

Masters Programme in Computer and Information Engineering

Anton Augustsson



# Contents

<b>1 Basic Course in Mathematics</b>	<b>5</b>
1.1 Basic . . . . .	6
1.1.1 Mängder . . . . .	6
1.1.2 Intervall . . . . .	6
1.1.3 Potenslagar . . . . .	6
1.1.4 Logaritmer . . . . .	6
1.2 Komplexa tal . . . . .	7
1.2.1 Polärform . . . . .	7
1.3 Absolut Belopp . . . . .	8
1.4 Summor . . . . .	8
1.4.1 Aritmetiska summor . . . . .	8
1.4.2 Geometriska summor . . . . .	8
1.5 Kombinatorik . . . . .	8
1.5.1 Multiplikations principen . . . . .	8
1.5.2 Kombinationer . . . . .	8
1.5.3 Binomial satsen . . . . .	9
1.6 Funktioner och kordinatsystem . . . . .	9
1.6.1 Avståndsformeln . . . . .	9
1.6.2 Elipser . . . . .	9
1.7 Polynom division . . . . .	9
1.8 Trigonometri . . . . .	10
1.8.1 Tillvägagångs sätt . . . . .	11
<b>2 Program Design and Data Structures</b>	<b>13</b>
2.1 Coding convention . . . . .	14
2.2 Design approach . . . . .	14
2.2.1 Process . . . . .	14
2.3 Recursion . . . . .	14
2.3.1 Recursion types . . . . .	14
2.4 Complexity . . . . .	14
2.4.1 Growth . . . . .	14
2.5 Recurrences . . . . .	14
2.5.1 Closed Form . . . . .	15
2.6 Higher-Order Function . . . . .	17
2.6.1 Higher-Order Functions on Lists . . . . .	17
2.7 Data types . . . . .	17
2.7.1 Basic . . . . .	17
2.7.2 Maybe Type . . . . .	17
2.7.3 New types and enumeration types . . . . .	18
2.7.4 Inductive Data Types . . . . .	19
2.7.5 Trees . . . . .	20

2.7.6	Other data types . . . . .	22
2.7.7	Graphs . . . . .	23
2.8	Important syntax . . . . .	24
2.8.1	Let . . . . .	24
2.8.2	IO . . . . .	25
2.9	Sorting Algorithms . . . . .	25
<b>3</b>	<b>Algebra 1</b>	<b>27</b>
3.1	logik . . . . .	28
3.1.1	värde tabeller . . . . .	28
3.1.2	Implikationer . . . . .	28
3.2	Mängder . . . . .	29
3.3	Bevis . . . . .	29
3.3.1	Induktions bevis . . . . .	29
3.3.2	Motsägelse bevis . . . . .	29
3.4	Delbarhet . . . . .	29
3.4.1	Största Gemensama Delaren (SGD) . . . . .	30
3.4.2	Primtal . . . . .	30
3.5	Diofantiska ekvationer . . . . .	31
3.6	Talbaser . . . . .	31
3.6.1	konvertera från decimal bass till annan bas . . . . .	31
3.6.2	konvertera från annan bas till decimal bas . . . . .	31
3.6.3	Andra exempel . . . . .	32
3.7	Functioner . . . . .	32
3.7.1	Inversen . . . . .	32
3.7.2	Relatioiner . . . . .	32
3.8	Summor . . . . .	32
3.8.1	Aritmetiska summor . . . . .	32
3.8.2	Geometriska summor . . . . .	32
3.9	Kongruensräkning . . . . .	33
3.10	Kardinalitet . . . . .	33
3.10.1	Uppräkneligamängder . . . . .	33
3.11	Polynom . . . . .	33
3.11.1	Polynom division . . . . .	33
3.11.2	Faktorsatsen . . . . .	33
<b>4</b>	<b>Single Variable Calculus</b>	<b>35</b>
4.1	Basic . . . . .	36
4.1.1	Mängder . . . . .	36
4.1.2	Interval . . . . .	36
4.1.3	Funktion . . . . .	36
4.1.4	Trigonometri . . . . .	37
4.1.5	Exempel: Trigonometri . . . . .	38
4.2	Gränsvärden . . . . .	38
4.2.1	Kontinuitet . . . . .	39
4.3	Derivator . . . . .	39
4.3.1	Kjedje regeln . . . . .	39
4.3.2	L'Hôpital's rule . . . . .	39
4.3.3	Medelvärdessatsen . . . . .	39
4.3.4	Rolle . . . . .	40
4.3.5	växande funktioner . . . . .	40
4.3.6	Högreordnings derivator . . . . .	40
4.3.7	Impericit derivering . . . . .	40
4.3.8	invers funktioner . . . . .	40

4.3.9	exponetial och logaritm . . . . .	41
4.3.10	odefinerad form . . . . .	41
4.3.11	inversa trigometriska funktioner . . . . .	41
4.4	Grafritning . . . . .	42
4.5	Optemering . . . . .	43
4.6	Talfölder och serier . . . . .	43
4.6.1	Serier med varierande tecken . . . . .	44
4.6.2	Potensserier . . . . .	44
4.6.3	Taylor serier . . . . .	45
4.7	Integraler . . . . .	46
4.7.1	variabelsubstition . . . . .	47
4.7.2	Integration av rationella funktioner . . . . .	48
4.7.3	Partiell integration . . . . .	49
4.7.4	Generalisande integraler . . . . .	50
4.7.5	Volymberäkningar . . . . .	51
4.8	Differential ekvationer . . . . .	52
4.8.1	superabla ekvationer . . . . .	52
4.8.2	Linjära differentialekvationer av ordning 1 . . . . .	52
4.8.3	Linjära differentialekvationer av ordning 2 . . . . .	53
<b>5</b>	<b>Computer Architecture</b> . . . . .	<b>57</b>
5.1	ISA1 . . . . .	58
5.1.1	MIPS instructions . . . . .	58
5.1.2	Sequencing . . . . .	59
5.1.3	Converge to Assambly code . . . . .	59
5.2	ISA2 . . . . .	59
5.2.1	MIPS type format . . . . .	59
5.2.2	Procedure . . . . .	60
5.2.3	Other ISA . . . . .	61
5.3	Arithmetic . . . . .	62
5.3.1	Binary numbers . . . . .	62
5.3.2	Negative integers . . . . .	62
5.3.3	Operations . . . . .	62
5.3.4	Non integer numbers (Floating and fixed point) . . . . .	62
5.3.5	Overflow . . . . .	63
5.4	Logic . . . . .	63
5.5	processor control and datapath . . . . .	65
5.5.1	Clock . . . . .	66
5.5.2	Critical path . . . . .	66
5.6	pipeline . . . . .	66
5.7	Pipeline hazards . . . . .	68
5.7.1	Data hazards . . . . .	68
5.7.2	Control hazards . . . . .	68
5.7.3	Structural hazards . . . . .	68
5.8	Predicting Branches and Exceptions . . . . .	69
5.8.1	Static predictors . . . . .	69
5.8.2	Dynamic predictors . . . . .	69
5.8.3	Exceptions . . . . .	70
5.9	Input/Output . . . . .	71
5.10	Cache . . . . .	72
5.10.1	Memory hierarchy . . . . .	72
5.10.2	Cache misses 3C's . . . . .	74
5.11	Virtual Memory . . . . .	74
5.12	Parallism . . . . .	76

5.12.1 Multicore . . . . .	76
5.12.2 Paralel programming . . . . .	77
5.12.3 Synchronization . . . . .	77
5.12.4 Cache coherency . . . . .	78
5.12.5 ILP . . . . .	78
<b>6 Imperativ and Object-Oriented Programming Methodology</b>	<b>81</b>
6.1 Introduction . . . . .	82
6.1.1 Test . . . . .	82
6.2 Imperative programming i C . . . . .	82
6.2.1 Data strukturer . . . . .	82
6.3 Object-Oriented programming i Java . . . . .	83
<b>7 Linear Algebra and Geometry I</b>	<b>85</b>
7.1 Linjära ekvationssystem . . . . .	86
7.1.1 Total Matris . . . . .	86
7.2 Vektorer/koordinater i planet och rummet . . . . .	87
7.2.1 Bas . . . . .	87
7.3 Skalärprodukt och vektorprodukt . . . . .	88
7.3.1 Skalärprodukt . . . . .	88
7.3.2 Ortogonal projektion . . . . .	89
7.3.3 Enhetsvektoror och ON-baser . . . . .	91
7.3.4 Vektorprodukten . . . . .	92
7.3.5 Area och Volym . . . . .	93
7.4 Linjer och plan . . . . .	93
7.4.1 ortogonal projektion på plan . . . . .	95
7.5 Matrisräkning . . . . .	95
7.5.1 Transponat . . . . .	96
7.5.2 Matrisinvers . . . . .	96
7.6 Determinanter . . . . .	97
7.6.1 Ko-faktornar . . . . .	98
7.6.2 Geometri: parallelepiped . . . . .	98
7.7 Vektorer i $\mathbb{R}^n$ . . . . .	98
7.8 Linjära avbildningar $\mathbb{R}^n \rightarrow \mathbb{R}^m$ . . . . .	98
7.8.1 Matristransformationer och linjära funktioner . . . . .	98
7.8.2 Injektiv/Surjektiv/Bijektiv . . . . .	100
<b>8 Linear Algebra II</b>	<b>101</b>
8.1 Grudläggande teori . . . . .	102
8.1.1 Ekvationssystem och matris räkning . . . . .	102
8.1.2 determinanter . . . . .	103
8.1.3 flerdimisionel dvs $R^n$ . . . . .	103
8.1.4 Funktioner . . . . .	103
8.1.5 Linjer . . . . .	103
8.2 Vektorrum . . . . .	103
8.3 Underrum och linjära höljet . . . . .	104
8.4 Linjärt oberoende . . . . .	104
8.5 Bas . . . . .	104
8.6 Basomvandling . . . . .	105
8.7 Linjär avbildning . . . . .	106
8.8 Matrisen av en linjär avbildning . . . . .	106
8.9 Basbyte av linjära avbildningar . . . . .	107
8.10 Kärna och bild av en linjär avbildning . . . . .	107
8.11 Egenvärden och egenvektoror . . . . .	108

8.12 Diagonalisering . . . . .	109
8.13 Inre produkrum . . . . .	109
8.14 Symetriska och positiva definita matriser . . . . .	110
8.15 On-baser och Gram-Schmidt-ortonormalisering . . . . .	110
8.16 Isometriska avbildningar och spektralsatsen . . . . .	112
8.17 Andragradskurvor och andragradsytor . . . . .	112
8.18 System av linjära differentialekvationer . . . . .	113
<b>9 Computer System with Project Work</b>	<b>115</b>
9.1 M1 . . . . .	116
9.1.1 CPU . . . . .	116
9.1.2 Register . . . . .	116
9.1.3 Memory . . . . .	116
9.1.4 CPU context . . . . .	116
9.1.5 Memory allocation . . . . .	116
9.1.6 Kernel . . . . .	116
9.1.7 Multiprogramming . . . . .	116
9.2 M2 . . . . .	116
9.2.1 Network communication . . . . .	116
9.2.2 Processes . . . . .	116
9.3 M3 . . . . .	116
9.3.1 Dispatcher . . . . .	116
9.3.2 PCB . . . . .	117
9.3.3 IO bound/CPU bound . . . . .	117
9.3.4 Schedular algorithems . . . . .	117
9.4 M4 . . . . .	117
9.4.1 Concurrency . . . . .	117
9.4.2 Parallelism . . . . .	117
9.4.3 Client-Server module . . . . .	117
9.4.4 Threads . . . . .	117
9.4.5 Locks . . . . .	118
9.4.6 Spinlock . . . . .	118
9.4.7 TestAndSet . . . . .	118
9.4.8 Swap . . . . .	118
9.4.9 Bounded waiting . . . . .	118
9.4.10 Semaphores . . . . .	118
9.4.11 Mutex lock . . . . .	118
9.4.12 Deadlock . . . . .	118
9.4.13 Mutual exclusion . . . . .	118
9.4.14 Deadlock prevention . . . . .	118
9.4.15 Deadlock avoidance . . . . .	118
9.4.16 Clock Synchronization . . . . .	118
9.4.17 Mutual exclusion . . . . .	118
9.4.18 Desired properties of Transactions . . . . .	118
9.4.19 WiFi . . . . .	118
9.4.20 Bluetooth . . . . .	119
9.4.21 ISM-band . . . . .	119
9.4.22 Properties of a medium . . . . .	119
9.4.23 Sampling . . . . .	119
9.4.24 Multithreading models . . . . .	119
9.4.25 Bounded buffer . . . . .	119
9.4.26 Priority inversion . . . . .	119
9.5 M5 . . . . .	119
9.5.1 Single contiguous . . . . .	119

9.5.2	Swapping . . . . .	119
9.5.3	Memory resident . . . . .	119
9.5.4	Partitioned allocation . . . . .	119
9.5.5	Logical address space . . . . .	119
9.5.6	Address binding . . . . .	119
9.5.7	Memory management unit (MMU) . . . . .	120
9.5.8	Fragmentation . . . . .	120
9.5.9	Compaction . . . . .	120
9.5.10	Memory protection . . . . .	120
9.5.11	Shared pages . . . . .	120
9.5.12	Translation lookaside buffer . . . . .	120
9.5.13	Operating System . . . . .	120
9.5.14	Access methods . . . . .	120
9.5.15	RAM . . . . .	120
9.5.16	Volatile memory . . . . .	120
9.5.17	Persistent data storage . . . . .	120
9.5.18	The file . . . . .	120
9.5.19	Block data storage . . . . .	120
9.5.20	File control block . . . . .	120
9.5.21	Directory . . . . .	120
9.5.22	Contiguous allocation . . . . .	120
9.5.23	Linked allocation . . . . .	120
9.5.24	Indexed allocation . . . . .	120
9.5.25	FAT . . . . .	120
9.5.26	iNode . . . . .	121
9.5.27	Directory . . . . .	121
9.6	M6 . . . . .	121
9.6.1	Classic historical ciphers . . . . .	121
9.6.2	Modern cryptography . . . . .	121
9.6.3	Public key exchange . . . . .	121
9.6.4	Certification Authority (CA) . . . . .	121
9.6.5	Chain of trust . . . . .	121
9.6.6	Self-issued certificates . . . . .	121
9.6.7	Usage of modern cryptography . . . . .	121
9.6.8	Man-in-middle attack . . . . .	121
9.6.9	Chosen-plaintext attack . . . . .	121
9.6.10	Side-channel attack . . . . .	121
9.6.11	Replay attack . . . . .	122
9.6.12	Security in the Internet stack . . . . .	122
9.6.13	Firewall . . . . .	122
<b>10</b>	<b>System Design with User Perspective</b>	<b>123</b>
10.1	Documentation . . . . .	124
10.1.1	Implicit documentation . . . . .	124
10.1.2	Implicit documentation . . . . .	124
10.2	Human factors . . . . .	124
10.2.1	Color combinations . . . . .	124
10.2.2	Personas . . . . .	124
10.2.3	Memory . . . . .	124
10.2.4	What is needed to design a system? . . . . .	124
10.3	Model-View-Control . . . . .	124
<b>11</b>	<b>Probability and Statistics DV</b>	<b>125</b>
11.1	Statistisk mätt och begreppet sannolikhet . . . . .	126

11.1.1 Begrepp . . . . .	126
11.2 Sannolikheter och slumpvariabler . . . . .	127
11.2.1 Betingade sannolikheter . . . . .	127
11.2.2 Kedjer av händelse . . . . .	127
11.2.3 Oberonde händelser . . . . .	127
11.3 Fördelningar . . . . .	127
11.3.1 Binomial-fördelningar . . . . .	128
11.3.2 Possion-fördelningar . . . . .	128
11.3.3 Likformig/rektangulär-fördelningar . . . . .	128
11.3.4 Exponential-fördelningar . . . . .	128
11.3.5 Normalfördelning-fördelningar . . . . .	129
11.3.6 Läges och spridningsmått . . . . .	129
11.4 Olikheter . . . . .	129
11.4.1 Markovs olikhet . . . . .	129
11.4.2 Thebysjövs olikhet (chebyshev) . . . . .	129
11.4.3 Fördelnings funktioner . . . . .	129
11.4.4 Oberonede slupvariabel . . . . .	130
11.4.5 Fördelning av summor . . . . .	130
11.4.6 Central gränsvärdessatsen (CGS) . . . . .	130
11.5 Simulering av slumptal . . . . .	130
11.5.1 äkta slumppmässiga tal . . . . .	130
11.5.2 Pseudoslumppmässiga tal . . . . .	131
11.6 Statistikens grunder . . . . .	132
11.6.1 Allmänt . . . . .	132
11.6.2 Medelfel . . . . .	132
11.6.3 Skattning av varians . . . . .	132
11.6.4 Väntevärdsriktig . . . . .	132
11.6.5 Konfidensintervall för $\mu$ från $N(\mu, \sigma^2)$ med känt $\sigma$ . . . . .	133
11.6.6 Example . . . . .	133
11.6.7 Konfidensintervall för $p$ från binomialfördelad . . . . .	133
11.6.8 Konfidensintervall för skillnad i väntevärde . . . . .	133
11.6.9 Ensidiga intervall . . . . .	133
11.6.10 Stickprov i par . . . . .	133
11.7 Regressions . . . . .	134
11.7.1 Modell . . . . .	134
11.7.2 Modellens giltighet . . . . .	134
11.7.3 Användning av modellen . . . . .	134
11.8 Stokastiska processer . . . . .	134
11.8.1 Bornulli processer . . . . .	134
11.8.2 Poisson processer . . . . .	135
11.9 Markovkedjor . . . . .	136
11.9.1 Ehrenfestmodellen . . . . .	136
11.9.2 Google-kedjan . . . . .	137
11.9.3 Hashfunktioner . . . . .	137
11.9.4 Kollisionmodell . . . . .	137
11.9.5 Markov Buffer/Markovkö . . . . .	137
<b>12 Digital Technology and Electronics</b>	<b>139</b>
12.1 Circuit Theory . . . . .	140
12.1.1 Basic electrical quantities . . . . .	140
12.1.2 Prefixes . . . . .	141
12.1.3 Unit grammar . . . . .	141
12.1.4 Ohm's law . . . . .	141
12.1.5 Circuit elements . . . . .	141

12.1.6 Circuit terminology . . . . .	141
12.1.7 Series Resistor . . . . .	142
12.1.8 Parralel resistor . . . . .	143
12.1.9 Simplify resistor network . . . . .	143
12.1.10 Voltage divider . . . . .	144
12.1.11 Kirchhoff's laws . . . . .	144
12.1.12 Node voltage method . . . . .	145
12.1.13 Mesh current method . . . . .	147
12.1.14 Capacitor i-v equation . . . . .	147
12.1.15 inductor i-v equation . . . . .	147
12.1.16 RC circuit . . . . .	148
<b>12.2 Amplifier . . . . .</b>	<b>150</b>
12.2.1 Inside Op-Amp . . . . .	150
12.2.2 Implementation of Op-Amp . . . . .	151
12.2.3 Gerneral example . . . . .	155
<b>12.3 Semiconductors . . . . .</b>	<b>157</b>
12.3.1 Diodes . . . . .	157
12.3.2 Transistors . . . . .	162
<b>12.4 Digital Circuits: Combinatorial and Sequential Networks . . . . .</b>	<b>165</b>
12.4.1 Combinational . . . . .	166
12.4.2 Sequential . . . . .	168
12.4.3 Shift registers . . . . .	172
12.4.4 Multiplixer . . . . .	172
<b>12.5 AC Circuit Analysis &amp; Filtering . . . . .</b>	<b>173</b>
12.5.1 Impedance . . . . .	175
12.5.2 Filters . . . . .	175
<b>13 Numerical Methods and Simulation . . . . .</b>	<b>179</b>
13.1 Matlab . . . . .	180
13.2 Aritmatic . . . . .	180
13.2.1 IEEE . . . . .	180
13.2.2 Maskinepsilon . . . . .	180
13.2.3 Diskretiseringsfel . . . . .	180
13.3 ODE . . . . .	180
13.3.1 Numeriska metoder . . . . .	181
13.3.2 Högre årdningens ODE . . . . .	183
13.4 Analys . . . . .	183
13.4.1 Analys av metoder . . . . .	183
13.4.2 Konsistent . . . . .	183
13.4.3 Rättstält . . . . .	183
13.4.4 Noggrannhetsordning . . . . .	183
13.4.5 Stabilitet . . . . .	184
13.4.6 Stabilitet generella ekvationer . . . . .	184
13.4.7 Stabilitet för system . . . . .	185
13.5 Stokastiska metoden . . . . .	185
13.5.1 Monte Carlo . . . . .	185
13.5.2 Invers Transform Sampling (ITS) . . . . .	185
13.5.3 Gillespie algorithm/Stochastic Simulation Algorithm (SSA) . . . . .	186
13.6 Ordbesta . . . . .	186
<b>14 Signal and Transforms . . . . .</b>	<b>189</b>
14.1 Introduction . . . . .	190
14.1.1 System Clasification . . . . .	190
14.2 Continuous-time signals and time-domain analysis . . . . .	192

14.2.1 Continuous Time Signals . . . . .	192
14.2.2 Time Domain Analysis of Continuous Time Systems . . . . .	192
14.3 Continuous Time Fourier Series and Transform . . . . .	193
14.3.1 Continuous Time Fourier Series . . . . .	193
14.3.2 Continuous Time Fourier Transforms . . . . .	194
14.3.3 Frequency Domain Analysis of Continuous Time Systems . . . . .	195
14.4 Laplace transform . . . . .	195
14.4.1 Properties . . . . .	197
14.4.2 Transfer function . . . . .	198
14.4.3 Bode plot . . . . .	199
14.5 Filtering theory . . . . .	202
14.5.1 Filtering Theory and Ideal Filters . . . . .	202
14.5.2 Filter Approximations . . . . .	203
14.6 Sampling and Reconstruction . . . . .	204
14.6.1 Sampling . . . . .	204
14.6.2 Reconstruction . . . . .	205
14.6.3 Quantization . . . . .	205
14.7 Discrete time signals and systems . . . . .	206
14.7.1 Discrete Time Signals . . . . .	206
14.7.2 Time Domain Analysis of Discrete Time Systems . . . . .	207
14.8 Discrete Time Fourier Analysis . . . . .	208
14.8.1 Discrete Time Fourier Series and Transform . . . . .	208
14.8.2 Discrete Fourier Transform . . . . .	210
14.9 z-transform . . . . .	211
14.9.1 Definition and Properties . . . . .	211
14.9.2 Transfer Function . . . . .	213
14.10 Digital Filters . . . . .	215
14.10.1 Introduction . . . . .	215
14.10.2 Finite Impulse Response Filters . . . . .	215
14.10.3 Infinite Impulse Response Filters . . . . .	216
14.11 How it is connected . . . . .	217
<b>Appendices</b>	<b>219</b>
<b>15 Database Design I Sammanfattning</b>	<b>229</b>
15.1 Ch1. Databases and Database Users . . . . .	230
15.2 Ch3. Data Modeling Using the Entity-Relationship (ER) Model . . . . .	230
15.3 Ch4. Enhanced Entity-Relationship (EER) Modeling . . . . .	231
15.4 Ch5. The Relational Data Model and Relational Database Constraints . . . . .	232
15.4.1 Constraints . . . . .	232
15.5 Ch.13 Normalization . . . . .	232
15.5.1 Anomalies . . . . .	232
15.5.2 Normal forms . . . . .	233
15.6 Ch.9 . . . . .	234
15.7 Ch.6 SQL . . . . .	234
15.7.1 Terminology . . . . .	234
15.7.2 SQL types . . . . .	234
15.7.3 SQL examples . . . . .	235
15.7.4 Constraints . . . . .	236
15.8 Ch.20 Transaction . . . . .	237
15.8.1 Transaction Support in SQL . . . . .	238
<b>Appendices</b>	<b>241</b>
<b>16 Industrial Management</b>	<b>245</b>

16.1 Verksamheter . . . . .	246
16.1.1 Entreprenörskap . . . . .	246
16.1.2 Inovation . . . . .	246
16.1.3 Verksamhetstyp . . . . .	246
16.2 Planering för verksamheten . . . . .	247
16.2.1 Företagets strategiska mål . . . . .	247
16.2.2 Stratige modeler . . . . .	248
16.2.3 Ekonomisktyrning (Managment control) . . . . .	249
16.2.4 Ekonomiskstyrning modeller . . . . .	250
16.3 Att agera på marknaden . . . . .	250
16.3.1 Typer av marknader . . . . .	250
16.3.2 Marknadsföring . . . . .	251
16.3.3 Strategisk prissättning . . . . .	251
16.3.4 Tjänsteorientering . . . . .	252
16.4 Organisera verksamhetern . . . . .	252
16.4.1 Typer av hierarkiska organisationer . . . . .	252
16.4.2 Projektorganisation . . . . .	253
16.4.3 Rerulstatvärdesmetoden . . . . .	253
16.4.4 Produktion . . . . .	253
16.4.5 Lager Optimering . . . . .	254
16.4.6 Vad gör en ledare? . . . . .	254
16.5 Kalkyler för produkter och order . . . . .	255
16.5.1 Begrepp i flödet . . . . .	255
16.5.2 Kalkylbegrepp . . . . .	255
16.5.3 Kalkylmodeller . . . . .	255
16.5.4 Påläggskalkyl . . . . .	257
16.5.5 Aktivitetsbaserad kalkyl, ABC . . . . .	258
16.5.6 Bidragskalkyl (Contribution Margin Costing) . . . . .	259
16.5.7 Stegkalkyl . . . . .	260
16.5.8 Andra självkostnadskalkyler . . . . .	260
16.6 Investeringsbeslut . . . . .	260
16.7 Uppföljning av verksamheten . . . . .	262
16.7.1 Redovisningens regler . . . . .	264
16.7.2 Måt . . . . .	264
16.8 Analysera förändring . . . . .	265
16.9 Verksamhetens finansiering . . . . .	267
16.9.1 Anläggningskapital (Fixed capital) . . . . .	267
16.9.2 Rörelsekapital (Working capital) . . . . .	267
16.9.3 Safety capital . . . . .	268
16.9.4 Other terminology . . . . .	268
16.9.5 Analyser . . . . .	268
<b>17 Introduction to Computer Control Systems</b>	<b>271</b>
17.1 Intro . . . . .	272
17.1.1 Control with and without feedback . . . . .	272
17.1.2 Typical design requirements . . . . .	272
17.1.3 Laplace repetition . . . . .	272
17.1.4 Transfer function of a controller . . . . .	273
17.1.5 Poles and zeros . . . . .	273
17.1.6 Static gain of system . . . . .	273
17.2 PID control . . . . .	273
17.2.1 Feedback control based on error . . . . .	274
17.2.2 Classical closed-loop design methods . . . . .	275
17.3 Good Controller? . . . . .	278

17.3.1 Method 1: Frequency description of system . . . . .	278
17.3.2 Tracking properties of closed-loop system . . . . .	278
17.3.3 Method 2: Design via open-loop system . . . . .	279
17.3.4 General linear feedback control . . . . .	284
17.4 Design general linear feedback . . . . .	285
17.4.1 System states . . . . .	285
17.5 Nonlinear time-invariant system . . . . .	287
17.5.1 Minimal realization of state-space form . . . . .	289
17.5.2 Method: Design of state-feedback control . . . . .	289
17.5.3 Estimation via simulation . . . . .	290
17.6 Sensitivity and robustness . . . . .	291
17.6.1 Robustness: model error and stability . . . . .	292
17.7 Digital Controllers . . . . .	292
17.7.1 Sampling continuos-time output . . . . .	292
17.7.2 Generating continuous-time input . . . . .	294
17.7.3 Discrete-time controllers . . . . .	294
17.8 Final notes before exam . . . . .	294
<b>18 Introduction to Machine Learning</b>	<b>297</b>
18.1 Introduction . . . . .	298
18.1.1 Types of Learning . . . . .	298
18.1.2 Notation and Definitions . . . . .	298
18.2 Linear Regression as Machine Learning . . . . .	299
18.2.1 gradient descent . . . . .	300
18.3 Probability and Naive Bayes Classification . . . . .	300
18.4 Logistic Regression . . . . .	301
18.4.1 Cost function . . . . .	301
18.4.2 Gradient dissent . . . . .	301
18.5 Support Vector Machines . . . . .	302
18.5.1 Quadratic programming . . . . .	302
18.5.2 Slack Variables . . . . .	302
18.5.3 kernel trick . . . . .	302
18.5.4 Gaussian Kernels . . . . .	303
18.6 Some feature engineering and Cross validation . . . . .	303
18.6.1 k-fold cross validation . . . . .	303
18.6.2 Estimating Hyper parameters - Grid search . . . . .	303
18.6.3 One-Hot encoding . . . . .	304
18.6.4 Boosting for feature selection of linear models . . . . .	304
18.6.5 Co-variance matrix . . . . .	304
18.6.6 Correlation matrix . . . . .	304
18.6.7 Heatmap . . . . .	304
18.7 Clustering and classifiers . . . . .	305
18.7.1 k-nearest neighbor classifier . . . . .	305
18.7.2 Hierarchical clustering . . . . .	305
18.7.3 K-Means . . . . .	305
18.7.4 DBSCAN . . . . .	306
18.8 Information theory and Decision Theory . . . . .	306
18.8.1 Decision Trees . . . . .	306
18.8.2 ID3 algorithm . . . . .	306
18.8.3 Measuring Information . . . . .	306
18.9 Principle component analysis (PCA) . . . . .	308
18.9.1 Covariance Matrix . . . . .	308
<b>19 Independent Project in Information Engineering</b>	<b>309</b>

19.1 Design thinking . . . . .	309
19.2 Innovation & Immaterialrätt . . . . .	309
19.3 Källor och kritisk granskning . . . . .	309
19.4 Crash-course inom dataskydd . . . . .	310
19.5 Etik . . . . .	310
19.6 Presentationsteknik . . . . .	310
<b>20 Software Engineering and Project Management</b>	<b>313</b>
20.1 Part 1 Introduction to Software Engineering . . . . .	313
20.1.1 Introduction . . . . .	313
20.1.2 Software processes . . . . .	314
20.1.3 Agile software development . . . . .	316
20.1.4 Requirements engineering . . . . .	319
20.1.5 System modeling . . . . .	320
20.1.6 Architectural design . . . . .	321
20.1.7 Software testing . . . . .	322
20.2 Part 2 System Dependability and Security . . . . .	324
20.3 Part 3 Advanced Software Engineering . . . . .	324
20.4 Part 4 Software Management . . . . .	324
<b>21 Introduction to Parallel Programming</b>	<b>325</b>
21.1 Intro . . . . .	325
21.1.1 Why parallel computing? . . . . .	326
21.1.2 Basic Concepts . . . . .	327
21.2 Threads and locks . . . . .	331
21.2.1 PThreads . . . . .	331
21.2.2 Locks . . . . .	331
21.2.3 Spin-lock . . . . .	332
21.2.4 Queue Locks . . . . .	333
21.3 OpenMP . . . . .	334
21.3.1 Loops . . . . .	335
21.3.2 Syncronisation . . . . .	335
21.3.3 Environment Variables . . . . .	336
21.3.4 Data Environment . . . . .	336
21.4 MPI . . . . .	336
21.4.1 Basic functions . . . . .	336
21.4.2 Deadlock with MPI . . . . .	336
21.4.3 Broadcast and reduce . . . . .	337
21.4.4 Other functions . . . . .	337
<b>22 Real Time Systems</b>	<b>339</b>
22.1 RTOS . . . . .	340
22.1.1 Time management . . . . .	340
22.1.2 Task management . . . . .	340
22.1.3 Interrupt handling . . . . .	341
22.1.4 Memory management . . . . .	342
22.1.5 Exception handling . . . . .	342
22.1.6 Task scheduling . . . . .	342
22.1.7 Task synchronization . . . . .	342
22.1.8 Types of tasks . . . . .	342
22.2 ADA . . . . .	342
22.3 Scheduling . . . . .	343
22.3.1 Scheduling algorithms . . . . .	344
22.3.2 Schedulability tests and analysis . . . . .	345

22.3.3 Jitter . . . . .	349
22.4 Workload Models . . . . .	349
22.5 Synchronization . . . . .	351
22.5.1 Blocking . . . . .	351
22.5.2 Non preemption protocol (NPP) . . . . .	352
22.5.3 Basic Priority Inheritance Protocol (BIP) . . . . .	352
22.5.4 Immediate Priority Inheritance and HLP . . . . .	353
22.5.5 Priority Ceiling Protocols (PCP) . . . . .	353
22.6 Multiprocessor scheduling . . . . .	354
22.6.1 Multiprocessor Scheduling of Task Graphs . . . . .	354
22.6.2 Multiprocessor Scheduling of Task Graphs . . . . .	355
22.7 UPPAAL . . . . .	356
22.8 Real time communication (RTC) . . . . .	356
<b>23 Advanced Software Design</b>	<b>359</b>
23.1 Domain Model . . . . .	359
23.2 Class Diagram . . . . .	359
23.3 GRASP . . . . .	361
23.4 Design patterns . . . . .	361
23.4.1 Factory method . . . . .	361
23.4.2 Abstract factory . . . . .	361
23.4.3 Builder . . . . .	361
23.4.4 Object pool . . . . .	361
23.4.5 The Observer . . . . .	361



# Chapter 1

## Basic Course in Mathematics

## 1.1 Basic

### 1.1.1 Mängder

Naturliga tal:  $\mathbb{N} = \{1, 2, 3, \dots\}$

Heltal:  $\mathbb{Z} = \{\dots - 2, 1, 0, 1, 2, \dots\}$

Rationella tal:  $\mathbb{Q} = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0 \right\}$

Irrationella tal:  $\mathbb{P} = \mathbb{R} \setminus \mathbb{Q}$  eller  $\{x \mid x \in \mathbb{R}, x \notin \mathbb{Q}\}$

Reella tal:  $\mathbb{R} = \mathbb{P} \cup \mathbb{Q}$

$$\left(\frac{a}{b}\right)^{-3} = \left(\frac{b}{a}\right)^3$$

$$\sqrt{a} = a^{\frac{1}{2}}$$

### 1.1.2 Intervall

$$[a, b] = \{x \mid a \leq x \leq b\}$$

$$[a, \infty[$$

$$]-\infty, \infty[$$

Tillvägagångs sätt

$$\frac{2}{x-3} < \frac{5}{x}$$

$$\begin{aligned} \frac{2}{x-3} - \frac{5}{x} &< 0 \\ \frac{(x)2}{x(x-3)} - \frac{5(x-3)}{x(x-3)} &< 0 \\ \frac{2x - 5x + 15}{x(x-3)} &< 0 \\ \frac{-3x + 15}{x(x-3)} &< 0 \\ \frac{-3(x-5)}{x(x-3)} &< 0 \\ x \neq 0, x \neq 3 \end{aligned}$$

Värde tabell:

	$x < 0$	$0 < x < 3$	$3 < x < 5$	$5 < x$
$x-5$	-	-	-	+
-3	-	-	-	-
x	-	+	+	+
$x-3$	-	-	+	+
hela	+	-	+	-

### 1.1.3 Potenslagar

$$(1): b = a^x \Leftrightarrow \log_a(b) = x \text{ för: } a > 0, b > 0, a \neq 1$$

$$(2): \log_a\left(\frac{b}{c}\right) = \log_a(b) - \log_a(c)$$

$$(3): \log_a(b * c) = \log_a(b) + \log_a(c)$$

$$(4): \log_a(b^d) = d \log_a(b)$$

$$(5): \log_a(b) = \frac{\log_f(b)}{\log_f(a)}$$

$$(6): \log_a(a) = 1$$

$$(7): \log_a(1) = 0$$

$$(8): a^{\log_a(x)} = x$$

$$(9): \log_{a^c}(b) = \frac{1}{c} \log_a(b)$$

Tillvägagångs sätt

$$\textbf{Lös: } \log_3(x) + \log_x\left(\frac{1}{27}\right) = 2$$

$$\log_3(x) - 3 \log_x(3) = 2$$

$$\log_3(x) - 3 \frac{\log_3(3)}{\log_3(x)} = 2$$

$$\log_3(x) - 3 \frac{1}{\log_3(x)} = 2$$

$$\log_3(x)^2 - 3 = 2 \log_3(x)$$

$$y = \log_3(x)^2$$

$$y^2 - 2y - 3 = 0$$

$$\text{pq-formeln: } y = 1 \pm \sqrt{4} = 1 \pm 2$$

$$\log_3(x) = 3 \Leftrightarrow x = 3^3 = 27$$

$$\log_3(x) = -1 \Leftrightarrow x = 3^{-1} = \frac{1}{3}$$

## 1.2 Komplexa tal

$$z = a + bi, \operatorname{Re}(z) = a, \operatorname{Im}(z) = b$$

$$\bar{z} = a - bi$$

$$|z| = \sqrt{a^2 + b^2}$$

Där  $b(\operatorname{Im}(z))$  är för y-axeln och  
 $a(\operatorname{Re}(z))$  är för x-axeln

Samma räkneregler för reella tal gäller för komplexa tal

### 1.2.1 Polärform

$$\arg(z) = \alpha + 2\pi * n, n \in \mathbb{Z}$$

$\arg(z)$  är vinkeln mellan a (x-axeln) linjen  $|z|$ .

$$\operatorname{Arg}(z) \in ]-\pi, \pi[$$

$$\operatorname{Arg}(z) \in \arg(z)$$

$$a = |z| \cos \alpha$$

$$b = |z| \sin \alpha$$

$$z = |z| \cos \alpha + i * |z| \sin \alpha$$

**Polärform:**

$$z = |z|(\cos \alpha + i * \sin \alpha)$$

**Eulers formel:**

$$z = |z|e^{i\alpha}$$

**Sats:**

$$|z * w| = |z| * |w|$$

$$\arg(z * w) = \arg(z) + \arg(w)$$

$$\arg\left(\frac{z}{w}\right) = \arg(z) - \arg(w)$$

**De Movre's formel:**

$$\begin{aligned} & (\cos(\theta) + i * \sin(\theta))^n \\ &= \cos(n * \theta) + i * \sin(n * \theta) \end{aligned}$$

**Binomisk ekvation**

$$\begin{aligned} & (\cos(\theta) + i * \sin(\theta))^n \\ &= \cos(n * \theta) + i * \sin(n * \theta) \end{aligned}$$

### Tillvägagångs sätt

Visa lösningarna i det komplexa tal planet

$$z^5 = \sqrt{3} + i$$

$$|\sqrt{3} + i| = \sqrt{\sqrt{3}^2 + 1^2} = \sqrt{4} = 2$$

$$\begin{cases} \sqrt{3} = 2 \cos(\alpha) \\ 1 = 2 \sin(\alpha) \end{cases}$$

$$\alpha = \frac{\pi}{6}$$

$$\begin{aligned} & |z|^5 (\cos(5 * \theta) + i * \sin(5 * \theta)) \\ &= 2(\cos(\frac{\pi}{6}) + i * \sin(\frac{\pi}{6})) \end{aligned}$$

Vilket ger:

$$|z|^5 = 2 \Leftrightarrow |z| = 2^{\frac{1}{5}}$$

$$5 * \theta = \frac{\pi}{6} + 2\pi n \Leftrightarrow \theta = \frac{\pi}{30} + \frac{2\pi n}{5}, n \in \mathbb{Z}$$

I polärform blir det då:

$$z = 2^{\frac{1}{5}}(\cos(\frac{\pi}{30} + \frac{2\pi n}{5}) + i * \sin(\frac{\pi}{30} + \frac{2\pi n}{5}))$$

Eulers formel:  $z = 2^{\frac{1}{5}}e^{i(\frac{\pi}{30} + \frac{2\pi n}{5})}$

$$n = 0, 1, 2, 3, 4$$

$$z_1 = 2^{\frac{1}{5}}e^{i(\frac{\pi}{30})}$$

$$z_2 = 2^{\frac{1}{5}}e^{i(\frac{\pi}{30} + \frac{2\pi}{5})}$$

$$z_3 = 2^{\frac{1}{5}}e^{i(\frac{\pi}{30} + \frac{4\pi}{5})}$$

$$z_4 = 2^{\frac{1}{5}}e^{i(\frac{\pi}{30} + \frac{6\pi}{5})}$$

$$z_5 = 2^{\frac{1}{5}}e^{i(\frac{\pi}{30} + \frac{8\pi}{5})}$$

### 1.3 Absolut Belopp

Tillvägagångs sätt

$$|2x - 8| + |1 - x| - 2|x - 3| = 8 + 3x$$

**lösning**

$$\left( \begin{array}{l} 2x - 8 = 0 \\ x = 4 \end{array} \right) \left( \begin{array}{l} 1 - x = 0 \\ x = 1 \end{array} \right) \left( \begin{array}{l} x - 3 = 0 \\ x = 3 \end{array} \right)$$

$$\text{I } \begin{cases} x \leq 1 \\ -(2x - 8) + (1 - x) + 2(x - 3) = 8 + 3x \end{cases}$$

$$\text{I } \begin{cases} x \leq 1 \\ -4x = 5 \end{cases} \quad \begin{cases} x \leq 1 \\ x = -\frac{5}{4} \end{cases} \text{ lösning}$$

$$\text{II } \begin{cases} 1 < x \leq 1 \\ -(2x - 8) - (1 - x) + 2(x - 3) = 8 + 3x \end{cases}$$

$$\text{II } \begin{cases} 1 < x \leq 3 \\ -2x = 7 \end{cases} \quad \begin{cases} 1 < x < 3 \\ x = -\frac{7}{2} \text{ Ej lösning} \end{cases}$$

$$\text{III } \begin{cases} 3 \leq x < 4 \\ -(2x - 8) - (1 - x) - 2(x - 3) = 8 + 3x \end{cases}$$

$$\text{III } \begin{cases} 3 \leq x \leq 4 \\ -(2x - 8) - (4x - 3) = 8 + 3x \end{cases}$$

$$\begin{cases} 3x < 4 \\ x = \frac{5}{6} \text{ Ej lösning} \end{cases}$$

$$\text{IV } \begin{cases} x \geq 4 \\ (2x - 8) - (1 - x) - 2(x - 3) = 8 + 3x \end{cases}$$

$$\text{IV } \begin{cases} x \geq 4 \\ x = -\frac{11}{2} \text{ Ej lösning} \end{cases}$$

Exempel: summor

$$\sum_{k=n}^{2n} (2^k - k)$$

**Lösning**

sätter  $f = 0 = k - n$

$$\sum_{f=0}^n (2^{f+n} - (f + n))$$

$$\begin{aligned} &= 2^n * \sum_{f=0}^n (2^f) - \sum_{f=0}^n (f + n) \\ &= \frac{2^n(2^{n+1} - 1)}{2 - 1} - \frac{3n(n + 1)}{2} \\ &= 2^{2n+1} - 2^n - \frac{3n(n + 1)}{2} \end{aligned}$$

### 1.4 Summor

#### 1.4.1 Aritmetiska summor

$$s_n = a_1 + a_2 + a_3 + \dots + a_n = \frac{n(a_1 + a_n)}{2}$$

#### 1.4.2 Geometriska summor

Börjar alltid med exponenten 0 och gör om summan så att den passar i följande talföljd:

$$s_n = a + ak + ak^2 + \dots + ak^{n-1} = \frac{a(k^n - 1)}{k - 1}$$

#### 1.5 Kombinatorik

##### 1.5.1 Multiplikations principen

permutationer  $p(a,b)$  då ordningen spelar roll. Antal sätt: (exemplet: antal sätt av måltider 7 företer 5 varmrätter 4 efterätter  $(7*5*4)$ )

$$n_1 \cdot n_2 \cdot \dots \cdot n_m$$

##### 1.5.2 Kombinationer

Kombination  $c(a,b)$  då ordningen inte spelar roll. Antal sätt: (exemplet: antal del-mängder två element ( $n =$  element  $k =$  antal element som kan väljas))

$$\binom{n}{k} = \frac{n!}{k!(n - k)!}$$

$$\binom{n}{k} = \frac{n \cdot (n - 1)(n - 2) \cdot \dots \cdot (n - 1(k - 1))}{k!}$$

**Pascal triangel**

$n$							
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1
	0	1	2	3	4	5	6
			$k$				

**1.5.3 Binomial satsen**

$$(a+b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k}$$

**1.6 Funktioner och kordinatsystem**

Kom ihåg att när det stor (x-a) förflyttas den i x-axeln a steg till höger  $\rightarrow$  medans (x+a) förflyttas den a steg till vänster  $<-$

**1.6.1 Avståndsformeln**

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

**1.6.2 Ellipser**

Förenkla till denna formeln:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Där a är avståndet på x-axeln och b är avståndet på y-axeln

**Hyperbol**

Ser väldigt anerlunda ut från ellipser

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

**Circkel**

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} = 1$$

Där  $a = r$

**1.7 Polynom division**

Exempel: Polynom division

Man vet att ekvationen  $z^4 - 2z^3 - 7z^2 + 26z - 20 = 0$  har roten  $z = 2 + i$ . Lös ekvationen fullständigt.

Då är  $z = 2 + i$  en lösning, men också konjugatet enligt faktorsatsen, dvs  $z = 2 \pm i$ . Vilket betyder att följande går att factorisera ut polynomet.

$$(z - (2 + i))(z - (2 - i)) = z^2 - 4z + 5$$

$$\begin{array}{r} \text{Långdivision} \quad \begin{array}{ccc} \text{(liggande} & \text{stolen)}: & \\ x^2 & + 2x & - 4 \end{array} \\ \hline x^2 - 4x + 5) \overline{\quad x^4 - 2x^3 \quad - 7x^2 + 26x - 20} \\ \quad - x^4 + 4x^3 \quad - 5x^2 \\ \hline \quad 2x^3 - 12x^2 + 26x \\ \quad - 2x^3 \quad + 8x^2 - 10x \\ \hline \quad - 4x^2 + 16x - 20 \\ \quad 4x^2 - 16x + 20 \\ \hline \quad 0 \end{array}$$

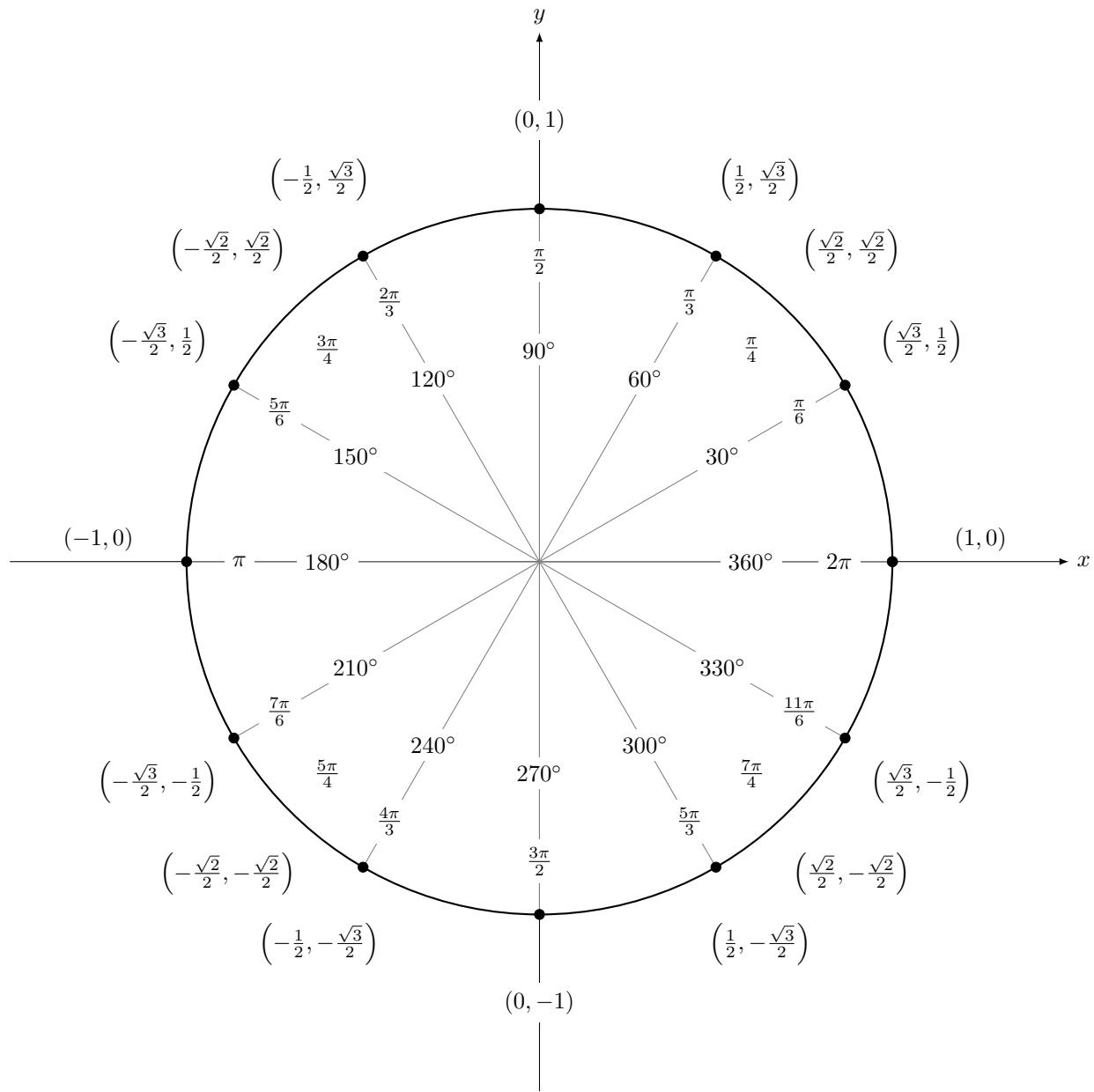
$z^2 + 2z - 4 = 0$  Är också en lösning som till slut ger följande.

$$z = -1 \pm \sqrt{5}$$

$$z = 2 \pm i$$

Varge n grads polynom har alltid n stycken komplexa lösningar

## 1.8 Trigonometri



**Sats:**

$$\begin{aligned}360^\circ &= 2\pi \text{ rad} \\v_g &= v_r * \frac{180^\circ}{\pi} \\v_r &= v_g * \frac{\pi}{180^\circ}\end{aligned}$$

**Additionsformlerna:**

$$\begin{aligned}\sin(\alpha + \beta) &= \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha) \\ \sin(\alpha - \beta) &= \sin(\alpha)\cos(\beta) - \sin(\beta)\cos(\alpha) \\ \cos(\alpha + \beta) &= \cos(\alpha)\cos(\beta) - \sin(\beta)\sin(\alpha) \\ \cos(\alpha - \beta) &= \cos(\alpha)\cos(\beta) + \sin(\beta)\sin(\alpha)\end{aligned}$$

**Sats:**

$$\begin{aligned}-1 \leq \sin t &\leq 1 \\-1 \leq \cos t &\leq 1\end{aligned}$$

**Trigonometriska ettan:**

$$\begin{aligned}(\sin t)^2 + (\cos t)^2 &= 1 \\ \sin^2 t + \cos^2 t &= 1\end{aligned}$$

**Sats:**

$$\begin{aligned}\cos(-t) &= \cos(t) \\ \sin(-t) &= -\sin(t) \\ \tan(-t) &= \frac{\sin(-t)}{\cos(-t)} = \frac{-\sin(t)}{\cos(t)}\end{aligned}$$

**1.8.1 Tillvägagångs sätt**

$$\begin{aligned}\cos \frac{\pi}{12} &= \cos \left( \frac{\pi}{3} - \frac{\pi}{4} \right) = \cos \frac{\pi}{3} \cos \frac{\pi}{4} + \sin \frac{\pi}{3} \sin \frac{\pi}{4} \\ &= \left( \frac{1}{2} \right) \left( \frac{1}{\sqrt{2}} \right) + \left( \frac{\sqrt{3}}{2} \right) \left( \frac{1}{\sqrt{2}} \right) = \frac{1 + \sqrt{3}}{2\sqrt{2}}\end{aligned}$$



## Chapter 2

# Program Design and Data Structures

## 2.1 Coding convention

### VARIANT

A (recursive) function terminates if it has a variant. The variant need to follow all of the flowing rules

- Needs to decrease every recursive call
- All ways positive

### Side effects

All IO functions have side effects in order to separate pure Haskell function with impure functions that changes the state with is commonly the case with imperative and object oriented programming. Every IO function has a side effects.

### INVARIANT

A data types invariant is what value are allowed for the data type to work. Similar to preconditions for a function. An example is integer data type that can only use positive integers therefor the invariant is positive integers.

## 2.2 Design approach

- top-down design (Cheating): Is to break down a complex system in to subsystems to solve the problem. Most often is to write everything by scratch.
- bottom-up design (Stacking): Is to piece existing system together to create a more complex system. little is programmed, most is copied.
- dodging: Get some code working more quickly, make progress with some part of the system and back-paddle to the dodged part later. The reason is to develop insight that will help solve the larger problem.

### 2.2.1 Process

1. Data Description
2. Data Examples
3. Function Description
4. Function Examples
5. Function Template
6. Code
7. Tests
8. Review and Refactor

**Programming to an Interface** More dynamic, can change models, less code to write and a layer of abstraction. ADT

## 2.3 Recursion

### 2.3.1 Recursion types

1. Simple recursion: There is at most one recursive call (in each branch) and the argument is decremented by one.
2. Complete recursion: Some argument becomes smaller in the recursive call, but not necessarily.
3. Multiple recursion: There are multiple recursive calls (in the same branch).
4. Mutual recursion: Two or more functions are defined in terms of each other.
5. Nested recursion: An argument to a recursive call is computed by a recursive call.
6. Recursion on a generalized problem: Sometimes, no suitable recursion scheme is obvious.

## 2.4 Complexity

### 2.4.1 Growth

1. Big  $\theta$  Notation: estimate of growth in intervals determine by constants Definition For non-negative functions  $f$  and  $g$ ,  $f(n) = \theta(g(n))$  if and only if there exist  $n_0 \geq 0$  and  $c_1, c_2 > 0$  such that for all  $n > n_0$   $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ .  $\theta(g(n))$  is the set of all functions  $f(n)$  that are bounded below and above
2. Big  $\Omega$  Notation: estimate of growth Lower bound
3. Big  $O$ : Notation: estimate of growth upper bound

### Relation

$$O(g(n)) \cap \omega(g(n)) = \theta(g(n))$$

## 2.5 Recurrences

### Example:

sumList [] = 0  
sumList (x:xs) = x + sumList xs

1. pattern matching [] takes  $t_0$  time

2. pattern matching  $(x : xs)$  takes  $t_1$  time
3. Adding  $x$  with recursive call takes  $t_{add}$
4. Then the recursive call takes  $T(n - 1)$

$$T(n) = \begin{cases} t_0 & \text{if } n = 0 \\ T(n - 1) + t_{add} + t_1 & \text{if } n > 0 \end{cases}$$

### 2.5.1 Closed Form

1. Use the substitution method to obtain a closed form for the following recurrence:

$$\begin{aligned} f(0) &= 5 \\ f(n) &= f(n - 1) + n + 2, n > 0 \end{aligned}$$

**Hint:**  $1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$  — you do not need to prove this fact.

### Expansion Method

$$\begin{aligned} f(0) &= 5 \\ f(1) &= f(0) + n + 2 = n + 5 + 2 \\ f(2) &= f(1) + n + 2 = 2n + 5 + 2 + 2 \\ f(3) &= f(2) + n + 2 = 3n + 5 + 2 + 2 + 2 \\ f(n) &= +n^2 + 5 + n \cdot 2 \end{aligned}$$

### Induction proof

Step1: test with base case sense the base case is predefine for 0 we do  $n = 1$

$$\begin{aligned} f(1)_{VL} &= 1^2 + 5 + 1 \cdot 2 = 8, \\ f(1)_{HL} &= f(0) + 1 + 2 = 5 + 3 = 8 \\ f(1)_{VL} &= f(1)_{HL} \end{aligned}$$

Step2: assumption for p  $f(p) = p^2 + 5 + p \cdot 2$

$$\begin{aligned} f(p + 1)_{VL} &= f(p) + (p + 1) + 2 = (p^2 + 5 + p \cdot 2) + (p + 1) + 2 = \\ &= (p^2 + (p + 1)) + 5 + (p + 1) \cdot 2 = (p^2 + 2p + 1) + 5 + (p + 1) \cdot 2 \\ f(p + 1)_{HL} &= (p + 1)^2 + 5 + (p + 1) \cdot 2 = (p^2 + 2p + 1) + 5 + (p + 1) \cdot 2 \\ f(p + 1)_{VL} &= f(p + 1)_{HL} \end{aligned}$$

Conclusion: according to induction hypothesis the recurrence of the function is equal to

$$2n + 5 + \frac{n(1 + n)}{2}$$

### Substitution Method

$$\begin{aligned} f(n) &= f(n - 1) + n + 2 \\ &= (f(n - 2) + (n - 1) + 2) + 1n + 2 \\ &= f(n - 2) + (n - 1) + n + 2 \cdot 2 \\ &= (f(n - 3) + (n - 2) + 2)(n - 1) + n + 2 \cdot 2 \\ &= f(n - 3) + (n - 2) + (n - 1) + n + 3 \cdot 2 \\ &\quad \vdots \\ &= f(n - k) + (n - (k - 1)) + (n - (k - 2)) + (n - (k - 3)) + \dots + n + k \cdot 2 \\ &\quad \vdots \\ &= f(n - n) + (n - (n - 1)) + (n - (n - 2)) + (n - (n - 3)) + \dots + n + n \cdot 2 \\ &= f(0) + 1 + 2 + 3 + \dots + n + n \cdot 2 = 5 + 1 + 2 + 3 + \dots + n + n \cdot 2 \end{aligned}$$

We can see the following patterns

$$2n + 5 + \sum_{k=1}^n (k) = 2n + 5 + \frac{n(1+n)}{2}$$

### Induction proof

Step1: test with base case since the base case is predefine for 0 we do  $n = 1$

$$f(1)_{VL} = 2 \cdot 1 + 5 + \frac{1(1+1)}{2} = 2 + 5 + 1 = 8,$$

$$f(1)_{HL} = f(0) + 1 + 2 = 5 + 3 = 8$$

$$f(1)_{VL} = f(1)_{HL}$$

Step2: assumption for p  $f(p) = 2p + 5 + \frac{p(1+p)}{2}$

$$\begin{aligned} f(p+1)_{HL} &= f(p) + (p+1) + 2 = (2p + 5 + \frac{p(1+p)}{2}) + (p+1) + 2 = \\ &= 2(p+1) + 5 + \frac{p(1+p)}{2} + p = 2(p+1) + 5 + \frac{2(p+1) + p(1+p)}{2} = \\ &= 2(p+1) + 5 + \frac{p^2 + 2p + p + 2}{2} = \\ f(p+1)_{HL} &= 2(p+1) + 5 + \frac{(p+1)(p+2)}{2} = 2(p+1) + 5 + \frac{p^2 + 2p + p + 2}{2} \\ f(p+1)_{VL} &= f(p+1)_{HL} \end{aligned}$$

Conclusion: according to induction hypothesis the recurrence of the function is equal to

$$2n + 5 + \frac{n(1+n)}{2}$$

## 2.6 Higher-Order Function

### 2.6.1 Higher-Order Functions on Lists

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldl :: (a -> b -> a) -> a -> [b] -> a
foldr :: (a -> b -> b) -> b -> [a] -> b
```

#### map

maps a function to each element in list.

#### filter

filters elements with a condision

```
filter :: (a -> Bool) -> [a] -> [a]
filter (<6) [6,3,0,1,8,5,9,3] = [3,0,1,5,3]
```

#### foldl

foldl recurses over a list “from the left,” i.e., it initially applies the given operation to the first list element and the given start value. starts from the left (first element) and apply the function to with each element. Similar to an accumulator. No one uses it since it is some what useless.

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl (*) 1 [1,2,3,4] = 24
```

#### foldr

foldr recurses over a list “from the right,” i.e., it initially applies the given operation to the last list element and the given start value. starts from the right (last element) and apply the function to with each element. Similar to an accumulator.

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr (*) 1 [1,2,3,4] = 24
```

## 2.7 Data types

### 2.7.1 Basic

<b>String</b> :: [ 'char' ]	<i>—list of charecters</i>
<b>List</b> :: []	<i>—undefine element types and elements</i>
<b>Tuple</b> :: ()	<i>—Predefine elements</i>
<b>Char</b> :: ''	<i>—single carecter</i>
<b>Int</b> :: 1	<i>—Hole number with define size</i>
<b>Integear</b> :: 1	<i>—Hole number with undefine size</i>
<b>Float</b> :: 1.1	<i>—Real number with double-precision</i>
<b>Double</b> :: 1.1	<i>—Real number with single-precision</i>

### 2.7.2 Maybe Type

If the return is maybe “nothing” then the “Maybe type” is used, since it dose not have to return the a specific value. It is not polymorphic since you have to specify the type, however “Just” is at of it self polymorphic function. If a operation that requires a specific type one needs to remove “Just”, for instance by a let function.

### 2.7.3 New types and enumeration types

New types: One create more relevant names and format of existing enumeration types. Overload is a problem with the use of the same operations that can not be used for the same data types. One needs to create new operations if it is not from the same type class.

Enumeration types: Instead of new types `type` enumeration types uses the operation call `data`. The difference is that enumeration types is independent from existing types therefor one becomes more flexible and precise.

#### example

```
data newTypeOfDataDerection = North | South | East | West
deriving (Show) — in order to print

:t North
North :: newTypeOfDataDerection

— We can use new types in pattern matching
oposit :: newTypeOfDataDerection -> newTypeOfDataDerection
oposit North = South
```

#### Type classes

A type class is a set of types that support certain related operations. No function is applied by default to the new type, therefor you can write “deriving” the following functions are good to have

#### type classes to deriving for new data types

```
deriving (Show) — in order to print in ghci and print normal
deriving (Eq) — To test equality
deriving (Ord) — the first one has the smallest value, order matter for comparison
```

#### New type classes

```
class Eq a where
(==) :: a -> a -> Bool
```

#### New type classes Instances

```
instance Eq Colour where
(==) = eqColour
```

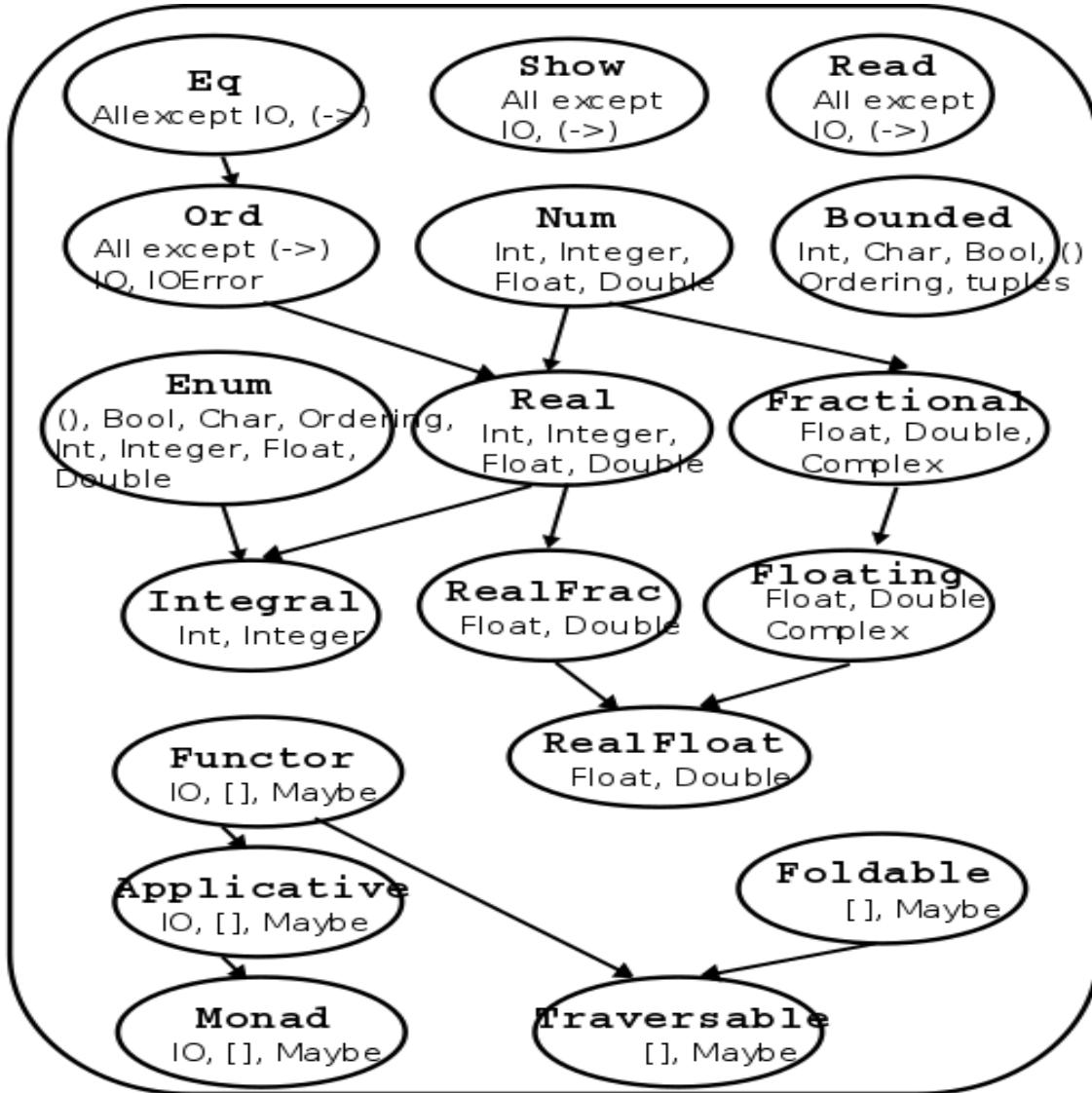


Figure 2.1: Type class ?

#### 2.7.4 Inductive Data Types

Uses a base case then the inductive step as cases of. Multiple arguments. Type constructure

```

data AExp = Atom Int
          | Plus AExp AExp
          | Times AExp AExp

eval (Atom i)      = i
eval (Plus a b)   = eval a + eval b
eval (Times a b) = eval a * eval b
  
```

## 2.7.5 Trees

### Terminology

1. Search tree: All nodes on the left side is less then the parent and opposite on the right side
2. Out-degree: is how many children it has. Binary trees has out-degree 0 – 2
3. root Node: is a parent to any number of children at the top of the hierarchy
4. sub Node: is a parent to any number of children
5. Leaf: is a nod that has no children
6. Node: every element
7. Height: most steps from the root node to a leaf

### Representation

1. Inorder: (Left, Root, Right)
2. Pre-order: (Root, Left, Right)
3. Post-order: (Left, Right, Root)

```

data FBTree a = Leaf a
      | Node (FBTree a) a (FBTree a)
deriving (Show)

Node (Leaf 1) 5 (Leaf 21)
rootValue :: FBTree a -> a
rootValue (Node _ a _) = a
rootValue (Node x) = x

height :: FBTree a -> Int
height (Leaf _) = 0
height (Node b a c) = 1 + max (height b) (height c)

```

### Binary tree

- Insertion:  $O(1)$  ( $O(h)$  if search tree)
  - Delition:  $O(n)$  ( $O(h)$  if search tree)
  - Search:  $O(n)$  ( $O(h)$  if search tree)
  - Height:  $O(n)$  (n nodes)
1. Full binary tree: has node out-degree either 2 or 0 worst-case complexity of  $O(\log n)$ .
  2. Binary tree: each node has up to an out-degree of 2

### Binary search tree

- Insertion:  $O(h)$  ( $O(n)$  if search tree)
- Delition:  $O(h)$
- Search:  $O(n)$
- Height:  $O(n)$  (n nodes)

### Red and black trees

- Insertion:  $O(\log n)$
- Deletion:  $O(\log n)$
- Search:  $O(\log n)$
- Element:  $O(2^r)$  (rank r)
- Height:  $O(2 \cdot \log_2(n+1))$  (n nodes)

Better then a normal binary tree since it balance the tree, therefor it becomes smaller and more efficient to search in. One should use red and black tree when there is a large number of nodes, say 50.

Definition: A red-black tree is a binary search tree where every node is colored either red or black, with the following balancing invariants:

1. No red node has a red parent.
2. Every path from the root to an empty subtree contains the same.
3. A red-black tree with n nodes has height at most  $2 \cdot \log_2(n+1)$ .
4. there are 4 cases to rebalance (1;4) is similar so is (2;3).

### Algorithm

1. Perform a standard binary-search-tree insertion.
2. Color the new node red.
3. Rebalance the tree, if there is a red node with a red parent.

### Binomial Trees and heaps

- Insertion:  $O(\log n)$
- Search:  $O(\log n)$  (number of trees n)
- Element:  $2^r$  (rank r, n=1 then r=0)

A heap can be used to implement a priority queue, where elements are added to a pool and assigned a priority. In a min-priority queue, extraction of an element yields an element with minimum priority. The smallest node is the root and every child is equal or larger then its parent.

Binomial Trees is a data structure by linking trees of rank  $r - 1$  together. A binomial heap (Vuillemin, 1978) is a list of binomial trees such that each tree satisfies the min-heap property (hence the root of each tree contains its minimum key); and the trees have strictly increasing ranks.

### Terminology

1. Link: putting together two trees.
2. Merge: putting together two heaps
3. Binomial Heap: a list (forest!) of Binomial Trees
4. Binomial Trees have the largest subtree to the left, while Binomial

### Binomial heaps

1. Heaps, extracting minimum element in worst case  $O(\log |h|)$
2. Binomial Trees, The height (here: number of edges on the longest branch) is  $r$ .
3. Binomial Trees, There are  $2^r$  nodes in the tree.
4. Binomial Trees, There are  $\binom{r}{k}$  nodes at level k. (Hence its name!)
5. Binomial Trees, The root has r subtrees of ranks  $r - 1, r - 2, \dots, 1, 0$ .
6. A binomial heap h has at most  $\lceil \lg |h| \rceil + 1$  binomial trees.
7. Inserting a binomial tree into a binomial heap is like addition with base 2.
8. merging is made with too cases either (case 1) when one is smaller or (case 2) when they are equal

`BinoTree = Node Int Int [BinoTree] — Node rank`

### 2.7.6 Other data types

#### Tables, Stacking and queuing

1. Table: a list of key-value pairs
2. Stacks: elements accessed in Last-In First-Out (LIFO) order
3. Queues: elements accessed in First-In First-Out (FIFO) order

#### Table operations

```

empty :: Table k v
insert :: Eq k  $\Rightarrow$  Table k v  $\rightarrow$  k  $\rightarrow$  v  $\rightarrow$  Table k v
exists :: Eq k  $\Rightarrow$  Table k v  $\rightarrow$  k  $\rightarrow$  Bool
lookup :: Eq k  $\Rightarrow$  Table k v  $\rightarrow$  k  $\rightarrow$  Maybe v — value from key
delete :: Eq k  $\Rightarrow$  Table k v  $\rightarrow$  k  $\rightarrow$  Table k v
iterate :: Table k v  $\rightarrow$  (b  $\rightarrow$  (k, v)  $\rightarrow$  b)  $\rightarrow$  b  $\rightarrow$  b — Foldr
keys :: Table k v  $\rightarrow$  (b  $\rightarrow$  k  $\rightarrow$  b)  $\rightarrow$  b  $\rightarrow$  b — all keys
values :: Table k v  $\rightarrow$  (b  $\rightarrow$  v  $\rightarrow$  b)  $\rightarrow$  b  $\rightarrow$  b — all values

```

#### Stack operations

```

— interface
newtype Stack a = StackImpl [a] — opaque!

empty :: Stack a
isEmpty :: Stack a  $\rightarrow$  Bool
push :: a  $\rightarrow$  Stack a  $\rightarrow$  Stack a — insert
top :: Stack a  $\rightarrow$  a — the first value
pop :: Stack a  $\rightarrow$  (a, Stack a) — take out

```

#### Queue operations

```

— interface
newtype Queue a = Q [a] — opaque

empty :: Queue a
isEmpty :: Queue a  $\rightarrow$  Bool
head :: Queue a  $\rightarrow$  a
enqueue :: Queue a  $\rightarrow$  a  $\rightarrow$  Queue a — take out element
dequeue :: Queue a  $\rightarrow$  Queue a — insert element
toList :: Queue a  $\rightarrow$  [a]

```

### Hashtables

- Key value lookup (an index)
- Array is only define for small index, hash has no limit on available keys.
- Typically we have  $n$  possible keys from set  $U$  for a hashtable (which is an array) with  $m$  slots, where  $n \geq m$ .
- since there is infinitely many elements and limited amount of key there will be element with the same key, therefore a coalition is created.
- Worst-Case Retrieval: time complexity of retrieving a element.
- Load Factor: How much data is in the table  

$$\frac{\text{elements}}{\text{slots}}$$
- Rehashing: make the hashtable more balanced.

### Collision Resolution by Chaining

- Most commonly used collision resolution
- Let each array slot (also called a bin) hold a list of elements (called a chain).
- In other words, When collision then add it to a list in that element

### Collision Resolution by Open Addressing

- Start with a table with each element is  $\perp$  previously used  $\Delta$ .
- Probing: is a function to insert items in a hashtable therefore resolves collisions
- Types of probing:
  - Linear probing:  $f(i) = i$ .
  - Quadratic probing:  $f(i) = c_2 \cdot i^2 + c_1 \cdot i$ , where  $c_2 \neq 0$ .
  - Double hashing:  $f(i) = i \cdot h''(i)$ , where  $h''$  is another hash function.
- Inserting with Linear Probing: Insert it to the next available key
- Deleting with Linear Probing: Ignores  $\perp$  and  $\Delta$  index will change.

## 2.7.7 Graphs

### Types of graphs

- list
- tree

- forest
- Directed Acyclic Graph (DAG)

### Treminolage

- Node, vertex (plural: vertices)
- Edge connects two nodes.
- Self-loop edge from node to itself
- Adjacent nodes connected by an edge
- Degree number of edges from or to a node
- In-degree number of edges to a node
- Out-degree number of edges from a node

### Representation

- **Adjacency Matrix** — a 2-dimensional array  $A$  of 0/1 values, with  $A[i, j]$  containing the number of edges between nodes  $i$  and  $j$  (undirected graph), or from node  $i$  to node  $j$ 
  - + edge existence testing in  $\theta(1)$  time
  - finding next outgoing edge in  $O(|V|)$  time
  - + compact representation for dense graphs (when  $|E|$  is close to  $|V|^2$ )
- **Adjacency List** — a 1-dimensional array  $Adj$  of adjacency lists, with  $Adj[i]$  containing a list of the nodes adjacent to node  $i$ .
  - + finding next outgoing edge in  $\theta(1)$  time
  - edge existence testing in  $O(|V|)$  time
  - + compact representation for sparse graphs (when  $|E|$  is much smaller than  $|V|^2$ )
- **Edge List** — a list of tuples,  $(i, j)$ , for each edge  $(i, j)$  (plus a list of the nodes).
  - edge existence test in  $\theta(|E|)$ .
  - finding next outgoing edge in  $O(|E|)$  (unless appropriately sorted).
  - + compact representation for sparse graphs (when  $|E|$  is much smaller than  $|V|^2$ )

### topological sort

A topological sort is a linear ordering of all the nodes in a directed acyclic.

### Algorithm

1. Select a node with in-degree 0.
2. Output it.
3. Remove it.
4. Repeat (from 1) until no nodes are left.

Total running time is  $\theta(|V| + |E|)$ .

### Graph Traversals

- Breadth-first search (BFS). Uses a queue for each grey node.  
Time complexity:  $\theta(|V| + |E|)$  — linear in the size of the graph.
- Depth-first search (DFS). Uses a stack for each (grey) node and has a rest of nodes (white).  
Time complexity:  $\theta(|V| + |E|)$  — linear in the size of the graph.

#### Breadth-First Search: Algorithm

Input: Some node A.

1. Paint A gray. Paint other nodes white. Add A to an initially empty FIFO queue of gray nodes. All grey nodes is in the queue. It is a queue not a stack so first in first out.
2. Dequeue head node, X. Paint its undiscovered (white) adjacent nodes gray and enqueue them. Paint X black. Repeat until queue is empty. For every black node add it to BFS order (black)

#### Depth-First Search: Algorithm

Input: Some node A.

### DFS(G)

1. Paint all nodes white.

2. For each node v in G: if v is (still) white, DFS-Visit(G,v). Each subsequent call to DFS-Visit in line 2 is called a restart.

### DFS(G)

1. Colour v gray.
2. For each node u adjacent to v: if u is white, DFS-Visit(G,u).

### Strongly-Connected Components

- Strongly connected component (SCC): maximal set of nodes where there is a path from each node to each other node.
- Many algorithms first divide a digraph into its SCCs, then process these SCCs separately, and finally combine the sub-solutions. (This is not divide & conquer, since a different algorithm is run on each SCC!)
- An undirected graph can be decomposed into its connected

  1. Enumerate the nodes of G in DFS finish order, starting from any node
  2. Compute the transpose G T (that is, reverse all edges)
  3. Make a DFS in G T, considering nodes in reverse finish order from original DFS
  4. Each tree in this depth-first forest is a strongly connected component

## 2.8 Important syntax

### 2.8.1 Let

```

let x = 1 in x * 2 == 2
case x of
  1 -> "Hello"
  2 -> "H"
  3 -> "Hel"

  input: 3 == "Hel"

# other examples
let f x y = x + 3 >= y + 3.1 in f 1 1 == False

f x = let g z = z+1 in g (g x)
f 1 == 3

```

```
:t div


:: Integral a => a -> a -> a

:t (/)
(/) :: Fractional a => a -> a -> a


```

## 2.8.2 IO

**monads** Is used to return an IO and uses a operation ( $\gg=$ ) that replaces do-notation

