TUM ModSim, SoSe 2023

Mitschriften basierend auf der Vorlesung von Dr. Hans-Joachim Bungartz

Zuletzt aktualisiert: 21. Juli 2023

Introduction

About

Hier sind die wichtigsten Konzepte der ModSim Vorlesung von Dr. Hans-Joachim Bungartz im Sommersemester 2023 zusammengefasst.

Die Mitschriften selbst sind in Markdown geschrieben und werden mithilfe einer GitHub-Action nach jedem Push mithilfe von Pandoc zu einem PDF konvertiert.t

Eine stets aktuelle Version der PDFs kann über modsim_SS23_IN2010_merge.pdf heruntergeladen werden.

Implementation

Außerdem befindet sich eine Implementation von verschiedenen Algorithmen im Ordner `/algorithms` auf GitHub. Diese sind in Python und unter der Verwendung von NumPy geschrieben.

How to Contribute

1. Fork this Repository
2. Commit and push your changes to **your** forked repository
3. Open a Pull Request to this repository
4. Wait until the changes are merged

Contributors

Inhaltsverzeichnis

Focus Analysis / Calculus

Foundations

Functions and their representations

- One-Dimensional
$$f : D \subseteq \mathbb{R}^n \to \mathbb{R}^m, x \mapsto f(x)$$
- Multidimensional
$$f : D \subseteq \mathbb{R} \to \mathbb{R}, x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto f(x) = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix}$$

Names for special types of functions

- Curves: $n = 1$ and $m \in \mathbb{N}$
  - plane curves (2D): $n = 1$ and $m = 2$
  - space curves (3D): $n = 1$ and $m = 3$
- Surfaces: $n = 2$ and $m = 3$
- Scalar fields: $n \in \mathbb{N}$ and $m = 1$
- Vector fields: $n = m$

Topology concepts in higher dimensions

There is an analogous concept to open and closed intervals in multi-dimensional spaces.
Given a domain $D \subseteq \mathbb{R}^n$ and its complement $D^c = \mathbb{R}^n \setminus D$

- A point x is called *inner point* if there exists an arbitrarily small ball around this points that fullly lies inside D.
- The set of all inner points of D is called the *interior* of D and is denoted as $\mathring{D}$.
- The domain is called open if $D = \mathring{D}$
- A point $x_0 \in \mathbb{R}^n$ is called *boundary point* if any arbitrarily small ball around this point intersects with both D and its complement D^c
- The set of all boundary points of D is called the *boundary* of D, denoted ∂D
- The set $\bar{D} = D \cup 0\partial D$ is called the *closure* of D

Using these definitions there are multiple attributes assignable to domains.
A domain D is called:

- *closed* if $\partial D \subseteq D$, i.e. $\bar{D} = D$
- *bounded* if $\exists K \in \mathbb{R} : ||x|| < K, \forall x \in D$
- *compact* if it is closed and bounded
- *convex* if all points on a straight line between to points in D are themselves element of D

Continuity

We define continuity in multi-dimensional spaces using converging vector sequences.
A sequence $(x^{(k)})$ converges to the limit x if
$$\lim_{k \to \infty} ||x^{(k)} - x|| = 0$$

Converges of a vector sequence is also equivalent to the convergence of all components.
A vector function is then called continuous at $a \in D$ if for all sequences $(x^{(k)})_{k\in\mathbb{N}_0}$ in D converging to a the corresponding sequence $(f(x^{(k)}))_{k\in\mathbb{N}_0}$ in $\mathbb{R}^m$ converges to $f(a)$ and continuous on D if this holds for all points $a \in D$

Partial Differentiation

Gradient

The Gradient of a function gives the direction of the steepest ascent of the function. It requires that f represents a scalar field.

When applying the limit definition of the derivative to a function in higher dimesions it is not clear from which direction the derivative should be taken.

Using

$$\frac{\partial f}{\partial v}(a) = \lim_{h\to 0} \frac{f(a+hv) - f(a)}{h}$$

we can define the directional derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ along a vector $v \in \mathbb{R}^n$ at a point $a \in \mathbb{R}^n$.

If we use the coordinate vectors e_i as basis vectors for $\mathbb{R}^n$ we can define the *Gradient* of f at a as

$$\nabla f(a) = \text{grad} f(a) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(a) \\ \vdots \\ \frac{\partial f}{\partial x_n}(a) \end{pmatrix}$$

For continuous functions the directional derivative at the point a along a vector v can be computed as

$$\frac{\partial f}{\partial v}(a) = \langle \nabla f(a), v \rangle$$

Example:

$$f(x,y) = x^2 + y^2 \to \nabla f(a) = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

Hessian Matrix

The Hessian matrix of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $a \in \mathbb{R}^n$ is the matrix of all second partial derivatives of f at a.

$$H_f(a) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(a) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(a) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(a) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(a) & \frac{\partial^2 f}{\partial x_2^2}(a) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(a) & \frac{\partial^2 f}{\partial x_n \partial x_2}(a) & \dots & \frac{\partial^2 f}{\partial x_n^2}(a) \end{pmatrix}$$

Example:

$$f(x,y) = x^2 + y^2 \to H_f(a) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Jacobian Matrix

The Jacobian matrix of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ at a point $a \in \mathbb{R}^n$ is the matrix of all partial derivatives of f at a.

In contrast to the Gradient the Jacobian matrix works for vector fields. It gives an analogue to the gradient for vector fields.

$$Df(a) = J_f(a) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(a) & \frac{\partial f_1}{\partial x_2}(a) & \dots & \frac{\partial f_1}{\partial x_n}(a) \\ \frac{\partial f_2}{\partial x_1}(a) & \frac{\partial f_2}{\partial x_2}(a) & \dots & \frac{\partial f_2}{\partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a) & \frac{\partial f_m}{\partial x_2}(a) & \dots & \frac{\partial f_m}{\partial x_n}(a) \end{pmatrix} = \begin{pmatrix} \nabla f_1(a)^T \\ \nabla f_2(a)^T \\ \dots \\ \nabla f_m(a)^T \end{pmatrix}$$

Example:

$$f(x,y) = \begin{pmatrix} x^2 + y \\ x^2 + y^2 \end{pmatrix} \to J_f(a) = \begin{pmatrix} 2x & 1 \\ 2x & 2y \end{pmatrix}$$

