PERSONAL - Modeling and Simulation

- Introduction to Mathematical Modeling
- Decision Models: Games, Strategies, Elections
  - Generic Decision Model (1 Player)
    - Strategies under Certainty
    - Strategies under Risk
  - Two-Player Zero Sum Game
    - Strategy
  - Group Decision Making (Elections)
    - Collective Choice Functions
- Scheduling
  - Process Scheduling (DAG)
  - Job-Shop Problems
  - Types of Algorithms
- Population Dynamics
  - Model of Malthus (1 Species)
  - Model of Verhulst (1 Species)
  - Logistic Growth (1 Species)
  - Multiple Species
    - Arms Race
    - Equilibriums
- Ordinary Differential Equations (ODEs)
- Control Engineering, Fuzzy Logic
  - Linguistic Variables
  - Fuzzy Control System
- Heat Transfer (3D Space + Time), PDEs
  - Partial Differential Equations (PDEs)
    - Solving Linear Systems
- (Data) Traffic Simulation
  - Modelling Queuing Systems
  - Stochastic Processes

- Traffic Flow
  - Unsteady Situations

Introduction to Mathematical Modeling

- **descriptive tools**: algebraic equations, ODEs, PDEs
- **manageability and solveability**: existence, uniqueness of solutions; continuous dependency on input data
- **well-posed-problem**: existence + uniqueness + stability of solution
- **ill-posed-problem**: potentially unstable; slight disturbances may completely falsify or invalidate results
  - **inverse problem**: result given, initial configuration wanted
- **solution approaches**: analytic, heuristic, direct- or approximate-numerical
- **assessment**: validation, accuracy
- **classification**: discrete vs. continuous, deterministic vs. stochastic

Decision Models: Games, Strategies, Elections

- **certainty**: *all* rules, actions, conditions, consequences etc. are known
- **risk**: *some* elements are known, others unknown
- **uncertainty**: *no* rules, actions, conditions, consequences etc. are known

Generic Decision Model (1 Player)

- S: player / decision maker
- $A_1, ..., A_m$: actions / strategies
- $s_1, ...s_n$: states
- N: payoff matrix
  - $N_{ij} \in \mathbb{R}$: payoff for action A_i (rows) in state s_j (columns); the higher, the better

Strategies under Certainty

- **best possible outcome**: choose most profitable action for self
  - **formally**: choose $\hat{i} \in \{1, ..., m\}$ such that $N_{\hat{i}j} = \max_i N_{ij}$

Strategies under Risk

- **caution (max-min payoff)**: find the *minimum (worst case)* from each row, then find the *maximum (best case)* among these
  - **formally**: choose $\hat{i} \in \{1, ..., m\}$ such that $N_{\hat{i}\hat{j}_{\hat{i}}} = \max_i \min_j N_{ij}$

- **full risk (max-max payoff)**: find the *maximum (best case)* from each row, then find the *maximum (best case)* among these
  - **formally**: choose $\hat{i} \in \{1, ..., m\}$ such that $N_{\hat{i}j_{\hat{i}}} = \max_i \max_j N_{ij}$
- **alternative-caution (min-max risk)**: find the *maximum (highest payoff)* from each row of *risk matrix*, then find the *minimum (lowest risk)* among these
  - **formally**: choose $\hat{i} \in \{1, ..., m\}$ such that $R_{\hat{i}j_{\hat{i}}} = \min_i \max_j R_{ij}$
    - $N_j = \max_i N_{ij}$: maximal payoff with state j
    - $R_{ij} = N_j - N_{ij}$: risk at action i with state j; for each column, subtract each value in column from maximum of column
      - **example**: $N = \begin{pmatrix} 0 & 100 \\ 1 & 1 \end{pmatrix}, R = \begin{pmatrix} 1-0 & 100-100 \\ 1-1 & 100-1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 99 \end{pmatrix}$

Two-Player Zero Sum Game

- S_1: *win* player
  - $A_1, ..., A_m$: win player actions (*rows* of matrix)
- S_2: *loss* player
  - $B_1, ..., B_n$: loss player actions (*columns* of matrix)
- a_{ij}: payoff of S_2 to S_1