```
class Monad m where
  (">>=) :: m a -> (a -> m b) -> m b
  return :: a -> m a
```

## 2.9 Sorting Algorithms

1. Insertion Sort: One at a time
2. Bubble Sort: sort in order two a time starting from left and work to the right side.
3. Merge Sort: Divide and Conquer, split into even pieces and sort them and then merge/sort again.
4. Quicksort: Divide and Conquer, takes a pivot to split smaller and larger.



## Chapter 3

### Algebra 1

### 3.1 logik

- Utsaga: ett påstående som erhåller antigen värderna sann(S) eller falsk(F) vilket ge en stluten utsaga eller ej ett sannings värde vilket kallas för öpna utsagor
- Kombinerade utsagor:
- Kunjuktion utsagor: består av två utsagor som vi kallar  $A$  och  $B$
- and:  $\wedge(A \wedge B)$
- Dissjunktion utsagor: består av två utsagor som vi kallar A eller B
- or:  $\vee(A \vee B)$
- icke utsaga A (motsatsen):  $\neg A$
- Alla:  $\forall$
- Minst en:  $\exists$
- $\neg(\forall x : A) \Leftrightarrow \exists x : \neg A$
- $\neg(\exists x : A) \Leftrightarrow \forall x : \neg A$

**Exempel:**

Alla reala tal  $x$  gäller att  $(x + 1)^2 = 0$

$$\forall x : (x + 1)^2 = 0$$

$$\neg(\forall x : (x + 1)^2 = 0) = \exists x : (x + 1)^2 \neq 0$$

$\neg A$ : Det finns reala tal  $x$  gäller att  $(x + 1)^2 \neq 0$

#### 3.1.1 värde tabeller

Kunjuktions värdetabel:

A	B	$A \wedge B$
S	S	S
S	F	F
F	S	F
F	F	F

Dissjuktions värdetabel:

A	B	$A \vee B$
S	S	S
S	F	S
F	S	S
F	F	F

#### 3.1.2 Implikationer

A medför B:  $A \Rightarrow B$  (implikation)

A och B medför varandra:  $A \Leftrightarrow B$  (ekvivalens)

**Implication värdetabell:**

A	B	$A \Rightarrow B$
S	S	S
S	F	F
F	S	S
F	F	S

**Ekvivalens värdetabel:**

A	B	$A \Leftrightarrow B$
S	S	S
S	F	F
F	S	F
F	F	S

**Exempel1:**

$$x = \sqrt{6 - x} \quad (3.1)$$

$$x = \sqrt{6 - x} \Rightarrow x^2 = 6 - x \Leftrightarrow x^2 + x - 6 = 0$$

$$\text{pq-formeln: } x = -\frac{1}{2} \pm \sqrt{\frac{1}{4} + \frac{24}{4}} \\ \Leftrightarrow (x = -3) \vee (x = 2)$$

Eftersom det är en implikation är höger och inte ekvivalens så behöver inte rötterna vara sanna

Testar för falska rötter:

$$2 = \sqrt{6 - 2} \text{ Sann}$$

$$2 = \sqrt{6 - (-3)} \text{ Falsk } 2 \neq \sqrt{6 - (-3)}$$

**Exempel2:**

$$(x + 2)(x + 1) = 2x(x + 1) \quad (3.2)$$

$$(x + 2)(x + 1) = 2x(x + 1) \text{ is not}$$

$$\Rightarrow (x + 2) = 2x \text{ (x=0 is not allowed)}$$

Insted do ass following:

$$(x + 2)(x + 1) = 2x(x + 1)$$

$$\Leftrightarrow (x + 2)(x + 1) - 2x(x + 1) = 0$$

$$\Leftrightarrow (x + 1)(x + 2 - 2x) = 0$$

$$\Leftrightarrow (x + 1 = 0) \vee (2 - x = 0)$$

svar ej: x=1 och x=2

svar: x=1 eller x=2

svar:  $x = 1 \vee x = 2$

## 3.2 Mängder

Naturliga tal:  $\mathbb{N} = \{0, 1, 2, 3, \dots\}$

Heltalet:  $\mathbb{Z} = \{\dots, -2, 1, 0, 1, 2, \dots\}$

Rationella tal:  $\mathbb{Q} = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0 \right\}$

Irrationella tal:  $\mathbb{P} = \mathbb{R} \setminus \mathbb{Q}$  eller  $\{x \mid x \in \mathbb{R}, x \notin \mathbb{Q}\}$

Reella tal:  $\mathbb{R} = \mathbb{P} \cup \mathbb{Q}$

$$A \cup B = \{x : (x \in A) \vee (x \in B)\}$$

$$A \cap B = \{x : (x \in A) \wedge (x \in B)\}$$

$$A \setminus B = \{x : (x \in A) \wedge (x \notin B)\}$$

$$A^\# = \{x : (x \in X) \wedge (x \notin A)\}$$

### Exempel:

$$\text{Bevisa: } X \setminus (A \cup B) = (X \setminus A) \wedge (X \setminus B)$$

$$\begin{aligned} x \in (X \setminus (A \cup B)) &\Rightarrow (x \in X) \wedge (x \notin (A \cup B)) \\ &\Rightarrow (x \in X) \wedge (x \notin A) \wedge (x \notin B) \\ &\Rightarrow (x \in X \setminus A) \wedge (x \in X \setminus B) \\ &\Rightarrow x \in (X \setminus A) \wedge (X \setminus B) \\ &\Rightarrow X \setminus (A \cup B) \subseteq (X \setminus A) \wedge (X \setminus B) \end{aligned}$$

## 3.3 Bevis

### 3.3.1 Induktions bevis

- steg 1: bevisa att det gäller för basfallet.
- Steg 2: Bevisa att  $p \Rightarrow p'$ .
  - a. Antar att det stämmer för  $p$ .
  - b. Vissar att med att stoppa in antagandet i  $p+1$  så blir  $h_l = v_l$ .

Tips: Förenkla  $h_l$  först och sedan  $v_l$  på klad papper fram och tillbaka.

### Exempel Recursion:

$$a_1 = 2, a_{n+1} = \frac{7a_n}{7-a_n}, n \in \mathbb{N}$$

$$\text{Vissa med induktion att } a_{n+1} = \frac{14}{7-2n}$$

Bevis med induktions

steg 1: visar att påståendet som vi kallar p gäller för basfallet ( $n=1$ )

$$VL_1 : a_{1+1} = a_2 = \frac{7 \cdot 2}{7-2} = \frac{14}{5},$$

$$HL_1 : a_{1+1} = a_2 = \frac{14}{7-2} = \frac{14}{5}$$

steg 2: visar att  $p_m \Rightarrow p_{m+1}$

steg 2a: antar att  $p_m$  gäller

$$a_{m+1} = \frac{14}{7-2m}$$

steg 2b: bevisar attt  $p_m \Rightarrow p_{m+1}$

genom att använda antagandet

$$\begin{aligned} VL_{m+1} a_{m+2} &= \frac{7a_{m+1}}{7-a_{m+1}} = \frac{7 \frac{14}{7-2m}}{7-\frac{14}{7-2m}} \\ &= \frac{\frac{7 \cdot 14}{7-2m}}{\frac{7(7-2m)-14}{7-2m}} = \frac{7 \cdot 14}{7(7-2m+2)} = \frac{14}{7-2(m+1)} \end{aligned}$$

$$HL_{m+1} : \frac{14}{7-2(m+1)}$$

Enligt induktionsprincipen är  $p_m$  sann för alla  $n = 1, 2, 3, \dots$  VSB

### 3.3.2 Motsägelse bevis

- Steg 1: Formulera utsagan och icke utsagan
- Steg 2: Hitta en motsägelse med utsagan

Tips: Förenkla båda led, tänk på teorin vi har, bättre att gå vaga motiveringar en inga alls

Bevis med motsägelse

Antar att motsatsen är sann

## 3.4 Delbarhet

$a$  är delbar med  $b$ , altså kvoten ger ingen rest. Vi följer:  $a \mid b$

**Divitions algoritmen:**

$$a, b \in \mathbb{Z}$$

$$a \geq 0 \wedge b \geq 0$$

$$a | b \Rightarrow (q \in \mathbb{Z} : q \geq 0) \wedge (r \in \mathbb{Z} : 0 \leq r \leq a)$$

Sådant att

$$b = qa + r$$

$q$  = kvoten,  $r$  = resten

### 3.4.1 Största Gemensama Delaren (SGD)

$$SGD(a, b)$$

$$a = bq + r$$

$$0 \leq r \leq b$$

Förenkla  $\frac{114}{96}$

$$SGD(114, 96) :$$

$$114 = 1 * 96 + 18$$

$$96 = 5 * 18 + 6$$

$$18 = 3 * 6 + 0$$

$$\frac{114}{6} = 19$$

$$\frac{96}{6} = 16$$

$$\frac{19}{16}$$

**Euklides algoritm:**

$$SGD(a, b)$$

$$a = bq_1 + r_1$$

$$b = r_1q_2 + r_2$$

$$r_1 = r_2q_3 + r_3$$

$$r_2 = r_3q_4 + r_4$$

.

.

.

$$r_{k-3} = r_{k-2}q_{k-1} + r_k$$

$$r_{k-2} = r_{k-1}q_k + 0$$

$$SGD(a, b) = r_k$$

Om  $r_k = 1 \Rightarrow a \in \text{primtal} \vee b \in \text{primtal}$

$$a, b \in \mathbb{Z}$$

$$x, y \in \mathbb{Z}$$

$$SGD(a, b) = ax + by$$

**Aritmetiska fundamentalsatsen:**

$$a \in \mathbb{Z} \wedge a \geq 2 \Rightarrow$$

$\Rightarrow a$  kan endast primtalsfaktoriseras på  
ETT SÄTT

**Lemma 2.7:**

$$a \geq 2 \wedge a \notin \text{Primatal} \Rightarrow$$

$\Rightarrow q \in \text{Primatal} \wedge q | a \wedge a$  går att  
primtals faktoriera

### 3.4.2 Primtal

**Sats:** För att bestäma om tal  $a$  är ett primtal

$$a \geq 1 \wedge a \in \text{Primatal}$$

om ett tall  $p$  finns som delar  $a$  gäller  
följande

$$2 \leq p \leq \sqrt{a} \leq a$$

**Exempel:**

Bestäm om 211 är ett primtal

Om 211 är ett primtal så finns det inte en äkta delare a

$$2 \leq a \leq \sqrt{211}$$

$$\sqrt{211} \approx 16$$

$$a : \{2, 3, 5, 7, 11, 13\}$$

a kan inte vara en äktadelare

$$ax + by = C \Leftrightarrow 12x + 20y = 296$$

steg1: Testar om  $SGD(a, b) \mid c$

$$SGD(20, 12) = 4 \Rightarrow 4 \mid 296$$

steg2: delar SGD med HL och VL

$$\frac{12x - 20y}{4} = \frac{296}{4} \Leftrightarrow 3x - 5y = 74$$

$SGD(5, 3) :$

$$5 = 1 \cdot 3 + 2$$

$$3 = 1 \cdot 2 + 1$$

$$2 = 2 \cdot 1 + 0$$

steg3: Hjälp ekvation för att hittax<sub>0</sub>, y<sub>0</sub>

$$3x_0 - 5y_0 = 1$$

$$1 = 3 - 1 * 2$$

$$1 = 3 - (5 - 3)$$

$$1 = 2 \cdot 3 - 1 \cdot 5$$

$$x_0 = 2, y_0 = -1$$

steg4: almäna lösningen  $x = Cx_0 - bn, y = Cy_0 + an$

$$x = 74 \cdot 2 - 5n = 148 - 5n$$

$$y = 74 \cdot (-1) + 3n = -74 + 3n$$

steg5: hittar godtyckliga lösningar

Intervallet som n ligger i för x-termen:

$$n = 29, 28, 27, \dots, x = 148 - 5 \cdot 28 = 148 - 140 = 8$$

Intervallet som n ligger i för y-termen:

$$n = 27, 28, 29, \dots, x = -74 + 3 \cdot 28 = 74 - 84 = 10$$

$$n = 28, x = 8, y = 10$$

$$12 \cdot 8 + 20 \cdot 10 = 296$$

**Euklides algoritm:** Det finns oändligt många primtal

**Bevis:** Motsägelsebevis

Antar att det finns ändligt många primtal

$$p_1, p_2, P_3, \dots, p_n$$

$$M = \prod_{k=1}^n n + 1 \Rightarrow M > p_j, j = 1, 2, 3, \dots, n$$

Vissar att M är ett primtal

$1 < b < M \wedge b \mid M$  Där b är minsta äkta delaren av M

$$\Rightarrow M = p_1 * b \Rightarrow p_1 \mid 1 \wedge p \geq 2$$

Detta är falskt eftersom båda utsågorna kan inte vara samtidigt. Eftersom motsatsen inte fungerar resulterar det i att satsen är sann.

## 3.5 Diofantiska ekvationer

**Sats:**

$$ax + by = c \wedge a, b, c \in \mathbb{Z} a \neq 0, b \neq 0$$

$\Rightarrow$

$$ax + by = c \text{ Där } SGD(a, b) = 1$$

Har den anmäla lösningen:

$$x = Cx_0 - nb \wedge y = Cy_0 + na$$

## 3.6 Talbaser

### 3.6.1 konvertera från decimal bass till annan bas

$$175_8 = 1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0$$

**Exempel:** En lastbil lastas med 12kg packet och 20kg paket. Totalt väger lasten 296, hur många av varje packet?

### 3.6.2 konvertera från annan bas till decimal bas

$$1609_{10} = (3 \cdot 8^3 + 1 \cdot 8^2 + 1 \cdot 8^1 + 1 \cdot 8^0) = 3111_8$$

eller så kan man använda euklides algoritm

skriv 517 i talbas 3

$$517 = 172 \cdot 3 + 1$$

$$172 = 57 \cdot 3 + 1$$

$$57 = 20 \cdot 3 + 0$$

$$20 = 6 \cdot 3 + 2$$

$$6 = 2 \cdot 3 + 0$$

$$2 = 0 \cdot 3 + 2$$

Svar:  $517_{tio} = 202011_{tre}$

### 3.6.3 Andra exempel

**Exempel:** Skriv  $137_{nio}$  i bass tre

$$\begin{aligned} 137_{nio} &= 1 \cdot 9^2 + 3 \cdot 9^1 + 7 \cdot 9^0 \\ &= 1 \cdot 3^4 + 3 \cdot 3^2 + 7 \cdot 3^0 \\ &= 1 \cdot 3^4 + 1 \cdot 3^3 + 0 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0 \\ &= 11021_{tre} \end{aligned}$$

## 3.7 Functioner

**Typer av funktioner:**

- Injektion: alla element  $x$  har olika värden  $y$   
 $f : A \rightarrow B, \{\forall x \in A : x_1 \neq x_2, f(x_1) \neq f(x_2)\}$
- Surjektion: mängd  $D$  är definitions mängden  
 $\{g : C \rightarrow D, g(x) = y, (\forall y \in D \wedge \exists x \in C)\}$
- Bijektion: Injektion  $\wedge$  Surjektion

**Kareskapprodukten:**

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}$$

$$\text{Låt } A = \{1, 2, 3\} \wedge B = \{x, y, z, w\}$$

$$A \times B :$$

$$\begin{aligned} &\{(1, x), (1, y), (1, z), (1, w) \\ &(2, x), (2, y), (2, z), (2, w) \\ &(3, x), (3, y), (3, z), (3, w)\} \end{aligned}$$

### 3.7.1 Inversen

En funktions invers kan enda

$f : A \rightarrow B \wedge$  Bijektiv  $\Rightarrow f^{-1}(x)$  Finns, där

$$(1) x = f^{-1}(y) \Leftrightarrow y = f(x)$$

$$(2) D_{f^{-1}} = V_f \Leftrightarrow D_f = V_{f^{-1}}$$

$$(3) x = f^{-1}(f(x)), x \in D_f = V_{f^{-1}}$$

$$(3) y = f^{-1}(f(y)), y \in D_f = V_{f^{-1}}$$

### 3.7.2 Relatioiner

Relation:  $xRy$

Reflexiv:  $\forall x \in X : xRx$

Symetrisk:  $xRy \Rightarrow yRx, x \in X \wedge y \in X$

Transitiv:  $(xRy) \wedge (yRz) \Rightarrow xRz, \forall x, y, z \in X$

Ekvivalensrelation: Reflexiv och Symetrisk Transitiv

## 3.8 Summor

### 3.8.1 Aritmetiska summor

$$s_n = a_1 + a_2 + a_3 + \dots + a_n = \frac{n(a_1 + a_n)}{2}$$

### 3.8.2 Geometriska summor

Börjar altid med exponenten 0 och gör om summan så att den passar i följande talföljd:

$$s_n = a + ak + ak^2 + \dots + ak^{n-1} = \frac{a(k^n - 1)}{k - 1}$$

**Exempel:**

$$\sum_{k=n}^{2n} (2^k - k)$$

sätter  $f = 0 = k - n$

$$\sum_{f=0}^n (2^{f+n} - (f+n)) = 2^n * \sum_{f=0}^n (2^f) - \sum_{f=0}^n (f+n)$$

$$\frac{2^n(2^{n+1} - 1)}{2 - 1} - \frac{3n(n+1)}{2} = 2^{2n+1} - 2^n - \frac{3n(n+1)}{2}$$

## 3.9 Kongruensräkning

Räkneregler:

$$a + b \pmod{n} \equiv a \pmod{n} + b \pmod{n}$$

$$a \cdot b \pmod{n} \equiv a \pmod{n} \cdot b \pmod{n}$$

$$a^b \pmod{n} \equiv (a \pmod{n})^b$$

**Exempel:** Vilket är det minsta positiva resten som kan erhållas vid division av  $19^{18}$  med 17?

$$\begin{aligned} 19^{18} &\equiv 2^{18} \pmod{17} \equiv 2^4 \cdot 2^4 \cdot 2^4 \cdot 2^4 \cdot 2^2 \pmod{17} \\ &\equiv (-1)^4 \cdot (-1)^4 \cdot (-1)^4 \cdot (-1)^4 \cdot 2^2 \pmod{17} \end{aligned}$$

svar: resten blir 4

## 3.10 Kardinalitet

Kardinalitet eller "mäktighet" är ett sett att räkna med mängders sorlek och alla oändliga mängder har samma kardinalitet fast det är en delmängd. Naturliga tall har samma kardinalitet som reala tal trots att naturliga tal är en del mängd av reala talen

Låt  $A$  och  $B$  vara mängder. Vi säger att  $A$  och  $B$  har samma kardinalitet då det finns en bijektion  $\exists f : A \rightarrow B, A \sim B$ . Vi säger att  $A$  står i relation med  $B$  omm  $A$  och  $B$  har samma kardinalitet  $ARB$

### 3.10.1 Uppräkneligamängder

En mängd  $X$  sägs vara uppräknelig omm  $X$  har samma kardinalitet som  $\mathbb{N}$   $\exists g : \mathbb{N} \rightarrow X$  där  $g$  är bijektiv. Exempel på uppräknliga mängder är  $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \{1, 5, 78\}$  Exempel på ej uppräknliga mängder  $\mathbb{R}, (0, 1)$

## 3.11 Polynom

### 3.11.1 Polynom division

- Triviala delare: ej heltals kvot med delaren, har en kostant sådant  $\lambda \cdot f, \lambda \notin \mathbb{Z}$
- Äkta delare: heltals kvot med delaren  $f(x), \exists a \in \text{polynom}: a | f(x)$

- Irreducible: om polynomet endast har triviala delare det finns lösningar heltaslösningar  $f(x) = 0$

- Reducible: om polynomet har äkta delare.

- Multiplisitet: vilken grad polynomet har.

**Exempel:** Man vet att ekvationen  $z^4 - 2z^3 - 7z^2 + 26z - 20 = 0$  har roten  $z = 2 + i$  Lös ekvationen fullständigt.  $z = 2 + i$  Är en lösning är också konjugatet en lösning enligt faktorsatsen  $\bar{z} = a - bi$ .  $z = 2 \pm i$  Vilket betyder att följande går att factorisera ut polynomet.  $(z - (2 + i))(z - (2 - i)) = z^2 - 4z + 5$

**Långdivision (liggande stolen):**

$$\begin{array}{r} x^2 + 2x - 4z^2 + 2z - \\ x^2 - 4x + 5 \overline{\underline{-x^4 - 2x^3 - 7x^2 + 26x - 20}} \\ \quad -x^4 + 4x^3 - 5x^2 \\ \hline \quad 2x^3 - 12x^2 + 26x \\ \quad -2x^3 + 8x^2 - 10x \\ \hline \quad -4x^2 + 16x - 20 \\ \quad 4x^2 - 16x + 20 \\ \hline 0 \end{array}$$

$4 = 0$  Är också en lösning som till slut ger följande.  $z = -1 \pm \sqrt{5} \wedge z = 2 \pm i$ . Varge n grads polynom har alltid n stycken komplexa lösningar.

### 3.11.2 Faktorsatsen

$$f(x) = (x - \alpha)p(x)$$

**Exempel, Gemensam root hos två polynom:**

$$p(x) = x^4 - x^3 + x^2 + 2 = 0,$$

$$g(x) = x^3 + 4x^2 + 4x + 3 = 0$$

Har en Gemensam root

Eftersom polynomen har en Gemensam root vet vi att det finns en gemensam  $(x - \alpha)$

$$f(x) = (x - \alpha)p_1(x)$$

$$g(x) = (x - \alpha)p_2(x)$$

euklides algoritm:

$$\begin{aligned} f(x) &= (x - 5)g(x) + 17(x^2 + x + 1) \\ g(x) &= \left(\frac{1}{17}(x + 3)\right)(17(x^2 + x + 1)) \\ &\quad + 17(x^2 + x + 1) + 0 \end{aligned}$$

$$\begin{aligned} h(x) &= x^2 + x + 1 \\ f(x) &= (x - 5)(x + 3)h(x) + 17h(x) \\ &= h(x)((x - 5)(x + 3) + 17) \\ &= h(x)(x^2 - 2x + 2) \\ g(x) &= (x + 3)h(x) \end{aligned}$$

$$\begin{aligned} f(x) = 0 &\Leftrightarrow (h(x) = 0 \vee x^2 - 2x + 2 = 0) \\ h(x) = 0 &\Rightarrow x = -\frac{1}{2} \pm \frac{\sqrt{3}}{2}i \\ x^2 + 2x + 2 = 0 &\Rightarrow x = 1 \pm \sqrt{1 - 2} = 1 \pm i \end{aligned}$$

**Exempel, Heltalslösning med okänd konstant:**

$x^3 + bx^2 - 7x - 7 = 0$  har en heltals lösning

Eftersom ekvationen har en heltals lösning  
vet vi att  $\alpha \mid 7$

Kandidaterna är  $\{1, -1, 7, -7\}$

$$I(x = 1) 1 + b \cdot 1 - 7 - 7 = 0 \Rightarrow b = 13 \in \mathbb{Z}$$

$$II(x = -1) -1 + b \cdot -1 + 7 - 7 = 0 \Rightarrow b = 1 \in \mathbb{Z}$$

$$III(x = 7) 7 + b \cdot 7 - 49 - 7 = 0 \Rightarrow b = - \notin \mathbb{Z}$$

$$IV(x = -7) 7 + b \cdot -7 + 49 - 7 = 0 \Rightarrow b = - \notin \mathbb{Z}$$

svar:  $b=13$ ,  $b=1$

**Exempel, Rationell root:**

$g(x) = 2x^3 + 2x^2 + 2x + 3 = 0$  har en rationell root  
Eftersom ploynomet har en ratiionel root vet vi att:

$$\frac{p}{q}, p, q \in \mathbb{Z}, SGD(p, q) = 1, p \mid 3 \wedge q \mid 2 (a_0)$$

Det möjliga delarna är  $p = \pm 1, \pm 3, q = \pm 1, \pm 2$

$x = -\frac{3}{2}$  är den enda av kandidaterna som ger en  
sann root

Vilket vi löser genom polynom divition

**Exempel, Renimagnär root:**

$z^4 + 4z^3 + 8z^2 + 12z + 15 = 0$  har en renimagnär root

Eftersom ploynomet har en rentimagnär root vet vi att  
 $z = bi, b \in \mathbb{R}, b \neq 0$

$$\begin{aligned} 0 &= b^4i^4 + 4b^3i^3 + 8b^2i^2 + 12bi + 15 \\ &= b^4 - 4b^3i - 8b^2 + 12bi + 15 \\ &= (b^4 - 8b^2 + 15) + (12b - 4b^3)i \\ &= 0 = b^4 - 8b^2 + 15 \\ 0 = 12b - 4b^3 &= 4b(3 - b^2) \Rightarrow b = 0, b = \sqrt{3}, b = -\sqrt{3} \\ b = 0 &\text{ är ej en giltig lösning.} \end{aligned}$$

Enlight faktorsatsen får vi följande

$$p(z) = (z - \sqrt{3}i)(z + \sqrt{3}i)Q(z) = (z^2 + 3)Q(z)$$

Vilket vi löser genom polynom divition

**Exempel, Dubbel root:**

$p(t) = t^3 - 5t^2 + 3t + 9 = 0$  har en dubbel root

Eftersom ploynomet har en dubbel root vet vi att  
derivatan ger os en root  $p'(t) = 0$

Att räkna ut derivatan gör man genm formeln

$$ax^b \Rightarrow b \cdot ax^{b-1}$$

$$p'(t) = 3t^2 - 10t + 3 = 0 \Rightarrow t_1 = 3, t_2 = \frac{1}{3}$$

dock är sista en falsk root

$$p(t) = x - 3^2Q(t)$$

Vilket vi löser genom polynom divition

**Exempel, SGD polynom:**

$$\begin{aligned} \text{Förenkla } &\frac{x^4 + x^3 + 2x - 4}{x^4 - x^3 - 2x - 4} \\ SGD(x^4 + x^3 + 2x - 4, x^4 - x^3 - 2x - 4) : & \end{aligned}$$

$$x^4 + x^3 + 2x - 4 = 1 \cdot x^4 - x^3 - 2x - 4 + (2x^3 + 4x)$$

$$x^4 - x^3 - 2x - 4 = \frac{x}{2} - \frac{1}{2} \cdot (2x^3 + 4x) + (-2x^2 - 4)$$

med polynom divition

$$2x^3 + 4x = -x \cdot (-2x^2 - 4) + 0$$

Med SGD så kan vi ta ett polynom som är hjämt delbart  
ex:  $-2(x^2 + x)$

$$\text{polynom divition: } \frac{x^4 + x^3 + 2x - 4}{x^2 + x} = x^2 + x - 2$$

$$\text{polynom divition: } \frac{x^4 - x^3 - 2x - 4}{x^2 + x} = x^2 - x - 2$$

$$\frac{x^4 + x^3 + 2x - 4}{x^4 - x^3 - 2x - 4} = \frac{x^2 + x - 2}{x^2 - x - 2}$$

## Chapter 4

# Single Variable Calculus

## 4.1 Basic

### 4.1.1 Mängder

Naturliga tal:  $\mathbb{N} = \{0, 1, 2, 3..\}$  or  $\{1, 2, 3..\}$

Heltal:  $\mathbb{Z} = \{\dots -2, 1, 0, 1, 2..\}$

Rationella tal:  $\mathbb{Q} = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z} \right\}$

Irrationella tal:  $\mathbb{P} = \mathbb{R} \setminus \mathbb{Q}$  eller  $\{x \mid x \in \mathbb{R}, x \notin \mathbb{Q}\}$

Reella tal:  $\mathbb{R} = \mathbb{P} \cup \mathbb{Q}$

Värde tabell:

	$x < 0$	$0 < x < 3$	$3 < x < 5$	$5 < x$
$x-5$	-	-	-	+
-3	-	-	-	-
$x$	-	+	+	+
$x-3$	-	-	+	+
hela	+	-	+	-

### 4.1.3 Funktion

- $f : A \rightarrow B$
- $A$ : domain/definitionsmängd
- $B$ : Målmängd/kodomängd/värdemängd
- Injektion: alla element  $x$  har olika värden  $y$   
 $f : A \rightarrow B, \{\forall x \in A : x_1 \neq x_2, f(x_1) \neq f(x_2)\}$
- Surjektion: mängd  $D$  är definitions mängden  
 $\{g : C \rightarrow D, g(x) = y, (\forall y \in D \wedge \exists x \in C)\}$
- Bijektion: Injektion  $\wedge$  Surjektion
- $f \circ g = f(g(x))$
- Begränsad: funktionen är inom ett intervall, altså nedåt och uppåt
- Begränsat nedåt: funktionen är endast begrensat nedåt
- Begränsat uppåt: funktionen är endast begrensat uppåt
- Jäm function: altså är den symetrisk, ex:  
 $x^2, f(-x) = f(x)$
- Udda function: altså är den spefälvänt symetrisk, ex:  
 $x^3, f(-x) = -f(x)$
- Polynom  $p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$   
 $= \sum_{i=0}^n (a_i x^i)$
- Rationell funktion:  $R(x) = \frac{P(x)}{Q(x)}$ , ex:  $x^{-1} = \frac{1}{x}$
- ej polynom men rationell funktion

### 4.1.2 Intervall

Open intervall =  $(1, 4)$

Closed intervall =  $[1, 4]$

$$[a, b] = \{x \mid a \leq x \leq b\} \quad (4.1)$$

$$[a, \infty[ \quad (4.2)$$

$$]-\infty, \infty[ \quad (4.3)$$

#### Exempel: Olikheter och intervall

$$\frac{2}{x-3} < \frac{5}{x} \quad (4.4)$$

$$\frac{2}{x-3} - \frac{5}{x} < 0$$

$$\frac{(x)2}{x(x-3)} - \frac{5(x-3)}{x(x-3)} < 0$$

$$\frac{2x - 5x + 15}{x(x-3)} < 0$$

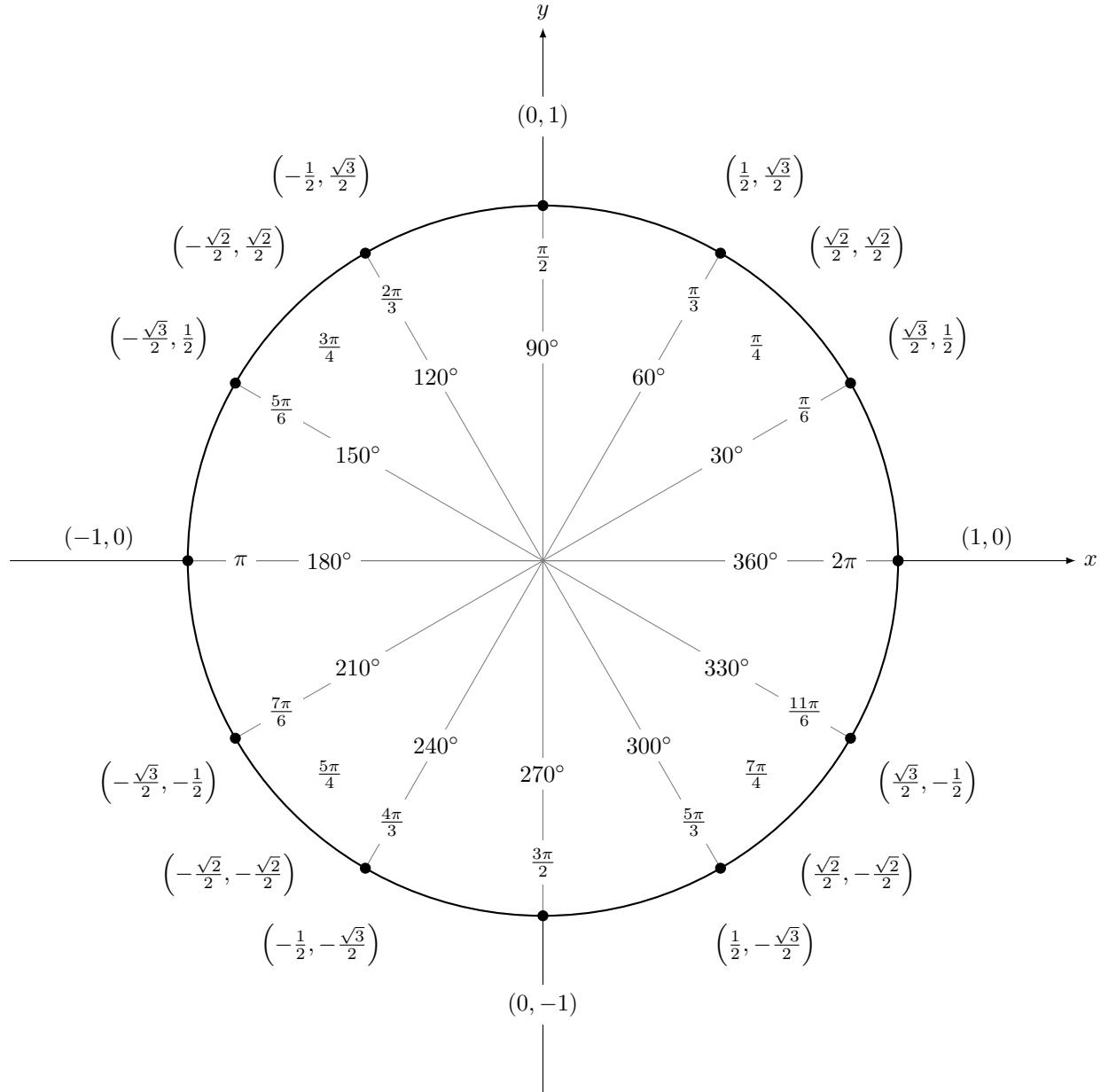
$$\frac{-3x + 15}{x(x-3)} < 0$$

$$\frac{-3(x-5)}{x(x-3)} < 0$$

$$x \neq 0, x \neq 3$$

#### 4.1.4 Trigonometri

$\sin$	0	$\frac{\sqrt{0}}{2}$	$\frac{\sqrt{1}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{4}}{2}$
$\cos$	$\frac{\sqrt{4}}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{1}}{2}$	$\frac{\sqrt{0}}{2}$	



**Sats:**

$$360^\circ = 2\pi \text{rad}$$

$$v_g = v_r \cdot \frac{180^\circ}{\pi}$$

$$v_r = v_g \cdot \frac{\pi}{180^\circ}$$

**Sats:**

$$-1 \leq \sin t \leq 1$$

$$-1 \leq \cos t \leq 1$$

**Sats:**

$$\cos(-t) = \cos(t)$$

$$\sin(-t) = -\sin(t)$$

$$\tan(-t) = \frac{\sin(-t)}{\cos(-t)} = \frac{-\sin(t)}{\cos(t)}$$

**Additionsformlerna:**

$$\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha)$$

$$\sin(\alpha - \beta) = \sin(\alpha)\cos(\beta) - \sin(\beta)\cos(\alpha)$$

$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\beta)\sin(\alpha)$$

$$\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\beta)\sin(\alpha)$$

**Trigonometriska ettan:**

$$(\sin t)^2 + (\cos t)^2 = 1$$

$$\sin^2 t + \cos^2 t = 1$$

### 4.1.5 Exempel: Trigonometri

$$\begin{aligned} \cos \frac{\pi}{12} &= \cos \left( \frac{\pi}{3} - \frac{\pi}{4} \right) = \cos \frac{\pi}{3} \cos \frac{\pi}{4} + \sin \frac{\pi}{3} \sin \frac{\pi}{4} \\ &= \left( \frac{1}{2} \right) \left( \frac{1}{\sqrt{2}} \right) + \left( \frac{\sqrt{3}}{2} \right) \left( \frac{1}{\sqrt{2}} \right) = \frac{1 + \sqrt{3}}{2\sqrt{2}} \end{aligned}$$

## 4.2 Gränsvärden

$$\lim_{x \rightarrow x_0} f = L \wedge \lim_{x \rightarrow x_0} g = M$$

$$\Rightarrow \lim_{x \rightarrow x_0} (f + g) = L + M$$

Squeeze therum:  $h(x) \leq f(x) \leq g(x)$ ,

$$\lim_{x \rightarrow x_0} h(x) = \lim_{x \rightarrow x_0} g(x_0) = L$$

$$\Rightarrow \lim_{x \rightarrow x_0} f(x) = L$$

$$\lim_{x \rightarrow 0} \frac{\sin(x)}{x} = 1$$

$\lim_{x \rightarrow \infty} \sin(x)$  är ej definierad

### Example: Limit för sin

$$\text{Lös: } \lim_{x \rightarrow 0} \frac{\sin(2x^2)}{x^2}$$

$$\lim_{x \rightarrow 0} \frac{2 \cdot \sin(2x^2)}{2x^2} = 2 \cdot \lim_{x \rightarrow 0} \frac{\sin(2x^2)}{2x^2} = 2$$

### Example: 0/0

$$\text{Lös: } \lim_{x \rightarrow 0} \frac{x \cdot \left( \cos(x) + \frac{\sin(x)}{x} \right)}{x + x^2}$$

$$\lim_{x \rightarrow 0} \frac{x \cdot \left( \cos(x) + \frac{\sin(x)}{x} \right)}{x(1+x)} = \frac{1+1}{1} = 2$$

### Example: roten ur i nämnaren

$$\text{Lös: } \lim_{x \rightarrow 0} \frac{x}{\sqrt{x+9}-3}$$

$$\lim_{x \rightarrow 0} \frac{x(\sqrt{x+9}+3)}{(\sqrt{x+9}-3)(\sqrt{x+9}+3)}$$

$$= \lim_{x \rightarrow 0} \frac{x(\sqrt{x+9}+3)}{x+9-9} = \lim_{x \rightarrow 0} \sqrt{x+9} + 3 = 6$$

### Example: roten ur i nämnaren

$$\text{Lös: } \lim_{x \rightarrow -\infty} \frac{2x-1}{\sqrt{3x^2+x+1}}$$

$$\lim_{x \rightarrow -\infty} \frac{x(2-\frac{1}{x})}{\sqrt{x^2(3+\frac{1}{x}+\frac{1}{x^2})}}$$

$$= \lim_{x \rightarrow -\infty} \frac{x(2-\frac{1}{x})}{x\sqrt{3+\frac{1}{x}+\frac{1}{x^2}}} = \frac{-2}{\sqrt{3}}$$

**Example: infinity sin and squeeze**

Lös:  $\lim_{x \rightarrow \infty} \frac{\sin(x)}{x}$

$$\begin{aligned} \left| \frac{\sin(x)}{x} \right| &= \frac{|\sin(x)|}{|x|} \leq \frac{1}{|x|} \\ \Rightarrow \text{squeeze } \lim_{x \rightarrow \infty} \left| \frac{\sin(x)}{x} \right| &= 0 \\ \Rightarrow \lim_{x \rightarrow \infty} \frac{\sin(x)}{x} &= 0 \end{aligned}$$

**4.2.1 Kontinuitet**

En funktion är kontinuerlig i punkt  $x_0$ ,  $\forall \varepsilon \wedge \exists \delta : |x - x_0| < \delta \Leftrightarrow |f(x) - f(x_0)| < \varepsilon$ . Alltså så finns det inga hopp i funktionen. Man kan dra penan på grafen utan att släppa om funktionen bestor av flera uttryck är funktionen kontinuerlig om alla utrycken är det.

**4.3 Derivator**

- Tangent: linjen som har samma lutning i en punkt som funktionens derivata  $y - f(x_0) = f'(x_0)(x - x_0) \Rightarrow y = ax + b$
- Samband: Deriverbar  $\Rightarrow$  Kontinuerlig

**Räkneregler**

$f$	$f'$
$c$	0
$x$	1
$x^n$	$nx^{n-1}$
$a^x$	$a^x \ln a$
$e^{kx}$	$ke^x$
$\sin x$	$\cos x$
$\cos x$	$-\sin x$
$\tan x$	$\frac{1}{\cos^2 x}$
$\ln x$	$\frac{1}{x}$

$$(f + g)' = f' + g'$$

$$(cf)' = cf'$$

$$(fg)' = f'g + fg' \text{ Produktregeln}$$

$$\left( \frac{f}{g} \right)' = \frac{f'g - fg'}{g^2} \text{ Kvotregeln, } g \neq 0$$

**4.3.1 Kjedje regeln**

Om funktionen  $f$  är deriverbar i  $x$  och funktionen är deriverbar ig  $g(x)$

$$\begin{aligned} \Rightarrow h(x) &= f \circ g = f(g(x)) \text{ deriverbar i } x \\ h'(x) &= (f(g(x)))' = f'(g(x))g'(x) \end{aligned}$$

**Example: Kjedje regeln**

Lös:  $(x^x)'$

$$\begin{aligned} (x^x)' &= (e^{x \ln x})' = e^{x \ln x} \left( \ln x + x \frac{1}{x} \right) \\ &= e^{x \ln x} (1 + \ln x) = x^x (1 + \ln x) \end{aligned}$$

**4.3.2 L'Hôpital's rule**

$$\begin{aligned} f, g : \mathbb{R} \rightarrow \mathbb{R} \wedge \left( \lim_{x \rightarrow x_0} \frac{f}{g} = \frac{0}{0} \vee'' \lim_{x \rightarrow x_0} \frac{f}{g} = \frac{\pm \infty}{\pm \infty} \right) \\ \Rightarrow \lim_{x \rightarrow x_0} \frac{f}{g} = \lim_{x \rightarrow x_0} \frac{f'}{g'} \end{aligned}$$

**Example: L'Hôpital's rule**

$$\begin{aligned} \text{Lös: } \lim_{x \rightarrow 1} \frac{x^4 - 6x^3 + 5x^2}{x^3 - 8x^2 + 7x} \\ \lim_{x \rightarrow 1} \frac{x^4 - 6x^3 + 5x^2}{x^3 - 8x^2 + 7x} ='' \frac{0}{0} \\ \text{L'Hôpital's rule} \Rightarrow \lim_{x \rightarrow 1} \frac{4x^3 - 18x^2 + 10x}{3x^2 - 16x + 7} \\ = \frac{-4}{-6} = \frac{2}{3} \end{aligned}$$

**4.3.3 Medelvärdessatsen**

Anta att  $f$  är kontinuerlig på  $[a, b]$  och deriverbar på  $(a, b)$   $\Rightarrow \exists c \in (a, b)$

så att  $\frac{f(b) - f(a)}{b - a} = f'(c)$

**Example: Medelvärdessatsen**

Bevisa att för varje  $a > b$  gäller följande

$$|\cos a - \cos b| \leq |a - b|$$

$f(x) = \cos x$  är kont och deriverbar i  $\mathbb{R} \wedge [a, b]$

$$\text{MVS } \Rightarrow \exists c \in (a, b) : f'(c) = \frac{f(b) - (a)}{b - a}$$

$$= \frac{\cos(b) - \cos(a)}{b - a} = -\sin(c)$$

$$\Rightarrow \left| \frac{\cos(b) - \cos(a)}{b - a} \right| = |- \sin(c)|$$

$$\Rightarrow \frac{|\cos(b) - \cos(a)|}{|b - a|} = |- \sin(c)| \leq 1$$

$$\frac{|\cos(b) - \cos(a)|}{|b - a|} \leq 1$$

$$\Rightarrow |\cos(b) - \cos(a)| \leq |a - b|$$

**4.3.4 Rolle**

Anta att  $f$  är kontinuerlig på  $[a, b]$  och deriverbar på  $(a, b)$ . Om  $f(a) = f(b) \Rightarrow \exists c \in (a, b) : f'(c) = 0$ .

**4.3.5 växande funktioner**

- växande:  $x_1 < x_2 \Rightarrow f(x_1) \leq f(x_2)$
- strängt växande:  $x_1 < x_2 \Rightarrow f(x_1) < f(x_2)$
- avtagande:  $x_1 < x_2 \Rightarrow f(x_1) \geq f(x_2)$
- stänkt avtagande:  $x_1 < x_2 \Rightarrow f(x_1) > f(x_2)$

**4.3.6 Högreordnings derivator**

- andragrads derivator:  $(f')' = f'' = \frac{d^2 f}{dx^2}$
- tredjegrads derivator:  $(f'')' = f''' = \frac{d^3 f}{dx^3}$
- ntegrads derivator:  $f^{(n)} = f^n = \frac{d^n f}{dx^n}$

**4.3.7 Impericit derivering**

Deriverar båda sidorna om modifierat.

**Example: Impericit derivering**

Bestäm tangenten till  $x^3 + y^3 = 6xy$  i  $(3, 3)$

$$(3^3 + 3^3 = 6 \cdot 3 \cdot 3) \text{-sant}$$

$$y - y_0 = f'(x_0)(x - x_0) \Rightarrow y - 3 = f'(3)(x - 3)$$

$$\text{Implicit derivering: } 3x^2 + 3y^2 y' = 6y + 6xy'$$

$$\Rightarrow 3x^2 y' - 6xy' = 6y - 6x^2$$

$$\Rightarrow y'(3x^2 - 6x) = 6y - 6x^2 \Rightarrow y' = \frac{6y - 6x^2}{3x^2 - 6x}$$

$$\Rightarrow y'(3) = \frac{63 - 3 \cdot 3^2}{3 \cdot 3^2 - 6 \cdot 3} = -1$$

$$\Rightarrow y - 3 = -(x - 3)$$

$$\Rightarrow x + y = 6 \vee y = -x + 6$$

**4.3.8 invers funktioner**

$f : A \rightarrow B \wedge \text{Bijektiv} \Rightarrow f^{-1}(x)$  Finns, där

$$(1) x = f^{-1}(y) \Leftrightarrow y = f(x)$$

$$(2) D_{f^{-1}} = V_f \Leftrightarrow D_f = V_{f^{-1}}$$

$$(3) x = f^{-1}(f(x)), x \in D_f = V_{f^{-1}}$$

$$(3) y = f^{-1}(f(y)), y \in D_f = V_{f^{-1}}$$

### 4.3.9 exponential och logaritm

$$\frac{a^{-3}}{b} = \frac{b^3}{a}$$

$$\sqrt{a} = a^{\frac{1}{2}}$$

$$a^{x+y} = a^x a^y$$

$$(a^x)^y = a^{xy}$$

$$a^{-x} = \frac{1}{a^x}$$

$$a^{\frac{m}{n}} = \sqrt[n]{a^m}$$

$$(1): b = a^x \Leftrightarrow \log_a(b)$$

= x för:  $a > 0, b > 0, a \neq 1$

$$(2): \log_a\left(\frac{b}{c}\right) = \log_a(b) - \log_a(c)$$

$$(3): \log_a(b \cdot c) = \log_a(b) + \log_a(c)$$

$$(4): \log_a(b^d) = d \log_a(b)$$

$$(5): \log_a(b) = \frac{\log_f(b)}{\log_f(a)}$$

$$(6): \log_a(a) = 1$$

$$(7): \log_a(1) = 0$$

$$(8): a^{\log_a(x)} = x$$

$$(9): \log_{a^c}(b) = \frac{1}{c} \log_a(b)$$

### 4.3.10 odefinerad form

$$\frac{0}{0}, \infty \cdot 0, 1^\infty, \infty^0, 0^0$$

### 4.3.11 inversa trigometriska funktioner

#### arcsin

$$\cos : \left[ \frac{-\pi}{2}, \frac{\pi}{2} \right] \rightarrow [-1, 1]$$

$$\arcsin 1 = \frac{\pi}{2}$$

$$\arcsin 0 = 0$$

$$\arcsin \pi = \text{odefinerad}$$

$$\sin(\arcsin(x)) = x$$

$$\arcsin(\sin(x)) = x$$

$$(\arcsin(x))' = \frac{1}{\sin'(\arcsin(x))} = \frac{1}{\cos(\arcsin(x))}$$

$$= \frac{1}{\sqrt{\cos^2(\arcsin(x))}} = \frac{1}{\sqrt{1 - \sin^2(\arcsin(x))}} = \frac{1}{\sqrt{1 - x^2}}$$

#### arccos

$$\cos : [0, \pi] \rightarrow [-1, 1]$$

$$\arccos 1 = 0$$

$$\arccos 0 = \frac{\pi}{2}$$

$$\arccos \pi = \text{odefinerad}$$

$$\cos(\arccos(x)) = x$$

$$\arccos(\cos(x)) = x$$

$$(\arccos(x))' = \frac{1}{\sqrt{1 - x^2}}$$

#### arctan

$$\cos : \left[ \frac{-\pi}{2}, \frac{\pi}{2} \right] \rightarrow \mathbb{R}$$

$$\tan(\arctan(x)) = x$$

$$\arctan(\tan(x)) = x$$

$$(\arctan(x))' = \frac{1}{1 + x^2}$$

**exempel**

$$\begin{aligned}\tan(\arccos x) &= \frac{\sin(\arccos x)}{\cos \arccos x} = \frac{\sqrt{1-x^2}}{x} \\ \cos(\arctan x) &= \cos \frac{\arcsin x}{\arccos x} = \frac{1}{1+x^2} \\ \sin(\arccos x) &= \sqrt{1-x^2} \\ \cos(\arcsin x) &= \sqrt{1-x^2}\end{aligned}$$

## 4.4 Grafritning

Ta reda på extrem punkterna och rita utifrån det

**Extremvärden**

- global minimum punkt:  $x_0, \forall x : f(x) \geq f(x_0)$
- lokal minimum punkt:  $x_0, \forall x \text{ nära } x_0 : f(x) \geq f(x_0)$
- global maximum punkt:  $x_0, \forall x : f(x) \leq f(x_0)$
- lokal maximum punkt:  $x_0, \forall x \text{ nära } x_0 : f(x) \leq f(x_0)$
- Kritisk punkt:  $f'(x) = 0$

**Komplexity**

- konvex: Tangent liger altid under funktionen  $y'' > 0$ .
- konkav: Tangent liger altid över funktionen  $y'' < 0$ .
- Inflektionspunkt: då funktionen byter från konvex till konkav eller tvärtom.

**Exemple:** Datan som behövs beräknas vid

**grafritning**

Rita funktionen  $y = (x^2 - 1)^3$

(1) Kollar om funktionen är Konternuelig:  
Eftersom funkrionen består av polynom är den konternuerlig

(2) Extrem punkter:

(I) Derivatan: funktionen är deriverbar

$$y' = 3(x^2 - 1)^2 \cdot 2x = 0$$

$$\Rightarrow x = 0$$

(II) Singulär punkter:

eftersom funktionen är deriverbar i alla punkter i definitions mängden

Så finns det inga singulär punkter

(III) End värdet: Eftersom funktionen är ej definerad i ett intervall så finns det inga

(3) Komplexitet:

Andra derivatan avgör om funktionen är knvex eller konkav

$$\begin{aligned}y'' &= (6x(x^2 - 1)^2)' = (6x(x^4 - 2x^2 + 1))' \\ &= (6x^5 - 12x^3 + 6x)' = 30x^4 - 36x^2 + 6 = 0 \\ &\Rightarrow t^2 - \frac{6}{5}t + \frac{1}{5} = 0\end{aligned}$$

$$\Rightarrow t = \frac{6}{10} \pm \sqrt{\frac{36}{100} - \frac{20}{100}} = \frac{6}{10} \pm \frac{4}{10}$$

$$t = 1 \vee t = \frac{1}{5} \Rightarrow x = \pm 1 \vee x = \pm \frac{1}{\sqrt{5}}$$

(4) Asymptoter:

(I) lodräta asymptoter:

$$\lim_{x \rightarrow +\infty} y = +\infty$$

$$\lim_{x \rightarrow -\infty} y = +\infty$$

(II) vågräta asymptoter:

Eftersom funktionen inte är en kvot finns det inga sådana asymptoter

$f'$	$x < -1$	$x = -1$	$-1 < x < -\frac{1}{\sqrt{5}}$	$x = -\frac{1}{\sqrt{5}}$	$\dots$
$f''$	–	–	–	–	...
$f$	avtagande kokav	0 avtagande inflektions punkt	+	0 avtagande konvex	...

## 4.5 Optemering

Sats: om funktionen  $f(x)$  är knternuerlig på det slutna intervallet  $[a, b]$  så antar det sitt största värde och minsta värde där  $\exists x_1, x_2 \in [a, b]$  så att  $f(x_1) \leq f(x) \leq f(x_2)$ .

### Exempel: Max/Min värde

Låt  $f(x) : [-1, 2] \rightarrow \mathbb{R}$ ,  $f(x) = x^3 - 3|x|$

(1) Konternuerlig:

$f$  är konternuerlig i intervallet

(2) Extrem punkter:

(I) Derivatan: funktionen är deriverbar i intervallet förutom då  $x = 0$

lät  $f$  bestå av  $f_1(x) = x^3 - 3x, x \geq 0$

$$\wedge f_1(x) = x^3 + 3x, x < 0$$

$$\Rightarrow f'_1(x) = 3x^2 - 3, x \geq 0 \wedge f'_2(x) = 3x^2 + 3, x < 0$$

$$f'_1(x) = 0 \Rightarrow x = 1 \in [1, 2], f(1) = -2$$

$$f'_2(x) = 0 \Rightarrow x = \text{Odefinerad}$$

(II) Singulär punkter:

$f$  är inte deriverbar då  $x = 0, f(0) = 0$

(III) End punkterna: Eftersom funktionen är ej definierad i ett intervall så finns det inga

$$f(-1) = -4$$

$$f(2) = 2$$

(3) Största och minsta värdet:

$$f(-1) < f(1) < f(0) < f(2)$$

Svar: Max är 2, min är -4

## 4.6 Talföljder och serier

### Def: Talföljder

En talföljd är en funktion  $a : \mathbb{N} \rightarrow \mathbb{N}$

Vi skriver  $a_n$  istället för  $a(n)$ ,  $a_2 = a(2)$

Vi säger att  $a_n \rightarrow a \in \mathbb{N}$  så är den konvergent

Vi säger att  $a_n \rightarrow \infty$  så är den divergent eller ej existerande

Konvergent kan vara begränsad uppåt eller nedåt

Talföljd kan vara växande eller avtagande

$$a_n \rightarrow a \vee b_n \rightarrow b$$

Omm  $a$  och  $b$  existerar  $a_n + b_n \rightarrow a + b$

### Exempel: Derivatan av serier

$$\text{Ange } f'(2), f(x) = \sum_{n=1}^{\infty} \frac{(x-2)^n}{n^2 2^{2n}}$$

$$\text{Anger den första termen } \left(\frac{x-2}{4}\right)' = \frac{1}{4}$$

$$f'(x) = \frac{1}{4} + \sum_{n=2}^{\infty} \frac{n(x-2)^{n-1}}{n^2 2^{2n}} \text{ Inre och yttre derivatan}$$

$$f'(2) = \frac{1}{4} + \sum_{n=2}^{\infty} \frac{n(2-2)^{n-1}}{n^2 2^{2n}} = \frac{1}{4} + 0 = \frac{1}{4}$$

**Sats:** Om  $(a_n)^\infty$  är konvergent  $\Rightarrow (a_n)$ ,  $n = 1$  är begränsad

### Sats:

Låt  $a > 0$

(I)  $a^n \rightarrow 0$  om  $a < 1$

(II)  $a^n \rightarrow +\infty$  om  $a > 1$

$$(a_n)_{n=1}^{\infty} = \sum_{k=0}^n a_n = a_1 + a_2 + a_3 + \dots$$

### Sats: Geometrisk serie

$$s_n = a + ak + ak^2 + \dots + ak^{n-1} = \frac{a(k^n - 1)}{k - 1}$$

$$\sum_{n=0}^{\infty} r^n \Rightarrow$$

Konvergent om  $|r| < 1$

Divergent om  $|r| > 1$

**Sats:**

$$\sum_{n=0}^{\infty} a_n \text{ konvergent} \Rightarrow a_n \rightarrow 0$$

**Sats: p-serie**

$$\sum_{n=0}^{\infty} \frac{1}{n^p} \Rightarrow$$

konvergent om  $\Rightarrow p > 1$ divergent om  $\Rightarrow p \leq 1$ **Sats:**Antag att  $0 \leq a_n \leq b_n$  för varje  $n \in \mathbf{N}$ 

$$\sum_{n=1}^{\infty} a_n \text{ konvergent} \Rightarrow a_n \rightarrow 0$$

$$(I) \sum_{n=1}^{\infty} b_n \text{ konvergerar} \Rightarrow \sum_{n=1}^{\infty} a_n \text{ konvergerar}$$

$$(II) \sum_{n=1}^{\infty} a_n \text{ divergerar} \Rightarrow \sum_{n=1}^{\infty} b_n \text{ divergerar}$$

**Sats:**Antag att  $a_n > 0, b_n > 0$  för varje  $n \in \mathbf{N}$ 

$$\text{och antag att } \frac{a_n}{b_n} \rightarrow L \neq 0 \wedge L < \pm\infty$$

Då gäller att serien  $\sum_{n=1}^{\infty} a_n$  och  $\sum_{n=1}^{\infty} b_n$   
är båda divergenta

**Sats: kvotkriterium**

$$\text{Låt } a_n > 0 \wedge \lim_{n \rightarrow \infty} \frac{a_{n+1}}{a_n} \rightarrow L \Rightarrow$$

$$(I) 0 \leq L \leq 1 \Rightarrow \text{Konvergerar } \sum_{n=1}^{\infty} a_n (a_n \rightarrow 0)$$

$$(II) L > 1 \Rightarrow \sum_{n=1}^{\infty} a_n \text{ divergerar } (a_n \rightarrow +\infty)$$

**Exempel: Måste kunna**

$$\begin{aligned} & \sum_{n=1}^{\infty} \frac{5^n n!}{n^n} \\ \text{Låt } a_n = \frac{5^n n!}{n^n} \Rightarrow \frac{a_{n+1}}{a_n} &= \frac{5^{n+1}(n+1)!/n^n}{5^n n!/n^n} \\ &= 5 \frac{(n+1)n^n}{(n+1)^{n+1}} = 5 \left(\frac{n}{n+1}\right)^n = \\ &= 5 \frac{1}{(1+1/n)^n} \rightarrow \frac{5}{e} > 1 \Rightarrow \text{divigerar} \end{aligned}$$

**Sats: Rotkriteriet (Ovanlig)**

$$\text{Låt } a_n > 0 \wedge \lim_{n \rightarrow \infty} \sqrt[n]{a_n} \rightarrow L \Rightarrow$$

$$(I) 0 \leq L \leq 1 \Rightarrow \text{Konvergerar } \sum_{n=1}^{\infty} a_n (a_n \rightarrow 0)$$

$$(II) L > 1 \Rightarrow \sum_{n=1}^{\infty} a_n \text{ divergerar } (a_n \rightarrow +\infty)$$

#### 4.6.1 Serier med varierande tecken

**Sats: leibinz**

Antag att  $\sum_{n=1}^{\infty} a_n$  är en alternerande serie

och att om  $|a_n| \rightarrow 0 \wedge |a_n| \geq |a_{n+1}| \Rightarrow \text{konvergent}$

**Def: absolut konvergent**

Endast om  $\sum_{n=1}^{\infty} |a_n|$  konvergerar

$\Rightarrow$  då är den absolutkonvergent

om seriern är absolutkonvergent så är den också  
konvergent behöver inte vara tvärtom

#### 4.6.2 Potensserier

**Def: Potensserier**

En potensserier kring  $x_0$  är en serie på formen

$$\sum_{n=1}^{\infty} a_n (x - x_0)^n, a_n \in \mathbb{R} \text{ (koefficienten)}$$

$x$  är en variabel

**Def: Konvergens radie**

$$T(x) = \sum_{n=1}^{\infty} \frac{f(x_0)^{(n)}}{n!} (x - x_0)^n$$

vid exempelvis grad 2 så blir det andra derivatan man räknar ut

$$R = L^{-1} = \frac{1}{L}$$

$L$  är där serien konvergerar,  $R$  är konvergens radie  
 $\sum_{n=1}^{\infty} a_n (x - x_0)^n$ ,  $a_n \in \mathbb{R}$  (koefficienten)

$x$  är en variabel

Om konvergens radien är noll så finns det inte några intervall för konvergens eller divergens

**Sats: Potensserier**

En potensserie kring  $x_0$  är en serie på formen

$$\sum_{n=1}^{\infty} a_n (x - x_0)^n a_n \in \mathbb{R} a_n$$

koficienten  $x$

är en variabel

När det står ex  $(3x + 2)^n$  så måste det stå med i  $a_n, 3^n$

**Exempel: Potensserier**

$\sum_{n=1}^{\infty} \frac{(x - 7)^n}{n(n+1)}$  är en potensserie där

$$x_0 = 7, a_n = \frac{1}{n(n+1)}$$

$$|\frac{a_{n+1}}{a_n}| = \left| \frac{1/((n+1)(n+2))}{1/(n(n+1))} \right| = \left| \frac{n}{n+2} \right| = \left| \frac{n}{n(1+2/n)} \right|$$

$$\Rightarrow \left| \frac{1}{1} \right| = 1, \text{ då } n \rightarrow \infty$$

$\Rightarrow R = \frac{1}{1} = 1 \Rightarrow$  vi får följande intervall av potenserien

Absolutkonverget för  $|x - 7| < 1 \Leftrightarrow 6 < x < 8$

Divergent för  $|x - 7| > 1 \Leftrightarrow 8 < x \vee x < 6$

Testar edge fallen  $x = 6, x = 8$

$$\sum_{n=1}^{\infty} \frac{(8-7)^n}{n(n+1)} \Rightarrow \left| \sum_{n=1}^{\infty} \frac{(1)^n}{n(n+1)} \right|,$$

$$\left| \frac{(1)^n}{n(n+1)} \right| \rightarrow 0 \Rightarrow \text{absolutkonvergerar}$$

$$\sum_{n=1}^{\infty} \frac{(6-7)^n}{n(n+1)} \Rightarrow \left| \sum_{n=1}^{\infty} \frac{(-1)^n}{n(n+1)} \right|,$$

$$\left| \frac{(-1)^n}{n(n+1)} \right| \rightarrow 0 \Rightarrow \text{absolutkonvergerar}$$

**Svar:**

Absolutkonverget för  $6 \leq x \leq 8$

Divergent för  $8 < x \vee x < 6$

**4.6.3 Taylor serier****Def: Talorpolynom**

Taylorpolynom av grad  $n \in \mathbb{N}$ , kring  $x = x_0$

$$f(x) = P_n(x) = \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k, 0! = 1, f^{(0)} = f$$

När  $x \approx x_0 \Rightarrow f(x) \approx p_n(x)$

**Sats: Talor sats**

Antag att  $f \in C^{n+1}$ .

$f(x) = p_n(x) + R_{n+1}(x)$  Där felet är  $R_{n+1}(x)$

$$R_{n+1}(x) = \frac{f^{(n+1)}(c)}{(n+1)!} (x - x_0)^{n+1}, x \vee x_0 \leq c \leq x_0 \vee x$$

**exempel: felet**

Upskata felet hos  $f(x) = \sin x$ , med grad 5

Sista termen är  $\frac{x^5}{5!} \Rightarrow$  Felet  $\frac{f^{(6)}(c)}{6!}x^6$ , ( $0 < c < 1$ )  
 $\left| \frac{f^{(6)}(c)}{6!}x^6 \right| \leq \frac{|f^{(6)}(c)|}{6!} = \frac{|\sin c|}{6!} \leq \frac{1}{6!} = \frac{1}{720}$

**Exempel: grad n vid specificerad punkt** Lös:  
 Hitta Taylorpolynomet, ordning  $2n - 1$ , för  $\sin 2x$   
 vid  $x = \frac{\pi}{2}$

$$\begin{aligned} f(x) &= \sin(2x) & f\left(\frac{\pi}{2}\right) &= \\ & & 0 & \\ f'(x) &= -2 \cos(2x) & f\left(\frac{\pi}{2}\right) &= \\ & & -2 & \\ f''(x) &= -2^2 \sin(2x) & f\left(\frac{\pi}{2}\right) &= \\ & & 0 & \\ f^{(3)}(x) &= -2 \cos(2x) & f\left(\frac{\pi}{2}\right) &= \\ & & 2^3 & \\ f^{(4)}(x) &= -2^2 \sin(2x) & f\left(\frac{\pi}{2}\right) &= \\ & & 0 & \\ f^{(5)}(x) &= -2^4 f'(x) & f\left(\frac{\pi}{2}\right) &= \\ & & -2^5 & \\ p_{2n-1}(x) &= -2(x - \frac{\pi}{2}) + \frac{2^3}{3!}(x - \frac{\pi}{2})^3 - \frac{2^5}{5!}(x - \frac{\pi}{2})^5 + \dots + \frac{(-1)^n \cdot 2^{2n-1}}{(2n-1)!}(x - \frac{\pi}{2})^{(2n-1)} & & \end{aligned}$$

**Def: maclaurin polynomial**

Taylorpolynom av grad  $n \in \mathbb{N}$ , då  $x_0 = 0$

Big-O notation: Vi säger att  $g(x) = O(f(x))$   
 när  $x \approx x_0$ , värsta fall

Man kan också säga istället för Big-O notation en rest

$$R_{n+1}(x) = x^{n+1}H(x) \text{ där } H(x) \text{ begränsad i intervallet}$$

$$e^x = \sum_{k=0}^n \frac{x^k}{k!} + O(x^{n+1})$$

$$\sin x = \sum_{k=0}^n \frac{(-1)^{k-1}}{(2k-1)!} x^{2k-1} + O(x^{2n})$$

$$\cos x = \sum_{k=0}^n \frac{(-1)^k}{(2k)!} x^{2k} + O(x^{2n+1})$$

$$\ln(1+x) = \sum_{k=0}^n \frac{x^k}{k} (x)^x + O(x^{n+1})$$

**4.7 Integraler****Regler no.1**

$$\int_a^a f(x)dx = 0$$

**Regler no.2**

$$\int_a^b f(x)dx = - \int_b^a f(x)dx$$

**Regler no.3 (Linjearitet)**

$$A, B \in \mathbb{R}$$

$$\begin{aligned} \int_a^b (A \cdot f(x)dx + B \cdot g(x)) &= A(\int_a^b f(x)dx) + B(\int_a^b g(x)dx) \\ &= A \int_a^b f(x)dx + B \int_a^b g(x)dx \end{aligned}$$

Fungerar på subtraction men INTE  
 multiplicatione eller divition!

**Regler no.4**

$$c \in [a, b]$$

$$\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx$$

**Regler no.5**

$$f(-x) = -f(x) \text{ är udda}$$

$$\int_{-a}^a f(x)dx = 0$$

$$f(-x) = f(x) \text{ är jämn}$$

$$\int_{-a}^a f(x)dx = 2 \int_0^a f(x)dx$$

**Regler no.7**

$$\left| \int_a^b f(x)dx \right| \leq \int_a^b |f(x)|dx$$

**Regler no.8**

$$f(x) \leq g(x) \Rightarrow \int_a^b f(x)dx \leq \int_a^b g(x)dx$$

**Sats: Medelvärdessatsen för integraler**

$$\begin{aligned} f : [a, b] \rightarrow \mathbb{R} &\text{ konternuelig } \wedge \exists c \in (a, b) \\ \Rightarrow \int_a^b f(x)dx &= f(c)(b - a) \end{aligned}$$

**Sats: Fanalysens huvudsats, Fundemetal theorem of calculus**

Satsen är den vi använder för att lösa integraler utan geometrisk tolkning

$$\begin{aligned} f : [a, b] \rightarrow \mathbb{R} &\text{ konternuelig} \\ \Rightarrow F(x) &= \int_a^x f(t)dt, \quad a \leq x \leq b \end{aligned}$$

**Sats:**

$$\begin{aligned} f : [a, b] \rightarrow \mathbb{R} &\text{ konternuelig} \\ \Rightarrow F &\text{ är e primitiv funktion till } f(F'(x) = f(x)) \\ \int_a^b f(x)dx &= F(b) - F(a) \\ \text{Vi skriver } \int_a^b f(x)dx &= [F(x)]_a^b = F(b) - F(a) \end{aligned}$$

**Exempel: arean mellan två grafer**

$$\begin{aligned} \int_a^b (f(x) - g(x))dx &\text{ där f är översta} \\ &\text{funktionen och g är understa} \\ \int_0^1 (x - x^2)dx &= [x^2/2 - x^3/3]_0^1 \\ &= 1/2 - 1/3 = 1/6 \end{aligned}$$

**Primitiva funktioner (Obestämda integralen)**

$\int f dx$	$F$
$\int 1 dx$	$x + c$
$\int n^n dx$	$x^{n+1}/(n+1) + c$
$\int 1/x dx$	$\ln x  + c$
$\int \sin x dx$	$-\cos x + c$
$\int \cos x dx$	$\sin x + c$
$\int e^x dx$	$e^x + c$
$\int 1/\sqrt{1-x^2} dx$	$\arcsin x + c$
$\int 1/(x^2+1) dx$	$\arctan x + c$
$\int 1/\cos^2 x dx$	$\tan x + c$

**Exempel:**

$$\begin{aligned} \int_a^b e^x dx &= [e^x]_a^b = e^b - e^a \\ \int_e^x dx &= e^x + c, \quad c \in \mathbb{R} \end{aligned}$$

**Def: medelvärdet**

$$\text{Avg} f = \frac{1}{b-a} \int_a^b f dx$$

**4.7.1 variabelsubstition**

$$\begin{aligned} \int f'(g(x))g'(x)dx &= f(g(x)) + c \\ \int_e^x dx &= e^x + c, \quad c \in \mathbb{R} \end{aligned}$$

**Exempel:**

$$\begin{aligned} \int e^{x^2} 2x dx &\\ \text{Låt } u &= x^2 \Rightarrow du = 2x dx \Rightarrow \int e^{x^2} 2x dx = \int e^u du = \\ &= e^u + c = e^{x^2} + c, \quad c \in \mathbb{R} \end{aligned}$$

### Integraler som följer mönster

Integralles som innehåller  $\sqrt{x^2 + a^2}$ ,  $a > 0$   
 $x = a \tan u \Rightarrow u = \arctan x$ ,  $-\pi/2 < u < \pi/2$

$$\Rightarrow \sqrt{x^2 + a^2} = \sqrt{a^2 + a^2 \tan^2 u}$$

$$= \sqrt{a^2(1 + \tan^2 u)} = a \sqrt{\frac{\sin^2 u + \cos^2 u}{\cos^2 u}} = a \frac{1}{\cos u}$$

Exempel:  $\int \frac{1}{(1 + 25x^2)^{3/2}} dx$

$$\int \frac{dx}{\sqrt{1 + 25x^2}^3} = \int \frac{dx}{\sqrt{25(1/25 + x^2)}^3}$$

$$= \frac{1}{5^3} \int \frac{dx}{\sqrt{1/205x^2}^3}$$

Låt  $x = 1/5 \tan u \Leftrightarrow u = \arctan 5x$

$$\Rightarrow dx = \frac{1}{5 \cos^2 u} du$$

$$\Rightarrow \frac{1}{5^3} \int \frac{dx}{\sqrt{1/205x^2}^3}$$

$$= \frac{1}{5^3} \int \frac{\cos^3 u}{1/5^3} \frac{1}{5 \cos^2 u} du = \frac{1}{5} \int \cos u du$$

$$= \frac{1}{5} \sin u + c = \frac{1}{5} \sin \arctan 5x + c$$

Kan nu förenkla med trigonometriska regler

och få:  $\frac{x}{\sqrt{1 + 25x^2}} + c$

Integralles som innehåller  $\sqrt{x^2 - a^2}$ ,  $a > 0$

$$x = \frac{a}{\cos u} \Rightarrow u = \arccos \frac{a}{x}$$

$$\Rightarrow \sqrt{x^2 - a^2} = \sqrt{\frac{a^2}{\cos^2 u} - a^2} = \sqrt{\frac{a^2 - a^2 \cos^2 u}{\cos^2 u}}$$

$$= \sqrt{\frac{a^2 \sin^2 u - a^2 \cos^2 u}{a^2 \cos^2 u}} = a |\tan u|$$

Integralles som innehåller  $\sqrt{ax + b}$   
 $ax + b = u^2$

Exempel:  $\int \frac{dx}{2 + \sqrt{x}}$

Låt:  $u^2 = x \Rightarrow 2udu = dx$

$$\Rightarrow \int \frac{2udu}{2 + u} = 2 \int \frac{u+2-2}{2+u} du = 2 \left( \int du - 2 \int \frac{du}{2+u} \right)$$

$$= 2u - 4(\ln 2 + u) + c$$

### 4.7.2 Integration av rationella funktioner

Om det står beräkna generaliserade integralen så ska man beräkna med detta Rationella funktioner:  $R(x) = \frac{P(x)}{Q(x)}$ ,  $P, Q$  är polynom.

#### Algoritm

- Polynom division om grad av täljare är större än grad av nämnare
- Faktorisera nämnaren i reala faktorer
- Partiella bråk och sedan integrera dem, ( dela upp i mindre lösbara integraler)

#### Partiella bråk

När nämnaren har en förstagradsfaktor med potens  $(x - \alpha)^n$  så ansätts termerna.

$$\frac{A_1}{x - \alpha} + \frac{A_2}{(x - \alpha)^2} + \dots + \frac{A_n}{(x - \alpha)^n}$$

Andragradsfaktor  $(x^2 + ax + b)^m$  get upphov till ansatzen

$$\frac{B_1x + C_1}{x^2 + ax + b} + \frac{B_2x + C_2}{(x^2 + ax + b)^2} + \dots + \frac{B_mx + C_m}{(x^2 + ax + b)^m}$$

$x - a$	$\frac{A}{x-a}$
$(x - a)^k$	$\frac{A_1}{x-a} + \frac{A_2}{x-a^2} + \dots + \frac{A_k}{x-a^k}$
$x^2 + bx + c$ , $(b^2 - 4c < 0)$	$\frac{Ax+B}{x^2+bx+c}$
$x^2 + bx + c$ , $(b^2 - 4c < 0)$	$\frac{A_1x+B_1}{x^2+bx+c} + \frac{A_2x+B_2}{(x^2+bx+c)^2} + \dots + \frac{A_1x+B_1}{(x^2+bx+c)^k}$

$$\begin{aligned}\int \frac{1}{ax+b} dx &= \frac{1}{a} \ln |ax+b| + c \\ \int \frac{x}{x^2+a^2} dx &= \frac{1}{2} \ln |x^2+a^2| + c \\ \int \frac{x}{x^2-a^2} dx &= \frac{1}{2} \ln |x^2-a^2| + c \\ \int \frac{dx}{x^2+a^2} &= \frac{1}{a} \tan^{-1} \frac{x}{a} + c \\ \int \frac{dx}{x^2-a^2} &= \frac{1}{2a} \ln \frac{x-a}{x+a} + c\end{aligned}$$

$$\int \frac{x^3+2}{x^2-5x+4} dx$$

1. Polynomdivision

$$P(x) = x^3 + 2, Q(x) = x^2 - 5x + 4$$

Med polynom divition så får vi kvot

$x+5$  och rest  $21x-18$

$$x^3 + 2 = (x+5)(x^2 - 5x + 4) + 21x - 18$$

2. Faktorisera nämnaren i reala faktorer

$$\begin{aligned}&\Rightarrow \int \frac{x^3+2}{x^2-5x+4} dx \\ &= \int \frac{(x+5)(x^2-5x+4) + 21x-18}{x^2-5x+4} dx \\ &= \int (x+5)dx + \int \frac{21x-18}{x^2-5x+4}\end{aligned}$$

3. Integrarar partiella bråk

$$\begin{aligned}\frac{21x-18}{x^2-5x+4} &= \frac{21x-18}{(x-4)(x-1)} = \frac{A}{x-4} + \frac{B}{x-1} \\ \Rightarrow \frac{21x-18}{(x-4)(x-1)} &= \frac{Ax-A+Bx-4B}{(x-4)(x-1)} \\ \Rightarrow A+B &= 21 \wedge -A-4B = -18 \\ \Rightarrow A &= 22 \wedge B = -1\end{aligned}$$

$$\int (x+5)dx + \int \frac{22}{x-4} dx - \int \frac{1}{x-1} dx$$

Svar:  $x^2/2 + 5x + 22 \ln|x-4| - \ln|x-1| + c, c \in \mathbb{R}$

### 4.7.3 Partiell integration

Om det är integraler med två funktioner så hjälper oftast partiell integration

#### Formel

$$\begin{aligned}\int_a^b f' g dx &= [fg]_a^b - \int_a^b fg' dx \\ \int f' g dx &= fg - \int fg' dx\end{aligned}$$

#### Bevis

$$\begin{aligned}(f(x)g(x))' &= f'(x)g(x) + f(x)g'(x) \Rightarrow \\ \int_a^b (fg)' dx &= \int_a^b f' g dx + \int_a^b fg' dx \\ \Rightarrow \int_a^b f' g dx &= \int_a^b (fg)' dx - \int_a^b fg' dx \\ \int_a^b f' g dx &= [fg]_a^b - \int_a^b fg' dx\end{aligned}$$

#### Exempel: polynom \* trigonomotrik

$$\begin{aligned}\int_0^{\pi/2} x \sin x dx \\ \int_0^{\pi/2} (-\cos x)' dx &= [-x \cos x]_0^{\pi/2} + \int_0^{\pi/2} \cos x (x)' dx \\ &= [-x \cos x]_0^{\pi/2} + [\sin x]_0^{\pi/2} = 1\end{aligned}$$

#### Exempel: polynom \* exponensiel

$$\begin{aligned}\int e^x x^2 dx \\ \int (e^x)' x^2 dx &= [e^x x^2] - \int e^x 2x dx = e^x x^2 - 2 \int (e^x)' x dx \\ &= e^x x^2 - 2e^x x + 2 \int (e^x)' dx = e^x x^2 - 2e^x x + 2e^x + c\end{aligned}$$

**Exempel: exponensiel \* trigonomotrik**

$$\int e^x \cos x dx$$

$$\begin{aligned} \text{Låt } I &= \int e^x \cos x dx = \int (e^x)' \cos x \\ &= e^x \cos x - \int e^x \sin x dx = e^x \cos x + e^x \sin x \\ &\quad - \int e^x \cos x dx \Rightarrow I = e^x \cos x + e^x \sin x - I \\ \Rightarrow I &= \frac{e^x \cos x + e^x \sin x}{2} + c \end{aligned}$$

**Exempel: bonus ln**

$$\begin{aligned} \int_1^2 \ln x dx \\ \int_1^2 x' \ln x dx = \dots = 2 \ln 2 - 1 \end{aligned}$$

**4.7.4 Generaliseringande integraler**

**Def:** Antag att  $f$  är kontinuerlig i  $a, b$  och att  $\lim_{x \rightarrow a^+} f(x) = \infty$ . Vi definierar den generaliseringade integralen  $\int_a^b f(x) dx = \lim_{\varepsilon \rightarrow 0^+} \int_{a+\varepsilon}^b f(x) dx$ , ”konstingst”. Om gränsvärdet existerar och är ändlig säger vi att integralen är konvergent, annart är den divergent.

**Beteckning**

- $\varepsilon$  Andvänds för små tal  $\lim_{\varepsilon \rightarrow 0^+}$
- $M$  Andvänds för stora tal  $\lim_{M \rightarrow +\infty}$
- $c$  Andvänds för konstanter  $+c$

**Sats:**

$$\int_a^{+\infty} \frac{dp}{x^p}, \quad a > 0$$

$p > 1 \Rightarrow$  Konvergerar

$p \leq 1 \Rightarrow$  Divergerar

**Sats: Jämförelsesatsen**

Anta att  $f$  och  $g$  är konternuerliga och  $0 \leq f(x) \leq g(x), (a \in [-\infty, +\infty], b \in (-\infty, \infty])$

(I) om integralen är konvergent  $\int_a^b g(x) dx$   
 $\Rightarrow \int_a^b f(x) dx$  är också konvergent

(II) om integralen är divergent  $\int_a^b f(x) dx$   
 $\Rightarrow \int_a^b g(x) dx$  är också divergent

**Sats:**

Om  $f(x)$  är positiv konternuerlig och avtagande i intervallet  $x \geq N$ , så är serien  $\sum_{n=N}^{\infty} f(n)$  konvergent

precis när  $\int_N^{\infty} f(x) dx$  är konvergent

**Exempel:**

Betämm om serien konvergerar eller divergerar

$$\sum_{n=10}^{\infty} \frac{1}{n \ln n (\ln(\ln n))^2}$$

$$f(x) = \frac{1}{x \ln x (\ln(\ln x))^2}$$

är positiv, konternuerlig och avtagande då nämnaren är stängt positivt ökande för  $x \geq 10$

$$\int_{10}^{+\infty} f(x) dx = \lim_{M \rightarrow +\infty} \int_{10}^M f(x) dx,$$

$$(u = \ln \ln x \Rightarrow du = 1/\ln x \cdot 1/x dx) \Rightarrow \lim_{M \rightarrow +\infty} \int_{\ln \ln 10}^{\ln \ln M} \frac{1}{u^2} du$$

$$\lim_{M \rightarrow +\infty} [-1/u]_{\ln \ln 10}^{\ln \ln M}$$

$$= \lim_{M \rightarrow +\infty} (-1/\ln \ln M + 1/\ln \ln 10)$$

$$= 1/\ln \ln 10$$

$$\sum_{n=10}^{\infty} \frac{1}{n \ln n (\ln(\ln n))^2} \text{ konvergerar}$$

**Sats:**

Antag att  $a_n > 0$ ,  $b_n > 0$  för varje  $n \in \mathbb{N}$

och antag att  $\frac{a_n}{b_n} \rightarrow L \neq 0 \wedge L < \pm\infty$

Då gäller att serien  $\int_{n=1}^{\infty} a_n$  och  $\int_{n=1}^{\infty} b_n$  är båda divergenta

### 4.7.5 Volymberäkningar

$$V = \int_a^b A(x)dx \text{ Rotationsvolymer runt x-axeln}$$

$$\Leftrightarrow \pi \int_a^b ((g(x))^2 - (f(x))^2) dx \text{ g är övre f är undre}$$

$$V = 2\pi \int_a^b xf(x)dx \text{ Rotationsvolymer runt y-axeln}$$

$$\Leftrightarrow 2\pi \int_a^b x(g(x) - f(x))dx \text{ g är övre f är undre}$$

**exempel** Beräkna volymen av den kropp som uppstår när området som begränsas av kurvan  $y = 4x - x^2 - 3$  och x-axeln roteras kring y-axeln.

$$y = 0 \Leftrightarrow 4x - 3 - x^2 = 0 \Leftrightarrow x = 1 \vee x = 3$$

$$\wedge y > 0, x \in [1, 3]$$

$$\Rightarrow V = 2\pi \int_1^3 x(4x - 3 - x^2)dx$$

$$= 2\pi \int_1^3 (4x^2 - 3x - x^3)dx$$

$$= 2\pi [4/3x^3 - 3/2x^2 - x^4/4]_1^3 = 16\pi/3$$

### Kurvlängd

$$L = \|\vec{x}_0 - \vec{x}_1\| = \sqrt{(a - c)^2 + (b - d)^2}$$

$$\Rightarrow L = \int_a^b \sqrt{1 + (f'(x))^2}$$

### exempel

Find the lenght of curve  $y = x^3/12 + 1/x$  from  $x = 1$

to  $x = 4$ .

$$\begin{aligned} f(x) &= x^3/12 + 1/x \\ L &= \int_1^4 \sqrt{1 + (f'(x))^2} dx \\ &= \int_1^4 \sqrt{1 + (x^2/4 - 1/x^2)^2} dx \\ &= \int_1^4 \sqrt{1 + x^4/16 - 1/2 + 1/x^4} dx \\ &= \int_1^4 \sqrt{x^4/16 + 1/x^4 + 1/2} dx \\ &= \int_1^4 \sqrt{(x^2/4 + 1/x^2)^2} dx = \int_1^4 (x^2/4 + 1/x^2) dx \\ &= [x^2/4 + 1/x^2]_1^4 = 6 \end{aligned}$$

### Rotationsarea

$$A = \int_a^b 2\pi|f(x)|\sqrt{1 + (f'(x))^2} dx$$

### exempel: Gabriel's Horn/Torricell's trumpet

Bestäm volymen och arean av den kropp som uppstår när området som begränsas av kurvan  $y = 1/x, x \geq 1$  och x-axeln roteras kring x-axeln.

$$V = \pi \lim_{M \rightarrow +\infty} \int_1^M \left(\frac{1}{x}\right)^2 dx = \pi \lim_{M \rightarrow +\infty} [-1/x]_1^M = \pi \cdot 1 = \pi$$

$$A = \lim_{M \rightarrow +\infty} \int_1^M 2\pi|1/x|\sqrt{1 + (-1/x^2)^2} dx$$

$$= 2\pi \lim_{M \rightarrow +\infty} \int_1^M \frac{\sqrt{1 + 1/x^4}}{x} dx$$

$$= 2\pi \lim_{M \rightarrow +\infty} \int_1^M \frac{x^2 \sqrt{1 + 1/x^4}}{x^3} dx$$

$$= 2\pi \lim_{M \rightarrow +\infty} \int_1^M \frac{\sqrt{x^4 + 1}}{x^3} dx$$

$$> 2\pi \lim_{M \rightarrow +\infty} \int_1^M \frac{\sqrt{x^4}}{x^3} dx$$

$$= 2\pi \lim_{M \rightarrow +\infty} \int_1^M \frac{x^2}{x^3} dx$$

$$= 2\pi \lim_{M \rightarrow +\infty} \int_1^M \frac{1}{x} dx \text{ Divergerar enligt p-satsen till } \infty$$

Eftersom integralen har en undre begränsning som divergerar även divergerar också integralen till  $+\infty$

## 4.8 Differential ekvationer

Tangent plan (Slope field): Andvärds för att se hur kurvan ser ut utefrån olika start värden.

grad: högsta graden på derivatan

$$ty'''(t) - 4y'(t) + 5t^2y(t) = e^t \text{ har grad 3}$$

Linjär diffirential ekvation: diff funktionerna har ingen upphöjning

$$y'' - 4t^2t' + e^t y = 0 \text{ är linjär}$$

$$t^2y''' + 5ty' - 4y^2 = 5 \text{ är inte linjär}$$

Homogen: om  $h(t) = 0 \Rightarrow$  ODE är homogen

Inhomogen: om  $h(t) \neq 0 \Rightarrow$  ODE är inhomogen

$$y''' - \sin^2(t)y' = 5y \text{ är homogen}$$

$$e^{t^2}y^{(5)} - y'' + 4ty(t) = e^t + t^3 \text{ är inhomogen}$$

**Sats:** superposition principle

$$\text{Låt } a_n y^{(n)} + a_{n-1} y^{(n-1)} + \dots + a_1 y' + a_0 y = 0$$

Om  $y_1(x), y_2(x)$  är lösningar till differentials ekvanationen

$\Rightarrow Ay_1(x) + By_2(x)$ ,  $A, B \in \mathbb{R}$  är en lösning till differentials ekvanationen

### 4.8.1 superabla ekvationer

**Def:** superabla ekvationer En differentialekvation är separabel om den kan skrivas på formeln  $\frac{dy}{dx} = f(x)g(y)$ .

**Exempel:** Vilka är separabel

$$(I) y' = x + y \text{ är inte separabel}$$

$$(II) \frac{dy}{dx} = 1 + e^y \text{ är separabel}$$

**Exempel:** beräkning

$$\text{Lös } y'(x) = (1 + e^{-x})(y^2 - 1)$$

$$y = \pm 1$$

$$\begin{aligned} \frac{dy}{y^2 - 1} &= (1 + e^{-x})dx \Rightarrow \int \frac{dy}{y^2 - 1} = \int (1 + e^{-x})dx (*) \\ \int \frac{1}{(y+1)(y-1)} dy &= \int \frac{((y+1)-(y-1))}{2(y+1)(y-1)} dy \\ &= \int \frac{1}{2(y-1)} dy - \int \frac{1}{2(y+1)} dy \\ (*) &\Rightarrow \int \frac{1}{2(y-1)} dy - \int \frac{1}{2(y+1)} dy = \int (1 + e^{-x}) dx \\ &\Rightarrow 1/2 \ln |y-1| - 1/2 \ln |y+1| = x - e^{-x} + c \\ &\Rightarrow \ln \left| \frac{y-1}{y+1} \right| = 2x - 2e^{-x} + 2c \\ &\Rightarrow e^{\ln \left| \frac{y-1}{y+1} \right|} = e^{2x - 2e^{-x} + 2c} \\ &\Rightarrow y = \frac{1 + e^{2x - 2e^{-x}} + 2c}{1 - e^{2x - 2e^{-x}} + 2c} \vee y = \frac{1 - e^{2x - 2e^{-x}} + 2c}{1 + e^{2x - 2e^{-x}} + 2c} \vee y = \pm 1 \end{aligned}$$

### 4.8.2 Linjära differentialekvationer av ordning 1

**Methodology**

$$y' + p(x)y = q(x)$$

Om  $g(x) = 0 \Rightarrow$  homogen och därmed seperable

Om  $g(x) \equiv 0$  Multiplisera med  $e^{M(x)}$  För att kuna andvända produktregeln så vi kan slå ihop  $y, y'$

$M(x) = \int p(x)dx$  är en primitiv funktion till  $p(x)$

$$\Rightarrow c = 0$$

$$\Rightarrow e^{M(x)}y' + e^{M(x)}p(x)y(x) = e^{M(x)}q(x)$$

Antifunktionen slår ut  $p(x)$

$$\Rightarrow e^{M(x)}y' + (e^{M(x)})'y(x) = e^{M(x)}q(x)$$

VL kan vi andvända produktregeln bakvänt

$$\Rightarrow (e^{M(x)}y(x))' = e^{M(x)}q(x)$$

Integratorar båda led

$$\Rightarrow \int (e^{M(x)}y(x))' dx = \int (e^{M(x)}q(x)) dx$$

Antiderivatan slår ut derivatan

$$\Rightarrow e^{M(x)}y(x) = \int (e^{M(x)}q(x)) dx \text{ Får y ensamt}$$

$$\Rightarrow y(x) = e^{-M(x)} \int (e^{M(x)}q(x)) dx$$

**Exempel**

$$\text{Lös } (1+t^2)y' + ty = \frac{1}{\sqrt{1+t^2}}$$

$$y' + \frac{t}{(1+t^2)y} = \frac{1}{(1+t^2)\sqrt{1+t^2}}$$

Linjär ode, ordning 1

$$p(t) = \frac{t}{(1+t^2)y}, q(t) = \frac{1}{(1+t^2)\sqrt{1+t^2}}$$

$$M(t) = \int \frac{t}{(1+t^2)y} dt, \text{ Låt } u = 1+t^2$$

$$\Rightarrow dt = du/2t \Rightarrow M(t) = \int \frac{1}{2u} du = \frac{1}{2} \ln|u| \\ = \frac{1}{2} \ln t^2 + 1$$

$$e^{M(t)} = e^{\frac{1}{2} \ln t^2 + 1} = \sqrt{t^2 + 1}$$

$$\Rightarrow \sqrt{t^2 + 1}y' + \frac{t\sqrt{t^2 + 1}}{t^2 + 1}y = \frac{1}{1+t^2}$$

$$\Rightarrow \sqrt{t^2 + 1}y' + \frac{t}{\sqrt{t^2 + 1}}y = \frac{1}{1+t^2}$$

$$\Rightarrow (\sqrt{t^2 + 1}y)' = \frac{1}{1+t^2} \Rightarrow \sqrt{t^2 + 1}y$$

$$= \int \frac{1}{1+t^2} \Rightarrow \sqrt{t^2 + 1}y = \arctan(t) + c$$

$$\Rightarrow y = \frac{\arctan(t)}{\sqrt{t^2 + 1}} + \frac{c}{\sqrt{t^2 + 1}}, c \in \mathbb{R}$$

### 4.8.3 Linjära differentialekvationer av ordning 2

#### Methodology

Det finns 3 metoder för att lösa Linjära differentialekvationer av ordning 2

$ay'' + by' + cy = 0$  vilket är den generella formeln.

Antag att lösningen är på formen

$$y = e^{rx}, y' = re^{rx}, y'' = r^2 e^{rx}$$

$$\Rightarrow ar^2 e^{rx} + bre^{rx} + ce^{rx} = 0$$

$(ar^2 + br + c)e^{rx} = 0 \Rightarrow ar^2 + br + c = 0$  andvänder pq-formeln och får följande

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

(i) Om  $b^2 - 4ac > 0 \Rightarrow r_1, r_2 \in \mathbb{R} \wedge r_1 \neq r_2$

$$\Rightarrow y_1 = e^{r_1 x}, y_2 = e^{r_2 x}$$

är lösningen till  $ay'' + by' + cy = 0$

$\Rightarrow y_k = c_1 e^{r_1 x} + c_2 e^{r_2 x}, c_1, c_2 \in \mathbb{R}$  superposition

(ii)  $b^2 - 4ac = 0 \Rightarrow r_1 = r_2 \in \mathbb{R}$  dubbelrot  $y_1 = e^{r_1 x}, y_2 = xe^{r_1 x}$

(iii) Om  $b^2 - 4ac < 0 \Rightarrow r_1 \neq r_2 \in \mathbb{C}$

$r_1 = k + li, r_2 = k - li$  Koplexatals konjugat enligt euler formel, alla lösningar är

$$y = c_1 e^{r_1 x} + c_2 e^{r_2 x}, c_1, c_2 \in \mathbb{C}$$

$$y = c_1 e^{r_1 x} + c_2 e^{r_2 x} = c_1 e^{(k+li)x} + c_2 e^{(k-li)x}$$

$$= c_1 e^{kx} e^{lix} + c_2 e^{kx} e^{-lix}$$

$$c_1 e^{kx} (\cos lx + i \sin lx) + c_2 e^{kx} (\cos -lx + i \sin -lx)$$

$$c_1 e^{kx} (\cos lx + i \sin lx) + c_2 e^{kx} (\cos lx - i \sin lx)$$

$$= e^{kx} ((c_1 + c_2) \cos lx + (c_1 - c_2)i \sin lx)$$

$$e^{kx} (\vec{c}_1 \cos lx + \vec{c}_2 \sin lx)$$

$$y = e^{kx} (c_1 \cos lx + c_2 \sin lx), c_1, c_2 \in \mathbb{R}$$

Vi för ett svar som är realt, men vi andvänder genväg med komplexa tal

#### Exempel Positiv kvot

$$\text{Lös } y''(x) - 4y'(x) + 3y(x) = 0$$

$$r^2(x) - 4r(x) + 3 = 0 \Rightarrow r_1 = 1, r_2 = 3$$

$$\Rightarrow y(x) = c_1 e^x + c_2 e^{3x}, c_1, c_2 \in \mathbb{R}$$

**Exempel noll kvot**

Lös  $y''(t) - 4y'(t) + 4$   
 $r^2 - 4r + 4 = 0 \Rightarrow (r - 2)^2 = 0 \Rightarrow r = 2$  dubbel rot  
 $y(t) = c_1 e^{2t} + c_2 t e^{2t}, c_1, c_2 \in \mathbb{R}$

**Exempel kplex kvot**

Lös  $y''(t) + 25y(t) = 0$   
 $r^2 + 25 = 0 \Rightarrow r^2 = -25 \Rightarrow r = \pm 5i$   
 $y(t) = c_1 \cos(5t) + c_2 \sin(5t), c_1, c_2 \in \mathbb{R}$

**Partikulär lösning****Methodology**

Partikulär lösning är för att gissa lösningen till det ikke homogena lösningarna  
 $ay'' + by' + cy = h(t), y(t) = y_h(t) + y_p(t)$

If  $f(x) = P_n(x)$ , try  $y_p = x^m A_n(x)$   
If  $f(x) = P_n(x)e^{rx}$ , try  $y_p = x^m A_n(x)e^{rx}$   
If  $f(x) = P_n(x)e^{rx} \cos(kx)$ ,  
try  $y_p = x^m e^{rx} (A_n(x) \cos(kx) + B_n(x) \sin(kx))$   
If  $f(x) = P_n(x)e^{rx} \sin(kx)$ ,  
try  $y_p = x^m e^{rx} (A_n(x) \cos(kx) + B_n(x) \sin(kx))$

För att ta reda på t eller k så kan tänka att den homogena lösningen är ej en lösning så börja med k=0 sen gå up tills det är sant

**Exempel trigonometrisk**

Lös  $y'' + 9y = \sin 3x$   
Sats: diff ekv kan skrivas på följande formel  $y = y_h + y_p$

Homogena lösningen

$$\begin{aligned} r^2 + 9 = 0 \Rightarrow r = \pm 3i &\Rightarrow y_h = A \sin 3x + B \cos 3x \\ \text{Particulöra lösningen} \\ \text{Sats } y_p = x^m(A_1 \sin 3x + A_2 \cos 3x), m = 1 \\ &\text{är godtaglig då det är första ekv som inte kan skrivas på homogen förmel} \\ y_p &= A_1 x \sin 3x + A_2 x \cos 3x \\ y'_p &= A_1 (\sin 3x + 3x \cos 3x) + A_2 (\cos 3x - 3x \sin 3x) \\ y''_p &= A_1 (3 \cos 3x + 3 \cos 3x - 9x \sin 3x) \\ &+ A_2 (-3 \sin 3x - 3 \sin 3x - 9x \cos 3x) \\ \Rightarrow y''_p &= A_1 (6 \cos 3x - 9x \sin 3x) \\ &+ A_2 (-6 \sin 3x - 9x \cos 3x) + 9(A_1 x \sin 3x + A_2 x \cos 3x) \\ &= \sin 3x \\ 6A_1 \cos 3x - 6A_2 \sin 3x &= \sin 3x \Rightarrow A_1 = 0, A_2 = -1/6 \\ \Rightarrow y_p &= -x/6 \cos 3x \\ \text{Den almäna lösningen blir på följande} \\ y(x) &= A \sin 3x + B \cos 3x - x/6 \cos 3x \end{aligned}$$

**Exempel polynom och Exponensiel**

Lös  $y'' + y' - 2y = 4e^2x + x^2$   
Sats: diff ekv kan skrivas på följande formel  $y = y_h + y_p$

Homogena lösningen

$$\begin{aligned} r^2 + r - 2 = 0 \Rightarrow r_1 &= 1, r_2 = -2 \\ y_h &= c_1 e^x + c_2 e^{-2x}, c_1, c_2 \in \mathbb{R} \\ \text{Particulöra lösningen} \\ \text{Sats } y_p = y_{p1} + y_{p2} \\ y'' + y' - 2y &= 4e^{2x} \\ y_{p1} &= x^m(Ae^{2x}), m = 0 \Rightarrow y'_{p1} = 2Ae^{2x} \Rightarrow y''_{p1} = 4Ae^{2x} \\ &\Rightarrow 4Ae^{2x} + 2Ae^{2x} - 2Ae^{2x} = 4e^{2x} \Rightarrow A = 1 \Rightarrow y_{p1} = e^{2x} \\ y_{p2} &= Ax^2 + Bx + C \Rightarrow y'_{p2} = 2Ax + B \Rightarrow y''_{p2} = 2A \\ &\Rightarrow (2A) + (2Ax + B) - (2(Ax^2 + Bx + C)) = x^2 \\ &\Rightarrow -2A = 1, 2A - 2B = 0, 2A + B - 2C = 0 \\ &\Rightarrow A = -1/2, B = -1/2, C = -3/4 \\ \Rightarrow y_{p2} &= -1/2x^2 - 1/2x - 3/4 \\ \\ \Rightarrow y(x) &= c_1 e^x + c_2 e^{-2x} + e^{2x} - 1/2x^2 - 1/2x - 3/4 \end{aligned}$$

**Resonans****Exempel Resonans**

$$\text{Lösy}'' + y = \sin 2t$$

$$r^2 + 1 = 0 \Rightarrow r = \pm i$$

$$y_h = c_1 \cos t + c_2 \sin t$$

$$y_p = A \sin t + B \cos t$$

$$y'_p = 2A \cos 2t - 2B \sin 2t$$

$$y''_p = -4A \sin 2t - 4B \cos 2t$$

$$y''_p + y_p = \sin 2t \Rightarrow -4A \sin 2t - 4B \cos 2t$$

$$+ A \sin t + B \cos t = \sin 2t$$

$$\Rightarrow A = -1/3, B = 0 \Rightarrow y_p = -1/3 \sin 2t$$

$$\Rightarrow y(t) = c_1 \cos t + c_2 \cos t - 1/2 \sin 2t$$

**Serielösningar**

Antag att serien  $\sum_{n=0}^{\infty} c_n(x - x_0)^n$  konvergerar för

alla  $n = 0$

$x \in x_0 - R, x_0 + R$  Då är funktionen

$f(x) = \sum_{n=0}^{\infty} c_n(x - x_0)^n$  deriverbar i

$(x_0 - R, x_0 + R) \wedge f'(x) \sum_{n=0}^{\infty} n c_n(x - x_0)^{n-1}$

Den sista serien konvergerar också i intervallet

$(x_0 - R, x_0 + R)$



## Chapter 5

# Computer Architecture

## 5.1 ISA1

### 5.1.1 MIPS instructions

The instructions that assembly language use for the MIPS proses.

#### Terminology

- Register file: 32 registers from R0-R31, each is 32 bits.
- Memory: Large each adders stores a word (4 x Registers). Address is always increments of 4. Memory is segmented into 4 8-bits of data to create a word.
- PC (Program counter): Increase with 4 each time to go to the next memory address.
- Instruction Registers: Retrieve the current instructions.
- Control: Inserts data to the register file and add the operation to ALU (Compute).
- ALU (Compute): Calculates the value.
- Memory Address: Call the value from a specific address. From ex “lw”.
- Data Register: Retries and directs value from memory to register file.

#### The operations most used

Function	Instruction	Effect
<b>add</b>	add R1, R2, R3	R1 = R2 + R3
<b>sub</b>	sub R1, R2, R3	R1 = R2 - R3
<b>add immediate</b>	addi R1, R2, 145	R1 = R2 + 145
<b>multiply</b>	mult R2, R3	hi, lo = R2 * R3
<b>divide</b>	div R2, R3	low = R2/R3, hi = remainder
<b>and</b>	and R1, R2, R3	R1 = R2 & R3
<b>or</b>	or R1, R2, R3	R1 = R2   R3
<b>and immediate</b>	andi R1, R2, 145	R1 = R2 & 145
<b>or immediate</b>	ori R1, R2, 145	R1 = R2   145
<b>shift left logical</b>	sll R1, R2, 7	R1 = R2 << 7
<b>shift right logical</b>	srl R1, R2, 7	R1 = R2 >> 7
<b>load word</b>	lw R1, 145(R2)	R1 = memory[R2 + 145]
<b>store word</b>	sw R1, 145(R2)	memory[R2 + 145] = R1
<b>load upper immediate</b>	lui R1, 145	R1 = 145 << 16
<b>branch on equal</b>	beq R1, R2, 145	if (R1 == R2) go to PC + 4 + 145*4
<b>branch on not equal</b>	bne R1, R2, 145	if (R1 != R2) go to PC + 4 + 145*4
<b>set on less than</b>	slt R1, R2, R3	if (R2 < R3) R1 = 1, else R1 = 0
<b>set less than immediate</b>	slti R1, R2, 145	if (R2 < 145) R1 = 1, else R1 = 0
<b>jump</b>	j 145	go to 145
<b>jump register</b>	jr R31	go to R31
<b>jump and link</b>	jal 145	R31 = PC + 4; go to 145

Figure 5.1: MIPS operation table. From ?.

note: memory operations like (sw R1, 0(R2)) stores the word in position R2+0 where 0 is the increment since if it is a loop it is needed. The value at that position is R1.

### 5.1.2 Sequencing

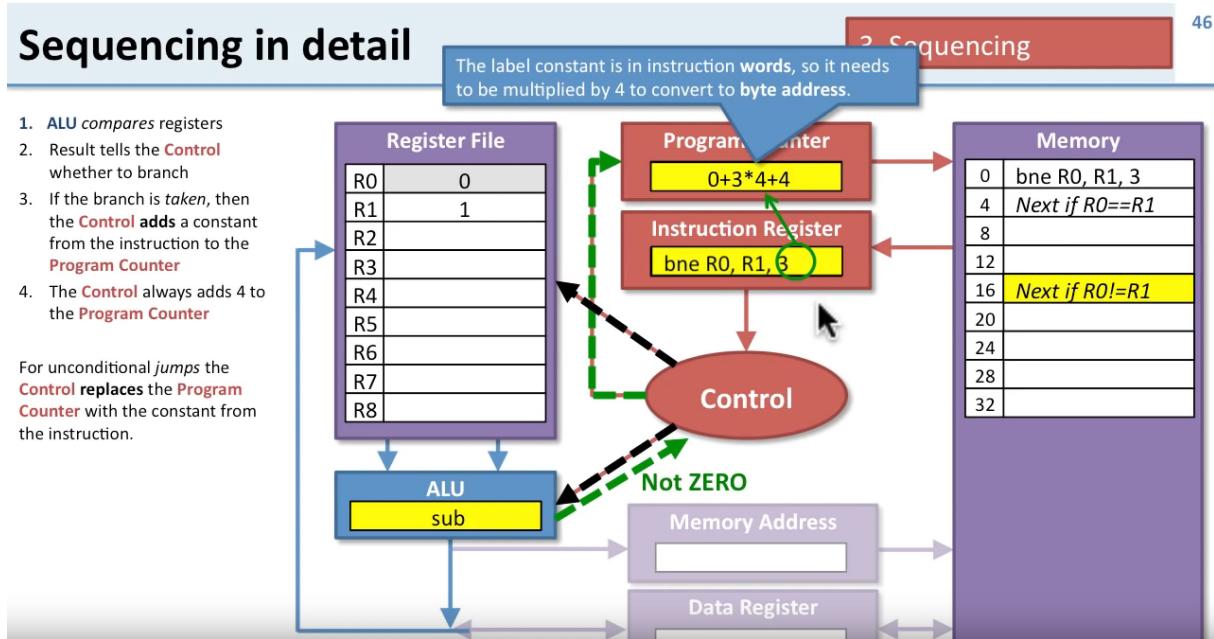


Figure 5.2: Sequencing. From ?

### 5.1.3 Converge to Assambly code

If Else example

$$R1 = i; R2 = j; R3 = h$$

```

if (i==j)
    h = i+j;
else
    h = i-j;
bne R1, R2, DoElse
    add R3, R1, R2
    j SkipElse
DoElse:
    sub R3, R1, R2
    SkipElse:

```

Figure 5.3: Assambly exampel. From ?

## 5.2 ISA2

### 5.2.1 MIPS type format

- R-format, Arithmetic and logical (add)
- I-format, Load/store, branch and immediates (addi, beq)
- J-format, Jump (j skipelse)

Name	Bit Fields						Notes
	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	(32 bits total)
R-Format	op	rs	rt	rd	shmt	funct	Arithmetic, logic
I-format	op	rs	rt	address/immediate (16)			Load/store, branch, immediate
J-format	op	target address (26)					

Figure 5.4: MIPS Types. From ?

A instruction allwas ends with 00 it means that when a instruction is done we update the program couner to by 4 or more it changes the posistion of the 16bit intermidjet for i-format and can. (bne, beq)

### 5.2.2 Procedure

Procedure is how a function call is handeld. It is devided into two parts Caller and Callee, the caller call the procedure (callee). The oporation for contolling procedure is called jump-and-link (jal). It stores the return address (PC+4) in \$ra (R31). Where (ra = register address (updates automatically)) (PC = project counter) (R31 = value). jr \$ra returns the value.

#### Stacking

To avoid conflicts with writing over the callers register files, one uses staking in predefined ranges to allow store data independent from the callers. (R29 = store stack data \$sp) When a register file is added the stock pointer is moved and when it is done it returns with (jr) and then reset the previously used register files. when it has used what it needs it removes them when leaded one value at a time

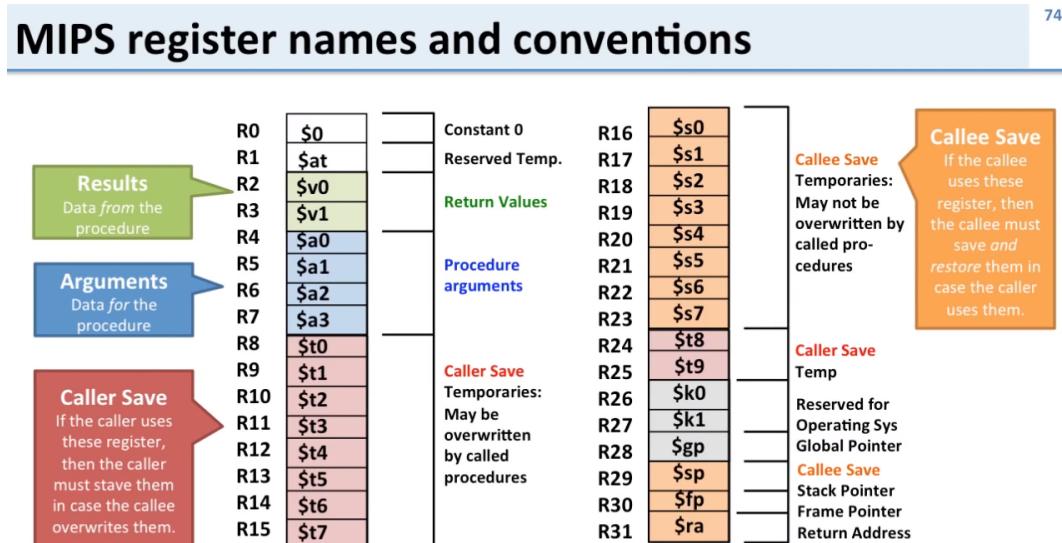


Figure 5.5: MIPS register. From ?

Notice that the register files for the callee save is \$sx and for the caller \$tx.