Calculation rules for the Jacobian

- Addition rule: $J(f + g) = J_f + J_g$
- Homogeneous rule: $J(cf) = cJ_f$
- Product rule: $J(f^T \cdot g) = f(x)^T J_g(x) + g(x)^T J_f(x)$

Laplace Operator

The Laplace operator is a second order partial derivative operator. It is defined on Scalar fields and is used to compute the rate of change of a scalar field.

$$\Delta f = \nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

Example:

$$f(x,y) = x^2 + y^2 \to \Delta f(a) = 2 + 2 = 4$$

Divergence

The Divergence of a vector field is the rate of shrinkage or expansion around a point. It is defined as the sum of the partial derivatives of the components of the vector field.

$$\mathrm{div} f = \sum_{i=1}^{n} \frac{\partial f_i}{\partial x_i} = \nabla \cdot f$$

Example:

$$f(x,y) = \begin{pmatrix} x^2 \\ y^2 \end{pmatrix} \to \mathrm{div} f(a) = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} = 2x + 2y$$

Curl / Rotation

The Curl of a vector field is the rate of rotation around a point. It is defined as the cross product of the partial derivatives of the components of the vector field.

$$\mathrm{rot} f = \nabla \times f = \begin{pmatrix} \frac{\partial f_3}{\partial x_2} - \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_1}{\partial x_3} - \frac{\partial f_3}{\partial x_1} \\ \frac{\partial f_2}{\partial x_1} - \frac{\partial f_1}{\partial x_2} \end{pmatrix}$$

Example:

$$f(x,y,z) = \begin{pmatrix} x^2 y \\ y^2 x \\ yz \end{pmatrix} \to \mathrm{rot} f(a) = \begin{pmatrix} z \\ 0 \\ y^2 - x^2 \end{pmatrix}$$

Taylor Expansion

It is also possible to approximate functions of multiple variables by Taylor expansions, by using the analog for higher order derivatives for functions of multiple variables.

Coordinate Transformations

A bijection between two coordinate systems is called a coordinate transformation. It is a function $\phi : \mathbb{R}^n \to \mathbb{R}^m$ that maps points in one coordinate system to points in another coordinate system and vice versa.

Jacobian Matrix of a Coordinate Transformation

The Jacobian matrix of this transformation is called *coordinate transformation matrix* and is defined as the matrix of all partial derivatives of the coordinate transformation.

Its determinant is called the *Jacobian determinant*.

Example:

We define the Transformation ϕ from polar coordinates to cartesian coordinates as follows:

$$\begin{aligned} \phi(r,\phi) = \begin{pmatrix} r\cos\phi \\ r\sin\phi \end{pmatrix} &:= \begin{pmatrix} x \\ y \end{pmatrix} \\ \implies J_\phi(r,\phi) &= \begin{pmatrix} \cos\phi & -r\sin\phi \\ \sin\phi & r\cos\phi \end{pmatrix} \\ \implies \det J_\phi(r,\phi) &= r \end{aligned}$$

Roots and Optima

Newton's Method

Newtons method in higher dimensions works similar to the one dimensional case. The only difference is that we have to compute the gradient and the Hessian matrix instead of the derivative.

1D Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Higher Dimensional Newton's Method

$$\mathbf{x_{n+1}} = \mathbf{x_n} - (\mathbf{J_f}(\mathbf{x_n}))^{-1} \cdot \mathbf{f}(\mathbf{x_n})$$

It konverges quadratically to a root in the neighborhood of the initial guess.

Optima

If f is a scalar field and $\mathbf{x_0}$ is a point in the domain of f:

- $\mathbf{x_0}$ is a global maximum if $f(\mathbf{x_0}) \geq \mathbf{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{D}$.
- $\mathbf{x_0}$ is a global minimum if $f(\mathbf{x_0}) \leq \mathbf{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{D}$.
- $\mathbf{x_0}$ is a local maximum if $f(\mathbf{x_0}) \geq \mathbf{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{B}_\epsilon(\mathbf{x_0})$.
- $\mathbf{x_0}$ is a local minimum if $f(\mathbf{x_0}) \leq \mathbf{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{B}_\epsilon(\mathbf{x_0})$.

All local extrema can be found from critical-points ($\nabla f(\mathbf{x}) = \mathbf{0}$) or at the boundary points.

But analog to the one-dimensional case, its also possible that a critical-point is neither a minum or a maximum point, because it is a "Saddle-Point"

Criteria

A critical Point $\mathbf{x_0} \in \mathbf{D}$ is:

- A local maximum if $H_f(\mathbf{x_0})$ is negative definite
- A local minum if $H_f(\mathbf{x_0})$ is positive definite
- A saddle point if $H_f(\mathbf{x_0})$ is indefiite

Curves and Surfaces

Curve

A curve is a mapping $\gamma : I \subseteq \mathbb{R} \to \mathbb{R}^n$. In other words it maps a line into higher dimensional space.

Example:

$$\gamma(t) = \begin{pmatrix} \cos(2t) \\ \sin(t) \end{pmatrix}$$

Surface

A surface is a mapping $\gamma : I \subseteq \mathbb{R}^2 \to \mathbb{R}^n$. In other words it maps a 2d-surface into higher dimensional space.

Quadrature

Integral over rectangular domains

If $f : [a, b] \times [c, d] \to \mathbb{R}$ is a scalar field, the "sum" of scalar values along the area of the Domain can be computed as:

$$\int \int_D f(x, y)dxdy = \int_a^d \int_c^d f(x, y)dxdy$$

Integration over simple domains

A 2d-standard Domain is defined as $D = \{(x, y)|x \in [a, b], l(x) \leq y \leq u(x)\}$. This means such a domain has one-varing paramater, which determines the lower- and upper bound of the surface at that point.

We can integrate over such domains using:

$$\int \int_D f(x, y)dxdy = \int_{x=a}^b \int_{l(x)}^{u(x)} f(x, y)dxdy$$

Integration under coordinate transformations

If $B, D \subseteq \mathbb{R}^n$ are Domains and $\phi : B \to D$ is a transformation between these domains. The Domain-Integral can be rewritten as:

$$\int_D f(x_1, \ldots, x_n)dx_1 \cdots dx_2 = \int_B f(\phi(t_1, \ldots t_n)) \cdot |\det(J_\phi(t_1, \ldots t_n))| \; dt_1 \cdots dt_2$$

Partial differential equations

A PDE is a differential equation with multiple changing, variables.