Strategy

- **win player**: maximize guaranteed win; find the *minimum* of each *row*, then find *maximum* among these
  - **formally**: choose $\hat{i}$ such that $a_{\hat{i},j_{\hat{i}}} = \max_i \min_j a_{ij}$
- **loss player**: minimize guaranteed loss; find *maximum* of each *column*, then find minimum among these
  - **formally**: choose $\hat{j}$ such that $a_{i_{\hat{j}},\hat{j}} = \min_j \max_i a_{ij}$
- **inequality**: guaranteed win of S_1 is *less or equal* to guaranteed loss of S_2
  - **formally**: $a_{\hat{i},j_{\hat{i}}} \leq a_{\hat{i},\hat{j}} \leq a_{i_{\hat{j}},\hat{j}}$
  - **saddle point / equilibrium / optimal strategy pair**: the same strategy is optimal for both players
    - **formally**: $\max_i \min_j a_{ij} = a_{\hat{i},\hat{j}} = \min_j \max_i a_{ij}$

Group Decision Making (Elections)

- A: set of *candidates* (whom the public will vote for)
- $r : A \to \mathbb{N}$: surjective *rank mapping* of candidate to their respective rank from 1 to k

    - there can be *duplicate* rankings (so $1:...,2:...,2:...,4:...$)
    - $r(x)$ is the place of candidate x
    - **preference relation**: x is better than y if x's rank is lower than y's
        - **formally**: $x \varrho y \iff r(x) < r(y)$
        - **transitive and asymmetric**
            - **transitive**: if $x \varrho y$ and $y \varrho z$, then $x \varrho z$
            - **asymmetric**: $x \varrho y$ and $y \varrho x$ cannot hold simultaneously
        - **alternative**: $x \varrho^* y \iff r(x) \leq r(y)$
            - $x \varrho y \iff \neg(y \varrho^* x)$
- $I = \{1, ..., n\}$: set of voters, where each voter i has a rank mapping r_i (i.e. their own preferences)
- P_A: set of *all possible relations* on A to be found through a rank mapping
    - **formally**: $P_A = \{\varrho \subset A \times A \mid \varrho \text{ is generated by a rank mapping } r\}$

Collective Choice Functions

- **collective choice function**: $K : P_A \times ... \times P_A \to P_A$ (result same candidates as input candidates)
    - should adhere to **democratic basic rules**
        - K must be *total* on P_A^n (individual freedom of decision making, any ϱ_i allowed)
        - result of K must always lie in P_A (result in same set as choices; holds in two-party systems)
        - **Pareto condition**: supremacy of collective; if all voters prefer x, it can't be that y wins
        - independence of irrelevant alternatives; ballots with *identical ranking* w.r.t x and y must yield same *collective ranking* w.r.t x and y
        - exclusion of a dictator; no voter can always prevail
- **majority decision (Condorcet method)**: $x \varrho y \iff N(x, y) > N(y, x)$ with $N(x, y)$ as number of voters with $x \varrho_i y$
    - can lead to *voting paradox* ($x \varrho y, y \varrho z, z \varrho x$, violates transitivity), violates rule 2
- **rank addition**: sum up rank mappings of individual preferences
    - **formally**: $x \varrho y \iff \sum_{i=1}^{n} r_i(x) < \sum_{i=1}^{n} r_i(y)$
    - violates rule 4
- **(!) Arrow's impossibility theorem**: in the case of *more than two candidates* and *more than one voter*, there *cannot* exist a collective choice function that *satisfies all five democratic basic rules*

Scheduling

- $A_1, ..., A_n$: tasks / jobs

- $M_1, ..., M_m$: machines
- $t_i^{(j)}$: execution time for task A_i on machine M_j

Process Scheduling (DAG)

- s_i: start time of task i
- t_i: execution time of task i
- $c_i = s_i + t_i$: completion time of task i
- $A_i \to A_j$: precedence condition (task A_j can only be started after A_i is finished)
  - **formally**: $A_i \to A_j \iff c_i \leq s_j$
- **schedule**: function, assigns start time s_i to every task i
  - **admissible schedule**: all precedence conditions are fulfilled

Job-Shop Problems

- **job-shop model**: a job is broken down into subjobs, each with their own execution times, requiring a machine, where each machine can only process one subjob at a time with *no recirculation* (i.e. every job requires every machine at most once)
- **open shop model**: arbitrary permutability of a tasks subjobs between respective machines
- **flow shop model**: job-shop but all subjobs pass through the machines in the same order