```

integer_array_sum:
    addi $sp, $sp, -4    # increase stack-pointer by one word
    sw $s0, 0($sp)      # save array index i to stack
    addi $sp, $sp, -4    # increase stack-pointer by one word
    sw $s1, 0($sp)      # save element n to stack

    addi $v0, $zero, 0    # Initialize Sum to zero.
    add $s0, $zero, $zero # Initialize i to zero.

ia_loop:
    beq $s0, $a1, ia_done    # Done if i == N
    lw $s1, 0($a0)          # n = A[i]
    add $v0, $v0, $s1        # Sum = Sum + n
    addi $a0, $a0, 4         # address = ARRAY + 4*i
    addi $s0, $s0, 1         # i++
    j ia_loop               # next element

ia_done:
    lw $s1, 0($sp)          # loads i from stack-pointer
    lw $s0, 4($sp)          # loads n from stack-pointer
    addi $sp, $sp, 8         # restore stack-pointer

    jr $ra                  # return to caller

```

Figure 5.6: code ex. From ?

### 5.2.3 Other ISA

- **Accumulator** (1 register)
  - 1 address      **add A**                   $acc \leftarrow acc + mem[A]$
- **General purpose register file** (load/store)
  - 3 addresses    **add Ra Rb Rc**         $Ra \leftarrow Rb + Rc$
  - **load Ra Rb**           $Ra \leftarrow Mem[Rb]$
- **General purpose register file** (Register-Memory)
  - 2 address      **add Ra addressB**     $Ra \leftarrow Mem[addressB]$
- **Stack** (not a register file but an operand stack)
  - 0 address      **add**                     $tos \leftarrow tos + next$   
 $tos = top\ of\ stack$

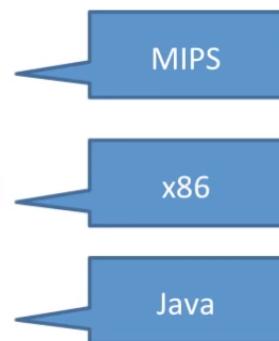


Figure 5.7: Other ISA. From ?

Stack	Accumulator	Register (Register-memory)	Register (Load-store)
Push A	Load A	Load R1, A	Load R1, A
Push B	Add B	Add R1, B	Load R2, B
Add	Store C	Store C, R1	Add R3, R2, R1
Pop C			Store C, R3

JVM	PDP-8, 8008, 8051	x86	MIPS, PPC, ARM, SPARC
-----	-------------------	-----	-----------------------

Figure 5.8: Other ISA instructions. From ?

## 5.3 Arithmetic

### 5.3.1 Binary numbers

- Binary numbers reduces noise and disregard the exact voltage to result in on off mode.
- In computer science octal and hexadecimals are also often used because of its properties. every octal digit represents 3 in decimal every hex digit represents 4 in decimal
- msb=most significant bit
- lsb=least significant bit
- Optimization:  $0010 * 0101$  2 operations  $\rightarrow 0101 * 0010$  1 operation
- Multiplication: (fixed point need to shift decimal point)  $0010 * 0101 = 0010 + (0010 \text{ shifting}[00]) = 1010$   
 $0011.010 * 0001.110 =$
- Addition: (Same for fixed points and integers)  $1110 + 1000 = \text{carry}[1]0110$  Overflow  $0101 + 0001 = 0110$   
Not overflow

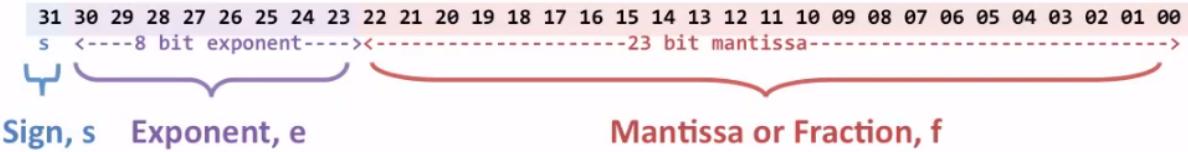
### 5.3.2 Negative integers

- Signed magnitude: msb is the sign  $1011_2 = -(2+1)_{10} = -3_{10}$
- Two's complement: if msb is 1 then negative number every other digit after is positive  $1011_2 = -8 + 2 + 1_{10} = -5_{10}$

### 5.3.3 Operations

#### 5.3.4 Non integer numbers (Floating and fixed point)

- Converting binary to decimal:  $0.111 = 1/2 + 1/4 + 1/8 = 0.875$
- Fixed point: for defined ranges 1.001 and 0.100 one can choose large and small representations
- Floating point: IEEE standard, see following image
- Standard floating point is in binary so FFFF is at most 1111 and with two's complement 0111



$$(-1)^s \times (1.f) \times 2^{(e - 127)}$$

Figure 5.9: Floating Point. From ?

### 5.3.5 Overflow

- input: msb = 1 for both numbers and output: msb = 0 (overflow)
- input: msb = 0 for both numbers and output: msb = 1 (overflow)
- input: msb = 0 for both numbers and output: msb = 0 (Not overflow)
- input: msb = 1 for both numbers and output: msb = 1 (Not overflow)
- input: msb = 1 and msb = 0 (May overflow)

## 5.4 Logic

# Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	$\bar{A}$	$AB$	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table border="1"> <tr> <th>A</th><th>X</th></tr> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </table>	A	X	0	1	1	0	<table border="1"> <tr> <th>B</th><th>A</th><th>X</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"> <tr> <th>B</th><th>A</th><th>X</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1"> <tr> <th>B</th><th>A</th><th>X</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"> <tr> <th>B</th><th>A</th><th>X</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table border="1"> <tr> <th>B</th><th>A</th><th>X</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table border="1"> <tr> <th>B</th><th>A</th><th>X</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

Figure 5.10: Floating Point. From ?

- truth table, is used to represent all possible inputs and then the corresponding output.
- To determine the possible schematics one can write for the input A and B “!A and B” not A and B. One often gets unnecessary complexity, one can often simplify the schematics. It is only done with the inputs that have an output of one.

- De Morgan's Law  $!(A + B) = !A \cdot !B$  and  $\cdot$  works the same way but are different.
- karna map is used to easily determine the schematics. It is a matrix. It is possible to use more inputs than two. In the matrix one looks at when output is one.
- A cabal with 4 wires can take 4 bits. Overflow needs one extra wire on output

### Building blocks

- Building blocks are the first layer of abstraction since one can not see the logical gates constructed of. For example add is a building block.
- Two types of logic, combinational and sequential. combinational: output just depends on input sequential: output depends on the state. State is stored in memory update on clock.
- MUXes (Multiplexers): choose input from a bus/ routing signal. 2-bit decision for input 4-bit selection. Starts from position 0.
- DEMUXes (Demultiplexers): opposite, take one signal and decides where it wants to be sent to Bus is a multi-bit signal "wire"
- Decoder: binary to hot, can choose position in array.
- Encoder: hot to binary.
- Adders: adds carries on extra bit to handle overflow

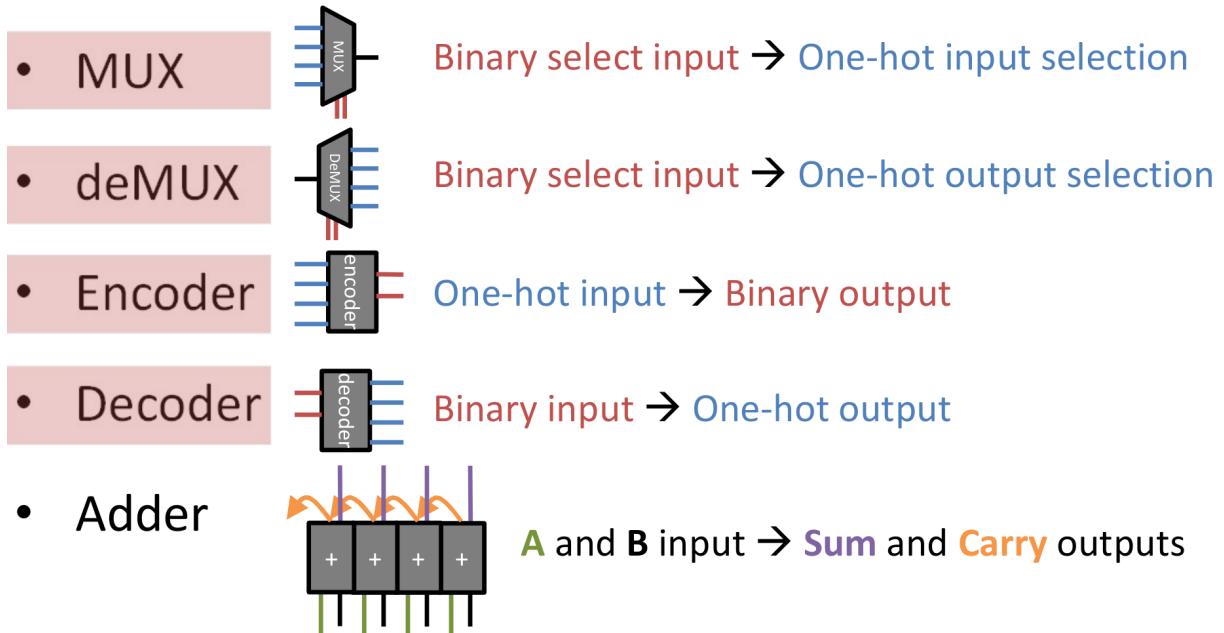


Figure 5.11: Floating Point. From ?

### Latches

- One use is for counters with a clock state triggers the count. So when input is one the output is 0 and when the clock clicks it is one then when the input is 0 the output is still 1.
- flip flop: is a edge triggered latch that has a clock trigger to open or close latch and then can save it to SRAM cell

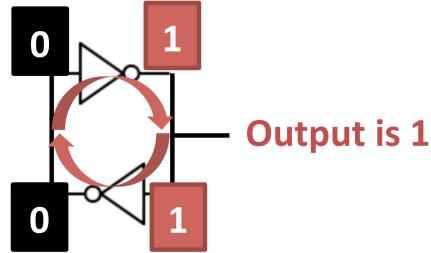


Figure 5.12: latche. From ?

## Memory

- SRAM (Static Random Access Memory) Big, fast and expensive. Loops thou value in order to store it. To write to memory one uses a switch to update value.
- DRAM (Dynamic Random Access Memory) Smaller, Slower and cheaper. Uses a transistor to store value and a capacitor to store the charge so the data dose not disappear. That is why it is slower become the capacitor needed to be recharge

## 5.5 processor control and datapath

There is 3 main part of the mips datapath as seen in the following image:

- **ALU operations (add, or, etc.)**
  - Load the instruction
  - Calculate the next PC
  - Read the register file
  - Do the ALU operation
  - Write data back to the register file
- **Memory access (load/store)**
  - Load the instruction
  - Calculate the next PC
  - Read the register file
  - Calculate the address
  - Read/Write the data memory
  - Write data back to the register file
- **Instruction fetch (branch)**
  - Load the instruction
  - Calculate the next PC
  - Read the register file
  - Calculate the branch address
  - Do a branch comparison
  - Update the PC

Figure 5.13: mips datapath 3 parts

An overview of the mips datapath with j-format instruction:

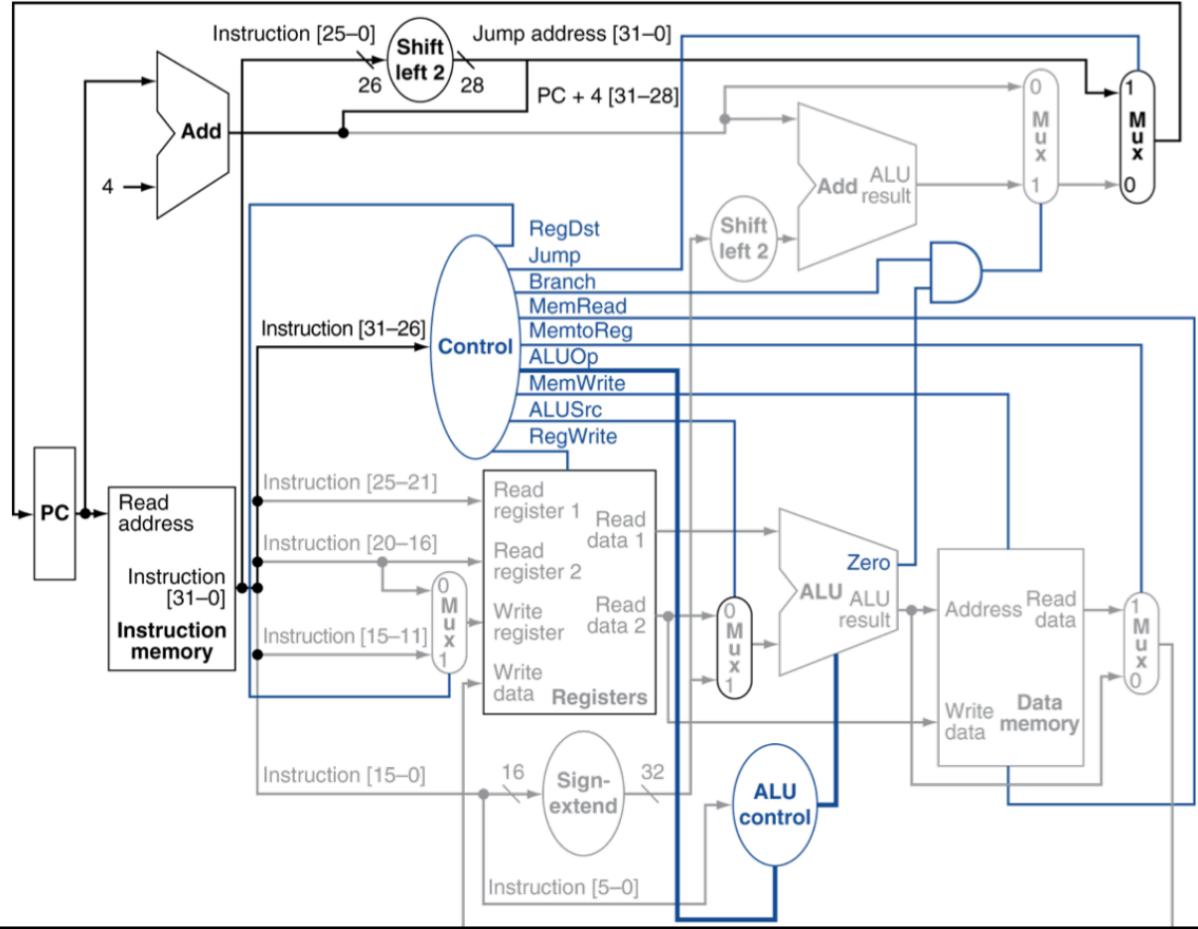


Figure 5.14: mips datapath. From ?

Think about the controller like a decoder that decodes the opt code from the different formats.

### 5.5.1 Clock

A clock is used to update the state and continue with the other instructions. Every state element uses a clock. In mips processor clock goes to memory, pc, rf and dm. The clock unit is in (MHz) converting from (ns) is simple. Ex 10ns:  $1/10\text{ns} = 0.1\text{MHz}$ .

### 5.5.2 Critical path

The longest path of the datapath is the critical path. Often PC is the fastest so one then calculates the amount of time it takes for the instruction has travel the data path and refernd to know what the critical path is. The longest part is data memory size it is big and slow. One often say read and write takes constant time regardless of the amount of different read and write.

## 5.6 pipeline

The purpose is to split a instruction cycle into multiple stages to run instructions in parallel to improve preformans. Itch stage has its own pipeline register file for storing the nessasary registers. Pipeline registers has a preferment reduction since it takes time, therfore more pipeline stages dose not equal greater preformans

over a certain amount. One issue of preformans is balance the stages inorder to get a good clock frequency. Not all stages are needed for eatch operation or instruction

### Mips stages

- IF= Instruction fetch
- ID= Decode and RF Read
- Ex= ALU Execute
- MEM= Memory
- WB= RF Write back

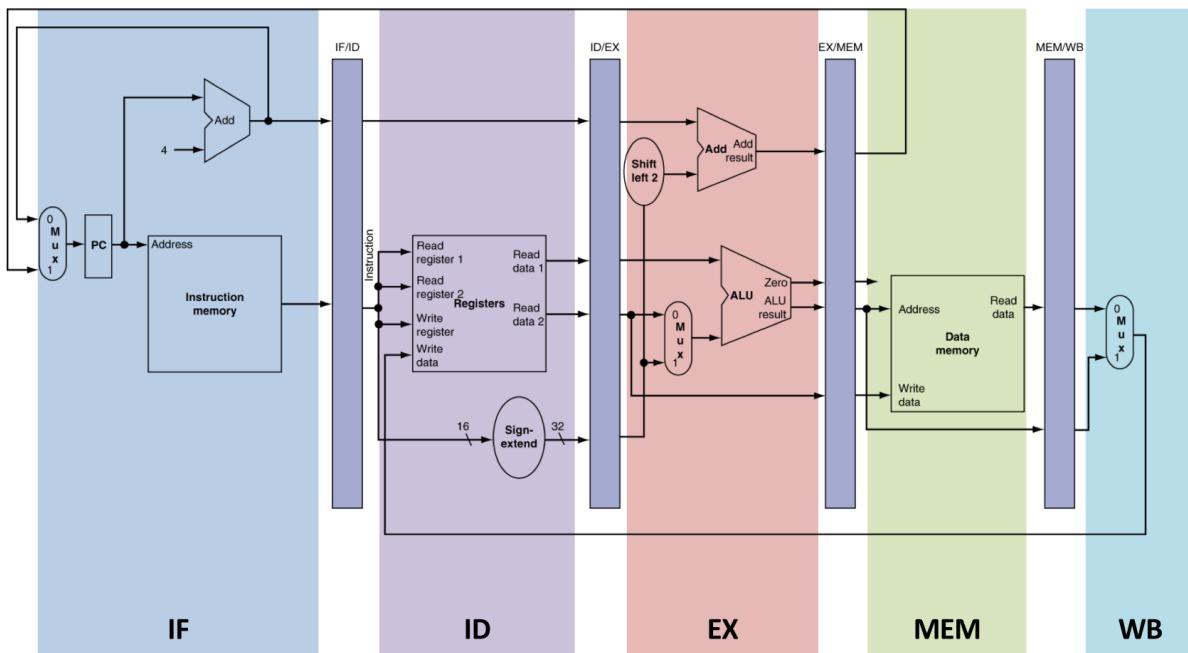


Figure 5.15: Pipeline. From ?

### Terminology

- Bubbles= Is detected by the hardware where there is no instuctions
- Nop= Is a sudo instruction for a stall type instruction (do not do anything)
- Delay slots= a stall type for branches. Try to fill in those slots with usful instructions.
- Interference= Can not read and write at the same time.
- Double pumping= Spiting write to first clock cycle and the second cycle is read.
- Forwarding= Getting data from a different pipeline stage from a register file.

### Calculating time coplexity

- Latency: (stages\*(penalty per stage))  
overhead: (instructions time (ex 100ns)) / (stages (ex 1000)) + (register overhead (1ns)) = 1.1ns Total time: 1.1\*1000

## 5.7 Pipeline hazards

### 5.7.1 Data hazards

#### The issue

- Data is not available where we need it (later in the pipeline; not written back yet).
- Data is not available when we need it (need to read memory first).

#### Fixing the issue

- Forward the data to where we need it.
- Stall if data is not ready yet (NOPs or Bubbles).

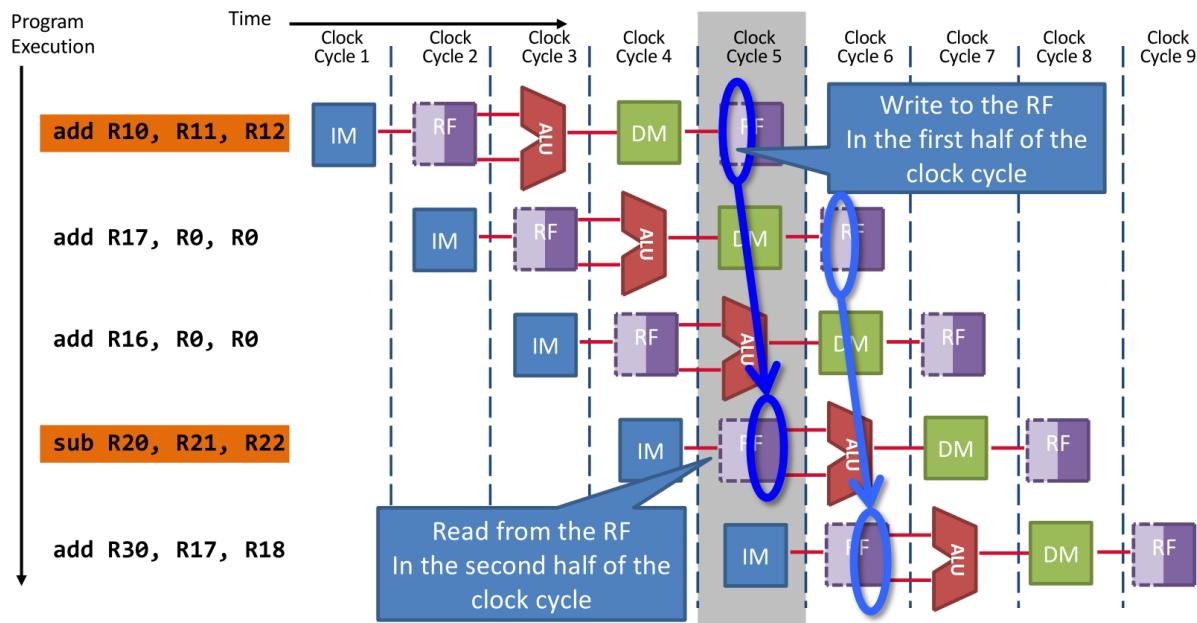


Figure 5.16: Bubble pump. From ?

### 5.7.2 Control hazards

#### The issue

- Don't know which instruction is next when we need to fetch it.

#### Fixing the issue

- Calculate branch as early as possible.
- Stall with a branch delay slot.

### 5.7.3 Structural hazards

#### The issue

- Can't do the instruction because the hardware is busy.

#### Fixing the issue

- Build more hardware (double-pumped register file). Double-pumped write at the first half cycle and read at the other half

### Calculating time complexity

- Branch delay  

$$\text{(How many branches (ex 20\%)) / (How many useful instructions can be filled in (ex 50\%))} = 20\%/0.5 = 10\%$$
- Instructions per cycle

## 5.8 Predicting Branches and Exceptions

When the prediction is wrong, clean up is needed.

- “Kill” or “Squash” so they don’t execute (they were wrong)
- Prevent them from writing: disable RegWrite, MemWrite in the pipeline
- Turn them into NOPs: change opcode to add R0, R0, R0 in the pipeline

### 5.8.1 Static predictors

- Predict always not taken
- Predict always taken
- Backwards-Taken, Forward-Not-Taken (BTFNT)

### 5.8.2 Dynamic predictors

The implementation of branch predictors look something like this:

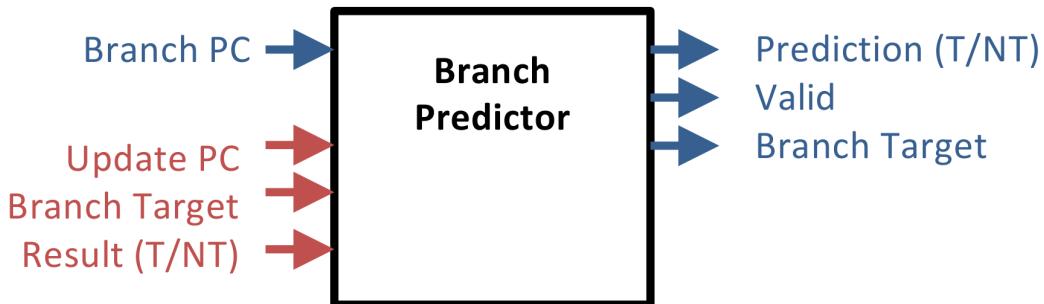


Figure 5.17: branch-predictor. From ?

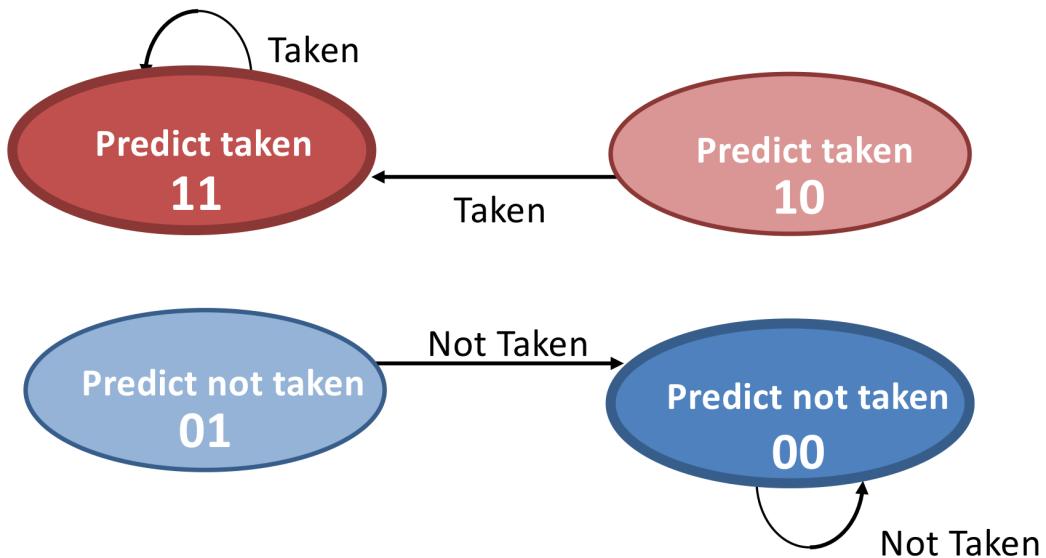


Figure 5.18: 2-bit predict. From ?

### BTB

- Branch Target Buffer (BTB)
- Save a table with PC in order to have history that we can predict on

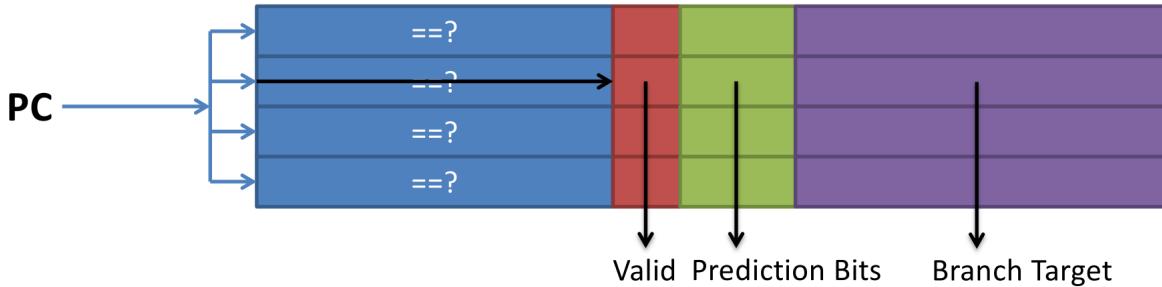


Figure 5.19: btb. From ?

### 5.8.3 Exceptions

Exceptions are non-normal events that interrupt the normal flow of instructions

- Divide by zero
- Misaligned memory access
- Page fault
- Memory protection violation

Interrupts are external events that interrupt the normal flow of

- **Synchronous vs. asynchronous**
  - Synchronous: occur at the same place every time a program executes
  - Asynchronous: caused by external devices and happen at different times
- **User requested vs. coerced**
  - Coerced are hardware events the user can't control
  - User requested are from the user
- **User maskable vs. nonmaskable**
  - Can the user disable the exception/interrupt?
- **Within vs. between instructions**
  - Does the event prevent the current instruction from completing, or interrupt after it?
- **Resume vs. terminate**
  - Can the event be handled (corrected) by the OS or program, or must the program be terminated?

Figure 5.20: exceptions. From ?

## 5.9 Input/Output

### Terminology

- nvram (flash): Similar to ram but it saves data when there is no power
- Busses: Parallel: many bits at once (e.g., 32 bits together in one clock) Used to be used everywhere Still used inside chips
- Serial links: Serial: one bit at a time (e.g., 32 clock cycles to send 32 bits) Used to be used only where distances were long (e.g., networks) Now used for most off-chip communications
- memory-mapped I/O: Manual handling of I/O devices Map portions of the address space to I/O devices Read and write to those addresses to access the
- direct memory access (DMA): Hardware who manage I/O devices (dynamically)
- Polling: The device puts its status somewhere The OS repeatedly checks for it to change
- Interrupt: When the device is ready, it gets the processor's attention by signaling an interrupt The OS then jumps to an interrupt handler to handle the event
- Throughput: x/s The read and write speed
- Latency: s/x Accessing time.
- Overhead: any combination of excess or indirect computation time, memory, bandwidth, or other resources that are required to perform a specific task

### Data transfers: manual copy

```

add R2, R0, R0      // Counter starts at 0
loop:
    lw R4, 0x12f0(R0) // Read the I/O device
    sw R4, 0xfe00(R2) // Store the results
    addi R2, R2, 4     // Next location
    bne R3, R2, loop
done:

```

### Data transfers: DMA

```

addi R2, R0, 0xfe00 // destination
addi R3, R0, 230400 // number of words to copy
sw R1, 0x100b(R0) // set the device address
sw R2, 0x1008(R0) // set the destination address
sw R3, 0x1004(R0) // set the length and start
wait:
    lw R2, R0(0x1000) // Wait for it to be done
    bne R2, R0, wait
done:

```

## 5.10 Cache

### 5.10.1 Memory hierarchy

• Registers	3 accesses/cycle	32-64
• Cache	1-10 cycles	8kB-256kB
• Cache	40 cycles	4-20MB
• DRAM	200 cycles	4-16GB
• Flash	1000+ cycles	64-512GB
• Hard Disk	1M+ cycles	2-4TB

Figure 5.21: memory-hierarchy. From ?

### 3-types of cache types

- Fully-associative: Have to search all blocks, but very flexible
- Direct-mapped: Only one place for each block, no flexibility
- Set-associative: Only have to search one set for each block, flexible because each set can store multiple blocks in its ways

### Write policies

- Write-through: slow, simple
- Write-back: fast (keeps the data just in the cache), more complex

### Terminology

- Data: What is being stored
- Tag: What address the data has
- Index: What we use to determine where we want to put the data
- Cache line (block) size: How many tag's there are

- Valid bit: If the data is correct
- Dirty bit: The data has been change and we need to write it to memory
- Type of cache: Fully-associative (FA), Set-associative (SA), Direct-mapped (DM)
- Replacement policy: Write-through, Write-back

### Cache blocks

- tag has 30bits and the 2 other bits is to determin what data 63 bits for each entry 32bit data 30bit tag 1bit valid-bit
- larger block of data -> fewer tags -> more efficient storage
  - 1 word cache block we have 1 32bit data
  - 2 word cache block we have 2 32bit data
  - 4 word cache block we have 4 32bit data
  - we load more data when we have space for it we will then have spatial locality
  - Last 2: determine the byte within the word
  - Next N: determine which word in the cache block
  - Remaining 32-N-2: tag

### Calculate overhead

- Data= Cache-Lines \* Byte-Lines
- Overhead= Cache-Lines \* (Tags-per-line + valid-bit)
- % data= Overhead / Data

### Calculate Address (Tag Index Offset)

- Direct mapped:  
 $\text{Offset} = \log(\text{number of data blocks}) + 2\text{-bit}$  (4-bit cache block size if 64 then log 64)  
 $\text{Index} = \log(\text{number of cache lines})$   
 $\text{Tags} = 32(\text{bit processor}) - (\text{Index} + \text{Offset})$
- x-set associative:  
 $\text{Offset} = \log(\text{number of data blocks}) + 2\text{-bit}$  determines how the address looks like  
 $\text{Index} = \log((\text{number of cache lines})/x)$   
 $\text{Tags} = 32(\text{bit processor}) - (\text{Index} + \text{Offset})$
- fully associative:  
 $\text{Offset} = \log(\text{number of data blocks}) + 2\text{-bit}$  determines how the address looks like  
 $\text{Index} = 0$   
 $\text{Tags} = 32(\text{bit processor}) - (\text{Offset})$

- Determine which bits in a 32-bit address are used for selecting the **byte (B)**, selecting the **word (W)**, indexing the **cache (I)**, and the cache **tag (T)**, for each of the following caches:
- **64-line, direct-mapped, write-through, 8 byte line**
  - 8 byte line = 2 words per line, need 1 word bit
  - 64 lines, need 6 bits for indexing
  - **TTTT TTTT TTTT TTTT TTTI IIII IWBB**
- **256-line, fully-associative, write-back, 16 byte line**
  - 16 byte line = 4 words per line, need 2 word bits
  - 256 lines, but fully associative! 0 bits for index! (data can go anywhere)
  - **TTTT TTTT TTTT TTTT TTTT TTTT WWBB**
- **4096-line, 4-way set-associative, write-back, 64 byte line**
  - 64 byte line = 16 words per line, need 4 word bits
  - 4096 lines/4-ways = 1024 sets, need 10 bits for indexing
  - **TTTT TTTT TTTT TTTT IIII IIII IIWW WWBB**

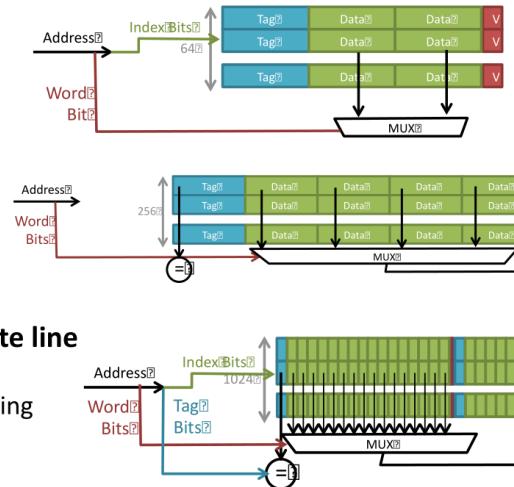


Figure 5.22: cache-format. From ?

### Calculate Avrage memory access time

- Avrage-cycles= CacheL1\*Cycles + CacheL2\*Cycles + Dram\*Cycles

#### 5.10.2 Cache misses 3C's

- cold/compulsory miss: When the program first begin it doesn't have anything in the cache
- conflicts miss: to small to store the necessary data
- capacity miss: map's at the same block an then overwrite unnecessarily

## 5.11 Virtual Memory

### Why use VM

- Map memory to disk ("unlimited" memory)
- Keep programs from accessing each other's memory (security)
- Fill holes in the RAM address space (efficiency)

### Terminology

- Translation= map address
- Page tables= for each program keep track of all translations
- Fine grain= page table with spesific address
- Coarse grain= page table with address mapped ranges
- Page Table Entries (PTEs)= number of translation
- Translation Lookaside Buffer TLB= All of the pages, fast translation via hardware
- VA= Virtual program addresses
- PA= Physical RAM addresses

- Page offset = point to a range and then use the offset to determine where
- Translation Lookaside Buffer (TLB) = page table cache (Faster)
- Multilevel page table translation = page table points to other page tables (inception)

### Combining TLB and cache

- Physical caches: slow and needs the translation first and then save get the PA also known as Physical-Index, Physical-Tagged (PIPT)
- Virtual caches: fast uses only virtual addresses, no translation, difficult for protection also known as Virtual-Index, Virtual-Tagged (VIVT):
- Virtual-Index, Physically-Tagged (**VIPT**): VA for index PA for tags, fast does translation and fetch from cache at the same time, most commonly used. We need a comparator to see if the PA tags from cache matches the TLB PA only then we can say if we had a hit or a miss. The VA offset is what the PA tag is selected and the VA tag (number of virtual pages) is what the TLB uses. Mostly used for L1 cache not L2.

### TLB

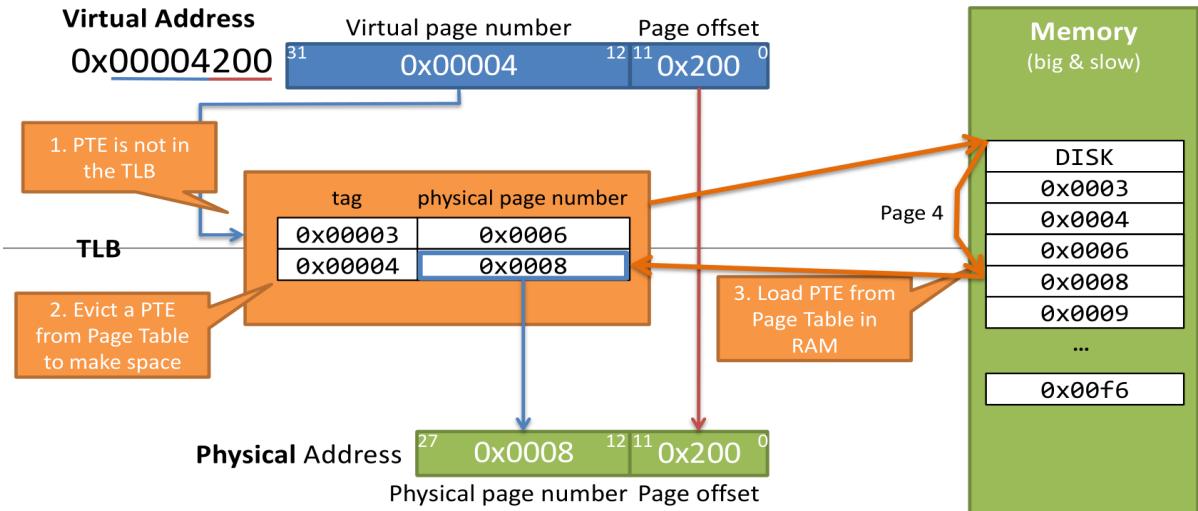


Figure 5.23: tlb

useful conversion:  $2^n = xB$

0		10	1kB	20	1MB	30	1GB
1	2B	11	2kB	21	2MB	31	2GB
2	4B	12	4kB	22	4MB	32	4GB
3	8B	13	8kB	23	8MB		
4	16B	14	16kB	24	16MB		
5	32B	15	32kB	25	32MB		
6	64B	16	64kB	26	64MB		
7	128B	17	128kB	27	128MB		
8	256B	18	256kB	28	256MB		
9	512B	19	512kB	29	512MB		

Figure 5.24: conversion. From ?

### Calculating Page sizing

- Number-of-Virtual-Pages=  $2^{32}$ /pages
- Bits-used-for-Page-Offset= log(pages)
- Bits-used-for-VPN= 32(bit processor)- Bits-used-for-Page-Offset

### Calculating TLB size

- TLB-size=Entries\*Pages

## 5.12 Parallism

### 5.12.1 Multicore

powerwall

$$P = CV^2f$$

C = capacitor, smaller transistors smaller capacitors

V = voltage, decreasing makes it slower

f = frequency, reducing clock speed

### 5.12.2 Paralel programming

#### Average processors

Calculate how many cores are used in average we need to know  
 how chunks (work) there is in total  
 how many time units (cycles think of a reverse pyramid)

there is 16 input data and we have 8-cores, we want to calculate the total sum  
 what is the average processor being used  
 $15 \text{ chunks}, 4 \text{ time units} \Rightarrow 15/4 = 3.75$

#### parallel issues

- Most programs can not utilize parallelism, need to devide the program to different executions.
- We also need to share cache and therefore have performance issue since we can't use the entire cache.

#### How much faster

75% parallel, 25% nonparallel we have hundred thousand cores  
 $\Rightarrow$  Parallelism takes  $(0.75/100000 + 0.25 * 1)$   
 Singular takes  $(0.75 * 1 + 0.25 * 1) \Rightarrow 4 * \text{faster}$

$$\text{Speedup with Amdahl's law } \text{Speedup} = \frac{1}{(1 - P) + P/S}$$

$P$  = Parallel fraction

$S$  = Speed up of the parallel part

$$\Rightarrow \frac{1}{(1 - 0.75) + 0.75/100000} = \frac{1}{0.25 + 0} = 4$$

### 5.12.3 Synchronization

- Fix the issue with 2 processors accessing the same value at the same time.
- We can use atomic swap to first set lock to 1 then check the data.

#### locks

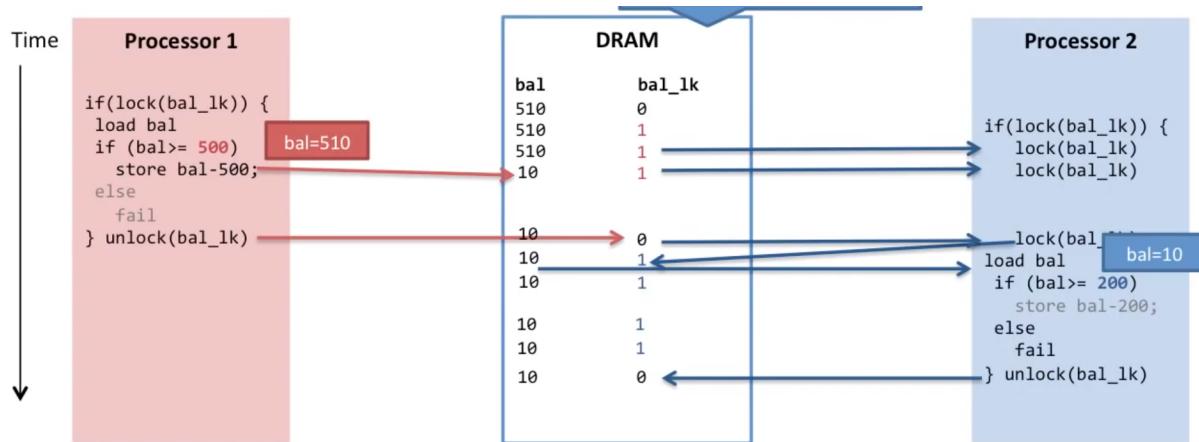


Figure 5.25: locks

#### 5.12.4 Cache coherency

- How we use caches to save memory
- Locks: if the data has been accessed. (not a guarantee of protection)
- Snooping: Look what the other processors are storing in their private cache

snooping

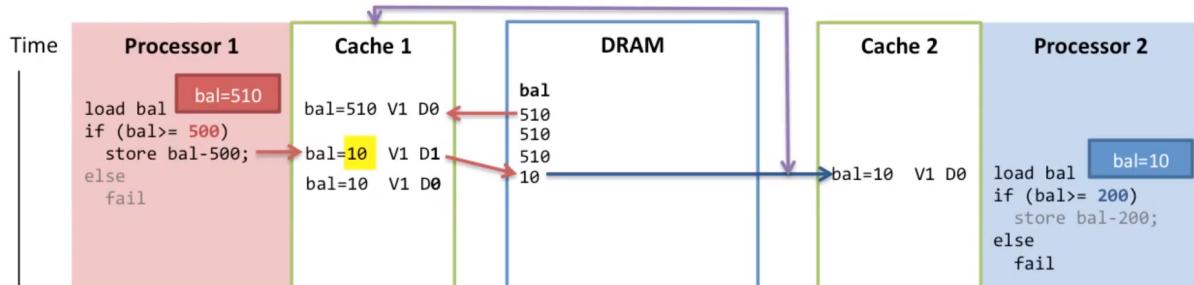


Figure 5.26: locks. From ?

#### 5.12.5 ILP

- Instruction level parallelism (ILP)
- better to have out-of-order execution since we can find independent instructions
- We use Dual-issue pipeline so we can have one for memory instructions and one for non memory instruction
- It makes ISA promise with in order execution
- Issue with data hazards, more complexity

dual issue pipeline

- Regular Path
- Ld/St Path
- Added:
  - More RF ports
  - 2<sup>nd</sup> instruction fetch
  - 2<sup>nd</sup> sign-extension
  - 2<sup>nd</sup> ALU
  - More forwarding logic and paths
- Now we can issue both a ld/st and any other instruction at the same time!

```

1: add r1, r2, r3
1: ld r4, r5
2: sub r7, r1, r4
2: st r8, r9
3: or r5, r8, r9
3: nop

```

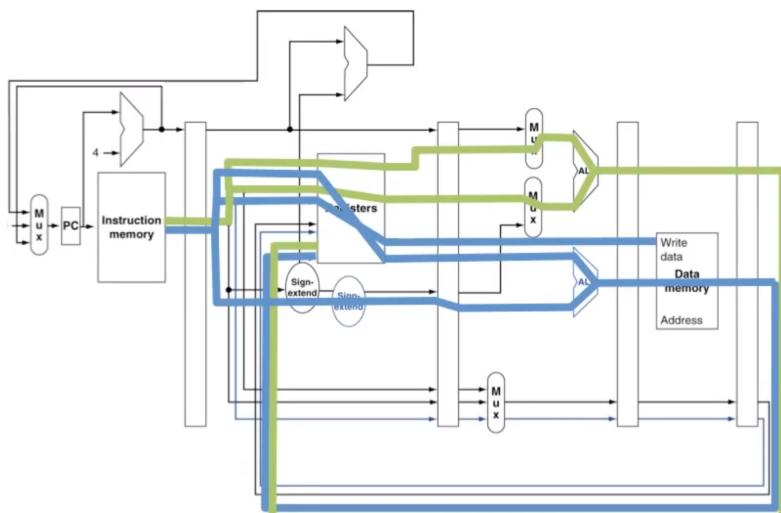


Figure 5.27: dual-issue-pipeline. From ?



## Chapter 6

# Imperativ and Object-Oriented Programming Methodology

## 6.1 Introduction

### 6.1.1 Test

Innan: Läs igenom standard instruktionerna. Sätt upp mjukvara miljö på skollans datorer.

Under: Det är mycket tid. Läs igenom instruktionerna på uppgiften noggrant.

## 6.2 Imperative programming i C

C is a strict programming language, we must therefore write every type of variables. Like any other imperative programming language it execute sequentially and every argument is evaluated in order. It also allow side effects like changing the state freely.

### Vilkorsatser

```
if (expr) { expr; }
if (age > 100) { puts("Very old"); }

if (expr) { expr; }
else { expr; }
if (age % 2 == 0) { puts("Even"); }
else { puts("Odd"); }

expr ? expr : expr
a < b ? b : a;
```

### Shortcuts loop operations

```
n_fakultet *= n--;
```

### 6.2.1 Data structurer

#### stack och heap

I kortids minnet som c kompilatorn hanterar skjälv så sappar datan i en stack. Stack funkar som talrikar som man staplar på varandra och tar den som du läser ut.

Heap är för långtids mine och störe data som arrayer. Det fungerar som ett rutat papper där varje ruta är en viss data size.

**Shortcuts loop operations** För att använda heapen, (rita på det rutade papret) måste man följa 4 steg. steg 1: räkna ut hur många rutor vi behöver Använd plattformsberoende hjälpmedel och vanlig aritmetik

```
sizeof(T) * antal
int size = sizeof(int) * 1024
```

#### steg 2: reservera motsvarande yta

Här kan det gå fel — det kanske inte finns plats på papperet

```
T *namn = malloc(antal bytes);
int *skonummer = malloc(size);
```

#### steg 3: använd ytan hejvilt

Men sudda först! (Beror på datastrukturen)

#### steg 4: lämna tillbaka platsen när du är klar

Annars kommer det gå dåligt i ett framtida steg 2

```
free(namn);
free(skonummer);
```

### Structar

För att skapa egna data structurer som använder man sig av "struct" operatorn. Om man vill gömma fler data typer i en så kan man använda en "union". För att göra en typ så använder man "typedef" och avslutar med

```
namnet_t{}
```

### Pekare

Pekare pekar var datan finns så man kan skicka information utan att sicka stora mängder data.

```
int a
int *b
```

Syntaxen som används är: pekartypen

```
int *
```

Skillnaden mellan arrayer och pointers är väldigt liten i c. Därför är samma operation för string (