Examples

- First order:
  - Traffic flow: $u_t + vu_x = f(x,t)$, where v is velocity, t is time and x is a point along the road. The resulting $u(t,x)$ describes the cars at point x and time t
- Second order:
  - Heat equation: $u_t t - c^2 \Delta u = 0$
  - Laplace equaiton: $-\Delta u = 0$

Introduction to Mathematical Modeling

Terminology

Model

A model is a simplified image of a partial reality.

- Practical Models
  - Wind tunnel
- Scale model
- Abstract Models
  - Mathematical models

Derivation

Questions to ask when modeling:

- What exactly should be modeled?
  - Population growth
  - Rocket trajectory
- Which attributes play a role in the model?
  - Population size, children per family, death rate...
  - Rocket mass, thrust, air resistance...
- What relations exist between the attributes?
  - Population growth is proportional to the population size
  - Rocket thrust is proportional to the fuel consumption
- What mathematical tools are needed to describe the relations?
  - Differential equations, probability theory, statistics, algebraic equations and inequalities, automata theory, graph theory, etc.

Simulation Tasks

- What is the goal of the simulation?
  - Find an arbitrary solution
  - Find the only solution
  - Show that a solution exists
  - Solve a constrained optimization problem
  - Find a critical point

Analysis of the Model

- What is the behavior of the model?
  - Is the model stable?
  - Does it converge to a steady state?
  - Is the solution point, really an optimum?
  - Is the solution unique or do exist better solutions?

Problems are **well-posed** if the following conditions are met:

- The solution exists
- The solution is unique
- It is stable

Aplicability of the Model

- Is enough input data available, to run the model?
- Is the hardware available to run the model?
- Is it fast enough, to be useful?
- Is it sensitive to small changes in the input data?

Mathematical Modeling

Process of formal derivation and analysis of mathematical models.

1. Informal description of the problem
2. Semi-formal description of the problem, using tools of the specific discipline
3. A strict formal description of the problem, (consistent)

Simulation

A virtual, computer based experiment with a mathematical model.

The goals of simulation are:

- To understand the behavior of the system
  - Why earthquakes occur
  - Why buildings collapse
- To optimize a system
  - Better flight schedules
  - higher throughput
- To predict the behavior of the system
  - Climate change
  - Characteristics of a new drug

Simulation Pipeline

1. Modeling the system
2. Numerical methods needed to solve the model
3. Implementation of the numerical methods in an efficient way
4. Visualization of the results
5. Validation of the model
6. Embedding the model in a larger system i.e a wheater forecast

Discrete Modeling and Simulation

Decision Model

Decisions can be made based of:

- **certanty**: all the information is known
- **risk**: the probability of each outcome is known
- **uncertainty**: the probability of each outcome is not known

Model

This way of making decision can be modeled using a Payoff Matrix.

	State 1	State 2	$\cdots$	State n
Action 1	a_{11}	a_{12}	$\cdots$	a_{1n}
Action 2	a_{21}	a_{22}	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Action m	a_{m1}	a_{m2}	$\cdots$	a_{mn}

The **payoff** is the value of the Action based on the State.

Example Payoff Matrix - Prisoner's Dilemma

If both players cooperate, they will both get a sentence of 5 years in prison. If one of them cooperates and the other doesn't, the one that cooperates will be free and the other will get a sentence of 10 years in prison. If both of them don't cooperate, they will both get a sentence of 1 year in prison.

(Reward A, Reward B)	Player A cooperates	Player A doesn't cooperate
Player B cooperates	$-5, -5$	$-10, 0$
Player B doesn't cooperate	$0, -10$	$-1, -1$

Startegies

- **Certainity**: if you are certain about the initial state, your action will simply be the one that gives you the highest payoff.
  1. **Maximum**
     - Find Action i as $\arg\max_i N_{ij}$. In other words, find the action that gives you the highest payoff.
       * In the example above, if you would somehow know that Player A will cooperate, it is of your best interest to also cooperate. Since this will reduce your sentence from 10 years to 5 years.
- **Risk**: There are multiple Strategies you can take:
  1. **Caution - (max-min payoff)**
     - Find Action i as $\arg\max_i \min_j N_{ij}$. In other words, find the action that gives you the highest minimum payoff.

    - In the example above, Player B would choose to cooperate, since chosing this way, will guarantee him a sentence of maximum 5 years.
2. **Full Risk - (max-max payoff)**
   - Find Action i as $\arg\max_i \max_j N_{ij}$. In other words, find the action that gives you the highest maximum payoff.
     - In the example above, Player B would choose to cooperate, since the best case scenario for him is to get a sentence of 0 years in prison, if Player A doesn't cooperate.
3. **Alternative Caution** - (min-max payoff)
   - Find Action i as $\arg\min_i \max_j R_{ij}$. Where $R_{ij} = \max_i N_{ij} - N_{ij}$.
4. **Pessimism-Optimism**
   - Let $m_i = \min_j N_{ij}$ and $M_i = \max_j N_{ij}$, and $\alpha \in [0, 1]$.
   - Find Action i as $\arg\max_i(\alpha m_i + (1-\alpha)M_i)$.
   - This allows you to choose how much risk you want to take. If $\alpha = 0$, you will choose the action that gives you the highest maximum payoff. If $\alpha = 1$, you will choose the action that gives you the highest minimum payoff.

Two-Player Zero-Sum Games

Idea: Both players choose their actions simultaneously.

Assumptions:

- Both players know the payoff matrix.
- Both players act with caution. If they alternate turns:
  - S1 tries to maximize the minimum payoff.
  - S2 tries to minimize the maximum loss.

Equilibrium

If you find a state where both players are happy with their payoff, you have found an equilibrium. No player will want to change their action.

$$\max_i \min_j N_{ij} = a_{\hat{i}\hat{j}} = \min_j \max_i N_{ij}$$

Group Decision Making

Question: How to combine decisions from multiple people in a **democtratic** way, to find the best collective decision?

- A is the set of all candidates.
- $r : A \to \mathbb{R}$ is the ranking function.
  - Each voter i orders the candidates to his preference with numbers from 1 to $|A|$, where a lower number means a higher preference. He does this by defining a ranking function r_i.
- The set $P_A = \{\rho \subset A \times A | \rho$ is a transitive, and asymetric relation$\}$ is the set of all possible rankings.
  - This means in P_A a voter cant have equal preferences, between two candidates.