Types of Algorithms

- **Online Algorithm**: processes input in a serialized way, i.e. the entire input does not need to be available from the start (e.g. insertion sort)
- **Offline Algorithm**: must have the whole input data available from the beginning (e.g. selection sort)
- **Greedy Algorithm**: always makes the locally optimal choice at each stage (e.g. Dijkstra's algorithm)
- **Approximation Algorithm**: computes a solution in polynomial time to (typically NP-hard) optimization problems and guarantees that the solution is within a certain factor of the optimal solution

Population Dynamics

- time $\to$ *ordinary* differential equations
- time + space $\to$ *partial* differential equations

Model of Malthus (1 Species)

- γ: *constant* birth rate per time unit and per individual
- δ: *constant* death rate per time unit and per interval
- $\lambda = \gamma - \delta$: *constant* growth rate
- **ODE**: $\dot{p}(t) = \lambda p(t)$

- **solution**: $p(t) = p_0 e^{\lambda t}$ with $p(0) = p_0$ (exponential growth or exponential decay)

Model of Verhulst (1 Species)

- $\gamma(t) = \gamma_0 - \gamma_1 p(t)$: *linear* birth rate per time unit
- $\delta(t) = \delta_0 + \delta_1 p(t)$: *linear* death rate per time unit
  - $\gamma_0 > \delta_0 > 0$
  - $\gamma_1, \delta_1 > 0$
- $p_\infty = \frac{\gamma_0 - \delta_0}{\gamma_1 + \delta_1}$: population limit
- **ODE**: $\dot{p}(t) = -m \cdot (p(t) - p_\infty)$ with $m = \gamma_1 + \delta_1$
- **solution**: $p(t) = p_\infty + (p_0 - p_\infty) e^{-mt}$ with $p(0) = p_0$

Logistic Growth (1 Species)

- **ODE**: $\dot{p}(t) = a \cdot p(t) - b \cdot p^2(t)$ with $p(0) = p_0$ and $a \gg b > 0$ (S-shape)
- **solution**: $p(t) = \frac{a \cdot p_0}{b \cdot p_0 + (a - b \cdot p_0) e^{-at}}$
- **limit**: $\lim_{t \to \infty} p(t) = \frac{a}{b}$
  - $p_0 < \frac{a}{b} \implies$ inflection point exists, so $p(t) < \frac{a}{b}$ for all t
  - $p_0 > \frac{a}{b} \implies$ monotonically decreasing, so $p(t) > \frac{a}{b}$ for all t

Multiple Species

- $\dot{p}(t) = f(p(t), q(t)) \cdot p(t)$
  $\dot{q}(t) = g(p(t), q(t)) \cdot q(t)$
  - f, g: growth rates
  - **equilibrium**: no more population change
    - $f(\overline{p}, \overline{q}) = g(\overline{p}, \overline{q}) = 0$
  - **stationary solution**: $p(t) = \overline{p}, q(t) = \overline{q}$
  - $f(\overline{p}, \overline{q}) = g(\overline{p}, \overline{q}) = 0 \implies \dot{p}(t) = \dot{q}(t) = 0$

Arms Race

- $x(t), y(t)$: arms expenditures
- m, n: disarmament rates
- a, b: armament rates
- c, d: constant contributions to (dis-)armament
- **ODEs**: $\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} -m & a \\ b & -n \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix}$

Equilibriums

- **attractiveness of equilibrium**: negative real parts of the eigenvalues of the Jacobian of $\begin{pmatrix} f(p,q)\cdot p \\ g(p,q)\cdot q \end{pmatrix}$
- **competition**: both species have the same characteristics and compete for the same habitat (e.g. lions vs leopards)
  - $f_p(p,q), f_q(p,q), g_p(p,q), g_q(p,q) < 0, \quad p,q > 0$
  - **linear model**: $f(p,q) = a_1 + a_2 \cdot p + a_3 \cdot q, \quad g(p,q) = a_4 + a_5 \cdot p + a_6 \cdot q$
    - $a_1, a_4 > 0$
      $a_2, a_3, a_5, a_6 < 0$
      $a_2 \cdot a_6 > a_3 \cdot a_5$
    - $\overline{p} = \frac{a_3 a_4 - a_1 a_6}{a_2 a_6 - a_3 a_5}, \quad \overline{q} = \frac{a_1 a_5 - a_4 a_2}{a_2 a_6 - a_3 a_5}$
      - equilibrium ensured if $\frac{a_2}{a_5} > \frac{a_1}{a_4} > \frac{a_3}{a_6}$
- **predator-prey**: one species is the natural predator of another (e.g. foxes vs. rabbits)
  - $f_p(p,q), g_p(p,q), g_q(p,q) < 0, \quad f_q(p,q) > 0, \quad p,q > 0$ (i.e. predator P w/ f benefits from high population of prey Q w/ q)
  - **linear model**: $f(p,q) = a_1 + a_2 \cdot p + a_3 \cdot q, \quad g(p,q) = a_4 + a_5 \cdot p + a_6 \cdot q$
    - $a_2, a_5, a_6 < 0$
      $a_3 > 0$ (for predator P and prey Q, can vary depending on who's who)
- **symboisis**: both benefit from eachother in a community (e.g. parasite vs. host)

