Lagrangian formalism

- Refine generalized coordinates $q_1 \ldots q_n$
- Calculate energy $L = E_{kin}(\underline{q}, \dot{\underline{q}}) - E_{pot}(\underline{q})$
- Calculate Lagrangian equation $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\underline{q}}}\right) - \frac{\partial L}{\partial \underline{q}} = Q$
- Calculate Equation of Motion $\underline{H}(\underline{q})\,\ddot{\underline{q}} + C(\underline{q}, \dot{\underline{q}}) = \underline{\tau}$

Newton-Euler formalism

- Cut free system
- Simulation of all forces/torques for each body $m\,\ddot{\underline{x}} = \sum_i \underline{f}_i$
- Eliminate constraint forces to derive equation of motion $\ddot{\underline{x}} = f(\underline{x}, \dot{\underline{x}}, \underline{f}_{ext})$

Simulation Loop for Constrained Rigid Body Systems (JMJT Approach)

- Current system state
- Constraint Jacobi matrix
- Calculation of Constraint forces
- Integration of equation of motion
- Recalculation of inertia tenser

Sensors

BSDF = BRDF + BTDF

$$\begin{bmatrix} {}^{IS}x \\ {}^{IS}y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & & c_x & 0 \\ & f_y & c_y & 0 \\ & & 0 & 0 \\ & & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} {}^{ES}x \\ {}^{ES}y \\ {}^{ES}z \\ 1 \end{bmatrix}$$

$$f_x = \frac{w}{s_x} f$$

$$f_y = \frac{h}{s_y} f$$

$${}^{ES}z \cdot {}^{IS}\underline{x} = {}^{IS}\underline{\underline{T}}_{ES} \cdot {}^{ES}\underline{x}$$

$$\underline{p} = \underline{o} + t \cdot \underline{d}$$