A colletive choice function $K : P_A^{\otimes n} \to P_A$ is a function that takes all n rankings from the voters and returns a combined ranking, which is the collective decision.

Majority Decision (Condorcet Method)

For each voter let $N(x, y)$ be the number of voters that prefer x over y.

	$r_i(x)$	$r_i(y)$	$r_i(z)$
$i = 1$	1	2	3
$i = 2$	3	1	2
$i = 3$	2	3	1

Therefore:

$N(a,b)$	x	y	z
x	0	2	1
y	1	0	2
z	2	1	0

Define:

- $a\rho b$ if $N(a,b) > N(b,a)$

It follows:

- $x\rho y$ because $N(x,y) > N(y,x)$
- $y\rho z$ because $N(y,z) > N(z,y)$
- $z\rho x$ because $N(z,x) > N(x,z)$

So: $x >_\rho y >_\rho z >_\rho x$. This is a cycle, and therefore there we cannot find a collective decision.

This is bad, because even tough even tough every voter had a preference, we couldn't find a collective decision.

This violates rule 2 of a democratic decision. Because sometimes we cannot find a collective decision.

Rank Addition

This method simply adds the rankings of all voters together, and finds the collective decision from that.

Define $x\rho y$ if $\sum_{i=1}^{n} r_i(x) < \sum_{i=1}^{n} r_i(y)$.

	$r_i(x)$	$r_i(y)$	$r_i(z)$
$i=1$	1	2	3
$i=2$	2	1	3
Σr_i	3	3	6

So by this method x and y are equally good, and z is the worst. This means we cannot find a unnique collective decision.

But this method yields another problem:

If we vote again:

	$r_i(x)$	$r_i(y)$	$r_i(z)$
$i=1$	1	2	3
$i=2$	3	1	2
Σr_i	4	3	5

We get a different result. This times y is the clear winner. But every candidate still has the same preference between x and y but this time y is the winner.

This is bad, because the collective decision should not change if the relative order between candidates doesn't change.

This violates rule 4 of a democratic decision.

Rules of Democraty

1. For every set of Ballots, it must be possible to find a collective decision. (K is defined for all P_A^n)
2. The result of the collective decision must be included in A. No external candidate can win.
3. If all voters decide unanimously, the collective decision must be the same. (Pareto Principle)
4. Ballots with the same order of candidates for every pair (x,y) must yield the same collective decision. (Independence of Irrelevant Alternatives)

5. No voter can always determine the collective decision. (Non-Dictatorship)$$

Arrow's Theorem

If $|A| > 2$, and more than 1 voter, then there is no collective choice function K that satisfies all 5 rules of democracy.

Scheduling

Problem:

- A Process consists of n tasks $A_1, A_2, \ldots, A_n$.
- There exist machines $M_1, M_2, \ldots, M_m$.
- Each task A_i needs a machine M_j to be executed. With an execution time of $t_i^{(j)}$

The goal is to find a schedule, that minimizes the total execution time.

Example: Process Scheduling

- Execution times is purely task-dependent.
- Start time s_i, completion time $c_i = s_i + t_i$.
- $A_i \to A_j$ means that A_i must be executed before A_j.
  - There might be no cycle in this graph.

Algorithm

Idea: Start the tasks as soon as possible.

```
add_initial_vertex()
#forward
while there are vertices left:
    find a vertex where all predecessors are already scheduled
    mark this vertex as scheduled
    set its start time as s=max(c of all predecessors)
    set its completion time as c=s+t
#backward
while there are vertices left:
    find a vertex where all successors are already processed
    mark this vertex as processed
    set its latest_end_time as l=min(s of all successors)
    set its latest_start time as latest_s=l-s

#critical path
critical_path=[]
for all vertices:
    if s==latest_s:
        add vertex to critical path
```

Job Shop Scheduling

- Every Job A_i consists of mutiple subjobs $A_{i1}, A_{i2}, \ldots, A_{ik}$.
  - Each subjob needs a machine M_j to be executed.
  - No circulation. Every Job requires M_j at most once. # Continuos Modeling

Population Dynamics

Model of Malthus

- There exists only one species P

- constant birth rate γ
- constant death rate δ
  - constant growth rate $\lambda = \gamma - \delta$

This leads to the following equation:

$$p(t + \Delta t) = p(t) + \lambda p(t)\Delta t$$

in the limit $\Delta t \to 0$:

$$\dot{p}(t) = \lambda p(t)$$

This has the solution:

$$p(t) = p_0 e^{\lambda t}$$

Verhulst Model

- The model of Malthus is not realistic, since the population cannot grow indefinitely
- At some point the population will be saturated
- Ideas:
  - The population growth rate is proportional to the population size
  - linear birth rate $\gamma(t) = \gamma_0 - \gamma_1 p(t)$
    - Larger population size leads to smaller birth rate
  - linear death rate $\delta(t) = \delta_0 + \delta_1 p(t)$
    - Larger population size leads to larger death rate

This leads to the following equation:

$$\dot{p}(t) = \gamma(t) - \delta(t) = \gamma_0 - \gamma_1 p(t) - \delta_0 - \delta_1 p(t) = -m \cdot (p(t) - p_\infty)$$

where $m = \gamma_1 + \delta_1$ and $p_\infty = \frac{\gamma_0 + \delta_0}{m}$

This has the solution:

$$p(t) = p_\infty + (p_0 - p_\infty)e^{-mt}$$

This models starts to saturate right at the beginning, meaning that the growth rate shrinks right from the start. This is not realistic, since the population needs some time to run into resource limitations.

Logistic Model

- The model of Verhulst is not realistic, since the growth rate shrinks right from the start
- When using a quadratic term, it allows for an inflection point

$$\dot{p}(t) = (a - bp(t))p(t) = ap(t) - bp^2(t)$$

This has the solution:

$$p(t) = \frac{ap_0}{bp_0 + (a - bp_0)e^{-at}}$$

If $p_0 < \frac{a}{b}$, then the population will grow exponentially until it reaches the inflection point.Then it will start to saturate.

If $a >> b$, then the the quadratic term will kick in very late, meaning that the population will grow exponentially for a long time.