Ordinary Differential Equations (ODEs)

- ***see NumProg script*** (machine numbers, rounding and rounding errors, condition, stability, stiffness, explicit / implicit euler, heun, runge-kutta, local / global discretization error, order of convergence, ill-conditioned and well-conditioned problems)
- **initial value problem**: value at the beginning of the considered time interval is given
  - **finite differences approximation**: $y_{k+1} = y_k + \delta t \cdot f(t_k, y_k)$
- **boundary value problem**: values on the boundary points of the interval are given
  - **Dirichlet boundary conditions**: values $y(t)$ are given
    - **finite differences approximation**: $\ddot{y} = \frac{y(t+\delta t) - 2y(t) + y(t-\delta t)}{(\delta t)^2}$
    - **central difference**: $\dot{y}(t) = \frac{y(t+\delta t) - y(t-\delta t)}{2\delta t}$
  - **Neumann boundary conditions**: values of first derivative $\dot{y}(t)$ are given

Control Engineering, Fuzzy Logic

- X: crisp set (i.e. a mathematical set in the typical sense)
- $\tilde{A}$: fuzzy set over X, characterized by a membership function

- $\mu(\cdot, X, \tilde{A}) : X \to [0, 1]$: membership function, stating the degree of membership for each element x to which it belongs to the corresponding fuzzy set
- $\text{supp}(\tilde{A}) = \{x \in X : \mu(x, X, \tilde{A}) > 0\}$: support, all elements with positive membership degree
- $\tilde{A}_\alpha = \{(x, \mu(x, X, \tilde{A}) : x \in X \wedge \mu(x, X, \tilde{A}) \geq \alpha\}$: α-level-set / -cut, all elements with degree above a certain threshold
- $\tilde{A} \uparrow \alpha$: cut fuzzy set, "cut" / limit all degrees higher than α down to α

Linguistic Variables

- **linguistic variable**: *name* v and *possible values* (linguistic terms), where each *linguistic variable* has an associated *crisp set* X and each *linguistic term* is a *fuzzy set* defined over the crisp set X (i.e. each term has a respective membership function)
  - **example**:
    - linguistic variable: "TEMPERATURE"
    - assigned crisp set: real axis (temperature values)
    - linguistic terms: "FREEZING" - "COLD" - "NEUTRAL" - "WARM" - "HOT"

Fuzzy Control System

1. **Fuzzification**: definition of linguistic variables, terms and membership functions
2. **Rule Base**: definition of IF-THEN rules
3. **Inference Operators**: transfer fuzziness of measured variables to control variables
4. **Defuzzification**: calculation of crisp control values

Heat Transfer (3D Space + Time), PDEs

- **heat equation**: $\kappa \cdot \Delta T = T_t$
  - $\kappa = \frac{k}{\rho c}$: thermic diffusion coefficient
    - k: thermal conductivity
    - ρ: density
    - c: specific heat capacity
  - $\Delta T := T_{xx} + T_{yy} + T_{zz}$: Laplace operator (sum of second derivatives)
  - *linear, second order PDE in d dimensions*
    - **linear**: only linear combinations of derivatives (i.e. no products etc.)
    - **second order**: only function, first and second derivative
    - $d = 4$: three spatial dimensions + time; $\vec{x} = (x, y, z; t)$

- $A = \begin{pmatrix} \kappa & 0 & 0 & 0 \\ 0 & \kappa & 0 & 0 \\ 0 & 0 & \kappa & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \quad , a = f = 0$

Partial Differential Equations (PDEs)