```
char *
```

som en pointer (

```
int *
```

### Linked list

Varge element har data av någon typ samt en pekare till nästa element. Head and tail kan beskriva en sådan lista, då head är första elementet och tail är rästen.

Tid komplexiteten är: On. Linjär

### Hash table

Hash table är ett mer effektivt sätt att hantera data istället för en array. Då man inte behöver källa varge element vad den finns utan andvänder en hash function för att skapa ett ”nummer” som pekar till vad datan finns någonstans exempelvis kan det vara mod element i hash table då vet vi har vi ska ska söka efter. Ett problem som kommer uppstå är att data mappas till samma plats, då får man andvända exempelvis linuer probing som säger att om det är fult på platsen gå till nästa tills det finns och om det inte finns så protesterar programet. Varge element inehåller inget

värde eller deleted varde om det är tomma.

Tid komplexiteten är: O1. Konstant men inte i verkligheten

## 6.3 Object-Oriented programming i Java

kör den mest spesifika implementationen av metoden single dispage, vi bryr os ombjectet vi kör metoden på men inte argumentet



## Chapter 7

# Linear Algebra and Geometry I

## 7.1 Linjära ekvationssystem

**3 gundläggande operationer för att lösa linjära ekvationer**

1. (Tvärpilar) Byt om på två ekvationer
2. (Lambda) Multiplicera båda sidorna av en ekvation med  $\lambda \neq 0$
3. (Lambda pil) addera  $\lambda$  gånga en ekvation till en annan ekvation.

**Generell lösning**

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Lambda pil upp:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Ger nya ekvationssystemet

Lambda pil upp:

$$\begin{cases} (a_1 + \lambda a_2)x + (b_1 + \lambda b_2)y = c_1 + \lambda c_2 \\ a_2x + b_2y = c_2 \end{cases}$$

-Lambda pil upp:

$$\begin{cases} (a_1 + \lambda a_2)x + (b_1 + \lambda b_2)y = c_1 + \lambda c_2 \\ a_2x + b_2y = c_2 \end{cases}$$

**Exempel linjär ekvationssystem**

$$\begin{cases} 2x + 3y = 4 \\ 3x - y = 6 \end{cases}$$

Andvänder rad operationer för att lösa ekv systemet

$$\left( \begin{array}{c} -\frac{3}{2} \\ 1 \end{array} \right) \text{ pil ned} \begin{cases} 2x + 3y = 4 \\ 3x - y = 6 \end{cases}$$

$$(3x - y) - \frac{3}{2}(2x + 3y) = 6 - \frac{3}{2} \cdot 4$$

$$\Leftrightarrow \left( 3 - \frac{3}{2} \cdot 2 \right) x + \left( -1 - \frac{9}{2} \right) y = 6 - 6$$

$$\Leftrightarrow -\frac{11}{2}y = 0$$

$$\left( \begin{array}{c} -\frac{2}{11} \\ 1 \end{array} \right) \text{ ned} \begin{cases} 2x + 3y = 4 \\ -\frac{11}{2}y = 0 \end{cases}$$

$$(-3)\text{pil upp} \begin{cases} 2x + 3y = 4 \\ y = 0 \end{cases}$$

$$\left( \begin{array}{c} \frac{1}{2} \\ 1 \end{array} \right) \text{ upp} \begin{cases} 2x = 4 \\ y = 0 \end{cases}$$

$$\begin{cases} x = 2 \\ y = 0 \end{cases}$$

**Kontrol:** stoppar in  $x$  och  $y$  i ekvationerna

$$2 \cdot 2 + 3 \cdot 0 = 4 \text{ (stämmer)}$$

$$3 \cdot 2 - 0 = 6 \text{ (stämmer)}$$

### 7.1.1 Total Matris

**Termonologi**

- Rader och Kolonner: Rader är vågräta delen av matrisen (ekvationen) Kolonner är lodräta delen (koeficenterna)
- ledande ekvivalent:
- trappstegs matris: När ledande ekvivalent är i trapp form går ned max ett steg
- rad kanonisk: När alla av de ledande ekvivalent inte har någon i samma kolonn

**Exempel matriser**

$$\left\{ \begin{array}{l} x + 2y + z = -1 \\ 2x + (a+3)y + 3z = -4 \\ x + (3-a)y + (a-2)z = a-1 \end{array} \right.$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 2 & a+3 & 3 & -4 \\ 1 & 3-a & a-2 & a-1 \end{array} \right)$$

(-1 rad1 till rad2), (-2 rad1 till rad3)

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 0 & a-1 & 1 & -2 \\ 0 & 1-a & a-3 & a \end{array} \right)$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 0 & a-1 & 1 & -2 \\ 0 & 0 & a-2 & a-2 \end{array} \right)$$

$a \neq 1 \wedge a \neq 2$

$(\frac{1}{a-2} \text{rad } 3)$

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 0 & a-1 & 1 & -2 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

(-1rad 3 till rad 2), (-1rad 3 till rad 1)

$$\left( \begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & a-1 & 0 & -3 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

$(\frac{1}{a-1} \text{rad } 2)$

$$\left( \begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & a-1 & 0 & -3 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & 1 & 0 & \frac{3}{1-a} \\ 0 & 0 & 1 & 1 \end{array} \right)$$

(-2rad 2 till rad 3)

$$\left( \begin{array}{ccc|c} 1 & 0 & 0 & -2 - \frac{6}{1-a} \\ 0 & 1 & 0 & \frac{3}{1-a} \\ 0 & 0 & 1 & 1 \end{array} \right)$$

$$\left\{ \begin{array}{l} x = -2 - \frac{6}{1-a} \\ y = \frac{3}{1-a} \\ z = 1 \end{array} \right.$$

**Kontroll:** stoppar in x,y,z i ekvationerna

$$\left\{ \begin{array}{l} \left( \frac{2a-8}{1-a} \right) + 2 \cdot \left( \frac{3}{1-a} \right) + 1 = -1 \\ 2 \cdot \left( \frac{2a-8}{1-a} \right) + (a+3) \cdot \left( \frac{3}{1-a} \right) + 3 \cdot 1 = -4 \\ \left( \frac{2a-8}{1-a} \right) + (3-a) \cdot \left( \frac{3}{1-a} \right) + (a-2) \cdot 1 = a-1 \end{array} \right.$$

## 7.2 Vektorer/koordinater planet och rummet

i

**Räkne regler vektorer**

$$A1 \vec{u} + \vec{v} = \vec{v} + \vec{u}$$

$$A2 \vec{u} + (\vec{v} + \vec{w}) = (\vec{v} + \vec{u}) + \vec{w}$$

$$A3 \vec{u} + \vec{0} = \vec{u}$$

$$A4 \vec{u} + \vec{v} = \vec{0} \Leftrightarrow \vec{v} = -\vec{u}$$

$$M1 1\vec{u} = \vec{u}$$

$$M2 k(l\vec{u}) = (kl)\vec{u}$$

$$M3 (k+l)\vec{u} = k\vec{u} + l\vec{u}$$

$$M4 k(\vec{u} + \vec{v}) = k\vec{u} + k\vec{v}$$

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \text{ för } \mathbb{R}^2$$

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \text{ för } \mathbb{R}^3$$

**Definition: Parallela vektorer**

Parallela omm  $\exists k : \vec{u} = k\vec{v} \vee k\vec{u} = \vec{v}$

### 7.2.1 Bas

Standard bas är välbikant då i planet är x och y axeln medans i ett rum är x, y och z. Baser som inte är standard är vektorer som ej är parallella som då skappar axlar som ej behöver vara vinkelräta.

**Definition: Bas**

Bas i plan  $\forall \vec{x}, \exists k_1, k_2 : \vec{x} = k_1\vec{u} \vee k_2\vec{v}$

Bas generell  $\vec{x} = k_1\vec{u}_1 + k_2\vec{u}_2 + \dots + k_n\vec{u}_n$

$$\underline{u} = (\vec{u}_1 \vec{u}_2 \dots \vec{u}_n)$$

$$\vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \vec{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \vec{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

**Exempel: kordenatar för vektor i bas**

$$\text{Låt: } \vec{u} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \vec{v} = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$$

$$\text{Hitta kordenarerna för vektorn } \vec{x} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

Lösning: vi måste hittar reala tal  $k_1$  och  $k_2$  så att

$$\begin{pmatrix} 0 \\ 6 \end{pmatrix} = k_1 \begin{pmatrix} 2 \\ 1 \end{pmatrix} + k_2 \begin{pmatrix} -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2k_1 - 2k_2 \\ k_1 + k_2 \end{pmatrix}$$

$$\begin{cases} 2k_1 - 2k_2 = 0 \\ k_1 + k_2 = 6 \end{cases} \Rightarrow \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

**Exempel: Om det är en bas**

För vilken värde på  $a$  är vektorerna en bas för  $\mathbb{R}^3$

$$\vec{u}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \vec{u}_2 = \begin{pmatrix} 1 \\ 2 \\ a \end{pmatrix}, \vec{u}_3 = \begin{pmatrix} 1 \\ a \\ 3 \end{pmatrix}$$

Vi måste hitta ett  $a$  sådant att

$$\begin{aligned} k_1 \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, k_2 \begin{pmatrix} 1 \\ 2 \\ a \end{pmatrix}, k_3 \begin{pmatrix} 1 \\ a \\ 3 \end{pmatrix} &= \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \\ \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & a \\ 3 & a & 3 \end{pmatrix} &\xrightarrow{\left[ \begin{array}{c|cc} \square & -2 & \\ \square & + & \\ \hline & + & \end{array} \right]^{-3}} \sim \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & (a-2) \\ 0 & (a-3) & 0 \end{pmatrix} \end{aligned}$$

Om  $a = 2$ : då är sista ekvationen  $0 = ?$  och vi ser att lösningarna får parametrar och därför: antingen ingen lösningar eller oändligt många lösningar. Oavsett -ej bas  
Om  $a = 3$ : då blir det samma problem som  $a = 2$

Svar: de tre vektorer är en bas om  $a$  inte är 2 eller 3

**Definition: Punkter i planet**

Från origo: alla  $P = (x, y)$  och  $\overrightarrow{OP} = \begin{pmatrix} x \\ y \end{pmatrix}$

$$\begin{aligned} \text{Om } A = (a_1, a_2) \text{ och } B = (b_1, b_2) \text{ då är } \overrightarrow{AB} \\ = \begin{pmatrix} b_1 - a_1 \\ b_2 - a_2 \end{pmatrix} \end{aligned}$$

**Definition: Längd**

$$\text{Längd vektor i plan } |\vec{v}| = \sqrt{a^2 + b^2}$$

$$\text{Längd vektor i rum } |\vec{v}| = \sqrt{a^2 + b^2 + c^2}$$

## 7.3 Skalärprodukt och vektorprodukt

### 7.3.1 Skalärprodukt

$$\vec{u} \bullet \vec{v} = |\vec{u}| |\vec{v}| \cos \theta$$

$$\vec{u} \bullet \vec{v} = u_1 v_1 + u_2 v_2 + \dots + u_3 v_3$$

### Räkneregler

$$\vec{u} \bullet \vec{v} = \vec{v} \bullet \vec{u}$$

$$\vec{u} \bullet (\vec{v} + \vec{w}) = \vec{u} \bullet \vec{v} + \vec{u} \bullet \vec{w}$$

$$\lambda(\vec{u} \bullet \vec{v}) = (\lambda \vec{u}) \bullet \vec{v} = \vec{u} \bullet (\lambda \vec{v})$$

$$\vec{u} \bullet \vec{u} = |\vec{u}|^2$$

$$\vec{u} \bullet \vec{u} = 0 \Leftrightarrow \vec{u} = \vec{0}$$

### Exempel: parrallel och ortogonal

För vilka värden på  $a$  och  $b$  är vektorerna i

$$\mathbb{R}^3 \begin{pmatrix} 1 \\ a \\ 2 \end{pmatrix} \text{ och } \begin{pmatrix} b \\ 8 \\ a \end{pmatrix}$$

(a) parallel?, (b) ortognala?

(a)

$$\begin{cases} k = b \\ ak = 8 \\ 2k = a \end{cases} \Rightarrow$$

$$\begin{cases} k = b \\ (2k)k = 8 \Rightarrow k^2 = 4 \Rightarrow k = 2 \\ 2k = a \end{cases}$$

$$k = b = 2, a = 4$$

(b)

$$\begin{pmatrix} 1 \\ a \\ 2 \end{pmatrix} \bullet \begin{pmatrix} b \\ 8 \\ a \end{pmatrix} = 1b + a2 + 2a = b + 4a = 0$$

**Exempel: Längd-formeln**

$$|\vec{v}| = \sqrt{|\vec{v}|^2} = \sqrt{|\vec{v}| \bullet \vec{v}}$$

Beräkna längden av (ON-bas):  $\vec{v} = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}$

$$\begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix} \bullet \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix} = 2^2 + 1^2 + 2^2 = 9 \Rightarrow$$

$$|\vec{v}| = \sqrt{9} = 3$$

Därmed är längden: 3

**Exempel: Vinkel-formeln**

Låt  $\vec{u}$  och  $\vec{v}$  vara vektorer och vinkel blir då:

$$\theta = \arccos \frac{\vec{u} \bullet \vec{v}}{|\vec{u}| |\vec{v}|}$$

Beräkna vinkeln av (ON-bas):

$$\vec{u} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \text{ och } \vec{v} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \bullet \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = 1 \cdot 2 + 2^2 + 2 \cdot 1 = 8$$

$$|\vec{u}| = \sqrt{1^2 + 2^2 + 2^2} = \sqrt{9} = 3$$

$$|\vec{v}| = \sqrt{2^2 + 2^2 + 1^2} = \sqrt{9} = 3$$

$$\arccos \frac{\vec{u} \bullet \vec{v}}{|\vec{u}| |\vec{v}|} = \arccos \frac{8}{3 \cdot 3} = \arccos \frac{8}{9} \approx 27.27^\circ$$

**Exempel: Längd av två vektorer**

Låt  $u$  och  $v$  vara två vektorer sådana att

$$|\vec{u}| = 4, |\vec{v}| = 2 \text{ och vinkeln mellan } \vec{u} \text{ och } \vec{v} \text{ är } \frac{2\pi}{3}$$

Bestäm längden av  $3\vec{u} - 2\vec{v}$

$$\begin{aligned} \sqrt{|3\vec{u} - 2\vec{v}|^2} &= \sqrt{9|\vec{u}|^2 + 4|\vec{v}|^2 - 2|\vec{u}||\vec{v}| \cos \frac{2\pi}{3}} \\ &= \sqrt{9 \cdot 16 + 4 \cdot 4 - 4 \cdot 3 \cdot 8 \cdot \frac{-1}{2}} = \sqrt{13 \cdot 16} = 4\sqrt{13} \end{aligned}$$

**Exempel: beräkna skalärprodukten**

$$\underline{u} \begin{pmatrix} 9 \\ 9 \end{pmatrix} \bullet \underline{u} \begin{pmatrix} -2 \\ -1 \end{pmatrix}$$

$$\underline{u} = (\vec{u}_1, \vec{u}_2), \vec{u}_1 \bullet \vec{u}_1 = 9, \vec{u}_1 \bullet \vec{u}_2 = 6, \vec{u}_2 \bullet \vec{u}_2 = 8$$

$$\underline{u} \begin{pmatrix} 9 \\ 9 \end{pmatrix} \bullet \underline{u} \begin{pmatrix} -2 \\ -1 \end{pmatrix} = (9\vec{u}_1 - 2\vec{u}_2) \bullet (9\vec{u}_1 - 1\vec{u}_2)$$

$$= -18|\vec{u}_1|^2 + (-9 - 18)\vec{u}_1 \bullet \vec{u}_1 - 9|\vec{u}_2|^2$$

$$= -18 \cdot 9 - 27 \cdot (6) - 9 \cdot 8 = -396$$

**7.3.2 Ortogonal projektion**

Hitta punkt på linjen som är närmast en punkt. Punkten är ortogonala (vinkelräta)

Parallell koposant skrivs  $\vec{v}_{||l}$  och ortogonal skrivs  $\vec{v}_{\perp l}$ .

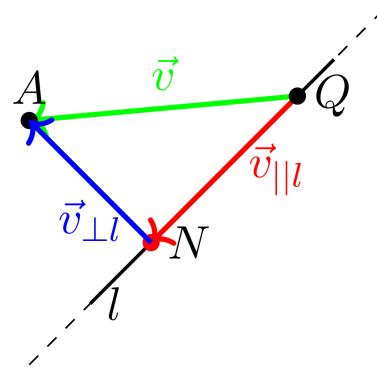


Figure 7.1: Ortogonal projektion

**Ortogonal projektion Formeln**

$$\vec{v}_{||l} = \frac{\vec{u} \bullet \vec{v}}{|\vec{u}|^2} \vec{u}$$

**Exempel: Hitta parallell och ortogonal kom-**

**posanten**

$$\vec{v} = \vec{v}_{\perp l} + \vec{v}_{\parallel l}$$

$$\vec{v}_{\parallel l} = \frac{\vec{u} \bullet \vec{v}}{|\vec{u}|^2} \vec{u}$$

Beräkna ortogonal komposanten av:

$$\vec{v} = \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix} \text{ och } \vec{u} = \begin{pmatrix} -4 \\ 4 \\ 2 \end{pmatrix}$$

Beräkna först parrallel komposanten:

$$\vec{v}_{\parallel \vec{u}} = \frac{\vec{u} \bullet \vec{v}}{|\vec{u}|^2} \vec{u} = \frac{-18}{36} \vec{u} = \begin{pmatrix} 2 \\ -2 \\ 1 \end{pmatrix}$$

Ortogonal komposanten blir då:

$$\vec{v}_{\perp \vec{u}} = \vec{v} - \vec{v}_{\parallel \vec{u}} = \begin{pmatrix} 1-2 \\ -2+2 \\ 3-1 \end{pmatrix}$$

$$\text{Svar: } \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}$$

**Exempel: Närmaste punkt och avståndet**

Bestäm avståndet från punkten  $P = (-2, 4, 3)$

till linjen

$$\begin{cases} x + 2y - z = 1 \\ x - y + 5z = 4 \end{cases}$$

Finn även den punkt på linjen  $l$  som ligger närmast punkten  $P$

Skriver ekvationen på parameter form

$$\begin{aligned} & \left( \begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 1 & -1 & 5 & 4 \end{array} \right) \xrightarrow[-1]{+} \left| \begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 0 & -3 & 6 & 3 \end{array} \right| \cdot 1/3 \\ & \sim \left( \begin{array}{ccc|c} 1 & 2 & -1 & 1 \\ 0 & 1 & -2 & -1 \end{array} \right) \xrightarrow[-2]{+} \sim \\ & \left( \begin{array}{ccc|c} 1 & 0 & 3 & 3 \\ 0 & 1 & -2 & -1 \end{array} \right) \xrightarrow[-1]{+} \sim \\ & l : \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ 0 \end{pmatrix} + t \begin{pmatrix} -3 \\ 2 \\ 1 \end{pmatrix} \text{ dvs } (p_0 + t\vec{v}) \end{aligned}$$

Hittar en godtycklig punkt ex  $t = 1$

$$\Rightarrow Q = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

(1.) beräknar vektor från godtyckliga punkten till den givna.

$$\overrightarrow{PQ} = \begin{pmatrix} 0 - (-2) \\ 1 - 4 \\ 1 - 3 \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ -2 \end{pmatrix}$$

(2.) Beräknar ortogonala projectionen.

$$\begin{aligned} \overrightarrow{NQ} &= \overrightarrow{PQ}_{\parallel v} = \frac{\begin{pmatrix} 2 \\ -3 \\ -2 \end{pmatrix} \bullet \begin{pmatrix} -3 \\ 2 \\ 1 \end{pmatrix}}{9+4+1} \begin{pmatrix} -3 \\ 2 \\ 1 \end{pmatrix} \\ &= \frac{-14}{14} \begin{pmatrix} -3 \\ 2 \\ 1 \end{pmatrix} \end{aligned}$$

(3.) Beräknar punkten från närmaste punkt till.

$$\begin{aligned} \overrightarrow{ON} &= \overrightarrow{OQ} + \overrightarrow{QN} = \overrightarrow{OQ} - \overrightarrow{NQ} \\ &= \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} -3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -3 \\ 3 \\ 2 \end{pmatrix} \end{aligned}$$

(4.) Beräknar längden

$$\begin{aligned} & \left| \begin{pmatrix} -2 \\ 4 \\ 3 \end{pmatrix} - \begin{pmatrix} -3 \\ 3 \\ 2 \end{pmatrix} \right| \\ &= \sqrt{1^2 + 1^2 + 1^2} = \sqrt{3} \end{aligned}$$

Svar: avståndet är  $\sqrt{3}$  och punkten är  $(-3, 3, 2)$

**Exempel: Spegling**

Bestäm speglingen av punkten  $A : (1, 3, -3)$  i planet  $\pi$  som går genom origo och innehåller punkterna  $(-1, 1, 1)$  och  $(3, 3, 1)$

Beräknar planets ekvation

$$\vec{n} = \overrightarrow{OQ} \times \overrightarrow{OP} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \\ 6 \end{pmatrix} = 2 \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}$$

$-x + 2y + 3z = 0$  Eftersom den går egenom origo

$$\begin{aligned} \overrightarrow{NA} &= \overrightarrow{OA}_{||\vec{n}} = \frac{\begin{pmatrix} 1 \\ 3 \\ -3 \end{pmatrix} \times \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}}{1+4+9} \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix} \\ &= \frac{14}{14} \begin{pmatrix} -1 \\ 2 \\ -3 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ -3 \end{pmatrix} \end{aligned}$$

Beräkna speglingen rita då är det uppenbart

$$\overrightarrow{OA'} = \overrightarrow{OA} - 2\overrightarrow{NA} = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix} - 2 \begin{pmatrix} -1 \\ 2 \\ -3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 3 \end{pmatrix}$$

**Exempel Hitta punkt ortogonal projek-****tion**

Låt  $l$  vara linjen genom punkten  $Q = (1, 2, 3)$  parallell med vektorn

$$\vec{u} = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}$$

Hitta den punkt  $N$  på  $l$  som är närmast  $A = (1, 7, 4)$

Lösning: Rita figur och tolka. Vi ortogonalt projeicerar

$$\vec{v} = \overrightarrow{QA} = \begin{pmatrix} 1-1 \\ 7-2 \\ 4-3 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ 1 \end{pmatrix} \text{ på vektorn } \vec{u}$$

$$\vec{v}_{||l} = \vec{v}_{||\vec{u}} = \frac{\begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \bullet \begin{pmatrix} 0 \\ 5 \\ 1 \end{pmatrix}}{\begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \bullet \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}} \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} = \frac{25}{50} \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}$$

Detta är ju igen vektorn som pekar frpn  $Q$  till närmaste punkten som därför blir

$$N = \left( 1 - \frac{1}{2}3, 2 - \frac{1}{2}4, 3 - \frac{1}{2}5 \right) = \left( -\frac{1}{2}, 0, \frac{1}{2} \right)$$

**Kontroll:** kollar om  $\overrightarrow{NQ} = k\vec{u}$ ,  $k \in \mathbb{R}$

$$\begin{pmatrix} 1 - (-1/2) \\ 2 - 0 \\ 3 - 1/2 \end{pmatrix} = \begin{pmatrix} 3/2 \\ 2 \\ 5/2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \text{ (stämmer)}$$

**Räkneregler ortogonal**

För vektorer  $\vec{u}$ ,  $\vec{v}$  och  $\vec{w}$  och skalär  $\lambda \in \mathbb{R}$

$$(\lambda\vec{v})_{||\vec{u}} = \lambda(\vec{v}_{||\vec{u}})$$

$$(\vec{v} + \vec{w})_{||\vec{u}} = \vec{v}_{||\vec{u}} + \vec{w}_{||\vec{u}}$$

$$(\lambda\vec{v})_{\perp\vec{u}} = \lambda(\vec{v}_{\perp\vec{u}})$$

$$(\vec{v} + \vec{w})_{\perp\vec{u}} = \vec{v}_{\perp\vec{u}} + \vec{w}_{\perp\vec{u}}$$

**7.3.3 Enhetsvektorer och ON-baser**

En enhetsvektor är en vektor med längd 1. Om man har en vektor  $\vec{u}$  kan man skala om den med ett positivt tal så den får längd 1. En bas  $\underline{u} = (\vec{u}_1 \vec{u}_2 \dots \vec{u}_n)$  kallas en ortonormal bas (ON-bas) omm:

- (1) Alla vektorerna är enhetsvektorer  $|\vec{u}_1|^2 = \vec{u}_i \cdot \vec{u}_j = 1$   
 (2) Varje par av vektorer  $\vec{u}_i$  och  $\vec{u}_j$  för  $i \neq j$  är ortogonala:  $\vec{u}_i \cdot \vec{u}_j = 0$   $\hat{\vec{u}} = \frac{1}{|\vec{u}|} \vec{u}$

Skalärproducten har samma räkneregler i ON-baser som i standard bas

### 7.3.4 Vektorprodukten

I högersystem kan representeras av en höger hand där tumen är vektorn 1 pekfingret är 2 och längfingret är 3. **Definition: högersystem**

En bas  $\underline{u} = (\vec{u}_1 \vec{u}_2 \vec{u}_3)$  kallas ett högersystem omm: set ifrån spetsen av  $\vec{u}_3$  vridas  $\vec{u}_1$  moturs till  $\vec{u}_2$

**Definition: Vektorprodukten** Givet två icke-parallelala vektorer  $\vec{u}$  och  $\vec{v}$  med vinkel  $\theta$  mellan dem definieras  $\vec{u} \times \vec{v}$  som den entydiga vektor som uppfyller:

- (a)  $\vec{u} \times \vec{v}$  är ortogonal mot båda  $\vec{u}$  och  $\vec{v}$
- (b) längden av  $\vec{u} \times \vec{v}$  är  $|\vec{u} \vec{v} \sin \theta|$
- (c)  $(\vec{u} \vec{v} \vec{u} \times \vec{v})$  är ett högersystem

I fallet där  $\vec{u}$  och  $\vec{v}$  är parallella är  $\vec{u} \times \vec{v} = \vec{0}$ .

### Definition: produkter

Sala om: tal  $\cdot$  vektor = vektor

Skalär produkt: vektor  $\bullet$  vektor = tal

Vektor produkt: vektor  $\times$  vektor = vektor

### Räkneregler: Vektorprodukten

För vektorer  $\vec{u}, \vec{v}, \vec{w}$  och ett tal  $\lambda$  gäller

$$\vec{u} \times \vec{v} = -\vec{v} \times \vec{u}$$

$$\vec{u} \times (\vec{v} + \vec{w}) = \vec{u} \times \vec{v} + \vec{u} \times \vec{w}$$

$$\lambda(\vec{u} \times \vec{v}) = \lambda(\vec{u}) \times \vec{v} = \vec{u} \times (\lambda \vec{v})$$

### Vektorprodukten metologi

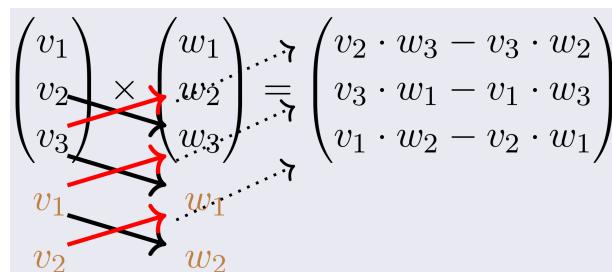


Figure 7.2: Vektorprodukten. From

### Exempel: Hitta basen

Låt  $\vec{u}_1 = \begin{pmatrix} 4 \\ 8 \\ 1 \end{pmatrix}$  och  $\vec{u}_2 = \begin{pmatrix} 4 \\ -1 \\ -8 \end{pmatrix}$

Verifiera att dessa är ortogonala, och hitta en vektor  $\vec{u}_3$

Så att  $\underline{u} = (\hat{\vec{u}}_1 \hat{\vec{u}}_2 \hat{\vec{u}}_3)$  är en ON-bas  
(och skriv ut denna)

Lösning: Först kollar vi att skalärprodukten är 0 :

$$\begin{pmatrix} 4 \\ 8 \\ 1 \end{pmatrix} \bullet \begin{pmatrix} 4 \\ -1 \\ -8 \end{pmatrix} = 16 - 8 - 8 = 0$$

Därmed ortogonala. Nu måste vi hitta en vektor som är ortogonal mot båda vektorprodukten är just det

$$\vec{u}_3 = \begin{pmatrix} 4 \\ 8 \\ 1 \end{pmatrix} \times \begin{pmatrix} 4 \\ -1 \\ -8 \end{pmatrix} \begin{pmatrix} -64 + 1 \\ 4 + 32 \\ -4 - 32 \end{pmatrix} = \begin{pmatrix} -63 \\ 36 \\ -36 \end{pmatrix}$$

Längderna av dessa är (räkar vara lika)

$$|\vec{u}_2|^2 = |\vec{u}_1|^2 = 4^2 + (\pm 8)^2 + (\pm 1)^2 = 81$$

och vi kunna göra liknade beräkning för  $\vec{u}_3$ , men vi vet redan att:

$$|\vec{u}_3| = |\vec{u}_1||\vec{u}_2| \sin \theta = 9 \cdot 9 \cdot 1 = 81$$

Desras tre normering blir därför:

$$\begin{aligned} \hat{\vec{u}}_1 &= \frac{1}{9} \begin{pmatrix} 4 \\ 8 \\ 1 \end{pmatrix}, \hat{\vec{u}}_2 = \frac{1}{9} \begin{pmatrix} 4 \\ -1 \\ -8 \end{pmatrix}, \hat{\vec{u}}_1 = \frac{1}{81} \begin{pmatrix} -63 \\ 36 \\ -36 \end{pmatrix} \\ &= \frac{1}{9} \begin{pmatrix} -7 \\ 4 \\ -4 \end{pmatrix} \end{aligned}$$

Så basen är:  $\underline{u} = \left( \frac{1}{9} \begin{pmatrix} 4 \\ 8 \\ 1 \end{pmatrix}, \frac{1}{9} \begin{pmatrix} 4 \\ -1 \\ -8 \end{pmatrix}, \frac{1}{81} \begin{pmatrix} -63 \\ 36 \\ -36 \end{pmatrix} \right)$

### Exempel: Uttryck vektorer i varandra

## (2.5.8.b)

Uttryck  $w$  i  $u$  och  $v$

$$|u| = 6, |v| = 8, |w| = 7, u \text{ och } v \text{ bildar vinkeln } \frac{\pi}{6},$$

$$u \text{ och } w \text{ vinkeln } \frac{\pi}{2} \text{ och } v \text{ och } w \text{ vinkeln } \frac{2\pi}{3}$$

Altså ska hitta  $\vec{w} = k_1 \vec{u} + k_2 \vec{v}, k_1, k_2 \in \mathbb{R}$

Vi ser att  $k_2 < 0$  enligt bild, därmed får vi följande

$$-k_2 \cdot \cos \frac{\pi}{3} = 7 \Rightarrow -k_2 \cdot \frac{1}{2} \cdot 8 = 7 \Rightarrow k_2 = \frac{7}{4}$$

Beräknar  $k_1$

$$6k_1 = \frac{7}{4} \cdot 8 \cdot \sin \frac{\pi}{3} \Rightarrow k_1 = \frac{7}{6}\sqrt{3}$$

$$\vec{w} = \frac{7\sqrt{3}}{6}\vec{u} - \frac{7}{4}\vec{v}$$

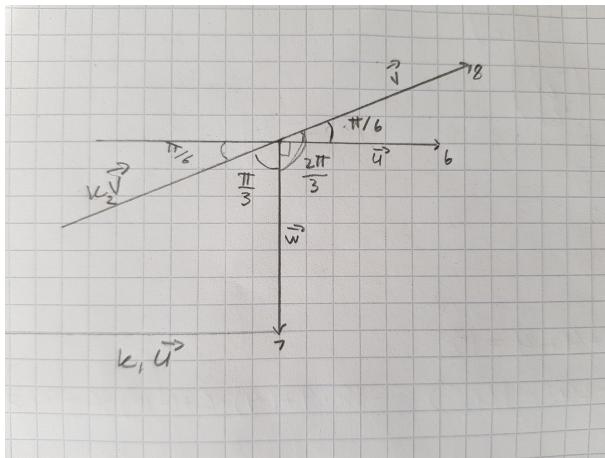


Figure 7.3: 2.5.8.b

### 7.3.5 Area och Volym

Area parallelogram:  $|\vec{u}| |\vec{v}| \sin \theta = |\vec{u} \times \vec{v}|$

Volym parallelogram:  $|(\vec{u} \times \vec{v}) \bullet \vec{w}|$

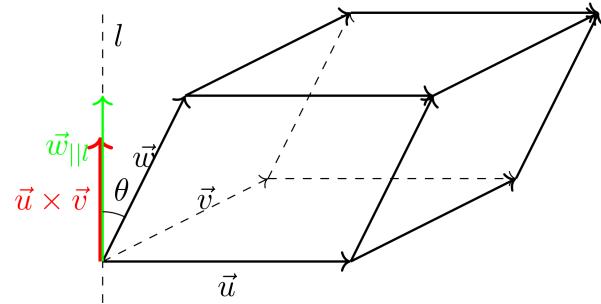


Figure 7.4: Ortogonal projektion

### Exempel: Hitta volymen

Hitta volymen av parallellipipeden med sidorna

$$\vec{u} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \vec{v} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \vec{w} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

och avgör om  $(\vec{u} \vec{v} \vec{w})$  är ett högersystem, ett vänstersystem, eller ej en bas.

Lösning: Vi beräknar

$$\vec{u} \times \vec{v} = \begin{pmatrix} 1 \cdot 1 - 1 \cdot 1 \\ 1 \cdot 2 - 1 \cdot 1 \\ 1 \cdot 1 - 1 \cdot 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

$$(\vec{u} \times \vec{v}) \bullet \vec{w} = 0 + 1 - 2 = -1$$

Så det är ett vänstersystem och volymen är 1

## 7.4 Linjer och plan

$$\text{Parameterform: } \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + t \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}, t \in \mathbb{R}$$

$$\text{Normalform: } ax + by = c \Rightarrow \vec{n} = \begin{pmatrix} a \\ b \end{pmatrix}$$

$$\text{Plan: } \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} + t \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} + s \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

**Exempel: Skriv i parameterform**

Om vi lösar ekvationen (vars lösningar är en linje):

$$x + 3y = 4;$$

$$y = t \Rightarrow x = 4 - 3t \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix} + t \begin{pmatrix} -3 \\ 1 \end{pmatrix}$$

Hitta parameterform för linje genom givna punkter

$$A = \begin{pmatrix} -21 \\ 20 \end{pmatrix}, B = \begin{pmatrix} -24 \\ -22 \end{pmatrix}$$

$$\overline{AB} = \overline{OB} - \overline{OA} = \begin{pmatrix} -24 - (-21) \\ -22 - 20 \end{pmatrix} = \begin{pmatrix} -3 \\ -42 \end{pmatrix} \Rightarrow$$

$$L : \underline{e} \begin{pmatrix} x \\ y \end{pmatrix} = \underline{e} \begin{pmatrix} -21 \\ 20 \end{pmatrix} + t \underline{e} \begin{pmatrix} -3 \\ -42 \end{pmatrix}$$

**Exempel: Hitta planets ekvation**

Uppgift: Hitta en ekvation för planet som går genom  $(1, 2, 3)$  och är parallell med vektorerna

$$\vec{u} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \quad \vec{v} = \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}$$

Lösnig: Vi måste hitta en vektor  $\vec{n}$  som är ortogonal mot planet

Inte så lätt i 3D som för linje i 2D. Men vi har en formel.

$$\vec{n} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \times \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 45 - 48 \\ 42 - 36 \\ 32 - 35 \end{pmatrix} = \begin{pmatrix} -3 \\ 6 \\ -3 \end{pmatrix}$$

Det kan vi skriva på normal formen:

$$-3x + 6y - 3z = c$$

För att hitta  $c$  så insätter vi det kända punkten

$$(1, 2, 3)$$

$$c = -3 \cdot 1 + 6 \cdot 2 - 3 \cdot 3 = 0$$

$$\text{Vi får formen: } -3x + 6y - 3z = 0$$

**Exempel: Beskriva linje på normalform**

Uppgift: Hitta normalvektorn för linje

$$A = (16, 4), B = (9, 12)$$

$$\overline{AB} = \begin{pmatrix} 9 - 16 \\ 12 - 4 \end{pmatrix} = \begin{pmatrix} -7 \\ 8 \end{pmatrix}$$

$$L : \overline{OA} + t \overline{AB} = \begin{pmatrix} 16 \\ 4 \end{pmatrix} + t \begin{pmatrix} -7 \\ 8 \end{pmatrix}$$

$$\left\{ \begin{array}{l} x = 16 - 7t \\ y = 4 + 8t \end{array} \right.$$

$$t = \frac{16 - x}{7} = \frac{y - 4}{8} \Leftrightarrow 128 - 8x = 7y - 21$$

$$8x + 7y = 149$$

**Exempel: Skärning mellan plan**

$$\left\{ \begin{array}{l} -3x + y + 4z = 4 \\ x - 4y - 5z = -5 \end{array} \right.$$

$$\left\{ \begin{array}{l} 0 - 11y - 11z = -11 \\ x - 4y - 5z = -5 \end{array} \right.$$

$$\left\{ \begin{array}{l} 0 + y + z = 1 \\ x - 4y - 5z = -5 \end{array} \right.$$

$$\left\{ \begin{array}{l} x + 0 - z = -1 \\ 0 + y + z = 1 \end{array} \right.$$

$$\begin{pmatrix} t - 1 \\ 1 - t \\ t \end{pmatrix}$$

**Exempel: Skärning mellan plan och linjen**

$$3x + 3y + 4z = -7$$

$$\left\{ \begin{array}{l} x = 2 - 3t \\ y = 1 - 3t \\ z = 3t \end{array} \right.$$

$$3(2 - 3t) + 3(1 - 3t) + 4(3t) = -7 \Rightarrow -6t = -16$$

$$\Rightarrow t = \frac{8}{3}$$

$$\left\{ \begin{array}{l} x = 2 - 3t = -6 \\ y = 1 - 3t = -7 \\ z = 3t = 8 \end{array} \right.$$

### 7.4.1 ortogonal projektion på linje

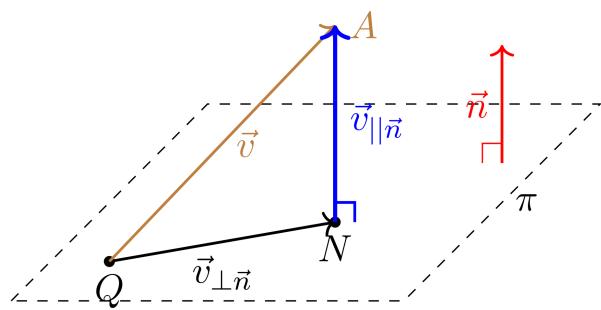


Figure 7.5: Ortogonal projektion på plan

**Exempel:** Hitta närmaste punkten i planet

Uppgift: hitta närmaste punkten till punkten  $A = (1, -5, 2)$  i planet  $P: x + 2y - z = 1$ . Hitta även avståndet mellan  $A$  och planet

Lösning: Först väljer vi godtycklig punkt  $Q = (1, 0, 0)$  i planet, och beräknar

$$\vec{v} = \overrightarrow{QA} = \begin{pmatrix} 1-1 \\ -5 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ -5 \\ 2 \end{pmatrix}$$

Normalvektor till planet  $x + 2y - z = 1$

$$\vec{n} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$

$$\vec{v}_{\parallel \vec{n}} = \frac{0 \cdot 1 + (-5) \cdot 2 + 2 \cdot (-1)}{1^2 + 2^2 + (-1)^2} \vec{n} = -2\vec{n}$$

Så vi får koordinaterna:

$$\overrightarrow{ON} = \overrightarrow{OA} - \overrightarrow{AN} = \overrightarrow{OA} - \vec{v}_{\parallel \vec{n}}$$

$$= \begin{pmatrix} 1 \\ -5 \\ 2 \end{pmatrix} - (-2)\vec{n} = \begin{pmatrix} 3 \\ -1 \\ 0 \end{pmatrix}$$

så närmaste punkten är  $N = (3, -1, 0)$

Avståndet är  $| -2\vec{n} | = 2\sqrt{6}$

Hitta den punkt på linjen

$$l : \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} + t \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, t \in \mathbb{R}$$

som är närmast linjen

$$k : \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix} + s \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, s \in \mathbb{R}$$

$$A = (-1 + t, t, t), B = (-1 + s, 2s, -1 + s)$$

$$\overrightarrow{AB} = \begin{pmatrix} (-1 + s) - (-1 + t) \\ (2s) - (t) \\ (-1 + s) - (t) \end{pmatrix} = \begin{pmatrix} s - t \\ 2s - t \\ -1 + s - t \end{pmatrix}$$

Vektorn ska vara ortogonal mot:  $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$

$$\begin{cases} 1(s - t) + 1(2s - t) + 1(-1 + s - t) = 0 \\ 1(s - t) + 2(2s - t) + 1(-1 + s - t) = 0 \end{cases}$$

$$= \begin{cases} -1 + 4s + -3t = 0 \\ -1 + 6s + -4t = 0 \end{cases}$$

$$\Rightarrow s = \frac{3t + 1}{4} \Rightarrow -1 + \frac{3}{2}(3t + 1) - 4t = 0$$

$$\begin{cases} t = -1 \\ s = -1/2 \end{cases} \text{ (Kontrollera)} - 1 - 2 + 3 = 0,$$

$$-1 - 3 + 4 = 0$$

$$A = (-2, -1, -1), B = (-3/2, -1, -3/2)$$

$$\Rightarrow \overrightarrow{AB} = \begin{pmatrix} 1/2 \\ 0 \\ -1/2 \end{pmatrix}$$

Svar: punkten på linjen  $l$  är  $A = (-2, -1, -1)$

## 7.5 Matrisräkning

**Begräpp:** matriser

diagonal

huvuddiagonalen

Kummutterar:  $AB = BA$

$A = (a_{ij})_{r \times k}$

Rang: antalet ledande kofisenter i trappsteksmatris

**Exempel:** Hitta närmaste punkten på linje till

**Räkneregler: matriser**

Addition: endast i sama form

Multiplication:  $(r \times k)(k \times m) \Rightarrow r \times m$

$$(AB)C = A(BC)$$

$$\lambda(AB) = (\lambda A)B = A(\lambda B)$$

$$A(B + C) = AB + AC$$

$$(B + C)A = BA + CA$$

Villkor:
Kvadratisk

$$\underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 3 & -1 & 5 \end{pmatrix}}_A$$

$B$

$\begin{pmatrix} 1 & 1 & 5 \\ 1 & 0 & 2 \\ 2 & -1 & 3 \end{pmatrix}$

$\begin{pmatrix} 3 & 1 & 9 \\ 12 & -2 & 28 \end{pmatrix}$

Figure 7.6: Ortogonal projektion på plan

**Exempel: Multiplication av matriser**

$$\begin{pmatrix} 2 & 3 & 4 \\ -1 & 2 & 2 \\ -5 & -2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 2a & 3b & 4c \\ -a & 2b & 2c \\ -5a & -2b & c \end{pmatrix}$$

**Definition: Enhetsmatrisen**

Enhetsmatrisen  $I_n$

$$A^0 I_n = I_n$$

$$A : 2 \times 2 A I_2 = A$$

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

**7.5.1 Transponat****Definition: Transponaten**

$A^t$  betyder inte  $A$  upphöjt till t

$$A = (a_{ij})_{r \times k} \Rightarrow A^t = A = (\alpha_{ij})_{k \times r}$$

$$A = (\alpha_{ij}) = (a_{ij})$$

**Exempel: Transponaten**

Symmetrisk  $A = A^t$

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}^t = \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix}$$

**Räkneregler: Transponaten**

$$(A + B)^t = A^t + B^t$$

$$(\lambda A)^t = \lambda A^t$$

$$(A^t)^t = A$$

$$(AB)^t = B^t A^t$$

**7.5.2 Matrisinvers****Räkneregler: Matrisinvers**

$$AB = BA = I$$

Inversen finns endast om matrisen är kvadratisk  
 $A$  är inventerbara

$AX = B$  har entydiga lösningar för alla  $B$

$AX = 0$  har enbart lösningen  $X = 0$

$A$  har rangn

$A \sim I$  (är radekvivalent med)  $I$

**Exempel: Matrisinvers 3x3**

$$\begin{aligned} \left( \begin{array}{ccc} 1 & 2 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & -1 \end{array} \right) &\sim \left( \begin{array}{ccc|ccc} 1 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 \end{array} \right) \\ &\sim \left( \begin{array}{ccc|ccc} 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 \end{array} \right) \\ &\sim \left( \begin{array}{ccc|ccc} 1 & 0 & -1 & -1 & 2 & 0 \\ 0 & 1 & 1 & 1 & -1 & 0 \\ 0 & 1 & 1 & 1/2 & -1/2 & -1/2 \end{array} \right) \\ &\sim \left( \begin{array}{ccc|ccc} 1 & 0 & 0 & -1/2 & 3/2 & -1/2 \\ 0 & 1 & 0 & 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 & 1/2 & -1/2 & -1/2 \end{array} \right) \\ &\sim A^{-1} = \left( \begin{array}{ccc} -1/2 & 3/2 & -1/2 \\ 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \end{array} \right) \\ &= \frac{1}{2} \left( \begin{array}{ccc} -1 & 3 & -1 \\ 1 & -1 & 1 \\ 1 & -1 & -1 \end{array} \right) \end{aligned}$$

**Räkneregler:** Matrisinvers

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

## 7.6 Determinanter

Ekvations system har unika lösningar då koefficientmatrisen är inverterbar. Som är ekvivalent med att determinanten är nollskild.

**Exempel:** determinant många led genväg

$$\begin{pmatrix} 3 & 2 & -3 & 24 & 1005 \\ 0 & 1 & 23 & 14 & 15 \\ 0 & 0 & 3 & 7 & -5 \\ 0 & 0 & 2 & 4 & 3 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

Vi det finns endast två produkter som inte innehåller en nol faktor

$$\begin{pmatrix} (3) & 2 & -3 & 24 & 1005 \\ 0 & (1) & 23 & 14 & 15 \\ 0 & 0 & (3) & (7) & -5 \\ 0 & 0 & (2) & (4) & 3 \\ 0 & 0 & 0 & 0 & (5) \end{pmatrix}$$

$$= 3 \cdot 1 \cdot 3 \cdot 4 \cdot 5 - 3 \cdot 1 \cdot 6 \cdot 2 \cdot 5 = 0$$

**Exempel:** determinant 3x3

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} =$$

$$= 1 \cdot 5 \cdot 9 - 1 \cdot 6 \cdot 8 - 2 \cdot 4 \cdot 9 +$$

$$+ 2 \cdot 6 \cdot 7 + 3 \cdot 4 \cdot 8 - 3 \cdot 5 \cdot 7 =$$

$$= 45 - 48 - 72 + 84 + 96 - 105 = 0$$

**Sats:** Om  $B$  är matrisen  $A$  där man har bytt om på rad  $i$  och  $j$  är  $\det A = -\det B$

1. Bytt två rader om i  $A$  ändras determinaten sitt tecken
2. Skalas en rad om med  $\lambda$  skalas också determinaten om med  $\lambda$
3. Addera en rad något på en annan rad i  $A$  ändrar inte denna determinant
4.  $\det(A) = \det(A^t)$

$$5. \det(AB) = \det(A)\det(B)$$

**Exempel:** determinant 3x3

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & 8 & 7 \\ 1 & 3 & 6 & 5 \\ 0 & 0 & 2 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & 8 & 7 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 2 & 3 \end{pmatrix} = - \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 1 \\ 0 & 3 & 8 & 7 \\ 0 & 0 & 2 & 3 \end{pmatrix}$$

$$= - \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 2 & 3 \end{pmatrix} = - \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 0 & 11 \end{pmatrix}$$

$$= -1 \cdot 1 \cdot (-1) \cdot 11 = 11$$

**Exempel:** okänt x (determinant)

$$\begin{pmatrix} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ x & x^2 & x^4 & x^5 \\ x^2 & x^3 & x^4 & x^6 \end{pmatrix}$$

$$x \begin{pmatrix} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ 1 & x & x^3 & x^4 \\ x^2 & x^3 & x^4 & x^6 \end{pmatrix} = \text{rad } 4 - \text{rad } 1x^3$$

$$\begin{pmatrix} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ 1 & x & x^3 & x^4 \\ 1 & x & x^2 & x^4 \end{pmatrix} = x^3 \begin{pmatrix} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ 1 & x & x^3 & x^4 \\ 0 & 0 & 0 & x \end{pmatrix}^{R4}$$

$$= \text{rad } 3 - \text{rad } 1x^3(x^4 - x^3) \begin{pmatrix} 1 & x & x^2 \\ 1 & x^2 & x^3 \\ 1 & x & x^3 \end{pmatrix}$$

$$= x^6(x-1)^2 \begin{pmatrix} 1 & x & x^2 \\ 1 & x^2 & x^3 \\ 0 & 0 & x \end{pmatrix}$$

$$= x^6(x-1)(x^3 - x^2) \begin{pmatrix} 1 & x \\ 1 & x^2 \end{pmatrix} = x^9(x-1)^3 = 0$$

### 7.6.1 Ko-faktorna

$$\det(A) = a_{i1}C_{i1} + a_{i2}C_{i2} + \dots + a_{in}C_{in}$$

där  $a_{ij}$  är teknet o  $C_{ij}$  är kofisienten

$\tilde{A}^t$  är alla kofaktorerna från  $A C_{ij}$

$$\det(a^{-1}) = \det(1/\det(a))$$

$$\begin{aligned} & \left( \begin{array}{cccc} 2 & 1 & 1 & -1 \\ (1) & (0) & (2) & (0) \\ 1 & -2 & 1 & 2 \\ 3 & 1 & -1 & 5 \end{array} \right)^{R2} \\ &= (-1)^{2+1} \cdot 1 \cdot \begin{pmatrix} 1 & 1 & -1 \\ -2 & 1 & 2 \\ 1 & -1 & 5 \end{pmatrix} \\ &+ (-1)^{2+2} \cdot 0 \cdot \begin{pmatrix} 2 & 1 & -1 \\ 1 & 1 & 2 \\ 3 & -1 & 5 \end{pmatrix} \\ &+ (-1)^{2+3} \cdot 2 \cdot \begin{pmatrix} 2 & 1 & -1 \\ 1 & -2 & 2 \\ 3 & 1 & 5 \end{pmatrix} \\ &+ (-1)^{2+4} \cdot 0 \cdot \begin{pmatrix} 2 & 1 & 1 \\ 1 & -2 & 1 \\ 3 & 1 & -1 \end{pmatrix} \\ &- 1(5 + 2 - 2 - (-1 - 2 - 10)) + 0 \\ &- 2(-20 + 6 - 1 - (6 + 4 + 5)) + 0 \\ &= -18 - 2 \cdot (-30) = 42 \end{aligned}$$

### 7.6.2 Geometri: parallellepiped

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \bullet \left( \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \times \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \right)$$

### 7.7 Vektorer i $\mathbb{R}^n$

Skallär produkt och ärmad längd är samma regler som innan. Vektor mellan punkter är också samma regler. Vinkeln är samma som innan (orthogonal projection). Vinkel formel:  $\theta = \arccos \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|}$  (Cauchy-Schwarz olikheten) logist!  $|\vec{v} \bullet \vec{w}| \leq |\vec{v}| |\vec{w}|$  ( $|3 - 1| \leq |3| - 1|$ )

Linjärt oberoende: inga parametrar vid lösning av  $c_1\vec{v}_1 + c_2\vec{v}_2 + \dots + c_n\vec{v}_n = \vec{0}$ . Linjära Höljet (spannen): mängden av alla möjliga linjärakombinationer av vektorerna. Det linjära höljet av vektorerna är hela  $\mathbb{R}^n$  omm  $RangV = n$ .

Bas definnerar lika dant i fler dimensioner, för att vara en bas  $\mathbb{R}^n$  måste de vara linjärt oberoende och det linjära höljet är hela  $\mathbb{R}^n$ .

## 7.8 Linjära avbildningar $\mathbb{R}^n \rightarrow \mathbb{R}^m$

### 7.8.1 Matristransformationer och linjära funktioner

#### Terminologi

Standardmatris: En matris som multipliseras med argumentet för att få svaret

Sammansättning: Låt  $f : A \rightarrow B, g : B \rightarrow C$

$$(g \circ f)(x) = g(f(x))$$

Vektor produkt:  $F : \mathbb{R}^3 \rightarrow \mathbb{R}^3$

$$F(\vec{x}) = \vec{a} \times \vec{x}$$

**Definition:** En funktion  $T : \mathbb{R}^k \rightarrow \mathbb{R}^n$  kallas linjär om den uppfyller:

För alla  $\vec{v}, \vec{w} \in \mathbb{R}^k$  och  $\lambda \in \mathbb{R}$  gäller

$$(i) \quad T(\vec{v} + \vec{w}) = T(\vec{v}) + T(\vec{w})$$

$$(ii) \quad T(\lambda \vec{v}) = \lambda T(\vec{v})$$

#### Formel standard matris

$$\begin{aligned} [T] & \left( \begin{array}{cccc} | & | & \dots & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_k \\ | & | & \dots & | \end{array} \right) \\ &= \left( \begin{array}{cccc} | & | & \dots & | \\ T(\vec{v}_1) & T(\vec{v}_2) & \dots & T(\vec{v}_k) \\ | & | & \dots & | \end{array} \right) \end{aligned}$$

**Exempel: hitta standard matris**

Hitta standardmatrisen  $[T]$  för den linjära funktionen  $T : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  som uppfyller att

$$\begin{aligned} T\begin{pmatrix} 0 \\ -4 \end{pmatrix} &= \begin{pmatrix} 8 \\ 8 \\ -8 \end{pmatrix} \wedge T\begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ -4 \end{pmatrix} \\ [T]\begin{pmatrix} 0 & -1 \\ -1 & -1 \end{pmatrix} &= \begin{pmatrix} 8 & -1 \\ 8 & 0 \\ -8 & -4 \end{pmatrix} \\ [T] &= \begin{pmatrix} 8 & -1 \\ 8 & 0 \\ -8 & -4 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ -1 & -1 \end{pmatrix}^{-1} = \begin{pmatrix} 3 & -2 \\ 2 & -2 \\ 2 & 2 \end{pmatrix} \end{aligned}$$

Kontrol: multiplisera standard matrisen med input och få output

**Exempel: hitta standard matris ortogonal projection**

Hitta standardmatrisen för  $P : \mathbb{R}^3 \rightarrow \mathbb{R}^3$   
där  $P$  är den ortogonala projektionen på planet  
 $\pi : 2x + y + 3z = 0$

$$T(\vec{n}) = T\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Hittar två godtykliga punkter på planet:

$$A = (1, 1, -1), B = (-1, 2, 0)$$

$$\begin{aligned} \overrightarrow{OA} &= \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}, \overrightarrow{OB} = \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix} \\ [T]\begin{pmatrix} 3 & 1 & -1 \\ 1 & 1 & 1 \\ 2 & -1 & 0 \end{pmatrix} &= \begin{pmatrix} 0 & 1 & -1 \\ 0 & 1 & 1 \\ 0 & -1 & 0 \end{pmatrix} \\ \Rightarrow [T] &= \begin{pmatrix} 0 & 1 & -1 \\ 0 & 1 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 3 & 1 & -1 \\ 1 & 1 & 1 \\ 2 & -1 & 0 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 5/7 & -1/7 & -3/7 \\ -1/7 & 13/14 & -3/14 \\ -3/7 & -3/14 & 5/14 \end{pmatrix} \end{aligned}$$

Kontrol: multiplisera standard matrisen med input och få output

**Exempel: hitta standard matris spegling**

Hitta standardmatrisen för  $P : \mathbb{R}^3 \rightarrow \mathbb{R}^3$   
där  $P$  är speglingen i planet  $\pi : -x + 2y - 2z = 0$

$$T(\vec{n}) = T\begin{pmatrix} -1 \\ 2 \\ -2 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}$$

Hittar två godtykliga punkter på planet:

$$A = (0, 1, 1), B = (2, 1, 0)$$

$$\begin{aligned} \overrightarrow{OA} &= \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \overrightarrow{OB} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \\ [T]\begin{pmatrix} -1 & 0 & 1 \\ 2 & 1 & 2 \\ -2 & 1 & 0 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 1 \\ -2 & 1 & 2 \\ 2 & 1 & 0 \end{pmatrix} \\ \Rightarrow [T] &= \begin{pmatrix} 1 & 0 & 1 \\ -2 & 1 & 2 \\ 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \\ 2 & 1 & 2 \\ -2 & 1 & 0 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 7/9 & 4/9 & -4/9 \\ 4/9 & 1/9 & 8/9 \\ -4/9 & 8/9 & 1/9 \end{pmatrix} \end{aligned}$$

Kontrol: multiplisera standard matrisen med input och få output

### 7.8.2 Injektiv/Surjektiv/Bijektiv

Termenologi

[T]	kolonnvektorerna i [T]	funktionen T
$\text{rang}([T])=k$	linjär oberoende	injektiv
$\text{rang}([T])=n$	spannet är hela $\mathbb{R}^n$	surjektiv
$n=\text{rang}([T])=k$	är en bas för $\mathbb{R}^n = \mathbb{R}^k$	bijektiv

## Chapter 8

# Linear Algebra II

## 8.1 Grudläggande teori

### 8.1.1 Ekvationssystem och matrisräkning

$$\left\{ \begin{array}{l} x + 2y + z = -1 \\ 2x + (a+3)y + 3z = -4 \\ x + (3-a)y + (a-2)z = a-1 \end{array} \right.$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 2 & a+3 & 3 & -4 \\ 1 & 3-a & a-2 & a-1 \end{array} \right)$$

(-1 rad1 till rad2), (-2 rad1 till rad3)

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 0 & a-1 & 1 & -2 \\ 0 & 1-a & a-3 & a \end{array} \right)$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 0 & a-1 & 1 & -2 \\ 0 & 0 & a-2 & a-2 \end{array} \right)$$

$a \neq 1 \wedge a \neq 2$

$$(\frac{1}{a-2} \text{rad } 3)$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 0 & a-1 & 1 & -2 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

(-1rad 3 till rad 2), (-1rad 3 till rad 1)

$$\left( \begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & a-1 & 0 & -3 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

$$(\frac{1}{a-1} \text{rad } 2)$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & a-1 & 0 & -3 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

$$\left( \begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & 1 & 0 & \frac{3}{1-a} \\ 0 & 0 & 1 & 1 \end{array} \right)$$

(-2rad 2 till rad 3)

$$\left( \begin{array}{ccc|c} 1 & 0 & 0 & -2 - \frac{6}{1-a} \\ 0 & 1 & 0 & \frac{3}{1-a} \\ 0 & 0 & 1 & 1 \end{array} \right)$$

$$\left\{ \begin{array}{l} x = -2 - \frac{6}{1-a} \\ y = \frac{3}{1-a} \\ z = 1 \end{array} \right.$$

**Kontroll:** stoppar in x,y,z i ekvationerna

$$\left\{ \begin{array}{l} \left( \frac{2a-8}{1-a} \right) + 2 \cdot \left( \frac{3}{1-a} \right) + 1 = -1 \\ 2 \cdot \left( \frac{2a-8}{1-a} \right) + (a+3) \cdot \left( \frac{3}{1-a} \right) + 3 \cdot 1 = -4 \\ \left( \frac{2a-8}{1-a} \right) + (3-a) \cdot \left( \frac{3}{1-a} \right) + (a-2) \cdot 1 = a-1 \end{array} \right.$$

### 8.1.2 determinanter

$$\begin{aligned}
 & \left( \begin{array}{cccc} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ x & x^2 & x^4 & x^5 \\ x^2 & x^3 & x^4 & x^6 \end{array} \right) \\
 & x \left( \begin{array}{cccc} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ 1 & x & x^3 & x^4 \\ x^2 & x^3 & x^4 & x^6 \end{array} \right) \\
 & = \text{rad } 4 - \text{rad } 1x^3 \left( \begin{array}{cccc} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ 1 & x & x^3 & x^4 \\ 1 & x & x^2 & x^4 \end{array} \right) \\
 & = x^3 \left( \begin{array}{cccc} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^3 & x^4 \\ 1 & x & x^3 & x^4 \\ 0 & 0 & 0 & x \end{array} \right)^{R4} \\
 & = \text{rad } 3 - \text{rad } 1x^3(x^4 - x^3) \left( \begin{array}{ccc} 1 & x & x^2 \\ 1 & x^2 & x^3 \\ 1 & x & x^3 \end{array} \right) \\
 & = x^6(x-1)^2 \left( \begin{array}{ccc} 1 & x & x^2 \\ 1 & x^2 & x^3 \\ 0 & 0 & x \end{array} \right) \\
 & = x^6(x-1)(x^3 - x^2) \left( \begin{array}{c} 1 \\ 1 \\ x^2 \end{array} \right) = x^9(x-1)^3 = 0
 \end{aligned}$$

### 8.1.3 flerdimisionel dvs $R^n$

räkneregler

### 8.1.4 Funktioner

polynom funktioner vid en viss grad också

### 8.1.5 Linjer

$$\begin{aligned}
 l : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix} + t \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \\
 \text{Då är riktnings vektorn } &\begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \\
 \text{Och går genom } &\begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix}
 \end{aligned}$$

## 8.2 Vektorrum

**Definition: vektor rum** En mängd  $\mathbb{V}$  kallas för en reellt vektorrum om:

1. Det finns en operator på  $\mathbb{V}$  som kallas addition och beräknas med  $+$ , sådant att om  $\vec{u}, \vec{v} \in \mathbb{V}$  så gäller  $\vec{u} + \vec{v} \in \mathbb{V}$ .
2. Det finns en poeration på  $\mathbb{V}$  som kallas skalning eller multipliseras med reella tal, som betecknas med  $\cdot$ , sådan att om  $\lambda \in \mathbb{R} \wedge \vec{v} \in \mathbb{V}$  så gäller  $\lambda \cdot \vec{v} \in \mathbb{V}$  räknereglerna gäller som i la1

**Axiomen: vektor rum**

1.  $\vec{u} + \vec{v} = \vec{v} + \vec{u}$  (Kommutativ lag)
2.  $\vec{u}(\vec{v} + \vec{w}) = (\vec{u} + \vec{v}) + \vec{w}$  (Associativ lag)
3. Det finns ett nollement  $\vec{0}$  så att  $\vec{v} + \vec{0} = \vec{v}$
4. Till varje  $\vec{v} \in \mathbb{V}$  finns ett element  $-\vec{v}$  så att  $\vec{v} + (-\vec{v}) = 0$
5.  $1 \cdot \vec{v} = \vec{v}$
6.  $\lambda \cdot (\mu \cdot \vec{v}) = (\lambda \cdot \mu) \cdot \vec{v}$  (Associativ lag)
7.  $(\lambda + \mu) \cdot \vec{v} = \lambda \cdot \vec{v} + \mu \cdot \vec{v}$  (Distributiv lag)
8.  $\lambda \cdot (\vec{u} + \vec{v}) = \lambda \cdot \vec{u} + \lambda \cdot \vec{v}$  (Distributiv lag)

$\forall \lambda, \mu \in \mathbb{R} \wedge \vec{u}, \vec{v}, \vec{w} \in \mathbb{V}$

### 8.3 Underrum och linjära hörjet

Ett underrum är en delmängd  $u \neq 0$  som är sluten under addition och skalning

**Definition: Underrum** En delmängd  $\mathbb{U}$  av ett vektorrum  $\mathbb{V}$  kallas för ett underrum eller delrum av  $\mathbb{V}$  om  $\mathbb{U}$  är ett vektorrum med den addition och den multiplikation med reella tal som definierats i  $\mathbb{V}$ .

**Sats: Underrum**

En icketom mängd  $\mathbb{U}$  av ett vektorrum  $\mathbb{V}$  är ett underrum om och endast om följande gäller

1. Om  $\vec{u} \in \mathbb{U}$  och  $\vec{v} \in \mathbb{U}$ , Så är  $\vec{u} + \vec{v} \in \mathbb{U}$
2. Om  $\vec{u} \in \mathbb{U}$  och  $\lambda \in \mathbb{R}$ , Så är  $\lambda\vec{u} \in \mathbb{U}$

**Regler: Underrum**

1. Det snabbaste sättet att testa om det gäller är om nollvektorn finns om den inte gör det så bryter det mot 2-lagen
2. Homogena ekvationer och linjer/plan genom origo gäller det alltid för
3. Kontunerligt deriverbara, så gäller det att det är ett underrum

**Exempel: Underrum (Ex 1)**

$$\text{är } \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid x + y + z = 5 \right\} \text{ ett underrum?}$$

Nej, då:  $0 + 0 + 0 \neq 5$

**Definition: linjära hörjet** En delmängd  $\mathbb{U}$  av ett vektorrum  $\mathbb{V}$  kallas för ett underrum eller delrum av  $\mathbb{V}$  om  $\mathbb{U}$  är ett vektorrum med den addition och den multiplikation med reella tal som definierats i  $\mathbb{V}$ .

**Exempel: Underrum (Ex 4)**

Vilka vektorer i  $\mathbb{P}$  tillhör  $u = [1, 1+x, 1+x+x^2]$

Svar:  $u = P_2$  (eftersom)  $1 \in u, x \in u$ , och  $x^2 \in u$

Då alla  $a \cdot 1 + b \cdot x + c \cdot x^2 \in u$

Inga polynom av grad  $> 2$  tillhör  $u$ .

Vi för kombinera det olika elementen i  $u$  med addition och multiplication

### 8.4 Linjärt oberoende

**Ide: linjärt beronde**

Om vektorerna kan skrivas om en linjär kombination av det andra vektorerna

så är vektor linjärt beroende då det inte behöver vektor vi får reda på beroende genom att ställa upp ekv  $x_1\vec{v}_1 + x_2\vec{v}_2 + \dots + x_n\vec{v}_n = \vec{0}$

Om denna ekvation har icke triviala lösningar (ej noll) då är den beroende

**Exempel: om linjärt oberoende**

Är mängden  $\left\{ \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right\}$  i  $\mathbb{R}^2$  linjärt oberoende

$$x_1 \begin{pmatrix} 1 \\ 2 \end{pmatrix} + x_2 \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \vec{0} \Leftrightarrow \left( \begin{array}{cc|c} 1 & 2 & 0 \\ 2 & 1 & 0 \end{array} \right)$$

$x_1 = 0, x_2 = 0$  Endast triviala lösningar

### 8.5 Bas

**Ide: Bas och Dimension** Bas omm om vektorerna spannar upp hella spannet och det linjärt oberoende Standard basen  $e$ . Vi behöver tree vektorer för att utgöra en bas i  $\mathbb{R}^3$  är  $\vec{v}_1, \vec{v}_2, \vec{v}_3 \in \mathbb{R}^3$ .

Dimensioner är antallet oberoende vektorer som spänner upp "rummet" dimensioner blir då samma som antallet element som en bas av polynom"  $\mathbb{P}_3 = a + bx^1 + cx^2 + dx^3$  har 4 dimensioner  $\mathbb{R}^3$  har 3 dimensioner Dimensioner för matriser är  $n \times m$  ( $2 \times 2 - \text{matris} = \text{dim}4$ ) En bas med tre vektorer där vektorerna  $\mathbb{R}^4$  ger 3 dim

**Exempel:** Ta fram bas från plan

$$\text{låt } 2x - yz = 0$$

$$\text{Lösning: } x = y/2 - z/2$$

$$\text{Låt } y = s, z = t \text{ vi får då lösningen på parameter form} \\ \text{Lös } T_{\underline{f}}^e = \left( \begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 0 & -1 & 2 \\ 0 & 1 & 1 & -1 \end{array} \right)$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} s/2 - t/2 \\ s \\ t \end{pmatrix} = s/2 \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} + t/2 \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}$$

$$\text{Basen blir då } \begin{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} & \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \end{pmatrix}$$

## 8.6 Basomvandling

Vi kan ta inversen av matrisen av vektorerna som utgör en bas och få matrisen som kan multipliseras med vektor för att få omvandlingen.

**Sats:** basbyte

$$\vec{v}_{\underline{e}} = T \vec{v}_f \text{ Där } T \text{ är en matris}$$

$$T_{\underline{e}}^f = (T_{\underline{f}}^e)^{-1}$$

$$\vec{v}_f = T_{\underline{f}}^e \vec{v}_{\underline{e}}$$

**Exempel:** Från standar till annan bas

$$\text{låt } \underline{f} = (f_1 f_2) = \left( \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \right), \vec{v} = \begin{pmatrix} -2 \\ 5 \\ 1 \end{pmatrix}$$

Utryck  $\vec{v}$  i basen  $\underline{f}$

$$\vec{v}_{\underline{e}} = \begin{pmatrix} -2 \\ 5 \\ 1 \end{pmatrix} = x_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + x_2 \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

$$\begin{cases} 2x_1 + x_2 = -2 \\ x_1 + 2x_2 = 5 \\ x_1 + x_2 = 1 \end{cases} \Rightarrow \left( \begin{array}{cc|c} 2 & 1 & -2 \\ 1 & 2 & 5 \\ 1 & 1 & 1 \end{array} \right)$$

$$\sim \left( \begin{array}{cc|c} 2 & 1 & -2 \\ 0 & 1 & 4 \\ 0 & 0 & 0 \end{array} \right)$$

$$x_2 = 4, x_1 = 1/2(-2 - 4) = -3 \Rightarrow \vec{v}_f = \begin{pmatrix} -3 \\ 4 \\ 0 \end{pmatrix}$$

**Exempel:** Finn matrisen för bas byte vek-

torer

Låt  $\underline{f} = \left( \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right)$  Bestäm bas omvandling matrisen

**Exempel:** Finn matrisen för bas byte polynom

$$\text{låt } \underline{e} = (1 \ x) \text{ och } \underline{f} = (1 \ 1 - x)$$

$$\vec{f}_1 = 1 = \vec{e}_1 \Rightarrow \vec{f}_{1e} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\vec{f}_2 = 1 - x = \vec{e}_1 - \vec{e}_2 \Rightarrow \vec{f}_{2e} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$T_{\underline{e}}^f = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} \Rightarrow (T_{\underline{e}}^f)^{-1} T_{\underline{f}}^e = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$$

**Exempel:** Finn matrisen för bas byte mellan olika matriser

Basbyte mellan  $\underline{f} = \left( \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right)$  och  $\left( \begin{pmatrix} -1 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} \right)$

$$\text{lös } T_{\underline{f}}^g : \left( \begin{array}{cc|cc} 1 & 2 & -1 & 3 \\ 1 & 1 & 1 & 2 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 0 & 3 & 1 \\ 0 & 1 & -2 & 1 \end{array} \right) \Rightarrow \begin{pmatrix} 3 & 1 \\ -2 & 1 \end{pmatrix}$$

**Definition:** Nollrum, Kolonrum, raddrum.

låt A vara en  $m \times n$ -matris

1.  $\dim(\text{A:s kolonrum}) = \dim(\text{A:s raddrum}) = \text{rang A}$
2.  $\dim(\text{A:s kolonrum}) + \dim(\text{A:s nollrum}) = n$

**Exempel:** Finn Nollrum, Kolonrum,

### radrum.

Hitta en bas för kolonrummet, radrummet och nollrummet till följande matris. Bestäm även rummens dimensioner

$$A = \begin{pmatrix} 1 & 2 & 3 & 3 \\ 1 & 3 & 4 & 5 \\ -1 & 0 & -1 & 1 \end{pmatrix}$$

Låt kolonernana vara vektorer, då får vi följande gäller

$$x_1v_1 + x_2v_2 + x_4v_3 + x_4v_4 = 0$$

$$\Leftrightarrow (v_1 \ v_2 \ v_3 \ v_4) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = 0 \Leftrightarrow Ax = 0$$

Gausselimination ger oss  $A \sim \begin{pmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -s+t \\ -s-2t \\ s \\ t \end{pmatrix} = s \begin{pmatrix} -1 \\ -1 \\ 1 \\ 0 \end{pmatrix} + t \begin{pmatrix} 1 \\ -2 \\ 0 \\ 1 \end{pmatrix}$$

Basen för nollrummet blir då  $\begin{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \\ 0 \end{pmatrix} & \begin{pmatrix} 1 \\ -2 \\ 0 \\ 1 \end{pmatrix} \end{pmatrix}$

Basen för kolonrummet blir då det vektorer i A som har Pivot element dvs  $v_1, v_2$

$$\left( \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix} \right)$$

Basen för radrummet blir  $\begin{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ -1 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \end{pmatrix} \end{pmatrix}$

### Exempel: Derivatan av polynom

Är följande en linjär avbildning

$$F : C^1(a, b) \rightarrow C(a, b) \text{ definierad genom } F(f) = \frac{df}{dx}$$

$$\left( \frac{d}{dx}(af(x) + bg(x)) \right) = a \frac{df}{dx} + b \frac{dg}{dx}$$

$$\left( \frac{d}{dx}(\lambda af(x)) \right) = \lambda \left( \frac{d}{dx}(af(x)) \right)$$

Svar: ja  $F$  är en linjär avbildning

## 8.8 Matrisen av en linjär avbildning

$$\begin{array}{ccc} \mathbb{V} & \xrightarrow{F} & \mathbb{W} \\ \underline{\mathbf{v}} \cdot ? \uparrow & & \downarrow (?)_{\underline{\mathbf{w}}} \\ \mathbb{R}^n & \dashrightarrow^{(F)_{\underline{\mathbf{w}}}} & \mathbb{R}^m \end{array}$$

$$(F)_{\underline{\mathbf{w}}}^{\underline{\mathbf{v}}}(\mathbf{x})_{\underline{\mathbf{v}}} = (F(\underline{\mathbf{v}}(\mathbf{x})_{\underline{\mathbf{v}}}))_{\underline{\mathbf{w}}} = (F(\mathbf{x}))_{\underline{\mathbf{w}}}$$

Figure 8.1: Matris av en linjär avbildning. From ?

### Exempel: Rotations matrisen

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

## 8.7 Linjär avbildning

### Definition: Linjär avbildning

låt  $\mathbb{V}$  och  $\mathbb{W}$  vara vektorrum. En funktion

$$F : \mathbb{V} \rightarrow \mathbb{W}$$

kallas för linjär avbildning omm

$$1. F(\vec{v} + \vec{w}) = F(\vec{v}) + F(\vec{w})$$

$$2. F(\lambda \cdot \vec{v}) = \lambda \cdot F(\vec{v})$$

**Exempel: Hitta matrisen**

Linjär avbildning  $F : \mathbb{P}_3 \rightarrow \mathbb{P}_3$ ,  $F = p + p' + p'' + p'''$   
 Ange  $F$ :s matris i standarbasen.

Avbildar varje element i standardbasen  $(1 \ x \ x^2 \ x^3)$

$$F(1) = 1 + 1' + 1'' + 1''' = 1$$

$$F(x) = x + x' + x'' + x''' = x + 1$$

$$F(x^2) = x^2 + x^{2'} + x^{2''} + x^{2'''} = x^2 + 2x + 2 \cdot 1$$

$$F(x^3) = x^3 + x^{3'} + x^{3''} + x^{3'''} = x^3 + 3x^2 + 6x + 6 \cdot 1$$

$$\begin{pmatrix} 1 & 1 & 2 & 6 \\ 0 & 1 & 2 & 6 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**nom**

Vad är matrisen av derivatan  $\mathbb{P}_3 \rightarrow \mathbb{P}_2$   
 med avseende på  
 basen  $\underline{u} = (x^3 \ x^2 \ x \ 1)$  av  $\mathbb{P}_3$  och  $\underline{v} = (x^2 \ x \ 1)$  av  $\mathbb{P}_2$ ?

$$(H)_{\underline{f}}^{\underline{e}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

$T_{\underline{v}}^{\underline{f}}$ :

$$\vec{f}_1 = 1 = 0\vec{v}_1 + 0\vec{v}_2 + 1\vec{v}_3$$

$$\vec{f}_2 = x = 0\vec{v}_1 + 1\vec{v}_2 + 0\vec{v}_3$$

$$\vec{f}_3 = x^2 = 1\vec{v}_1 + 0\vec{v}_2 + 0\vec{v}_3$$

$T_{\underline{e}}^{\underline{u}}$ :

$$\vec{u}_1 = x^3 = 0\vec{e}_1 + 0\vec{e}_2 + 0\vec{e}_3 + 1\vec{e}_4$$

$$\vec{u}_2 = x^2 = 0\vec{e}_1 + 0\vec{e}_2 + 1\vec{e}_3 + 0\vec{e}_4$$

$$\vec{u}_3 = x = 0\vec{e}_1 + 1\vec{e}_2 + 0\vec{e}_3 + 0\vec{e}_4$$

$$\vec{u}_4 = 1 = 1\vec{e}_1 + 0\vec{e}_2 + 0\vec{e}_3 + 0\vec{e}_4$$

$$(H)_{\underline{v}}^{\underline{u}} = T_{\underline{v}}^{\underline{f}}(H)_{\underline{f}}^{\underline{e}}T_{\underline{e}}^{\underline{u}} =$$

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

## 8.9 Basbyte av linjära avbildningar

**Definition: Basbyte av linjära avbildningar**

låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara linjär. Låt  $\underline{e}$  och  $\underline{v}$  vara baser i  $\mathbb{V}$  och låt  $\underline{f}$  och  $\underline{w}$  vara baser av  $\mathbb{W}$ . Då gäller  $(F)_{\underline{f}}^{\underline{e}} = T_{\underline{f}}^{\underline{w}}(F)_{\underline{w}}^{\underline{v}}T_{\underline{v}}^{\underline{e}}$  Där vektorn kommer från höger (viktigt vid vilken ordning transformations matriserna står)

Ide: Vi omvandlar från en bas (ex standard bas) till en bas som är mer anpassat för uträkningen. Sedan omvandlar vi igen för att få svaret i den bas vi vill ha den i (ex standard bas).

**Exempel: Basbyte av linjär avbildning vektor**

**Exempel: Basbyte av linjär avbildning poly-**

## 8.10 Kärna och bild av en linjär avbildning

**Definition: Kärna och bild**

Låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara en linjär avbildning

**Kärnan** eller **nollrummet** av den linjära avbildningen  $F$   
 $\ker(F) = N(F) = \{\vec{v} \in \mathbb{V} | F(\vec{v}) = \vec{0}\}$

**Bilden** eller **värderummet** av den linjära avbildningen  $F$   
 $\text{Im}(F) = V(F) = \{\vec{w} \in \mathbb{W} | \exists \vec{v} : F(\vec{v}) = \vec{w}\}$

**Definition: Isomorfism, injektiv, surjek-**

tiv

$F : X \rightarrow y$  kallas

**injektiv** om  $F(x) = F(x') \Rightarrow x = x'$

**surjektiv** om  $\forall y \in Y \exists x \in X : f(x) = y$

**bijektiv** om det är både injektiv och surjektiv

Sats:

Låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara linjär

$F$  är injektiv omm  $\ker F = \{\vec{0}\}$

$F$  är surjektiv omm  $\text{Im } F = \mathbb{W}$  omm  $\text{rang}((F)_{\underline{w}}) = \dim \mathbb{W}$

Sats: Dimensionssatsen

För varje linjär avbildning  $F : \mathbb{V} \rightarrow \mathbb{W}$  gäller

$$\dim \mathbb{V} = \dim \ker(F) + \dim \text{Im}(F)$$

Sats: Isomorfism

Låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara en linjär avbildning. De följande är ekvivalenta  $F$  är en isomorf

$\dim \mathbb{V} = \dim \mathbb{W}$  och  $F$  är injektiva

$\dim \mathbb{V} = \dim \mathbb{W}$  och  $F$  är surjektiva

$$\det((F)_{\underline{w}}^v) \neq 0$$

$(F)_{\underline{w}}^v$  är en kvadratisk matris av rang  $\dim \mathbb{V} = \dim \mathbb{W}$

## 8.11 Egenvärden och egenvektorer

**Definition: Egenvärde och egenvektor** Låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara en linjär avbildning. En vektor  $\vec{v} \neq \vec{0}$  kallas en egenvektor till  $F$  om det finns  $\lambda \in \mathbb{R}$  så att  $F(\vec{v}) = \lambda \vec{v}$ . I detta fall kallas  $\lambda$  en **egenvärde** till  $F$ .  $\mathbb{V}_{F,\lambda} = \{\vec{v} \in \mathbb{V} | F(\vec{v}) = \lambda \vec{v}\}$  kallas **egenrummet** av  $F$  till egenvärdet  $\lambda$ .

**Definition: Sekularpolynomet** För en matris  $A$  kallas polynomet  $\chi_A(\lambda) = \det(A - \lambda I_n)$  **sekularpolynomet** till  $A$ . Om  $A = (F)_{\underline{e}}^e$ , så kallas  $\chi_A$  också sekularpolynomet till  $F$ .

**Exempel:**

$F : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  matrisen i standardbasen är

$$\begin{pmatrix} 1 & 2 & 2 \\ 0 & 2 & 1 \\ -1 & 2 & 2 \end{pmatrix}$$

1. Hittar egenvärdena

$$F(\vec{x}) = \lambda \vec{x} \Leftrightarrow A\vec{x} = \lambda \vec{x}$$

$$A\vec{x} - \lambda \vec{x} = \vec{0} \Leftrightarrow (A - \lambda I)\vec{x} = 0$$

Där  $\lambda$  är egenvärdet, och  $\vec{x}$  är egenvektorn

$$A - \lambda I = \begin{pmatrix} 1 - \lambda & 2 & 2 \\ 0 & 2 - \lambda & 1 \\ -1 & 2 & 2 - \lambda \end{pmatrix}$$

$$\det(A - \lambda I) = 4 - 2\lambda - (2 - \lambda)(4 - 3\lambda + \lambda^2)$$

$$= (2 - \lambda)(-2 + 3\lambda - \lambda^2)$$

$$\det(A - \lambda I) = 0 \Leftrightarrow \lambda = 2 \vee \lambda = 1$$

2. Hittar egenvektorerna

$$\lambda = 2 :$$

$$(A - 2I)\vec{x} \Leftrightarrow \left( \begin{array}{ccc|c} 1 - 2 & 2 & 2 & 0 \\ 0 & 2 - 2 & 1 & 0 \\ -1 & 2 & 2 - 2 & 0 \end{array} \right) \sim \left( \begin{array}{ccc|c} 1 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \Leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2t \\ t \\ 0 \end{pmatrix} = t \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}, t \in \mathbb{R}$$

Så egenrummet till  $\lambda = 2$  är linjära hörjet:  $\left[ \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \right]$

$$\lambda = 1 :$$

$$(A - 1I)\vec{x} \Leftrightarrow \left( \begin{array}{ccc|c} 1 - 1 & 2 & 2 & 0 \\ 0 & 2 - 1 & 1 & 0 \\ -1 & 2 & 2 - 1 & 0 \end{array} \right) \sim \left( \begin{array}{ccc|c} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \Leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -t \\ -t \\ t \end{pmatrix} = t \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, t \in \mathbb{R}$$

Så egenrummet till  $\lambda = 1$  är linjära hörjet:

$$\left[ \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \right]$$

## 8.12 Diagonalisering

**Definition: Diagonalisering** Låt  $F : \mathbb{V} \rightarrow \mathbb{V}$  vara en linjär avbildning. Då kallas  $F$  **diagonaliserbar** om det finns en bas  $\underline{v}$  som består av egenvektorer till  $F$ . (Det betyder att  $(F)\underline{v}$  är en diagonalmatris).

**Definition: Algebraisk och Geometrisk multiplicitet** **Algebraisk multiplicitetet** av en egenvärde  $\lambda_0$  till en matris  $A$  är det maximala talet  $m$ , så att  $\chi_A(\lambda) = (\lambda - \lambda_0)^m p(\lambda)$  för någon polynom  $p$ .

**Algebraisk multiplicitetet** av en egenvärde  $\lambda_0$  till en matris  $A$  är  $\dim \mathbb{V}_{A,\lambda_0}$ .

### Exempel:

Avför om matrisen är diagonaliserbar och isåfäl vad är den matrisen

$$M = \begin{pmatrix} -2 & 4 & 4 \\ 0 & 2 & 4 \\ 0 & 0 & -2 \end{pmatrix}$$

1. Hittar egenvärden:

(se i kapitlet om enhetsvektorer)

$$(-2 - \lambda)(2 - \lambda)(-2 - \lambda) = 0$$

$$\lambda = 2 \text{ (alg mult 1)} \vee \lambda = -2 \text{ (alg mult 2)}$$

2. Hittar egenvekter:

(se i kapitlet om enhetsvektorer)

$$\lambda = 2 \Leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} t \\ t \\ 0 \end{pmatrix} = t \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, t \in \mathbb{R}$$

geo mult 1 därmed så är den hittills diagonaliserbar

$$\begin{aligned} \lambda = -2 &\Leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} s \\ -t \\ t \end{pmatrix} \\ &= s \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + t \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}, s, t \in \mathbb{R} \end{aligned}$$

geo mult 2 därmed diagonaliserbar

3. Sätter upp diagonala matrisen

$$\begin{aligned} F_{\underline{e}}^e &= T_{\underline{e}}^v F_{\underline{v}}^u T_{\underline{v}}^e \Leftrightarrow \begin{pmatrix} -2 & 4 & 4 \\ 0 & 2 & 4 \\ 0 & 0 & -2 \end{pmatrix} = \\ &\quad \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \end{aligned}$$

## 8.13 Inre produkrum

**Definition: Skalärprodukt** Låt  $\mathbb{V}$  vara ett vektorrum. En **skalärprodukt** på  $\mathbb{V}$  är en operation som tillordnar varje par av vektorer  $\vec{u}, \vec{v} \in \mathbb{V}$  en skalär  $(\vec{u}|\vec{v}) \in \mathbb{R}$  så att följande villkor gäller: bilinjär

1.  $(\vec{u} + \vec{v}|\vec{w}) = (\vec{u}|\vec{w}) + (\vec{v}|\vec{w})$
2. Symetrisk  $(\lambda \vec{u}|\vec{v}) = \lambda(\vec{u}|\vec{v}) = (\vec{u}|\lambda \vec{v})$
3. Positiv definit  $(\vec{u}|\vec{v}) = (\vec{v}|\vec{u})$

4. Om  $\vec{u} \neq \vec{0}$  så  $(\vec{u}|\vec{u}) > 0$

**Definition: Avstånd och vinklar** Låt  $\mathbb{V}$  vara ett vektorrum med enskärprodukt och  $\vec{u}, \vec{v} \in \mathbb{V}$

1. Längden (normen) av  $\vec{u}$  betecknas  $|\vec{u}|$  och defineras som  $|\vec{u}| = \sqrt{\vec{u}|\vec{u}|}$ .
2. Avståndet mellan  $\vec{u}$  och  $\vec{v}$  defineras som  $\vec{u} - \vec{v}$
3. Om  $\vec{u} \neq 0$  och  $\vec{v} \neq 0$  defineras vinkeln mellan  $\vec{u}$  och  $\vec{v}$  som  $\cos^{-1} \frac{(\vec{u}|\vec{v})}{|\vec{u}||\vec{v}|}$
4. Vi säger att  $\vec{u}$  och  $\vec{v}$  är ortogonala om  $(\vec{u}|\vec{v}) = 0$

**Definition: Trianglar**

Låt  $\mathbb{V}$  vara ett vektorrum med en skärprodukt och  $\vec{u}, \vec{v} \in \mathbb{V}$

Då bildar  $\vec{u}, \vec{v}$  och  $\vec{u} + \vec{v}$  en triangel

$$|\vec{u} + \vec{v}|^2 = |\vec{u}|^2 + |\vec{v}|^2 + 2(\vec{u}|\vec{v})$$

**Definition: Ortogonal projektion och komponent**

1.  $\vec{u}_{\parallel \vec{u}} := \frac{(\vec{v}|\vec{u})}{(\vec{u}|\vec{u})} \vec{u}$  kallas den ortogonala projektionen av  $\vec{v}$  på  $\vec{u}$ .
2.  $\vec{u}_{\perp \vec{u}} := \vec{v} - \vec{v}_{\parallel \vec{u}}$  kallas den ortogonala komponenten av  $\vec{v}$  med avseende på  $\vec{u}$ .

## 8.14 Symmetriska och positiva definita matriser

**Formel: skärprodukt uttrykt med matriser**  
 $(\vec{u}|\vec{y}) = \vec{u}^t A \vec{y}$  där  $A$  är  $(n \times n)$ -matrisen som uppfyller följande krav:

1. Symmetrisk om  $A^t = A$ .
2. Positivt definit om  $\vec{x}^t A \vec{x} > 0, \forall \vec{x} \in \mathbb{R}^n, \vec{x} \neq \vec{0}$ .

**Exempel: skärprodukt uttrykt med ma-**

triser

På  $\mathbb{R}^2$  definierar vi

$$(\vec{x}|\vec{y}) = x_1 y_1 + 3x_1 y_2 + ax_2 y_1 + bx_2 y_2$$

För vilka värden på  $a$  och  $b$  är detta en skärprodukt?  $a, b \in \mathbb{R}$

Lösning:

1. Symmetrisk:  $(\vec{x}|\vec{y}) = (\vec{y}|\vec{x})$   
 $= y_1 x_1 + 3y_1 x_2 + ay_2 x_1 + by_2 x_2$

$$(\vec{x}|\vec{y}) = \vec{x}^t \begin{pmatrix} 1 & 3 \\ a & b \end{pmatrix} \vec{y} \Rightarrow a = 3$$

pga summetri

2. Pos def:  $(\vec{x}|\vec{x}) > 0, \forall \vec{x} \neq \vec{0}$

$$\begin{aligned} (\vec{x}|\vec{x}) &= (x_1 x_2) \begin{pmatrix} 1 & 3 \\ a & b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= x_1^2 + 3x_1 x_2 + 3x_2 + x_1 + bx_2^2 \\ &= x_1^2 + 6x_1 x_2 + bx_2^2 = (x_1 + 3x_2)^2 + (b - 9)x_2^2 \end{aligned}$$

vilket endast är positivt då  $b > 9$

## 8.15 On-baser och Gram-Schmidt-ortonormalisering

**Definition: ON-bas**

Låt  $\mathbb{V}$  vara ett vektorrum med en skärprodukt.

En mängd  $\{\vec{u}_1 \dots \vec{u}_n\} \subseteq \mathbb{V}$  kallas ortonormal

(ON) om

$$(\vec{u}_i|\vec{u}_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

$$|\vec{u}_i| = 1$$

Dvs alla vektorer är ortogonala mot varandra och att längden ska vara 1

**Definition: Koordinater i en ON-bas**

Låt  $\underline{u} = (\vec{u}_1 \dots \vec{u}_n)$  vara en ON-bas i  $\mathbb{V}$  och  
 $\vec{v}, \vec{w} \in \mathbb{V}$ . Då är

$$1. \vec{v}_{\underline{u}} = \begin{pmatrix} (\vec{v}|\vec{u}_1) \\ \vdots \\ (\vec{v}|\vec{u}_n) \end{pmatrix}$$

$$2. (\vec{v}|\vec{w}) = (\vec{v}|\vec{u}_1)(\vec{w}|\vec{u}_1) + \dots + (\vec{v}|\vec{u}_n)(\vec{w}|\vec{u}_n) \\ = \vec{v}_n \bullet \vec{w}_n$$

**Definition: Ortogonal komplement** Låt  $\mathbb{V}$  vara ett vektorrum med en skalärprodukt och  $\mathbb{U} \subseteq \mathbb{V}$  ett underrum. Då är  $\mathbb{U}^\perp = \{\vec{v} \in \mathbb{V} | (\vec{u}|\vec{v}) = 0, \forall \vec{u} \in \mathbb{U}\}$  ett underrum som kallas det ortogonala komplementet till  $\mathbb{U}$ .

**Exempel: Hitta en ON-bas med Gram-Schmidt****ortonormalisering**

$$U = [\vec{u}_1, \vec{u}_2, \vec{u}_3] \subseteq \mathbb{R}^4, \vec{u}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \vec{u}_2 = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \vec{u}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Hitta ON-bas

$$[\vec{u}_1] \subseteq [\vec{u}_1, \vec{u}_2] \subseteq [\vec{u}_1, \vec{u}_2, \vec{u}_3] \\ \text{där } [\vec{u}_1] = U_1, [\vec{u}_1, \vec{u}_2] = U_2, [\vec{u}_1, \vec{u}_2, \vec{u}_3] = U_3$$

$$\text{ON-bas } U_1 : f_1 = \frac{1}{|\vec{u}_1|} \vec{u}_1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

$$\text{ON-bas } U_2 : f_1, f_2 \vec{u}_{2 \perp \vec{u}_1} = \vec{u}_2 - (\vec{u}_2|f_1)f_1$$

$$= \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} - \frac{1}{\sqrt{3}}(1+0+2+0) \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \\ = \begin{pmatrix} 0 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

$$f_2 = \frac{1}{\vec{u}_{2 \perp \vec{u}_1}} \vec{u}_{2 \perp \vec{u}_1} = \frac{1}{\sqrt{3}} \begin{pmatrix} 0 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

$$\text{ON-bas } U_3 : f_1, f_2, f_3 \vec{u}_{3 \perp \vec{u}_2} = \vec{u}_3 - (\vec{u}_3|f_1)f_1 - (\vec{u}_3|f_2)f_2$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} - \frac{1}{\sqrt{3}}(0+0+1+0) \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

$$- \frac{1}{\sqrt{3}}(0+0+1+1) \frac{1}{\sqrt{3}} \begin{pmatrix} 0 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} - \frac{1}{3} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} - \frac{2}{3} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} -1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\Rightarrow f_3 = \frac{1}{\sqrt{3}} \begin{pmatrix} -1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Kontrollera att  $f_1, f_2, f_3$  är ortogonala mot varandra

## 8.16 Isometriska avbildningar och spektralsatsen

### Definition: isometrier

Låt  $\mathbb{V}$  och  $\mathbb{W}$  vara vektorrum med skalärprodukter.

En linjär avbildning

$$F : \mathbb{V} \rightarrow \mathbb{W}$$

kallas en isometri om  $|F(\vec{v})| = |\vec{v}|$

En isometrisk linjär avbildning kallas också en isometri.

dvs. Så är längden oförendrad.  $F$  är en isometri omm  
 $(F(\vec{u})|F(\vec{v})) = (\vec{u}|\vec{v})$

### Egenskaper: isometrier

**Sats:** Låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara en isometri. Då är  $F$  injektiv

**Sats:** Låt  $F : \mathbb{V} \rightarrow \mathbb{W}$  vara en iventerbar isometri. Då är  $F^{-1} : \mathbb{W} \rightarrow \mathbb{V}$  en isometri

**Sats:** Låt  $F : \mathbb{U} \rightarrow \mathbb{V}$  och  $G : \mathbb{V} \rightarrow \mathbb{W}$  vara en isometrier. Då är  $G \circ F : \mathbb{U} \rightarrow \mathbb{W}$  en isometri

**Sats: spektralsatsen** Låt  $\mathbb{V}$  vara ett ändligt dimensionellt vektorrum med en skalär produkt och  $F : \mathbb{V} \rightarrow \mathbb{V}$  en linjär avbildning. Då är följande villkor ekvivalenta

1.  $F$  är symmetrisk.
2.  $\mathbb{V}$  har en ON-bas av bestående av egenvektorer till  $F$ .

Låt  $A$  vara en  $(n \times n)$ -matris. Då är följande villkor ekvivalenta

1.  $A$  är symmetrisk.
2. Det finns en ortonormal matris  $T$  så att  $D = T^{-1}AT$  är en diagonalmatris.

## 8.17 Andragradskurvor och andragradsytör

$$Q(x) = 1 \Leftrightarrow 1x^t Ax = 1 \Leftrightarrow y^t Dy = 1$$

$$\Leftrightarrow \lambda_1 y_1^2 + \lambda_2 y_2^2 = 1$$

Om  $\lambda_1 = \lambda_2$  så berkriver  $Q(x) = 1$  en cirkel

Om  $\lambda_1 > 0$  och  $\lambda_2 > 0$  så berkriver  $Q(x) = 1$  en ellips

Om  $\lambda_1 > 0$  och  $\lambda_2 < 0$  så berkriver  $Q(x) = 1$  en hyperbel

### Exempel: Bestäm formen

$$Q(\vec{x}) = \frac{14}{5}x_1^2 + \frac{11}{5}x_2^2 + \frac{14}{5}x_1x_2$$

$$Q(\vec{x}) = (x_1 \ x_2) \begin{pmatrix} 14/5 & 2/5 \\ 2/5 & 11/5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\begin{aligned} 1. \text{ Egenvärden } \det A &= \left(\frac{14}{5} - \lambda\right)\left(\frac{11}{5} - \lambda\right) - \frac{4}{25} \\ &= 0 \end{aligned}$$

$$\lambda_1 = 2 > 0$$

$$\lambda_2 = 3 > 0$$

Därmed så beskriver  $Q(x) = 1$  en ellips

2. Egenvektor får vi en ON-bas

3. Ställer upp ekvationen och ritar ut den

$$Q(\vec{y}) = y^{-1} \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \vec{y} = 2y_1^2 + 3y_2^2 = 1$$

## 8.18 System av linjära differentialekvationer

$$y' = Ay$$

$$T^{-1}AT = D$$

$$y = Tz \Rightarrow y' = Tz' \wedge y' = Ay \Leftrightarrow z' = dz$$

$$z = \begin{pmatrix} c_1 e^{\lambda_1 t} \\ c_2 e^{\lambda_2 t} \\ \vdots \\ \vdots \\ c_n e^{\lambda_n t} \end{pmatrix} \text{ där } c_i \in \mathbb{R}^n \text{ och } y = Tz$$

$$c = T^{-1}y_0 \text{ där } z(0) = c = \begin{pmatrix} c_1 \\ \vdots \\ \vdots \\ c_n \end{pmatrix}$$

**Exempel:** Bestäm formen

Lös följande system av differentialekvationer

$$\begin{cases} \frac{dx(t)}{dt} = 2x(t) + y(t) + z(t) \\ \frac{dy(t)}{dt} = x(t) + 2y(t) + z(t) \\ \frac{dz(t)}{dt} = x(t) + y(t) + 2z(t) \end{cases} \quad x(0) = 3, y(0) = 2, z(0) = 1$$

1. Skriver up systemet

$$\begin{pmatrix} \frac{dx(t)}{dt} \\ \frac{dy(t)}{dt} \\ \frac{dz(t)}{dt} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} \text{ Där matrisen är } A$$

2. Diagonaliseras

2.1 egenvärden

$\lambda_1 = 1$  (multiplicitet 2),  $\lambda_2 = 4$  (multiplicitet 1)

2.1 eigenvektor

$$\lambda_1 = 1 \Rightarrow A - I = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\Rightarrow \vec{v}_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \wedge \vec{v}_2 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

$$\lambda_2 = 4 \Rightarrow A - 4I = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix} \Rightarrow \vec{v}_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Diagonal ekv blir  $A = TDT^{-1}$

$$\Rightarrow T = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & -1 & 1 \end{pmatrix} \wedge D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

3. Finner den allmäna lösningen

$$u' = Du \Rightarrow \begin{cases} u_1 = c_1 e^t \\ u_2 = c_2 e^t \\ u_3 = c_3 e^{4t} \end{cases} \text{ Där } c_1, c_2, c_3 \in \mathbb{R}$$

$$v' = Av \text{ Där } A^{-1}v' = Tu \Rightarrow v = Tu$$

$$\begin{aligned} v &= \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} c_1 e^t \\ c_2 e^t \\ c_3 e^{4t} \end{pmatrix} \\ &= \begin{pmatrix} c_1 e^t + c_2 e^t \\ c_2 e^t + c_3 e^{4t} \\ -c_1 e^t - c_2 e^t + c_3 e^{4t} \end{pmatrix} \end{aligned}$$

4. Finner lösningen till begynelsevärdet

$$\left( \begin{array}{ccc|cc} 1 & 0 & 1 & 3 \\ 0 & 1 & 1 & 2 \\ -1 & -1 & 1 & 1 \end{array} \right) \sim \left( \begin{array}{ccc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{array} \right) \Rightarrow \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}$$

$$\begin{cases} x(t) = e^t + 2e^{4t} \\ y(t) = 2e^{4t} \\ z(t) = -et + 2e^{4t} \end{cases}$$



## Chapter 9

# Computer System with Project Work

## 9.1 M1

### 9.1.1 CPU

The central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

### 9.1.2 Register

A processor register is a quickly accessible location available to a computer's central processing unit (CPU). Registers usually consist of a small amount of fast storage. A CPU only has a small number of registers.

### 9.1.3 Memory

Memory refers to the computer hardware integrated circuits that store information for immediate use in a computer; it is synonymous with the term "primary storage". The memory is much slower than the CPU register but much larger in size. Sources

### 9.1.4 CPU context

At any point in time, the values of all the registers in the CPU defines the CPU context. Sometimes CPU state

### 9.1.5 Memory allocation

### 9.1.6 Kernal

The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system.

### 9.1.7 Multiprogramming

## 9.2 M2

### 9.2.1 Network communication

- TCP for a reliable byte stream service
- UDP for an unreliable message forwarding service
- Four different delivery models: uni/broad/multi/any-cast

- The client/server model is common in application design
- Sockets API can be used for network programming
  - A socket is a communication handle abstraction
  - Properties of a socket can be set through socket options

### 9.2.2 Processes

- Zombie: A terminated process is said to be a zombie or defunct until the
- Orphan: An orphan process is a process whose parent process has terminated, though it remains running itself.
- Signals: are a limited form of inter-process communication used in
- File descriptor: The kernel keeps a table with information about a process's open file descriptors.
- Pipes: An (anonymous) pipe is a simplex FIFO communication channel that may be used for

## 9.3 M3

- DNS är ett distribuerat system av servrar
- ARP mappar IP-address till länklageraddress
- En IP-address har två delar för att identifiera nätve

### 9.3.1 Dispatcher

Another component that is involved in the CPU-scheduling function is the dispatcher, which is the module that gives control of the CPU to the process selected by the short-term scheduler. It receives control in kernel mode as the result of an interrupt or system call.

The CPU scheduler selects one process from among the processes in memory that are READY to execute. The scheduler dispatcher then gives the selected process control

### 9.3.2 PCB

The process control block (PCB) is a data structure in the operating system kernel containing the information needed to manage a particular process.

The Long-term scheduler (LTS) (aka job scheduler) decides whether a new process should be brought into the ready queue in main memory or delayed.

The medium-term scheduler (MTS) temporarily remove processes from main memory and places them in secondary storage and vice versa, which is commonly referred to as "swapping in" and "swapping out".

### 9.3.3 IO bound/CPU bound

An I/O-bound process spends more time doing I/O than computations and is characterised

An CPU-bound process spends more time doing computations and is characterised by few very long

Interactive processes interact constantly with their human users.

Batch processes do not interact with human users.

Real-time processes have very strong scheduling requirements.

### 9.3.4 Schedular algorithems

The first come, first served (commonly called FIFO – first in, first out) process scheduling algorithm is the simplest process scheduling algorithm. Processes are executed on the CPU in the same order they arrive to the ready queue.

Shortest Job First (SJF) scheduling assigns the process estimated to complete fastest, i.e, the process with shortest CPU burst, to the CPU as soon as CPU time is available.

An extension of SJF where the currently running process is preempted if the CPU burst of a process A arriving to the ready queue is shorter than the remaining CPU burst of the currently running process.

A priority number (integer) is associated with each process. The CPU is allocated to the process with the highest priority (smallest integer = highest priority).

Round Robin (RR) is a scheduling algorithm where time slices are assigned to each process in equal portions and in circular order.

A multi-level queue scheduling algorithm is used in scenarios where the processes can be classified into groups based on properties like process type, CPU time, IO access, memory size, etc.

In computer science, starvation is a problem encountered in multitasking where a process is perpetually denied necessary resources. Without those resources, the process can never finish its task.

Ageing is used to ensure that jobs with lower priority will eventually complete their execution.

## 9.4 M4

### 9.4.1 Concurrency

The ability of different parts or units of a program, algorithm, or problem to be executed out-of-order or in partial order, without affecting

### 9.4.2 Parallelism

In parallel systems, two tasks are actually performed simultaneously. Parallelism is when tasks

### 9.4.3 Client-Server modle

A distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

### 9.4.4 Threads

A thread of execution is the smallest sequence of instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

### Atomic

In concurrent programming, an operation (or set of operations) is atomic if it appears to the rest of

the system to occur at once without being interrupted.

### Race condition

A race condition or race hazard is the behaviour of an electronic, software or other system where the output is dependent on the sequence or timing of other uncontrollable events.

### Data race

A data race occurs when two instructions from different threads access the same memory location

### 9.4.5 Locks

In general, any solution to the critical section problem requires a tool/abstraction - a lock.

### 9.4.6 Spinlock

A spinlock is a lock where a task simply waits in a loop ("spins") repeatedly checking until the lock becomes available.

### 9.4.7 TestAndSet

The TestAndSet instruction atomically first sets the value at the target address to True and return the old value stored at the target address.

### 9.4.8 Swap

The Swap instruction atomically swaps the content of two memory

### 9.4.9 Bounded waiting

A bound must exist on how many times a task has to wait in order to enter a critical section.

### 9.4.10 Semaphores

The hardware solutions to the critical section problem using Swap and TestAndSet are complicated for programmers to use directly.

### 9.4.11 Mutex lock

Many locking libraries provides a special mutex lock which can only be used to provide mutual exclusion to critical sections.

### 9.4.12 Deadlock

In concurrent computing, a deadlock is a state in which each member of a group is waiting for some other member to take action, such as sending a message or more commonly releasing a lock.

### 9.4.13 Mutual exclusion

A chopstick can only be held by one philosopher at the time. Updating the amount of rice must be atomic.

### 9.4.14 Deadlock prevention

Preventing deadlocks by constraining how requests for resources can be made in the system and how they are handled (system design).

### 9.4.15 Deadlock avoidance

The system dynamically considers every request and decides whether it is safe to grant it at this point,

### 9.4.16 Clock Synchronization

- External synchronization
- Internal synchronization
- Hard in an asynchronous system

### 9.4.17 Mutual exclusion

- Critical section
- Need for coordination:
- Evaluation criteria for different solutions

### 9.4.18 Desired properties of Transactions

- Atomicity
- Consistency
- Isolation
- Durability

### 9.4.19 WiFi

- Several standards, from 2Mbit/s to 100's of Mbit/s
  - Using multiple channels for higher speeds

- Range: in general 20-100m, depending on the environment
- Can be used in ad hoc or infrastructure mode
  - Ad Hoc: like a wireless Ethernet
  - Infrastructure: a coordinating base station
- Networks identified with SSID:s
  - Used to scramble signal somewhat
- Encryption with WEP (bad), WPA(better), WPA2 (best)

#### 9.4.20 Bluetooth

- Several different versions
  - Different transmission power (and hence ranges)
- Different profiles for different applications
- Frequency hopping 1600 times/s in ISM band
- Peer to peer or piconet networks
- Max 7 simultaneous connections

#### 9.4.21 ISM-band

#### 9.4.22 Properties of a medium

- Bandwidth B
  - Frequency range for signals transmitted in medium
- Signal/noise ratio S/N, SNR
  - Frequency range for signals transmitted in
  - Signal level / noise level
  - Measured at the receiver
  - Can be approximated with Maxwell's equations
- Capacity  $C=B \log_2 (1+S/N)$

#### 9.4.23 Sampling

Bandwidth B Hz is sampled at  $2^*B$  Hz

#### 9.4.24 Multithreading models

#### 9.4.25 Bounded buffer

A bounded buffer lets multiple producers and multiple consumers share a single buffer. Producers write data to

#### 9.4.26 Priority inversion

A higher priority task is “preempted” by a lower priority one.

### 9.5 M5

#### 9.5.1 Single contiguous

Single contiguous allocation is the simplest memory management technique. All the computer’s memory, usually with the exception of a small portion reserved for the operating system, is available to the single application.

#### 9.5.2 Swapping

A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.

#### 9.5.3 Memory resident

Certain programs, can be marked as being memory resident, which means that the operating system is not permitted to swap them out to a storage device; they will always remain in memory.

#### 9.5.4 Partitioned allocation

Partitioned allocation divides primary memory into multiple memory partitions, usually contiguous areas of memory.

#### 9.5.5 Logical address space

A logical address is the address at which a memory cell appears to reside from the perspective of an executing application program.

#### 9.5.6 Address binding

Address binding is the process of mapping the program’s logical (or virtual addresses) to corresponding physical memory addresses.

### **9.5.7 Memory management unit (MMU)**

The MMU is a computer hardware unit having all memory references passed through itself, primarily performing the translation of logical memory addresses to physical addresses.

### **9.5.8 Fragmentation**

Fragmentation is a phenomenon in which storage space is used inefficiently, reducing capacity and often performance.

### **9.5.9 Compaction**

Move all processes to one end of the address space producing one large hole at the other end of the address space.

### **9.5.10 Memory protection**

Memory protection between processes implemented by associating a valid bit with each entry in the per process page table.

### **9.5.11 Shared pages**

Paging makes it possible for processes to share parts of their memory space with each other.

### **9.5.12 Translation lookaside buffer**

A translation lookaside buffer (TLB) is a memory cache that is used to reduce the time taken to access a user memory location. The TLB stores the recent translations of virtual memory to physical memory.

### **9.5.13 Operating System**

Controls the hardware and coordinates its use among the various application programs for various users.

### **9.5.14 Access methods**

Data can be accessed in different patterns. These patterns can be divided into two main categories: Sequential access, Random access (aka direct access)

### **9.5.15 RAM**

The main memory aka primary memory is a form of Random Access Memory (RAM).

### **9.5.16 Volatile memory**

Volatile memory, in contrast to non-volatile memory, is computer memory that requires power to maintain the stored information; it retains its contents while powered on but when the power is interrupted, the stored data is quickly lost.

### **9.5.17 Persistent data storage**

In computer science, persistence refers to the characteristic of state that outlives the process that created it.

### **9.5.18 The file**

A file is a named collection of related information that is recorded on secondary storage.

### **9.5.19 Block data storage**

A file is made up of fixed length logical records (blocks)

### **9.5.20 File control block**

A File Control Block (FCB) is a file system structure in which the state of an open file is maintained.

### **9.5.21 Directory**

A directory is a file system cataloging structure which contains references to other computer files, and possibly other directories.

### **9.5.22 Contiguous allocation**

Each file occupies a set of contiguous blocks on the disk.

### **9.5.23 Linked allocation**

Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.

### **9.5.24 Indexed allocation**

Indexed allocation brings all pointers together into an index

### **9.5.25 FAT**

File Allocation Table (FAT) is a computer file system architecture and a family of industry-standard file systems utilizing it. The FAT file system is a legacy file system which is simple and robust.

### 9.5.26 iNode

In Unix-style file systems the file control block is called iNode. The inode is a data structure that describes a filesystem object such as a file or a directory.

### 9.5.27 Directory

A directory is a file system cataloging structure which contains references to other computer files, and possibly other directories.

## 9.6 M6

### 9.6.1 Classic historical ciphers

Caesar cipher (monoalphabetic), Vigenere cipher (polyalphabetic).

### 9.6.2 Modern cryptography

If a lot of smart people for a long time have failed to solve a specific problem, it is unlikely that a solution will appear soon.

- Encryption method is well-known
- Secret guarded by a n-bit key
  - Encryption and decryption in  $O(n)$  time
  - Key guessing in  $O(2^n)$  time
- Key management is crucial
- CIA triad represent desirable properties
  - Confidentiality
  - Integrity
  - Availability

### 9.6.3 Public key exchange

Alice

Create secret key  $S_A$   
 Compute  $T_A = g^{S_A} \bmod p$   
     Send  $T_A$  to Bob  
 Compute  $K_1 = T_B^{S_A} \bmod p$

Bob

Create secret key  $S_B$   
 Compute  $T_B = g^{S_B} \bmod p$   
     Send  $T_B$  to Alice  
 Compute  $K_2 = T_A^{S_B} \bmod p$

### 9.6.4 Certification Authority (CA)

Issues digital certificates: Digitally signed with the private key of the CA, Authorize a public key.

### 9.6.5 Chain of trust

### 9.6.6 Self-issued certificates

Some sites present self-issued certificates: A little like designing your own drivers license.

### 9.6.7 Usage of modern cryptography

- Symmetric cryptography in encrypted sessions
  - public-key cryptography not fast enough
- Asymmetric cryptography in certain situations
  - To establish a symmetric key
  - Digital signatures and verification
- Cryptographic hashing for verification of authentic data

### 9.6.8 Man-in-middle attack

the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other

### 9.6.9 Chosen-plaintext attack

presumes that the attacker can obtain the ciphertexts for arbitrary plaintexts. The goal of the attack is to gain information that reduces the security of the encryption scheme

### 9.6.10 Side-channel attack

any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself. Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited.

### 9.6.11 Replay attack

valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it, possibly as part of a spoofing attack by IP packet substitution. This is one of the lower-tier versions of a man-in-the-middle attack. Replay attacks are usually passive in nature.

### 9.6.12 Security in the Internet stack

- Application layer: DNSSEC
- Transport layer: SSL/TLS
- Network layer: IPsec
- Link layer: WEP/WPA, MAC filtering, ...

### 9.6.13 Firewall

## Chapter 10

# System Design with User Perspective

## 10.1 Documentation

- One of the most important things in life is to understand...
- I want you to understand WHY you need to document a program!
- Then the HOW becomes much easier!
- The documentation is the user interface to the program code!
- It is possible to work without it, but who want to code in assembler?

### 10.1.1 Implicit documentation

- Names can tell a story (meaningful names)
- Names can also be classifying the variable (x,y positions. i,j,k index)
- Choose the right algorithms and data structures

### 10.1.2 Explicit documentation

- Several different versions of documentation
  - Meta documentation (Each file: Description, author, protocols, file dependencies)
  - External document (Manuals/Installation instruction, list of tree structures)
  - Inline documentation (Difficult code, TODO)

## 10.2 Human factors

### 10.2.1 Color combinations

There are some color combinations with humans have difficulty processing one of which is Red text on Blue background and Blue text on red background.

### 10.2.2 Personas

Who are the real people who will use this program. With that information make sure it is well suited for

them and there needs as well as for there difficulties.

### 10.2.3 Memory

Our memory is like a net wire each connection is from one piece of information to another there for memory needs to be associated with the necessary areas. For instance when we design a web shop we should have the checkout on the right top since for most web shops have that and therefore people make that connection that this is a web shop therefore the checkout is on the top right.

### 10.2.4 What is needed to design a system?

- Requirement specification
- Specify in detail the requirements on the software
  - What should it be able manage?
    - \* Number of customers, number of items in store, etc.
  - Which functions are needed?
- Order a beverage, check stored items, etc.

## 10.3 Model-View-Controller

Initially a Object-Oriented (interface) Design model. Purpose: Making the View separated from the Model.

- View is only presentation of a Designed Interaction!
- Start by modelling the Model and Control (as a command Interface)
- Then implement the View (both control and presentation).
  - Create a good metaphor to the things that happen in

## Chapter 11

# Probability and Statistics DV

## 11.1 Statistisk mått och begreppet sannolikhet

### 11.1.1 Begrepp

- Deterministiska modeller: enkla modeller som inte tar hänsyn till fel
- Sannolikhetsteori: modelera slumpmässiga fel
- s..k: a..u slumpmässig data
- Beskrivande statistik:
  - Population: Alla bilar i uppsala
  - Stickprov: 100 utvalda bilar i uppsala
  - Enhet: En av det 100 utvalda
  - Variabler: Motorstyrka, dragkrok, automat etc
    - \* Kvalitativa: (dragkrok, automati/-manuel)
    - \* Kvantitativa: (Motorstyrka, vikt)
  - Tvärsnitts data: Befolning i svenska städer 1 jan 2018
  - Longitudinella data: Befolking i uppsala under 1 jan 1960 till 2020
- Statistisk mått:
  - Lägesmått:
    - \* Aretmetisk modelera:  $\bar{x} = \frac{\sum_{k=1}^n x_k}{n}$
    - \* median:  $(/1, /1, /2, 4, /4, /4, /7), (/1, /1, 2, 4, 4, 4)$
  - Kvartiler: Tre punkter, fyra i quarters, som delar upp tal serie
    - \* 1:a kvartilen (Nedre kvartil) mittpunkten av den nedre halvan
    - \* 3:e kvartilen (Övre kvartil) mittpunkten av den övre halvan
  - Spridningsmått:
    - \* Kvartal bred: 3:e kvartalen – 1:a kvartalen
    - \* Stickprovs standard devianse (standard deviaion):  $s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
    - \* Spridnings diagram: Positiv korelation, negativ korelation, ingen korelation, perfect korelation

- Diskret variable: räknerligt många värden
- Kontinuerlig variabel: tar oräknerligt många värden, värden i ett interval
- Dickreta visualiseras: med stolpdoagram
- Kontinuerlig variabel visualisering: med histogram
- Lådiagram:
- Sannolikhetsteori: Vi antar att ett förseks genomförs  $n$  gånger oberoende av varandra. En händelse  $A$  inträfar  $f$  gånger
  - Frekvenskvoten:  $\frac{f}{n}$
  - Experimentellt bestört närmevärde: frekvenskvoten på  $p(A)$ 
    - \* Def: (Frekvensbaserad sannoliket):  $p(A) \lim_{n \rightarrow \infty}$
    - \* Def: (Klassisk sannoliket) Anta att färsek kan utföras på  $m$  olika olika sätt varav  $g$  är gynnsama (innebär  $A$ ). Då är  $p(A) = g/m$
  - Utfallsrum:  $\Omega$  alla möjliga värden som slumpvariabeln kan ta
  - Händelser: är delmängden av utfalsrummet
  - Slumpvariabler: Stokastisk variabel

### Kotmogorovs axiom

- I För varje händelse  $A$  gäller att  $P(A) \geq 0$
- II Sannolikhet för utfallsrummet är  $1P(\Omega) = 1$
- III Om  $A$  och  $B$  är händelser och  $A \cap B = \emptyset$  (oförsemliga händelser) gäller  $P(A \cup B) = P(A) + P(B)$   
 $\Rightarrow P(A \cup B) = P(A) + P(B) - P(A \cap B)$

## 11.2 Sannolikheter och slumpvariabler

### Använtbara räkneregler

- 1  $P(A^*) = 1 - P(A)$
- 2  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- 3  $P(A|B) = \frac{P(A \cap B)}{P(B)}$

### 11.2.1 Betingade sannolikheter

Type	Dyglig	Defekt	Tot
Äldre	170	10	180
Ny	115	5	120

$A$  = "Slumpmässigt vald produkt är dyglig"

$B$  = "Slumpmässigt vald produkt är tillverkad vid äldre maskin."

$$P(A) = \frac{285}{300} = 0.95$$

$C$  = Slumpmässigt vald produkt är dyglig givet att den är tillverkad vid äldre maskiner

$$P(C) = P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{170}{180} \approx 0.94$$

### 11.2.2 Kedjer av händelse

$$\begin{aligned} P(A \cap B \cap C) &= P(C|A \cap B)P(A \cap B) \\ &= P(C|A \cap B)P(B|A)P(A) \end{aligned}$$

### Bayes sats

$$P(B|A) = \frac{P(B)P(A|B)}{P(A)}$$

### Lagen om total sannolikhet

$$P(B) = P(B|A)P(A) + P(B|A^*)P(A^*)$$

### 11.2.3 Oberonde händelser

$$P(A|B) = P(A)$$

Betingad sannolikhet:  $P(A \cap B) = P(A|B)P(B)$

om A och B är oberonder

$$\Rightarrow P(A \cap b) = P(A)P(B)$$

### Födelsedagsparadoksen

$$1 - \left( \prod_{k=1}^n \frac{365-k}{365} \right)$$

### Slumpvariablel

Är en function från utfalsrumet  $\Omega$  till någon mängd  $E$ ,  $X : \Omega \rightarrow E$

$$X \in \{0, 1, 2, 3\}$$

$P(X = 0) + P(X = 1) + P(X = 2) + P(X = 3) = 1$   
Kolmogorovs axiom

Sannolikhetsfaktor? är  $P_X(x)$   $P_X(x) = P(X = x)$

## 11.3 Fördelningar

Diskreta fördelningar	Kontinuerliga fördelningar
$\Omega = \{1, 2, 3\}$	$\Omega = [0, 1]$
$\Omega = \{0, 2, \dots\}$	$\Omega = \mathbb{R}$
Sannolikhetsfunktion $p_X(x) = P(X = x)$	Täthetsfunktion $f_X(x) = \int_a^b f_X(x)dx = P(a \leq x \leq b)$ $P(X = x) = 0$
$\sum_{x \in \Omega} p_X(x) = 1$	$\int_{\Omega} f_X(x)dx = 1$
$E[X] = \sum_{x \in \Omega} x p_X(x)$	$E[X] = \int_{\Omega} x f_X(x)dx$
$V[X] = E[X^2] - E[X]^2$	$V[X] = E[X^2] - E[X]^2$
$F_X(x) = \sum_{i \leq x} p_X(i)$ $= p(X \leq x)$	$F_X(x) = \int_{-\infty}^x f_X(x)dx = 1$ $= P(X \leq x)$ $= P(X < x)$

### 11.3.1 Binomial-fördelningar

Tillämpning: man utför något n antal gånger med sannolikheten p att det lyckas.

Om  $X$  är binomialfördelad med paramter  $n$  och  $p$ .  
Då gäller:

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}, x = 0, 1, 2, \dots, n$$

$$X \sim Bin(n, p)$$

$$E[X] = np$$

$$V[X] = np(1-p)$$

$$dbinom(x, n, p)$$

$$pbinom(x, n, p, lower.tail = FALSE)$$

### 11.3.2 Possion-fördelningar

Tillämpning: För att modelera sällsynta händelser.

Om  $X$  är possionsfördelad med paramter  $m$ .  
Då gäller

$$P(X = x) = \frac{m^x}{x!} e^{-m}, x = 0, 1, 2, \dots, n$$

$$X \sim Po(\mu)$$

$$E[X] = \mu$$

$$V[X] = \mu$$

$$dpois(x, \mu)$$

$$ppois(x, \mu, lower.tail = FALSE)$$

### 11.3.3 Likformig/rektangulär-fördelningar

Tillämpning: Lika fördelade inom ett intervall.

Om  $X$  är likformig på intervallet  $[a, b]$ . Då gäller

$$f_X(x) = \frac{1}{b-a}, a \leq x \leq b$$

$$X \sim Re(a, b)$$

$$E[X] = (a + b)/2$$

$$V[X] = (b - a)^2/12$$

$$dunif(x, a, b)$$

$$punif(x, a, b)$$

### 11.3.4 Exponential-fördelningar

Tillämpning: Livslängd/väntetid.

Om  $X$  är exponentialfördelad med paramter  $a > 0$ .

Då gäller

$$f_X(x) = \frac{1}{\lambda} e^{-x/\lambda}, x \leq lambda$$

$$X \sim Exp(\lambda)$$

$$E[X] = 1/\lambda$$

$$V[X] = 1/\lambda$$

$$dexp(x, \lambda)$$

$$pexp(x, \lambda)$$

### 11.3.5 Normalfördelning-fördelningar

Tillämpning: Allt möjligt.

Om  $X$  är normalfördelad med paramter  $\mu$  och  $\sigma^2$ .

Då gäller

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, -\infty < x < \infty$$

$$X \sim N(\mu, \sigma^2)$$

$$E[X] = \mu$$

$$V[X] = \sigma^2$$

*dnorm*( $x, \mu, \sigma$ ) täthetsfunktionen  
(behövs inte för kontinuerliga)

*pnorm*( $x, \mu, \sigma$ ) fördelningsfunktion  
(behövs inte för kontinuerliga)

### 11.3.6 Läges och spridningsmått

#### Väntevärde

$$(\text{Diskret}) E[X] = \sum_{x \in \Omega} x P_X(x)$$

$$(\text{Kontinuerlig}) E[X] = \int_{\Omega} x f_X(x) dx$$

#### varians

Om  $X$  är slumpvariabler med väntevärde  $\mu$ .

Då gäller

$$(\text{Diskret}) V[X] = \sum_{x \in \Omega} (x - \mu)^2$$

$$(\text{Kontinuerlig}) V[X] = \int_{\Omega} (x - \mu)^2 f_X(x) dx$$

#### Alternativ form

$$(\text{Diskret}) E[X^i] = \sum_{x \in \Omega} x^i P_X(x)$$

$$(\text{Kontinuerlig}) E[X^i] = \int_{\Omega} x^i f_X(x) dx$$

$$V[X] = E[X^2] - (E[X])^2$$

## 11.4 Olikheter

### 11.4.1 Markovs olikhet

Om  $X$  är ickenegatic ( $x \leq 0$ ) och  $a > 0$ . Då gäller

$$p(X \geq a) \leq \frac{E[X]}{a}$$

### 11.4.2 Thebysjovs olikhet (cheby-shev)

Om  $X$  är en slumpvariabel med  $E[X] = \mu$  och

$V[X] = \sigma^2$ . Då gäller:

$$p(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

### 11.4.3 Fördelnings functioner

$$(\text{Diskret}) p(X \leq x) = F_X(x) = \sum_{i \leq x} P(X = i)$$

$$(\text{Kontinuerlig}) p(X \leq x) = F_X(x) = \int_{-\infty}^x f_X(t) dt$$

#### Normalfördelning

Vi kan inte integrera täthetsfunctionen  
-ingen stängd form för fördelings funktionen  
special fall:  $X \sim N(0, 1)$  (standard normalfördelning)  
Vi betecknar fördelnings funktionen  $\Phi(x) = P(X \leq x)$

#### kvartier

För  $0 < x < 1$  defineras  $\alpha$ -kvantilen  $x_\alpha$  till  
slupvariabel  $x$  som en lösning till  
 $F_X(x_\alpha) = 1 - \alpha$

### 11.4.4 Oberonede slupvariabel

Två slumpvariabler  $x_1 \wedge x_2$  kallas oberonade om  
 $P(x_1 \in A \cap x_2 \in B) = P(x_1 \in A)P(x_2 \in B)$   
 för alla mängder  $A \subseteq \Omega_1, B \subseteq \Omega_2$

$$\begin{aligned} E[Y] &= a + bE[x] \\ V[Y] &= b^2V[x] \end{aligned}$$

### räkneregler

$$\begin{aligned} Y &= a_1x_1 + a_2x_2 + \dots + a_nx_n \text{ slupvariabler} \\ x_i &\text{ konstanter } a_i \\ E[Y] &= a_1E[x_1] + a_2E[x_2] + \dots + a_nE[x_n] \end{aligned}$$

### Exempel (505): räkneregler

Låt  $X \sim N(10, 4), Y \sim N(3, 1)$  vara oberonade  
 slumpvariabler.

Beräkna sannolikheten att  $X > 3Y$

$$\begin{aligned} \text{Låt } Z &= X - 3Y \Rightarrow E[Z] = E[X] - 3E[Y] \\ &= 10 - 3 * 3 = 1 \\ V[Z] &= E[X] + (-3)^2E[Y] = 10 = (-3)^2 * 3 = 13 \\ \text{Enligt normalfördelingen så får vi } Z &\sim N(1, 13) \\ \text{Därmed kan vi beräkna följande sannolikhet} \\ P(X - 3Y > 0) &= P(Z > 0) = 1 - P(z \leq 0) \\ 1 - \Phi(-1/\sqrt{13}) &= 1 - \Phi(1 - \Phi(1/\sqrt{13})) = \Phi(0.28) \\ &= 0.61 \end{aligned}$$

### väntevärde av produkter

Om  $x_1, \dots, X_n$  är oberonade då gäller  
 $E[x_1, \dots, X_n] = E[x_1] \dots E[x_n]$

### Special fall

$$\begin{aligned} \bar{x} &= \frac{1}{n}(x_1, \dots, X_n) \\ &\text{där } x \text{ är oberonade och diktordelade med} \\ E[x_i] &= \mu, V[x_i] = \sigma^2 \\ E[x] &= \mu, V[x] = \frac{\sigma^2}{n} \end{aligned}$$

### 11.4.5 Fördelning av summor

Binomialfördelning, Poissonfördelning, Linjärkombination av normalfördelade variabler.

### 11.4.6 Central gränsvärdessatsen (CGS)

Exakt fördelning för summor är svårt i allmänhet  
 därmed använder man CGS

$$\begin{aligned} Y &= X_1 + X_2 + \dots + X_n \\ Y &\sim N(\mu_Y, \sigma_Y^2) \end{aligned}$$

### Example

$$\begin{aligned} X_i &\sim Bin(1, 0.2) \\ Y &= \sum_{i=1}^{30} X_i \\ P(Y \leq 8) &=? \end{aligned}$$

$$\begin{aligned} \text{CGS: } Y &\sim N(\mu_Y, \sigma_Y^2) \\ \mu_Y &= E[Y] = 30 * E[X_i] = 30 * 0.2 = 6 \\ \sigma_Y^2 &= V[Y] = 30 * V[X_i] = 30(1 * 0.2 * 0.8) = 4.8 \\ P(Y \leq 8) &= pnorm(8, 6, sqrt(4.8)) \approx 0.82 \\ P\left(\sum_{i=1}^{30} X_i \leq 8\right) &\approx 0.87 \end{aligned}$$

### 11.5 Simulerings av slump-tal

#### 11.5.1 äkta slumpprässiga tal

Hårdvaru genererade tal:

- Tärningar, roletthjul ..
- Radiaktivit sönderfall
- Atmosfäriskt brus (random.org)

### 11.5.2 Pseudoslumpmässiga tal

Dator genererade slump tal. Kommer att uppredasig.

- Tar in en seed som input för att generera slumptal
- Default seed är oftast tid
- det är deterministiska
- Har en period med nya tal sedan så upprepar det sig

#### Von Neumann

1. Väljer seed:  $u_0 = 0.1111$
2. Skapar  $y_0 = 1111$
3. Beräknar  $y_0^2 = 1234321$
4. Fyller på från venster med noll för att få 8 siffror 01234321
5. Skapar genom att ta det fyra mittersta siffrorna  $y_1 = 2343$
6.  $0.2343 \Rightarrow y_1^2 = 5489649 \Rightarrow y_2 = 4896 \Rightarrow u_2 = 0.4896$

#### Kongruens

$$V_{n+1} = aV_n + b \pmod{c}$$

Vi är ett heltal mellan 0 och  $c - 1$  och  $u_1 = \frac{V_i}{c}$   
a,b,c måste väljas noggrant

(talteori, c måste vara ett primtal)  
vanliga val är  $a = 7^7 = 16807$ ,  $b = 0$ ,  
 $c = 2^{31} - 1 = 2147483647$   
hat svagheter, används inte längre

#### XOR-generator

Extrem snabb, lätt att förstå.

1. Väljer seed:  $m$  binära bits (heltal mellan 0 och  $2^m - 1$ ).
2. Skifta alla bits  $l$  steg åt vänster och fyll i från höger med nollar.
3. XOR med seed och det skiftade talet (seed update).
4. Skifta seed update  $m - l = r$  steg åt höger och fyll i nollar från vänster.
5. XOR med seed update och högerskiftet.
6. Konvertera till ett tal mellan 0 och 1.

#### Mersenne Twister

##### Exempel (505): räkneregler

Givet 5 pseudoslumpmässiga tal från  $Re[0, 1]$

$$u_1 = 0.8147, u_2 = 0.9058, u_3 = 0.1270, u_4 = 0.9134, u_5 = 0.634$$

Simulera 5 doser? från slumpvariablen  $X$  med

$$f_X(x) = \frac{x}{2}, 0 \leq x \leq 2$$

Beräna  $E[x]$  och jämför med medvärder av de simulerade dosetaner

$$\begin{aligned} \text{Tar fram primitiva funktionen } F_X(x) &= \int_0^x \frac{t}{2} dt \\ &= [t^2/4]_0^x = \frac{x^2}{4} \\ (\text{Tar fram inversen}) y &= \frac{x^2}{4} \Rightarrow x = 2\sqrt{y} \end{aligned}$$

$$x_1 = F_x^{-1}(u_1) = 2\sqrt{u_1} = 1.8052$$

$$x_2 = F_x^{-1}(u_2) = 2\sqrt{u_2} = 1.9035$$

$$x_3 = F_x^{-1}(u_3) = 2\sqrt{u_3} = 0.7127$$

$$x_4 = F_x^{-1}(u_4) = 2\sqrt{u_4} = 1.9114$$

$$x_5 = F_x^{-1}(u_5) = 2\sqrt{u_5} = 1.5905$$

$$\begin{aligned} \text{Beräknar väntevärde } E[x] &= \int_0^2 x \frac{x}{2} dx = [\frac{x^3}{6}]_0^2 \\ &= \frac{8}{6} = \frac{4}{3} \approx 1.33 \end{aligned}$$

## 11.6 Statistikens grunder

### 11.6.1 Allmänt

Skattning av en okänd parameter  $\theta$  från en familj  $F_X(\theta)$  är en funktion  
 $\hat{\theta} = t = g(x_1, x_2, \dots, x_n)$

### 11.6.2 Medelfel

$\sigma^2 \wedge p$  kan vara okända

Vi kan definiera medelfellet genom att använda skattningen.

$$s = \hat{\sigma} \wedge \hat{p}$$

$$\begin{aligned} V[\hat{\mu}] &= \frac{\sigma^2}{n} \\ V[\hat{p}] &= \frac{\hat{p}(1 - \hat{p})}{n} \\ d[T_\mu] &= \frac{s}{\sqrt{n}} \\ d[T_p] &= \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}} \end{aligned}$$

Utifrån medelfelet definierar vi konsistens och effektivitet.

**Konsistans**  $V[T] \rightarrow 0$  då  $n \rightarrow \infty$

**Effektivitet** Givet estimatoren  $T_1, T_2$  så är  $T_1$  effekten att  $T_2$  om  $V[T_1] < V[T_2]$  ( $D[T_1] < D[T_2]$ )

### Exempel

Vi beräknar variansen på  $T_\mu$  och  $T_p$

$$\begin{aligned} V[T_\mu] &= V[1/n(x_1, x_2, \dots, x_n)] \\ V[T_\mu] &= 1/n^2(V[x_1] = V[x_2] + \dots + V[x_n]) \\ V[T_\mu] &= \frac{1}{n^2}n\sigma^2 = \frac{\sigma^2}{n} \\ V[T_p] &= V[x/n] \\ V[T_p] &= \frac{1}{n^2}V[x] \\ V[T_p] &= \frac{1}{n^2}np(1 - p) \\ V[T_p] &= \frac{p(1 - p)}{n} \end{aligned}$$

### 11.6.3 Skattning av variansen

$N(\mu, \sigma^2)$  -Vill skatta  $\sigma^2$  det går att visa att

$$S^2 = \hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

är en venteriktig skattning av variansen

Vi kan visa att

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2 + (n_3 - 1)S_3^2}{(n_1 - 1) + (n_2 - 1) + (n_3 - 1)}$$

är en väntevärdsriktig skattning av  $\sigma^2$

### Def

Låt  $A$  och  $B$  vara funktioner av  $x_1, x_2, \dots, x_n$  så att Då kallas  $[A, B]$  ett  $100(1 - \alpha)$ -procent konfidensintervall för  $\theta$  (med konfidensgrad  $1 - \alpha$ )

### 11.6.4 Väntevärdsriktig

#### Exempel

$X \sim N(m_1 - m_2, 4)$ ,  $Y \sim N(m_1 + m_2, 5)$

a. Visa att  $\hat{m}_1 = (X + Y)/2$  är väntevärdsriktig skattning av  $m_1$

$$\begin{aligned} E(\hat{m}_1) &= E[(X + Y)/2] = 1/2(E[X] + E[Y]) \\ &= 1/2(m_1 - m_2 + m_1 + m_2) = \frac{1}{2}2m_1 = m_1 \end{aligned}$$

därmed så är den väntevärds riktig

b. Beräkna standardavvikelsen för  $\hat{m}_1$

$$\begin{aligned} D(\hat{m}_1) &= \sqrt{V(\hat{m}_1)} = \sqrt{V\left(\frac{x+y}{2}\right)} = \sqrt{\frac{1}{4}(V[X] + V[Y])} \\ &= \frac{\sqrt{9}}{2} = \frac{3}{2} \end{aligned}$$

### 11.6.5 Konfidensintervall för $\mu$ från $N(\mu, \sigma^2)$ med känt $\sigma$

$$\text{Estimatorn } \hat{\mu} = \bar{x} = \frac{1}{n}(X_1 + X_2 + \dots + X_n)$$

$$\bar{X} \sim N(\mu, \sigma^2/n)$$

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \sim N(0, 1)$$

$$A = \bar{X} - \lambda_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

$$B = \bar{X} + \lambda_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

### 11.6.6 Example

Vid tillverkningsprocessen av axlar av rundstål kontrollers diametern.

Följande diametrar (mm) uppmättes:

30.02, 30.12, 30.07, 29.95, 30.05, 29.90, 30.01

Konstruera ett konfidensintervall, med konfidensgrad 0.95, för den förväntade diametern.

$$n = 7, \alpha = 0.05$$

$$\begin{aligned} \bar{X} &= \frac{30.02 + 30.12 + 30.07 + 29.95}{n} \\ &\cdot \frac{30.05 + 29.90 + 30.01}{n} = \frac{210.12}{7} \\ s &= \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \\ &= \frac{1}{6} ((30.02 - \bar{x})^2 + (30.02 - \bar{x})^2) \end{aligned}$$

### 11.6.7 Konfidensintervall för $p$ från binomialfördelad

Finns många alternativa metoder

$$Bin(n, p) \sim N(np, np(1-p)) \text{ om } np(1-p) \geq 10$$

$$\Rightarrow I_p = [\hat{p} \pm \lambda \sqrt{\hat{p}(1-\hat{p})/n}]$$

konfidensintervall för  $\theta$  (med konfidensgrad  $1 - \alpha$ )

### 11.6.8 Konfidensintervall för skillnad i väntevärde

Vill ofta jämföra två grupper

$$x_1, \dots, x_n \text{ från } N(\mu_1, \sigma_1^2)$$

$$y_1, \dots, y_n \text{ från } N(\mu_2, \sigma_2^2)$$

Söker  $\mu_1 - \mu_2$ . Om detta intervall inhåller 0 då kan vi med konfidensgrad  $1 - \alpha$  säga att det är skillnad på väntevärdena

### Okända varianser

$$s_p^2 = \frac{(n-1)s_1^2 + (m-1)s_2^2}{(n-1) + (m-1)}$$

$$\frac{\bar{X} - \bar{Y} - (\mu_1 - \mu_2)}{s_p \sqrt{1/n + 1/m}}$$

$$I_{\mu_1 - \mu_2} = [\bar{X} - \bar{Y} \pm t_{\alpha/2}(n+m-2)s_p \sqrt{1/n + 1/m}]$$

### 11.6.9 Ensidiga intervall

Låt  $A$  och  $B$  vara funktioner av  $x_1, \dots, x_n$

För ett nedåt begänsat konfidensintervall gäller

$$p(\theta \geq A) = 1 - \alpha$$

För ett uppåt begrensat gäller

$$p(\theta \leq B) = 1 - \alpha$$

### 11.6.10 Stickprov i par

Fotgängare	A	B	
1	43	32	
2	81	90	
3	11	7	
4	49	31	
5	22	26	Tiden det tar för A re-
6	143	168	
7	24	31	
8	56	39	
9	31	29	
10	53	57	

spektive B att upptäcka Pfordgängaren baserat står i tabellen. Konstruera ett lämpligt konfidensintervall baserat på datan. Vilken algoritm borde and-

vändas.

Vi väljet ett konfidensintervall på 95%,  $Z = A - B$

$$\bar{Z} = 3/10 = 0.3, s_Z^2 = \frac{1}{10-1} \sum_{i=1}^{10} (Z_i - \bar{Z})^2 = 13.08,$$

$$t_{0.025}(t) = 2.26$$

$$I_\mu = [\bar{Z} \pm t_{0.025}(t) \frac{s}{\sqrt{n}}]$$

$$= [-9.05, 9.65]$$

0 finns i intervallet, med konfidensgrad 0.95 kan ingen skillnad påvissas mellan algoritmerna samla in mer data, allternativt använd vilken algoritm som helst.

## 11.7 Regression

### 11.7.1 Modell

Givet observationsparen  $x_1, \dots, x_n$  och  $y_1, \dots, y_n$  ansätter man följande modell.

$$Y_i = m + kx_i + \epsilon_i \text{ Där } \epsilon_i \sim N(0, \sigma^2)$$

### 11.7.2 Modellens giltighet

Konrelationskoeficent:  $r = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$

Förklaringsgrad

$$R^2 = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$$

### Residualer

$$e_i = y_i - (\hat{m} + \hat{k}x_i)$$

Residualer bör uppfylla visa krav

- (1) konstat varians, oberonde av x
- (2) Residualerna bör vara oberonde av varandra
- (3) Residualerna bör vara normalfördelade

Vi bedömer dessa visuelt

- (i) Plottar residualerna i ett histogram, där vi kan se om det kan vara normal fördelat
- (ii) Plottar residualerna i q-q plot  
På x-axeln kvatiler från en s..fördelning  
På y-axeln kvatiler från residualerna
- (iii) Ritar ett spridnings diagram över x-värdena mot residualerna ej ett mönster  $\Rightarrow$  från X
- (iv) Ritar ett spridnings diagram över residualerna mot det förutspodda y-värdet  
vill se ett jämt utpridning utan mönster

### 11.7.3 Användning av modellen

Konfidensintervallet för parametern k

$$V[\hat{m}] = \frac{\sigma^2}{n} \frac{1}{S_{xx}} \sum_{i=1}^n x_i^2$$

$$V[\hat{k}] = \frac{\sigma^2}{S_{xx}}$$

## 11.8 Stokastiska processer

### 11.8.1 Bornulli processer

Kommunikationskanal, överföring av slumpmässiga data. Tiden är uppdelad i luckor (slots)  $k = 1, 2, 3, \dots, n$  varje lucka kan hantera ett paket

### 11.8.2 Poisson processer

#### Proposition:

För en poissonprocess  $N(t), t \geq 0$  gäller att  
 (a) inkreten

$$N(t_1), N(t_2) - N(t_1), \dots, N(t_k) - N(t_{k-1})$$

är oberonnde slumpvariabel för alla

$$0 \leq t_1 \leq \dots \leq t_k \text{ och}$$

$$(b) N(t) - N(s) \sim Po(\lambda(t-s))$$

#### Ex:

Man att aantal fel på en kommunikationskabel är 1.7. Total aantal fel beskrivs av en poisson process med parameter  $\lambda = 1.7$  Vad är sannolikheten att det finns mer än två fel på 0.5 kilometer?

$$\begin{aligned} N(0.5) &\sim Po(\lambda(0.5 - 0)) = Po(0.85) \\ \Rightarrow P(N(0.5) > 2) &= 1 - P(N(0.5) \leq 2) \\ &= 1 - ppois(2, 0.85) \approx 0.0549 \end{aligned}$$

#### Förtunning

Låt  $N(t)$  vara en poisson process med parameter  $\lambda$ , Låt  $J_n, n \in \mathbb{N}$  vara en följ av i.i.d.  $Be(p)$

$$P(J_i = 1) = p, P(J_i = 0) = 1 - p$$

$$M(t) = \sum_{k=1}^{\infty} \mathbb{1}_{T_k \leq t} J_k$$

Vi kan tolka att vi skippar vissa händelser

#### Superposition

Låt  $N(t)$  och  $N_2(t)$  vara en poisson process med parameter  $\lambda_1$  och  $\lambda_2$

Vi vill visa att  $M(t) = N_1(t) + N_2(t)$  är en poisson process med parameter  $\lambda_1 + \lambda_2$

#### Ex:

Antal inkomade samtal till en mobil kan beskrivas som en poissonprocess med paramter 0.5 per time och antal sms som en poissonprocess medparamter 2 två fel på 0.5 kilometer?

(a) sannoliket att ingen kommunikationskabel

inkomer på en time  $N_1(t)$  med parameter

$$\lambda_1 = 2, \text{ -antal sms}$$

$N_2(t)$  med parameter  $\lambda_2 = 0.5, \text{ -antal samtal}$

$$M(t) = N_1(t) + N_2(t) \text{ med parameter } 2.5,$$

-antal kommunicationer

$$= 1 - ppois(2, 0.85) \approx 0.0549$$

#### Spatial process

En samling punkter i en region  $S \subseteq \mathbb{R}^2$

Om  $N(A)$  räknar aantal punkter i en mängd  $A$

(där  $A$  är mätbar)

då gäller att

$$*N(A) \sim Po(\lambda|A|)$$

\* Om  $A \cap B = \emptyset$  är  $N(A)$  och  $N(B)$  oberonnde

#### M/M/ $\infty$ -modellen

$N(t)$  -Poissonprocess, ankomster upp till tid t ankomsterna tar tid att behandla

hur många ankomster pågår vid en viss tid?

$X_t$  -antal pågående ankomster vid tid t

Vi antar att varje ankomst behandlas på  $\exp(\lambda)/\text{tid}$

## 11.9 Markovkedjor

En Stokastisk process  $X_n$  i diskret tid med diskret tillståndsrum  $E$  kallas en Markovkedja om den har Markovegenskapen

$$\begin{aligned} P(X_n = x_n | X = x_0, X_1 = x_1, \dots, X_{n-1} = x_{n-1}) \\ = P(X_n = x_n | X_{n-1} = x_{n-1}), \forall x_1 \in E \wedge n \geq 1 \end{aligned}$$

och övergångssanolikheten  $p_{xy}$

$$= P(X_n = y | X_{n-1} = x)$$

är oberonde från  $n$

Vi fokuserar på  $E = 0, 1, \dots, r$  och  $E = \mathbb{Z}_{\geq 0}$

Övergångsmatris:

$$\begin{pmatrix} p_{00} & p_{01} & \dots & p_{0r} \\ p_{10} & p_{11} & \dots & p_{1r} \\ \vdots & \vdots & \dots & \vdots \\ p_{r0} & p_{r1} & \dots & p_{rr} \end{pmatrix}$$

### 11.9.1 Ehrenfestmodellen

Oavsett startpunkt tenderar kedjan mot ett ekvilibrium. Övergångsmatris:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1/r & 0 & 1 - 1/2 & 0 & \dots & 0 & 0 \\ 0 & 2/r & 0 & 1 - 2/r & \dots & 0 & 0 \\ \vdots & & & & & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 1 & 0 \end{pmatrix}$$

#### Def: stationär

En fördelning  $\pi$  kallas stationär för en Markovkedja med övergångsmatris  $P$  om den löser ekvationssystemet

$$\pi = \pi P$$

- $\pi$  är en egenvektor för  $P$  med egenvärde 1
- Om  $\pi$  är ursprungsfördelningen  $p^0 = \pi$   
 $\Rightarrow p^n = \pi, \forall n \geq 0$

#### Def: asymptotisk

En fördelning  $\pi$  är en asymptotisk fördelning för Markovkedjan  $X_n$  om  $\lim_{n \rightarrow \infty} P(X_n = k) = \pi_k, \forall k \geq 0$  oberonde avursprungsfördelningen  $p^0$

Asymptotiska fördelningar är alltid stationära,

#### Def: Irreducibel, Aperiodisk

En markovkedja  $X_n$  kallas

- Irreducibel om  $P(X_n = j | X_0 = i) > 0$  för något  $n$  och alla  $i, j \in E$
- Aperiodisk om största gemensamma delaren av mängden  $n : P(X_n = i | X_0 = 1) > 0$   
är 1 för alla  $i$

Om en kedja är irreducibel och något  $p_{ii} > 0$  är kedjan aperiodisk.

#### Def: Tillståndsrummet

Låt  $X_n$  vara aperiodisk och irreducibel

- (1) Om tillsåndsrummet är ändligt finns en unik stationär fördelning som också är asymptotisk
- (2) Om tillsåndsrummet är oändligt, då om en stationär fördelning existerar är den unik och asymptotisk

#### Exempel: Markovkedjor

$$P = \begin{pmatrix} 3/4 & 0 & 1/4 \\ * & 1/3 & 0 \\ 1/4 & 1/2 & ** \end{pmatrix}$$

- (a) Ange \* och \*\*
- (b) Rita övergångsgrafen
- (c) Argumentera för att kedjan är aperiodisk och irreducibel
- (d) Bestäm den stationära fördelningen

$$(a) * = 1 - (1/3 + 0) = 2/3, ** = 1 - (1/4 + 1/2) = 1/4$$

(b) Rita övergångsgrafen

- (c) Som man ser på övergångsgrafen att den man kan ta sig till alla positioner därmed så är den irreducibel vilket också medföljer aperiodisk

$$(d) (\pi_0 \pi_1 \pi_2) = (\pi_0 \pi_1 \pi_2) \cdot p \begin{pmatrix} 3/4 & 0 & 1/4 \\ 2/3 & 1/3 & 0 \\ 1/4 & 1/2 & 1/4 \end{pmatrix}$$

$$\begin{cases} \pi_0 = 3/4\pi_0 + 2/3\pi_1 + 1/4\pi_2 \\ \pi_1 = 0\pi_0 + 1/3\pi_1 + 1/2\pi_2 \\ \pi_2 = 1/4\pi_0 + 0\pi_1 + 1/4\pi_2 \end{cases}$$

$$\pi_2 = \pi_0/3, \pi_1 = \pi_0/4$$

dock  $\pi_0 + \pi_1 + \pi_2 = 1$  Normalisering:  $\pi = (12/19, 3/19, 4/19)$

### 11.9.2 Google-kedjan

Google skapar en graph av alla sidor  
Tillståndet efter  $n$  steg beskrivs av en  
Markovkedja med denna övergångsmatrisen  
Google letar efter den asymptotiska fördelningen  
på kedjan och rankar sidorna i sökresultatet  
enligt sannolikheterna i den asymptotiska  
fördelningen

### 11.9.3 Hashfunktioner

En funktion  $h$  som tar  $n$  visare och sparar som  
någon av  $m$  möjliga hasvärden ( $m < n$ )

### 11.9.4 Kollisionmodoll

Modeleras med Bernoulli-processen

### 11.9.5 Markov Buffer/Markovkö

$X_n$  -antal packet som anländer under slot  $n$   
 $Q_n$  -antal packet i kö slutet av slot  $n$



## Chapter 12

# Digital Technology and Electronics

### Resources

- <https://www.falstad.com/circuit/circuitjs.html>
- <http://www.32x8.com/var4kmapx.html>
- <https://www.tinkercad.com/dashboard?type=circuits&collection=designs>
- <https://www.symbolab.com/solver/system-of-equations-calculator>

## 12.1 Circuit Theory

### 12.1.1 Basic electrical quantities

#### Current

$i(A)$ : amps

Positive repels Positive. Negative repels Negative. Positive and Negative force of attraction. Copper is commonly used in wires since there is only one electron in last orbital. This makes it easy to move therefore it is highly conductive.

$I = q^-/\text{sec}$  Current is charge per second.

Current is written from Positive to Negative. Since back in history Ben Franklin who studied electricity and came up with theories. However they were unaware that electrons existed and therefore made the false assumption that the current goes from positive to negative. With in fact the electron is attracted to the positive end and therefore flows that way. Ben Franklin 1747. JJ Thompson 1897  $e^-$ . However positive charge does exist like in your body then it would be true.

Electron current is from negative to positive. Conventional current (the current we know as) is from positive to negative.

#### Voltage

$v(V)$ : volt

Voltage is similar to gravity. It is the potential difference in the two poles, in other words like potential energy like in the gravity analogy.

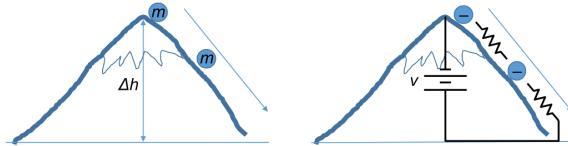


Figure 12.1: volt. From ?

$$V = \frac{\Delta U}{q} \quad \text{Where } \Delta U \text{ is the potential energy difference and } q \text{ is charge particle}$$

### Power

Power is defined as the rate energy ( $U$ ) is transformed or transferred over time. We measure power in units of joules/second, also known as watts. 1 watt = 1 joules / second)

An electric circuit is capable of transferring power. Current is the rate of flow of charge, and voltage measures the energy transferred per unit of charge. We can insert these definitions into the equation for power:  $p = \frac{dU}{dt} = \frac{dU}{dq} * \frac{dq}{dt}$ . ( $dU = \Delta U$ )

$$p = v * i \quad (12.1)$$

### 12.1.2 Prefixes

Number	Prefix	Symbol	Note
$10^{+10}$	tera-	$T$	
$10^{+9}$	giga-	$G$	
$10^{+6}$	mega-	$M$	
$10^{+3}$	kilo-	$k$	the only $> 1$ prefix in lower case
$10^0$			
$10^{-3}$	milli-	$m$	
$10^{-6}$	micro-	$\mu$	be careful $\mu$ mu doesn't turn into "m"
$10^{-9}$	nano-	$n$	
$10^{-12}$	pico-	$p$	

### 12.1.3 Unit grammar

Symbol name	example names	Symbol	example symbols	Named after
second	1 millisecond	$s$	$2ns$	
meter	300 kilometer	$m$	$35nm35$	
hertz	10 kilohertz	$Hz$	$100MHz$	Hertz
ohm	2 megohm	$\Omega$	$47k\Omega$	Ohm
farad	10 picofarad	$F$	$220pF220$	Faraday
ampere	35 microamp	$A$	$65mA$	Ampère
volt	11 kilovolt	$V$	$5\mu V$	Volta

### 12.1.4 Ohm's law

- V (Voltage): V (volt)
- I (Current): A (amps)
- R (Resistance):  $\Omega$  (omega)

$$V = IR \quad (12.2)$$

### 12.1.5 Circuit elements

### 12.1.6 Circuit terminology

- *Closed circuit* – A circuit is closed if the circle is complete if all currents have a path back to where they came from.
- *Open circuit* – A circuit is open if the circle is not complete if there is a gap or opening in the path.
- *Short circuit* – A short happens when a path of low resistance is connected (usually by mistake) to a component. The resistor shown below is the intended path for current, and the curved wire going

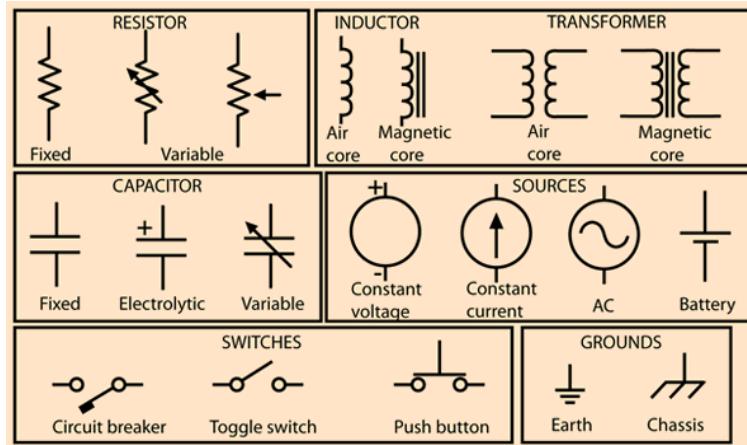


Figure 12.2: From Circuit elements

around it is the short. Current is diverted away from its intended path, sometimes with damaging results. The wire shorts out the resistor by providing a low-resistance path for current (probably not what the designer intended).

- *Node* - Between elements
- *Branch* - path between two nodes.
- *Loop* - Close path in a circuit (Branch - Nodes + 1)

### 12.1.7 Series Resistor

$$R_{series} = R_1 + R_2 + \dots + R_n \quad (12.3)$$

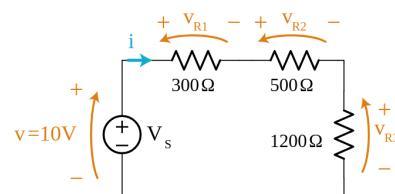


Figure 12.3: voltage distributes between resistors in series. From

$$R_{series} = 300\Omega + 500\Omega + 1200\Omega = 2000\Omega$$

$$i = \frac{v}{R_{series}} = \frac{10V}{2000\Omega} = 5mA$$

$$v_{R1} = i \cdot R1 = 5mA \cdot 300\Omega = 1.5V$$

$$v_{R2} = i \cdot R2 = 5mA \cdot 500\Omega = 2.5V$$

$$v_{R3} = i \cdot R3 = 5mA \cdot 1200\Omega = 6.0V$$

$10V - 1.5V - 2.5V - 6.0V = 0V$  Therefore correct

### 12.1.8 Parralel resistor

$$\frac{1}{R_p} = \frac{1}{R_1} + \frac{1}{R_2} \quad (12.4)$$

$$R_{parallel} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}} \quad (12.5)$$

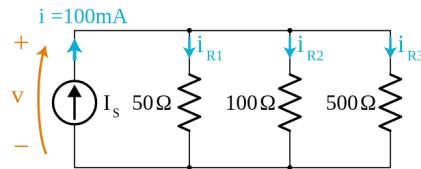


Figure 12.4: current distributes between resistors in parallel. From

$$R_{parallel} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

$$R_{parallel} = \frac{1}{\frac{1}{0.02} + \frac{1}{0.01} + \frac{1}{0.002}} = \frac{1}{0.032} = 31.25\Omega$$

$$v = i \cdot R_{parallel} = 100mA \cdot 31.25\Omega = 3.125V$$

$$i_{R1} = \frac{v}{R1} = \frac{3.125V}{50\Omega} = 62.5mA$$

$$i_{R2} = \frac{v}{R2} = \frac{3.125V}{100\Omega} = 31.25mA$$

$$i_{R3} = \frac{v}{R3} = \frac{3.125V}{500\Omega} = 6.25mA$$

$100mA - 62.5mA - 31.25mA - 6.25mA = 0mA$  Therefore correct

### 12.1.9 Simplify resistor network

Start from the furthest point. Make parallel and serial resistors become one.

### 12.1.10 Voltage divider

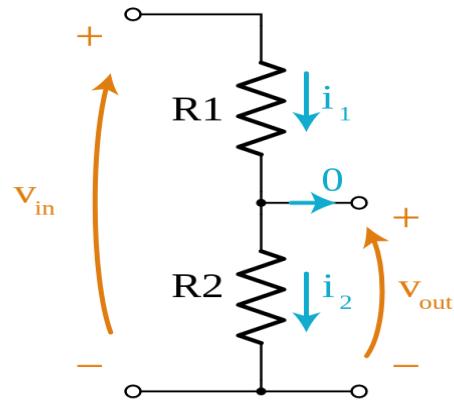


Figure 12.5: voltage-divider. From

$$\begin{aligned} v_{in} &= i(R_1 + R_2) \Rightarrow I = \frac{1}{R_1 + R_2} v_{in} \\ v_{out} &= iR_2 \Rightarrow v_{out} = \frac{R_2}{R_1 + R_2} v_{in} \\ &= \frac{1}{\frac{R_1}{R_2} + 1} v_{in} \end{aligned}$$

### 12.1.11 Kirchhoff's laws

#### Kirchhoff's current law

KCL

$$\sum i_{in} = \sum i_{out} \quad (12.6)$$

#### Kirchhoff's voltage law

KVL

$$\sum_n V_n = 0 \quad (12.7)$$

$$\sum V_{rise} = V_{drop} \quad (12.8)$$

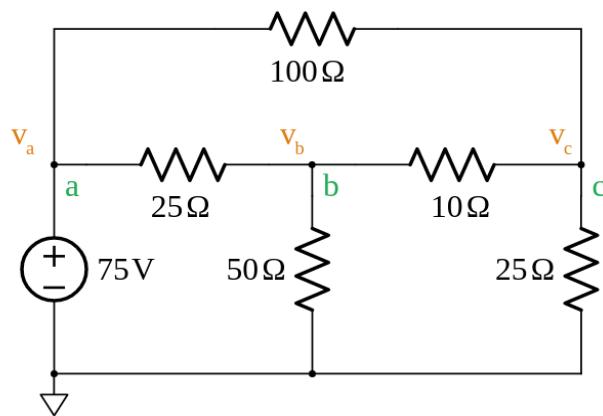
**12.1.12 Node voltage method**

Figure 12.6: Example node voltage method. From

**1. Assign a reference node (ground).**

Assign it under Voltage source

**2. Assign node voltage names to the remaining nodes.**

See image for  $v_a$ ,  $v_B$ ,  $v_c$ ,  $a$ ,  $b$ ,  $c$

**3. Solve the easy nodes first, the ones with a voltage source connected to the reference node.**

$V_a$  is the easiest since it is the same as the input  $v_a = 75V$

**4. Write Kirchhoff's Current Law for each node. Do Ohm's Law in your head.**

$$\text{Node } b: +\frac{v_a - v_b}{25} - \frac{v_b}{50} + \frac{v_c - v_b}{10} = 0$$

$$\text{Node } c: +\frac{v_a - v_c}{100} - \frac{v_b - v_c}{10} + \frac{v_c}{25} = 0$$

**5. Solve the resulting system of equations for all node voltages.**

$$\text{Node } b: -\frac{v_b}{25} - \frac{v_b}{50} - \frac{v_b}{10} + \frac{v_c}{10} = -\frac{v_a}{25}$$

$$\Rightarrow -\frac{8}{50}v_b + \frac{1}{10}v_c = -\frac{75}{25} = -3$$

$$\text{Node } c: +\frac{v_b}{10} - \frac{v_c}{100} - \frac{v_c}{10} + \frac{v_c}{25} = -\frac{v_a}{100}$$

$$\Rightarrow +\frac{1}{10}v_b - \frac{15}{100}v_c = -\frac{75}{100} = -\frac{3}{4}$$

Solve by gauseelimination and then get  $v_b = +37.5V$ ,  $v_c = +30V$

**6. Solve for any currents you want to know using Ohm's Law.**

$$i_{ab25\Omega} = \frac{75 - 37.5}{25} = 1.5A \text{ arrow pointing right}$$

$$i_{bg50\Omega} = \frac{37.5}{50} = 0.75A \text{ arrow pointing down}$$

$$i_{bc10\Omega} = \frac{37.5 - 30}{10} = 0.75A \text{ arrow pointing right}$$

$$i_{ac100\Omega} = \frac{75 - 30}{100} = 0.45A \text{ arrow pointing right}$$

$$i_{cg25\Omega} = \frac{30}{25} = 1.2A \text{ arrow pointing down}$$

### 12.1.13 Mesh current method

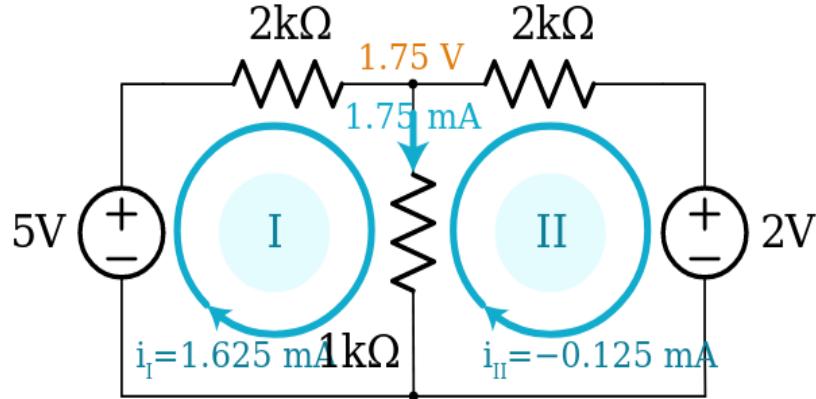


Figure 12.7: Mesh current method. From

Mesh I:

- + The voltage source – voltage drop over first resistor – voltage over second resistor = 0 (KVL)
- +  $5V - 2000i_I - 1000(i_I - i_{II}) = 0$

Mesh II:

- + Voltage over first resistor – Voltage over second resistor – Voltage source
- +  $1000(i_I - i_{II}) - 2000i_{II} - 2V = 0$

Gauseelimination gives us  $i_{II} = -0.125mA$ ,  $i_I = +1.625mA$

### 12.1.14 Capacitor i-v equation

$$i = C \frac{dv}{dt}, v = \frac{1}{C} \int_0^T idt + v_0 \quad (12.9)$$

- C is the capacitance, a physical property of the capacitor (ex:  $10\mu F$ )
- $\frac{dv}{dt}$  reate of change for volt over time (ex:  $100volts/second$ )

### 12.1.15 inductor i-v equation

$$v = L \frac{di}{dt}, \frac{1}{L} \int_0^T v dt + i_0 \quad (12.10)$$

- L is the inductance, a physical property of the inductor (ex:  $1mH$ )
- $\frac{di}{dt}$  is the reate of change for the current over time (ex:  $300ampere/second$ )

### 12.1.16 RC circuit

#### RC natural response

When initializes the circuit with a voltage and the capacitor will then be charged. Then we disconnect the voltage source, the capacitor will act as a battery and edvenshula discharge.

$$v(t) = v_0 e^{-t/RC} \quad (12.11)$$

- Time constant:  $\tau = RC$ . It is the time it takes to charge the capacitor, through the resistor.
- $v_0$  is the voltage at  $t = 0$

**Example:**

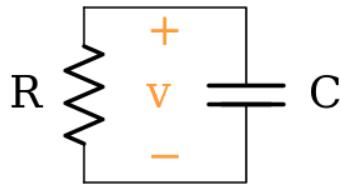


Figure 12.8: EX RC natural response. From

- a. Write the expression for  $v(t)$

$$v(t) = V_0 e^{-t/RC} = 1.4 e^{-t/(3k\Omega \cdot 1\mu F)} = 1.4 e^{-t/3ms}$$

- b. What is  $v(t)$  when  $t = RC$ ?

$$\tau = RC = 3 \times 10^3 \cdot 1 \times 10^{-6} = 3ms$$

$$v(3ms) = 1.4 e^{-3ms/3ms} = 1.4 \cdot 1.4 \cdot 0.3679 = 0.515V$$

- c. Plot  $v(t)$

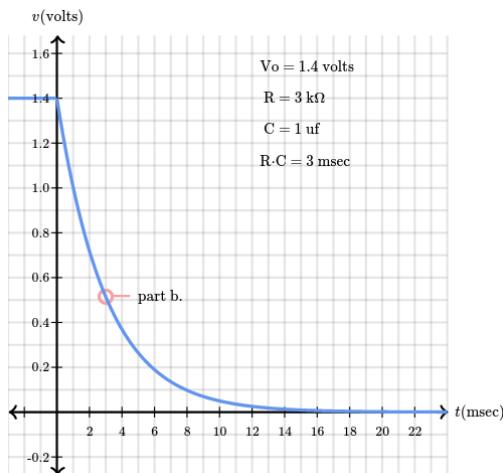


Figure 12.9: Plot RC natural response. From

**RC step response**

$$\text{Total} = \text{Forced response} + \text{Natural response} \quad (12.12)$$

- Forced response: No start charge on capacitor and connected voltage supply
- Total response: what the circuit actually response.

$$v(t) = V_s + (V_0 - V_s)e^{-t/RC} \quad (12.13)$$

- $V_s$  is the height of the voltage step
- $V_0$  is the initial voltage on the capacitor

**LC natural response**

$$s^2 + 1/LC = 0 \quad (12.14)$$

**Euler's identities**

$$\begin{aligned} e^{+jx} &= \cos x + j \sin x \\ e^{-jx} &= \cos x - j \sin x \end{aligned}$$

where  $j$  is the name for  $\sqrt{-1}$

**Current function of time**

$$i(t) = \sqrt{\frac{C}{L}} V_0 \sin \omega_o t \quad (12.15)$$

**Natural frequency of LC circuit**

$$\omega_o \equiv \sqrt{\frac{1}{LC}} \quad (12.16)$$

**RLC**

The RLC end text circuit is the electronic equivalent of a swinging pendulum with friction.

$$s = \frac{-R \pm \sqrt{R^2 - 4L/C}}{2L} \quad (12.17)$$

$$s = -\alpha \pm \sqrt{\alpha^2 - \omega_o^2} \quad (12.18)$$

where  $\alpha = \frac{R}{2L}$  and  $\omega_o = \frac{1}{\sqrt{LC}}$

## 12.2 Amplifier

Saturates mean that it will become a state line since it can not become more than the input nor can it be less than what is going in the opposite direction.  $v_0 = A(v^+ - v^-)$

The current to  $v^-$  and  $v^+$  will be close to zero and in an ideal circuit will be zero.  
We can say  $v^- = v^+$  when there is a feedback loop (negative terminal and noninverting amp).

### 12.2.1 Inside Op-Amp

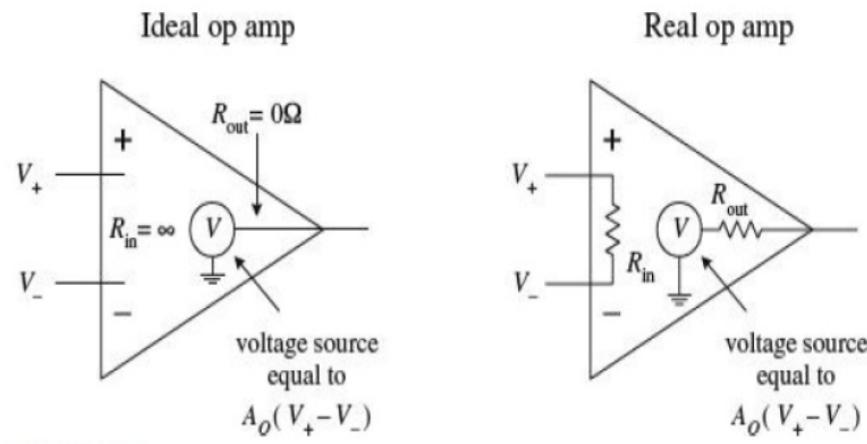


Figure 12.10: Inside opAmp. From

In ideal op amp the resistancce will be infinity. However the in realaty there is not infinit resistance therfore if one would have a voltage devider with larger resistanse then the internal resistanse the current will go threw  $V_+$  and  $V_-$ . The output in  $0\Omega$  so we will not have any voltage drop, not in reality however.

### 12.2.2 Implementation of Op-Amp Comparator

An op amp without negative feedback (a comparator)

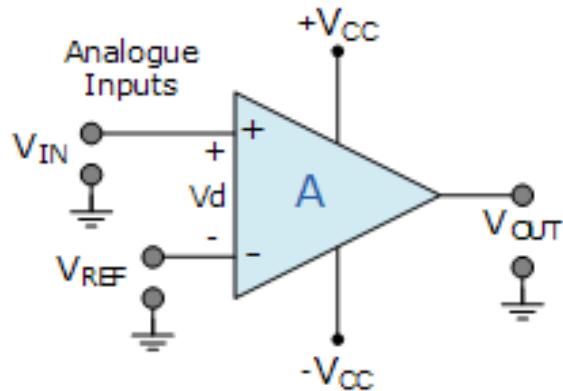


Figure 12.11: Op-Amp comparator. From

When  $V_+ > V_-$  then  $V_o = +V_{cc}$ . When  $V_+ < V_-$  then  $V_o = -V_{cc}$

#### Non inverting (with feedback loop)

Amplifies the input but dose not change it sign (non inverting). An op amp with negative feedback (a non-inverting amplifier)

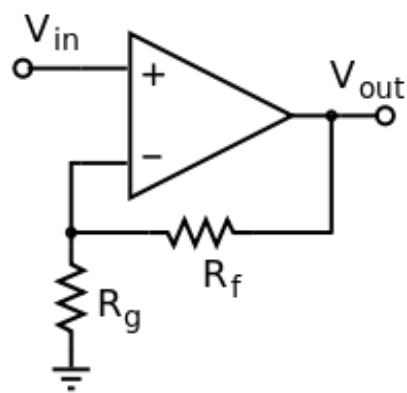


Figure 12.12: Op-Amp Feedback (Non inverting). From

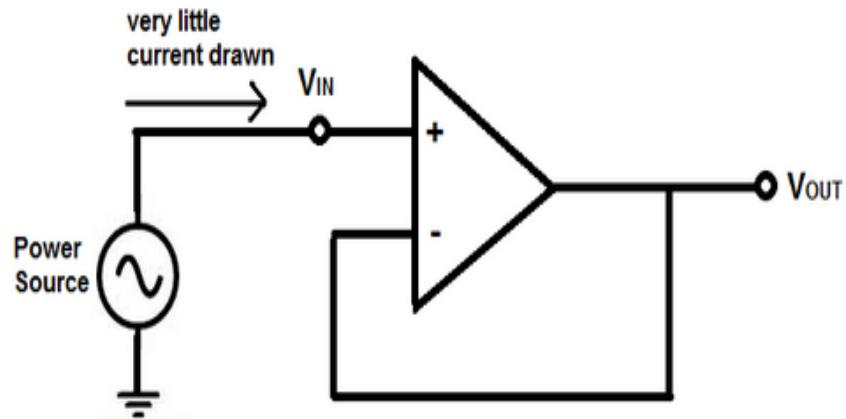
**Voltage follower**

Figure 12.13: Op-Amp voltage follower. From

$$v_+ = v_- = v_{in} = v_{out}$$

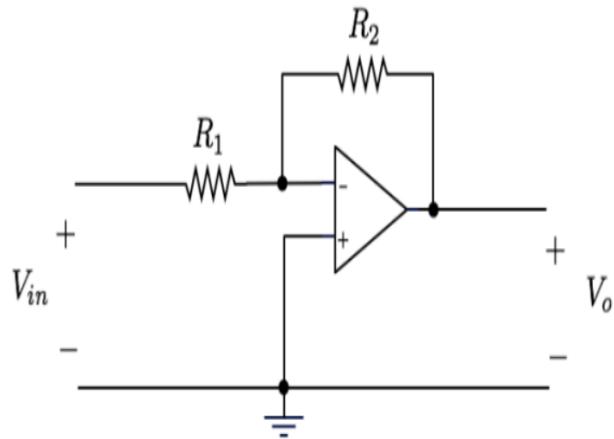
**Inverting**

Figure 12.14: Op-Amp voltage follower. From

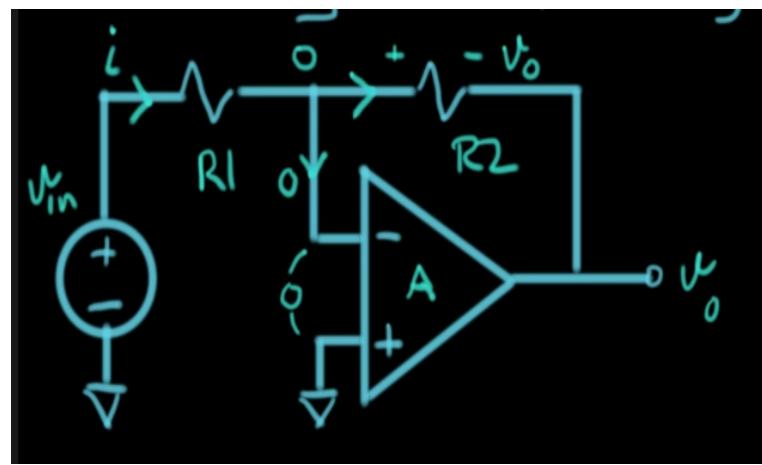


Figure 12.15: Op-Amp calculations for inverting. From

As seen from the image the current will be  $i = v_{in}/R_1$  and  $i = -v_0/R_2$ . We combined them and get:  

$$V_0 = \frac{-R_2}{R_1} V_{in}$$

### Summing

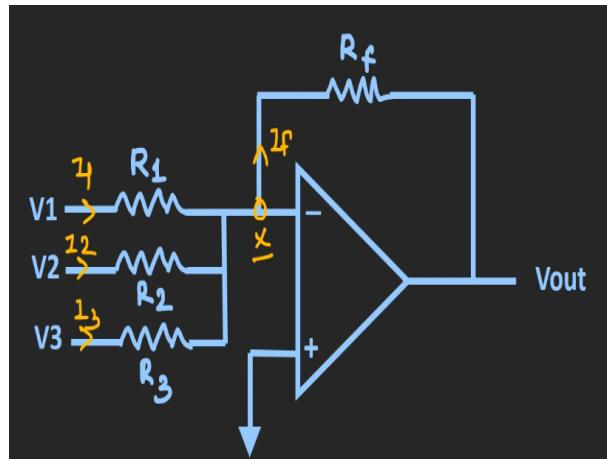


Figure 12.16: Op-Amp summing. From

Applying KCL we get:  $I_f = I_1 + I_2 + I_3 \frac{v_1 - 0}{R_1} + \frac{v_2}{R_2} + \frac{V_3}{R_3} = \frac{0 - v_{out}}{R_f} \Rightarrow v_{out} = -(\frac{R_f}{R_1}V_1 + \frac{R_f}{R_2}V_2 + \frac{R_f}{R_3}V_3)$

### 12.2.3 Gerneral example

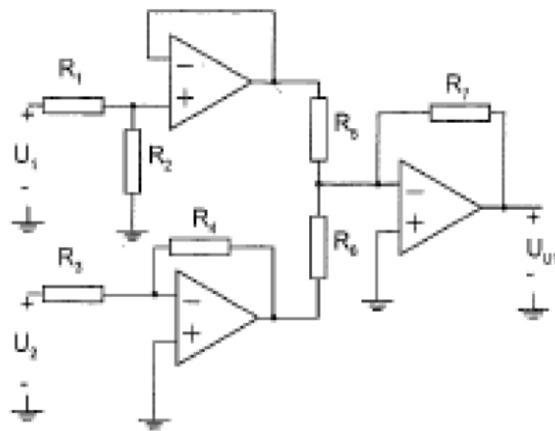


Figure 12.17: Op-Amp example. From

**General opservations:** We can divide the circuit into three pieces. The one on the top is a *Voltage follower*, we can call it circuit 1. The one on the bottom is a *Inverting op-amp* as is the middle one. We can name the bottom one circuit 2 and the middle one is circuit 3.

#### We start with circuit 3:

We name node **A** between  $R_5$  and  $R_6$ . KCL:  $I = I_5 + I_6$  and  $I = I_7$  since there is no current in  $-$  terminal.

$$I = \frac{-U_{ut}}{R_7} = I_5 + I_6 = \frac{v_{O1}}{R_5} + \frac{v_{O2}}{R_6}$$

Because of Ohm's law and  $v_{overresistor} = v_{head} - v_{tail}$ .

#### To solve $v_{O1}$ we need to solve circuit 1:

Applying KCL between  $R_1$  and  $R_2$  we get  $I_1 = I_2 + I_3$  were  $I_3 = 0$  since there is no current going in the  $+$  terminal.  $I_1 = I_2$  Ohm's law result in  $\frac{U_1 - v_{O1}}{R_1} = \frac{V_{O1} - 0}{R_2}$  since in a voltage follow circuit

$$v^- = v^+ = v_{out}. \text{ We then get } v_{out} = \frac{R_2}{R_1 + R_2} U_1$$

$$\text{then } I_5 = \frac{R_2}{(R_1 + R_2)R_5} U_1.$$

#### To solve $v_{O2}$ we need to solve circuit 2:

KCL gives  $I_3 = I_4$  since there is no current in  $-$  terminal.

$$\text{Ohm's law } \frac{U_2 - v^-}{R_3} = \frac{v^- - v_{O2}}{R_4}.$$

$$v^- = v^+ = 0 \text{ with result in } \frac{U_2}{R_3} = \frac{-v_{O2}}{R_4}.$$

$$v_{O2} = \frac{-R_4}{R_3} U_2$$

$$\text{The current } I_6 = \frac{-R_4}{R_3 R_6} U_2$$

#### Going back to circuit 3:

$$I = I_5 + I_6 = \frac{R_2}{(R_1 + R_2)R_5} U_1 + \frac{-R_4}{R_3 R_6} U_2$$

$$\text{results in } \frac{-U_{ut}}{R_7} = \frac{R_2}{(R_1 + R_2)R_5} U_1 + \frac{-R_4}{R_3 R_6} U_2.$$

Finnal result:  $U_{out} = -R_7 \left( \frac{R_2}{R_5(R_1+R_2)} U_1 - \frac{R_4}{R_3 R_5} U_2 \right)$

## 12.3 Semiconductors

Semiconductors electrical conductivity is between conductors and insulators. The material becomes more “conductivity” or less resistance when the temprature rises. This behavier is oposite to that of metals.

### 12.3.1 Diodes

#### Ideal diodes

Cundoct electricity only one way. Current flowes threow Anode (A) to Cathode (K).

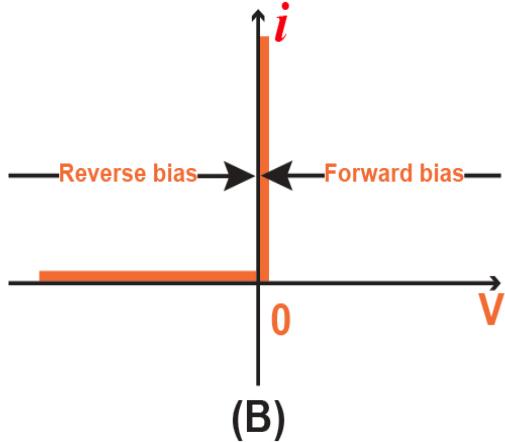


Figure 12.18: ideal diode iv-curv. From

#### Real diodes

**reverse breakdown** is when the diode lets negative current. On normal diodes it will perminantly fail, probably melts.

$$i = I_s(e^{qv_d/kT} - 1) \quad (12.19)$$

- $k$  is boltsmans constant ( $1.38 * 10^{-23} J/K$  Jouls/Kelvin)  $0K = -273^{\circ}C$
- $T$  is tempreture of device (kelvin)
- $I_s$  is strucual current (silicon  $10^{-12} A$ ), the small reverse current
- $q$  is the charge of an electron ( $1.1609 * 10^{-19} C$ )

Difficult to use formula unless numerical analasys.

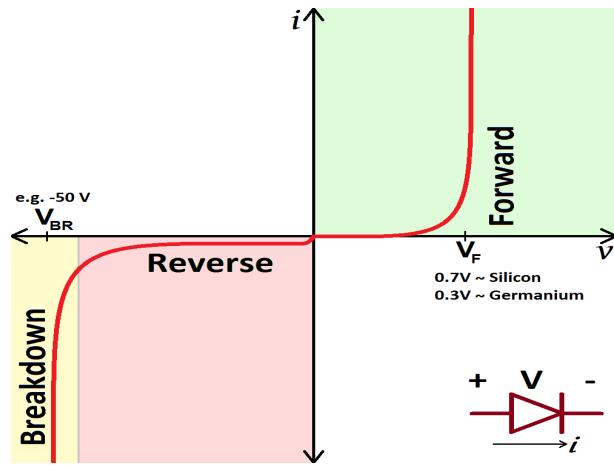


Figure 12.19: real diode iv-curv. From

### The DC model of a diode

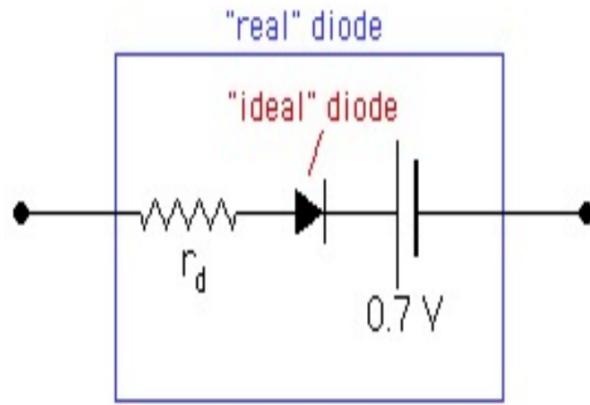


Figure 12.20: real diode dc model. From

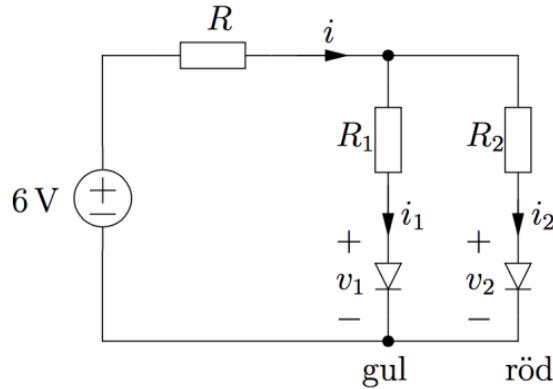
**Example: diode**

Figure 12.21: diodes example. From

$$i_1 = 40mA, v_1 = 2v$$

$$i_2 = 50mA, v_2 = 1.6v$$

Kirchhoff's strömlag:  $i = i_1 + i_2 = 40mA + 50mA = 90mA = 0.09A$

Kirchhoff's spänningsslag: (Loop 1)

$$+ 6 - iR - R_1 i_1 - 2 = 0$$

Kirchhoff's spänningsslag: (Loop 2)

$$+ 6 - Ri - R_2 i_2 - 1.6 = 0$$

Vi har 3 okända och 2 eqvationer. Närmed så kan vi sätta ett värde på R sådant att motstånden ärstöre än noll

$$R = 20\Omega \Rightarrow R_1 = 55\Omega; R_2 = 52\Omega$$

### Zener Diodes

Allows to enter and reenter the reverse breakdown with out permanently failing. symbol. Justly lower voltage for reverse breakdown used for analog to digital conversion and vice versa and a voltage regulator.

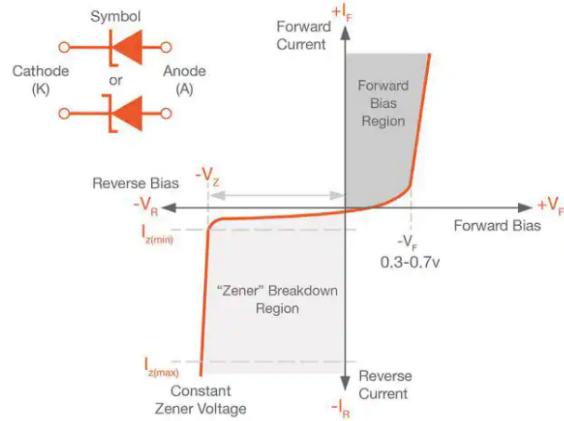


Figure 12.22: zener diode iv-curv. From

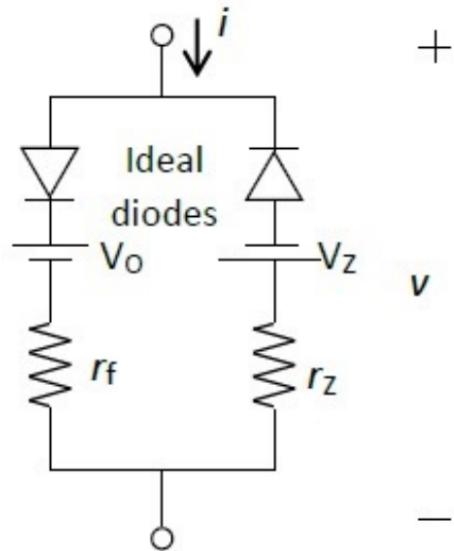


Figure 12.23: zener diode dc model. From

**Example** Let  $V_{in} = 9V$  and  $V_z = 5V$ . Determent the minimum value of  $R$  if the maximum current is  $20mA$  threw  $R$ , assume  $R_L = \text{inf}$ . Then determan the minimum resistance for  $R_L$

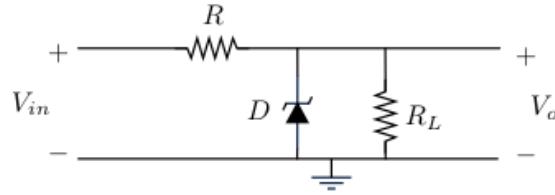


Figure 12.24: zener diode example. From

Let  $I$  be the current over  $R$ ,  $I_1$  the current over the zener diode and  $I_2$  be the current over  $R_L$ .

$$\begin{aligned} I &= \frac{V_{in} - V_o}{R} \\ \Rightarrow R &> \frac{9 - 5}{20 \cdot 10^{-3}} \Omega = 200\Omega \end{aligned}$$

KCL:  $I = I_1 + I_2$

$I_1 = 0$  if  $R_L$  is to low

$$\begin{aligned} I_2 &= \frac{V_0 - 0}{R_L} \\ \Rightarrow I = I_2 &\Rightarrow 20mA = \frac{5v}{R_L} \\ \Rightarrow R_L &> \frac{5}{20 \cdot 10^{-3}} \Omega = 250\Omega \end{aligned}$$

### 12.3.2 Transistors

All Semiconductors are non linear. But we can use transistors with linear modules.  
NPN and PNP symbols. Base (B), Collector (C) and Emitter (E)

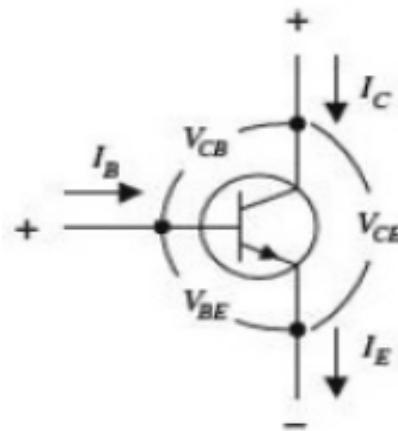


Figure 12.25: Transistor NPN. From

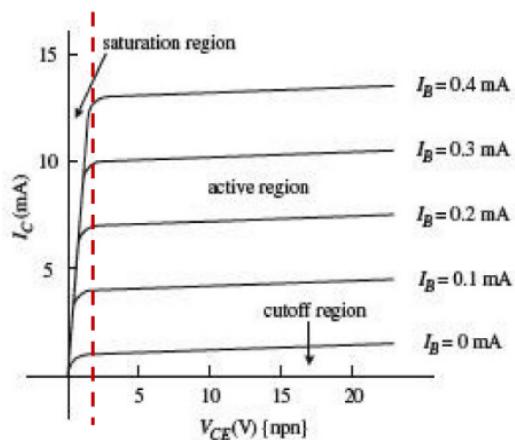


Figure 12.26: Transistor Operational mode. From

**Transistor modes****Cutoff:** Open switch

$$I_B = I_E = I_C = 0 \\ V_{BE}, V_{BC} < 0$$

**Saturation:** Closed switch

$$V_{CE} \leq V_{CE,sat}, I_B > 0$$

**Active:** Dependent current source

$$I_E = I_B + I_C, I_C = h_{FE}I_B = \beta I_B \\ V_{CE} > V_{CE,sat}, I_B > 0, V_{BC} = V_B - V_C < 0$$

**Example: Transistor** What is the largest value that  $R_B$  can have so that the transistor behaves as a switch (saturation mode)? The transistor has a current gain  $h_{FE}$  between 100 and 300. Suppose that  $V_{CE,sat} = 0.2V$  and  $V_{BE,sat} = 0.7$ .

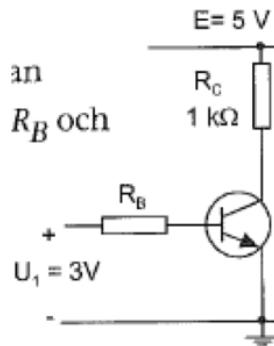


Figure 12.27: Transistor example. From

Om transistorn är bottnad (max current) gäller det att potentialen i C (mellan  $R_C$  och transistorn)

$$V_{CE} = V_C - V_E = V_C - 0 = V_{CE,sat} \\ V_C = V_{CE,sat} = 0.2V$$

⇒ Vi har då ett spänningsfall över  $R_C$  på  $5 - 0.2 = 4.8V$ .

$$R_C \cdot I_C = 4.8V \quad \text{Ohm's law} \Rightarrow I_C = \frac{4.8}{10^3} = 4.8mA$$

Vi vet att  $I_C = \beta \cdot I_B$  men är ej säkra på  $\beta$ :s värde. Det kan vara så långt som 100.

Vi kan behöva  $I_B = \frac{I_C}{100} = 48\mu A$  för att inte  $I_C$  ska bli för liten. Alltså:  $I_B > 48\mu A$  för att garantera bottnad transistor.

Betrakta nu slingan där  $I_B$  går.

$$\text{Potentialvandring: } U_1 - R_B I_B - V_{BE,sat} = 0, \quad 3 - R_B I_B - 0.7 = 0$$

$$\text{eller } 2.3 = R_B I_B \Leftrightarrow I_B = \frac{2.3}{R_B}$$

$$\text{Använd olikhet } I_B > 48\mu A \Rightarrow \frac{2.3}{R_B} > 48\mu A \Rightarrow \frac{2.3}{48 \cdot 10^{-6}} > R_B \text{ eller } R_B < 47.9k\Omega$$

## 12.4 Digital Circuits: Combinatorial and Sequential Networks

### Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	$\bar{A}$	$AB$	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table border="1"><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table border="1"><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1"><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1	<table border="1"><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

Figure 12.28: Logic Gates. From

**Boolean algebra** (or:  $+$ ), (and:  $\cdot$ )

$$(x')' = x$$

$$x + x = x$$

$$x \cdot x = x$$

$$x + (y + x) = (x + y) + z$$

$$x(yz) = (xy)z$$

$$x + x' = 1$$

$$x \cdot x' = 0$$

$$x(y + z) = xy + xz$$

$$x + yz = (x + y)(x + z)$$

$$x + 1 = 1$$

$$x + xy = x$$

$$x \cdot 0 = 0$$

$$xy + \bar{x}z = xy + \bar{x}z + yz$$

$$x + 0 = x \quad (x + y)(\bar{x} + z) = (x + y)(\bar{x} + z)(y + z)$$

$$x \cdot 1 = x$$

$$\overline{(x + y)} = \overline{xy}$$

$$\overline{(xy)} = \overline{x} + \overline{y}$$

### 12.4.1 Combinational

$$f(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = x_6x_4(x_5 + x_3 + x_0) + x_2x_1$$

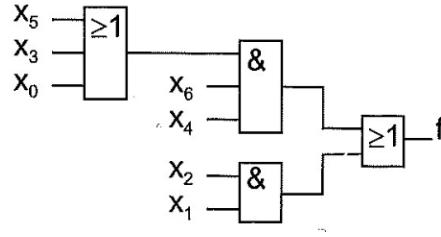


Figure 12.29: Boolean factorization. From

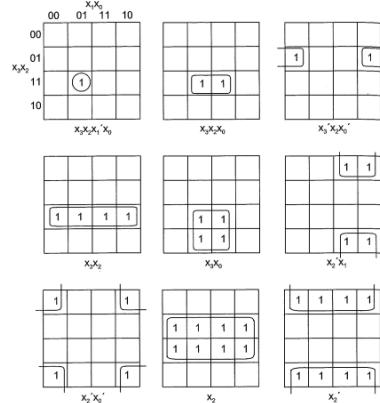


Figure 12.30: KMap patterns. From

KMMaps works from 4 variables.

*Delays*, there is a small delay for each gate.

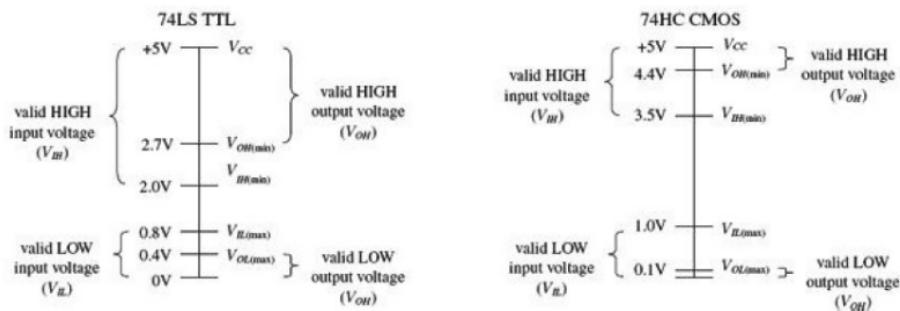


Figure 12.31: Logic families. From

**Example: design circuit**

$$f(x_3, x_2, x_1, x_0) = \sum(0, 2, 4, 5, 6, 7) + d(8, 10, 12, 14) \quad - \text{sum-of-products} \quad (12.20)$$

$X_1 X_0$

$0 = 0000$	00	01	11	10
$2 = 0010$	1	0	0	1
$4 = 0100$	1	1	1	1
$5 = 0101$	-	0	0	-
$6 = 0110$	-	0	0	-
$7 = 0111$				
$X_3 X_2$				
$8 = 1000$				
$10 = 1010$				
$12 = 1100$				
$14 = 1110$				

$$f = \overline{x_0} + \overline{x_3}x_2 \quad (12.21)$$

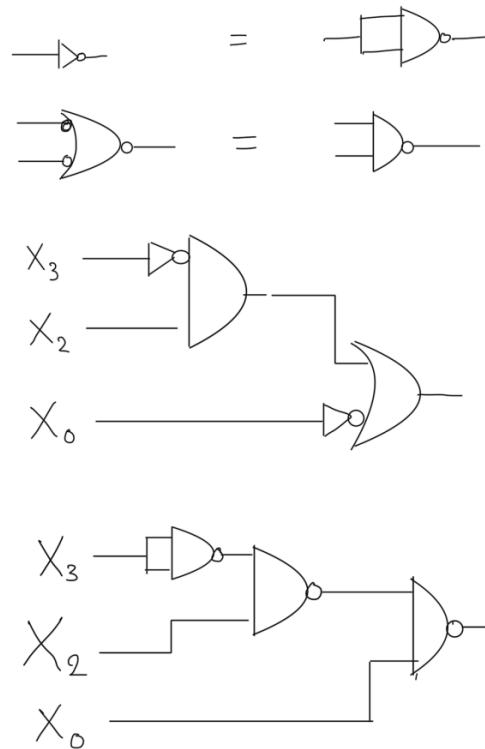


Figure 12.32: logic Combinational logic.

### 12.4.2 Sequential

Remembers the state what happened before.

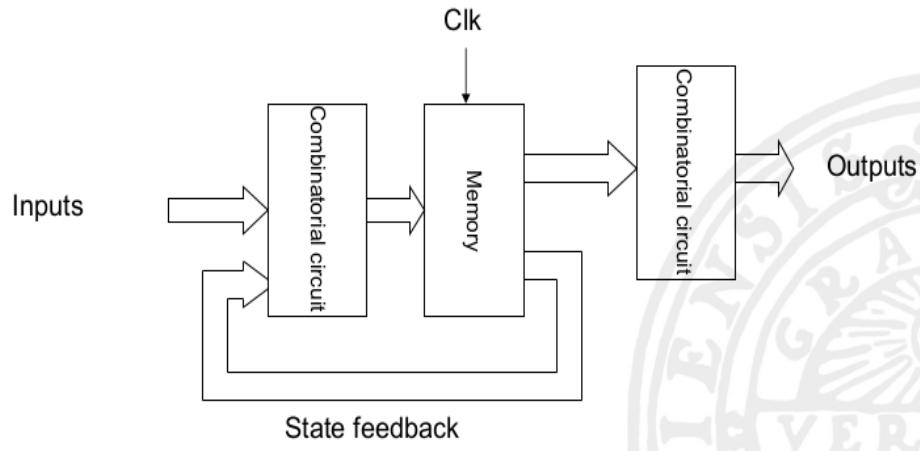


Figure 12.33: Sequential circuits moore type. From

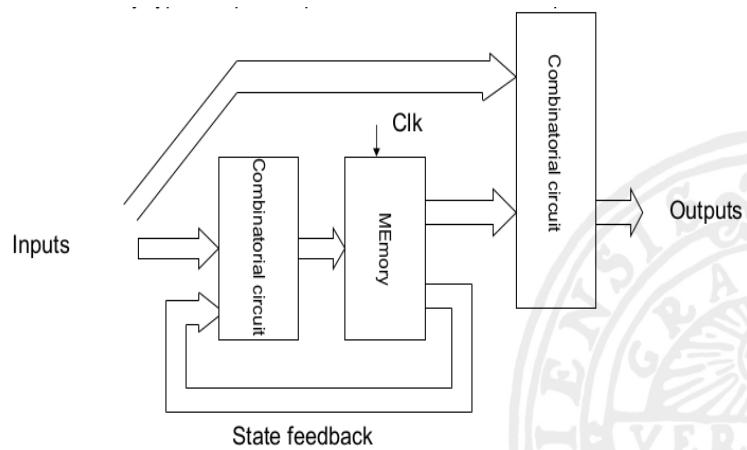


Figure 12.34: Sequential circuits mealy type. From

## Memory

Description	Circuit
SR-latch (memory cell)	
Gated SR-latch	
Gated D-latch	
D Flip Flops	

Table 12.1: Memory. From

### Timind diagram

D flip flop vs. D latch

Rising edge:  $\uparrow\downarrow$

Faling edge:  $\downarrow\uparrow$

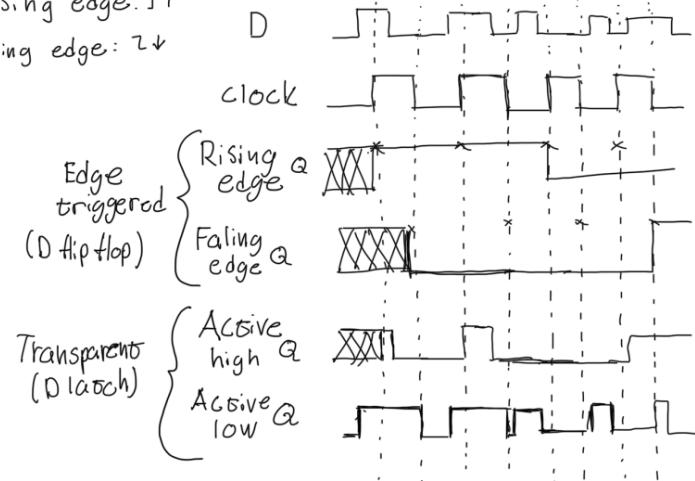


Figure 12.35: Timing diagram.

### State diagram

#### State code

State name	$q_1$	$q_0$
S0	0	0
S1	0	1
S2	1	0
S3	1	1

**State Table** In sequential circuits we use *state table* instead of *truth table*. ex of state table format:

Previous state	Input	Next state	Output
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	0 1	1
0 1	1	1 0	1

**Moore-circuit** Note: in more machine we need a end state for output 1

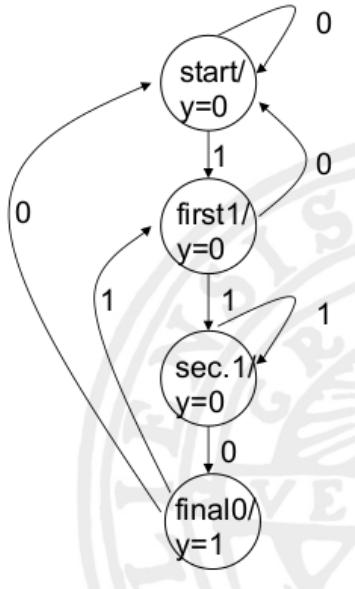


Figure 12.36: State diagram moore. From

**Mealy-circuit** Note: in mealy machine we dont need that.

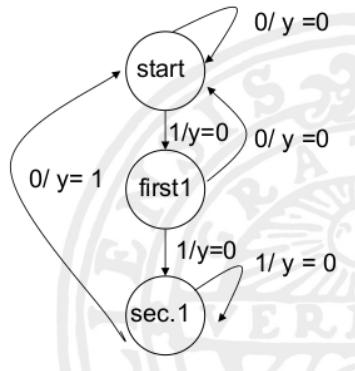


Figure 12.37: State diagram mealy. From

### 12.4.3 Shift registers

Each bit is processed simultaneously. Makes much simpler circuit.

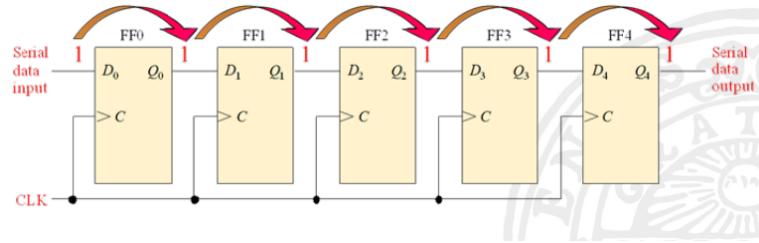


Figure 12.38: Shift register. From

### 12.4.4 Multiplexer

$$f(A, B, C) = \sum(0, 1, 5, 6)$$

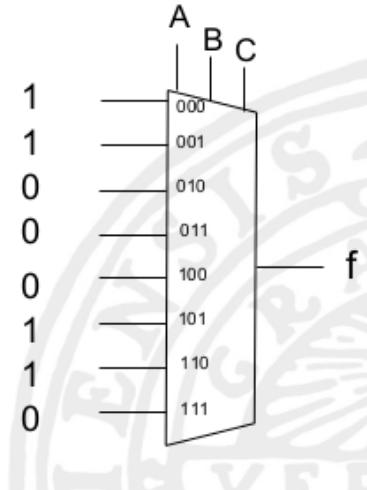


Figure 12.39: Mux. From

## 12.5 AC Circuit Analysis & Filtering

Note:  $G(\omega) = H(\omega)$

- *sinusoidal steady state* same as AC (analysis)
- *phasor* a complex number representation of sinusoid
- *transfer function* relationship with input and output voltage  $G(jw) = \frac{v_{out}}{v_{in}}$
- *Impedance* representation of  $\frac{v_{in}}{i}$  (z)
- *Angular frequency*  $w = 2\pi f$  where  $f$  is frequency in Hz with cycles per second
- *Operational frequency*  $\frac{1}{\sqrt{2}}$

$$v(t) = v_p \cos wt + \phi \quad (12.22)$$

### Phasor 3 representations

- Rectangular form  $z = x + jy$
- polar form  $z = r \angle \phi$
- exponential form  $z = ze^{j\phi}$

### Phasor conversion

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \phi &= \arctan \frac{x}{y} \\ x &= r \cos(\phi) \\ y &= r \sin(\phi) \\ e^{j\phi} &= \cos \phi + j \sin \phi \\ \cos \phi &= Re(e^{j\phi}) \\ \sin \phi &= Im(e^{j\phi}) \end{aligned}$$

### Phasor operations

$$\begin{aligned} \text{addition: } z_1 + z_2 &= (x_1 + x_2) + j(y_1 + y_2) \\ \text{subtraction: } z_1 - z_2 &= (x_1 - x_2) + j(y_1 - y_2) \\ \text{multiplication: } z_1 z_2 &= r_1 r_2 \angle (\phi_1 + \phi_2) \\ \text{division: } \frac{z_1}{z_2} &= \frac{r_1}{r_2} \angle (\phi_1 - \phi_2) \\ \text{inverse: } \frac{1}{z} &= \frac{1}{r} \angle -\phi \\ \text{square root: } \sqrt{z} &= \sqrt{r} \angle (\phi/2) \\ \text{complex conjugate: } z^* &= x - jy \end{aligned}$$

### Transfer function

$$\begin{aligned} H(j\omega) &= |H(j\omega)| \angle H(j\omega) \\ H(w) &= \frac{v_o(w)}{v_i(w)} \end{aligned}$$

**Euler formula**

$$e^{jx} = \cos x + j \sin x$$

$$e^{-jx} = \cos x - j \sin x$$

They are used to represent sinus wave.

We instead calculate with complex exponentials with euler formula instead of sinus waves

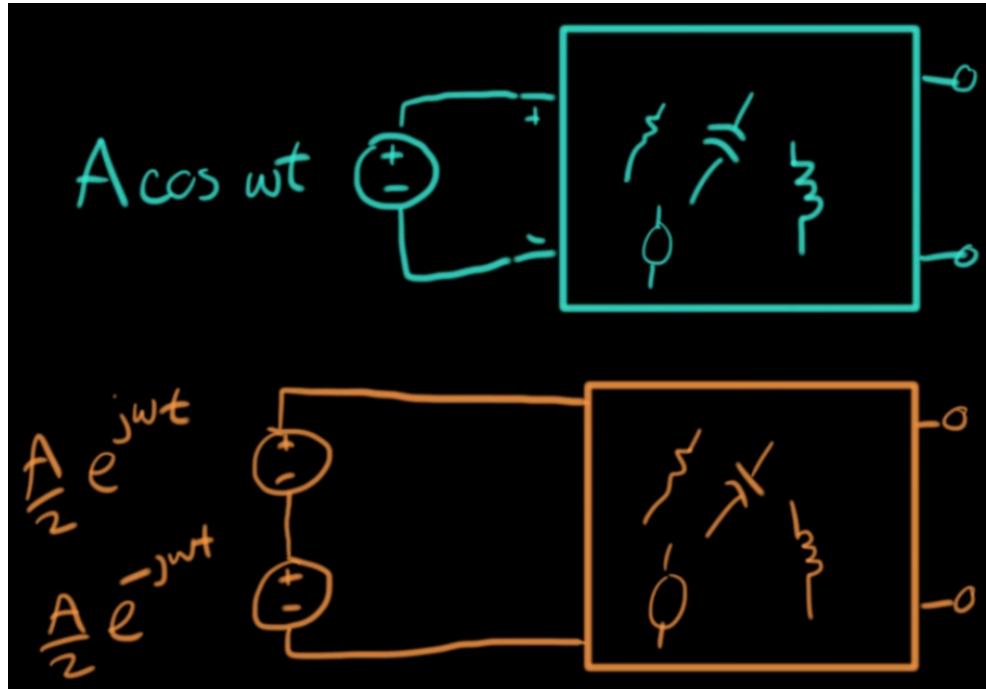


Figure 12.40: AC euler represent demostration. From

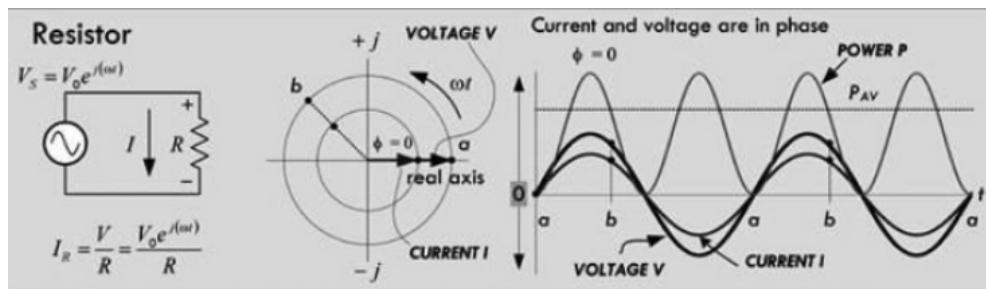
**AC Resistor representation**

Figure 12.41: AC resistor represent. From

### 12.5.1 Impedance

*Impedance*  $\frac{v_{in}}{i}$  (z)

$$\begin{aligned}\text{Resistor : } \frac{v}{i} &= R \\ \text{Inductor : } \frac{v}{i} &= jwl \\ \text{Capasitor : } \frac{v}{i} &= \frac{1}{jwc}\end{aligned}$$

### 12.5.2 Filters

The filter can either be active and passive filters:

- *Passive filters* only use active components like resistor, capacitor and inductor
- *Active filters* use opamp (can amplify output)

There are 4 common filtering types:

- *High-pass filter (HP)* lets higher frequencies through
- *Low-pass filter (LP)* lets lower frequencies through
- *Bandpass filter (BP)* within a frequency span. Can be combined with HP and LP in series
- *Notch filter* opposite of BP

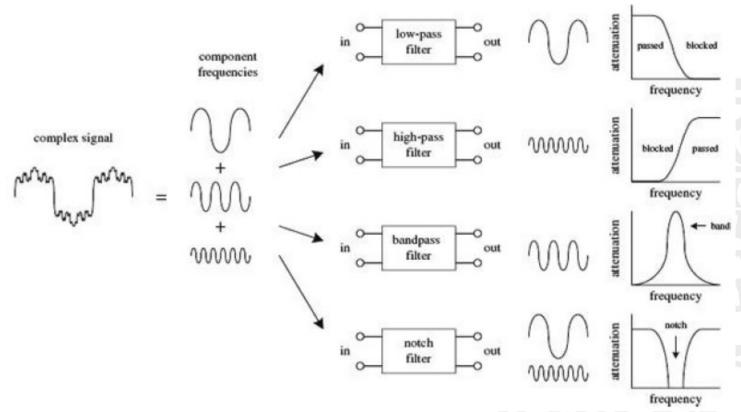


Figure 12.42: Signal filtering. From

Filters can be constructed as follows:

*RC filter, RL Filter, LC filter, L-filter,  $\pi$ -filter and T-filter.*

### Some resources to construct filters

- RF tools
- Filterwizard

Example: HP filter

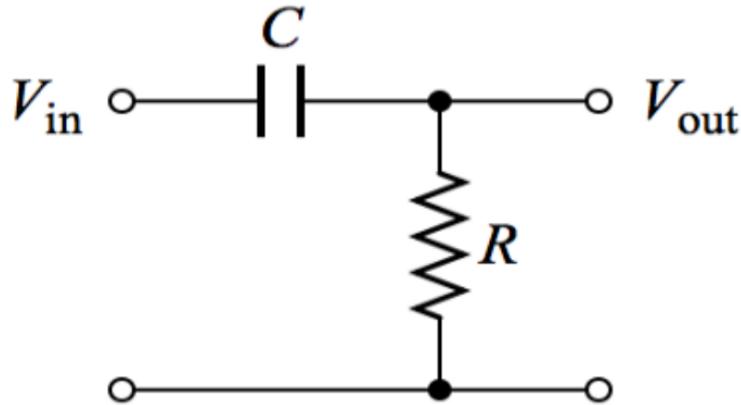


Figure 12.43: HP Filter Circuit.png. From

$$\begin{aligned}
 V_{out} &= V_{in} \frac{z_R}{z_C + z_R} && \text{-From voltage divider circuit} \\
 &= V_{in} \frac{R}{\frac{1}{j\omega c} + R} && \text{-From impedance axiom} \\
 &= V_{in} \frac{\frac{R}{1}}{\frac{1}{R(j\omega c)} + 1} = V_{in} \frac{R(j\omega c)}{R(j\omega c) + 1}
 \end{aligned}$$

Therefore we get  $G_{HP} = \frac{R(j\omega c)}{R(j\omega c) + 1}$

$$\begin{aligned}
 |G_{HP}| &= \frac{1}{\sqrt{2}} = \frac{R\omega c}{\sqrt{R^2\omega^2c^2 + 1}} \\
 &\Rightarrow \frac{\sqrt{R^2\omega^2c^2 + 1}}{\sqrt{2}} = R\omega c \\
 &\Rightarrow \frac{R^2\omega^2c^2 + 1}{2} = R^2\omega^2c^2 \\
 &\Rightarrow R^2\omega^2c^2 + 1 = 2R^2\omega^2c^2 \\
 &\Rightarrow R^2\omega^2c^2 = 1 \\
 &\Rightarrow R\omega c = 1 \\
 &\Rightarrow R = \frac{1}{\omega c}
 \end{aligned}$$

The following is given:  $\omega = 2\pi f$ , 1kHz for HP filter. The capacitor should not be larger than 2.2uF. Therefore we chose  $c = 1uF$ , this gives us  $R \approx 160\Omega$ .

$$\begin{aligned}
\arg(G_{HP}) &= \arg\left(\frac{160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f \cdot j}{160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f \cdot j + 1}\right) \\
&= \arg(160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f \cdot j) - \arg(160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f \cdot j + 1) \\
&= \frac{\pi}{2} - \arctan\left(\frac{160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f}{1}\right) \\
&= \frac{\pi}{2} - \arctan\left(\frac{\pi \cdot f}{3125}\right)
\end{aligned}$$

$$\begin{aligned}
\arg(G_{HP}) &= \arg\left(\frac{160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f \cdot j}{160 \cdot 2 \cdot \pi \cdot 10^{-6} \cdot f \cdot j + 1}\right) \\
&\Rightarrow \frac{z_1}{z_2} = \frac{r_1}{r_2} / \phi_1 - \phi_2 \\
&\Rightarrow \frac{z_1}{z_2} = \frac{\sqrt{(0.32 \cdot 10^{-3} \cdot f\pi)^2}}{\sqrt{(0.32 \cdot 10^{-3} \cdot f\pi)^2 + 1}} / \arctan 0 - \arctan \frac{1}{0.32 \cdot 10^{-3} \cdot f\pi} \\
&\Rightarrow \arg\left(\frac{z_1}{z_2}\right) = -\arctan \frac{1}{0.32 \cdot 10^{-3} \cdot f\pi} \\
&= -\arctan \frac{1}{0.32 \cdot 10^{-3} \cdot f\pi} \\
&= \frac{\pi}{2} - \arctan \frac{f\pi}{3125}
\end{aligned}$$

### Example: exponential representation

$$\begin{aligned}
H(j\omega) &= \frac{j\pi}{1 + j2\pi} = |H(j\omega)| e^{j\arg(H(j\omega))} \\
\Rightarrow |H(j\omega)| &= \frac{\sqrt{\pi^2}}{\sqrt{1^2 + 2^2\pi^2}} = \frac{\pi}{\sqrt{1 + 4\pi^2}}
\end{aligned}$$

$$\begin{aligned}
\arg(H(j\omega)) &= \arg\left(\frac{j\pi}{1 + j2\pi}\right) = \arctan\left(\frac{0}{\pi}\right) - \arctan\left(\frac{1}{2\pi}\right) \\
&= \frac{\pi}{2} - \arctan(2\pi) \\
\Rightarrow H(j\omega) &= \frac{\pi}{\sqrt{1 + 4\pi^2}} e^{j \arctan(2\pi)}
\end{aligned}$$

### Series of filter

One can commbind filters to create more complex filters. The *transfer function* will be the multiplication of

$$G_{tot} = G_1 \cdot G_2 \cdot \dots \cdot G_n \quad (12.23)$$

The *Gain* will be the gain for each individual transfer function multiplied

$$|G_{tot}| = |G_1| \cdot |G_2| \cdot \dots \cdot |G_n| \quad (12.24)$$

The phase function is the sum of the phase functions

$$\text{Arg}(G_{tot}) = \text{Arg}(G_1) + \text{Arg}(G_2) + \dots + \text{Arg}(G_n) \quad (12.25)$$



## Chapter 13

# Numerical Methods and Simulation

## 13.1 Matlab

<https://www.egr.msu.edu/aesc210/systems/MatlabCheatSheet.pdf>

### Viktiga syntax/comandons