Oscillations

- The logistic model does not allow for oscillations

$$\ddot{p}(t) + \mu\dot{p}(t) + \omega^2(p(t) - p_\infty) = 0$$

where μ is the damping factor and ω_0 is the natural frequency.

This has the solution:

$$p(t) = (p_0 - p_\infty)e^{-\frac{\mu}{2}t}\cos(\sqrt{\omega^2 - \frac{\mu^2}{4}}t) + p_\infty$$

Lotka-Volterra Model

- Model of two species P and Q
- P is the prey and Q is the predator

$$\dot{p}(t) = f(p(t), q(t)) \cdot p(t)$$
$$\dot{q}(t) = g(p(t), q(t)) \cdot p(t)$$

If both $\dot{p}(t)$ and $\dot{q}(t)$ are zero, then the system is in equilibrium. This means that the population sizes are constant from that point on.

$$\begin{bmatrix}\dot{p}(t)\\ \dot{q}(t)\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

The values at this point are called fixed points.

Attractive Equilibrium Points

- The fixed point is called attractive, if the system will converge to this point, if it starts close to it

$$F(p,q) = \begin{bmatrix}f(p,q)\cdot p\\ g(p,q)\cdot q\end{bmatrix}$$

The Jacobian matrix is defined as:

$$J_F = \begin{bmatrix}\frac{\partial f}{\partial p} & \frac{\partial f}{\partial q}\\ \frac{\partial g}{\partial p} & \frac{\partial g}{\partial q}\end{bmatrix}$$

If we look at the eigenvalues of the Jacobi Matrix at the fixed point $[\bar{p}, \bar{q}]^T$ and their real parts are both negative, then the fixed point is attractive.

Numerical ODE Solvers

There exist two problem categories:

- Initial value problem
  - The initial value is given
  - The goal is to find the solution at a later time $p(0) = p_0$
- Boundary value problem
  - Both the initial and the final value are given
  - The goal is to find the solution in between $p(0) = p_0$ and $p(T) = p_T$

Definitions

- Local error
  - Maximum error between the exact solution and the numerical solution after one step
  - $e_{loc} = \max_{n \in [0, N-1]} |\frac{f(t_{n+1}) - f(t_n)}{\Delta t} - \dot{y}(t_n, y(t_n))|$
- Global error
  - Maximum error accumulated over all steps
  - $e_{glob} = \max_{n=0,\dots,N} |y_n - f(t_n)|$
- Consistency
  - The method is consistent, if the local error goes to zero, if the step size goes to zero
  - $\lim_{\Delta t \to 0} e_{loc} = 0$
- Convergence
  - The method is convergent, if the global error goes to zero, if the step size goes to zero
    - $\lim_{\Delta t \to 0} e_{glob} = 0$
  - Convergence is stronger than consistency
  - Convergence = Consistency + Stability
- Condition
  - Property of the problem
  - Measure for sensitifity of the solution to small changes in the initial value
  - Does not depend on the algorithm

$$\kappa_{rel} = \frac{|x \cdot \dot{y}(t, x)|}{|y(t, x)|}$$

- Stability
  - Property of the algorithm
  - Measure for the sensitivity of the solution to small changes in the initial value
  - Depends on the algorithm (ϵ-stability)
  - Sometimes implcit methods are more stable than explicit methods
- Stiffness
  - A problem is stiff, when the solution of a Differential Equation only converges, if the step size is very small
  - Even if the method is consistent and stable, the solution will not converge, if the step size is too large
  - Solution: Implicit methods

Euler Method

$$y_{n+1} = y_n + \Delta t \cdot f(t_n, y_n)$$

- Consistency
  - $e_{loc}(\Delta t) = \mathcal{O}(\Delta t)$
- Convergence
  - $e_{glob}(\Delta t) = \mathcal{O}(\Delta t)$

Heun's Method (Runge-Kutta 2)

$$y_{n+1} = y_n + \frac{\Delta t}{2} \cdot (f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t \cdot f(t_n, y_n)))$$

- Consistency
  - $e_{loc}(\Delta t) = \mathcal{O}(\Delta t^2)$
- Convergence
  - $e_{glob}(\Delta t) = \mathcal{O}(\Delta t^2)$

Multi-Step Methods

- Since Runge-Kutta methods are costly to compute, we can use multi-step methods, those method use previous steps to compute the next step

Adams-Bashforth Method

$$y_{n+1} = y_n + \frac{\Delta t}{2} \cdot (3f(t_n, y_n) - f(t_{n-1}, y_{n-1}))$$

- Since this method needs two previous steps, we need to use a different method for the first step
  - Calculate the first step with the Euler method with a small step size

Implicit Methods

- Implicit methods are more stable than explicit methods

Implicit Euler Method

$$y_{n+1} = y_n + \Delta t \cdot f(t_{n+1}, y_{n+1})$$

You need to solve this equation for y_{n+1} and find a solution for y_{n+1}.

Predictor-Corrector Methods

- Predictor
  - Use an explicit method to predict the next step
- Corrector
  - Use an implicit method to correct the prediction

$$y_{n+1}^{pred} = y_n + \Delta t \cdot f(t_n, y_n)$$
$$y_{n+1} = y_n + \frac{\Delta t}{2} \cdot (f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{pred}))$$

Higher Order ODEs

- Higher order Derivatives can be transformed into a system of first order ODEs
- Introduce new Variables $y_1 = y, y_2 = \dot{y}, y_3 = \ddot{y}, \dots$
- Transform the ODE $y^{(n)} = f(t, y, y', y'', \dots, y^{(n-1)})$ into a system of first order ODEs

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\vdots \\ y_{n-1}' &= y_n \\ y_n' &= f(t, y_1, y_2, \dots, y_n) \end{aligned}$$