- **elliptic PDE**: matrix A is *positive or negative definite*
  - **example**: Laplace equation $\Delta u = 0$
- **hyperbolic PDE**: matrix A has *one positive* and $d - 1$ *negative eigenvalues* (or vice-versa)
  - **example**: heat equation $\Delta u = u_t$
- **parabolic PDE**: one eigenvalue of A is *zero*, all others have the *same sign*; rank of A and b together is d
  - **example**: wave equation $\Delta u = u_{tt}$
- *see slides for finite difference & finite element method...*

Solving Linear Systems

- ***see NumProg script*** (Richardson, Jacobi, Gauß-Seidel, Steepest Descent)
- **iterative methods**: $\lim_{i \to \infty} x^{(i)} = x$
  - **speed of convergence**: $||x - x^{(i+1)}|| < \gamma ||x - x^{(i)}||^s$ with $0 < \gamma < 1$ and convergence order s
- **relaxation methods / smoothers**: Richardson, Jacobi, Gauß-Seidel, SOR / damped
- $e^{(i)} = x^{(i)} - x$: *error* of current approximation
- $r^{(i)} = b - Ax^{(i)} = Ax - Ax^{(i)} = -Ae^{(i)}$: *residual* of current approximation
- $Mx^{(i+1)} + (A - M)x^{(i)} = b$: general formula
  - **solved**: $x^{(i+1)} = x^{(i)} + M^{-1}r^{(i)}$
  - $A = M + (A - M)$ with $Mx = b$ easy to solve and small diff. $A - M$ w.r.t matrix norm
  - **common**: $A = L_A + D_A + U_A$ (lower triangular, diagonal, upper triangular)
    - **Richardson**: $M = I$
    - **Jacobi**: $M = D_A$
    - **Gauß-Seidel**: $M = D_A + L_A$
    - **SOR**: $M = \frac{1}{\alpha} D_A + L_A$
- $x_k^{(i+1)} = x_k^{(i)} + \alpha y_k$
  - **damping**: multiplication of correction with a factor $0 < \alpha < 1$
    - common with Jacobi
  - **over-relaxation**: multiplication of correction with a factor $1 < \alpha < 2$

- **successive over-relaxation**: $\alpha > 1$ for Gauß-Seidel

(Data) Traffic Simulation

- T: random variable (some capital letter...)
- $p(T \leq t) = F_T(t) = \int_{-\infty}^{t} f_T(x)dx$: distribution of T
- $f_T(x)$: density of T
  - $\int_{-\infty}^{\infty} f_T(x)dx = 1$
- $E(T) = \int_{-\infty}^{\infty} t \cdot f_T(t)dt$: expectation value
- $V(T) = \sigma^2(T) = \int_{-\infty}^{\infty}(t - E(T))^2 \cdot f_T(t)dt$: variance, standard deviation
- $\varrho(T) = \frac{\sigma(T)}{E(T)}$: coefficient of variation (relative)
- $p(A|B) = \frac{p(A \cap B)}{p(B)}$: conditional probability
- $p(T \leq t + \Delta t \mid T > t) = \frac{F_T(t+\Delta t) - F_T(t)}{1 - F_T(t)}$: conditional probability
- $h_T(t) = \frac{f_T(t)}{1-F_T(t)}$: hazard rate; risk of interval end in time (e.g. a customer arrives in the next Δt seconds)
- **negative exponential distribution**: predicts time interval between two event occurences
- **Poisson distribution**: predicts number of event occurences in an interval (memoryless, i.e. independent of history)
  - ϑ: event (hazard) rate, constant
    - $h_T(t) = \vartheta = E(\frac{\#\text{ of events in } [t_1,t_2]}{t_2 - t_1})$
  - λ: arrival rate
  - μ: service rate
  - $E(T) = \frac{1}{\vartheta}$, $V(T) = \frac{1}{\vartheta^2}$, $\varrho(T) = 1$, $E(X) = \lambda$
- **Forward Recurence Time (FRT, VRZ)**: remaining time to next event
- **Backward Recurrence Time (BRT, RRZ)**: time elapsed since last event
  - $E(\mathrm{FRT}) = E(\mathrm{BRT}) = \frac{1}{2}\frac{E(T^2)}{E(T)} = \frac{E(T)}{2}(1 + \varrho^2(T))$
  - uniform distribution $[0, a]$: $E(\mathrm{FRT}) = \frac{a}{3} < E(T)$
  - Poisson process: $E(\mathrm{FRT}) = E(T)$
  - process with $\varrho(T) > 1$: $E(\mathrm{FRT}) > E(T)$ (*Hitchhiker's paradox*)