```
% creates a vector of N lenght
zeros(1,N);
% creates a matrix of N*N size
zeros(N);
% avrage value with is 2 in this cases
mean([1 2 2 3]);
vector(line,row);
% holes the current plot and plots
% new over old
hold on
% naming the plot x axis
xlabel("some x lable")
% naming the plot y axis
ylabel("some y lable")
title("some title for the plot")
ode45(@(t,y) 2*t, tspan, y0);
[t,y] = ode15s(@(t,y) myode(t,y,A,B),
              tspan, y0);
% gernerates a row of vector
linspace(StartPoint, EndPoint);
fzero(func, currentValueOfFunction,
      optimeset('Tolx', 1e-02));
% Vector not array
[0; 1; 2];
% the number of dimension of vector
numel(x0);
% generate values
repmat(x0(i), N, 1);
% the cumulative sum of Noise starting
% at the beginning of the first array
% dimension in.
% Noise whose size does not equal 1.
cumsum(rand)
```

## 13.2 Aritmatic

När beräkningar sker så använder den inte exaxta värdet utan konverterar till binärt och därmed får ungefärlig väde. Räknar inte med bråk utan floting point.

(Mantissa  $m$  hur många index det finns för  $d_i$ ), (Bas  $\beta$ ), (exponent  $e$ ). Man kan skriva ett decimaltal så här:  $x = m\beta^e$ . (Normaliserad form) är då första talet är en siffra ( $1.23 * 10^{-1}$  ej  $0.123$ )  $m = d_0 * 2^0 + d_1 * 2^{-1} + d_2 * 2^{-2} + \dots + d_n * 2^{-p}$ . Normaliserad form ger att  $d_0$  i alltid 1 i binär form och därmed så behöver man inte sparra den. Den kallas hidden bit.

$E$  används för att konvertera till en normal form tex  $0.d_1d_2..d_{52} \cdot 2^{-1022}$  i subnormal form blir  $1.d_1d_2..d_{52} \cdot 2^{E-1022}$ , där  $E = 1$ , i normal form.

### 13.2.1 IEEE

IEEE, ger en standard till olika representationer.

- Singleprecision (32 bit, enkel precision)
- Double precision (64 bit, dubbel precision)
- Extendedprecision (80 bit, utökad precision)

matlab använder double presission. Tal som är större än realmax i matlab blir Inf och tal som är ogiltiga är NaN.

### 13.2.2 Maskinepsilon

Def: Gapet mellan 1 och nästa representerbara tal, minsta talet  $\epsilon_M$  för vilket gäller  $1+\epsilon_M > 1$ . Det är ett mått på tals systemets precision.  $\epsilon_M \approx 10^{-16}$

```
if (x == y)
    ifabs ((x-y)/x) < tol
```

### 13.2.3 Diskretiseringfel

Vid exemplet när man beräknar derivatan så är avrundningsfelet den dominanta. Medans vid Störe h så domineras av diskretiseringfel

## 13.3 ODE

Skillnaden mellan metod och modell är att metoden är hur man gör, medan modell är strukturen av förföljelsen. För att man ska kunna lösa en ode så behöver den vara en entydlig lösning samt små skilnader på inparametrar ger littet skilnad i resultat.

### Löser med ode45

```
%%%%%% myODE.m
function yprim = myODE(t,y)
    % y'(t) = 3y(t)(1-y(t))
    yprim = 3*y*(1-y)
end
%%%%%% script.m
% Tidsintervall mellan 0 till 10
% Begynnelse vilkor på 0.1
[t,y] = ode45(@myODE, [0 10], 0.1)
plot(t,y)
```

### 13.3.1 Numeriska metoder

När numerisk fel är mindre än proportionellit ex  $h^2$   
då blir avrundningsfelet den dominanta

**Implécita metoder** Ovilkoliga stabilita.

**Explicit metoder** Kan vara effektivare, fämst för icke styva ode'r

#### Euler framåt (explicit metod)

Tar lutningen vid punkten man är i och beräknar värdet för ett givet steg.

**Ex:**

$$\frac{y'(x)}{x} - y \sin(x) = 0, \quad y(0) = 1$$

Euler framåt:  $y_{k+1} = y_k + h f(t_k, y_k)$

$$y'(x) = xy \sin(x), \quad f(x, y) = xy \sin(x)$$

$$y_{i+1} = y_i + h f(x_i, y_i) = y_i + h x_i \cdot y_i \sin(x_i)$$

Väljer  $h = 0.1$

$$y_0 = 1$$

$$y_1 = y_0 + h x_0 y_0 \sin(x_0) = 1 + 0.1 \cdot 0 \cdot 1 \sin 0 = 1$$

$$y_2 = y_1 + h x_1 y_1 \sin(x_1) = 1 + 0.1 \cdot 0.1 \cdot 1 \sin 0.1$$

$$= 1.001$$

$$y_3 = y_2 + h x_2 y_2 \sin(x_2)$$

$$= 1.001 + 0.1 \cdot 0.2 \cdot 1.001 \sin 0.2 = 1.005$$

\*

$$y_{10} = 1.2895$$

Aritmetiska lösning (räknat ut exakta värdet)

$$y(1) = e^{\sin(1)-1-\cos(1)} = 1.3514$$

$$\text{Fel} = 1.2895 - 1.3514 = -0.0619$$

$$\text{Väljer } h = 0.01, \quad y_{100} = 1.3448 \quad \text{Fel} = -0.00669$$

$$\text{Väljer } h = 0.01, \quad y_{1000} = 1.35076 \quad \text{Fel} = -6.74 \cdot 10^{-4}$$

Vi ser att felet är proportionellit mot steg ( $h$ ).

Fel  $\propto h$

#### Euler bakåt (implicit metod)

Tar lutningen ett steg framåt.

**Ex:**

Euler bakåt:

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_{k+1})$$

$$y'(x) = xy \sin(x), \quad y(0) = 1$$

$$y_{i+1} = y_i + h x_{i+1} y_{i+1} \sin(x_{i+1})$$

$$\Rightarrow y_{i+1} - h x_{i+1} \cdot y_{i+1} \sin(x_{i+1}) = y_i$$

$$\Rightarrow y_{i+1} = \frac{y_i}{1 - h x_{i+1} \sin(x_{i+1})}$$

Väljer  $h = 0.1$

$$y_0 = 1$$

$$y_1 = \frac{y_0}{1 - h x_1 \sin(x_1)} = \frac{1}{1 - 0.1 \cdot 0.1 \cdot \sin 0.1} = 1.000999$$

$$y_2 = \frac{y_1}{1 - h x_1 \sin(x_1)} = 1.00499$$

\*

$$y_{10} = 1.42556$$

( $h$  är steget mellan det approximerade linjerna)  
Aritmetiska lösning:  $y(1) = e^{\sin(1)-1-\cos(1)} = 1.3514$

$$\text{Fel} = 1.42556 - 1.3514 = +0.07412$$

$$\text{Väljer } h = 0.01, \quad y_{100} = 1.35825 \quad \text{Fel} = +0.00681$$

$$\text{Väljer } h = 0.001, \quad y_{1000} = 1.35211 \quad \text{Fel} = +6.75 \cdot 10^{-4}$$

Vi ser att felet är proportionellit mot steg ( $h$ ). Fel  $\propto h$

#### Trapetsmetoden (implicit metod)

Trapetsmetoden uses the average value of the euler framåt and backåt.

**Ex:**

Trapetsmetoden

$$y_{i+1} = y_i + 0.5 h (f(t_i, y_i) + f(t_{i+1}, y_{i+1}))$$

$$y' = xy \sin(x)$$

$$y_{i+1} = y_i + h/2(x_i \cdot y_i \sin(x_i) + x_{i+1} y_{i+1} \sin(x_{i+1}))$$

$$\Rightarrow y_{i+1} = \frac{y_i + h/2(x_i \cdot y_i \sin(x_i))}{1 - h/2(x_i) + \sin x_{i+1}}$$

$$y_0 = 1 \quad h = 0.1$$

$$y_1 = 0.00499$$

\*

$$y_{10} = 1.3343, \quad \text{Fel} = 0.00286$$

$$\text{Väljer } h = 0.01, \quad y_{100} = 1.35147 \quad \text{Fel} = +2.86 \cdot 10^{-5}$$

$$\text{Väljer } h = 0.001, \quad y_{1000} = 1.3514275 \quad \text{Fel} = +2.86 \cdot 10^{-7}$$

Vi ser att felet är proportionellit mot steg ( $h$ ). Fel  $\propto h^2$

### Heuns metod (explicit metod)

Trapetsmetoden ger bra approximationer men är dock implicit vilket gör det svårt för icke linjära ODE. Det kan huens metod lösa genom att approximera implicita delen.

#### Metod:

Huens metod

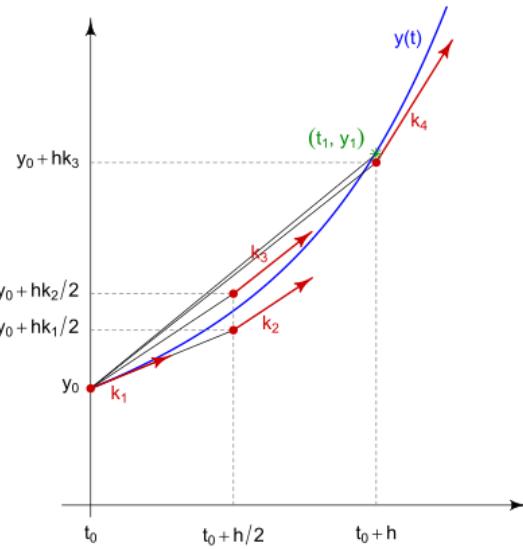
$$y_{i+1} = y_i + h_n/2[f(t_n, y_n) + f(t_{n+1}, h f(t_n, y_n))]$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i, y_i)$$

$$y_{i+1} = y_i + h(k_1 + k_2)$$

Vi ser att felet är propotionell mot steg ( $h$ ). Fel  $\epsilon$



#### Ex:

$$y' = 10 \cdot y(1 - \frac{y}{1000}), \quad y(0) = 1, \quad h = 0.1$$

i. löser ut  $y'(t)$

$$y'(t) = 10 \cdot y(1 - y/1000)$$

ii. Heuns metod, uttryck för  $y_{i+1}$

$$y_{n+1} = y_n + h_n/2(k_1 + k_2)$$

iii. Beräknar

$$k_1 = f(t_0, y_0) = 10 \cdot 1(1 - \frac{1}{1000}) = 9.99$$

$$k_2 = f(t_1, y_0 + hk_1) = f(t_1, 1.999)$$

$$= 10 \cdot 1.999(1 - \frac{1.999}{1000}) = 19.99(0.998001) \approx 19.95$$

$$\Rightarrow y_1 = y_0 + h/2(k_1 + k_2) = 1 + 0.05(k_1 + k_2)$$

$$\approx 2.497$$

### Runge-Kutta (explicit metod)

Runge-Kutta är noggrann numerisk approximering, betydligt bättre än euler framåt/backåt. Man beräknar 4 lutningar och tar typ medelvärdet av dem.

Figure 13.1: Runge-Kutta

#### Ex:

$$y'(t) - y(t) - t = 0, \quad y(0) = 0, \quad h = 0.1, \quad \text{RK 1-steg}$$

i. löser ut  $y'(t)$

$$y'(t) = y(t) + t \Rightarrow f(t, y) = y + t$$

ii. Runge-Kuttas, uttryck för  $y_{i+1}$

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

iii. Beräknar

$$k_1 = f(t_0, y_0) = y_0 + t_0 = 0$$

$$k_2 = f(t_0 + h/2, y_0 + hk_1/2)$$

$$= f(0.05, 0) = 0.05$$

$$k_3 = f(t_0 + h/2, y_0 + hk_2/2) = f(0.05, 0.0025) = 0.0525$$

$$k_4 = f(t_0 + h, y_0 + hk_3) = f(0.1, 0.00525) = 0.10525$$

$$k = (k_1 + 2k_2 + 2k_3 + k_4)/6$$

$$= (0 + 2 * 0.05 + 2 * 0.0525 + 0.10525)/6 = 0.31025/6$$

$$y_1 = y_0 + h \cdot (0.31025/6) = 0.1 \cdot (0.31025/6) \approx 0.00517$$

### 13.3.2 Högre årdningens ODE

$$x''(t) = -\frac{x(t)}{(\sqrt{x(t)^2 + y(t)^2})^3}$$

$$y''(t) = -\frac{y(t)}{(\sqrt{x(t)^2 + y(t)^2})^3}$$

Låt  $\bar{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$

$u_1 = x, u'_1 = u_2$   
 $u_2 = x', u'_2 = -\frac{u_1}{(\sqrt{u_1^2 + u_2^2})^3}$   
 $u_3 = y, u'_3 = u_4$   
 $u_4 = y', u'_4 = -\frac{u_3}{(\sqrt{u_1^2 + u_2^2})^3}$

$\bar{u}' = \begin{pmatrix} u_2 \\ -\frac{u_1}{(\sqrt{u_1^2 + u_2^2})^3} \\ u_4 \\ -\frac{u_3}{(\sqrt{u_1^2 + u_2^2})^3} \end{pmatrix}$

Löser i matlab:

Ställer upp ODE i separat fil

```
function u_out = satellitODE(t, u)
```

```
u_out = [ u(2);
-(u(1)/(sqrt(u(1)^2 + u(2)^2))^3);
u(4);
-(u(3)/(sqrt(u(1)^2 + u(2)^2))^3) ];
```

Sedan kan man kalla med följande script

```
tidsintervall = [0, 100];
```

```
% satelitens position (x(0),y(0))
% satelitens hastighet (x'(0), y'(0))
u0 = [0; 22223; 17000; 100];
[t, u] = ode45(@satellitODE,
               tidsintervall, u0, odeset)
plot(t, u)
```

## 13.4 Analys

Styg mångre ädringar under kort tid. *Adaptivt steglängdsval* som ode15s och ode45. Anpassar

steglängd efter styvhetssteg.

### 13.4.1 Analys av metoder

ODE:  $y' = f(t, y)$

Metod:  $y_{i+1} = y_i + h \cdot \Psi_k(t_i, y_{i+1}, y_i)$

### 13.4.2 Konsistent

Metoden är konsistent med ode'n om:

$$\lim_{h \rightarrow 0} \Psi_h = \Psi_0(t_i, y_i) = f(t_i, y_i)$$

### 13.4.3 Rättstält

$$\|\bar{F}(x, \bar{u}) - \bar{F}(x, \bar{z}) \leq L \cdot \|\bar{u} - \bar{z}\|$$

$$L = \max \|J(x, \bar{u})\| \quad \text{-jacobianen}$$

### 13.4.4 Noggrannhetsordning

Låt  $\phi(0)$  vara en funktion som uppfyller ode'n och är tillräckligt deriverbar. Bilda lokala trunkteringsfelet

$$\tau = \phi(t_{i+1}) - (\phi(t_i) + h\Psi_h(\phi(t_{i+1}), \phi(t_i)))$$

och taylorutveckla kring  $t_i$  och låt  $h \rightarrow 0$ . Om  $\Psi \rightarrow C \cdot h^{P+1}$  (dominerande term) Så har metoden noggrannhetsordning P

$$\begin{aligned} \tau_h \rightarrow C \cdot h^{P+1} &\Leftrightarrow \frac{\phi(t_{i+1}) - \phi(t_i)}{h} \\ &+ \Psi_h(t_i, \phi(t_{i+1}), \phi(t_i)) \rightarrow Ch^P \end{aligned}$$

**Exemple: Visa nu att Euler bakåt (implicit Euler) har noggrannhetsordning 1**

Lokala trunkteringsfelet  $\tau$  för implicit Euler metod är:

$$\tau = y(t_{k+1}) - y(t_k) - hf(t_{k+1}, y(t_{k+1}))$$

Vi kan då ersätta  $f(t_{k+1}, y(t_{k+1}))$  med  $y'(t_{k+1})$  (ode'n) vilket ger:

$$\tau = y(t_{k+1}) - y(t_k) - hy'(t_{k+1})||$$

Sedan analyserar vi det lokala trunkteringsfelet genom taylorutveckla den implecita delen av uttryck vid punkt  $t = t_k$ . Där av så är  $t_{k+1} = t_k + h$

$$\begin{aligned}\tau &= y(t_{k+1}) - y(t_k) - hy'(t_{k+1}) \\ &= y(t_k) + hy'(t_k) + \frac{h^2}{2}y''(t_k) + O(h^3) \\ &\quad - y(t_k) - h(y'(t_k) + y''(t_k) + O(h^2)) \\ &= \left(\frac{h^2}{2} - h^2\right)y''(t_k) + O(h^3) = O(h^2)\end{aligned}$$

Därmed så är det lockala trunkteringsfelet  $O(h^2)$  och nogranhets ordningen  $p$  i  $\tau_h = Ch^{p+1} \Rightarrow p = 1$ .  
V:S:V

### 13.4.5 Stabilitet

Vi undersöker Stabilitet på testekvationen

$$y' = \lambda y \quad \text{där } \operatorname{re}(\lambda) \leq 0$$

**Euler framåt** har stabilitetsområdet

$$|1 + h\lambda| \leq 1$$

Reella  $\lambda$  ger  $-1 \leq 1 + \lambda h \Leftrightarrow h \leq \frac{-2}{\lambda}$

**Euler bakåt** har stabilitetsområdet

$$\frac{1}{|1 - \lambda h|} \leq 1 \text{ dvs ovillkorligt stabil}$$

**Exemple: Visa nu att Euler bakåt (implicit Euler) är ovillkorligt stabil**

Testekvationen ses som sådan:

$$\begin{aligned}y' &= \lambda \cdot y \\ y_{i+1} &= y_i + hf(t_{i+1}, y_{i+1})\end{aligned}$$

Euler bakåt:

$$\begin{aligned}y_{i+1} &= y_i + h \cdot f(t_{i+1}, y_{i+1}) \\ &= y_i + h\lambda \cdot y_{i+1}\end{aligned}$$

Vi bryter ut  $y_{i+1}$ :

$$\begin{aligned}y_{i+1} - h\lambda \cdot y_{i+1} &= y_i \\ y_{i+1} \cdot (1 - h\lambda) &= y_i \\ y_{i+1} &= \frac{1}{1 - h\lambda} \cdot y_i\end{aligned}$$

Då ser vi att vårt stabilitetsvillkor är följande:

$$\left| \frac{1}{1 - h\lambda} \right| \leq 1$$

Vi antar (enligt uppgift) att  $\lambda$  är reellt och att  $\operatorname{Re}(\lambda) \leq 0$ . Detta medför att följande alltid kommer vara mindre än 1. Nämndare blir större än täljaren för alla värden på  $\lambda$ .

$$1 - h\lambda \geq 1 \Rightarrow \left| \frac{1}{1 - h\lambda} \right| \leq 1$$

för alla våra värden på  $\lambda$ . På grund av detta vet vi att oavsett värde på  $h$  så kommer den numeriska lösningen vara stabil, d.v.s ovillkorligt stabil.

### 13.4.6 Stabilitet generella ekvationer

Betrakta små rörelser av  $y$  (momentant)

$$\begin{aligned}y &= \tilde{y} + z \quad z\text{-litet, } \tilde{y} - \text{konstant} \\ z' &= f(t, \tilde{y}) \approx f(t, \tilde{y}) + z \cdot \frac{\delta f}{\delta y}(t, \tilde{y}) = C_1 + C_2 z\end{aligned}$$

Sats om den numeriska metoden är stabil för alla

$$C_2 = \frac{\delta f}{\delta y}(t, \tilde{y})$$

i lösningsområdet på testekvationen ( $\lambda = C_2$ )

### 13.4.7 Stabilitet för system

$$\bar{u} = F(t, \bar{u}) \quad \bar{u} = F(t, \bar{u}) \begin{bmatrix} f_1(t, \bar{u}) \\ f_2(t, \bar{u}) \\ \vdots \\ f_3(t, \bar{u}) \end{bmatrix}$$

Betrakta små rörelser (mementant)

lätt  $\bar{u} = \tilde{u} + \bar{z} \approx F(t, \tilde{u}) + F'(t, \tilde{u})\bar{z}$

$$\text{där } F'(t, \tilde{u}) = \begin{bmatrix} \frac{\delta f_1}{\delta u_1} & \frac{\delta f_1}{\delta u_2} & \cdots & \frac{\delta f_1}{\delta u_n} \\ \frac{\delta f_2}{\delta u_1} & \frac{\delta f_2}{\delta u_2} & \cdots & \frac{\delta f_2}{\delta u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta f_n}{\delta u_1} & \frac{\delta f_n}{\delta u_2} & \cdots & \frac{\delta f_n}{\delta u_n} \end{bmatrix}$$

$F(t, \tilde{u})$ -konstantlägre odningens term påverkar ej stabiliteten

Studera:  $\bar{z}' = A\bar{z}$  där  $A = F'(t, \tilde{u})$  (konstant matris) Låt  $\bar{w} = s^{-1}\bar{z} \Leftrightarrow \bar{z} = S\bar{w} \Rightarrow S\bar{w}' = AS\bar{w} \Leftrightarrow \bar{w}' = S^{-1}AS\bar{w}$  där  $D = S^{-1}AS$ . Vi väljer  $S$  så att  $D$  blir diagonal med  $A$ 's egenvärde i diagonalen,  $\bar{w}' = Dw$

$$\Rightarrow \begin{cases} w'_1 = \lambda_1 w_1 \\ w'_2 = \lambda_2 w_2 \\ \vdots \\ w'_n = \lambda_n w_n \end{cases}$$

sats: Om den numeriska metoden är stabil för  $w'_i = \lambda_i w_i$ , alla  $\lambda_i = \lambda(t, \tilde{u})$  i lösningsområdet så är den stabil för  $\bar{u}' = F(t, \bar{u}) \Rightarrow$  Utanför stabilitetsanalysen på testekvation  $y' = \lambda y$  men med  $\lambda_i(t, \tilde{u})$  (egenvärderna till jacobianen)

Not:  $\lambda$  - kan vara komplext (även om  $\bar{u}$  och  $F(t, \bar{u})$  är reella)

## 13.5 Stokastiska metoden

- Deterministisk metoder: Det vi tidigare kallat på. Får samma svar för sama input
- Stokastiska metoder: Varierar för same input, då slumpen spiller roll.

### 13.5.1 Monte Carlo

```
Indata: N (antal försök)
for i=1:N
    Gör en stokastisk simulering
    resultat(i)=resultat av simulering ovan
```