Boundary Value Problems

- Initial value problems are easy to solve, but boundary value problems are hard to solve

$$\ddot{y} = b \cdot y + c$$

Finite Difference Method

- Calculate the finite difference approximation of the ODE

$$\ddot{y}(t_n) = \frac{y_{n+1} - 2y_n + y_{n-1}}{\Delta t^2}$$

- Solve the system of equations via Gauss-Seidel or Jacobi

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta t^2} - b_i \cdot y_i = c_i$$

$$\begin{pmatrix} 2 + b_1 \cdot \Delta t^2 & -1 & 0 & \dots & 0 \\ -1 & 2 + b_2 \cdot \Delta t^2 & -1 & \dots & 0 \\ 0 & -1 & 2 + b_3 \cdot \Delta t^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 2 + b_{N-1} \cdot \Delta t^2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} -\Delta t^2 \cdot c_1 + y_0 \\ -\Delta t^2 \cdot c_2 \\ -\Delta t^2 \cdot c_3 \\ \vdots \\ -\Delta t^2 \cdot c_{N-1} + y_N \end{pmatrix}$$

- Solve the system of equations via Gauss-Seidel or Jacobi

Shooting Method

- Transform the boundary value problem into an initial value problem
- Solve the initial value problem with an ODE solver
- Adjust the initial value, until the boundary conditions are met
- Very expensive, since the ODE solver needs to be called multiple times

$$\ddot{y} = f(t, y, \dot{y}), \qquad y(t_0) = y_0, y(t_1) = y_1$$

Fuzzy Logic

Fuzzy logic is used to model feedback system. The goal is to reduce the error between the desired value and the actual value.

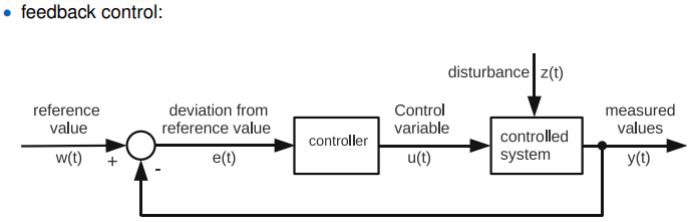

Abbildung 1: Feedback System

P Controller

A P controller is the simplest controller. It uses the error to calculate the output.

$$u(t) = K_p \cdot e(t)$$

It just sees the error and does not care about the change of the error or the integral of the error.

PID Controller

A PID controller is a controller, which uses the error, the change of the error and the integral of the error to calculate the output.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot \frac{de(t)}{dt}$$

- The proportional part directly counters the error
- The integral part counters the error in the long run
- The differential part reduces the overshoot and oscillation

Example Linear Feedback System

The State changes as follows:

$$\dot{x}(t) = A \cdot x(t)$$

The output with a P controller is:

$$\begin{aligned}\dot{x}(t) &= A \cdot x(t) + B(-K_p \cdot x(t)) \\ &= (A - B \cdot K_p) \cdot x(t)\end{aligned}$$

This system can be solved using the Ansatz $x(t) = ve^{\lambda t}$

Categories of the solution:

- All the real parts of the eigenvalues of $(A - B \cdot K_p)$ are negative, the system is stable
- At least one eigenvalue has a positive real part, the system is unstable, because it has an exponential term
- All real parts of the eigenvalues are zero, the system is oscillating
- All real parts of the eigenvalues are negative, and all imaginary parts are zero, the system is stable, and does not oscillate

Fuzzy Set

A fuzzy set is a set, where the membership is not binary, but a value between 0 and 1.

Example: The set of tall people

- A person with a height of 1.90m is a member of the set with a membership of 1
- A person with a height of 1.80m is a member of the set with a membership of 0.8
- A person with a height of 1.20m is a member of the set with a membership of 0.1

Set Operations

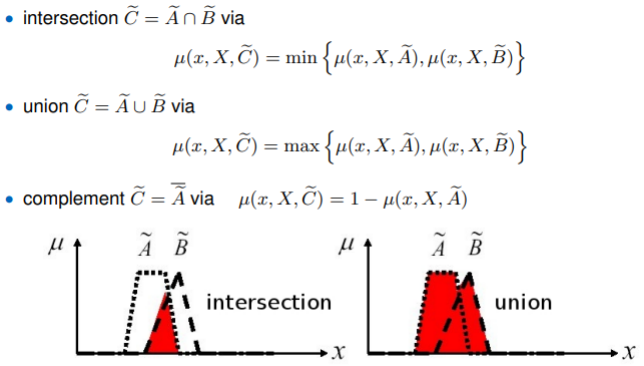

Abbildung 2: Fuzzy Logic Operations

Structure of a Fuzzy Logic System

1. Fuzzification
   - Transform the input into fuzzy sets
   - All Quantitities are transformed into linguistic variables with their membership functions
2. Create Rule Base
   - The rules are in the form of "IF ... THEN ..."
   - Example: If the temperature is high, then the fan should be fast

3. Inference
    - The fuzziness of the input is propagated through the rules
    - A fuzzy set of actions is created
4. Defuzzification
    - The fuzzy set of actions is transformed into a crisp value
    - The crisp value is the output of the fuzzy logic system

Differential Equations

A partial differential equation is an equation, which contains derivatives of multiple variables.

Example: Heat Equation

$$\frac{\partial u}{\partial t} = \alpha \cdot \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

Types Boundary Conditions

- Dirichlet Boundary Conditions
    - The value of the function is given at the boundary
    - Example: $y(0) = 0, y(1) = 1$
- Neumann Boundary Conditions
    - The derivative of the function is given at the boundary
    - Example: $y'(0) = 0, y'(1) = 1$

Finite Difference Method

In a finite difference method, the derivatives are approximated by finite differences.

The Domain consists of a grid of points. The points are indexed by i and j.

The first derivative can be approximated by the central difference:

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}$$

The second derivative can be approximated by the central difference:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

For each grid point, we can formulate the difference equation, points at the boundary are treated differently. (Dirichlet Boundary Conditions...)

Finite Element Method

In the finite element method, there is no approximation of the derivatives. Instead, the PDE is transformed into a weak formulation.

Weak Formulation

The weak formulation is obtained by multiplying the PDE with a test function and integrating over the domain.

$$\int_\Omega v \cdot \frac{\partial u}{\partial t} d\Omega = \alpha \int_\Omega v \cdot \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) d\Omega$$

Iterative Methods

Iterative methods are used to iterately solve linear systems of equations. They are often used, when the matrix is sparse.

The speed of convergence is measured by the spectral radius $\rho(M)$ of the iteration matrix M.

It is defined as the largest absolute value of the eigenvalues of M.