Modelling Queuing Systems

- **functional unit (FU)**: unit delimited by impact or assignment
  - **service unit (SU), instance**: functionality does not include data transport (e.g. cashier)
  - **canal (C)**: data transport between service units (e.g. CPU-RAM bus)

- **system**: service unit consisting of several service units (e.g. house of doctor's offices)
- **job**: occupies a service unit (e.g. print job)
- **dwelling time** y: *total time* that a job stays in the service unit from start to completion
  - $y = b + w$
- **capacity** k: maximum number of jobs a service unit can process at the same time
  - $k = 1$: *simple* service unit (e.g. portaloo)
- **filling** f: number of jobs in a service unit
  - $f = 0$: *empty* service unit
  - $f > 0$: *busy* service unit
  - $f = k$: *occupied* service unit#
  - $f = 1$:
    - $E(Y_S) = E(B_S), \;\; E(D_S) = \frac{1}{E(B_S)}$
  - $\frac{f}{k}$: *relative* filling
  - **closed queuing network**: constant filling
  - **open queuing network**: variable filling
- **throughput** d: average number of jobs completed within a time interval of a service unit
- **maximum throughput** c: maximum possible throughput
- **utilization** $r = \frac{d}{c}$: relative throughput
- **service time** b: net dwelling time of a job (i.e. dwelling time without waiting time or dwelling time with $f = 1$)
- **waiting time** w: waiting time of a job
- **Little's Law**: $E(F) = E(D) \cdot E(Y)$ (average filling is throughput times average dwelling time)
  - $k = 1 \implies E(F) = E(D) \cdot E(B) = \frac{E(D)}{c} = E(R)$
- **queuing system (QS)**: system in which jobs can't get lost (i.e. if service unit is occupied, job waits in canals)
  - **elementary queuing system (eIQS)**
    - one canal as waiting pool with capacity k_{WP}
    - one service unit with capacity k_U *or* k_U simple service units
- **inter-arrival time**: timespan between two consecutive arrival events
  - **D**: *deterministic* / *constant*
  - **M**: Markovian; *negative exponentially* distributed w/ distribution $1 - e^{-\lambda t}$ and expected value $\frac{1}{\lambda}$
  - **G**: general, *arbitrarily* distributed
- **Kendall notation**: $\text{arrival process} \mid \text{service process} \mid n = k_{SU} \mid k_{WP} \mid N \mid D$

- $M|M|1$: arrival and service process negative exponentially distributed, one service unit
  - $E(F) = \frac{r}{1-r}$
  - $E(D) = \lambda$
  - $E(B) = \frac{1}{\mu}$
  - $E(Y) = \frac{E(B)}{1-r}$
  - $E(W) = \frac{r}{1-r} \cdot E(B)$
- $M|G|1$: arrival process negative exponentially distributed, service process arbitrarily distributed, one service unit

- **task stream**: arrival process + type of task (e.g. $(M, \text{type } A)$)
- **task load**: set of task streams
- $\nu_i = \frac{E(D_i)}{E(D_S)}$: number of visits of vertex i (ratio between throughput at i and throughput of network)
- **traffic bottleneck**: vertex / vertices with maximum utilization $E(R_i)$ (max. 1)
  - $E(R_{TB}) = \max_i E(R_i) = \max_i \frac{E(D_i)}{c_i}$
  - $c_S = \frac{c_{TB}}{\nu_{TB}} = \min_i \frac{c_i}{\nu_i}$

Stochastic Processes

- **stochastic process**: random variable X depending on time t
  - **continuous process**: $X(t) \in \mathbb{R}$
  - **discrete process**: $X(t) \in \mathbb{N}, \mathbb{Z}$
  - **process in continuous time**: $t \in \mathbb{R}$
  - **process in discrete time**: t countable
- **state space**: set of possible values of X
- **stationary process**: distribution stays the same in time, $E(X(t)) = E(X)$ (as such, no absorbing states)
- **independent process**: not depending on former results, $p(X(t_2) \leq x_2 \mid X(t_1) = x_1) = p(X(t_2) \leq x_2)$
- **Markov process**: new state depends only on current state and not previous history
- **homogeneous Markov process (HMP)**: transition probabilities are constant, $p(X(t_{n+1}) \leq x_{n+1} \mid X(t_n) = x_n)$ (here discrete)
  - **irreducible**: all states are reachable from each other
    - all states are either transient or positively recurrent or null recurrent
    - if one state is periodic, all states are with same period length
  - **closed subset of states**: impossible to escape in complementary set
  - **absorbing state**: closed subset consisting of only one state