```
end
slutresultat = mean(resultat)
```

*Ensemble-prognoser* är att man kör flertalet gånger och kan ta medelvärdet för att få den troliga lösningen men sen också gemföra med och se hur stora fel/ osäkerhet är prognosen.

*Initialtilståndet*

Matlab har flera olika slumptals genererare

- `rand`: slumptal mellan 0 och 1
- `randn`: normalfördelade slumptal

Nogrannhetsordning Monte Carlo är obunden av denna metoder (vilket ex trapetsmetoden inte är).

Trapetsmetoden - Fel  $\sim O(n^{-2/d})$

MC - Fel  $\sim O(1/\sqrt{N})$

### 13.5.2 Invers Transform Sampling (ITS)

ITS är en algoritm för att generera slumptal ut ifrån en given fördelning.

Cumulative distribution function (CDF): is that you take the primitive function of a distribution. Man kan då bestämma  $\tau$  som är tiden för reaktion ska ske.

**Exempel: ITS**

$$f_\lambda(x) = \frac{1}{\lambda} e^{-x/\lambda}$$

1. Hitta primitiva funktionen

$$F_{\lambda(x)} = \int_0^x \frac{1}{\lambda} e^{-t/\lambda} dt = [-e^{-t/\lambda}]_0^x = 1 - e^{-x/\lambda}$$

2. Hitta inversen

$$F^{-1}: \text{Löser } y = 1 - e^{-x/\lambda}$$

$$y - 1 = -e^{-x/\lambda} \Rightarrow \ln(1 - y) = -x/\lambda$$

$$\Rightarrow x = -\lambda \ln(1 - y)$$

Vilket är exponentialfördelad

## Diskret slumptalsfördelning

Exempel: ITS Diskret slumptalsfördelning

- $$\begin{aligned}1: & \text{ Röd; } P_1 = 0.45 \\2: & \text{ Gul; } P_2 = 0.10 \\3: & \text{ Grön; } P_3 = 0.45 \\u = & 0.61\end{aligned}$$

ITS innebär i den diskreta versionen att  $r$ , numret på nästa utfall, ska väljas som det minsta tal  $x$  så att  $F(x) \leq u$ , där  $F(x)$  är den kumulativa fördelningsfunktionen. Med de givna värdena kan  $F(x)$  representeras av vektorn

$$F = \begin{bmatrix} 0.45 \\ 0.45 + 0.10 \\ 0.45 + 0.10 + 0.45 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0.55 \\ 1.00 \end{bmatrix}$$

Vi ser att  $F(3)$  är den första som är större än 0.61. Slutssatsen är att det kommer då vara  $r = 3$ , färgen grön som slumpas fram.

### 13.5.3 Gillespie algorithm/Stochastic Simulation Algorithm (SSA)

Simulering av reaktioner oftast i kemiska kontekst.

## Exemple

$$\begin{aligned} S'(t) &= -\beta \frac{I(t)}{N} S(t) \\ I'(t) &= \beta \frac{I(t)}{N} S(t) - \gamma I(t) \\ R'(t) &= \gamma I(t) \end{aligned}$$

*Reaktioner:*

$$\begin{aligned} r_1: & \quad \beta \frac{I(t)}{N} S(t) \\ r_2: & \quad \gamma I(t) \end{aligned}$$

*Reaktionsmatrix:*

$$\begin{matrix} r_1 \\ r_2 \end{matrix} \begin{bmatrix} S(t) & I(t) & R(t) \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} = R$$

Då har vi även reaktionshastighets-vektorn som följande.

Samt att  $C = \{\beta, \gamma, N\}$ .

$$\mathbf{P}(\tilde{\mathbf{y}}, \mathbf{C}) = \begin{bmatrix} C(1) \frac{y(1)}{C(3)} y(2) \\ C(2)y(1) \end{bmatrix}$$

I matlab:

## 13.6 Ordlista

- *Adaptiv steglängd*: Att den numeriska metoden i varje steg automatiskt ställer in steglängden så att det uppskattade felet blir lägre än toleransen.
  - *Cumulative distribution function*: du finner primitiva funktionen av distributionen.
  - *Deterministisk metoder*: En algoritm utan slumpmoment, så att utdata beror endast på indata.
  - *Deterministisk modell*: En modell utan slumpmoment, så att utdata beror endast på indata.
  - *Diskretiseringfel*: Vid exemplet när man beräknar derivatan så är domineras Diskretiseringfel
  - *Explicit metod*: En metod för numerisk lösning av ODE, där högerledet i metoden bara beror på kända värden.
  - *Global trunkteringsfelet*: Det sammanlagda felet i från det tidigare stegen med den numeriska metoden utgående från exakta lösningen i senaste tidpunkten.
  - *Implcit metod*: En numerisk metod för lösning av ODE, där högerledet i metoden innehåller  $y_{i+1}$ , så att man måste lösa en ickelineär ekvation i varje steg.
  - *ITS*: en algoritm för att generera slumptal ut ifrån en given fördelning
  - *Konsistent*: Den numeriska metoden går mot differentialekvationen då  $h$  går mot noll.
  - *Kancellation*: Förlust av signifikanta siffror vid subtraktion mellan jämnstora flyttal.
  - *Konvergens*: Den numeriska lösningen går mot den exakta lösningen då steglängden  $h$  går mot noll.
  - *lockala trunkteringsfelet*: Felet i ett steg med den numeriska metoden utgående från exakta lösningen i senaste tidpunkten.
  - *Mantissa*: bråkdelen i floating point

- *Maskinepsilon*: Det relativa fel man får då tal lagras i en dator (eller när beräkningar utförs) beror på hur talen lagras internt i datorn.
- *Monte Carlo-metod*: Upprepad stokastisk simulering samt statistik på de samlade resultaten.
- *Normalisering*: Konventionen att i flyttalsrepresentation ha precis en nollskild siffra före "decimalpunkten" (så att flyttalsrepresentationen blir entydig).
- *Noggrannhetsordning*: h-potensen i globala trunkeringsfelet hos en numerisk metod för lösning av ODE (där h är steglängden).
- *Overflow*: Motsatsen till underflow.
- *Stokastiska modell*: En modell med slumpmo-  
ment, så att utdata inte beror entydigt på in-data.
- *Styv*: En ODE för vilken en explicit differensmetod behöver ta mycket kortare steg för stabilitet än vad som skulle krävas för tillräcklig noggrannhet.
- *Stabilitet*: Störningskänsligheten hos den numeriska metoden.
- *SSA/Gillespie algoritm*:
- *Underflow*: Det som uppstår när vi försöker representera ett tal vars absolutbelopp är mindre än det till beloppet minsta normaliserade flyttalet.
- *jacobianen*



## Chapter 14

# Signal and Transforms

## 14.1 Introduction

### 14.1.1 System Classification

- *Continuous-time*: Defined for all times in a time period  $x(t)$
- *Discrete-time*: Defined for some times in a time period  $x[k]$
- *Analog*: Continuous time signal time-varying
- *Digital*: Discrete time signal
- *Linear/ Non-linear*: if  $\alpha x_1(t) + \beta x_2(t) \mapsto \alpha y_1(t) + \beta y_2(t)$  (superposition)
- *Periodic/ Aperiodic*:  $x(t) = x(t + T)$
- *Causal*: time dependent on previous not next values
- *time-variant/ time-invariant*  $x(t - T) \mapsto y(t - T)$
- *Deterministic* Same result each time no random.
- *stochastic* random signals (won't look at)
- *Stable/ unstable* bounded  $\lim_{t \rightarrow \infty} x(t) < \infty$
- *Invertible* Can invert
- *SISO* (single input single output)
- *MIMO* (multiple input multiple output)
- *BIBO* (bounded input-bounded output) stable/unstable

<b>Linear, time invariant (LTI)</b>	Linear, time variant
non-linear, time invariant (LTI)	non-linear, time variant

We only look at LTI only.

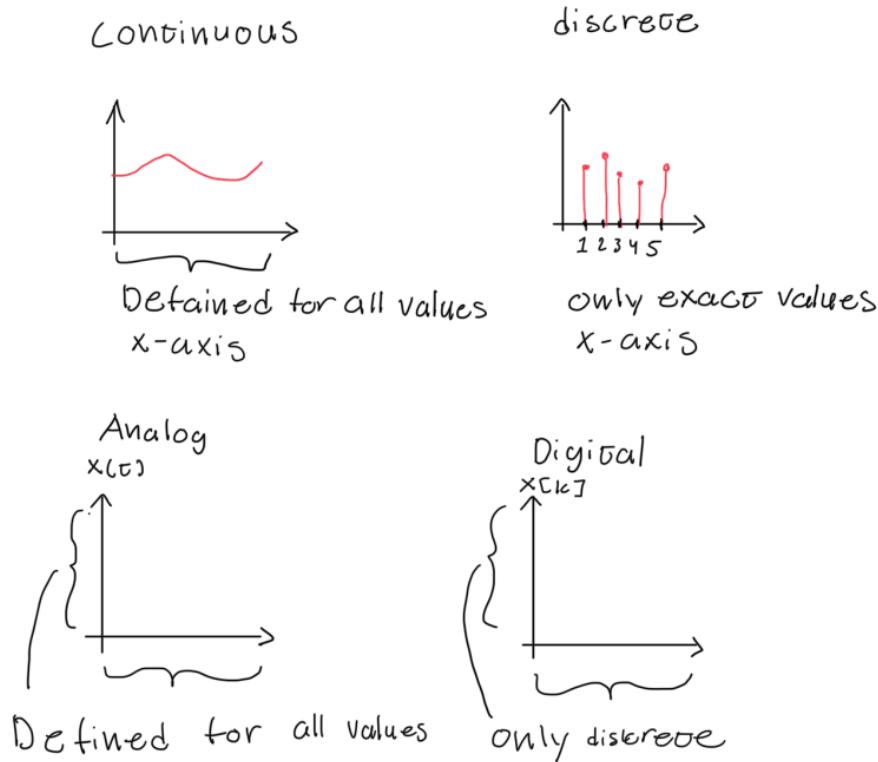


Figure 14.1: Signal plots

Any signal can be expressed by its even and odd component

$$x(t) = x_e(t) + x_o(t) \quad (14.1)$$

*Sampling:* continuous time signal.

$$x[k] \triangleq x(kT_s) \quad (14.2)$$

$T_s$  is the sampling time. The discrete signals at time  $k$  equals to the continuous time signal  $x(t)$  at time  $t = kT_s$ .

$$x(t) \mapsto y(t) \quad (14.3)$$

$$x[t] \mapsto y[t] \quad (14.4)$$

### Transform methods

A transform is an operator  $T\{\cdot\}$  from one domain to another.

$$X(\varsigma) = T\{x(t)\}$$

$x(t)$  and  $X(\varsigma)$  are a *transform pairs*

$$x(t) \circledast X(\varsigma)$$

The inverse transform  $T^{-1}\{\cdot\}$  restores the original signal.

$$x(t) = T^{-1}\{X(\varsigma)\}$$

### Example: Linear

Is  $\frac{dy(t)}{dt} + 3ty(t) = t^2x(t)$  linear? Assume zero initial conditions.

#### Solution:

$$\begin{aligned} \frac{dy(t)}{dt} + 3ty(t) &= t^2x(t) \\ \frac{d(\alpha y_1(t) + \beta y_2(t))}{dt} + 3t(\alpha y_1(t) + \beta y_2(t)) &= t^2(\alpha y_1(t) + \beta y_2(t)) \\ &= t^2(\alpha x_1(t) + \beta x_2(t)) \\ \alpha \frac{dy_1(t)}{dt} + \beta \frac{dy_2(t)}{dt} + \alpha 3ty_1(t) + \beta 3ty_2(t) &= \alpha t^2x_1(t) + \beta t^2x_2(t) \end{aligned}$$

therefore is the system linear.

**Example: Time invariant**

Is  $y(t) = tx(t - 2)$  time invariant?

**Solution:**

$$y(t - \tau) = tx((t - \tau) - 2) \neq (t - \tau)x((t - \tau) - 2)$$

therefore is the system time variant.

**Example: Casual**

Is  $y(t) = x(-t)$  casual?

**Solution:** When  $t < 0$  then  $t < -t$  with is not allowed in a causal system since it then depends on upcoming values. If we were to restrict that  $t \geq 0$  then it would be causal.

**Euler's identities**

$$e^{+jx} = \cos x + j \sin x$$

$$e^{-jx} = \cos x - j \sin x$$

$$\sin(\alpha) = \frac{e^{j\alpha} - e^{-j\alpha}}{2j}$$

$$\cos(\alpha) = \frac{e^{j\alpha} + e^{-j\alpha}}{2}$$

$$f_0 = \frac{1}{T_0}, w = 2\pi f_0, w_0 = \frac{2\pi}{T_0}$$

## 14.2 Continuous-time signals and time-domain analysis

### 14.2.1 Continuous Time Signals

- **Sine:**  $A \sin(\omega_0 t + \varphi)$ 
  - $\omega$  is *angular frequency*  $\omega_0 t = 2\pi f_0$  fpr *frequency domain*
  - $t$  is time since it is in *time domaian*
  - $\varphi$  is the *phase*
  - $\sin(\omega_0 t) = \frac{e^{j\omega_0 t} - e^{-j\omega_0 t}}{2j}$
- **Cosine:**  $A \cos(\omega_0 t + \varphi)$ 
  - $\cos(\omega_0 t) = \frac{e^{j\omega_0 t} + e^{-j\omega_0 t}}{2}$
- **Dirac Delta function:**  $\delta(t)$ 
  - $\delta(t) = 0$  for  $t \neq 0$

- $\int_{-\infty}^{\infty} \delta(t) dt = 1$
- $\int_{-\infty}^{\infty} x(t)\delta(t) dt = x(0)$
- $\int_{-\infty}^{\infty} x(t)\delta(t - t_0) dt = x(t_0)$

- **Rectangular pulse:**  $\text{rect}\left(\frac{t}{T_0}\right)$

- **Unit step:**  $u(t) = \begin{cases} 0 & \text{for } t < 0 \\ 1 & \text{for } t \geq 0 \end{cases}$

- **Sinc function:**  $\text{sinc}\left(\frac{t}{T_0}\right) = \frac{\sin\left(\frac{\pi t}{T_0}\right)}{\frac{\pi t}{T_0}}$

- **Periodic Signals:**  $x(t) = x(t + T_0)$

- **Energy Signals:**  $E = \int_{-\infty}^{\infty} |x(t)|^2 dt$

- **Power Signals:**  $P = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{T/2}^{-T/2} |x(t)|^2 dt$

- **Orthogonal Signals:**  $\langle x, y \rangle = x^T y = \sum_i x_i y_i$

**Signal Caracteristic**

- **Causal:**  $x(t) = 0$  for  $t < 0$
- **Even:** (flip on y-axis)  $x(t) = x(-t)$
- **Odd:** (flip on x-axis and y-axis)  $x(t) = -x(-t)$

**Signal operations**

- **Amplitude scaling** (Increase/decrease the amplitude value):  $y(t) = \alpha x(t)$
- **Offsetting** (Moves signal in y-direction):  $y(t) = x(t) + y_0$
- **Time shifting** (Move signal in x-direction):  $y(t) = x(t - \Delta t)$
- **Time scaling and inversion** (Increase/decrease width):  $y(t) = x(\alpha t)$

### 14.2.2 Time Domain Analysis of Continuous Time Systems

**Main properties**

$\delta(t)$  is even function

$$\delta(at + b) = \frac{1}{|a|} \delta\left(t + \frac{b}{a}\right)$$

$$\phi(t)\delta(t - \tau) = \phi(\tau)\delta(t - \tau)$$

$$\int_{-\infty}^{\infty} \phi(\tau)\delta(\tau - t) dt = \phi(t)$$

$$\text{Convolution: } (x * h)(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau) dt$$

$$\Rightarrow x(t) * \delta(t - T) = x(t - T)$$

**Impulse response**

Definition

$$x(t) = \delta(t) \mapsto y(t) = h(t)$$

Properties

*Impulse response:  $h(t)$* 

$\delta(t) \mapsto h(t)$  since LTI system with continuos time  
 $\alpha\delta(t - T) \mapsto \alpha h(t - T)$  since LTI

**Convolution integral**

Convolution integral. For any input  $x(t)$ , the output of an LTI system is (*convolution*):

$$y(t) = x(t) * h(t) = h(t)$$

Definition of convolution:

$$\int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau = \int_{-\infty}^{\infty} h(\tau)x(t - \tau)d\tau \quad (14.5)$$

For causal systems ( $h(t) = 0$  for  $t < 0$ ):

$$\int_0^{\infty} x(\tau)h(t - \tau)d\tau = \int_0^{\infty} h(\tau)x(t - \tau)d\tau \quad (14.6)$$

## Example: Convolution 1

$$\begin{aligned} \sin(t)u(t) * u(t) &= \int_{-\infty}^{\infty} \sin(\tau)u(\tau)u(t - \tau)d\tau \\ &= \int_0^t \sin(\tau)d\tau = [-\cos(\tau)]_0^t \\ &= -\cos(t) + \cos(0) = 1 - \cos(t) \end{aligned}$$

## Example: Convolution 2

$$\begin{aligned} Y(\omega) &= \frac{1}{2\pi} X(\omega) * \pi[\delta(\omega - 25) + \delta(\omega + 25)] \\ &= \frac{1}{2}[X(\omega) * \delta(\omega - 25) + X(\omega) * \delta(\omega + 25)] \\ &\quad \text{-(Constants can be moved out)} \\ &= \frac{1}{2}[X(\omega - 25) + X(\omega + 25)] \\ &\quad \text{-(Convolution of direct delta is just time shift)} \end{aligned}$$

**Sin In, Sine Out Principle**

The frequency dose not change.

$$x(t) = A_x \sin(\omega_0 t - \varphi_x) \mapsto y(t) = A_y \sin(\omega_0 t - \varphi_y) \quad (14.7)$$

Amplitude and phase reletionships ( $H(\omega_0)$ : complex number):

$$A_y = |H(\omega_0)|A_x \quad \text{and} \quad \varphi_y = \varphi_x + \angle H(\omega_0) \quad (14.8)$$

**14.3 Continuous Time Fourier Series and Transform****14.3.1 Continuous Time Fourier Series****Trigonometric continuos time Fourier series**

Any periodic continuos time signal can be expressed as

$$x(t) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)]$$

$$\begin{aligned} a_0 &= \frac{1}{T_0} \int_{(T_0)} x(t)dt \\ a_n &= \frac{2}{T_0} \int_{(T_0)} x(t) \cos(n\omega_0 t)dt \\ b_n &= \frac{2}{T_0} \int_{(T_0)} x(t) \sin(n\omega_0 t)dt \end{aligned}$$

**Exponential continuos time Fourier series**

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t}$$

$$\begin{aligned} \omega_0 &= \frac{2\pi}{T_0} \\ c_n &= \frac{1}{T_0} \int_{(T_0)} x(t) e^{-jn\omega_0 t} dt \end{aligned}$$

**Coefficients relationships of Exponential and trigonometric Fourier series**

$$\begin{aligned} c_0 &= a_0, \quad c_n = \frac{a_n - jb_n}{2}, \quad c_{-n} = \frac{a_n + jb_n}{2}, \quad \text{for } n \geq 1 \\ a_0 &= c_0, \quad a_n = 2Re\{c_n\}, \quad b_n = -2Im\{c_n\}, \quad \text{for } n \geq 1 \\ &\quad \text{Complex conjugate pair } c_n = c_{-n}^* \end{aligned}$$

- (Trigonometric) If  $x(t)$  is an even signal,  $b_n = 0$

- (Trigonometric) If  $x(t)$  is an odd signal,  $a_n = 0$
- (Exponential) If  $x(t)$  is an even signal,  $c_n$  is purely real
- (Exponential) If  $x(t)$  is an odd signal,  $c_n$  is purely imaginary

### 14.3.2 Continuos Time Fourier Transforms

For Aperiodic we use Fourier transform

$$X(\omega) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$

$$x(t) = \mathcal{F}^{-1}\{X(\omega)\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)e^{j\omega t} d\omega$$

#### Spectral Analysis

Analysing figure of magnitude  $|X(\omega)|$  and the phase  $\angle X(\omega)$ . Phase plot is very important.

The magnitude plot shows us what frequencies is dominant and not. It is even. One can also determin the output from an input signal from the spectrum since from the magnitude you can do times that of the input signal. The phase plot determines the phase shift for a frequency of the input signal  $w_0$ .

#### Phasor operations

addition:  $z_1 + z_2 = (x_1 + x_2) + j(y_1 + y_2)$

subtraction:  $z_1 - z_2 = (x_1 - x_2) + j(y_1 - y_2)$

multiplication:  $z_1 z_2 = r_1 r_2 / (\phi_1 + \phi_2)$

division:  $\frac{z_1}{z_2} = \frac{r_1}{r_2} / (\phi_1 - \phi_2)$

inverse:  $\frac{1}{z} = \frac{1}{r} / (-\phi)$

square root:  $\sqrt{z} = \sqrt{r} / (\phi/2)$

complex conjugate:  $z^* = x - jy$

#### Properties

##### • Linearity:

$x_1(t) \circledast X_1(\omega)$ , and  $x_2(t) \circledast X_2(\omega) \Rightarrow \alpha x_1(t) + \beta x_2(t) \circledast \alpha X_1(\omega) + \beta X_2(\omega)$

##### • Duality:

$X(t) \circledast 2\pi x(-\omega)$

##### • Time and frequency scaling:

$x(at) \circledast \frac{1}{|a|} X\left(\frac{\omega}{a}\right)$  for  $a \neq 0$

##### • Time and frequency shifting:

$x(t - \Delta t) \circledast X(\omega)e^{-j\omega\Delta t} \Rightarrow x(t)e^{j\omega_0 t} \circledast X(\omega - \omega_0)$

##### • Convolution and multiplication:

$$x_1(t) * x_2(t) \circledast X_1(\omega)X_2(\omega) \Rightarrow x_1(t) * x_2(t) \circledast X_1(\omega)X_2(\omega)$$

##### • Differentiation and integration:

$$\frac{d^n x(t)}{dt^n} \circledast (j\omega)^n X(\omega) \text{ where } n \text{ is for derivatives}$$

$$\Rightarrow \int_{-\infty}^t x(\tau)d\tau \circledast \frac{1}{j\omega} X(\omega) + \pi X(0)\delta(\omega)$$

##### • Symmetries:

$$X^*(\omega) = X(-\omega)$$

–  $|X(\omega)|$  is even

–  $\angle X(\omega)$  is odd

–  $\operatorname{Re}\{X(\omega)\}$  is even

–  $\operatorname{Im}\{X(\omega)\}$  is odd

##### • Periodicity:

– If  $x(t)$  is periodic,  $X(\omega)$  is discrete

– If  $x(t)$  is discrete,  $X(\omega)$  is periodic

##### • Parseval's theorem:

$$W = \int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\omega)|^2 d\omega$$

$$P = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} |x(t)|^2 dt = \sum_{n=-\infty}^{\infty} |c_n|^2$$

#### Fourier Transforms of Elementary Signals

##### • Dirac delta:

$$X(\omega) = \int_{-\infty}^{\infty} \delta(t)e^{j\omega t} dt = e^{-j\omega 0} = 1$$

$$x(t) = \delta(t) \circledast X(\omega) = 1$$

##### • Constant: (Duality property)

$$X(\omega) = 2\pi\delta(-\omega)$$

$$x(t) = 1 \circledast X(\omega) = 2\pi\delta(\omega)$$

##### • Rectangular pulse:

$$X(\omega) = \int_{-\infty}^{\infty} \operatorname{rect}\left(\frac{t}{T_0}\right) dt = \frac{e^{-j\frac{\omega T_0}{2}} - e^{j\frac{\omega T_0}{2}}}{-j\omega}$$

$$X(\omega) = \frac{2 \sin(\frac{\omega T_0}{2})}{\omega} = T_0 \operatorname{sinc}\left(\frac{\omega T_0}{2\pi}\right)$$

$$x(t) = \operatorname{rect}\left(\frac{t}{T_0}\right) \circledast X(\omega) = T_0 \operatorname{sinc}\left(\frac{\omega T_0}{2\pi}\right)$$

##### • Sinc function:

$$\operatorname{sinc}\left(\frac{t}{T_0}\right) = X_{\operatorname{rect}}\left(\frac{2\pi t}{T_0}\right)$$

$$\begin{aligned} -x(t) &= \text{sinc}\left(\frac{t}{T_0}\right) \quad \bullet \quad X(\omega) = \\ &T_0 \text{rect}\left(\frac{\omega T_0}{2\pi}\right) \end{aligned}$$

- **Periodic signals:** It's fourier transform becomes discrete even if the input signal is continuous

$$\begin{aligned} -X(\omega) &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t} e^{-j\omega t} dt = \\ &2\pi \sum_{n=-\infty}^{\infty} c_n \delta(\omega - n\omega_0) \end{aligned}$$

### 14.3.3 Frequency Domain Analysis of Continuous Time Systems

Frequency domain input-output relationship

$$Y(\omega) = H(\omega)X(\omega)$$

-(Since in the time domain  $y(t) = x(t) * h(t)$ )

Where  $Y(\omega)$  is the fourier transform LTI systems output signal,  $H(\omega)$  is the *frequency response* and  $X(\omega)$  is the input signals Fourier transform

$$H(\omega) = \int_{-\infty}^{\infty} h(t) e^{-j\omega t} dt$$

Polar form of Frequency domain input-output relationship

$$H(\omega) = |H(\omega)|e^{j\angle H(\omega)}$$

$$Y(\omega) = |X(\omega)||H(\omega)|e^{j(\angle X(\omega) + \angle H(\omega))}$$

Example: find output signal

What is the output signal  $y(t)$  if the input signal is  $x(t) = 2 \cos\left(\frac{\pi}{2}t\right)$  and the impulse response is  $h(t) = \text{rect}\left(\frac{t-1}{2}\right)$ ?

**Solution:**

$$\begin{aligned} x(t) &= 2 \cos\left(\frac{\pi}{2}t\right) \mapsto y(t) \\ &= 2|H\left(\frac{\pi}{2}\right)| \cos\left(\frac{\pi}{2} + \angle H\left(\frac{\pi}{2}\right)\right) \end{aligned}$$

$$H(\omega) = \mathcal{F}\{h(t)\} = \mathcal{F}\{\text{rect}\left(\frac{t-1}{2}\right)\}$$

$$= 2e^{-j\omega} \text{sinc}\left(\frac{2\omega}{2\pi}\right)$$

$$\Rightarrow H\left(\frac{\pi}{2}\right) = 2e^{-j\frac{\pi}{2}} \text{sinc}\left(\frac{2\frac{\pi}{2}}{2\pi}\right)$$

$$= 2e^{-j\frac{\pi}{2}} \text{sinc}\left(\frac{1}{2}\right) \approx 1.3e^{-j\frac{\pi}{2}}$$

$$\Rightarrow |H\left(\frac{\pi}{2}\right)| = 1.3, \angle H\left(\frac{\pi}{2}\right) = -\frac{\pi}{2}$$

Therefore is the output signal:

$$\begin{aligned} y(t) &= 2 \cdot 1.3 \cos\left(\frac{\pi}{2}t - \frac{\pi}{2}\right) \\ &= 2.6 \cos\left(\frac{\pi}{2}t - \frac{\pi}{2}\right) \end{aligned}$$

## 14.4 Laplace transform

General version of fourier transform. One issue with Fourier transform is that there could be undefined values. Therefor we introduce an exponential expression in order for it to converge quickly enuff. The definition *unitary laplace transform* is:

$$X(s) = \mathcal{L}\{x(t)\} = \int_0^{\infty} x(t) e^{st} dt$$

$$x(t) = \mathcal{L}^{-1}\{X(s)\} = \frac{1}{2\pi j} \int_{\sigma-j\omega}^{\sigma+j\omega} X(s) e^{st} ds$$

$X(s)$  is the unilateral laplace transform

$x(t)$  is a continuous time signal

$s = \sigma + j\omega$  is a complex variable

$\sigma$  is dampening factor

$\omega$  is the frequency

*Region of conversions* (ROC) is needed for the laplace transform to exist. In other words, that the integral

over  $x(t)e^{st}$  is finite and a transform exists

**Example:** Convert transfer function to impulse response

Determine is the impulse response of:

$$H(s) = \frac{1}{(s+2)(s+3)}$$

**Solution:** To determine the impulse response from the transfer function do we need to do the inverse laplace transfer of the transfer function.

$$\begin{aligned} H(s) &= \frac{c_1}{s+2} \frac{c_2}{s+3} = \frac{c_1(s+3)}{s+2} + \frac{c_2(s+2)}{s+3} \\ &= \frac{c_1s + 3c_1 + c_2s + 2c_2}{(s+2)(s+3)} \end{aligned}$$

Therefore do we get the following system of equations

$$\begin{cases} c_1 + c_2 = 0 \Rightarrow c_1 = -c_2 \\ 3c_1 + 2c_2 = 1 \end{cases}$$

Hence, the constants are  $c_1 = 1$  and  $c_2 = -1$ .

$$\begin{aligned} H(s) &= \frac{1}{s+2} - \frac{1}{s+3} \\ h(t) &= e^{-2t}u(t) - e^{-3t}u(t) \quad \text{-(transfer table)} \end{aligned}$$

Time domain $x(t)$	Laplace transform $X(s)$	ROC
<b>Dirac delta function</b>		
$x(t) = \delta(t)$	$X(s) = 1$	All $s$
<b>Unit step</b>		
$x(t) = u(t)$	$X(s) = \frac{1}{s}$	$\text{Re}\{s\} > 0$
<b>Exponential</b>		
$x(t) = e^{-at}u(t)$	$X(s) = \frac{1}{a+s}$	$\text{Re}\{s\} > -a$
<b>Ramp</b>		
$x(t) = tu(t)$	$X(s) = \frac{1}{s^2}$	$\text{Re}\{s\} > 0$
<b>Higher order ramp</b>		
$x(t) = t^n u(t)$	$X(s) = \frac{n!}{s^{n+1}}$	$\text{Re}\{s\} > 0$
<b>Cosine</b>		
$x(t) = \cos(\omega_0 t)u(t)$	$X(s) = \frac{s}{\omega_0^2 + s^2}$	$\text{Re}\{s\} > 0$
<b>Sine</b>		
$x(t) = \sin(\omega_0 t)u(t)$	$X(s) = \frac{\omega_0}{\omega_0^2 + s^2}$	$\text{Re}\{s\} > 0$
<b>Decaying cosine</b>		
$x(t) = e^{-at} \cos(\omega_0 t)u(t)$	$X(s) = \frac{a+s}{(a+s)^2 + \omega_0^2}$	$\text{Re}\{s\} > -a$
<b>Decaying sine</b>		
$x(t) = e^{-at} \sin(\omega_0 t)u(t)$	$X(s) = \frac{\omega_0}{(a+s)^2 + \omega_0^2}$	$\text{Re}\{s\} > -a$

Figure 14.2: Unilateral Laplace transform pairs. From

#### 14.4.1 Properties

- **Linjarity:**  $\alpha x_1(t) + \beta x_2(t) \rightleftharpoons \alpha X_1(s) + \beta X_2(s)$
- **Time and frequency scaling:**  $x(at) \rightleftharpoons \frac{1}{a}X(\frac{s}{a})$  for  $a > 0$
- **Time and frequency shifting:**
  - $x(t - \Delta t) \rightleftharpoons X(s)e^{-s\Delta t}$
  - $x(t)e^{at} \rightleftharpoons X(s - a)$
- **Convolution and multiplication:**
  - $x_1(t) * x_2(t) \rightleftharpoons X_1(s)X_2(s)$
  - $x_1(t)x_2(t) \rightleftharpoons \frac{1}{2\pi j}X_1(s) * X_2(s)$

- Differentiation and integration:  $\frac{d^n x(t)}{dt^n} \leftrightarrow sX(s) - x(t)$

- Initial and final value theorems:

$$\begin{aligned} - \lim_{t \rightarrow 0} x(t) &= \lim_{s \rightarrow \infty} sX(s) \\ - \lim_{t \rightarrow \infty} x(t) &= \lim_{s \rightarrow 0} sX(s) \end{aligned}$$

### 14.4.2 Transfer function

#### Solving differential equations

Can solve Differential equations. Use Laplace transforms then do the inverse transform. The analyzing system is useful to use transform. It is done the same way as for solving differential equations. This works for all dimensions.

$$\begin{aligned} H(s) &= \frac{Y(s)}{X(s)} = \frac{s^M b_M + s^{M-1} b_{M-1} + \dots + b_0}{s^N b_N + s^{N-1} b_{N-1} + \dots + a_0} \\ Y(s) &= H(s)X(s) \end{aligned}$$

#### Transfer function and impulse response

$$\begin{aligned} h(t) &\leftrightarrow H(s) \\ H(s) &= \int_0^\infty h(t)e^{-st} dt \end{aligned}$$

#### Transfer function and frequency response

The relationship between the transfer function  $H(s)$  and the frequency response  $H(j\omega)$  of a continuous time LTI system is given by

$$H(j\omega) = H(s)|_{j\omega}$$

#### Poles and Zeros, and Stability

$$\begin{aligned} H(s) &= K \frac{(s - z_1)(s - z_s) \dots (s - z_M)}{(s - p_1)(s - p_2) \dots (s - p_N)} \\ \text{where } z_M &\text{ is zeros} \\ \text{and } p_N &\text{ is poles} \end{aligned}$$

The system is proper if  $N \leq M$  and strictly proper if  $N < M$ . It is stable if all  $p_i < 0$ .

If  $p_i = 0$  then the output will never convert just alternate to infinity (unstable).  
If  $p_i > 0$  then it will increase its amplitude (unstable).

LTI system is stable if and only if the real part of all of its poles is negative.

*Steady-state* is when amplitude is  $A = |H(s)|_{s=j\omega_0}$  and phase  $\phi = H(s)_{s=j\omega_0}$

The poles tell us when the transfer function  $H(s)$  becomes a division by zero. The zeros meanwhile tells us when the function is zero. The Pole-Zero Map is a display from the top looking down where x is the marked when the expression goes to infinite and the zero is when the expression is zero.

### Example: Poles and Zeros

Consider a linear, time-invariant system described by the differential equation

$$\frac{d^2y(t)}{dt^2} + 6\frac{dy(t)}{dt} + 13y(t) = \frac{dx(t)}{dt} + 2x(t)$$

Assume that all initial conditions are zero. Determine whether or not the system is stable.

**Solution:** Determine the transfer function  $H(s)$

$$\begin{aligned}s^2Y(s) + 6sY(s) + 13Y(s) &= sX(s) + 2X(s) \\ Y(s)(s^2 + 6s + 13) &= X(s)(s + 2) \\ \frac{Y(s)}{X(s)} &= \frac{s + 2}{s^2 + 6s + 13} = H(s)\end{aligned}$$

We see that the system is proper since the degree of the numerator is less than the denominator.

Zeros:

$$\begin{aligned}s + 2 &= 0 \\ s = -2 \Rightarrow z_1 &= -2\end{aligned}$$

Poles:

$$\begin{aligned}s^2 + 6s + 13 &= 0 \\ s = -\frac{6}{2} \pm \sqrt{3^2 - 13} &= -3 \pm \sqrt{-4} = -3 \pm 2j \\ \Rightarrow p_{1,2} &= -3 \pm 2j\end{aligned}$$

Therefore we see that both poles are negative, therefore the system is stable.

*qualitative* impulse response.

### 14.4.3 Bode plot

Bode plots can relatively easily be constructed by hand, which was paramount before the computer era for analyzing complex dynamic systems.

Bode plots are a standardized way of plotting a system's frequency response function and a Bode plot actually consists of two plots: A Bode magnitude plot (or simply magnitude plot) and a Bode phase plot (or simply a phase plot).

Asymptote is the line where define the direction of the line.

$$|H(j\omega)|_{dB} = 20 \log_{10}(|H(j\omega)|)$$

### Bode Form of Transfer Function

$$\begin{aligned}
 H(s) &= K \frac{(s - z_1) \dots (s - z_M)}{s^l (s - p_1) \dots (s - p_{N-l})} \\
 H(s) &= K_0 \frac{\left(\frac{s}{z_1} + 1\right) \dots \left(\left(\frac{s}{\omega_{n,i}}\right)^2 + 2s\frac{\zeta_i}{\omega_{n,i}} + 1\right) \dots}{s^l \left(\frac{s}{p_1} + 1\right) \dots \left(\left(\frac{s}{\omega_{n,j}}\right)^2 + 2s\frac{\zeta_j}{\omega_{n,j}} + 1\right) \dots} \\
 K_0 &= K \frac{\prod_i (-z_i)}{\prod_i (-p_i)}
 \end{aligned}$$

There are  $l$  poles in the origin (the term  $s^{-l}$ ).

### Sketching Bode Plots

Example: ODE solving with transfer functions

Sketch Bode plots for the following transfer functions.

$$H(s) = \frac{s(s+100)}{(s+2)(s+20)}$$

**Solution:** First we write the transfer function as in bode normal Form

$$\begin{aligned} H(s) &= \frac{s(s+100)}{(s+2)(s+20)} = \frac{100s(\frac{s}{100} + 1)}{2 \cdot 20(\frac{s}{2} + 1)(\frac{s}{20} + 1)} \\ H(j\omega) &= \frac{100j\omega(\frac{j\omega}{100} + 1)}{2 \cdot 20(\frac{j\omega}{2} + 1)(\frac{j\omega}{20} + 1)} = \frac{100j\omega(1 + j\frac{\omega}{100})}{2 \cdot 20(1 + j\frac{\omega}{2})(1 + j\frac{\omega}{20})} \end{aligned}$$

$$\begin{aligned} \lim_{\omega \rightarrow 0} H(j\omega) &= 0, (-\infty dB) \\ \lim_{\omega \rightarrow \infty} H(j\omega) &= 1, (0dB) \end{aligned}$$

We can write the magnitude by adding the factor together

$$\begin{aligned} |H(j\omega)|_{dB} &= 20 \log_{10}\left(\frac{100}{2 \cdot 20}\right) + 20 \log_{10}(|j\omega|) - 20 \log_{10}(|1 + j\frac{\omega}{2}|) \\ &\quad - 20 \log_{10}(|1 + j\frac{\omega}{20}|) + 20 \log_{10}(|1 + j\frac{\omega}{100}|) \end{aligned}$$

The phase shift can be done similarly.

$$\begin{aligned} \angle \frac{100}{2 \cdot 20} &= 0 \\ \angle |j\omega| &= \frac{\pi}{2} \\ \angle \frac{1}{1 + j\frac{\omega}{2}} &= -\angle 1 + j\frac{\omega}{2} = \begin{cases} 0 & w = 0 \\ -\frac{\pi}{4} & w = 2 \\ -\frac{\pi}{2} & w \rightarrow \infty \end{cases} \\ \angle \frac{1}{1 + j\frac{\omega}{20}} &= -\angle 1 + j\frac{\omega}{20} = \begin{cases} 0 & w = 0 \\ -\frac{\pi}{4} & w = 20 \\ -\frac{\pi}{2} & w \rightarrow \infty \end{cases} \\ \angle 1 + j\frac{\omega}{100} &= \begin{cases} 0 & w = 0 \\ \frac{\pi}{4} & w = 100 \\ \frac{\pi}{2} & w \rightarrow \infty \end{cases} \end{aligned}$$

Now we can draw the bode plot see figure 14.3.

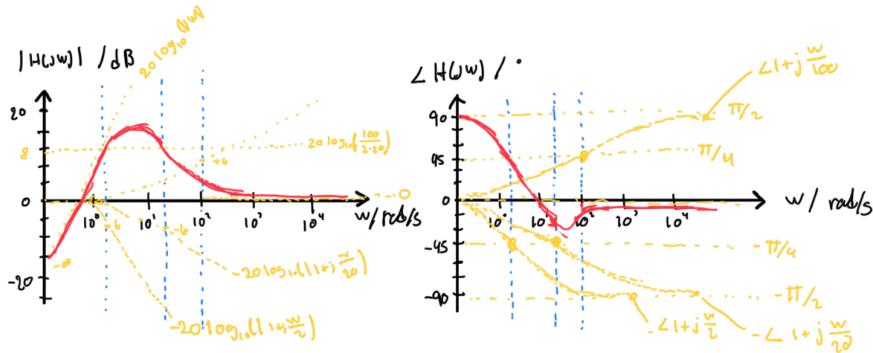


Figure 14.3: Bode plot example

## 14.5 Filtering theory

### 14.5.1 Filtering Theory and Ideal Filters

An ideal filter only contains a *passband* (the frequency range that the signal is not filtered out) and the *stopband* (the frequency which is filtered out)

- **Lowpass filter:**  $H_{LP}(\omega) = \text{rect}\left(\frac{\omega}{2\omega_c}\right)$
- **Highpass filter:**  $H_{HP}(\omega) = 1 - \text{rect}\left(\frac{\omega}{2\omega_c}\right)$
- **Bandpass filter:**  $H_{BP}(\omega) = \text{rect}\left(\frac{\omega}{2\omega_{c2}}\right) - \text{rect}\left(\frac{\omega}{2\omega_{c1}}\right)$

Where  $\omega_c$  is the cut-off frequency.

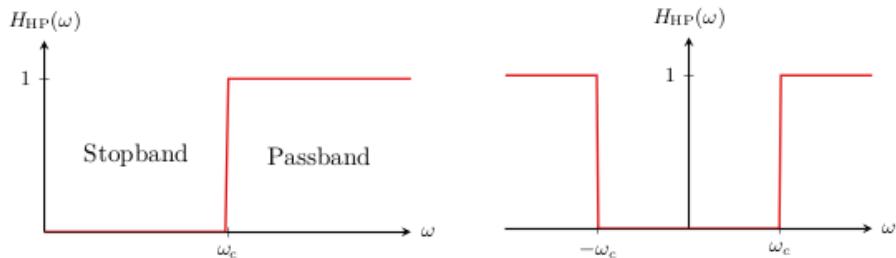


Figure 14.4: Lowpass and Highpass filter. From

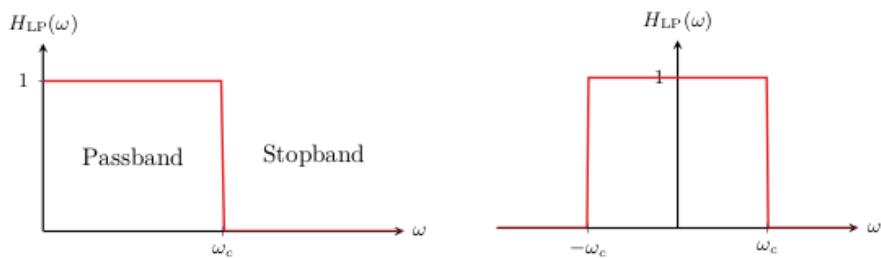


Figure 14.5: Bandpass and Bandstop filter. From

### 14.5.2 Filter Approximations

Ideal filters can not be implemented there for we use approximations.

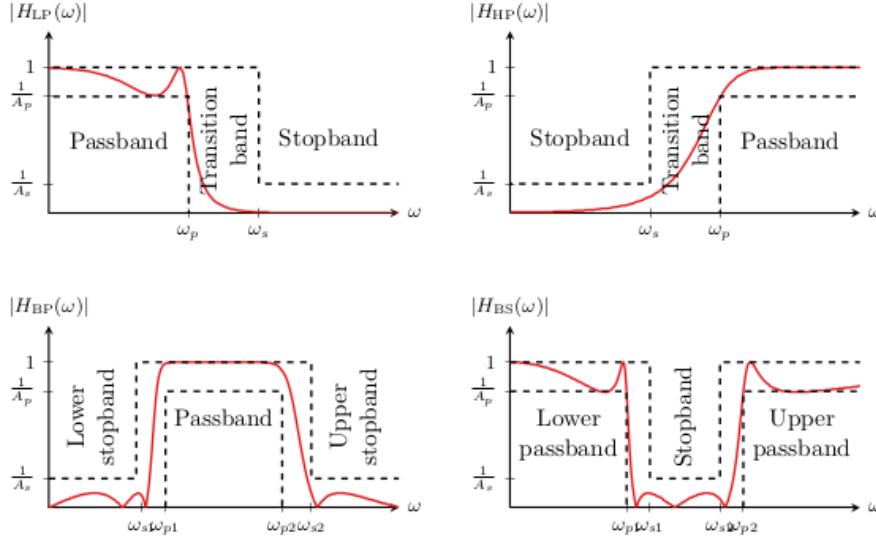


Figure 14.6: approximation-spectrum. From

- **passband ripple:** maximum amount of attenuation allowed in the passband.
- **passband edge frequency:** the frequency that marks the end of the requirement on the passband ripple
- **stopband attenuation:** the minimum attenuation in the stopband
- **stopband edge frequency:** the frequency that marks the start of the requirement on the stopband.

#### Cut-off frequency definition

$$|H(\omega_c)| = \frac{1}{\sqrt{2}} \approx 0.707 = -3dB$$

*Roll-off rate* is how quickly the real filter transition between the pass- and stopband.

$$R = \frac{d}{d\omega} \log_1 0(|H(\omega)|^2) \Big|_{\omega=\omega_c} = 2 \frac{d}{d\omega} \log_1 0(|H(\omega)|) \Big|_{\omega=\omega_c}$$

#### Approximation Types

- **Butterworth filter:**

$$- |H(\omega)| = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2N}}}$$

$$- N \geq \frac{\log \left( \sqrt{\frac{A_s^2 - 1}{A_p^2 - 1}} \right)}{\log \left( \frac{2\pi f_s}{2\pi f_p} \right)}$$

- **Chebyshev Type I filters:**

$$- |H(\omega)| = \frac{1}{\sqrt{1 + e^{2T_N^2} \left(\frac{\omega}{\omega_p}\right)^2}}$$

$$- N \geq \frac{\operatorname{arccosh} \left( \sqrt{\frac{A_s^2 - 1}{A_p^2 - 1}} \right)}{\operatorname{arccosh} \left( \frac{2\pi f_s}{2\pi f_p} \right)}$$

$$- T_N = \begin{cases} \cos(N \operatorname{arccos}(\omega)) & \text{for } |\omega| \leq 1 \\ \cosh(N \operatorname{arccosh}(\omega)) & \text{for } |\omega| > 1 \end{cases}$$

- **Chebyshev Type II filters:**

$$\begin{aligned} - |H(\omega)| &= \frac{1}{\sqrt{1 + [e^{2T_N^2} \left(\frac{\omega_s}{\omega}\right)]^{-1}}} = \\ &\sqrt{\frac{e^{2T_N^2} \left(\frac{\omega_s}{\omega}\right)}{1 + e^{2T_N^2} \left(\frac{\omega_s}{\omega}\right)}} \end{aligned}$$

$$- N =$$

$$- A_p = 10^{(\text{passband ripple}/20)}$$

$$-\epsilon = \sqrt{A_p^2 - 1}$$

- **Elliptic filters:**

$$-\lvert H(\omega) \rvert =$$

$$-N =$$

Where  $A_p$  is passband attenuation,  $A_s$  stopband attenuation.  $\omega_c$  is cutoff frequency in rad/s,  $f_c$  is cutoff frequency in Hz.  $\omega_s$  is the stopband edge frequency.  $\epsilon$  ripple control factor.  $N$  is the order.

- $R_p, R_s$  (in dB).

- $A_p = 10^{R_p/20}$

- $A_s = 10^{R_s/20}$

Butterworth Chebyshev

### Filter design

Example: Filter desing for Butterworth and Chebyshev type 1

TODO

Realed valued how to convert from dB to non dB what is sinc cosh arccosh

## 14.6 Sampling and Reconstruction

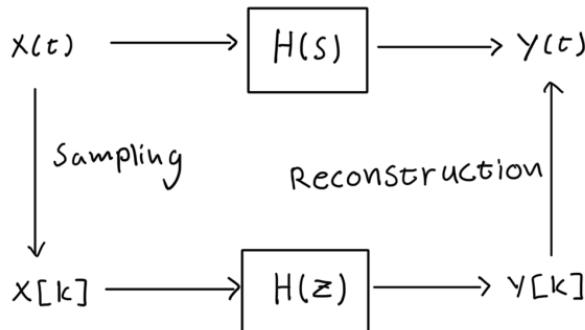


Figure 14.7: Sampling and Reconstruction

$$x[k] \triangleq x(kT_s)$$

Where  $T_s$  is the time delta for samplening and  $k$  is a integer.

We can do sampling in the time domain and in the frequency domain. We us a sampling signal with is a

diract delta function with is multiplied with the input signal in the time domain.

*Bandlimiting* is the limiting of signal's frequency do-maian representation.

### 14.6.1 Sampling

The sampling freqency  $\omega_s$  is equal to the higest frerquency  $\omega_b$  (b stands for bandwidth) time two.

$$\omega_s > 2\omega_b$$

#### Nyquist-Shannon Sampling Theorem

The creteria for sampling to work as inteded.

$$\omega_s = \frac{2\pi}{T_s}$$

$$\omega_s > 2\omega_b$$

$$f_s > 2f_c \text{ or } T_s < \frac{1}{2f_b}$$

$$\omega_N = \frac{\omega_s}{2}$$

In order to get the sampled signal we need to do the following.

$$x[k] = x(kT_s) = x_s(t) = x(t)s(t)$$

The *impulse train* ( $s(t)$ ) consisting of  $T_s$ -periodc Dirac delta functions.

$$s(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s)$$

Impulse train in the frequency domain yields:

$$S(\omega) = \mathcal{F}\{s(t)\} = \frac{2\pi}{T} \sum_{m=-\infty}^{\infty} \delta(\omega - m\omega_s)$$

And the output then becomes:

$$X_s(\omega) = \frac{1}{2\pi} X(\omega) * S(\omega)$$

*Oversampling* is the result of choosing a sampling freqency  $\omega_s$  larger then the Nyquist freqency  $\omega_N$ , but still smaller then the maximum freqency  $\omega_b$ .

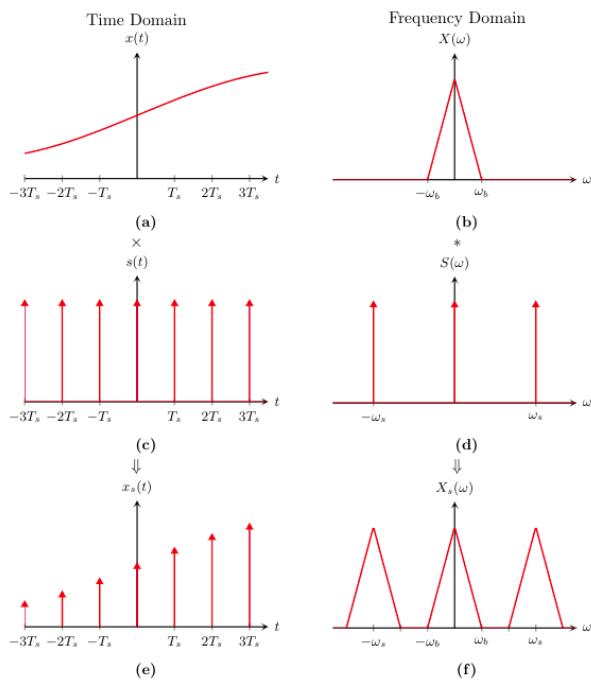


Figure 14.8: Sampling Graphs. From ?

### Aliasing and Anti-Aliasing

Aliasing occurs when

$$\omega_s < 2\omega_b$$

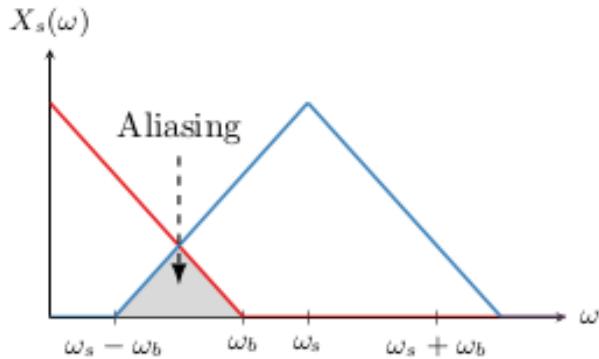


Figure 14.9: Aliasing. From ?

If there is overlap, one can not recover the signal and are therefore useless.

An anti-aliasing filter is placed before the analog to digital conversion (ADC). Ideally would the filter block above the maximum signal frequency  $\omega_b$  but this is not possible to implement such a filter which guarantees it and therefore will there be a small amount of aliasing leaking through.

### 14.6.2 Reconstruction

An ideal low pass filter can be used for reconstruction, since one can get the signal's frequencies. The filter is a rect in the frequency domain and therefore a sinc in the time domain.

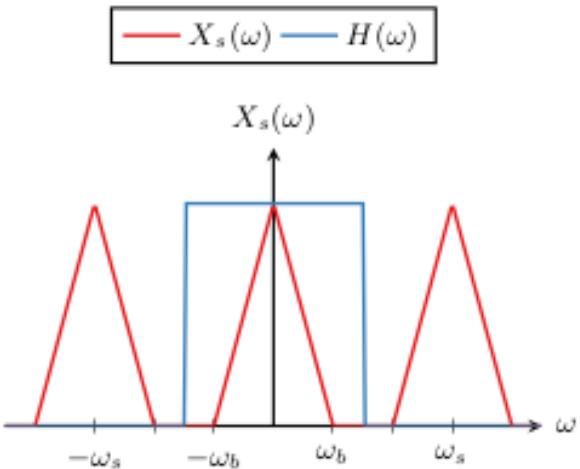


Figure 14.10: Ideal reconstruction. From ?

*Zero-order hold* (ZOH) is used to keep the output constant for the sampling period. After ZOH a smoothing filter is used.

$$x_{zoh}(t) = x_s(t) * \text{rect}\left(\frac{t - T_s/2}{T_s}\right)$$

$$X_{zoh}(\omega) = X_s(\omega)T_s \text{sinc}\left(\frac{\omega T_s}{2\pi}\right)e^{j\omega \frac{T_s}{2}}$$

### 14.6.3 Quantization

The amount of levels which the digital signal can hold.

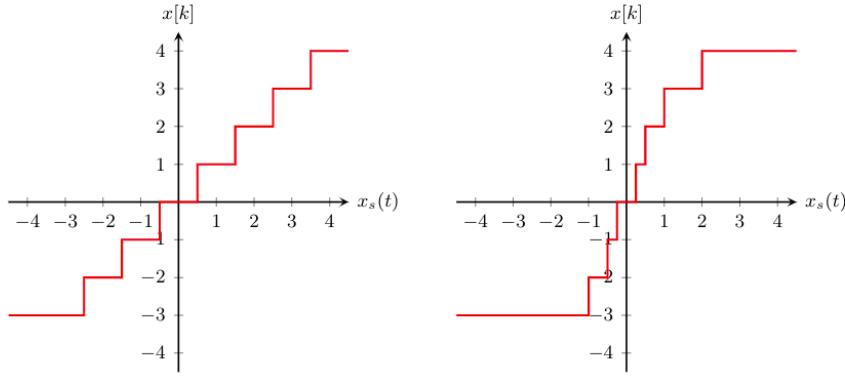


Figure 14.11: Two 3 bit quantizers ( $L = 2^3 = 8$ ). (a) uniform quantizer (b) non-uniform quantizer. From

## 14.7 Discrete time signals and systems

Discrete time signals are only defined on a valid integer  $k$ . Therefore we cannot have a stem one 0.5 for instance.

$K_0$  is the discrete time period. Therefore the following is true for periodic discrete time signals

$$x[k] = x[x + K_0]$$

also, the continuous time period can be derived

$$K_0 T_s = T_0$$

Example: Determine how many samples are needed

If you were to calculate the DFT using  $K$  samples of the signal without zero padding, how many samples do you need when the signal is

$$x[k] = 0.5 \cos\left(2\frac{2\pi}{10}k\right) + \cos\left(3\frac{3\pi}{10}k\right)$$

We know  $\frac{\omega_s}{K} \leq \Delta\omega$  where  $x_1(t) = \cos(\omega_1 t)$  and  $x_2(t) = \cos((\omega_1 + \Delta\omega)t)$

$$\begin{aligned}\Delta\Omega &= \frac{2\pi}{K}, \quad \Delta\Omega = 3\frac{2\pi}{10} - 2\frac{2\pi}{10} = \frac{2\pi}{10} \\ \frac{2\pi}{K} &= \frac{2\pi}{10} \Rightarrow K = 10\end{aligned}$$

Hence we need 10 samples.

### 14.7.1 Discrete Time Signals

#### Elementary Discrete Time Signals

- Sine and cosine signals:

- $x[k] = A \sin(\Omega_0 k + \varphi)$
- $x[k] = A \cos(\Omega_0 k + \varphi)$

- **Unit impulse:**

$$-\delta[k] = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{for } k \neq 0 \end{cases}$$

- **Rectangular pulse:**

$$-rect\left[\frac{k}{2N+1}\right] = \begin{cases} 1 & \text{for } |k| \neq N \\ 0 & \text{for } |k| > N \end{cases}$$

- **Unit step:**

$$-u[k] = \begin{cases} 0 & \text{for } k < 0 \\ 1 & \text{for } k \geq 0 \end{cases}$$

- **Sinc:**

$$-sinc\left[\frac{k}{K_0}\right] = \frac{\sin\left(\pi\frac{k}{K_0}\right)}{\pi\frac{k}{K_0}}$$

### Discrete Time Signal Properties

- **Periodic signals:**  $x[k] = x[k + K_0]$
- **Causal signals:**  $x[k] = 0$  for  $k < 0$
- **Energy and power signals:** (can never be both)

- $E = \sum_{k=-\infty}^{\infty} |x[k]|^2$  where  $0 < E < \infty$
- $P = \lim_{K \rightarrow \infty} \frac{1}{K+1} \sum_{k=-K/2}^{K/2} |x[k]|^2$  for aperiodic signals
- $P = \frac{1}{K_0} \sum_{k=0}^{K_0-1} |x[k]|^2$  for periodic signals

$$\Omega_0 = \frac{2\pi}{K_0}$$

### 14.7.2 Time Domain Analysis of Discrete Time Systems

#### Impulse Response

$$\begin{aligned} \delta[k] &\mapsto h[k] \\ \alpha\delta[k - k_0] &\mapsto \alpha h[k - k_0] \end{aligned}$$

#### Convolution Sum

input output relationship for discrete time (definition)

$$\begin{aligned} y[k] &= \sum_{m=-\infty}^{\infty} h[k]x[k-m] = h[k] * x[k] \\ y[k] &= \sum_{m=0}^{\infty} h[k]x[k-m] \text{ for causal systems} \end{aligned}$$

Property	
Commutation	$x_1[k] * x_2[k] = x_2[k] * x_1[k]$
Distribution	$x_1[k] * (x_2[k] + x_3[k]) = x_1[k] * x_2[k] + x_1[k] * x_3[k]$
Association	$x_1[k] * (x_2[k] * x_3[k]) = x_1[k] * x_2[k] * x_3[k]$
Time shifting	$x_1[k - k_1] * x_2[k - k_2] = y[k - k_1 - k_2]$
Duration	$x_1[k] = 0 \text{ for } k \geq K_1, x_2[k] = 0 \text{ for } k \geq K_2 \Rightarrow x_1[k] * x_2[k] = 0 \text{ for } k \geq K_1 + K_2$
Convolution with impulse function	$x[k] * \delta[k - K] = x[k - K]$
Convolution with step function	$x[k] * u[k] = \sum_{m=-\infty}^k x[m]$

Figure 14.12: Properties of the discrete time convolution sum. From

## 14.8 Discrete Time Fourier Analysis

### 14.8.1 Discrete Time Fourier Series and Transform

#### Definition and Properties

##### Definition of Discrete time Fourier transform (DTFT)

$$x[k] = \sum_{n=0}^{K_0-1} c_n e^{-jn\Omega_0 k}$$

Fourier transform for sampled continuous time signals:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \sum_{k=-\infty}^{\infty} \delta(t - kT_s) e^{-j\omega t} dt = \sum_{k=-\infty}^{\infty} x(kT_s) e^{-j\omega kT_s}$$

The frequency can be expressed:

$$\Omega = \frac{2\pi}{K} \quad \text{which is similar to } \omega = \frac{2\pi}{T}$$

Normalized frequency.

$$\Omega = \omega T_s = 2\pi \frac{f}{f_s}$$

Mapping frequencies:

$$\begin{aligned} \Omega = 2\pi &\Leftrightarrow \omega = \omega_s \\ \Omega = \pi &\Leftrightarrow \omega = \frac{\omega_s}{2} = \omega_N \end{aligned}$$

Sampling frequency.

$$X(\Omega) = \sum_{k=-\infty}^{\infty} x[k] e^{-j\Omega k}$$

### Definition of Discrete time Fourier transform

$$X(\Omega) = \sum_{k=-\infty}^{\infty} x[k]e^{-j\Omega k}$$

$$x[k] = \frac{1}{2\pi} \int_0^{2\pi} X(\Omega)e^{j\Omega k} d\Omega$$

Transfor pair and the important property that the fourier transform is alwase  $2\pi$  for discrete time.

$$x[k] \circledast X(\Omega)$$

$$X(\Omega) = X(\Omega + 2\pi)$$

Property	Time Domain	Frequency Domain
Periodicity	-	$X(\Omega) = X(\Omega + 2\pi)$
Symmetry	Real-valued	$X(-\Omega) = X^*(\Omega)$
Linearity	$\alpha x_1[k] + \beta x_2[k]$	$\alpha X_1(\Omega) + \beta X_2(\Omega)$
Time shifting ( $k_0$ : integer)	$x[k - k_0]$	$e^{-j\Omega k_0} X(\Omega)$
Time differencing	$x[k] - x[k - 1]$	$(1 - e^{-j\Omega}) X(\Omega)$
Frequency domain differentiation	$-j k x[k]$	$\frac{dX(\Omega)}{d\Omega}$
Time summation	$\sum_{n=-\infty}^k x[n]$	$\frac{X(\Omega)}{1 - e^{-j\Omega}} + \pi X(0) \sum_{m=-\infty}^{\infty} \delta(\Omega - 2\pi m)$
Convolution	$x_1[k] * x_2[k]$	$X_1(\Omega) X_2(\Omega)$
Multiplication <sup>†</sup>	$x_1[k] x_2[k]$	$\frac{1}{2\pi} X_1(\Omega) * X_2(\Omega)$
Parseval's theorem	$E = \sum_{k=-\infty}^{\infty}  x[k] ^2$	$E = \frac{1}{2\pi} \int_0^{2\pi}  X(\Omega) ^2 d\Omega$

Figure 14.13: Properties of the discrete time fourier transform. From

*Hermitian symmetry:*  $X(-\omega) = X^*(\omega)$  for real valued  $x[k]$

### Frequency Domain Analysis of Discrete Time Systems

$$y[k] = h[k] * x[k]$$

$$x_1[k] * x_2[k] \circledast X_1(\Omega) X_2(\Omega)$$

$$Y(\Omega) = H(\Omega) X(\Omega)$$

$$H(\Omega) = \sum_{k=-\infty}^{\infty} h[k] e^{-j\Omega k}$$

### 14.8.2 Discrete Fourier Transform

With DFT we can determine the spectrum without knowing the signal type of input signals  $x[k]$ .

#### Definition

$$X(l) = \sum_{k=0}^{K-1} x[k] e^{-jlk \frac{2\pi}{K}}$$

$$x[k] = \frac{1}{K} \sum_{l=0}^{K-1} X[l] e^{jlk \frac{2\pi}{K}}$$

Where  $L$  is the *Frequency bins*

Property	Time Domain	Frequency Domain
Periodicity	-	$X[l] = X[l + K]$
Symmetry	Real-valued	$X[l] = X^*[K - l]$
Linearity	$\alpha x_1[k] + \beta x_2[k]$	$\alpha X_1[l] + \beta X_2[l]$
Parseval's Theorem	$E = \sum_{k=0}^{K-1}  x[k] ^2$	$E = \frac{1}{K} \sum_{l=0}^{K-1}  X[l] ^2$

Figure 14.14: Properties of the discrete time Fourier transform. From ?

#### Graphical Interpretation of the DFT

Calculating the DFT of a sampled continuous time signal consists of the following three steps:

1. **Time domain sampling.** The convolution of  $X(\omega)$  and  $S(\omega)$  in the frequency domain, resulting in a spectrum for the sampled signal  $x_s(t)$  with  $\omega_s$ -periodic spectrum with frequency-shift copies of  $X(\omega)$ .
2. **Time limited.** Windowed signal  $x_{sw}(t) = x_s(t)w(t)$ , thus the Fourier transform is the convolution between the sampled signal's Fourier transform  $X_s(\omega)$  and  $W(\omega)$  (a sinc function). which yields the spectrum  $X_{sw}(\omega)$
3. **Frequency domain sampling.** Sampling the frequency domain with a sampling interval of  $\frac{\omega_s}{L}$ . This is a multiplication of the sampled signal in time domain and the window signal  $x_{sw}(t)$  with an impulse train in the frequency domain.

#### Relationship to the Continuous and Discrete Time Fourier Transforms

Discrete Fourier Transform (DFT) and discrete time Fourier Transform (DTFT) are not the same, however there are similar and their relation is

$$X(\Omega)|_{\Omega=l \frac{2\pi}{L}} = X[l]$$

Also DFT is non-zero.

For aperiodic CTFT can be approximated as

$$X(\omega)|_{\omega=l \frac{\omega_s}{L}} \approx T_s X[l]$$

if the continuous time signal is limited to interval  $0, \dots, (K-1)T_s$

In other words  $\Omega_l = l \frac{2\pi}{L}$ ,  $\omega_l = l \frac{\omega_s}{L}$ ,  $f_l = l \frac{f_s}{L}$ .

### Zero-Padding, Graphical Resolution, and Spectral Resolution

- *Zero-Padding* is when calculating the DFT for  $L > K$ .
- *Graphical resolution* is the appearance of the spectrum plot.
- *Spectral resolution* is the resolution which determines how accurate the calculated Fourier transform is.

### Windowing

- **Rectangular window:** Standard window. Most narrow mainlobe but the strongest sidelobes.
- **Bartlett and triangle windows:** smaller sidelobes than rectangular window but wider mainlobe.
- **Hamming window:** Smoother transitions lead to smaller sidelobes and hamming window is the smoothest and therefore the smallest sidelobes.

Windowing causes *smearing* (wide sidelobes) and *leakage* (new spectral content with sidelobes).

### Fast Fourier Transform (FFT)

One of the most important algorithms ever. It is used for determining the Fourier transform of a discrete signal. FFT is an efficient way of implementing DFT when  $K = 2^M$ , from  $O(K^2)$  to  $O(K \log_2(K))$ . Since the standard way of calculating the Fourier transform involves a matrix multiplication with  $K \times K$ , therefore the time complexity is  $O(K^2)$ . (the matrix is DFT). There exist symmetry in the DFT and other redundant properties, therefore we can optimize it. FFT is a factorization with split odds and evens into separate columns and then multiply with a diagonal matrix times another matrix.

The *twiddle factor*

$$W_K = e^{-j \frac{2\pi}{K}}$$

$$\begin{aligned} X[l] &= \sum_{k=0}^{\frac{K}{2}-1} x[2k] W_K^{lk} + W_K^l \sum_{k=0}^{\frac{K}{2}-1} x[2k+1] W_K^{lk} \\ &= X_1[l] + W_k^l X_2[l] \end{aligned}$$

## 14.9 z-transform

### 14.9.1 Definition and Properties

#### Definition

The discrete-time Fourier transform for causal signal can be expressed as

$$X(\Omega) = \sum_{k=0}^{\infty} x[k] r^{-k} e^{-j\Omega k}$$

where  $r^{-k}$  is a decaying exponential function with is used to scale  $x[k]$ . The z-transform can then be derived for  $z = re^{j\Omega}$ .

$$\begin{aligned} x[k] &\circlearrowleft X(z) \\ X(z) &= \sum_{k=0}^{\infty} x[k]z^{-k} \\ x[k] &= \frac{1}{2\pi j} \oint X(z)z^{k-1}dz \end{aligned}$$

Time domain $x[k]$	z-Transform $X(z)$	ROC
<b>Unit impulse</b> $x[k] = \delta[k]$	$X(z) = 1$	all $z$
<b>Unit step</b> $x[k] = u[k]$	$X(z) = \frac{1}{1 - z^{-1}}$	$ z  > 1$
<b>Exponential</b> $x[k] = a^k u[k]$	$X(z) = \frac{1}{1 - az^{-1}}$	$ z  >  a $
<b>Ramp</b> $x[k] = ku[k]$	$X(z) = \frac{z^{-1}}{(1 - z^{-1})^2}$	$ z  > 1$
<b>Cosine</b> $x[k] = \cos(\Omega_0 k)u[k]$	$X(z) = \frac{1 - z^{-1} \cos(\Omega_0)}{1 - 2z^{-1} \cos(\Omega_0) + z^{-2}}$	$ z  > 1$
<b>Sine</b> $x[k] = \sin(\Omega_0 k)u[k]$	$X(z) = \frac{z^{-1} \sin(\Omega_0)}{1 - 2z^{-1} \cos(\Omega_0) + z^{-2}}$	$ z  > 1$
<b>Decaying cosine</b> $x[k] = a^k \cos(\Omega_0 k)u[k]$	$X(z) = \frac{1 - az^{-1} \cos(\Omega_0)}{1 - 2az^{-1} \cos(\Omega_0) + a^2 z^{-2}}$	$ z  >  a $
<b>Decaying sine</b> $x[k] = a^k \sin(\Omega_0 k)u[k]$	$X(z) = \frac{az^{-1} \sin(\Omega_0)}{1 - 2az^{-1} \cos(\Omega_0) + a^2 z^{-2}}$	$ z  >  a $

Figure 14.15: z-transform pairs. From ?

### Properties

Property	Time Domain	z-Domain
Linearity	$\alpha x_1[k] + \beta x_2[k]$	$\alpha X_1(z) + \beta X_2(z)$
Time shifting	$x[k - m]$	$z^{-m}X(z)$
Convolution	$x_1[k] * x_2[k]$	$X_1(z)X_2(z)$
Scaling	$a^k x[k]$	$X\left(\frac{z}{a}\right)$
Time difference	$x[k] - x[k - 1]$	$(1 - z^{-1})X(z)$
Accumulation	$\sum_{m=0}^k x[m]$	$\frac{1}{1 - z^{-1}}X(z)$
Initial value theorem	-	$x[0] = \lim_{z \rightarrow \infty} X(z)$
Final value theorem	-	$\lim_{k \rightarrow \infty} x[k] = \lim_{z \rightarrow 1} (z - 1)X(z)$

Figure 14.16: properties z-transform. From ?

### 14.9.2 Transfer Function

#### Difference Equations and Transfer Functions

Input output relationship, is expressed as *difference equations* (recursive) Sometimes the difference equation are given as looking at  $N$  steps into the future, in that case the easiest approach is to shifting of difference equations.

**Definition:** Transfer function of discrete time LTI systems

$$\begin{aligned} y[k] + a_1y[k - 1] + \dots + a_Ny[k - N] &= b_0x[k] + b_1x[k - 1] + \dots + b_Mx[k - M] \\ H(z) &= \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + \dots + b_Mz^{-M}}{1 + a_1z^{-1} + \dots + a_Nz^{-N}} \\ Y(z) &= H(z)X(z) \end{aligned}$$

#### Transfer Function and Impulse Response

**Definition:** Relationship between the discrete time impulse response and transfer function

$$\begin{aligned} h[k] &\circledast H(z) \\ H(z) &= \sum_{k=0}^{\infty} h[k]z^{-k} \end{aligned}$$

#### Transfer Function and Frequency Response

**Definition:** Relationship between the discrete time transfer function and the frequency response

$$H(\Omega) = H(e^{j\Omega}) = H(z)|_{z=e^{j\Omega}}$$

if the impulse response is causal,  $z = e^{j\omega}$  and the DTFT converges.

### Poles and Zeros, and Stability

Similarly to the laplace transform we can derive the poles and zeros from the tranfer function. **Definition:**  
**Pole-zero form of the fiscrete time transfer function**

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_m z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}}$$

$$H(z) = b_0 z^{N-m} \frac{(z - z_1)(z - z_2) \dots (z - z_M)}{(z - p_1)(z - p_2) \dots (z - p_N)}$$

**Spectrum behavior for diffrent pole and zero positions** If the pole is inside the circle it will create a top which dose not go to infinity. It will however have a higher top the closer it is to the circle.

Zeros will have the oposite affect, creating a dip. The closer the the circle the futher down the dip will reach.

### Stability

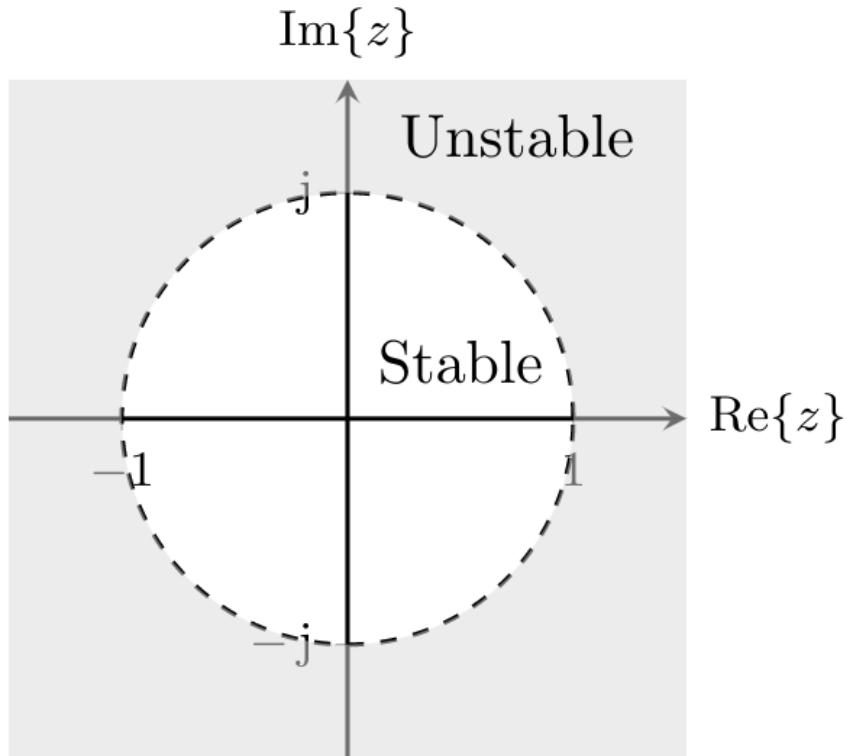


Figure 14.17: Stability for z-transforms. From ?

### Example: stability casual and frequency response

What is the systems frequency response and approximate it graphically? Also determin if the system is stable and causal.

$$y[k+2] + 5y[k+1] + 0.1y[k] = x[k+1]$$

We can see that the output only depends on previous values there for it can be causal.

First we determin the systems transfer function

$$\begin{aligned} z^2Y(z) + 5zY(z) + 0.1 &= zX(z) \quad \text{-(Time shifting property)} \\ Y(z)(z^2 + 5z + 0.1) &= zX(z) \\ \frac{Y(z)}{X(z)} &= \frac{z}{z^2 + 5z + 0.1} = H(z) \quad \text{-(Witch is the tranfer function)} \end{aligned}$$

Next we determin the poles to see if there are inside the complex unit circle, in order to determine the stability.

$$\begin{aligned} z^2 + 5z + 0.1 &= 0 \\ z = \frac{-5}{2} \pm \sqrt{\frac{5^2}{2^2} - 0.1} &= \frac{-5}{2} \pm \frac{\sqrt{24.6}}{2} \quad \text{-(pq-formula)} \\ p_1 \approx -0.0201, p_2 \approx -5 \end{aligned}$$

Since  $|p_2| > 1$  is the system not stable.

Determine the frequency response

$$H(\Omega) = H(z)|_{z=e^{j\Omega}} = \frac{e^{j\Omega}}{e^{2j\Omega} + 5e^{j\Omega} + 0.1}$$

Calculating some points to be able to approximate the graph:

$$\begin{aligned} |H(\Omega)|_{\Omega=2\pi}| &= |H(\Omega)|_{\Omega=0}| = \left| \frac{e^0}{e^0 + 5e^0 + 0.1} \right| \approx 0.16 \quad \text{-(2\pi-periodic)} \\ |H(\Omega)|_{\Omega=3\pi/2}| &= |H(\Omega)|_{\Omega=\pi/2}| = \left| \frac{e^{\pi/2}}{e^\pi + 5e^{\pi/2} + 0.1} \right| = \left| \frac{-j}{-1 + 5(-j) + 0.1} \right| \approx 2 \\ |H(\Omega)|_{\Omega=\pi}| &= \left| \frac{e^\pi}{e^{2\pi} + 5e^\pi + 0.1} \right| = \left| \frac{-1}{1 + 5(-1) + 0.1} \right| \approx 0.26 \end{aligned}$$

## 14.10 Digital Filters

### 14.10.1 Introduction

#### Background

#### Continuous Time Requirements and Digital Filters

#### Digital Filters and LTI Systems

### 14.10.2 Finite Impulse Response Filters

#### Definition and Properties

*Filter taps* are the coefficients  $b_n$ . Thus, an Nth order filter has  $N + 1$  filter taps (starts from 0 and ends at  $N$ )

FIR Filters are allways stable since it does not have any poles only zeros.

## FIR Filter Design

Ideal Frequency Response ( $\pi \leq \Omega < \pi$ )		Ideal Impulse Response
<b>Lowpass</b>	$H(\Omega) = \text{rect}\left(\frac{\Omega}{2\Omega_c}\right)$	$h[k] = \frac{\Omega_c}{\pi} \text{sinc}\left[\frac{\Omega_c k}{\pi}\right]$
<b>Highpass</b>	$H(\Omega) = 1 - \text{rect}\left(\frac{\Omega}{2\Omega_c}\right)$	$h[k] = \delta[k] - \frac{\Omega_c}{\pi} \text{sinc}\left[\frac{\Omega_c k}{\pi}\right]$
<b>Bandpass</b>	$H(\Omega) = \text{rect}\left(\frac{\Omega}{2\Omega_{c2}}\right) - \text{rect}\left(\frac{\Omega}{2\Omega_{c1}}\right)$	$h[k] = \frac{\Omega_{c2}}{\pi} \text{sinc}\left[\frac{\Omega_{c2} k}{\pi}\right] - \frac{\Omega_{c1}}{\pi} \text{sinc}\left[\frac{\Omega_{c1} k}{\pi}\right]$
<b>Bandstop</b>	$H(\Omega) = 1 - \text{rect}\left(\frac{\Omega}{2\Omega_{c2}}\right) + \text{rect}\left(\frac{\Omega}{2\Omega_{c1}}\right)$	$h[k] = \delta[k] - \frac{\Omega_{c2}}{\pi} \text{sinc}\left[\frac{\Omega_{c2} k}{\pi}\right] + \frac{\Omega_{c1}}{\pi} \text{sinc}\left[\frac{\Omega_{c1} k}{\pi}\right]$

Figure 14.18: Descrete time filter frequency responce and their impulse responses

## Windowing

### Parks-McClellan

#### 14.10.3 Infinite Impulse Response Filters

##### Definition and Properties

##### IIR Filter Design Using the Bilinear Transform

IIR filter design methods can be divided into two types:

1. Indirect methods that are based on converting continuous time filters to discrete time
2. Direct methods that directly design digital IIR filters in the discrete time domain.

##### Requirements

1. Filter requirement specification (passband and stopband edge frequencies, passband ripple, stopband attenuation);
2. Design of the analog filter transfer function  $H(s)$  (see Chapter 5);
3. Approximation of the analog filter transfer function using the bilinear transform.

##### Filter Coefficient Quantization and Stability

There could stabile in theroy but unstable in practice. Since the computer may have to round the number in a way witch makes the system unstable.

## 14.11 How it is connected

Concept	Continous signals $x(t)$	Discrete signals $x[k]$	Description
Impulse reponse	$\delta(t) \mapsto h(t)$	$\delta[k] \mapsto h[k]$	Is used to determin the output of the system, since the output is $y[k] = h[k] * x[k]$
Frequency reponse	$H(\omega)$	$H(\Omega)$ with period $2\pi$	Is the equivilant to the impulse reponce in the frequency domain.
Signal's spectrum	$X(\omega)$	$X(\Omega)$	Is used to view the spectrum where one can see what frequencies dominates
Transfer function	$H(s)$ (Laplace)	$H(z)$ (z-transform)	Laplace and z-transform are use to analyse a system, with pole and zeros.
Discrete Fourier transform	-	$X[l]$	FFT is an algorith to compute the DFT, it turns the samples (wich become a vector) and convert it to a there frequency componetnts. We can not allwase use the DTFT since it requires knoladge of the full signal, not just finite set of samples.
Spectrum from DFT calculation	-	$X(f_l)$ with period $f_s$ ( $\dots X(f)$ )	We can not alwase determin the signals spectrum analytically therfore we can approximate it by using the DFT

**Note:** The fourier tranforms are  $X(\omega)$  for Continous time,  $X(\Omega)$  for discrete time and  $X[l]$  for discrete (NO TIME).

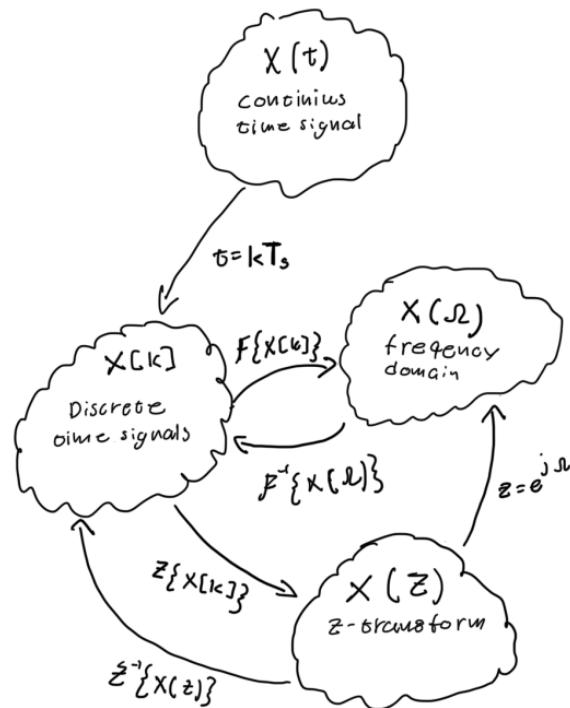


Figure 14.19: Connection mind map.

Example: sine in sine out for discrete time signal

The sampling time is  $T_s = 0.05$

$$x(t) = 0.25 \cos(2\pi/410t) \Rightarrow x[k] = 0.25 \cos(2\pi/410kT_s) = 0.25 \cos(\pi/4k)$$

$$H(\Omega) = H(z)|_{z=e^{j\Omega}} = \frac{1 - 0.8e^{-j\Omega} + 0.16e^{-2j\Omega}}{1 - e^{-j\Omega} + 0.5e^{-2j\Omega}} \quad \text{Where } \Omega_0 = \frac{\pi}{4}$$

$$\begin{aligned}|H(e^{j\pi/4})| &= 1.7 \\ \angle H(e^{j\pi/4}) &= 0.14 \quad (7.8^\circ)\end{aligned}$$

$$y[k] = 0.4 \cos\left(\frac{\pi}{4}k + 0.14\right)$$

# Appendices



Formula Tables  
Signals and Transforms

October 19, 2021

### 1 Continuous Time Fourier Series

#### Trigonometric Continuous Time Fourier Series

$$x(t) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)]$$

with

$$\begin{aligned} a_0 &= \frac{1}{T_0} \int_{(T_0)} x(t) dt \\ a_n &= \frac{2}{T_0} \int_{(T_0)} x(t) \cos(n\omega_0 t) dt \\ b_n &= \frac{2}{T_0} \int_{(T_0)} x(t) \sin(n\omega_0 t) dt \end{aligned}$$

#### Exponential Continuous Time Fourier Series

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t} \quad \text{with} \quad c_n = \frac{1}{T_0} \int_{(T_0)} x(t) e^{-jn\omega_0 t} dt$$

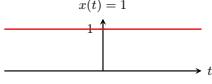
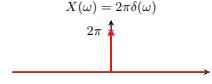
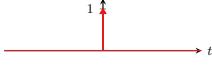
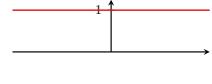
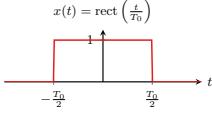
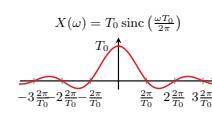
### 2 Continuous Time Fourier Transform

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \quad \text{and} \quad x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega$$

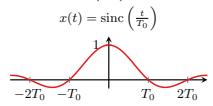
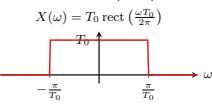
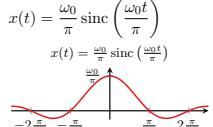
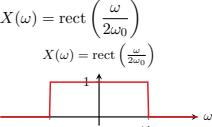
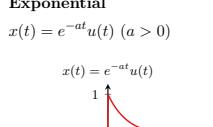
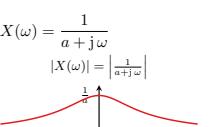
**Table 1.** Properties of the continuous time Fourier transform.

Property	Time Domain	Frequency Domain
Symmetry	Real-valued	$X(-\omega) = X^*(\omega)$
Linearity	$\alpha x_1(t) + \beta x_2(t)$	$\alpha X_1(\omega) + \beta X_2(\omega)$
Duality	$X(t)$	$2\pi x(-\omega)$
Scaling	$x(at)$	$\frac{1}{ a } X\left(\frac{\omega}{a}\right)$
Time shift	$x(t - \Delta t)$	$e^{-j\omega\Delta t} X(\omega)$
Frequency shift	$x(t)e^{j\omega_0 t}$	$X(\omega - \omega_0)$
Differentiation	$\frac{d^n x(t)}{dt^n}$	$(j\omega)^n X(\omega)$
Integration	$\int_{-\infty}^t x(\tau) d\tau$	$\frac{1}{j\omega} X(\omega) + \pi X(0)\delta(\omega)$
Convolution	$x_1(t) * x_2(t)$	$X_1(\omega) X_2(\omega)$
Multiplication	$x_1(t)x_2(t)$	$\frac{1}{2\pi} X_1(\omega) * X_2(\omega)$
Parseval's theorem	$E = \int_{-\infty}^{\infty}  x(t) ^2 dt$	$E = \frac{1}{2\pi} \int_{-\infty}^{\infty}  X(\omega) ^2 d\omega$

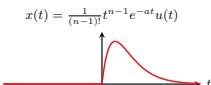
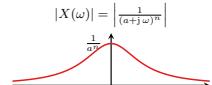
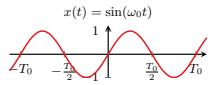
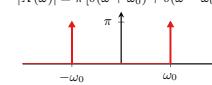
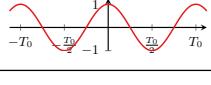
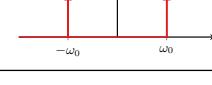
**Table 2.** Continuous time Fourier transforms of elementary signals.

Time Domain $x(t)$	Fourier Transform $X(\omega)$
<b>Periodic signals</b>	
$T_0$ -periodic ( $\omega_0 = \frac{2\pi}{T_0}$ )	$X(\omega) = 2\pi \sum_{n=-\infty}^{\infty} c_n \delta(\omega - n\omega_0)$ $\text{with } c_n = \frac{1}{T_0} \int_0^{T_0} x(t) e^{-jn\omega_0 t} dt$
<b>Constant</b> $x(t) = 1$ 	$X(\omega) = 2\pi\delta(\omega)$ 
<b>Dirac delta function</b> $x(t) = \delta(t)$ 	$X(\omega) = 1$ 
<b>Rectangular pulse</b> $x(t) = \text{rect}\left(\frac{t}{T_0}\right)$ $x(t) = \text{rect}\left(\frac{t}{T_0}\right)$ 	$X(\omega) = T_0 \text{sinc}\left(\frac{\omega T_0}{2\pi}\right)$ 

**Table 2.** Continuous time Fourier transforms of elementary signals (cont'd.).

Time Domain $x(t)$	Fourier Transform $X(\omega)$
<b>Sinc function</b>	
$x(t) = \text{sinc}\left(\frac{t}{T_0}\right)$	$X(\omega) = T_0 \text{rect}\left(\frac{\omega T_0}{2\pi}\right)$ 
$x(t) = \text{sinc}\left(\frac{t}{T_0}\right)$	$X(\omega) = T_0 \text{rect}\left(\frac{\omega T_0}{2\pi}\right)$ 
<b>Sinc function (alternative parametrization)</b>	
$x(t) = \frac{\omega_0}{\pi} \text{sinc}\left(\frac{\omega_0 t}{\pi}\right)$	$X(\omega) = \text{rect}\left(\frac{\omega}{2\omega_0}\right)$ 
$x(t) = \frac{\omega_0}{\pi} \text{sinc}\left(\frac{\omega_0 t}{\pi}\right)$	$X(\omega) = \text{rect}\left(\frac{\omega}{2\omega_0}\right)$ 
<b>Exponential</b>	
$x(t) = e^{-at} u(t) \quad (a > 0)$	$X(\omega) = \frac{1}{a + j\omega}$ 
$x(t) = e^{-at} u(t)$	$ X(\omega)  = \left  \frac{1}{a + j\omega} \right $ 

**Table 2.** Continuous time Fourier transforms of elementary signals (cont'd.).

Time Domain $x(t)$	Fourier Transform $X(\omega)$
<b>Multiple real-valued poles</b>	
$x(t) = \frac{1}{(n-1)!} t^{n-1} e^{-at} u(t)$ ( $a > 0, n = 1, 2, \dots$ )	$X(\omega) = \frac{1}{(a + j\omega)^n}$
	
<b>Sine</b>	
$x(t) = \sin(\omega_0 t)$	$X(\omega) = j\pi [\delta(\omega + \omega_0) - \delta(\omega - \omega_0)]$ $ X(\omega)  = \pi [\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]$
	
<b>Cosine</b>	
$x(t) = \cos(\omega_0 t)$	$X(\omega) = \pi [\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]$ $X(\omega) = \pi [\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]$
	

**3 Laplace Transform**

$$X(s) = \int_0^\infty x(t)e^{-st} dt \quad \text{and} \quad x(t) = \frac{1}{2\pi j} \int_{\sigma-j\omega}^{\sigma+j\omega} X(s)e^{st} ds$$

**Table 3.** Properties of the Laplace transform.

Property	Time Domain	Laplace Domain
Linearity	$\alpha x_1(t) + \beta x_2(t)$	$\alpha X_1(s) + \beta X_2(s)$
Scaling ( $a > 0$ )	$x(at)$	$\frac{1}{a} X\left(\frac{s}{a}\right)$
Time shift	$x(t - \Delta t)$	$e^{-s\Delta t} X(s)$
s-domain shift	$x(t)e^{at}$	$X(s-a)$
Differentiation (first order)	$\frac{dx(t)}{dt}$	$sX(s) - x(0)$
Differentiation (nth order)	$\frac{d^n x(t)}{dt^n}$	$s^n X(s) - s^{n-1}x(0) - s^{n-2}x^{(1)}(0) - \dots - x^{(n-1)}(0)$
Integration	$\int_0^t x(\tau) d\tau$	$\frac{1}{s} X(s)$
Convolution	$x_1(t) * x_2(t)$	$X_1(s)X_2(s)$
Multiplication	$x_1(t)x_2(t)$	$\frac{1}{2\pi j} X_1(s) * X_2(s)$
Initial value theorem	$\lim_{t \rightarrow 0} x(t)$	$\lim_{s \rightarrow \infty} sX(s)$
Final value theorem	$\lim_{t \rightarrow \infty} x(t)$	$\lim_{s \rightarrow 0} sX(s)$

**Table 4.** Table of unilateral Laplace transform pairs for causal signals  $x(t)$  ( $x(t) = 0$  for  $t < 0$ ).

Time domain $x(t)$	Laplace transform $X(s)$	ROC
<b>Dirac delta function</b>		
$x(t) = \delta(t)$	$X(s) = 1$	All $s$
<b>Unit step</b>		
$x(t) = u(t)$	$X(s) = \frac{1}{s}$	$\text{Re}\{s\} > 0$
<b>Exponential</b>		
$x(t) = e^{-at}u(t)$	$X(s) = \frac{1}{a+s}$	$\text{Re}\{s\} > -a$
<b>Ramp</b>		
$x(t) = tu(t)$	$X(s) = \frac{1}{s^2}$	$\text{Re}\{s\} > 0$
<b>Higher order ramp</b>		
$x(t) = t^n u(t)$	$X(s) = \frac{n!}{s^{n+1}}$	$\text{Re}\{s\} > 0$
<b>Cosine</b>		
$x(t) = \cos(\omega_0 t)u(t)$	$X(s) = \frac{s}{\omega_0^2 + s^2}$	$\text{Re}\{s\} > 0$
<b>Sine</b>		
$x(t) = \sin(\omega_0 t)u(t)$	$X(s) = \frac{\omega_0}{\omega_0^2 + s^2}$	$\text{Re}\{s\} > 0$
<b>Decaying cosine</b>		
$x(t) = e^{-at} \cos(\omega_0 t)u(t)$	$X(s) = \frac{a+s}{(a+s)^2 + \omega_0^2}$	$\text{Re}\{s\} > -a$
<b>Decaying sine</b>		
$x(t) = e^{-at} \sin(\omega_0 t)u(t)$	$X(s) = \frac{\omega_0}{(a+s)^2 + \omega_0^2}$	$\text{Re}\{s\} > -a$

#### 4 Discrete Time Fourier Series

$$x[k] = \sum_{n=0}^{K_0-1} c_n e^{\text{j} n \Omega_0 k} \quad \text{with} \quad c_n = \frac{1}{K_0} \sum_{k=0}^{K_0-1} x[k] e^{-\text{j} n \Omega_0 k}$$

#### 5 Discrete Time Fourier Transform

$$X(\Omega) = \sum_{k=-\infty}^{\infty} x[k] e^{-\text{j} \Omega k} \quad \text{and} \quad x[k] = \frac{1}{2\pi} \int_0^{2\pi} X(\Omega) e^{\text{j} \Omega k} d\Omega$$

**Table 5.** Properties of the discrete time Fourier transform.

Property	Time Domain	Frequency Domain
Periodicity	-	$X(\Omega) = X(\Omega + 2\pi)$
Symmetry	Real-valued	$X(-\Omega) = X^*(\Omega)$
Linearity	$\alpha x_1[k] + \beta x_2[k]$	$\alpha X_1(\Omega) + \beta X_2(\Omega)$
Time shifting ( $k_0$ : integer)	$x[k - k_0]$	$e^{-\text{j} \Omega k_0} X(\Omega)$
Time differencing	$x[k] - x[k - 1]$	$(1 - e^{-\text{j} \Omega}) X(\Omega)$
Frequency domain differentiation	$-\text{j} k x[k]$	$\frac{dX(\Omega)}{d\Omega}$
Time summation	$\sum_{n=-\infty}^k x[n]$	$\frac{X(\Omega)}{1 - e^{-\text{j} \Omega}} + \pi X(0) \sum_{m=-\infty}^{\infty} \delta(\Omega - 2\pi m)$
Convolution	$x_1[k] * x_2[k]$	$X_1(\Omega) X_2(\Omega)$
Multiplication <sup>†</sup>	$x_1[k] x_2[k]$	$\frac{1}{2\pi} X_1(\Omega) * X_2(\Omega)$

**Table 5.** Properties of the discrete time Fourier transform (cont'd.).

Property	Time Domain	Frequency Domain
Parseval's theorem	$E = \sum_{k=-\infty}^{\infty}  x[k] ^2$	$E = \frac{1}{2\pi} \int_0^{2\pi}  X(\Omega) ^2 d\Omega$

<sup>†</sup>The convolution is on the interval  $0 \dots 2\pi$ .

**Table 6.** Discrete time Fourier transforms of elementary signals.

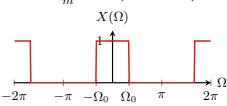
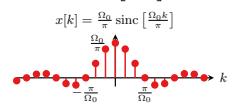
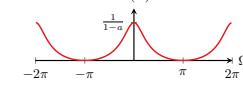
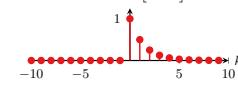
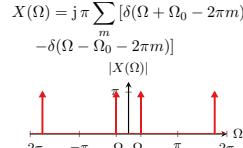
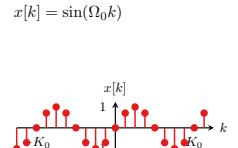
Time Domain $x[k]$	Fourier Transform $X(\Omega)$
<b>Periodic signals</b>	
$K_0$ -periodic ( $\Omega_0 = \frac{2\pi}{K_0}$ )	$X(\Omega) = 2\pi \sum_{n=-\infty}^{\infty} c_n \delta(\Omega - n\Omega_0)$ with $c_n = \frac{1}{K_0} \sum_{k=0}^{K_0-1} x[k] e^{-j n \Omega_0 k}$
<b>Constant</b> $x[k] = 1$	$X(\Omega) = 2\pi \sum_{m=-\infty}^{\infty} \delta(\Omega - 2\pi m)$
<b>Unit impulse</b> $x[k] = \delta[k]$	$X(\Omega) = 1$

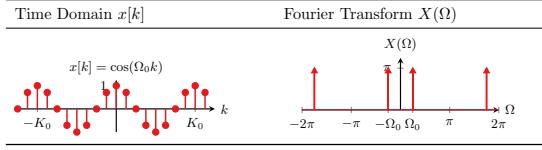
9

**Table 6.** Discrete time Fourier transforms of elementary signals (cont'd.).

Time Domain $x[k]$	Fourier Transform $X(\Omega)$
<b>Unit step</b>	
$x[k] = u[k]$	$X(\Omega) = \pi \sum_{m=-\infty}^{\infty} \delta(\Omega - 2\pi m) + \frac{1}{1 - e^{-j\Omega}}$
<b>Rectangular pulse</b>	
$x[k] = \text{rect}\left[\frac{k}{2N+1}\right]$	$X(\Omega) = \frac{\sin\left(\frac{(2N+1)\Omega}{2}\right)}{\sin\left(\frac{\Omega}{2}\right)}$
<b>Sinc function</b>	
$x[k] = \text{sinc}\left[\frac{k}{K_0}\right]$	$X(\Omega) = K_0 \sum_m \text{rect}\left(\frac{\Omega - 2\pi m}{\frac{2\pi}{K_0}}\right)$

10

Time Domain $x[k]$	Fourier Transform $X(\Omega)$
<b>Sinc function</b> (alternative parametrization)	
$x[k] = \frac{\Omega_0}{\pi} \operatorname{sinc}\left[\frac{\Omega_0 k}{\pi}\right]$	$X(\Omega) = \sum_m \operatorname{rect}\left(\frac{\Omega - 2\pi m}{2\Omega_0}\right)$ 
$x[k] = \frac{\Omega_0}{\pi} \operatorname{sinc}\left[\frac{\Omega_0 k}{\pi}\right]$	
	
<b>Exponential</b>	
$x[k] = a^k u[n] \quad ( a  < 1)$	$X(\Omega) = \frac{1}{1 - ae^{-j\Omega}}$ 
$x[k] = \operatorname{rect}\left[\frac{k}{2N+1}\right]$	
	
<b>Sine</b>	
$x[k] = \sin(\Omega_0 k)$	$X(\Omega) = j\pi \sum_m [\delta(\Omega + \Omega_0 - 2\pi m) - \delta(\Omega - \Omega_0 - 2\pi m)]$ 
	
<b>Cosine</b>	
$x[k] = \cos(\Omega_0 k)$	$X(\Omega) = \pi \sum_{m=-\infty}^{\infty} [\delta(\Omega + \Omega_0 - 2\pi m) + \delta(\Omega - \Omega_0 - 2\pi m)]$

**Table 6.** Discrete time Fourier transforms of elementary signals (cont'd.).**6 Discrete Fourier Transform**

$$X[l] = \sum_{k=0}^{K-1} x[k] e^{-j lk \frac{2\pi}{K}} \quad \text{and} \quad x[k] = \frac{1}{K} \sum_{l=0}^{K-1} X[l] e^{j lk \frac{2\pi}{K}}$$

**7 z-Transform**

$$X(z) = \sum_{k=0}^{\infty} x[k] z^{-k} \quad \text{and} \quad x[k] = \frac{1}{2\pi j} \oint X(z) z^{k-1} dz$$

**Table 7.** Properties of the z-transform.

Property	Time Domain	z-Domain
Linearity	$\alpha x_1[k] + \beta x_2[k]$	$\alpha X_1(z) + \beta X_2(z)$
Time shifting	$x[k - m]$	$z^{-m} X(z)$
Convolution	$x_1[k] * x_2[k]$	$X_1(z) X_2(z)$
Scaling	$a^k x[k]$	$X\left(\frac{z}{a}\right)$
Time difference	$x[k] - x[k - 1]$	$(1 - z^{-1}) X(z)$
Accumulation	$\sum_{m=0}^k x[m]$	$\frac{1}{1 - z^{-1}} X(z)$
Initial value theorem	-	$x[0] = \lim_{z \rightarrow \infty} X(z)$

**Table 7.** Properties of the z-transform (cont'd.).

Property	Time Domain	z-Domain
Final value theorem	-	$\lim_{k \rightarrow \infty} x[k] = \lim_{z \rightarrow 1} (z - 1)X(z)$

**Table 8.** Table of unilateral z-transform pairs for causal signals  $x[k]$  ( $x[k] = 0$  for  $k < 0$ ).

Time domain $x[k]$	z-Transform $X(z)$	ROC
<b>Unit impulse</b> $x[k] = \delta[k]$	$X(z) = 1$	all $z$
<b>Unit step</b> $x[k] = u[k]$	$X(z) = \frac{1}{1 - z^{-1}}$	$ z  > 1$
<b>Exponential</b> $x[k] = a^k u[k]$	$X(z) = \frac{1}{1 - az^{-1}}$	$ z  >  a $
<b>Ramp</b> $x[k] = ku[k]$	$X(z) = \frac{z^{-1}}{(1 - z^{-1})^2}$	$ z  > 1$
<b>Cosine</b> $x[k] = \cos(\Omega_0 k)u[k]$	$X(z) = \frac{1 - z^{-1} \cos(\Omega_0)}{1 - 2z^{-1} \cos(\Omega_0) + z^{-2}}$	$ z  > 1$
<b>Sine</b> $x[k] = \sin(\Omega_0 k)u[k]$	$X(z) = \frac{z^{-1} \sin(\Omega_0)}{1 - 2z^{-1} \cos(\Omega_0) + z^{-2}}$	$ z  > 1$
<b>Decaying cosine</b> $x[k] = a^k \cos(\Omega_0 k)u[k]$	$X(z) = \frac{1 - az^{-1} \cos(\Omega_0)}{1 - 2az^{-1} \cos(\Omega_0) + a^2 z^{-2}}$	$ z  >  a $
<b>Decaying sine</b> $x[k] = a^k \sin(\Omega_0 k)u[k]$	$X(z) = \frac{az^{-1} \sin(\Omega_0)}{1 - 2az^{-1} \cos(\Omega_0) + a^2 z^{-2}}$	$ z  >  a $

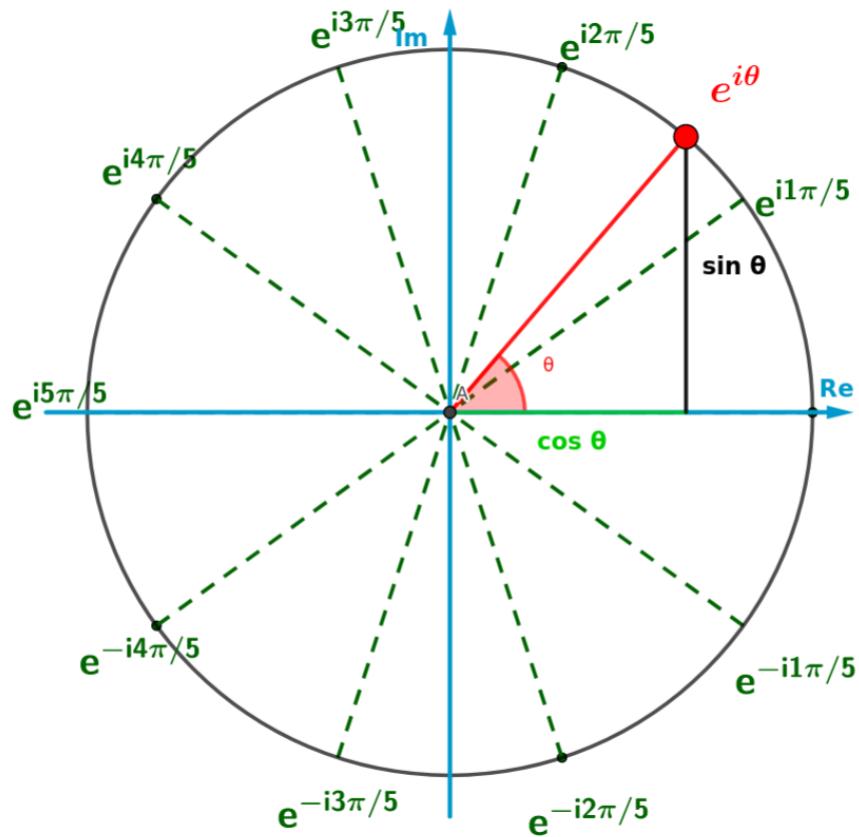


Figure 20: Complex unit circle. From ?

## Chapter 15

# Database Design I Samanfattning

### Resorcess

- <https://www.w3schools.com/sql/>
- <https://www.mysql.com/>
- <https://erdplus.com/>

## 15.1 Ch1. Databases and Database Users

- *Database*: A collection of related data.
- *Data*: Known facts that can be recorded and have an implicit meaning.
- *Mini-world*: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- *Database Management System (DBMS)*: A software package/ system to facilitate the creation and maintenance of a computerized database.
- *Database System*: The DBMS software together with the data itself. Sometimes, the applications are also included.

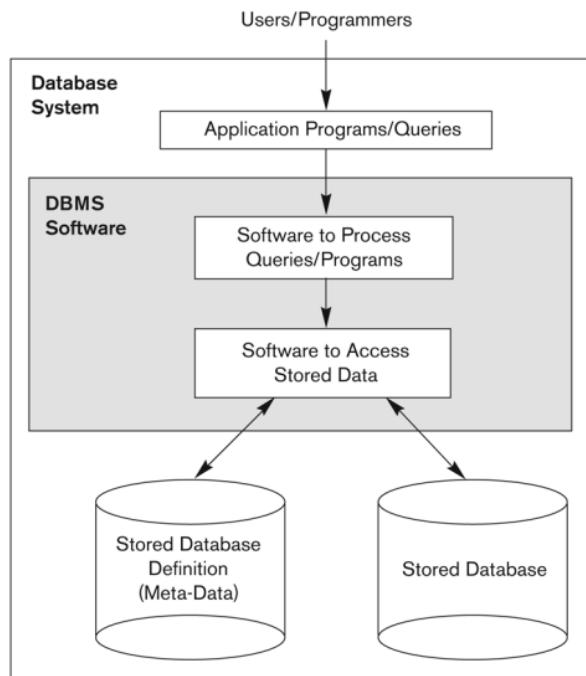


Figure 15.1: Database system. From ?

## 15.2 Ch3. Data Modeling Using the Entity-Relationship (ER) Model

Symbol	Meaning
rectangle	Entity
rectangle with a diagonal line	Weak Entity
diamond	Relationship
double diamond	Identifying Relationship
line with an oval	Attribute
line with a dashed oval	Key Attribute
line with a solid oval	Multivalued Attribute
line with three ovals connected by dots	Composite Attribute
line with a dashed oval	Derived Attribute
$E_1$ --- $R$ --- $E_2$	Total Participation of $E_2$ in $R$
$E_1$ --- $R$ --- $E_2$ , with $R$ having a '1' and $E_2$ having an 'N'	Cardinality Ratio 1: N for $E_1$ - $E_2$ in $R$
$R$ --- $E$ , with $R$ having '(min, max)'	Structural Constraint (min, max) on Participation of $E$ in $R$

Figure 15.2: NOTATION ER Diagram. From ?

### Definitions of ..

- *Entity*: Entities are specific things or objects in the mini-world that are represented in the database.
- *Entity set*: Each entity type will have a collection of entities stored in the database
- *Value sets*: Each simple attribute is associated with a value set
- *Weak Entity*: dependencies on other entity to exist
- *Relationship*: How entities are connected with one another

- *Relationship types*: Identifies the relationship name and the participating entity types and also identifies certain relationship constraints
- *Relationship Set*: The current set of relationship instances represented in the database
- *Recursive relationship*: Both participations are same entity type in different roles
- *Identifying Relationship*: Is the relationship for a entity to its owner
- *Attribute*: Data with the entity stores
- *Simple Attribute*: Containing a single atomic value
- *Key Attribute*: Unique attributes, can be used as keys
- *Multivalued Attributes*: Have multiple values like a list
- *Composite Attribute*: composed of several components
- *Derived Attributes*: Name(FirstName, MiddleName, LastName).

#### Relationship characteristics

- Double line: Needs to have the relationship
- Single line: Can have the relationship
- 1: Only has one
- N: Many has relationship (Used for one to many 1:N and N:1)
- M: Many relationship (Only used for many to many N:M and M:N)
- (1:1-samband): one to one
- (1:N-samband): one to many
- (N:M-samband): many to many
- Dotted key attribute: Is in combinations with the previous key

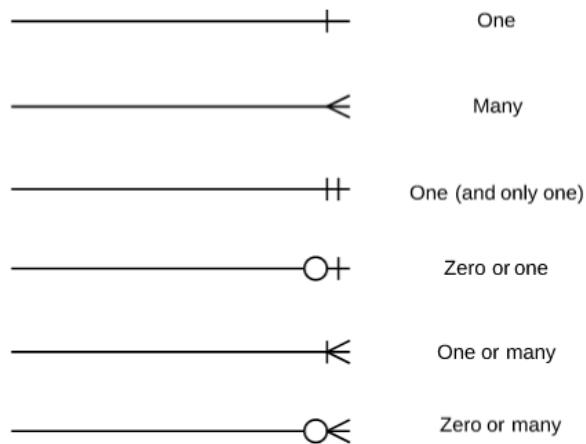


Figure 15.3: Crow foot notation. From ?

### 15.3 Ch4. Enhanced Entity-Relationship (EER) Modeling

EER includes all modeling concepts of basic ER, but also have additional concepts which are:

- *subclasses/superclasses*: additional meaningful subgroups of its entities, (ex: EMPLOYEE is further grouped into SECRETARY, ENGINEER, MANAGER)
- *Specialization/generalization*: is the process of defining a set of subclasses of a superclass, (ex: SECRETARY, ENGINEER, MANAGER is specialization of EMPLOYEE based upon job type)
  - *Disjointness Constraint*: an entity can be a member of at most one of the subclasses of the specialization
  - *Completeness Constraint*: Total specifies that every entity in the superclass must be a member of some subclass in the specialization/generalization

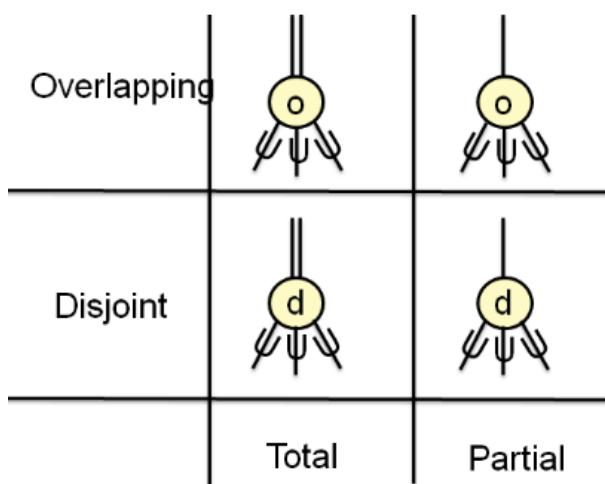


Figure 15.4: Overlapping, can be a member of multiple subclasses. Disjoint, can only be a member of at most one of the subclasses (Disjointness Constraint), Total, it must be a member of at least one (Completeness constraint). Partial, don't need to belong to any of the subclasses. From

## 15.4 Ch5. The Relational Data Model and Relational Database Constraints

### Definitions

- *Schema*: Is the description of the relation
- *Tuple*: is an ordered set of values (row)
- *Domain*: each attribute has a domain or a set of valid values
- *Relations*: Is a set of data (table)

### 15.4.1 Constraints

Constraints determine which values are permissible and which are not in the database

- **Inherent or Implicit Constraints**: These are based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute)
- **Schema-based or Explicit Constraints**: They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)
- **Application based or semantic constraints**: These are beyond the expressive

power of the model and must be specified and enforced by the application programs.

### Relational Integrity Constraints

There are three main types of (explicit schema-based) constraints that can be expressed in the relational model:

- *Domain constraint*: if one of the attribute values provided for the new tuple is not of the specified attribute domain
- *Key constraints*: if the value of a key attribute in the new tuple already exists in another tuple in the relation
- *Referential integrity*: if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
- *Entity integrity*: if the primary key value is null in the new tuple

## 15.5 Ch.13 Normalization

Youtube tutorial The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

### 15.5.1 Anomalies

Having redundant information creates some

- *Insertion anomalies*: Can not Insert unless there exist other data
- *Deletion anomalies*: By deleting an objects other objects will be deleted
- *Modification/Update anomalies*: An update result in updating many other objects attribute

### EXAMPLE OF AN INSERT ANOMALY

- Consider the relation:
  - EMP\_PROJ(Emp#, Proj#, Ename, Pname, No)
- Insert Anomaly:
  - Cannot insert a project unless an employee is assigned to it.

### EXAMPLE OF AN DELETE ANOMALY

- Consider the relation:

- EMP\_PROJ(Emp#, Proj#, Ename, Pname,
- Delete Anomaly:
  - When a project is deleted, it will result in deleting all the employees who work on that project.
  - Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

### EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:
  - EMP\_PROJ(Emp#, Proj#, Ename, Pname, No\_hours)
- Update Anomaly:
  - Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.

Note that it is not the relation *EMP\_PROJ* who has the Anomaly, it is *PROJ*.

### 15.5.2 Normal forms

Database Normalization Basics Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

- 1NF: All attributes depend on the key. Explanation 1NF.
  - Requirements:
    1. Each table cell should contain a single value, no multi-value or composite attributes.
    2. Columns which contain sets of values or nested records are not allowed.
  - Steps:
    - \* Eliminate duplicative columns from the same table.
    - \* Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).
- 2NF: All attributes depend on the whole key. In other words there should not be any partial dependencies, meaning that a attribute depends on a key constructing a composite primary key.

There for dose all attribute on the entire composite key. This is especially relevant for many to many reflations since a attribute can depend only on one foreign key but the primary key is the composition of the two foreign keys. Explanation 2NF.

- Requirements:
    1. Meet the requirements of 1NF.
    2. Not have duplicates in any table, if present, normalize further that table. Create relations between these new tables that are created and use foreign keys between them in order to indicate a relation is present.
  - Steps
    - \* Meet all the requirements of the first normal form.
    - \* Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
    - \* Create relationships between these new tables and their predecessors through the use of foreign keys.
  - 3NF: All attributes depend on nothing but the key. In other words does there not exist any attribute with depend on a other attribute or key witch is not the primary key. Explanation 3NF.
    - Requirements:
      1. Meet the requirements of 2NF.
      2. Have no transitive functional dependencies.
    - Steps:
      - \* Meet all the requirements of the second normal form.
      - \* Remove columns that are not dependent upon the primary key.
  - 4NF: (will not look at)
  - BCNF: (will not look at)
- primary key, full functional dependency Functional Dependencies FD
- Transmit dependencies FD1 FD2 FD3 ..

## 15.6 Ch.9

### ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 Relation Types
- Step 4: Mapping of Binary 1:N Relationship Types.
- Step 5: Mapping of Binary M:N Relationship Types.
- Step 6: Mapping of Multi-valued attributes.
- Step 7: Mapping of N-ary Relationship Types.

### Mapping EER Model Constructs to Relations

- Step 8: Options for Mapping Specialization or Generalization.
- Step 9: Mapping of Union Types (Categories).

## 15.7 Ch.6 SQL

### 15.7.1 Terminology

- *Table*
- *Row*
- *Column*
- *SQL schema*: Identified by a schema name and includes an authorization identifier and descriptors for each element
- *Schema elements*: includes the document element (the top-level element) in a schema definitionTables, constraints, views, domains, and other constructs
- *Catalog*: Named collection of schemas in an SQL environment.

- *Base tables (base relations)*: Relation and its tuples are actually created and stored as a file by the DBMS
- *Virtual relations (views)*: Created through the CREATE VIEW statement. Do not correspond to any physical file.
- *Aliases*: an alternative name to the relation (table). Declared by AS
- *Tuple*: a row value

### 15.7.2 SQL types

- Numeric
  - Integer numbers: INTEGER, INT, SMALLINT
  - Floating-point: FLOAT or REAL, DOUBLE PRECISION
- Character-string
  - Fixed length: CHAR(n), CHARACTER(n)
  - Varying length: VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n)
- Bit-string
  - Fixed length: BIT(n)
  - Varying length: BIT VARYING(n)
- Boolean
  - TRUE or FALSE or NULL
- DATE
  - YEAR, MONTH, DAY
- Additional data types
  - Timestamp: DATE, TIME
- INTERVAL

### 15.7.3 SQL examples