Relaxation Methods

Relaxation methods try to reduce the residual of the linear system of equations. By reducing the residual, the error is reduced as well.

$$r^{(i)} = b - Ax^{(i)} = Ax - Ax^{(i)} = A(x - x^{(i)}) = -Ae^{(i)}$$

All of these equations can be written in matrix form:

$$Mx^{(i+1)} + (A - M)x^{(i)} = b$$

This can be solved for $x^{(i+1)}$:

$$\begin{aligned}
x^{(i+1)} &= M^{-1}(b - (A - M)x^{(i)}) \\
&= M^{-1}b - M^{-1}(A - M)x^{(i)} \\
&= M^{-1}x^{(i)} - M^{-1}(A - M)x^{(i)} \\
&= x^{(i)} - (M^{-1}A - I)x^{(i)} \\
&= x^{(i)} - M^{-1}Ax^{(i)} + M^{-1}Mx^{(i)} \\
&= x^{(i)} - M^{-1}(Ax^{(i)} - Mx^{(i)}) \\
&= x^{(i)} - M^{-1}(Ax^{(i)} - b) \\
&= x^{(i)} + M^{-1}r^{(i)}
\end{aligned}$$

The error in the next iteration is:

$$e^{(i+1)} = x - x^{(i+1)} = x - x^{(i)} - M^{-1}r^{(i)} = e^{(i)} - M^{-1}r^{(i)} = e^{(i)} + M^{-1}Ae^{(i)} = (I - M^{-1}A)e^{(i)}$$

The error in the next iteration is the error in the current iteration multiplied by the iteration matrix $I - M^{-1}A$. One can see that the error is reduced, if the spectral radius of the iteration matrix is smaller than 1.

Richardson Iteration

The Richardson Iteration is the simplest relaxation method. It is defined as follows:

$$x^{(i+1)} = x^{(i)} + r^{(i)}$$

It can be written in matrix form as follows:

$$x^{(i+1)} = x^{(i)} + I^{-1}r^{(i)}$$

Jacobi Iteration

The Jacobi Iteration uses the diagonal of the matrix A to reduce the residual. It calculates the updated immediately at the beginning of the iteration.

$$\begin{aligned} y_i^{(i)} &= \frac{1}{a_{ii}} r_i^{(i)} \quad \text{for } i = 1, 2, ..., n \\ x^{(i+1)} &= x^{(i)} + y^{(i)} \end{aligned}$$

In matrix form:

$$x^{(i+1)} = x^{(i)} + D_A^{-1} r^{(i)}$$

where D_A is the diagonal of A.

Gauss-Seidel Iteration

The Gauss-Seidel works similar to the Jacobi Iteration, but it uses the already calculated values of x to calculate the updated values.

$$\begin{aligned} r &= b - \Sigma_{j=1}^{i-1} a_{ij} x_j^{(i+1)} - \Sigma_{j=i+1}^{n} a_{ij} x_j^{(i)} \\ y_i^{(i)} &= \frac{1}{a_{ii}} r_i^{(i)} \quad \text{for } i = 1, 2, ..., n \end{aligned}$$

$$x^{(i+1)} = x^{(i)} + y^{(i)}$$

In matrix form:

$$x^{(i+1)} = x^{(i)} + (D_A + L_A)^{-1} r^{(i)}$$

where L_A is the lower triangular part of A.

Krylov Subspace Methods

Conjugate Gradient Method

The Conjugate Gradient Method is an iterative method to solve a linear system of equations $Ax = b$. It uses Gram-Schmidt orthogonalization to find a basis for the Krylov subspace.

Algorithm:

$$p^{(0)} = r^{(0)} = b - Ax^{(0)}$$

$$\begin{aligned} \alpha_i &= \frac{r^{(i)T} r^{(i)}}{p^{(i)T} A p^{(i)}} \\ x^{(i+1)} &= x^{(i)} + \alpha_i p^{(i)} \\ r^{(i+1)} &= r^{(i)} - \alpha_i A p^{(i)} \\ \beta_i &= \frac{r^{(i+1)T} r^{(i+1)}}{r^{(i)T} r^{(i)}} \\ p^{(i+1)} &= r^{(i+1)} + \beta_i p^{(i)} \end{aligned}$$

Precoditioning

Typically, the convergence of iterative methods slows down as the grid size increases. This can be improved by using a preconditioner.

A preconditioner is a matrix M that approximates the inverse of A. Ideally, M should be A^{-1}, but this is not possible in practice as it is slow to compute. The other extreme is $M = I^1$, which is easy to compute, but does not improve the convergence.

Some compromise between these two extremes is needed:

- $M = D_A$ (Jacobi preconditioner)

Multigrid Methods

In order to avoid the slow convergence of iterative methods, on low frequency errors, multigrid methods can be used. They are based on the idea of solving the error equation $Ae = r$ on a coarser grid. The error is then interpolated back to the original grid.

They are very efficient for calculating Linear Systems of Equations.

Simulation of Traffic

Mathematical Models

- Random Variable
  - T: A random variable with distribution $F_T(t)$ and density $f_T(t)$
- Expected Value
  - $E[T] = \int_{-\infty}^{\infty} t f_T(t) dt$
- Variance
  - $Var[T] = \int_{-\infty}^{\infty} (t - E[T])^2 f_T(t) dt$
- Rho
  - $\rho = \frac{\sigma(T)}{E[T]} = \frac{\sqrt{Var[T]}}{E[T]}$
- Hazard Rate / Failure Rate
  - $h(t) = \frac{f_T(t)}{1 - F_T(t)}$ Describes the instantaneous rate of failure at time t given that the system has survived up to time t.

Example Uniform Distribution

- $T \sim U(0, 1)$
- $f_T(t) = 1$ for $0 \leq t \leq 1$ and 0 otherwise
- $F_T(t) = t$ for $0 \leq t \leq 1$, 1 for $t > 1$, and 0 otherwise

This means:

- $E[T] = \int_{-\infty}^{\infty} t f_T(t) dt = \int_0^1 t dt = \frac{1}{2}$
- $Var[T] = \int_{-\infty}^{\infty} (t - E[T])^2 f_T(t) dt = \int_0^1 (t - \frac{1}{2})^2 dt = \frac{1}{12}$
- $\rho = \frac{\sqrt{Var[T]}}{E[T]} = \frac{\frac{1}{\sqrt{12}}}{\frac{1}{2}} = \frac{1}{\sqrt{3}}$
- $h(t) = \frac{f_T(t)}{1 - F_T(t)} = \frac{1}{1-t}$ for $0 \leq t \leq 1$ and 0 otherwise

Hitchhiker's Paradox

In the example of a bus stop, given that the bus arrives at a random time T with a uniform distribution $U(0, 1)$, what is the expected time $E[T]$ that a passenger has to wait?