    - **recurrence probability**: $f_i = p(X(t_{n+k}) = i, k \geq 1 \mid X(t_n) = i)$, probability that sometime in the future, we revisit node i given we're currently in i
    - **recurrence time k or K**
    - **recurrent state**: $f_i = 1$ (i.e. we'll surely come back)
    - **periodic**: $k \in \{j \cdot k_0 \mid j, k_0 \in \mathbb{N}\}$ (i.e. happens every k units of time)
    - **positively recurrent**: $f_i = 1, E(K) < \infty$ (you'll be back in finite time)
    - **null recurrent**: $f_i = 1, E(K) = \infty$ (you would have to wait an infinite amount of time)
    - **transient**: $f_1 < 1$ (uncertain recurrence)
    - **state probabilities**: $p_i(t_k) = \sum_{j=1}^{n} p_j(t_{k-1}) \cdot p_{ji}$ (sum of probability of state j in last time step times transition probability from j to i) with $\sum_{i=1}^{n} p_i(t_k) = 1$
        - irreducible, aperiodic $\implies \exists \lim_{k \to \infty} p_i(t_k) = p_i$
            - transient / null rec.: $p_i = 0$
            - pos. recurrent: $p_i > 0$
    - **transition probabilities**: p_{ij} (discrete time)
    - **transition rates**: λ_{ij} (continuous time)
- **homogeneous birth-death process (HBDP)**: Markov chain where state values may only change in in- / decrements of 1 (as such, only one succesor state)

Traffic Flow

- **Euler description**: outside observer
- **Lagrange description**: perspective of participant
- l: size of car (all cars same size)
- v: **traffic velocity**, tempo of the cars in km/h
    - $v = v_{\max} \cdot \left(1 - \frac{\varrho}{\varrho_{\max}}\right)$
    - $v \to 0$ for $\varrho \to \varrho_{\max} = \frac{1}{l}$
    - **linear**: $v \to v_{\max}$ for $\varrho \to 0$
- ϱ: **traffic density**, number of cars per section in cars/km
- f: **traffic flow**, throughput at a checkpoint in cars/h
    - $f = \varrho \cdot v$
    - $f = f(\varrho) = v(\varrho) \cdot \varrho$
    - **parabolic**: $f(\varrho) = v_{\max} \cdot \varrho \cdot \left(1 - \frac{\varrho}{\varrho_{\max}}\right)$
        - $f \to 0$ for $\varrho \to \varrho_{\max}, \varrho \to 0$
    - **cubic**: $f(\varrho) = v_{\max} \cdot \varrho \cdot \left(1 - \alpha\varrho - \beta\varrho^2\right)$
        - $f \to 0$ for $\varrho \to \varrho_{\max}, \varrho \to 0$

        - $f' \to v_{\max}$ for $\varrho \to 0$
        - $v(\varrho_{\max}) = 0$
        - $f'(\varrho_{\text{opt}}) = 0$ or $f_{\max}$
    - **cubic, better**: $f(\varrho) = v_{\max} \cdot \varrho \cdot \left(1 - \left(\frac{\varrho}{\varrho_{\max}}\right)^{\alpha}\right)^{\beta}$
- $d = \frac{1-\varrho \cdot l}{\varrho - 1}$: distance of two cars (road length normalized to 1)

Unsteady Situations

- $\varrho(x,t)$: density at location x at time t
- $f(x,t)$: flow at location x at time t
    - f_ϱ: signal velocity (i.e. propagation of information of a change or disturbance); always less or equal to car velocity
        - $\varrho \to 0, \quad v \to v_{\max}, \quad f \to 0, \quad f_\varrho = v$
    - $\varrho_t(x,t) + f_x(x,t) = 0$
    - $\varrho_t(x,t) + f_\varrho(\varrho(x,t)) \cdot \varrho_x(x,t) = 0$
- $n(t) = \int_{x=a}^{x=b} \varrho(x,t) dx$: number of cars on the track at time t
    - $n_t(t) = f(a,t) - f(b,t) = -\int_a^b f_x(x,t) dx$
- $x = a$ entrance, $x = b$ exit