```
-- Creating tables

CREATE TABLE T1 (C1 VARCHAR(20), C2 VARCHAR(10), C3 INT);
INSERT INTO T1 VALUES ("a", "x", 1);
INSERT INTO T1 VALUES ("a", "y", 5);
INSERT INTO T1 VALUES ("b", "z", 2);
INSERT INTO T1 VALUES ("c", "z", 2);
INSERT INTO T1 VALUES ("d", "u", 3);

CREATE TABLE T2 (C1 VARCHAR(20), C2 VARCHAR(10), C3 INT);
INSERT INTO T2 VALUES ("a", "z", 4);
INSERT INTO T2 VALUES ("a", "x", 3);
INSERT INTO T2 VALUES ("b", "z", 2);
INSERT INTO T2 VALUES ("b", "x", 1);
INSERT INTO T2 VALUES ("c", "u", 2);

-- Select statements

SELECT COUNT(*)
FROM T1, T2;
-- T1 and T2 have 5 rows each there for is the result 5*5=25

SELECT Count(*)
FROM T1, T2
WHERE T1.C1=T2.C1;
-- 2+2+2+1=7

SELECT DISTINCT T1.C1
FROM T1, T2
WHERE T1.C2=T2.C2;
-- [a,b,c,d] (if the first is distinct) c,b,a,d (is the output)

SELECT T1.C2, AVG(T1.C3*T2.C3)
FROM T1, T2
WHERE T1.C2 = T2.C2
GROUP BY T1.C2;
-- [[u, 6],[x, 2],[z, 6]]

SELECT T1.C2, SUM(T1.C3*T2.C3)
FROM T1, T2
WHERE T1.C2 = T2.C2
GROUP BY T1.C2
HAVING SUM(T1.C3*T2.C3)>10
-- [z, 24] since u=6<10 and x=4<10 and z=2*4+2*2+2*4+2*2=24

SELECT count(*)
FROM T1 INNER T2 ON (T1.C1 = T2.C1);
-- Is an incorrect sql statement INNER JOIN is probably the right way