This is calle the FRT or Forward Recurrence Time. The BRT or Backward Recurrence Time is the time that has passed since the last bus has left.

$$E[FRT] = E[BRT] \approx \frac{1}{2} \frac{E[T^2]}{E[T]} = \frac{E[T]}{2}(1 + \rho^2(T))$$

This means that the expected time that a passenger has to wait in the case of uniform distribution is $\frac{1}{3}$.

If the coefficient of variation ρ is larger than 1, the expected time that a passenger has to wait is longer than $E[T]$. This means that even if the bus arrives, for example, on average every 10 minutes, the expected time that a passenger has to wait is longer than 10 minutes, because it is more likely to arrive in a longer section, where currently no bus is present.

Queueing Theory

Definitions

- *FU* Functional Unit:
  - *SU* The Service Unit provides the service, does not include transport
  - *C* The channel moves customers from the queue to the service unit
- A job occupies a service unit
- Dwelling Time y measures the total time a job spends in the system
  - $y = w + b$ where w is the waiting time and b is the service time
- Capacity k is the number of parallel jobs a Service Unit can handle
- Filling f is the number of jobs in the system
- Throughput d of a SU is the average number of jobs that leave the system per time unit
- Maximum Throughput c is the maximum number of jobs that can leave the system per time unit
- Utilization d/c is the relative throughput of a SU
- service time b of a job is the net-dwellin time in the SU without waiting

Little's Law

Little's Law states that the average number of jobs in the system is equal to the average throughput times the average dwelling time.

$$E[F] = E[D] \cdot E[Y]$$
$$(= E[D] \cdot E[B] = \frac{E[D]}{c} = E[R] \qquad \text{if k} = 1)$$

Example: The throughput of a MCDonalds is $E[D] = 50$ customer per hour (This means 50 customers leave the restaurant per hour). If the average customer spends $E[Y] = 30$ minutes in the restaurant, the average number of customers in the restaurant is $E[F] = E[D] \cdot E[Y] = 50 \left[\frac{customers}{hour}\right] \cdot 30\,[minutes] \cdot \frac{1}{60} \left[\frac{hours}{minute}\right] = 25\,[customers]$.

Kendall Notation

The Kendall Notation is a notation for queueing systems. It is of the form |Arrival Process|Service Process|Number of Service Units | Capacity of Queue| Maximum Clients | Queueing Discipline|.

Table of Possible Values:

Arrival Process / Service Process	Description
D	Deterministic
M	Markovian (Poisson) parameter λ or μ
G	General (arbitrary)

Queueing Discipline	Description
FCFS	First Come First Serve
LCFS	Last Come First Serve
PRIORITY	Priority Queue
ROUND ROBIN	Round Robin

Markov Chains

For markovian processes the state can be modeled with a markov chain. The state is the number of jobs in the system. The transition probabilities are the probabilities of arriving or leaving the system.

- irreducible: every state can be reached from every other state
- absorbing: once a state is reached, it is never left again

- closed subset: a subset of states that is irreducible and absorbing
- recurrence probability: probability that a state is reached again after leaving it
- transient: a state is transient if its recurrence probability is less than 1
- recurrence time: expected time until a state is reached again after leaving it
- periodic: a state is periodic if the expected time to reach it again is a multiple of the expected time to reach it again from the next state

A stable configuration is found when every inflow of a state is equal to the outflow of the state.

$$\left(\sum_{i \neq j} \lambda_{ij}\right) \cdot p_i = \sum_{i \neq j} p_j \cdot \lambda_{ji}$$
$$1 = \sum_{i=1}^{n} p_i$$

This can be used to calculate the probability of a state p_i.

Note that $\frac{\lambda}{\mu} = \frac{E[D]}{E[B]} = \frac{E[D]}{c} = r$

M/M/1 Queue

This stands for Markovian Arrival Process (arrival rate λ) / Markovian Service Process (service rate μ) / 1 Service Unit.

Using the equations from above, the following can be derived:

$$p_i = \left(\frac{\lambda}{\mu}\right)^i \cdot p_0$$
$$p_0 = 1 - \frac{\lambda}{\mu} = 1 - r$$

The average filling can be calculated as:

$$E[F] = \sum_{i=0}^{\infty} i \cdot p_i = \frac{r}{1-r}$$

With little's law, the average dwell time can be calculated as:

$$E[Y] = \frac{E[F]}{E[D]} = \frac{r/(1-r)}{\lambda} = \frac{E[B]}{1-r}$$

Random Numbers

Uniform Distribution

PCs can only generate pseudo random numbers. (For example using the LCPRNG algorithm).

Using transformations, a uniform distribution can be transformed into any other distribution.

Exponential Distribution

The exponential distribution is used to model the time between events in a Poisson process. It is the continuous counterpart to the geometric distribution.

$$f(x) = \lambda e^{-\lambda x}$$

it can be simulated using the inverse transform method:

$$
\begin{aligned}
&U \sim U(0,1) \\
&X \sim ? \\
F_X(x) &= \Pr(X \leq x) = \Pr(f(U) \leq x) = \Pr(U \leq f^{-1}(x)) = f^{-1}(x) \\
&\Rightarrow f(u) = F_X^{-1}(u) = -\frac{1}{\lambda}\ln(1-u)
\end{aligned}
$$

In our case:

$$X \sim Exp(\lambda) \Rightarrow F_X(x) = 1 - e^{-\lambda x}$$

This means that $F_X^{-1}(u) = -\frac{1}{\lambda}\ln(1-u)$.

Central Limit Theorem

When $X_1, X_2, \ldots, X_n$ are independent and identically distributed random variables with $E[X_i] = \mu$ and $Var[X_i] = \sigma^2$, then the sum of the random variables $S_n = \sum_{i=1}^{n} X_i$ is approximately normally distributed with $E[S_n] = n\mu$ and $Var[S_n] = n\sigma^2$ for large n.

We can transform any distribution into a normal distribution using the central limit theorem.

$$Z_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}} \sim N(0,1)$$

Traffic Flow