SELECT T1.C1
FROM T1 INNER JOIN T2 ON (T1.C1 = T2.C1)
WHERE (T1.C3>2);
```

```
-- [a, a] since there is only one a in T1 where T1.C3>2 and there is 2 a in T2
SELECT T2.C2
FROM T1 LEFT OUTER JOIN T2 ON (T1.C2 = T2.C2);
-- [x,x,NULL,z,z,z,u] the NULL comes from T1.C2="y"
```

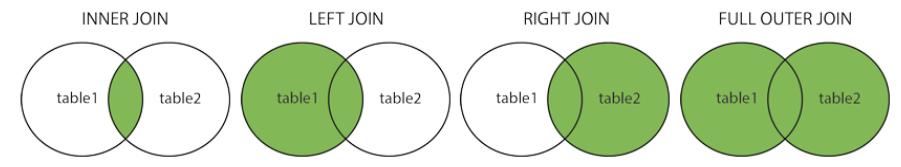


Figure 15.5: SQL JOIN statements. From

#### 15.7.4 Constraints

- **Key constraint:** A primary key value cannot be duplicated.
- **Entity Integrity Constraint:** A primary key value cannot be null.
- **Referential integrity constraints :** The “foreign key “ must have a value that is already present as a primary key, or may be null.

## 15.8 Ch.20 Transaction

### Terminology

- *Transaction*: Describes local unit of database processing
- *Transaction processing systems*: Systems with large databases and hundreds of concurrent users. Require high availability and fast response time
- *Single-user DBMS*: At most one user at a time can use the system
- *Multi-user DBMS*: Many users can access the system (database) concurrently
- *Multiprogramming*: Executes commands from one process, then suspends that process and executes commands from another process, etc.
- *Interleave processing*: Interleaving is a process or methodology to make a system more efficient, fast and reliable by arranging data in a noncontiguous manner.
- *Parallel processing*: Parallel computing is a type of computation in which many calculations or processes are carried out simultaneously.
- *Read-only transaction*: (read\_item(X))
- *Read-write transaction*: (read\_item(X) and write\_item(X))
- *Read set of transaction*: Set of all items read
- *Write set of a transaction*: Set of all items written
- *Concurrency control*: Transactions submitted by various users may execute concurrently
- *The lost update problem*: Occurs when two transactions that access the same database items have operations interleaved [https://www.youtube.com/watch?v=C\\_J6K8DodS8](https://www.youtube.com/watch?v=C_J6K8DodS8)
- *The temporary update problem*: Reading and updating from one transaction then read and write from another transaction. Then when reading another item we go back to the previous state. The problem is the dirty read of write then read from one transaction to another 6. <https://www.youtube.com/watch?v=d4ZiyuriOL0>
- *The incorrect summary problem*:  $T_3$  reads X after N is subtracted and reads Y before N is added; a wrong summary is the result (off by N) 7. <https://www.youtube.com/watch?v=IIcR3G5w4ZI>
- *The unrepeatable read problem*: Transaction T reads the same item twice. Value is changed by another transaction  $T'$  between the two reads. T receives different values for the two reads of the same item.
- *Recovery*:
  - *Committed transaction*: Effect recorded permanently in the database
  - *Aborted transaction*: Does not affect the database
  - *Transaction failures*:
    - \* Computer failure
    - \* Transaction or system error
    - \* Local error or exception conditions detected by the transaction
- *Transaction and System Concepts*
  - BEGIN\_TRANSACTION
  - READ or WR
  - END\_TRANSACTION

- COMMIT\_TRANSACTION
- ROLLBACK (or ABORT)
- *system log*: System log keeps track of transaction operations
- *ACID properties*
  - *Atomicity*: Transaction performed in its entirety or not at all
  - *Consistency preservation*: Takes database from one consistent state to another
  - *Isolation*: Not interfered with by other transactions
  - *Durability or permanency*: Changes must persist in the database
- *Schedule*: (or history) Order of execution of operations from all transactions
- *Serializable schedules*: Places simultaneous transactions in series
- *serial schedules*: Everything in series results limit of concurrency
- *Conflict equivalence*: Relative order of any two conflicting operations is the same in both schedules.
- *DBMS enforces protocols*: Set of rules to ensure serializability.

## Other SQL syntax

### Set operations

- UNION: The UNION operator is used to combine the result-set of two or more SELECT statements.
- EXCEPT: The SQL EXCEPT clause/operator is used to combine two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement
- INTERSECT: The INTERSECT clause in SQL is used to combine two SELECT statements but the dataset returned by the INTERSECT statement will be the intersection of the data-sets of the two SELECT statements

### Substring Pattern Matching

- LIKE: The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- BETWEEN: The BETWEEN operator selects values within a given range.

### Aliasing

#### 15.8.1 Transaction Support in SQL

- No explicit Begin\_Transaction statement
- Every transaction must have an explicit end statement
  - COMMIT
  - ROLLBACK
- Access mode is READ ONLY or READ WRITE
- Diagnostic area size option: Integer value indicating number of conditions held simultaneously in the diagnostic area
- Isolation level option
  - Dirty read
  - Nonrepeatable read

- Phantoms



# Appendices



$T_1$	$T_2$
<pre>read_item(<math>X</math>); <math>X := X - N</math>; write_item(<math>X</math>);</pre>	
<pre>Time ↓ read_item(<math>Y</math>);</pre>	<pre>read_item(<math>X</math>); <math>X := X + M</math>; write_item(<math>X</math>);</pre>

Figure 6: Temporary update problem. From ?

$T_1$	$T_3$
<pre>read_item(<math>X</math>); <math>X := X - N</math>; write_item(<math>X</math>);  read_item(<math>Y</math>); <math>Y := Y + N</math>; write_item(<math>Y</math>);</pre>	<pre>sum := 0; read_item(<math>A</math>); sum := sum + <math>A</math>; ⋮ read_item(<math>X</math>); sum := sum + <math>X</math>; read_item(<math>Y</math>); sum := sum + <math>Y</math>;</pre>

Figure 7: Incorrect summary problem. From ?



## Chapter 16

# Industrial Management

### Resorcess

- <https://hakan-kullven.webnode.se/>
- <https://www.youtube.com/watch?v=Q0o9S0q0Rr4>
- <https://www.youtube.com/watch?v=Kw-1nopchnA>

## 16.1 Verksamheter

### Företags former

	Enskild firma	Handelsbolag	Kommanditbolag	Ekonomisk förening	Aktiebolag
Juridisk person?	Nej	Ja	Ja	Ja	Ja
Antal ägare	1	Minst 2	Minst en komplementär, minst en kommanditdelägare	Minst 3	Minst 1
Ägarens betalningsansvar	Personligt	Solidariskt och personligt	Komplementär: obegränsat; kommanditdelägare: insats	Insats	Insats

Figure 16.1: Control without feedback. From ?

Note: om du med i ett handelsbolag eller kommanditbolag se till att du inte har ett kompanions avtal med juridiskt ombud? Så att du inte råkar illa ut.

### 16.1.1 Entreprenörskap

- Entreprenörskap
    - Entreprenörskap är en process, där individer identifierar möjligheter och baserar på deta går till handling.
    - Entreprenörsprocessen involverar en sökprocess
  - Entreprenörskap bygger i mycket på att det finns riskvilligt kapital
    - På en aktiemarknad
    - Lån och bidrag, crowd funding, affärsänglar, venture capital, industriella partners, finnansinstitut, bootstrapping...
- Sekventiell
  - Parallel
  - Interaktiv
  - agil
    - \* Scrum
    - \* Pulse
  - Andra metoder
    - \* Prototyping
    - \* CAD, Computer-aided design
    - \* Set-based design
    - \* QFD, Quality function development
    - \* DFMA, Design for manufacturing and assembly
    - \* FMEA, Failure mode and effect analysis

### 16.1.2 Inovation

- Produktutveckling
  - Alla aktiviteter som bidrar till utveckling och förbättring av bolagets erbjudande till sina kunder
  - Ofta bedrivs detta i projekt
- Utvecklingsprocessen

### 16.1.3 Verksamhetstyp

- **Råvarufängst:**  
Karakteristika: Stor personalrationalisering,

mekaniserat, kapitalintensivt, världsmarknadseller reglerade priser, kvalitetsortering, informations- och logistiksystem viktiga konkurrensmedel

Styrtal: Volym per tidsenhet, volym per arbetstimme, utbyte, kapitalomsättning, maksintutnyttjande, marginal, kostnad per ton - liter etc

- **Tillverkning:**

Karkatäristika: Ofta öga personalkostnader och kapitalkostnader, produktiviteten viktig, ställ- och ledtider är viktiga

Styrtal: Kapacitetsutnyttjande, produktivitet, ställtider, ledtider, materialutnyttjande, energiförbrukning, leveranssäkerhet

- Legotillverkning:

- Personalintensiv processtilverkning:

- Kapitalalintensiv processtillverkning:

- Sammansättning/motering:

- **Varudistribution:**

Karkatäristika: Mer logistik och system, kapacitetsutnyttjandet är viktigt, viktigt med stor kundkrets

Styrtal: Intäkt per kapacitetenhet, kostnad per kapacitetenhet, antal kunder per dag, antal leveranser per dag, bruttomarginal, kundnöjdhet

- Godstransport:

- Omlastning:

- Detaljhandel:

- **Kollektiva bastjänster:**

Karkatäristika: Styrs ofta av lagar eller andra regler, budgeten är viktig, kräver ofta stora inventeringar, kräver ofta stora volymer

Styrtal: Kostnad mot budget, antal kunder, antal klagomål

- Myndighetsutövning:

- Institutionella tjänster:

- Abonnentrelaterad förvaltning:

- **Servicesektorn:**

Karkatäristika: Kompetens och specialisering är viktig, arbetsintensiva verksamheter, hög och jämn beläggning är viktig, kundservice är en

konkurrensfaktor

Styrtal: Debiteringgrad, Kapacitetsutnyttjande, kundnöjdhet, klagomål per kund.

- **Lokala manuella tjänster:**

- **Kunskapsintensiva uppdragsverksamhet:**

- **Platsbunden konsumentservice:**

- **Uthyrning:**

- **Utbildning:**

- **Distanssupport:**

- **Artisteri:**

- **Spindleri:**

Karkatäristika: Långvariga affärsrelationer, standardiserat affärskoncept, bruttomarginaler viktiga, mer och mer IT

Styrtal: Marknadsandel, antal återkommande kunder täckningsbidrag per produkt

- **Förläggande:**

- **Kedjeorganiserande verksamhet:**

- **Mäkleri:**

## 16.2 Planering för verksamheten

### 16.2.1 Företagets strategiska mål

- **Mission:**

Vad företaget vill åstakommare

- **Vision:**

En ide om ett möjligt önskat framtida tillstånd

- **Affärsidé:**

En ide om hur ett företag ska kunna tjäna pengar på en affärsverksamhet

- **Strategier:**

Handlingsvägar på olika nivåer i företaget  
Lågkostnadsstrategi, differentieringsstrategi, fokusstrategi

- **Differentiation:** Compete on a large market e.g. Apple

- **Cost leadership:** Compete on price e.g. Ryanair

- **Focus:** Have a specific portion of the market e.g. Rolls Royce

### 16.2.2 Stratige modeler

#### SWOT-modellen



Figure 16.2: SWOT. From ?

#### Bostonmatrisen (BCG matrix)

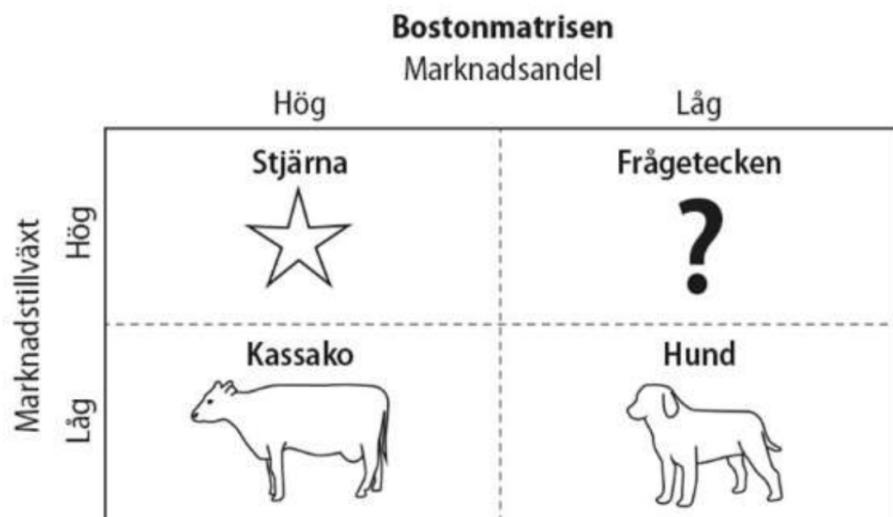


Figure 16.3: Bostonmatrisen. From ?

### Produktlivscykeln (Product life cycle)

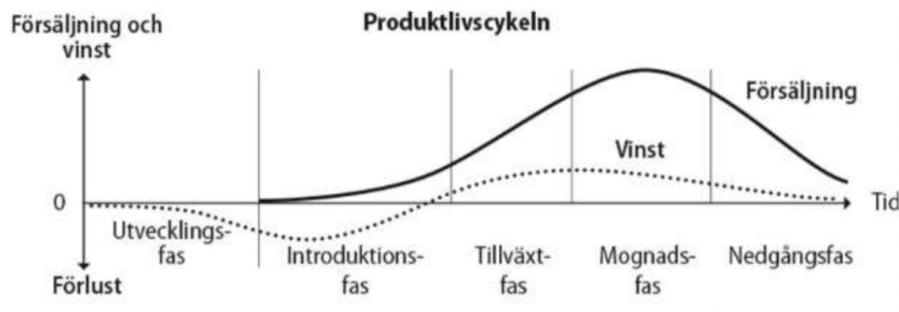


Figure 16.4: Produktlivscykeln. From ?

### Intressentmodellen (Stakeholder analysis)



Figure 16.5: Intressentmodellen. From ?

### 16.2.3 Ekonomisktystyrning (Management control)

- Styrning av personer
- Styrning av kultur
- Styrning av resultat
- Styrning av agerandet

### 16.2.4 Ekonomiskstyrning modeller

#### Det balanserade styrkortet (Balance scorecard)

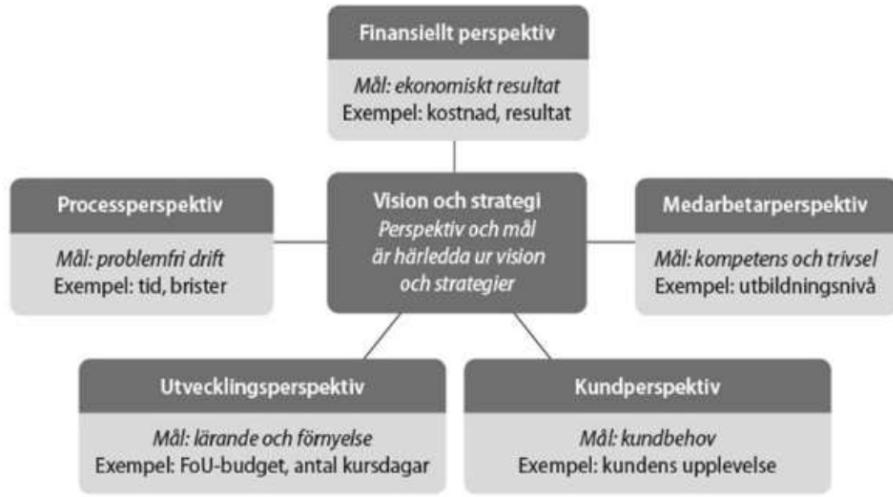


Figure 16.6: balanserade styrkort. From ?

#### Prognoser (Forecasting)

- Försök att se in i framtiden och vad som ska hända då
  - System för planering på olika nivåer
    - \* Aktivitetsplaner
    - \* Målstättning
    - \* Förväntade värden
  - Följa trender
    - \* Kritiska framgångsfaktorer
    - \* Nyckelindikatorer
    - \* Viktigaste drivarna

#### Budgetering (Budgeting)

Budgetering är en **Handlingsplan** för

- Samordning
- Styrning
- Kommunikation
- Uppföljning och kontroll

#### Faser (uppbryggnad eller nedbrytning)

1. Budgetdirektive ges
2. Budget ställs samman
3. Budgetar faställs

4. Genomförande
5. Uppföljning

#### Fördelarna med budgetering

- Kan påverka
- För delegering
- Skapar kostnadsmedvetande
- Informerar de anställda

#### Nackdelar med budgetering

- Tar mycket tid
- Slår aldrig in
- Begränsar möjligheter

### 16.3 Att agera på marknaden

#### 16.3.1 Typer av marknader

- **Fullständig (perfekt) konkurrens:** Många små företag med samma produkter som därmed konkurrerar med priset
  - Den osynliga handen
- **Monopolistisk konkurrens:** Väldigt vanlig. Företagen han liknande produkter där vissa kunder föredrar produkter

från enna företaget av diverse skäl och därmed konkurerar inte helt och hållt på pris.

- **Oligopol:**

Ett fåtal företag som kontrollerar marknaden.  
Kan komma överens om pris

- Varning för karteller

- **Monopol:**

Endast ett företag som kontrollerar marknaden och kan sätta priset själv.

- **Marknadsekonomi:**

Det ovanför är typer av marknadsekonomier.

- **Planeconomii:**

Förekommer oftas i delar av samhälle, som försvaret.

- **Blandekonomi:**

Kombination av marknadsekonomi och planekonomi.

### 16.3.2 Marknadsföring

#### Nätverssynsättet

- **Bygga upp förtroenden:**

- **Kundlojalitet:**

- **Eftermarknad:**

Marknad för service och underhåll, reservdelar, tilläggsförsäljning, utbildning med mera till följd av tidigare försäljning av kapitalvaror och industriella produkter

- **Strategiskt partnerskap:**

Konceptet bygger på att båda parter önskar

förstärka en ömsesidig samverkan.

- **Relationsmarknadsföring:**

inbefattar att företag samlar in stora mängder data i olika relationsmarknadsföringsprogram

- **Marknadsföringsplan:**

Den plan för att planera det följande konsepten

#### Konkurrensmedel, 4P

- **Produkt:** Varumärken

- **Pris**

- **Plats (Distribution)**

- Råvaruindustri > Tillverkningsindustri > Grossist > Detaljist > konsument

- **Promotion (reklam)**

### 16.3.3 Strategisk prissättning

- **Värdebaserad:** (kundbaserad)

Vad kunden är berädd att betala för den

- **Marknadsbaserad:** (konkurrentbaserad)

Hur mycket den ska kostnad om man jämför med marknaden

- **Kostnadsbaserad:**

Hur mycket företaget behöver att den kostar så det går runt.

Oftast är det en kombination av de tre.

Producentmarknad (B2B, Business to Business)

### 16.3.4 Tjänsteorientering

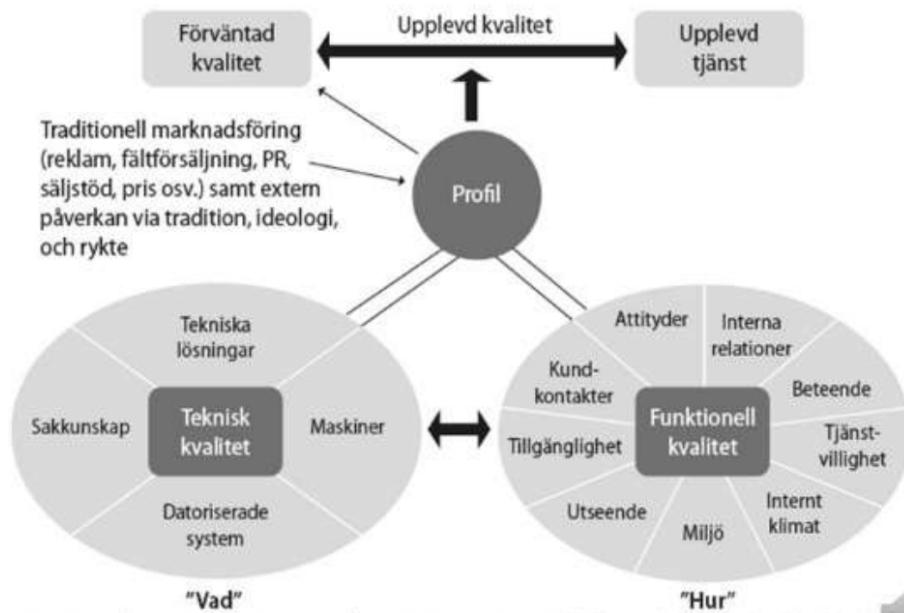


Figure 16.7: Tjänsteorientering. From ?

## 16.4 Organisera verksamhetern

### 16.4.1 Typer av hierarkiska organisationer

- **Linjeorganisation:** (U-form)

Har den mest tydliga hierarkiska organisationen med chefer och under chefer sedan arbetare

- **Linje-staborganisation:**

Är som Linjeorganisation men det har en eller flera så kallade *stab* som är en avdelning som inte ingår i beslutsledet utan mer finns som stöd för övriga enheter, och benämns då en linje/stabs-organisation.

### Operativa enheterna

Det finns också mer produkt fokuserade strukturer som (M-form) där Divisionerna är olika produkter.

En kombination av linje och produkt fokuserad struktur är MATRIX.

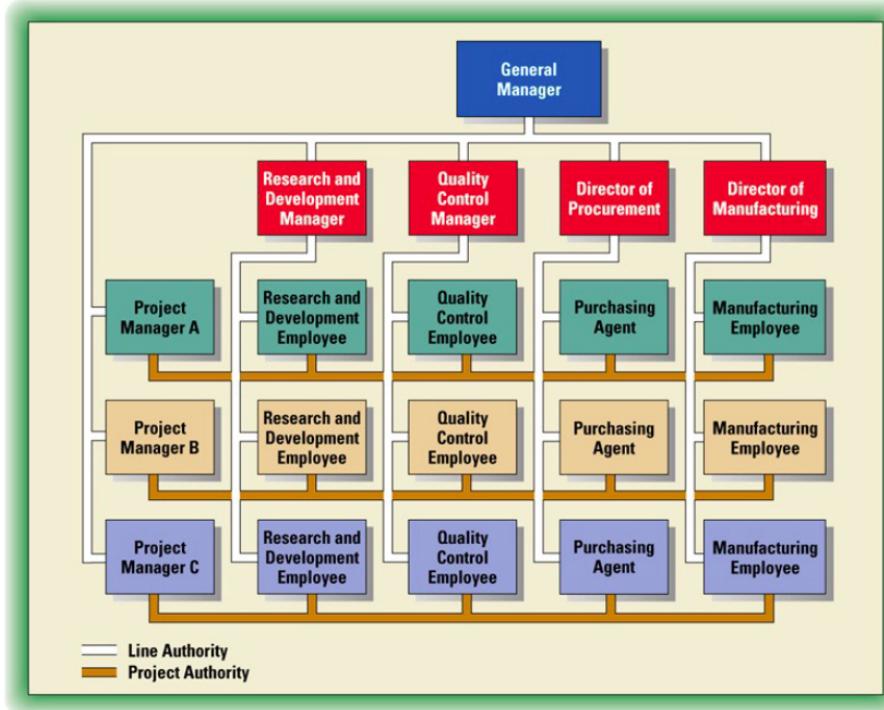


Figure 16.8: MATRIX. From ?

#### 16.4.2 Projektorganisation

- För en uppgift
  - Balansera resultat, slutdatum och resurser
- Involverar
  - Projektledare
  - Projektägare
  - Projektgrupp
  - Projektråd
  - Styrgrupp
  - Delprojekt
- Planering
  - Indelat i arbetspaket med milsöpar.
  - Planneras med *PERT* och *CPM*

#### 16.4.3 Rerulstatvärdesmetoden

Bedömd kostnad vid färdigt

$$= \frac{\text{Faktisk kostnad}}{\text{Resultatvärde}} \cdot \text{Budgeterad total kostnad}$$

Bedömd tid till färdigt

$$= \frac{\text{Planerad kostnad}}{\text{Resultatvärde}} \cdot \text{Budgeterad total tid}$$

#### 16.4.4 Produktion

##### Automationsnivå

- Manuell produktion
- Semi-automatisk produktion
- Automatisk produktion

##### Volym och variation

- Enstucksprocess
- Intermittent (batch) process
- Kontinuerlig process

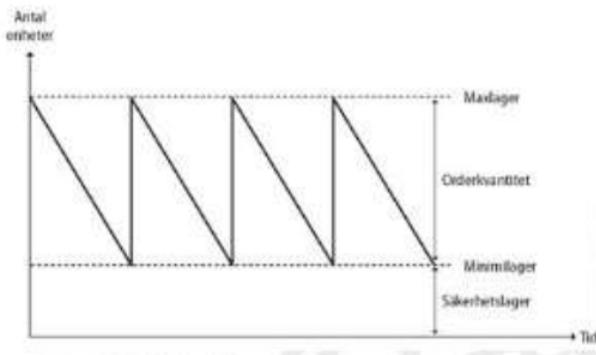
##### Layout i lokalen

- Fast position
- Funktionell layout: Placering utifrån aktivitet i produktionen
- Födesgrupper: celler
- Linjebaserad layout: produktorienterad placering

### 16.4.5 Lager Optimering

Optimal inköpskvantitet

$$= \sqrt{\frac{2 \cdot \text{Efertågan per år} \cdot \text{ordersäkostnad per order}}{\text{Lagerhållningskostnad per styck}}}$$



Figur 12 sid 126. Lagerinvärden

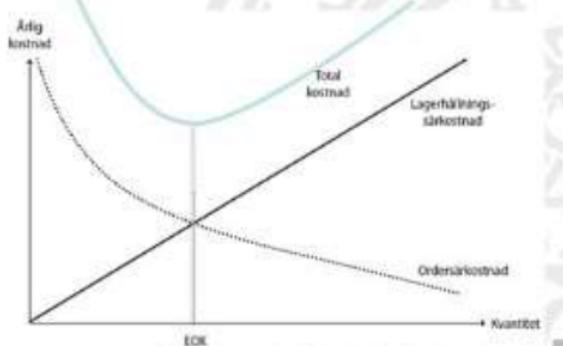


Figure 16.9: lager optimering grupper. From ?

Exempel WrapUp AB

TODO

### 16.4.6 Vad gör en ledare?

- Mintzberg

- Interpersonella rollen:  
Representant, vägledare och nätverkare
- Informationsrollen:  
Övervakare, informationsspridare och talesperson
- Beslutsrollen:  
Entreprenör, problemhanterare, resursfördelare och förhandlare

- Hamel

- Formulera och planera målförverkligande
- Motivera och samordna insatser
- Koordinera och styra aktiviteter
- Utveckla och tillsätta talanger
- Ackumulera och applicera kunskap
- Anskaffa och fördela resurser
- Skapa och värda relationer
- Balansera och tillmötesgå krav från investerare

### Bolagets ledning

- Bolagsstämma
  - Aktieägarna
  - Högsta beslutande organ
- Högsta ledningen
  - Styrelsen:  
Ansvarar för organisation och förvaltning
  - Företagsledningen:  
VD: löpande förvaltning
- Mellanchefer
- Operativa chefer
  - Första linjens chefer
  - Arbetsledare

### Ansvar

- Räntabilitetsansvar (Investment center): (intäkt - kostnad)/ Kapital  
Exempelvis: CEO
- Resultatansvar (Profit center): Intäkt - Kostnad  
Exempelvis:
- Intäktsansvar (Revenue center): Intäkter och kostnad  
Exempelvis: Key accountant manager
- Kostnadsansvar (Cost center): Kostnad  
Exempelvis: Chef ekonomi ansvarig
- Timansvar (Hour center): Timmar  
Exempelvis: Lathe manager

## 16.5 Kalkyler för produkter och order

### 16.5.1 Begrepp i flödet

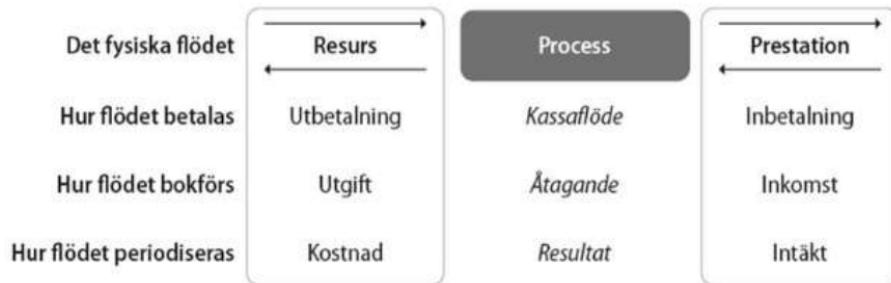


Figure 16.10: Begrepp i flödet. From ?

Inbetalning är när vi får pengarna på banken Inkomst är hur mycket våra kunder betala oss Intäkt?

### Periodisering

För att mäta produkts resultat så räknar vi resurserna som krävdes för produkten (kostnad) och jämför dessa med hur produkten presterade (intäkt). Periodisera betyder att det är gjämförbara.

### 16.5.2 Kalkylbegrepp



Figure 16.11: Kalkylbegrepp. From ?

- Direkt kostnad: Kostnad specifisk för produkten
- Indirekt kostnad: Kostnad som är delvis för den produkten (lathe)
- Särintäkter: Sär är relevant kostnad
- Samintäkter: Sam är irelevant kostnad.
- Rörlig kostnad: Förändrar sig med antal enhet
- Fast kostnad: Är alltid samma

### 16.5.3 Kalkylmodeller

Det finns kalkyler som görs före och efter beslut.



Figure 16.12: Kalkylmodeller. From ?

#### 16.5.4 Påläggskalkyl



Figure 16.13: Påläggskalkyl. From ?

#### Exempel Savox AB

Savox AB utför diverse lackeringsuppdrag för andra företag. För år 2018 har de budgeterat en direkt materialkostnad på 1 400 000 kronor, materialomkostnader på 280 000 kronor, direkta löner i produktionen på 400 000 kronor, indirekta produktionskostnader på 880 000 kronor för den planerade maskintiden 8 800 timmar, andra direkta produktionskostnader på 40 000 kronor, direkta försäljningskostnader på 100 000 kronor, och försäljning och administration på 1 200 000 kronor. Nu har företaget fått en order för lackering av en plastdetalj. För detta behövs färg för 5 000 kronor, 5 direkta arbetstimmar i produktionen vilka vardera kostar 1 200 kronor, 10 maskintimmar och en direkt försäljningskostnad på 1 000 kronor.

- Till vilket belopp uppgår företagets självkostnad enligt budget?
- Vilka pålägg och timkostnader bör företaget använda under året?
- Vad är kostnaden för den order de nu fått?

...

### 16.5.5 Aktivitetsbaserad kalkyl, ABC



Figure 16.14: ABC. From ?

#### Exempel Savox AB-C

Savox AB har bestämt sig för att använda en del ABC i sitt kalkylsystem. Jämför denna övning med den föregående, där de använder en påläggskalkyl. För år 2018 har de budgeterat en direkt materialkostnad på 1 400 000 kronor, materialhantering för 280 000 kronor med liter färg som drivare, produktion som kostar 400 000 kronor, maskinbearbetning som kostar 880 000 kronor för den planerade maskintiden 8 800 timmar, andra direkta produktionskostnader på 40 000 kronor, direkta försäljningskostnader på 100 000 kronor, försäljningsaktiviteter för 400 000 kronor med antal order som drivare, vilket beräknas till 400 för året, samt försäljning och administration för 800 000 kronor för vilket de inte finner en lämplig drivare och vilket de inte anser orsakas av order som sådana. Nu har företaget fått en order för lackering av en plastdetalj. För detta behövs 140 liter färg för 5 000 kronor, 5 direkta arbetstimmar i produktionen vilka vardera kostar 1 200 kronor, 10 maskintimmar och en direkt försäljningskostnad på 1 000 kronor.

- a. Till vilket belopp uppgår företagets självkostnad enligt budget?
- b. Vilka kostnader per drivare bör företaget använda under året?
- c. Vad är kostnaden för den order de nu fått?

...

### 16.5.6 Bidragskalkyl (Contribution Margin Costing)

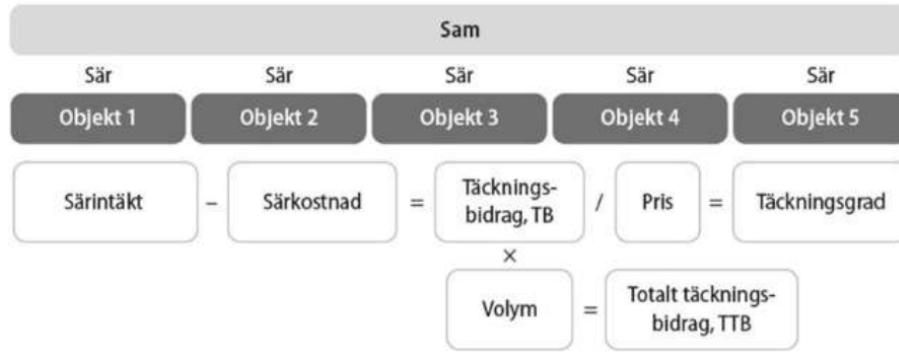


Figure 16.15: Bidragskalkyl. From ?

Särintäkt pris som betalt för produkten och särkostnaderna är vad det kostar att producera. räkningsgrad beteknas som bruttomarginal.

#### Beslut

Med bidragskalkylen så kan man använda den till att fatta vissa beslut som:

- Ska ordern accepteras?  
Tar endast hänsyn till beslutrelevanta, orderspecifika effekter på sikt
- Produktval vid en begränsning (en trång sektion)  
Den produkt som ger högst TTB, d.v.s högst TB per trång sektion, bör väljas
- Produktval vid flera begränsningar (flera tånga sektioner)  
Den kombination av produkter som ger högst TTB, enligt t.ex. linjär programmering, bör väljas
- Allmäna beslut om förändringar  
Tar endast hänsyn till beslutrelevanta, förändringsspecifika effekter på kort sikt

#### Exempel ChoicesAB

todo

Endast särkostnaden är relevant för beslutet. Tex så tar man inte in ens egna hyran i beräkningen för pris på en biobiljet är värt det eller inte.

### 16.5.7 Stegkalkyl

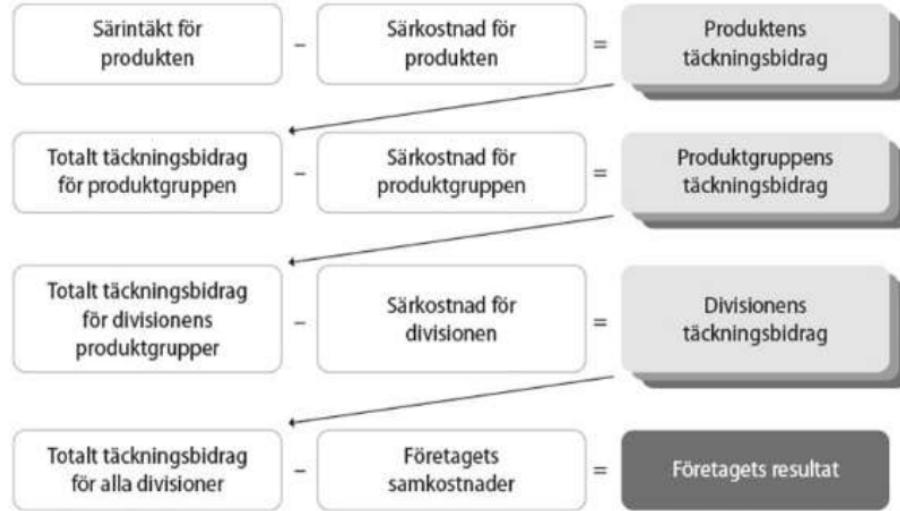


Figure 16.16: Stegkalkyl. From ?

### 16.5.8 Andra självkostnadskalkyler

Exempel

## 16.6 Investeringsbeslut

Grundprincip: Insättning på bank

- Ränta på ränta, slutvärde:  $\text{Kapital} \times (1 + r)^n$

- Nuvärde:  $\text{Kapital} \times \frac{1}{(1+r)^n}$

- Nuvärdesumma:  $\text{Kapital} \times \frac{1 - (1+r)^{-n}}{r}$

- Annuitet:  $\text{Kapital} \times \frac{r}{1 - (1+r)^{-n}}$



Figure 16.17: Typer av investeringar. From ?

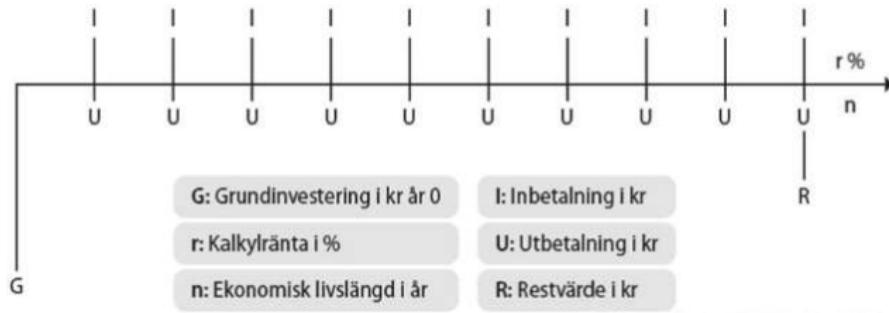


Figure 16.18: Kalkylbegrepp. From ?

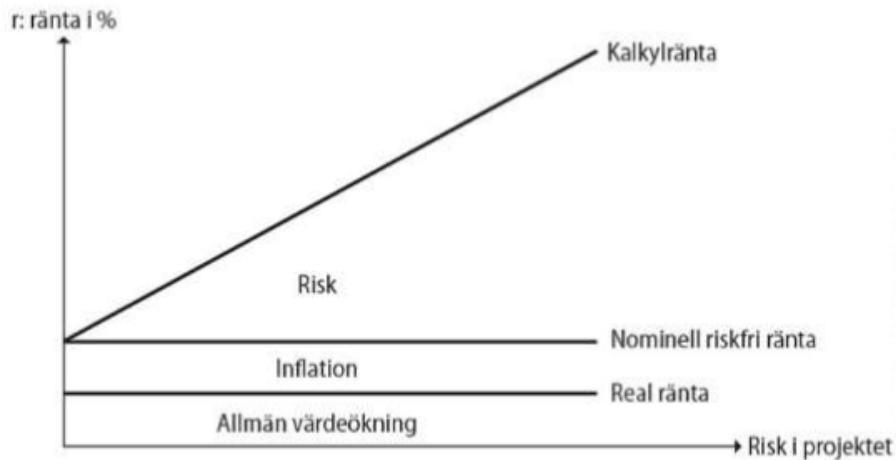


Figure 16.19: Kalkylräntan. From ?

## Metoder för investering

- Nuvärdemetoden
  - Positivt = lönsamt, ju högre desto bättre
  - Kan endast jämföra investeringar med samma ekonomiska livslängd
- Annuitetsmetoden
  - Positivt = lönsamt, ju högre desto bättre
- Internräntemetoden
  - Ju högre internränta desto bättre
  - Testa, eller dela grundinvesteringen med det årliga betalningsöverskottet
  - Kan bara använda om det också finns inbetalningar
- Återbetalningstid (pay-back)
  - $\sum$  inbetalningsöverskott = Grundinvesteringen, eller Grundinvestering/årligt inbetalningsöverskott
  - Ju kortare tid desto bättre
  - Problem; kortsiktigt, ingen ränta (i grundversionen)
- Investeringens räntabilitet
  - Genomsnittligt årligt inbetalningsöverskott divideras med genomsnittlig investering
  - Ju högre procent räntabilitet desto bättre
  - Problem: Tar inte hänsyn till ränta

shabloner, prusent (enkla) aktivitetbaserad kalkyl

---

Balanse teknig inventarie, sånt som inte ändrar sig

nuvärde annueitet återbetalning

Teknisk livslängd är så länge som en anläggning (eller del av anläggning) går att tekniskt/funktionellt att använda. Ekonomisk livslängd är den tid som det är ekonomiskt rationellt att använda anläggningen

kalkylränta är riks, real ränta och inflation real ränta hur det faktiskt förändras

återbetalningen metoden är inte riktigt rätt

räntan är inte bankräntan

## 16.7 Uppföljning av verksamheten

### Bokföring med T-konton

Balansräkning				Resultaträkning			
Tillgångar		Eget kapital & skulder		Utgifter		Inkomster	
Debet	Kredit	Debet	Kredit	Debet			Kredit
+             -		-             +					
Till exempel		Till exempel		Till exempel		Till exempel	
• Inventarier		• Aktiekapital		• Varuinköp		• Varuförsäljning	
• Lager		• Årets vinst		• Lokalhyra		• Arvoden	
• Kundfordringar		• Banklån		• Kontorsmaterial		• Hyresinkomster	
• Kassa, bank, bankgiro		• Leverantörsskulder		• Elenergi		• Ränteinkomster	

Figure 16.20: Bokföring med t-konton. From ?

notera att kredit och debit för balansräkningen respektive resultaträkningen ska vara samma. Debet betyder enbart vändstra sidan av räkningarna och Kredit är högra. Var man lägger tillgångar samt eget kapital & skulder respektive Utgifter och incomsenter är upptill företaget. Men skenerält så är det rällativt lika sätt man lägger upp det.

- SCO A2
- Kiosk är en ..
- FAST rapport
- årsredovisning
- årsmöte

### Årsredovisningens innehåll

Förvaltningsberättelse (FB): Väsentliga upplysningar

Resultaträkning (RR): Ekonomiskt utfall för perioden

Balansräkning (BR): Finansiell ställning vid periodens slut

Kassaflödesanalys: Orsaker till kassans förändring

Noter: Specifikationer till årsredovisningens poster i övrigt

Förslag till disposition av resultatet samt underskrifter: Från styrelsen

Revisionsberättelse: Rapport om årsredovisning och förvaltning. Uttalande om:  
1) styrelsens ansvarsfrihet, 2) fastställande av RR & BR, 3) förslag till disposition

Figure 16.21: Årsredovisningens innehåll. From ?

### Resultaträkning och balansräkning

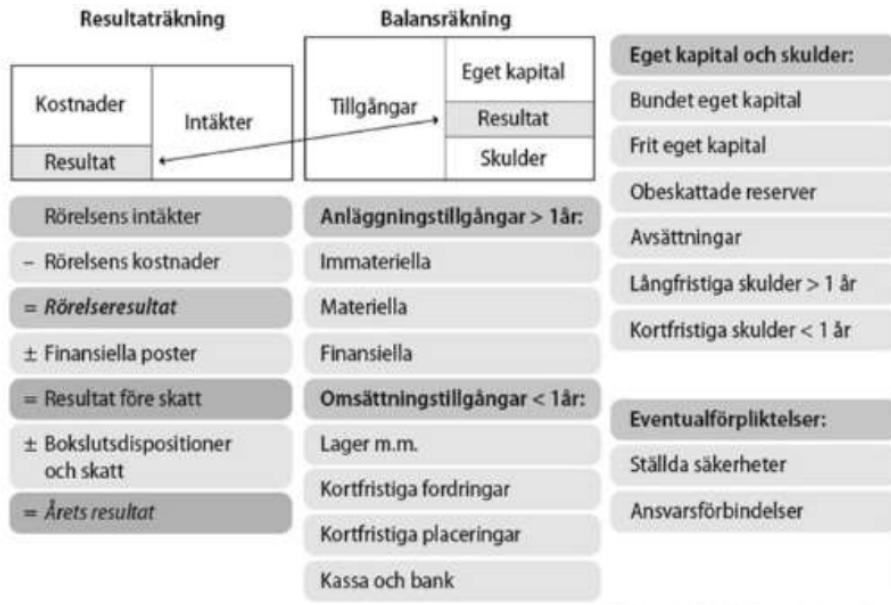


Figure 16.22: Resultaträkning och balansräkning. From ?

### 16.7.1 Redovisningens regler

- Lagar och regelverk
  - Bokföringslagen, Aktiebolagslagen, Årsredovisningslagen, Inkomstskattelagen, Skattebetalningslagen m.m.
  - Svensk kod för bolagsstyrning, Sarbanes-Oxley Act SOX
- Redovisningens principer
  - Rättvisande bild, "true and fair view"
  - Periodisering, fortlevand, jämförbarhets, försiktighet, realisering, osv
- Normgivare: God redovisningssed
  - Bokföringsnämnden, Rådet för finansiell rapportering, FAR SRS
  - IASB (Eng) International Accounting Standards Board med IFRS (International Financial Reporting Standards)
  - FASB (USA) Financial Accounting Standards Board

Revisor jobbar för aktieägarna. Det måste se till att det siffror som aktie ägarna får(redovisas) är rätt.

Namn och begrepp måste vara med och för årsredovisning

Det går att ändra räkenskapsår, men man måste anmäla att göra en sån ändring

### 16.7.2 Måt

- Omsättningstillgångar: Lägsta värdets princip (försiktighetsprincip). År tillgångar som förväntas säljas om ett år.
- Anläggningstillgångar: Långtids investeringar, mer än 1 år
- Värdering och bokslutsdispositioner:

EBITDA är ett annat finansielt mått.

- Soliditet:  $\frac{\text{Eget kapital}}{\text{Totalt kapital}}$
- Balanslikviditet:  $\frac{\text{Omsättningstillgångar}}{\text{Kortfristiga skulder}}$
- Kassalikviditet:  $\frac{\text{Omsättningstillgångar minus lager}}{\text{Kortfristiga skulder}}$

Lösamhet (räntabilitet)

- Räntabilitet på egen kapital,  $R_E$ : ägarnas avkastning  $\frac{\text{Årets resultat}}{\text{Genomsnittligt eget kapital}}$
- Räntabilitet på totalt kapitel,  $R_t$ : total avkastning  $\frac{\text{Resultat före skatt plus finansiella kostnader}}{\text{Genomsnittligt totalt kapital}}$

DuPoint-formeln

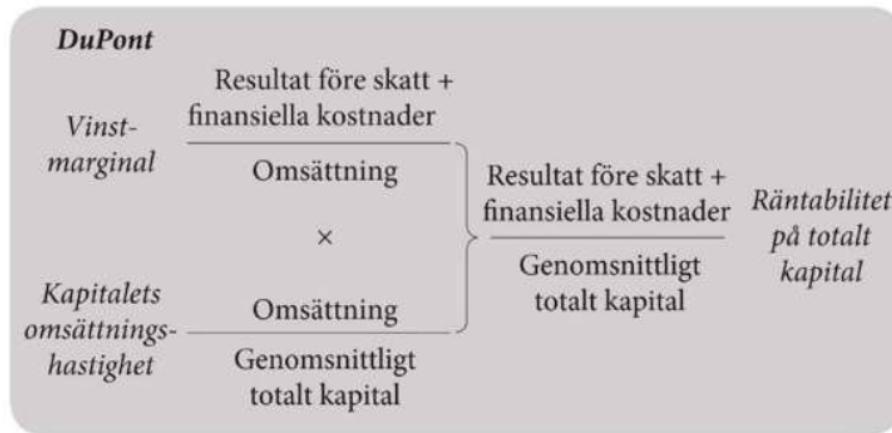


Figure 16.23: DuPoint-formeln. From ?

exempels

Vi har stabilt näringssiv i sverige. Vi har regler som skattar mycket när det går bra och lite när det går dårligt

Värde på tillgångar är upp till företaget att välja innom vissa sätta rammar som är samma för alla

Om man inte anmäler om konkurs när soliditet är för lågt så kommer man vara skildig till att betala ur egen plånbok.

Man har förökt redovisa hållbarhet, dock så har det inte kommit så långt med det.

## 16.8 Analysera förändring

Rörliga kostnader

- Rörlig kostnad: räck linje (Kronor totalt x volym)
- Avtagande rörliga kostnader: avtagande ökande kurva (kronor totalt x volym)
- Ökande rörliga kostnader: exponentiel kurva (kronor totalt x volym)

Fasta kostnader

- Fast kostnad: horizontelt sträck (kronor, totalt x Volym)
- Halvfast kostnad: ökar stegvis beroende på volym
- Halvfast kostnad: blir billigare per styck ju större volym men ökar vid en viss mängd, blir som en såg kurva

### Volymanalys



Figure 16.24: Volymanalys. From ?

#### Exempel: Fasta Rör AB

Företaget Fasta Rör AB säljer en produkt för 20 kronor per styck. Man hade i januari en total kostnad på 1 630 000 kronor, och i februari en total kostnad på 1 930 000 kronor. Inga andra skillnader noterades mellan perioderna än att volymen uppgick till 82 000 enheter i januari och till 102 000 enheter i februari.

- Direct costs: Costs that can directly be traced to the production of specific goods or services. Ex: Direct labour and material.
- Indirect costs: Costs that are necessary, but difficult to trace to the cost object. Ex: Overhead (rents, utilities, administration)
- Relevant costs: Costs that are affected by managerial choice in a certain business situation. Ex: Variable or marginal costs, incremental costs, specific costs, avoidable fixed costs, opportunity costs.
- Irrelevant costs: costs that won't be affected by a managerial decision. Ex: Sunk costs and costs which are same for different alternatives.
- Fixed costs: Costs that change in total when the volume changes. Ex: Rents, insurance, employee salaries, etc.

- Variable costs: Costs that don't change in total when the volume is changing. Ex: Sunk costs and costs which are same for different alternatives.

$$\text{Total Cost} = \text{Fixed Costs} + \text{Variable Costs}$$

Fundamental equation

$$\text{Operating Profit} = \text{Revenues} - \text{Total VC} - \text{Total FC}$$

## 16.9 Verksamhetens finansiering

Kapitalbehov



Figure 16.25: Kapitalbehov. From ?

Olika typer av Risker

- Verksamhetsrisker
  - Risker på grund av verksamheten
  - till exempel marknadsrisk, tillverkningsrisk, leverantörsrisker, och så vidare
- Finansiella risker
  - Risker till följd av hur investeringarna finansieras
  - till exempel risken för förändringar i valutakurserna, risken för förändringar i det funanaiella nettot, och risken för nya skatteregler.

*Kapitalrationalisering* går ut på att förbättra ett företags räntabilitet genom en förbättrad kapitalomsättning

### 16.9.1 Anläggningskapital (Fixed capital)

describes a company's investment in long-term assets - relatively permanent - e.g., for property, plant and equipment on the balance sheet

### 16.9.2 Rörelsekapital (Working capital)

is the amount of available capital that a company can readily use for day-to-day operations (measures a company's liquidity, operational efficiency, short-term financial health) - e.g., for inventory, accounts receivables, etc. *Rörelsekapital*

- Procentmetoden
  - Procentuell ändring i posterna p g a t.ex. omsättningsökning
- Genomsnittsmetoden
  - Kundfordringar = (kredittid i dagar/360 dagar) × försäljning

- Leverantörsskulder = (kredittid i dagar/360 dagar) × inköp
- Förråd = (minimilager + maximilager)/2 (eventuellt sen × kronor)
- Produkter i arbete i kronor = Antal produkter × halva produktkostnaden
- Färdiglager i kronor = Antalet produkter × produktkostnaden
- Resursmetoden och balansräkningsmetoden

### 16.9.3 Safety capital

Amount of capital to cover for variations in sales and the like.

### 16.9.4 Other terminology

*Resursmetoden*

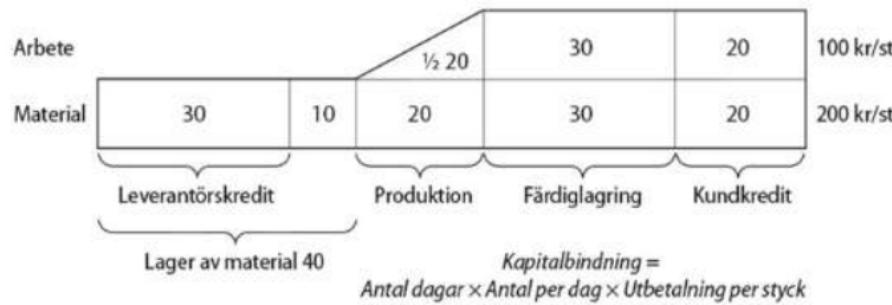


Figure 16.26: Resursmetoden. From ?

*Kreditmarknaden*

Främmande kapital kreditmarknad:

- Obligationslån
- Inteckningslån
- Reverslån
- Kontokredit
- Factoring
- Leasing
- Konvertibla skuldebrev

*Aktiemarknaden*

Eget kapital Aktiemarknad

- Sparade vinster
- Nyemission: fler aktie, för att göra det biligare att handla med aktien

### 16.9.5 Analyser

*kassaflödes analys*

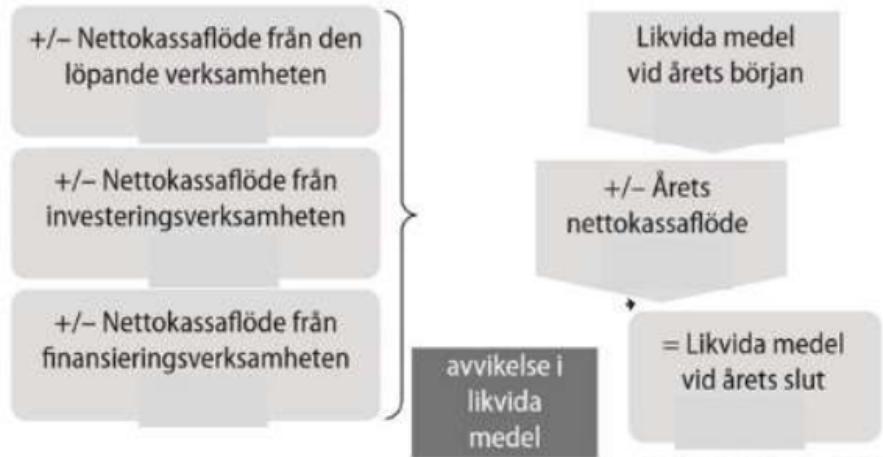


Figure 16.27: Kassaflödesanalysens komponenter. From ?

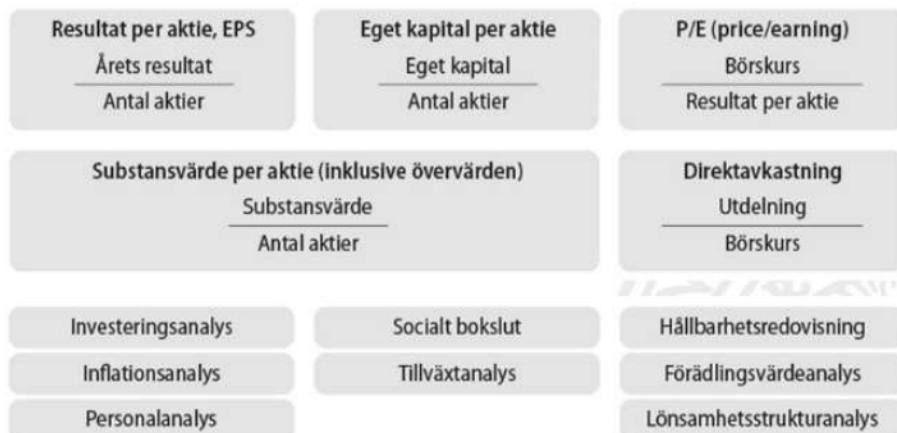
*Finansiell analys*

Figure 16.28: Kassaflödesanalys. From ?

Bank will get the money before the shareholder in case of bankrupsy



## Chapter 17

# Introduction to Computer Control Systems

### Resources

-

## 17.1 Intro

### 17.1.1 Control with and without feedback

**Control without feedback** is not as common in requires much more precise models and more sensors in order for it to work. In the case with a control for a segway without feedback requires a tone of sensors to measure every posable input with may change the angle of the handle.

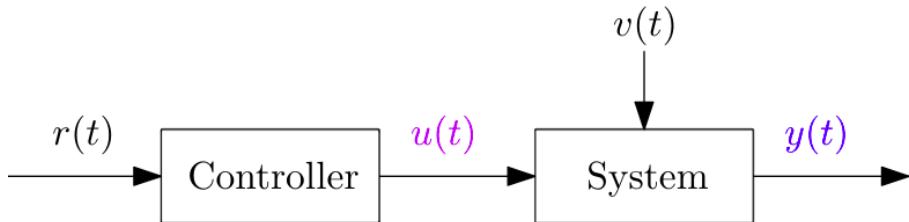
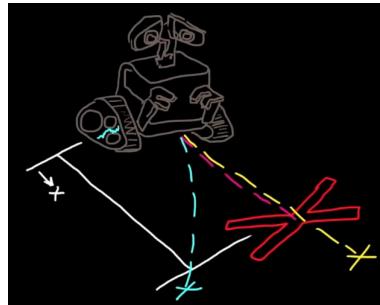
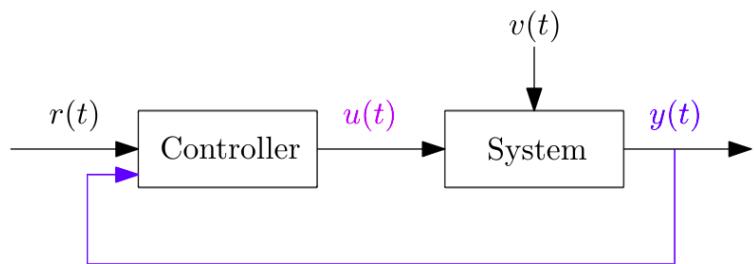


Figure 17.1: Control without feedback. From

**Control with feedback** is the most common. It is important to have feedback since the system rarely acts exactly like the model, for instance if you tell a robot to drive to a marking and without feedback you have to calculate the voltage needed to apply to the motors. Meanwhile with feedback you can adjust the steering and voltage in reference with the current position.



(a) Walle illustration. From



(b) Block diagram. From

Figure 17.2: Control with feedback. From

The controller part including the reference  $r(t)$  is Computer based and the Systems is generally physical.

### 17.1.2 Typical design requirements

1. **Stability:** Output  $y(t)$  will be *bounded* when reference  $r(t)$  is bounded
2. **Tracking:**  $y(t)$  tracks  $r(t)$  *quickly* with minimal *oscillations*
3. **Feasibility:** Controller should decide on feasible *input*  $u(t)$
4. **Robustness:** If a disturbance occurs, output  $y(t)$  should quickly *return* to the reference signal  $r(t)$ .

### 17.1.3 Laplace repetition

Transforms Solving ODE

### 17.1.4 Transfer function of a controller

$$Y(s) = G(s)U(s)$$

see image 17.2

The **Impulse response** is defined as follows

$$g(t) = \mathcal{L}^{-1}[G(s)]$$

and

$$y(t) = \int_0^t g(\tau) \delta(t - \tau) \equiv g(t)$$

### 17.1.5 Poles and zeros

The **zeros** tells us how much the  $G(s)$  oscillates and the **Poles** tells us how fast it converges. It is *stable* if the poles are negative.

Expressing  $G(s)$  via the poles

$$G(s) = \frac{B(s)}{A(s)} = \frac{b_1}{s + \sigma_1} + \dots + \frac{B_j(s)}{(s + \sigma_j)^2 + \omega_j^2} + \dots$$

Impulse response then becomes

$$g(\tau) = \sum_j b_j e^{Re\sigma_j t} = b_1 e^{\sigma_1 \tau} + \dots + b_j \sin(\omega_j \tau + \phi_j) e^{\sigma_j \tau} + \dots$$

### 17.1.6 Static gain of system

Assume stable system  $Y(s) = G(s)U(s)$  with input  $u(t)$  as a step function.  $U(s) = \frac{u_0}{s}$

Using the Final value theorem we get the static gain  $G(0)$

$$y_f = \lim_{t \rightarrow \infty} y(t) = \lim_{s \rightarrow 0} Y(s) = \lim_{s \rightarrow 0} sG(s) \frac{u_0}{s} = G(0)u_0$$

the static gain tells us what value dose the it converges

Static gain

System

$$G(s) = \frac{2}{s^2 + s + 1}$$

where the input to the system is a step response. What is the steady state value dose the output have?

From figure 17.2 we determine the output as  $y(t) = g(t)u(t)$

$$\lim_{t \rightarrow \infty} y(t) = \lim_{s \rightarrow 0} sY(s) = \lim_{s \rightarrow 0} sG(s)U(s) = \lim_{s \rightarrow 0} s \frac{2}{s^2 + s + 1} \frac{1}{s} = \frac{2}{1} = 2$$

## 17.2 PID control

<https://www.youtube.com/watch?v=UR0h0mjahp0>  
<https://www.youtube.com/watch?v=IB1Ir4oCP5k>

$$u(t) = \underbrace{K_p e(t)}_{P} + \underbrace{\int_{\tau=0}^t e(\tau) d\tau}_{I} + \underbrace{K_d \dot{e}(t)}_{D}$$

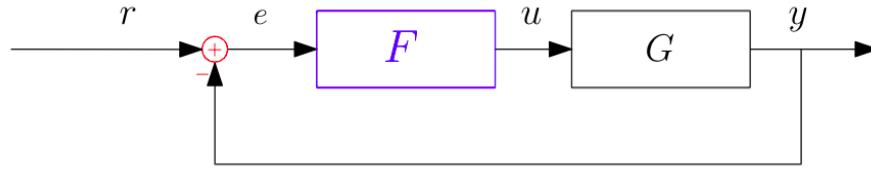


Figure 17.3: PID-controller. From

### 17.2.1 Feedback control based on error

$$e(t) \triangleq r(t) - y(t)$$

- current error:  $\propto e(t)$  (**Proportional**)  
The value is the error veted with a constant. How fast the system response.
- past error:  $\propto \int_{\tau=0}^t e(\tau) d\tau$  (**Integral**)  
The integral under the error from all past values. How fast the steady state error is removed
- change in error:  $\propto \dot{e}(t)$  (**Derivative**)  
The rate of change of the error. Faster response and better preforming.

using laplace we get:

$$\begin{aligned} U(s) &= K_p E(s) + K_i \frac{1}{s} E(s) + K_d s E(s) \\ &= \left( K_p + \frac{K_i}{s} + K_d s \right) E(s) \\ &= F(s) E(s) \end{aligned}$$

- System from  $u$  to  $y$ :  $G(s)$
- Closed-loop system from  $r$  to  $y$ :  $G_c(s)$
- Open-loop system from  $e$  to  $y$ :  $G_o(s) = G(s)F(s)$

The closed-loop is defined as:

$$G_c(s) = \frac{G(s)F(s)}{1 + G(s)F(s)}$$

We can define  $Y(s)$  as:

$$Y(s) = G_c(s)R(s)$$

The open-loop system is defined as:

$$G_o(s) = F(s)G(s)$$

Derive close loop

Derive  $G_c$  TODO

### Stationary error

$$\begin{aligned} e_f &= \lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} sE(s) \\ &= \lim_{s \rightarrow 0} s \frac{1}{1 + G_0(s)} \frac{r_0}{s} = \lim_{s \rightarrow 0} \frac{r_0}{1 + G(s)F(s)} \end{aligned}$$

$e_f$  approaches 0 if  $G(0)F(0) = \infty$ . (E.g. when  $F(s)$  contains  $\frac{1}{s}$ , i.e. integration)

### 17.2.2 Classical closed-loop design methods

- **Solve roots** of  $G_c(s)$  as a function of parameters
- **Routh's algorithm** for Stability check. You only get if the poles are on the left side or not.
- **Root-locus** plot for single parameter

#### Solve roots

##### Solve roots

The output is expressed as:

$$Y(s) = \frac{4s(s+2)}{s^3 + (3+8K_d)^2 + (2+8K_p)s + 8K_i} V(s)$$

what is the poles of the system, if a P-controller is used?

$$\begin{aligned} Y(s) &= \frac{4s(s+2)}{s^3 + (3+8K_d)^2 + (2+8K_p)s + 8K_i} V(s) \\ \text{set } K_i &= K_d = 0 \\ &= \frac{4(s+2)}{s^2 + 3s + 2 + 8K_p} V(s) \\ &\Rightarrow s^2 + 3s + 2 + 8K_p = 0 \\ &\Rightarrow s = -1.5 \pm \sqrt{0.25 - 8K_p} \quad \text{-using pq-formula} \\ &\Rightarrow 8K_p > 0.25 \Rightarrow K_p > 0.03125 \end{aligned}$$

#### Routh's algorithm

The denominator of the system can be constructed to find the pols, i.e. when the denominator is zero.

$$a_0s^n + b_0s^{n-1} + a_1s^{n-2} + b_1s^{n-3} + \dots = 0$$

**Step 1:** construct a table of the coefficient

$$\begin{array}{cccccc} a_0 & a_1 & a_2 & a_3 & \dots \\ b_0 & b_1 & b_2 & b_3 & \dots \\ c_0 & c_1 & c_2 & \dots \end{array}$$

**Step 2:**

$$c_k = \frac{b_0a_{k+1} - a_0b_k + 1}{b_0}$$

### Routh's algorithm

The output is expressed as:

$$Y(s) = \frac{4s(s+2)}{s^3 + (3+8K_d)^2 + (2+8K_p)s + 8K_i} V(s)$$

is the system stable, if a P-controller is used?

$$G_o(s) = \frac{4s(s+2)}{s^3 + (3)^2 + (2+8K_p)s} V(s)$$

$$\begin{array}{ccc} 1 & 2+8K_p & 0 \\ 1 & 2+8K_p & 0 \\ (24K_p + 6 - 8K_i)/3 & 0 & 0 \\ (8K_i) & 0 & \end{array}$$

$$\Leftrightarrow 24K_p + 6 - 8K_i > 0 \wedge K_i > 0$$

$$\Rightarrow 0 < K_i < \frac{3}{4} + 3K_p$$

**Root-locus**

Root-locus algorithm

$$G_0(s) = \frac{K}{s(s^2 + 2s + 2)}$$

$$G_c(s) = \frac{K}{s(s^2 + 2s + 2) + K \cdot \underbrace{1}_{Q(s)}}$$

**Step 1:** Solve the poles  $P(s) = s(s^2 + 2s + 2)$  Starting point ( $K=0$ )

$$\begin{aligned} z_{p1} &= 0 \\ z_{p2} &= -1 + j \\ z_{p3} &= -1 - j \end{aligned}$$

**Step 2:**  $Q(s) = 1$  Ending points  $Q(z_q) = 0 \Rightarrow$  no ending points ( $K \rightarrow \infty$ )

**Step 3:** Asymptotes number of asymptotes =  $\underbrace{\text{number starting}}_3 - \underbrace{\text{number ending}}_0$  the asymptotes

$\varphi$

$$\varphi_A = \frac{2q + 1}{\text{number of asymptotes}}$$

$$\begin{aligned} \varphi_A(q=0) &= \frac{2 \cdot 0 + 1}{3} \cdot \pi = \frac{\pi}{3} \\ \varphi_A(q=1) &= \frac{2 \cdot 1 + 1}{3} \cdot \pi = \pi \\ \varphi_A(q=2) &= \frac{2 \cdot 2 + 1}{3} \cdot \pi = \frac{5\pi}{3} = \frac{-\pi}{3} \end{aligned}$$

Intersection of the asymptotes

$$c = \frac{\sum_{p=1} z_p - \sum z_q}{\text{number of asymptotes}} = \frac{0 - 1 + j - 1 - j}{3}$$

see image 17.4

**Step 4:** Intersection with imaginary axis

$$\begin{aligned} s &= 6 + j\omega \\ 6 &= 0 \Rightarrow s = j\omega \\ s(s+1)(s+3) + k &= 0 \\ s^3 + 4s^2 + 3s + k &= 0 \\ \Leftrightarrow -j\omega^3 - 4\omega^2 + 3j\omega + K &= 0 \\ \Leftrightarrow \begin{cases} K - 4\omega^2 = 0 \\ \omega(\omega^2 - 3) = 0 \end{cases} &\Rightarrow \begin{cases} \omega = 0 \Rightarrow K = 0 \\ \omega = \pm\sqrt{3} \Rightarrow K = 12 \end{cases} \end{aligned}$$

**Step 5:** Intersection with real axis

$$\begin{aligned} k &= -s(s+1)(s+3) \\ \frac{\partial K}{\partial S} = 0 &\Leftrightarrow 3s^2 + 8s + 3 = 0 \quad \Rightarrow \text{No real solutions} \end{aligned}$$

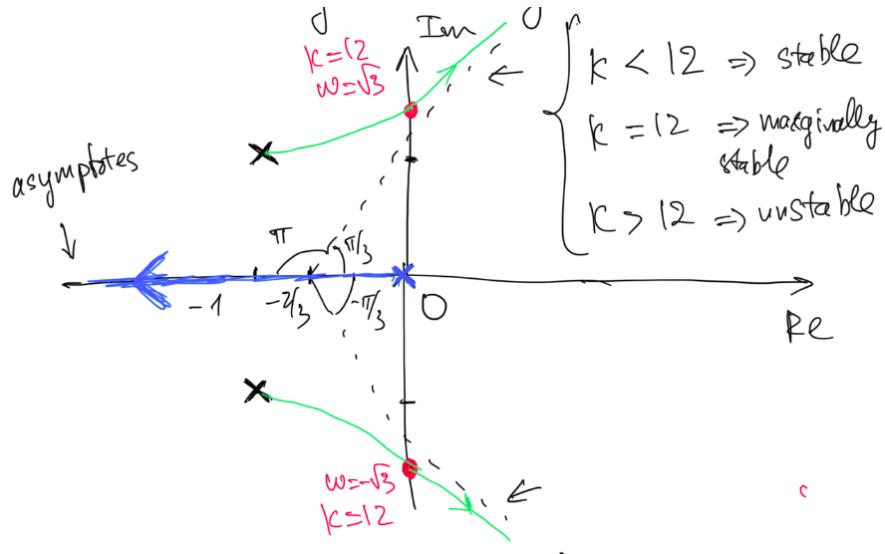


Figure 17.4: Root locus example

## 17.3 Good Controller?

### 17.3.1 Method 1: Frequency description of system

**Sine in/sine out** principle tells us that the frequency of a signal does not change when applying it to a system (the frequency response does not change the frequency of input signal).

**System amplitude- and phase plot** tells us how much the system amplifies the input signal, mean while the phase plot tells us how much it has been moved.

### 17.3.2 Tracking properties of closed-loop system

Any signal can be represented as a weighted sum of cosine- and sine signals

$$r(t) = \frac{1}{2\pi} \int R(i\omega) e^{i\omega t} d\omega$$

recall that  $G_c(s)|_{s=i\omega} = G_c(i\omega)$  and that

$$Y(i\omega) = G_c(i\omega)R(i\omega) = |G_c(i\omega)|e^{i\arg\{G_c(i\omega)\}}R(i\omega)$$

where ideally the magnitude  $|G_c(i\omega)| \approx 1$  and the phase  $\arg\{G_c(i\omega)\} = 0$

Performance metric in time domain: quickness: rise time  $T_r = t_{90\%} - t_{10\%}$  Damping: overshoot  $M = (y_{max} - y_f)/y_f$  Accuracy: static control error  $e_f = r_0 - y_f$

Performance metric in frequency domain: quickness: bandwidth  $\omega_B$  where  $|G_c(i\omega_B)| = |G_c(0)|/\sqrt{2}$  Damping: resonance peak level  $M_p = \max(|G_c(i\omega)|)$  Accuracy: static gain  $G_c(0)$

**Bode plot**

Example bode plot

$$G(s) = \frac{100(s+1)}{s(s^2 + 6s + 100)}$$

**Step 1:** determining the poles and zeros

Zeros :  $z_1 = -1$

Poles :  $p_0 = 0 \wedge p_1, p_2 = -3 \pm i\sqrt{91} = 10$

**Step 2:** sort poles and zeros

$\omega$	$ P_0 $	$ Z_1 $	$P_1$ and $P_2$
slope	0	-1	+1 -2

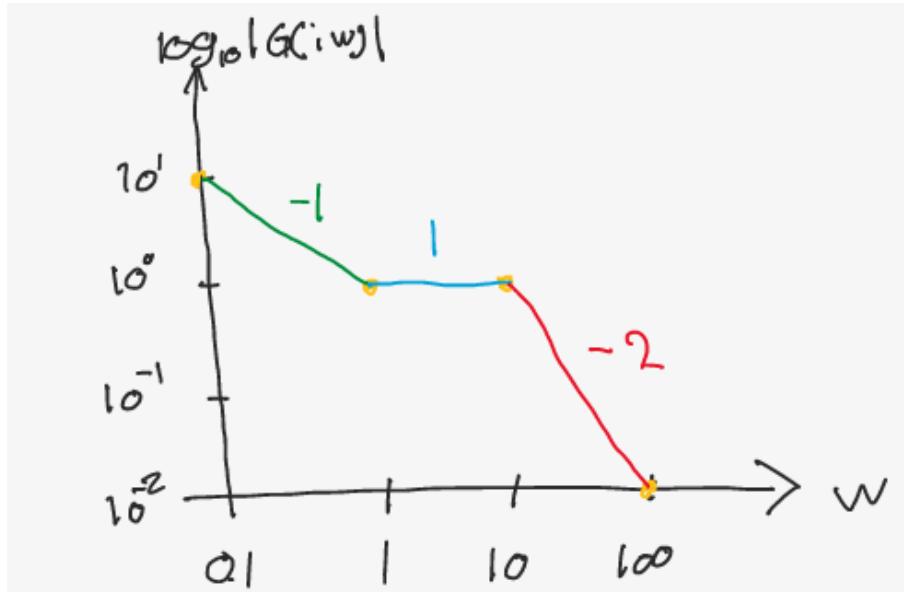
**Step 3:** draw  $\log_{10}(G(i\omega))$  starting at the first pole or zero at the origin see figure 17.5

Figure 17.5: example bode plot

**17.3.3 Method 2: Design via open-loop system**

It is easier to design using Open-loop system instead of closed loop with we did in Method 1.

Open-loop:  $G_o(s) = F(s)G(s)$

Closed-loop:  $G_c(s) = \frac{G_o(s)}{1 + G_o(s)}$

$G_c(s)$  stable  $\Leftrightarrow 1+G_o(s)$  no roots in right half-plane

**Nyquist curve**

$G_o(i\omega) = G(i\omega)F(i\omega)$ , over  $0 \leq \omega < \infty$

$$|G_c(i\omega)| = \frac{|G_o(i\omega)|}{|1 + G_o(i\omega)|} = \frac{|G_o(i\omega)|}{\underbrace{|G_o(i\omega) - (-1)|}_{\text{distance to } -1}}$$

Nyquist curve  $G_o(i\omega)$

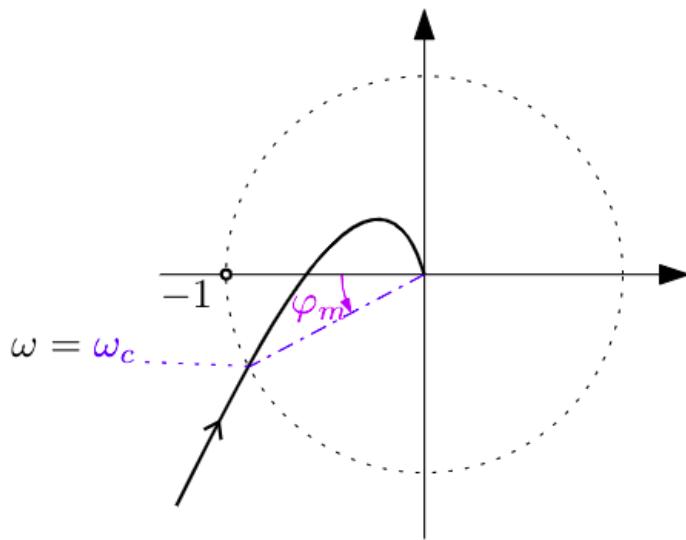


Figure 17.6: Nyquist curve. From

Note if the line is under  $-1$  then it is stable.

#### Crossover frequency and phase margin

Find  $\omega = \omega_c$  where  $|G_o(i\omega_c)| = 1$  and then define  
 $\varphi_m = \arg\{G_o(i\omega_c) + 180^\circ\}$

The Nyquist curve also tells us quickness and damping

- Crossover frequency  $\omega_c \Rightarrow$  bandwidth  $\omega_B$  of  $G_c$  (quickness)
- Phase margin  $\varphi_m \Rightarrow$  resonance peak  $M_p$  of  $G_c$  (damping)

Controller  $U(s) = F(s)(R(s) - Y(s))$  with three blocks:

$$F(s) = K \cdot F_{lead}(s)F_{lag}(s)$$

Design recipe:

1. Initialize with  $F_{lead}(s)F_{lag}(s) \equiv 1$  (P-control)
2. Find a  $K$  that yields stable closed-loop system (e.g. via Nyquist criterion)
3. Adjust  $F_{lead}(s)$  and  $F_{lag}(s)$  to improve frequency characteristics of  $G_o(s)$

Adjust frequency characteristics of open-loop system

$$\begin{aligned}\log |G_o(i\omega)| &= \log |G(i\omega)| + \log K + \log |F_{lead}(i\omega)| + \log |F_{lag}(i\omega)| \\ \arg G_o(i\omega) &= \arg G(i\omega) + 0 + \arg |F_{lead}(i\omega)| + \arg F_{lag}(i\omega)\end{aligned}$$

Nyquist curve

$$G_o(s) = \frac{K(s+10)}{s(s^2 + 3s + 50)}$$

Denominator:  $s(s^2 + 3s + 50)$  How many roots not LHP

**Step 1**

$$s(s^2 + 3s + 50) = 0 \Rightarrow \begin{cases} s = 0 \\ s = -\frac{3}{2} + j\frac{\sqrt{|9|}}{2} \\ s = -\frac{3}{2} - j\frac{\sqrt{|9|}}{2} \end{cases}$$

see image 17.7

$$\begin{array}{cccc} A & \rightarrow & B & \rightarrow \\ s = j\omega & & s = Re^{j\theta} & \\ \omega = 0^+ \rightarrow \infty & & R \rightarrow +\infty & \\ & & & s = j\omega \\ & & & \omega : -\infty \rightarrow 0^- \\ & & & \epsilon \rightarrow 0 \end{array}$$

**Step 2:** Investigate each part in open-loop transfer function

Part A:  $s = j\omega$  ( $\omega : 0^+ \rightarrow +\infty$ )

$$\begin{aligned} G_A(s = j\omega) &= \frac{K(j\omega + 10)}{j\omega(-w^2 + 3j\omega + 50)} = \frac{K(j\omega + 10)}{\omega(-3j\omega + j(50 - w^2))} \\ &= \frac{K(j\omega + 10)(-3\omega - j(50 - \omega^2))}{\omega(-9j\omega^2 + j(50 - \omega^2)^2)} = \frac{K(20 - \omega^2)}{9\omega^+(50 - \omega^2)^2} - j \frac{K(500 - 7\omega^2)}{\omega(9\omega^2 + (50 - \omega^2)^2)} \end{aligned}$$

$$\omega \rightarrow 0^+ : \quad G_A(j0^+) = \frac{K20}{50^2} - j\infty$$

$$\omega \rightarrow +\infty : \quad G_A(j\infty) = 0 - j0$$

Intersection of  $G_A(j\omega)$  with real axis

$$\begin{aligned} Im[G_A(j\omega)] &= 0 \Rightarrow w^2 - 3 = 0 \\ &\Rightarrow w = \sqrt{3} \end{aligned}$$

$$Re[G_A(j\sqrt{3})] = \frac{-K}{18}$$

Part B:  $s = Re^{j\theta}$   $R \rightarrow +\infty$

$$\begin{aligned} G_A(s = Re^{j\theta}) &= \frac{K(s+1)}{s^2(s+3)^2} = 0 \\ (R \rightarrow +\infty) \end{aligned}$$

Part C: and part A are asymmetric around real axis

Part D:  $s = \epsilon e^{j\theta}$   $\epsilon \rightarrow +\infty$

$$\begin{aligned} G_A(s = \epsilon e^{j\theta}) &= \frac{K(s+1)}{s^2(s+3)^2} = \infty \\ (\epsilon \rightarrow 0) \end{aligned}$$

Step 3: see image 17.8

Stable  $\Leftrightarrow [-1 + j0]$  is not enclosed in Nyquist plot

$$\begin{aligned} &\Leftrightarrow -1 < \frac{-K}{18} \\ &\Rightarrow K < 18 \end{aligned}$$

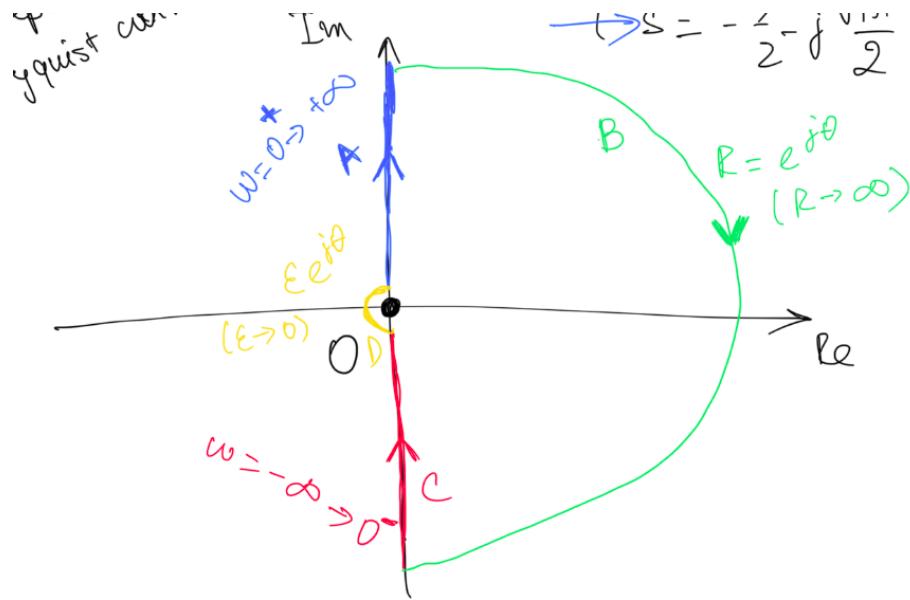


Figure 17.7: Nyquist curve example

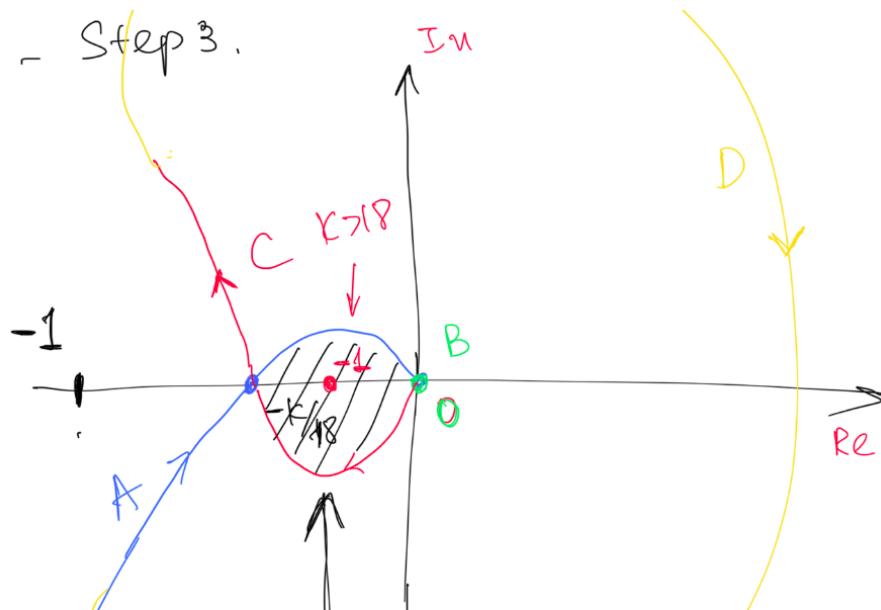


Figure 17.8: Nyquist curve example

The values for  $K$  can be calculated by (-Negative Intersection \*  $K < 1$ )

### Minimum phase systems

Definition: Minimum phase system

Among all systems  $G(s)$  with the same magnitude curve, the system with the least negative phase shift is a **Minimum phase system**

Note: A system is minimum phase  $\Leftrightarrow$  it has no poles nor zeros in the right half-plane, and contains no time delays.

### 17.3.4 General linear feedback control

$$U(s) = F_r(s)R(s) - F_y(s)Y(s).$$

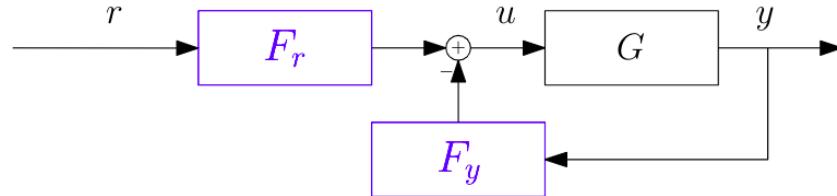


Figure 17.9: General linear feedback control. From

$$Y(s) = \frac{G(s)F_r(s)}{1 + G(s)F_y(s)}R(s)$$

Note:  $F_y(s) = F_r(s) \equiv F(s) \Rightarrow$  simple linear feedback

**Feedback with measureable disturbances and intermediate signals**

**Feedback with measureable disturbances** What if we added more sensors?

$$U(s) = F_r(s)R(s) - F_y(s)Y(s) + F_f(s)V(s).$$

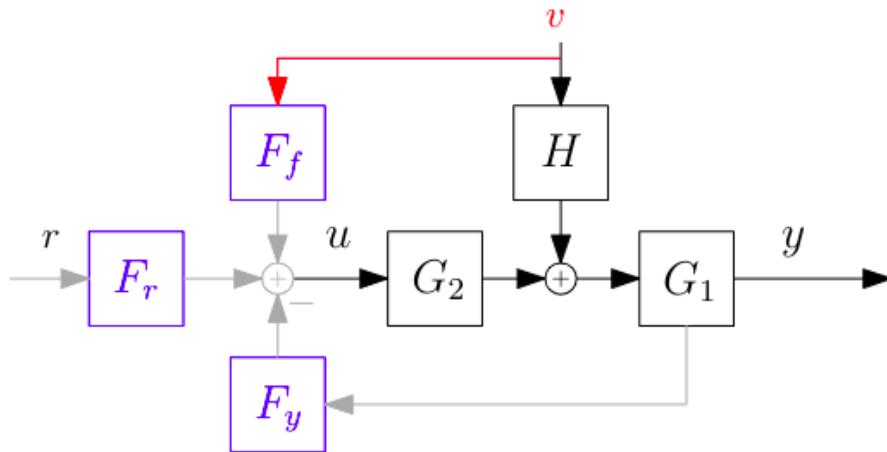


Figure 17.10: General feedback with measurable signals including disturbance. From

**Feedback with intermediate signals** TODO

**Combining them** TODO

## 17.4 Design general linear feedback

General feedback design methods include:

1. Pole placement (using observer)
2. Linear quadratic regulation
3. Model predictive control

### 17.4.1 System states

Definition: system states

System states at  $t = \text{summary of its past from which we can determine the output } y(t)$   
Transfer function yields

$$y(t) = \mathcal{L}^{-1}\{G(s)U(s)\} = \int_{\tau=0}^t g(t) \underbrace{u(t-\tau)}_{\text{past input}} d\tau$$

$$\begin{aligned} y(t) &= c_1 x_1(t) + c_2 x_2(t) + \dots + c_n x_n(t) + Du(t) \\ &= Cx(t) + Du(t) \end{aligned}$$

where

$$C = [c_1 \quad \dots \quad c_n] \text{ and } x(t) \triangleq \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

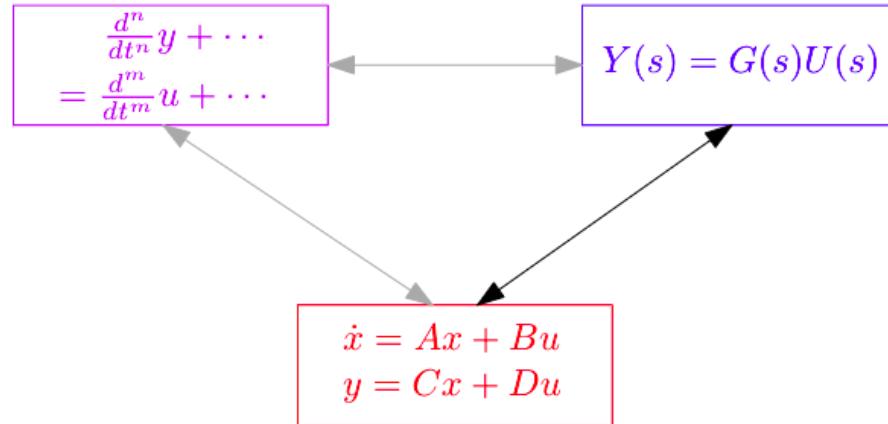


Figure 17.11: Translate LTI system models. From

$$G(s) = \underbrace{C}_{1 \times n} \underbrace{(sI - A)^{-1}}_{n \times n} \underbrace{B}_{n \times n} + \underbrace{D}_{1 \times 1} = \frac{b(s)}{a(s)}$$

#### Important property

- System matrix A:s eigenvalues  $\{\lambda\}$  given by solution to  $\det(\lambda I - A) = \lambda^n + a_1\lambda^{n-1} + \dots + a_n = 0$
- $a(s) = \det(sI - A)$  is a polynomial of order  $n$

- $G(s)$ :s poles  $p_i \subseteq A$ :s eigenvalues  $\lambda_j$

Given transfer function

$$G(s) = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_n}{s^n + a_1 s^{n-1} + \cdots + a_n}$$

one can choose e.g. **controllable canonical form**

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_n \end{bmatrix} &= \begin{bmatrix} -a_1 & -a_2 & -a_3 & \cdots & -a_n \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u \\ y &= [b_1 - a_1 b_0 \quad b_2 - a_2 b_0 \quad \cdots \quad b_n - a_n b_0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + b_0 u \end{aligned}$$

Figure 17.12: State-space form  $\leftarrow$  transfer function. From

Solution to state-space equation

$$x(t) = e^{At} x_0 + \int_0^t e^{A\tau} Bu(t-\tau) d\tau$$

matrix exponential  $e^{At}$  defined as

$$e^{At} = \mathcal{L}^{-1} \{(sI - A)^{-1}\}$$

- States  $x(t)$  at  $t$  summarize all past inputs  $u(t-\tau)$
- State space formulation easily incorporates initial condition
- Determine output  $y(t) = Cx(t) + Du(t)$

Asymptotically stable if

$$u(t) \equiv 0 \Rightarrow \lim_{t \rightarrow \infty} x(t) = 0$$

### stability

- $A$ :s eigenvalues are strictly in left half-plane  $\Leftrightarrow$  system is asymptotically stable.
- $A$ :s eigenvalues are strictly in left half-plane  $\Rightarrow$  system is input-output stable.

Example: State space

TODO Find transfer function F8 s14

## 17.5 Nonlinear time-invariant system

Feedback control using states

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned}\Leftrightarrow Y(s) = G(s)U(s)$$



Figure 17.13: State-feedback control. From

**State space description:**

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned}\Leftrightarrow G(s) = C(sI - A)^{-1}B$$

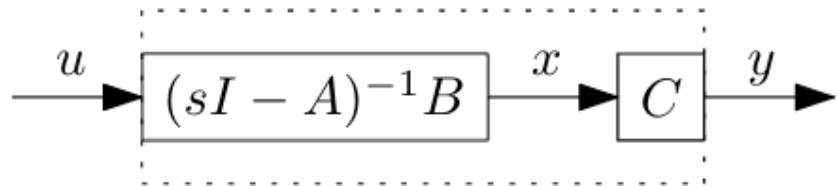


Figure 17.14: State-feedback control. From

$$u = -Lx + l_0r$$

**state-space equation:**

$$\dot{x} = Ax + B(-Lx + l_0r) = (A - BL)x + Bl_0r$$

The resulting open loop system

$$\begin{aligned}\dot{x} &= (A - BL)x + Bl_0r \\ y &= Cx\end{aligned}$$

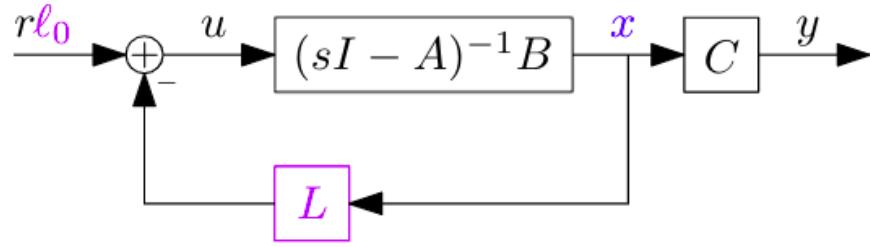


Figure 17.15: State-feedback control. From

**Controllable?**

A sought state  $x^*$  is *controllable* if we can move the system from  $x(0) = 0$  to  $x(T) = x^*$

$$\begin{aligned} x(T) &= e^{At}x_0 + \int_0^T e^{A\tau}Bu(T-\tau)d\tau \\ &= B\gamma_0 + AB\gamma_1 + \dots + A^{n-1}B\gamma_{n-1} \end{aligned}$$

$$S \triangleq [B \ AB \ \dots \ A^{n-1}B]$$

All states  $x^*$  are controllable  $S$ :s columns are linearly independent.  $\text{rank}(S) = n$  or  $\det(S) \neq 0$

$$\begin{aligned} y(t) &= Cx(t) \\ &= Ce^{At}x^* + 0 \end{aligned}$$

$$Cx^* = 0, CAx^*, \dots, CA^{n-1}x^* = 0$$

$$O \triangleq \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

**Observable?**

All states  $x^*$  are *observable*  $\Leftrightarrow O$ :s columns are linearly independent

Example: controllable

Is the following system controllable?

$$\begin{aligned} x(t) &= \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \\ y(t) &= [1 \ 1] x(t) \end{aligned}$$

$$\begin{aligned} \mathcal{O} &= \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = [n=2] = \begin{bmatrix} C \\ CA \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \end{bmatrix}}_{2 \times 2} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \\ \Rightarrow \det(\mathcal{O}) &= \det \left( \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \right) = (1)(-1) - (1)(-1) = 0 \end{aligned}$$

Therefore the system is not controllable

### 17.5.1 Minimal realization of state-space form

State-space form of  $G(s)$  is a *minimal realization* if vector  $x$  has the smallest possible dimension.

A state-space form is *minimal realization*  $\Leftrightarrow$  controllable and observable  $\Leftrightarrow A$ :s eigenvalues =  $G(s)$ :s poles

### 17.5.2 Method: Design of state-feedback control

State-space model with controller  $u = -Lx + l_0r$  where

$$L = [l_1 \ l_2 \ \dots \ l_n]$$

The closed loop system is expressed as

$$\begin{aligned} \dot{x} &= (A - BL)x + Bl_0r \\ y &= Cx \end{aligned}$$

The closed loop system as a transfer function

$$G_c(s) = C(sI - A + BL)^{-1}Bl_0$$

Calculations of eigenvalues/poles

$$\det(sI - A + BL) = 0$$

$$l_0 = \frac{1}{C(-A + BL)^{-1}B}$$

State-space form is controllable  $\Leftrightarrow L$  can be designed to yield arbitrarily placed poles (real and complex-conjugated) of the closed-loop system

### 17.5.3 Estimation via simulation

Controller

$$u = -L\hat{x} + l_0 r$$

where  $\hat{x}$  is an estimate of the actual state  $x$  that is unknown

we estimate via simulating the states

$$\dot{\hat{x}} = A\hat{x} + Bu, \quad \hat{x}(0) = \hat{x}_0$$

where  $\hat{x}_0$  is an initial guess.

### Estimation vi observer

Instead of simply guessing what  $\hat{x}$  is. We use a so called observer

$$\dot{\hat{x}} = A\hat{x} + Bu + \underbrace{K(y - C\hat{x})}_{\text{correction}}, \quad \hat{x}(0) = \hat{x}_0$$

where  $K$  is a matrix

$$K = \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{bmatrix}$$

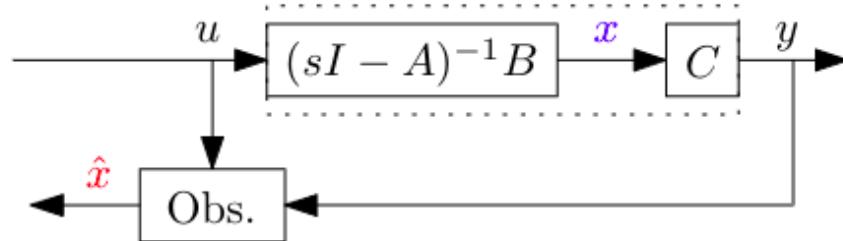


Figure 17.16: Estimating the states with observer. From

### Estimation error

Definition: Estimation error

*Estimation error:*

$$\tilde{x} \triangleq x - \hat{x}$$

Errors of observer described as system

$$\tilde{x}(t) = e^{(A - KC)t} \tilde{x}(0)$$

and therefore  $\|\tilde{x}(t)\|$  decays at a rate given by maximum

$$\operatorname{Re}\{\tilde{s}_i\}$$

where  $\tilde{s}_i$  are observer poles/eigenvalues of  $(A - KC)$

State-space form is observable ( $\det \mathcal{O} \neq 0$ )  $\Rightarrow$  matrix  $K$  can be chosen such that  $\tilde{x}$  vanish arbitrarily quick.

$K$  is solved by polynomial  $\det(sI - A + KC) = 0$  with desired roots in left halfplane

Controller using estimated states

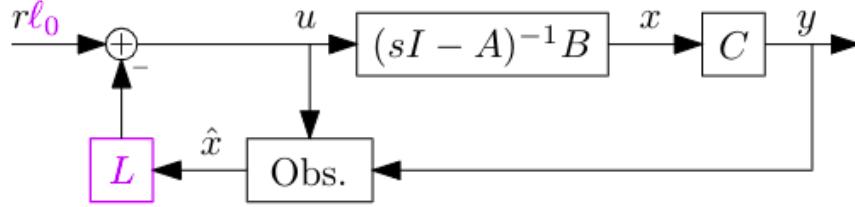


Figure 17.17: Estimating the states with observer. From

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad \begin{cases} u = -L\hat{x} + l_0r \\ \dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x}) \end{cases}$$

$$L : \begin{cases} U(s) = -L\hat{X}(s) + l_0R(s) \\ s\hat{X}(s) = A\hat{X}(s) + BU(s) + K(Y(s) - C\hat{X}(s)) \end{cases}$$

Closed-loop system with estimated states

Close loop system

$$\begin{aligned} \dot{x} &= (A - BL)x + \underbrace{BL\hat{x}}_{\text{effect of estimation error}} + Bl_0r \\ y &= Cx \end{aligned}$$

The closed-loop system with estimation error can be written as

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} &= \underbrace{\begin{bmatrix} A - BL & BL \\ 0 & A - KC \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\tilde{B}} l_0 r \\ y &= \underbrace{\begin{bmatrix} C & 0 \end{bmatrix}}_{\tilde{C}} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} \end{aligned}$$

## 17.6 Sensitivity and robustness

Sensitivity function:

$$S(s) \triangleq \frac{1}{1 + G_o(s)}$$

Complementary sensitivity function:

$$T(s) \triangleq 1 - S(s) = \frac{G_o(s)}{1 + G_o(s)}$$

$$S(s) + T(s) \equiv 1 \quad \forall 1 \tag{17.1}$$

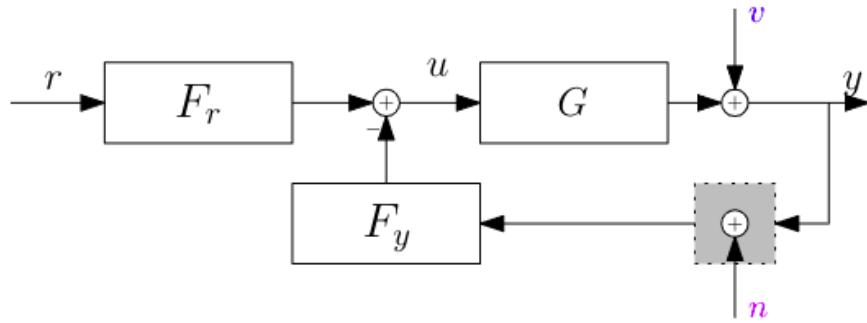


Figure 17.18: Control system with disturbances and noise. From

$$Y(s) = G_c(s)R(s) + S(s)V(s) + T(s)N(s)$$

where  $V(s)$  is the disturbance from external factors on the system. For example a wind or other forces.  $N(s)$  is the noise which can be for example from an inaccurate sensor or the sensor not coping with the quick changes.

We can either minimize the disturbance or the noise since  $S(s)$  and  $T(s)$  are connected see equation 17.1. Often the noise is high frequency and disturbance is lower frequency for example if someone is using the system that is a log

Ideally both  $|S(i\omega)|$  and  $|T(i\omega)| \gg 1$

$$|S(i\omega)| + |T(i\omega)| \geq |S(i\omega) + T(i\omega)| \equiv 1$$

In addition we want Nyquist contour

$$G_o(i\omega) = F_y(i\omega)G(i\omega) = \frac{T(i\omega)}{S(i\omega)}$$

### 17.6.1 Robustness: model error and stability

Assume

1. Controller  $F(s)$  stability the assumed system  $G(s)$
2.  $G(s)$  and  $G^0(s)$  have same number of poles in right half-plane.
3. Open-loop:  $F(s)G(s) \rightarrow 0$  and  $F(s)G^0(s) \rightarrow 0$  where  $|s| \rightarrow \infty$

Robustness criterium If assumptions are valid and  $T(i\omega)$  fulfills

$$|T(i\omega)| < \frac{1}{|\delta G(i\omega)|}, \quad -\infty \leq \omega \leq \infty$$

$\Rightarrow$  the real closed-loop system  $G_c^0(s)$  is also stable!

## 17.7 Digital Controllers

### 17.7.1 Sampling continuos-time output

A/D: Sampling feedback signal  $y(t) = y_c(kT)$

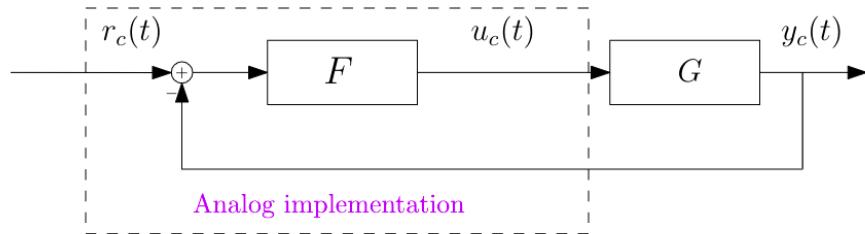


Figure 17.19: Digital systems and discrete-time clocks. From

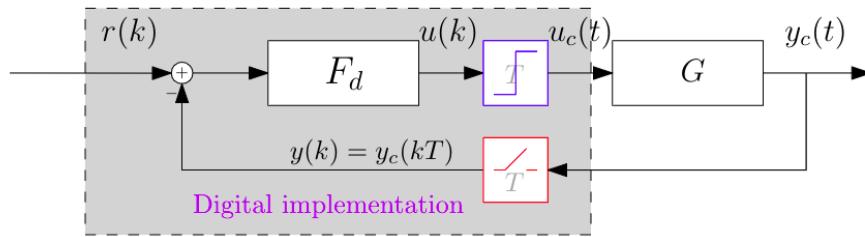


Figure 17.20: Digital systems and discrete-time clocks. From

Nyquist sampling theorem: If  $y_c(t)$  is bandlimited to  $\pm\omega_B = \pm2\pi f_b \Rightarrow$  it can be recovered exactly from  $y_c(kT)$  when sampling frequency

$$f_s \equiv \frac{1}{T} > 2f_B$$

### 17.7.2 Generating continuous-time input

Zero-order hold: continuous-time input is piecewise constant for  $k = 0, 1, 2, \dots$

$$u_c(t) = u_c(kT), kT \leq t < (k+1)T$$

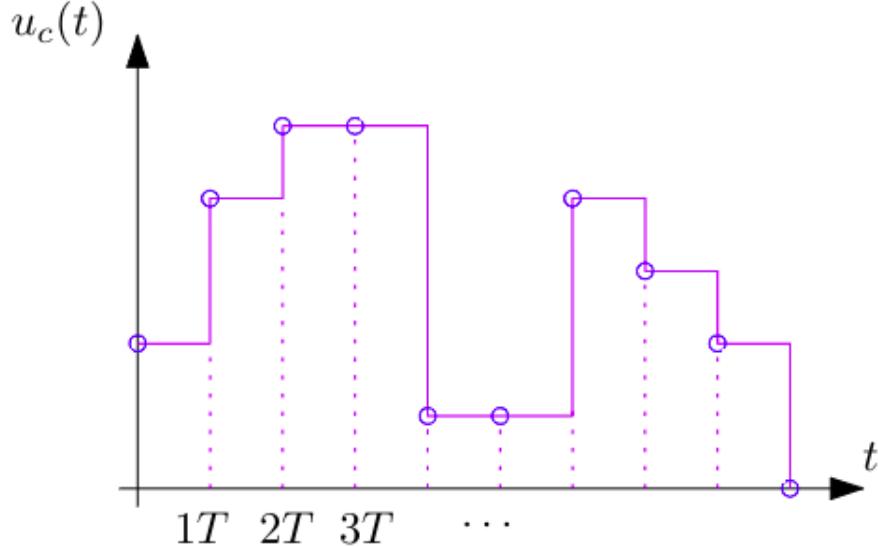


Figure 17.21: zero-order hold. From

#### State evolution when using zero-order hold input

State  $x_c(t)$  at time  $(k+1)T$  is determined at time  $kT$  as:

$$\begin{aligned} x(k+1) &= Fx(k) + Gu(k) \\ y(k) &= Hx(k) \end{aligned}$$

which are discrete-time states using matrices

$$\begin{aligned} F &= e^{AT} \\ G &= \int_{\tau=0}^T e^{AT} d\tau B = [\text{if } A^{-1} \text{ exists}] = A^{-1}(e^{AT} - I)B \\ H &= C \end{aligned}$$

If eigenvalues of  $A$ :  $\operatorname{Re}(s) < 0 \Rightarrow$  stable

If eigenvalues of  $F$ :  $|p| < 1 \Rightarrow$  stable

### 17.7.3 Discrete-time controllers

PID controller

$$u(k) = K_p e(k) + K_i T \underbrace{\sum_{n=0}^k e(n)}_{\approx \int_{\tau=0}^t e_c(\tau) d\tau} + K_d \underbrace{\frac{e(k) - e(k-1)}{T}}_{\frac{de_c(t)}{dt}}$$

## 17.8 Final notes before exam

Remember:

- $G_0 = FG$  (useful when calculating the nyquist marginally stable)
- poles of system in state space for is  $\det sI - A$
- $Y(s) = G(s)U(s)$  logically
- $Y(s) = G_c(s)R(s)$  logically
- $G(s) = C(sI - A)^{-1}B$

Thing that were missing from the exam:

- Forward- och cascade control (yes we have)
- Very little final value theorem
- What the diffrent PID parts does
- Roths algorithm
- Is the preformance spesifications met look at step response
- witch is the minimum phase
- general feedback control. is logical
- what is  $F_r$  and  $F_y$ . use the formulas writen on exams. L9,p18

Other info:

- (L3, P:) The system static gain is therefore  $G(0)$
- (L3, P17) Stationary error  $e_f$  approaches 0 if  $G(0)F(0) = \infty$ . (E.g. when  $F(s)$  contains 1 s, i.e. integration.)
- (L5, P19)  $G(s) = K_0 \dots$
- (L6, P9) lead block lag block, formula is given in formula sheet
- (L6, P13) Among all systems  $G(s)$  with the same magnitude curve, the system with the least negative phase shift is a minimum phase system.
- (L8, P8) the close loop representation in state space form
- (L8, P21)  $G_c$  and  $l_0$
- (L10, P6) Ideally S and T should be less then 1 but it is impossible since  $|S(i\omega)| + |T(i\omega)| \geq 1$
- (L10, P9) Nyquist contour  $G_o(i\omega) = F_y(i\omega)G(i\omega) = \frac{T(i\omega)}{S(i\omega)}$
- (L10, P11) Relative model error
- (L10, P14)  $|T(i\omega)| < \frac{1}{|\Delta G(i\omega)|}$  image



## Chapter 18

# Introduction to Machine Learning

### Resources

- <http://user.it.uu.se/~justin/Hugo/courses/machinelearning/>
- <https://scikit-learn.org/stable/>
- <https://numpy.org/>
- <https://pandas.pydata.org/>
- <https://matplotlib.org/>

## 18.1 Introduction

Overfitting is the property of a model such that the model predicts very well labels of the examples used during training but frequently makes errors when applied to examples that weren't seen by the learning algorithm during training. **Over fitting:** To high degree of model with to few data. **Bias:** The data that has been selected may not be a true representation of the true data set.

How many samples should we use? We know how many unknown parameters we have, then can we should have more data points then parameters at a minimum.

### 18.1.1 Types of Learning

- **Supervised Learning:**

- Desc: You are given labelled data. For each data-point you know what the correct prediction should be.
- Math Model: Labeled examples  $\{(x_i, y_i)\}_{i=1}^N$  where each element  $x_i$  among  $N$  is called a **feature vector**.
- Goal: The goal of a **supervised learning algorithm** is to use the dataset to produce a model that takes a feature vector  $x$  as input and outputs information that allows deducing the label for this feature vector.

- **Unsupervised Learning:**

- Desc: You just have data which is not labelled. This is given to algorithm. The most common algorithms do some sort of clustering, data-points that are similar are grouped together.
- Math Model: unlabeled examples  $\{x_i\}_{i=1}^N$  where  $x$  is the feature vector
- Goal: The goal of an **unsupervised learning algorithm** is to create a model that takes a feature vector  $x$  as input and either transforms it into another vector or into a value that can be used to solve a practical problem

- **Semi-Supervised Learning:**

- Desc: both labeled and unlabeled examples

- Goal: The goal of a **semi-supervised learning algorithm** is the same as the goal of the supervised learning algorithm

- **Reinforcement Learning**

- Desc: Different actions bring different rewards and could also move the machine to another state of the environment
- Goal: The goal of a **reinforcement learning algorithm** is to learn a policy

### 18.1.2 Notation and Definitions

- **Unbiased Estimators:**

$$\mathbb{E}[\hat{\theta}(S_X)] = \theta$$

$S_X = x_{i_1}^N = 1$ ,  $\hat{\theta}$  is a sample statistic and  $\theta$  is the real statistic.

- **Bayes' Rule:**

$$Pr(X = x | Y = y) = \frac{Pr(Y = y | X = x) Pr(X = x)}{Pr(Y = y)}$$

- **Parameter Estimation:**

Is an algorithm that can predict the parameter given there probability's?

- **Parameters vs. Hyperparameters:**

A Hyperparameter, set of numerical valued parameters with is decided by the data analysis. Parameters are variables that define the model learned by the learning algorithm. Examples of hyperparameter is  $C$  logistic regression which is the penalty strength.

- **Classification vs. Regression:**

- **Classification:** Each data point should be put into one of a finite number of classes. For example email should be classified as Spam or Ham. Pictures should be classified into pictures of cats, dogs, or sleeping students.
- **Regression:** Given the data the required prediction is some value. For example predicting house prices from the location and the size of the house

- **Model-Based vs. Instance-Based Learning:**

Model-Based create a model after training with data which then can be discarded unlike Instance-Based Learning which takes the data closes the the given inputs and predicts after that.

- **Shallow vs. Deep Learning** Shallow algorithm learns creates the parameters learned

only on the training examples.

- **Categorical**

A finite set of categoric with the data is a subset of. For example sex (Man or Woman) is a categorical value and weight is not.

- **Scaling**

if one feature have a more dominant number (larger) then you can scale (multiple with some number) the smaller number so they become similar

- **Probabilistic Classification**

Try to predicate the probability that an input sample  $x$  belongs to a class

- **Odds ratio**

Given an event with probability  $p$  we can take the odds ratio of  $p$  happening and  $p$  not happening

$$\frac{p}{1-p}$$

- **Multiclass classification**

- one vs all or one vs rest

Is when you set the one class to 1 and the rest to 0 for each of the classes. This will however result in uneven amount of positive and negative examples, thus perform badly

- one vs one

Is when you compare two classes at a time

## Hypotheses

$$h_{\theta_0, \theta_1}(x) = \theta_0 + \theta_1 x$$

where  $\theta_0$  and  $\theta_1$  are two weights/parameters.

## Measuring Error - RMS

A way to measure the error.

$$J(\theta_0, \theta_1, x, y) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta_0, \theta_1}(x^i) - y^i)^2$$

Also known as loss error function.

**Minimising regret**, Minimising root mean square error, minimising classification error.

**Parameters** of algorithm are the things that learned from the data. In neural networks you would call this the weight space.

## Training and Test Sets

The data should be divided into two parts

**Training Sets** This is the data you use to find the best parameters of the model or hypothesis. Machine learning can be seen as an optimisation problem find the parameters that best explain the data under some error/cost or loss function.

**Test Sets** The test set is what you use to validate your model. You are interested in the error/cost on this set.

## Confusion matrices

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 18.1: Actual prediction matrix. From ?

Accuracy

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Precision

$$\frac{TP}{TP + FP}$$

## 18.2 Linear Regression as Machine Learning

### Properties

- Supervised learning algorithm
- Numerical variables
- Regression problem

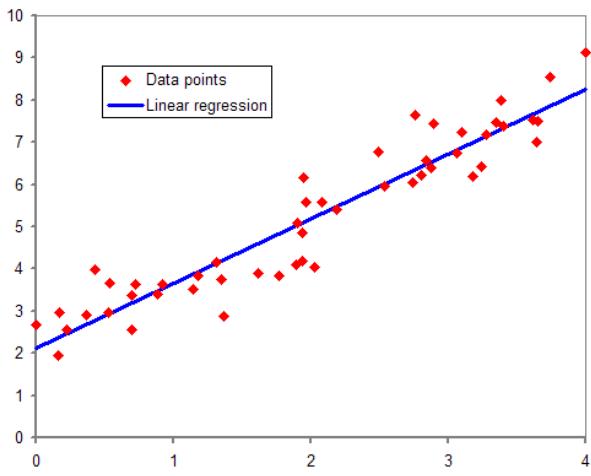


Figure 18.2: Linear regressionFrom ?

Explenation

With the linear hypothesis

$$h_{\theta_0, \theta_1}(x) = \theta_0 + \theta_1 x$$

we want the parameters  $\theta_0$  and  $\theta_1$  to give us as small of a error as possible

the error is calculated with **Cost function**

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta_0, \theta_1}(x^i) - y^i)^2$$

Where the goal is to **minimize**  $J(\theta_0, \theta_1)$ .

### 18.2.1 gradient descent

Gradient descent can be used for many thing, not so often use for linear regression. Essentialy what is dose it take a 360 degree view and look at what is the fastest way to get down the slope. It will repeate one step at a time untill it is at a local minimum.

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m \left( h_{\theta}(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Note that  $x_0 = 1$

### Linear Regression with Gradient Descent

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1(x^{(i)}) - y^{(i)})$$

$$\theta_1 \leftarrow \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1(x^{(i)}) - y^{(i)}) x^{(i)}$$

where  $\alpha$  is how aggressive the change should be and  $\frac{\partial}{\partial \theta_j}$  is the angle at the current point, this tells us were we should go and how big of a step.

$\leftarrow$  and  $:=$  is the same thing

If  $J$  is minimized then linear regression *always* converges to a global minimum not just a local minimum. The value of the global minimum dose not need to be 0 (a line docent need to have a derivative with is 0).

Example Discriptive example

## 18.3 Probability and Naive Bayes Classification

### Properties

- Supervised learning algorithm
- Categorical variables
- Classification problem

By calculating the probability we can decide witch class it belongs to by seeing witch has the biggest probability.

### Bayes theorem

$$\begin{aligned} P(H|E) &= \frac{P(H)P(E|H)}{P(E)} \\ &= \frac{P(H)P(E|H)}{P(H)P(E|H) + P(\neg H)P(E|\neg H)} \end{aligned}$$

where  $H$  is the hypothesis and  $E$  is the evidence.  $P(H)$  is called the prior,  $P(E|H)$  is called the likelihood and  $P(H|E)$  is the posterior.

for several cases one can express as followed:

$$P(A|b_1, b_2, \dots, b_n) = \frac{P(b_1|A)P(b_2|A)\dots P(b_n|A)P(A)}{P(b_1)P(b_2)\dots P(b_n)}$$

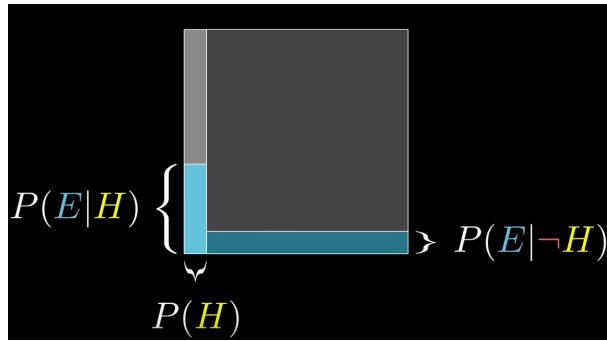


Figure 18.3: Bayes Theorem. From ?

- Experiment: is an outcome of several possible outcomes
- Sample space: the set of all possible outcomes
- Event: A subset of a sample space

## 18.4 Logistic Regression

### Properties

- Supervised learning algorithm
- Numerical and categorical variables
- Classification algorithm
- binary classification problems

Linear regression is not good at classification problems, since not all classes can be linear separable. Logistic regression is used for classification better than linear regression.

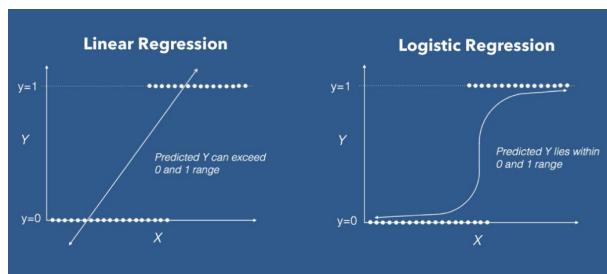


Figure 18.4: Linear vs logistic regression. From ?

### Logistic regression explanation

The prediction will be for value between 0 and 1 so it is best for binary classifications. Sigmoid function is the same as logistic function.

$$0 \leq h_{\theta}(x) \leq 1$$

$$\begin{aligned} h_{\theta}(x) &= g(\theta^T x) \\ g(z) &= \frac{1}{1 + e^{-z}} \\ \Rightarrow h_{\theta}(x) &= \frac{1}{1 + e^{-\theta^T x}} \end{aligned}$$

$$h_{\theta} = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \dots)$$

### 18.4.1 Cost function

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta} x^{(i)}, y^{(i)}) \\ &= -\frac{1}{m} \\ &\cdot \left[ \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] \end{aligned}$$

### 18.4.2 Gradient dissent

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$J(\theta, x, y) = \frac{1}{2m} \sum_{i=1}^m \left( \sigma(\theta^T x^{(i)} - y^{(i)}) \right)^2$$

Multiclass classification problem. We run classification problem for each class we do logistic regression for the class and not class. So if we have three classes we would have to do three logistic regression hypothesis. Then we can combine them to create a single hypothesis

### Cost function with regularized

Underfitting is when we use to low of degree of polynomial for the classification. Overfitting is when the model is too fitted and specific training and therefore performs badly on real data, i.e. to large degree.

We can manually look at the data set and select certain feature. We can also use a model to automatically do this with regularization.

If we set  $\lambda$  to a large number the parameter will automatically be small and therefore higher degrees will not effect the output as much. The regularization

parameter  $\lambda$  can automatically be determined  
<https://www.youtube.com/watch?v=IXPgm1e0I0o>

$$J(\theta) := \frac{1}{m} \left[ \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

### Gradient descent with regularization

$$\theta_j := \theta_j \left( 1 - \alpha \frac{\lambda}{m} \right) - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

## 18.5 Support Vector Machines

### Properties

- Supervised learning algorithm
- Numerical variables
- classification and regression problems

### SVM explained

Support Vector machines are models where the goal is to find a hyperplane (the line for the two ) with a margin either side that maximizes the space between the two clusters. The labels defining what is in the cluster and not is changed to  $-1$  and  $1$ .

We want to find weights  $\bar{w} \in \mathbb{R}^d$  and a constant  $b$  such that

$$\begin{cases} \bar{w} \cdot \bar{x} - b \geq 1 & \text{if } y = 1 \\ \bar{w} \cdot \bar{x} - b \leq -1 & \text{if } y = -1 \end{cases}$$

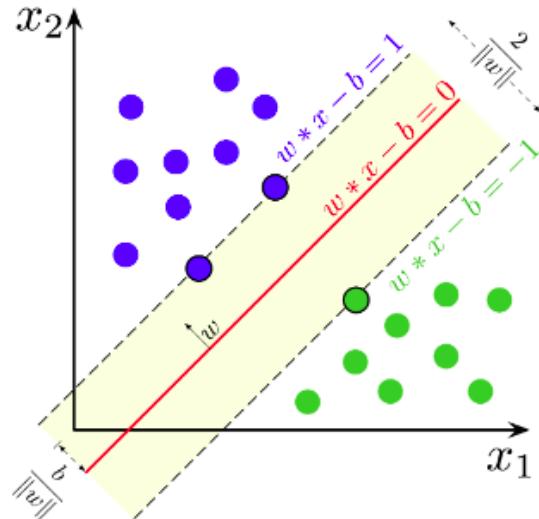


Figure 18.5: Bayes Theorem. From ?

Since the vectors are orthogonal to each other the product is 0

$$\bar{w} \cdot \bar{x} = 0$$

If we want the vector  $\bar{w}$  to be normal to the hyperplane

$$\begin{aligned} \bar{w} \cdot \bar{x} &= b \\ \Rightarrow \bar{w} \cdot \bar{x} - b &= 0 \end{aligned}$$

Hence the two points are  $\bar{x}_1$  where  $\bar{w} \cdot \bar{x}_1 - b = -1$  and  $\bar{x}_2$  where  $\bar{w} \cdot \bar{x}_2 - b = 1$

Maximising the distance between the two hyperplanes  $\bar{w} \cdot \bar{x} - b = 1$  and  $\bar{w} \cdot \bar{x} - b = -1$  we want to maximize  $t = \frac{2}{\|\bar{w}\|}$  so we minimize  $\frac{1}{2} \|\bar{w}\|^2$

### 18.5.1 Quadratic programming

Gradient descent will not work for all cases if there are a lot of quadratic terms but the problem is convex then we can use quadratic programming.

### 18.5.2 Slack Variables

when there is overlap Quadratic programming will not work

### 18.5.3 kernel trick

We can compute the inner product in the high-dimensional space by using a function on the lower dimensional vectors, i.e. The kernel trick is a way

to lower the demission to avoid componential expenses

#### 18.5.4 Gaussian Kernels

## 18.6 Some feature engineering and Cross validation

- Model Bias

A model is biased if the error/loss/cost function is high on the training data. It makes bad prediction on the training data.

- Model Variance

A model has high variance if the error/loss/cost function is high on the test data compared with the training data.

Bias variance trade off

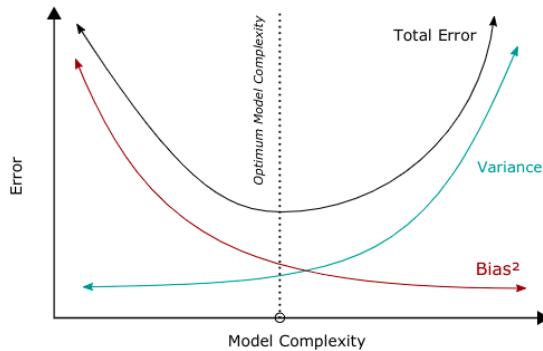


Figure 18.7: Bias variance trade off. From ?

Overfitting vs Bias

Two Goals

- Model Selection:

Estimating the performance of different models in order to chose the best one

- Model assessment:

Having chosen a final model, estimate its prediction error on new data.

- Training

This is what we use to train our different algorithms. Typical split 50%

- Validation

This is what we use to choose our model we pick the model with the best validation score. Typical split 25%

- Test

This is the data that you keep back until you have picked a model. You use this predict how well your model will do on real data. Typical split 25%

### 18.6.1 k-fold cross validation

It is used to evaluate which model is best.

If  $k = 5$ , then you have 5 parts  $T_1, \dots, T_5$  you would run 5 training runs

- Train on  $T_1, T_2, T_3, T_4$  evaluate on  $T_5$ .
- Train on  $T_1, T_2, T_3, T_5$  evaluate on  $T_4$ .
- Train on  $T_1, T_2, T_4, T_5$  evaluate on  $T_3$ .
- Train on  $T_1, T_3, T_4, T_5$  evaluate on  $T_2$ .
- Train on  $T_2, T_3, T_4, T_5$  evaluate on  $T_1$ .

Good values of  $k$  are 5 or 10. Obviously the larger  $k$  is the more time it takes to run the experiments.

#### *Hyper-parameters*

These are parameters to the learning algorithm that do not depend on the data. They are often continuous values such as the regularization parameter, but not allowance.

### 18.6.2 Estimating Hyper parameters - Grid search

We make divide our search space of the hyper parameters into a grid (evenly distributed possible values). Then we go through them all and find the parameters with minimize the error.

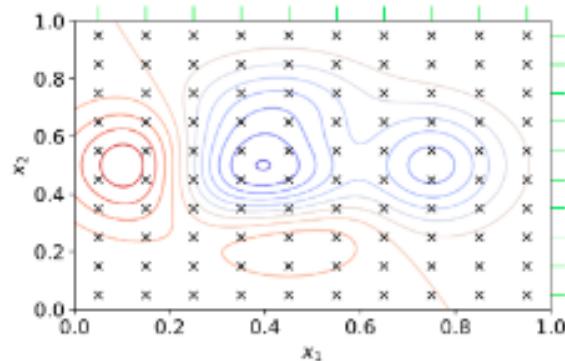


Figure 18.8: Grid search. From ?

Explanation

### 18.6.3 One-Hot encoding

Instead of classifying with natural number like 0, 1, 2 we only use 0 and 1. We use 1 for a class and 0 for the other classes.

### 18.6.4 Boosting for feature selection of linear models

Boosting is a general framework, and it can also be combined with cross-validation in a technique called bagging (bootstrap aggregating). The idea is very simple, learn you model one feature at time, at each stage pick the next feature that gives you optimal performance. You order the features in order of importance and this gives you models that are easier to interpret for humans.

Don't forget to scale your data, so that all dimensions have roughly the same range

Boosting for linear models advantages

- You order the variables in terms of importance
- There is the possibility to stop early when the model does not improve. This is a way of selecting a subset of the features

### 18.6.5 Co-variance matrix

Theremins the covariance between the features in a matrix. The larges eigen-value tell us the direction to project in order to maximize the variance in that dimension. <https://www.youtube.com/watch?v=152tSYtiQbw> [https://en.wikipedia.org/wiki/Covariance\\_matrix](https://en.wikipedia.org/wiki/Covariance_matrix)

### 18.6.6 Correlation matrix

How two features are moving with one another. Note that the correlation matrix is diagonally symmetric.

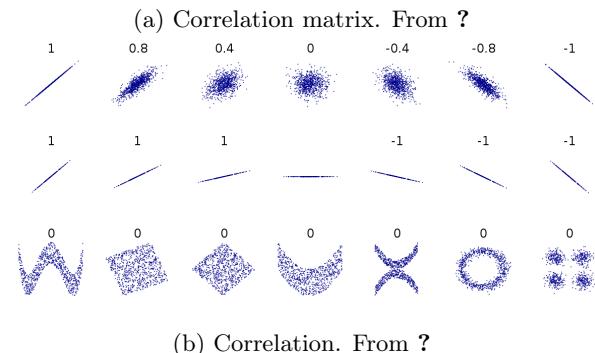
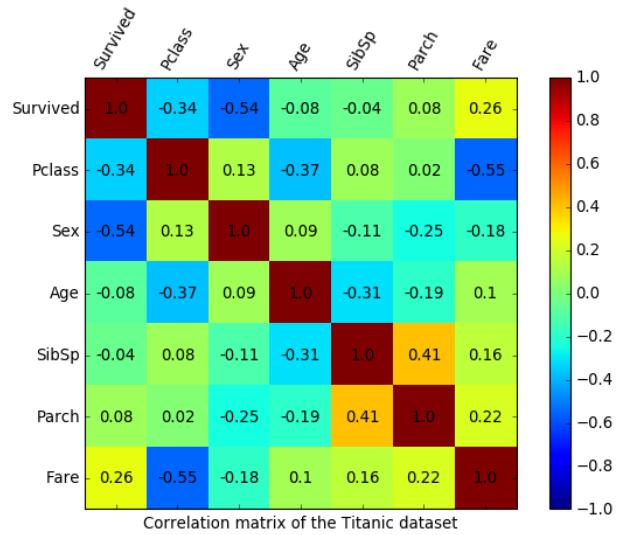


Figure 18.9: Correlation matrix, the two images are un related

### 18.6.7 Heatmap

This can be done with hierarchical clustering.

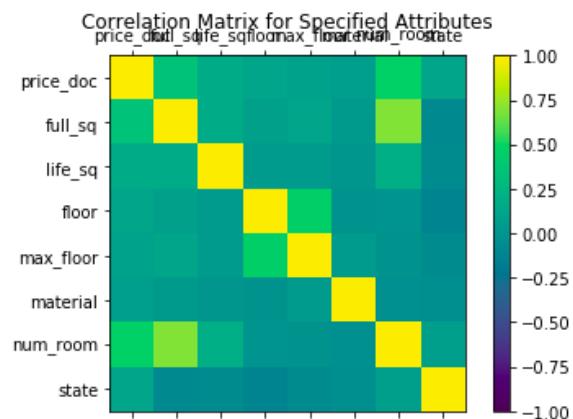


Figure 18.10: Heatmap. From ?

## 18.7 Clustering and classifiers

### 18.7.1 k-nearest neighbor classifier

#### Properties

- Un-supervised learning algorithm
- Numerical variables

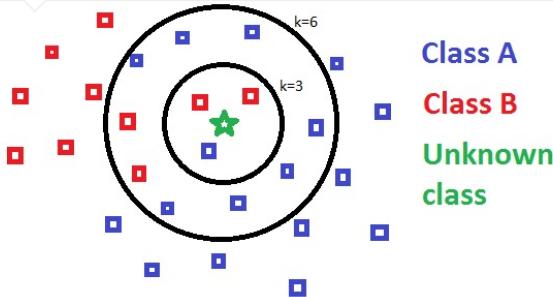


Figure 18.11: K-nn. From ?

Otherwise known as *k-NN*

- Very simple classifier
- Memory based, no model is learned you just have to remember the training data
- To classify a point, look at the  $k$ -closest points look at their classes and take a vote
- No need to do One-vs-Rest or One-vs-One.

Problems with k-NN

- As the size of the data set grows, and the more dimensions of the input data the computational complexity explodes. This is sometimes referred to as the curse of dimensionality
- With reasonable clever data structures and algorithms you can speed things up

### 18.7.2 Hierarchical clustering

#### Properties

- Un-supervised learning algorithm
- Numerical and categorical variables

Hierarchical clustering

Basic idea:

- Start with the same number of clusters as you have data points

- At each stage cluster together close together cluster
- Stop when you have enough clusters or everything is clustered together.

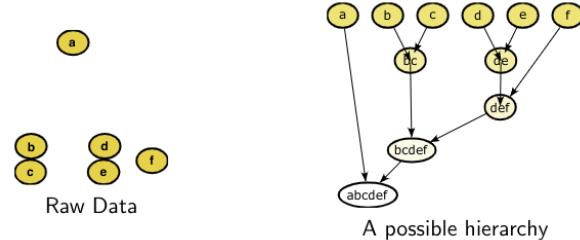


Figure 18.12: hierarchical clustering. From ?

#### Linkage Criteria

The act where a point is associated with a specific class.

- Maximum or complete linkage clustering:  $\max\{d(a, b) : a \in A, b \in B\}$
- Minimum or single-linkage clustering:  $\min\{d(a, b) : a \in A, b \in B\}$

### 18.7.3 K-Means

#### Properties

- Un-supervised learning algorithm
- Numerical variables

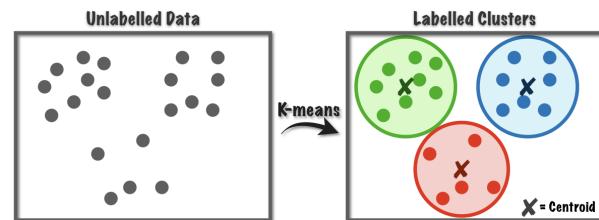


Figure 18.13: K-means. From ?

$$\mu = \frac{1}{N} \sum_{i=1}^n x_i$$

We want to find  $k$  centres  $\mu_1, \dots, \mu_k$  that minimizes the spread or max distance in each cluster.

First we randomly select the clusters centroids. Then we associate each point to a cluster with we then re-

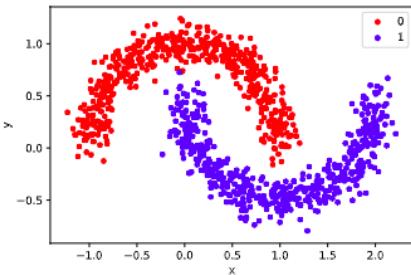
place the centroid so it minimizes it. Then we repeat this step.

H-Means explained

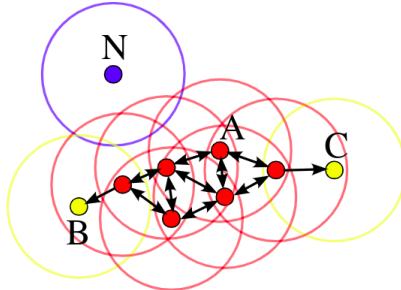
#### 18.7.4 DBSCAN

##### Properties

- Un-supervised learning algorithm
- Numerical variables



(a) DBSCAN example. From ?



(b) DBSCAN algorithm. From ?

Figure 18.14: DBSCAN

## 18.8 Information theory and Decision Theory

### 18.8.1 Decision Trees

##### Properties

- Supervised learning algorithm
- Categorical variables
- Classification and regression

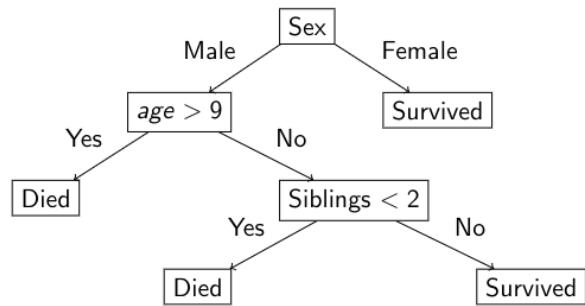


Figure 18.15: Decision Trees. From ?

It is NP-hard to find an ordering that gives the smallest tree. We can get exponential blow up in the size for different orders. To find the smallest tree we can use information theory.

### 18.8.2 ID3 algorithm

1. Take a feature (with the most entropy) and place it as the root node where the branches true or false or similar (dependent on entropy). Then remove the feature from the feature list.
2. Then take the new most entropy feature and place it as a node under the parent. Then remove that.
3. Then lastly we come to the conclusion (ex diabetes or not diabetes).

ID3 explained

### 18.8.3 Measuring Information

*Information theory* is the scientific study of the quantification, storage, and communication of digital information.

*Entropy*: a measurement of how spread out the data points are. High entropy means that the datapoints are evenly spread out and 0 means there are not.

$$H(X) = - \sum_{i=1}^n p_i \log_2(p_i) = \sum_{i=1}^n p_i \log_2 \left( \frac{1}{p_i} \right)$$

Properties of H and I Given an event that occurs with probability  $p$  we want a  $I(p)$  that measures the information of that event. We want  $I$  to have certain properties

- $I(p)$  is monotonically decreasing in  $p$ . The higher the probability of the event, the less information.

- $I(p) \geq 0$
- $I(1) = 0$ . An even with probability 1 has no information.
- if  $p$  and  $q$  are independent events then we want  $I(pq) = I(p) + I(q)$

Given these properties the only mathematically sensible choice is

$$I(p) = \log\left(\frac{1}{p}\right) = -\log(p)$$

Example: Unfair coin with probability  $p$

$$H(p, 1-p) = -p \log p - (1-p) \log(1-p)$$

If you plot the graph it will look something like figure

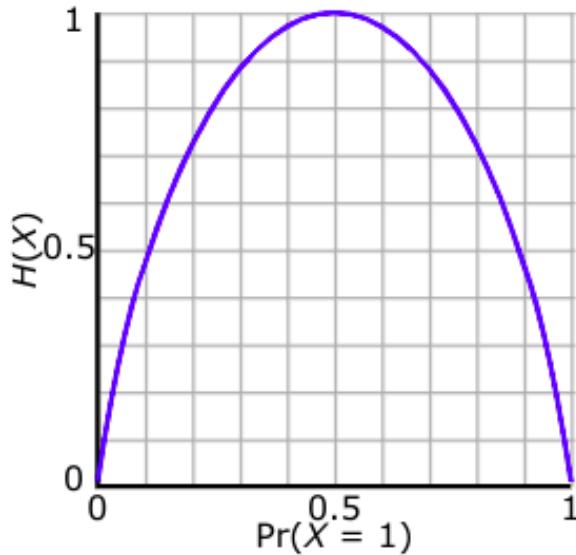


Figure 18.16: Decision Trees. From ?

**ID3 algorithm** the algorithm to construct a tree using information theory.

- ID3 is heuristic, i.e. practical solution but not optimal result, since we use information theory to make it easier to solve.
- ID3 will not construct the smallest tree.

*Conditional entropy*

$$H(X|Y = a) = - \sum_{x \in X} P(x|Y = a) \log_2 P(x|Y = a)$$

$$\text{where } P(X|Y) = \frac{P(X,Y)}{P(Y)}$$

*Information Gain*

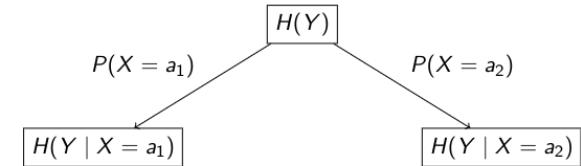


Figure 18.17: Decision Trees. From ?

$$H(Y) - \underbrace{\left( P(X = a_1)H(Y|X = a_1) + P(X = a_2)H(Y|X = a_2) \right)}_{\text{average information when split}}$$

	Cloudy ( $c$ )	Not Cloudy ( $\bar{c}$ )
Raining ( $r$ )	24/100	1/100
Not Raining ( $\bar{r}$ )	25/100	50/100

Figure 18.18: Example Conditional Entropy. From ?

### Example: Conditional entropy

See figure 18.18 The amount of days it is raining ( $24 + 1 = 25$ )

- $P(c|r) = 24/25$
- $P(\bar{c}|r) = 1/25$

$$H(Y|X = r) = - \underbrace{\frac{24}{25} \log_2 \frac{24}{25}}_{\text{Cloudy}} - \underbrace{\frac{1}{25} \log_2 \frac{1}{25}}_{\text{NotCloudy}} \approx 0.24$$

### Advantages of Decision trees

- Easy to understand what the algorithm has learned
- Can learn very non-linear boundaries
- Does not require that much processing
- Not many parameters to tune

### Disadvantages of Decision trees

- Prone to overfilling
- Small changes in the data can mean that you learn very different trees.
- Computationally more expressive than other methods

## 18.9 Principle component analysis (PCA)

Is dimensionality reduction method, i.e. reducing the number of dimension of a training set. We try to compress the data, i.e. remove the redundant data.

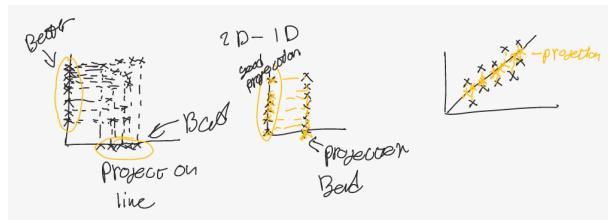


Figure 18.19: Example Conditional Entropy

covariance matrix is

$$\begin{aligned} B &= X - \bar{X} \\ C &= B^T B \end{aligned}$$

where  $X$  is the data matrix.

$$C\omega = \lambda\omega$$

Where  $C$  is Covariance matrix,  $\lambda$  is the eigenvalue and  $\omega$  is the eigenvector PCA discription

The principle components

$$T = B\omega$$

We then **reduce the dimension** by desigding on how many principle component we want this is done by taking the standard deviaton to including  $x\%$  of the data.

### 18.9.1 Covariance Matrix

Is a diagonaly symetric matrix of the covariance between two elements in the random vector.

The eigenvectors of the covariance matrix describes the direction of the line where the data points are maped to. <https://wiki.pathmind.com/eigenvector>

<https://www.youtube.com/watch?v=pmG4K79DUoI>  
<https://www.youtube.com/watch?v=rng04VJxUt4>

When the eigenvalues are relativly small we can remove those vectors since the do not contrebute much of the eigenvalue is in significant.

# Chapter 19

## Independent Project in Information Engineering

### 19.1 Design thinking

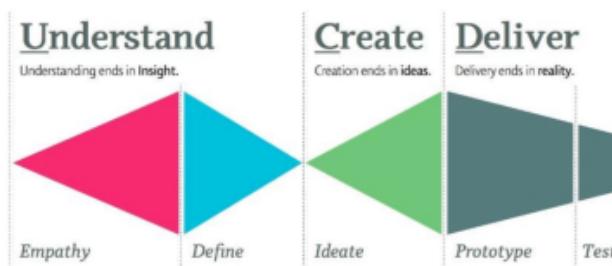


Figure 19.1: The waterfall model. From ?

### 19.2 Innovation & Immateriale rätt

En bra affärsidé:

- **Need.** Fråga de potentiella kunderna.
  - Vad för problem har du upplevt?
  - Beskriv den senaste gången du upplevde problemet?
  - Hur ofta upplever du problemet?
  - Hur brukar du lösa det?
  - Hur skulle ditt liv bli bättre om problemet lösades?
  - Hur mycket tid, pengar och energi kostar det dig?
- **Approach.** Hur du designar lösningen.

- **Benefit.** Du ska inte definiera, värdet kunden gör det.

- **Competition.** Google vad som finns.

### 19.3 Källor och kritisk granskning

Läs kritiskt

- Vad är det för typ av artikel?
- Är det värt att läsa, vad kan artikeln bidra med?
- Är den välskriven? (Begriplig, logisk, fakta/åsikter, referenser.)

The five Cs

- Category
- Context
- Correctness
- Contributions
- Clarity

Läs effektivt

**Pass 1** (5 -10 min): Läs titeln, abstract, introduktion, rubriker, sammanfattnings/conclusion och snapt kolla referenslistan. Utvärdera med Cs om det är värt att fortsätta.

**Pass 2** (max en timme): Läs artikeln men hoppa över svåra detaljer och antekna. Om det är värt att fortsätta gå till nästa pass.

**Pass 3** (fyra-fem timmar): Läs noggrant och förstå allt.

Vetenskapligt skrivande:

- Precist och entydligt
- Obekräftigt och neutralt
- Sammanhängande
- Utömmande, ekonomiskt/effektivt
- Klar och korrekt

får inte gynna obehöriga intressen och bör öppet redovisa ekonomiska och andra intressen som kan påverka tilltron till hans eller hennes opartiskhet och omdöme. Ingenjören bör enskilt och offentligt, i tal och skrift, sträva efter ett sakligt framställningssätt och undvika felaktiga, missvisande eller överdrivna påståenden.” <https://www.sverigesingenjorer.se/om-forbundet/sveriges-ingenjorer/hederskodex/>

## 19.6 Presentationsteknik

### 19.4 Crash-course inom dataskydd

Historia i EU

- Rätten till Privacy (offentligrätt): art 7 EU stadgan
- Rätten till Dataskydd (offentligrätt): art 8 EU stadgan
- Dataskydd-GDPR (offentligrättslig som närmar sig till civilrätten)

GDPR handlar om personuppgifter som sparas och hanteras

Om personuppgifter ska behandlas på något sätt så måste följande motiveras.

- Nödvändighet att behanda datan
- Andvändaren moste godkänna
- Datat ska vara skyddad från andras åtkomst

Inbyggd dataskydd (DPbD)

Om man kan hellt anonymisera datan så behöver man inte tillämpa GDPR. Dock är detta väldigt svårt och är istället Pseudonymisera.

### 19.5 Etik

God forskningssed

Klasisk filosofi

- Kants förnuftiga varelser
- Mills lyckobegrepp
- Aristoteles fronesis

sveriges ingenjörer hederskodex “Ingenjören bör respektera anförtrodda upplysningars konfidentiella natur samt andras rätt till uppslag, upfinningar, utredningar, planer och ritningar. Ingenjören

är bra presentation:

- Anpassad till åhörarna
- Bra flow, påläst, engagerat, humor
- Inte innantill
- Bra talteknik, artikulation, magstöd
- Väl utformade slides
- Alla involverade
- Kroppspråk
- Tydligt budskap

Det man tar upp:

- Motivationen först!
- Det viktigaste först, sen successivt mindre viktiga saker.
- Vad har ni gjort och hur, och varför? Hur gick det?
- Använd exempel!

Vad man inte tar med:

- Live-demo (men inspelad är ok)
- Referenslistan med relaterat arbete
- Vad man inte hinner
- Alla detaljer

Exempel på uppdrag:

- Motivera åhörarna, väcka intresse
- Beskriv problemet/problemställningen, varför det är viktigt
- Vad har ni gjort/byggt/åstadkommit?
- Hur gjorde ni? (men inte alla detaljer!)
- Vad blev resultatet, hur bra?
- Sammanfattning

Kontakt och nerver:

- Läs inte innan men det kan vara bra att ha manus eller stödord som stöd. Man kan också bara ha den första meningens.
- Ha ögonkontakt och le lite. Att le hjälper med nerverna.
- Andvänd kroppspråk.
- Öva mycket och när du över gör gärna framför andra eller bara prata för dig själv.

- Prata långsamt, andas, drik vatten.

Bra slides:

- Bilder och illustrationer för ren text.
- Bra att ha text också men inte för mycket åt en gång.
- Fake punkt listor.
- Det kan vara svårt att se text i slides om man sitter långt borta.



# Chapter 20

# Software Engineering and Project Management

## Resources

- <https://scrumguides.org/>
- <http://www.agilemodeling.com/artifacts/userStory.htm>
- <https://www.uml-diagrams.org/>

## 20.1 Part 1 Introduction to Software Engineering

### 20.1.1 Introduction

*Software Engineering* is tools, methods, and processes used in the development of software. It is important in order for large and complex systems to be managed, to prevent failure.

*Software* is more than just a program. It is the documentation, library, programs and everything else related to the program. Software can be two kinds: generic and customized.

The issues that can make the development and deployment of software difficult are:

- *Heterogeneity* (That it can run on any device, integrated with other software in other languages)
- *Business and social change*
- *Security and trust*
- *Scale Software*

### Software Engineering Code of Ethics and Professional Practice (Short Version)

**PREAMBLE** The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. PUBLIC – Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES – Software engineers shall be fair to and supportive of their colleagues.
8. SELF – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

#### 20.1.2 Software processes

##### Software process models

One of the three most common software process models (also known as SDLC model) are *The waterfall model*, *Incremental development*, and *Integration and configuration*.

##### Waterfall

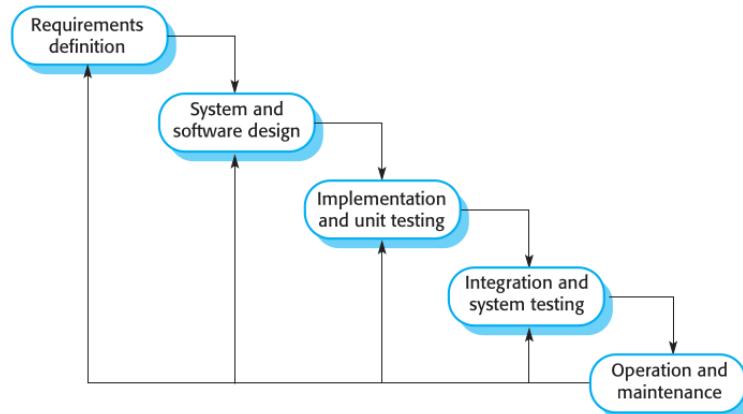


Figure 20.1: The waterfall model. From

Negative

- Bad when requirements change quickly.

- Not flexible.
- May delay the development process when waiting for customer approval.

Positive

- Well documented.
- Often more stable systems.

Used in

- Embedded systems
- Critical systems
- Large software systems that are part of broader engineering systems developed by several partner companies.

### Incremental development

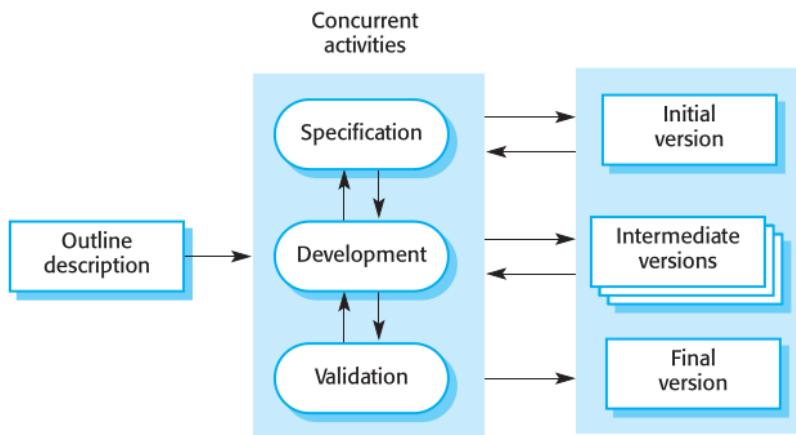


Figure 20.2: Incremental development model. From

Negative

- More difficult and time consuming to measure progress.
- Often the system structure degrades as new increments are developed. Therefore, regular refactoring is needed.

Positive

- Cheaper than waterfall.
- Easier to get feedback from customers since one can show a demonstration not just software design documents.
- Earlier delivery and deployment, since a functional version of the software exists.

Used in

- In software which requirement is likely to change during the development process, which is the case for most businesses.

### Integration and configuration

Negative

- May compromise requirements and therefore, not satisfying the real needs of the customer.

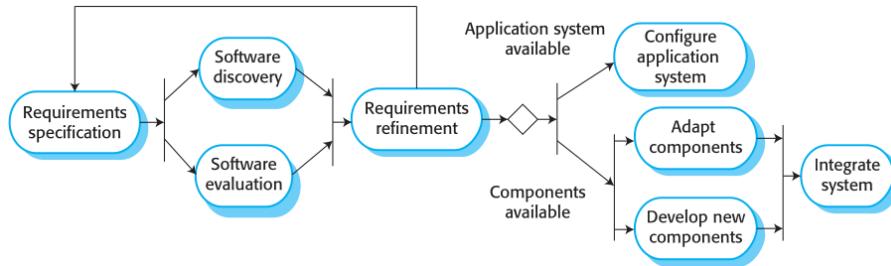


Figure 20.3: Integration and configuration model. From

- System evolution can be more difficult when the components are not controlled by the organization.

Positive

- Reducing the development of new software and therefore, reducing the cost and risks.
- Often faster delivery.

Used in

- When cost and time is limited.

### Process activities

The four basic process activities for a software processes are: Software specification, Software design and implementation, Software validation, and Software evolution.

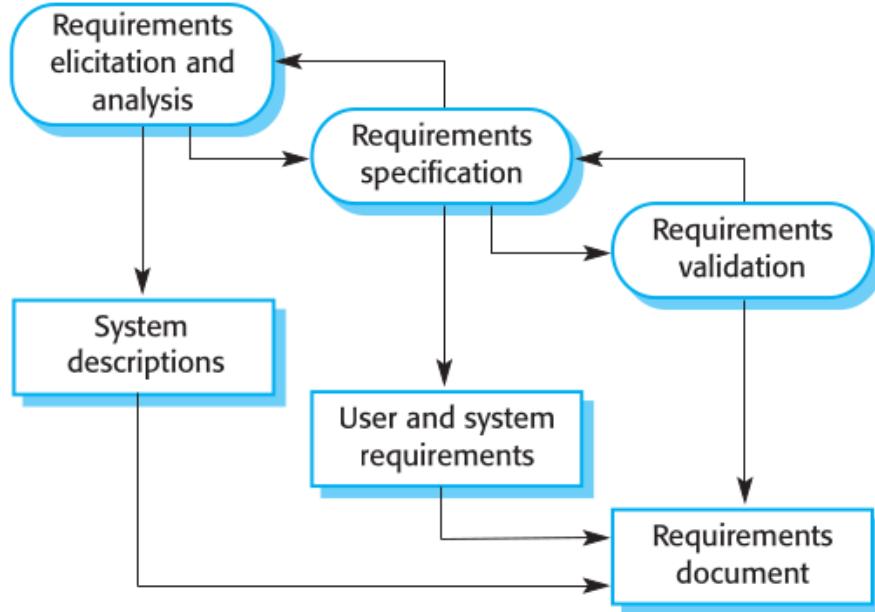


Figure 20.4: Software specification. From

#### 20.1.3 Agile software development

Agile methods are used to reduce process overhead and documentation, and increase increments software delivery.

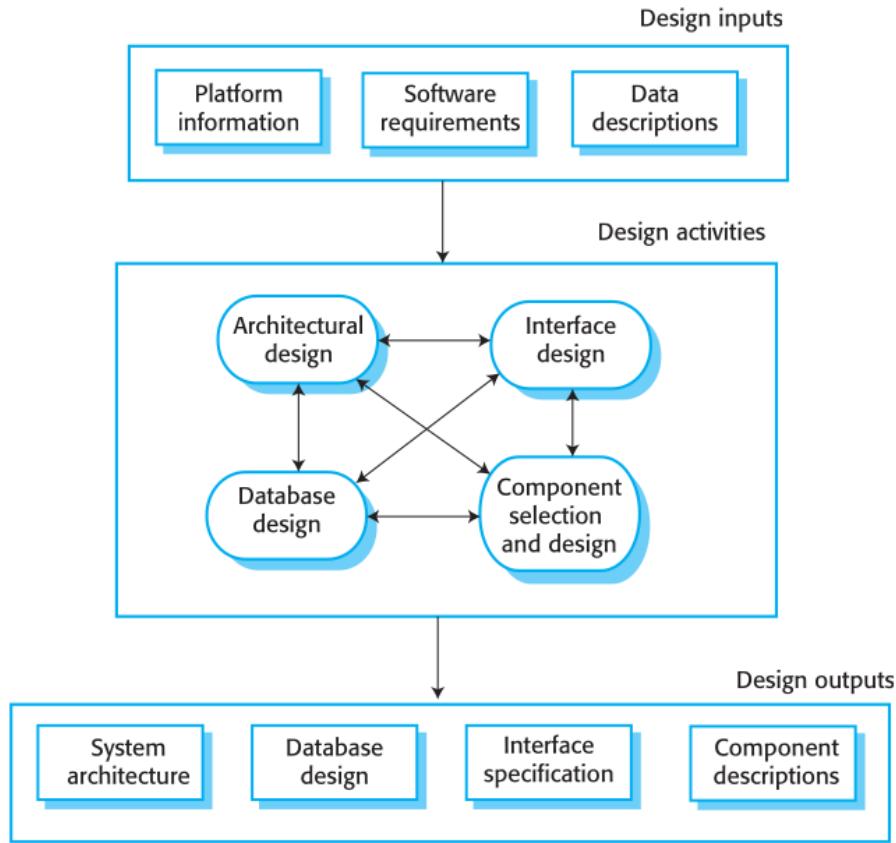


Figure 20.5: Software design and implementation. From

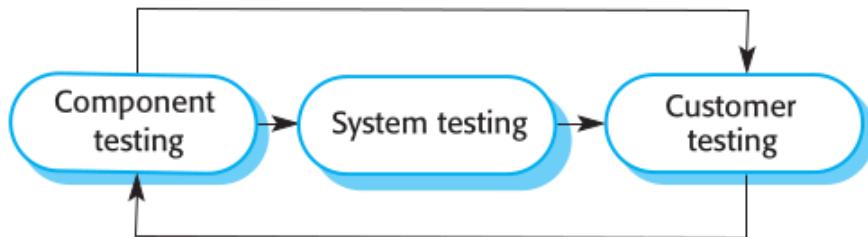


Figure 20.6: Software validation. From

A combination of plan-driven and agile may be used, but they can be used separately depending on the software being developed.

*User stories* are used to define the customer requirements to understand what needs to be done. From these, *tasks* are created to concretize the work.

*Test-first development* requires the tests to pass before proceeding to the next step. The tests are created from specific tasks. One issue with this method is “test-lag”, in other words when the pace of the developer is faster than the testers.

Introducing agile methods to an organization, which does not have that culture, is preferably done with a motivated them, which can then spread the methods across the organization.

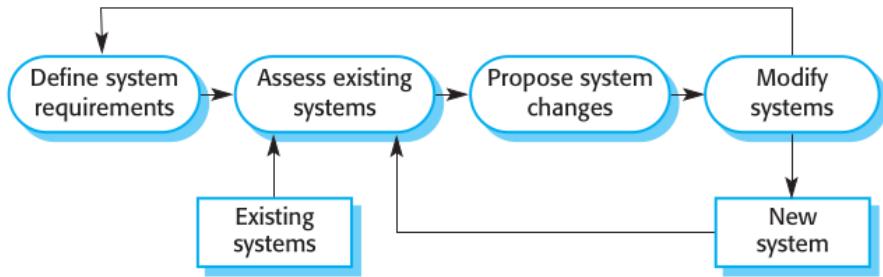


Figure 20.7: Software evolution. From

Agile methods also produce some issues, e.g. lack of product documentation, too much work for customers which may lead to difficulties of customer involved, and problems with continuity of the development team.

### Agile development practise

1. Incremental development
2. Customer involvement
3. Pair programming
4. Regular system releases to customers
5. Refactoring

### Scrum

*Scrum* provides with a framework of agile methods to ease the management of project.

The one who defined the requirements is called the *Product owner*. It is the task of the *ScrumMaster* to make sure that *Development team* fulfill this requirements. This is done with *Sprint*, which is to process of development. It used *Product backlog* to define the tasks. The estimate of the size of the product backlog for a single spring is called *Velocity*.

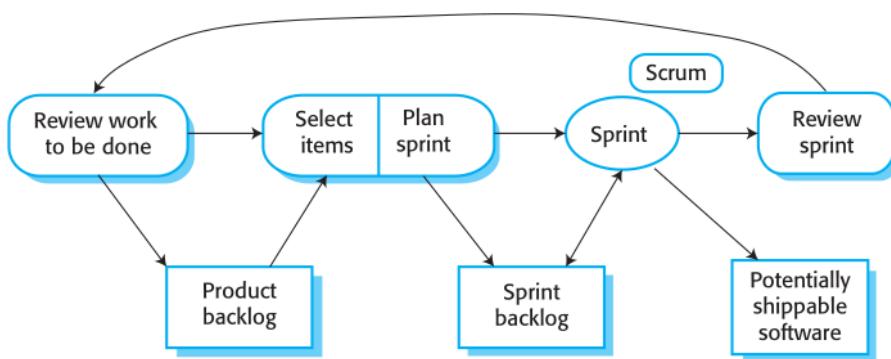


Figure 20.8: Scrum sprint cycle. From

### Scale agile methods

Plan-based practices needs to be used with agile practices. The plan based processes includes specifying the requirements initially and provide more documentation. Also, more coordination between teams are required, such as alignment of realizes, and common tools used in the development.

### 20.1.4 Requirements engineering

#### Functional and non-functional requirements

*Functional requirements* are specific description on what the system should do. On the other hand *non-functional requirements* does not describe a necessary for the service to work for the end user. It is concerned with properties such as:

- *speed* measured by processed transactions, response time, Screen refresh time.
- *size* measured by megabytes on disk.
- *Ease of use* measured by training time.
- *Reliability* measured by mean time to failure, rate of failure occurrence.
- *Robustness* measured by time to restart after failure, percentage of events cause failure.
- *Portability* measured by number of target systems.

#### Requirements engineering processes

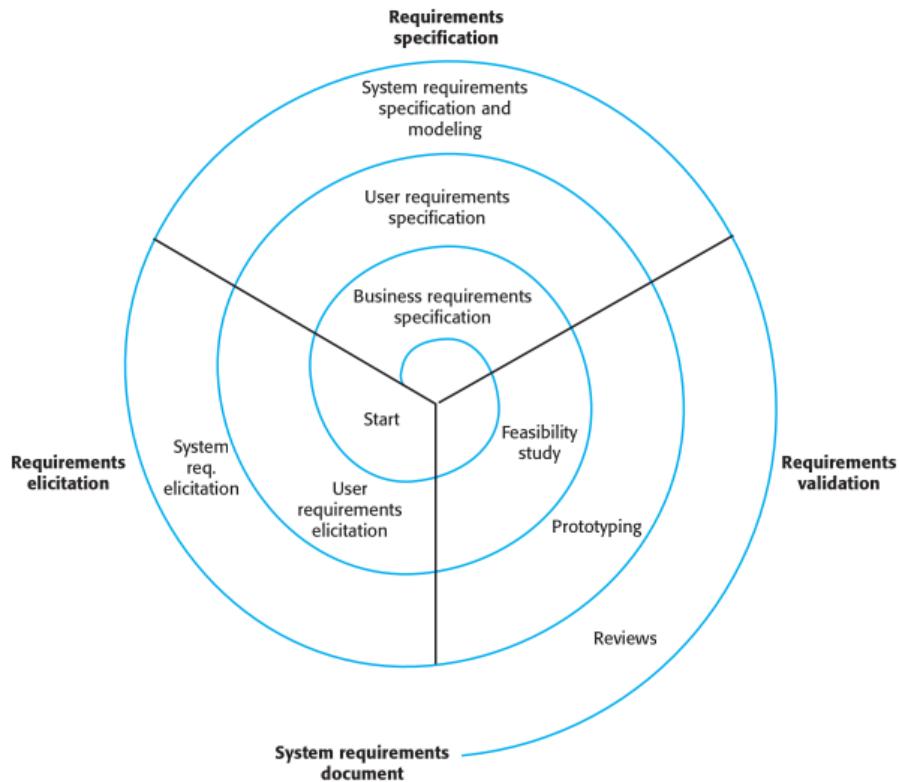


Figure 20.9: Scrum sprint cycle. From

#### Requirements elicitation

*Requirement elicitation* aims to understand the work and processes of the stakeholder to design the system for them and their needs. From this experience, requirements can be constructed.

Some techniques are:

- *Interviewing* which has two types

- Closed interviews
- Open interviews

However, begining with closed questions and then letting openended answers and discussions arise is often more preferable.

- *Ethnography* is an observation teqnique in order to study the stakholders day-to-day work.

*Stories* and *scenarios* are also usfull to construct to understand the complete workflow of the designing system.

### Requirements specification

The different ways to specify a requirement are

- *Natural language sentences*
- *Structured natural language*
- *Graphical notations*
- *Mathematical specifications*

### Requirements validation

Requirement validation is the process of determineng if the requirements are good or not. A number of checks can be maid to validate this:

- *Validity checks*
- *Consistency checks*
- *Completeness checks*
- *Realism checks*
- *Verifiability*

### Requirements change

It is inevitable that requirments changes. Requirements management is the process of controlling and managing these changes.

#### 20.1.5 System modeling

System models are used to make a abstract represenation of the system.

- *Context models*
  - Activity diagrams, shows the activities in the process. See 20.10.
- *Interaction models*
  - Use case diagrams, shows the interactions of the user. See 20.11
  - Sequence diagrams, shows the interaction between the user and the different objects of the system. See 20.12
- *Structural models*
  - Class diagrams, shows how the structure and inheritance of the classes for OOP development. See 20.13
- *Behavioral models*

- State diagrams, shows how the system reacts to internal and external events. See 20.14

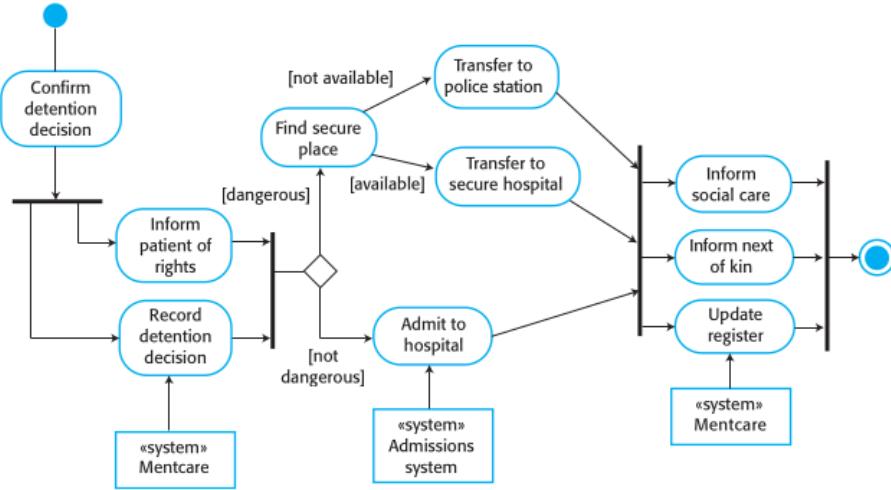


Figure 20.10: Activity diagram. From

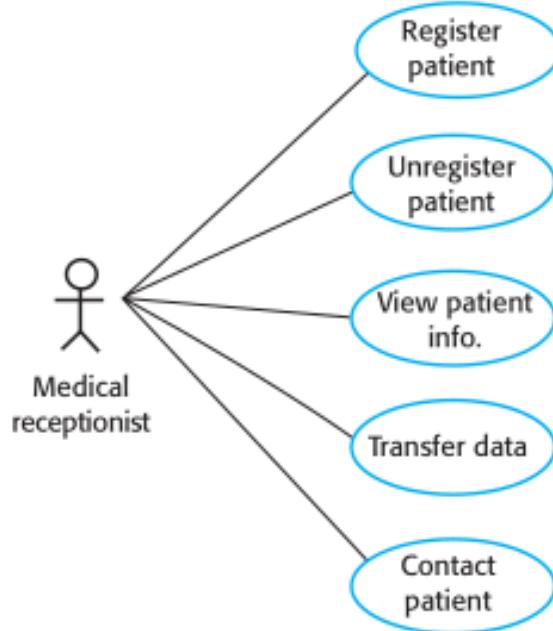


Figure 20.11: Scrum sprint cycle. From

### 20.1.6 Architectural design

Architectural design decisions one has to consider is performance, security, safety, availability, and maintainability.

The Architectural views are logical view, process view, development view, and phisical view.

There are severla Architectural patterns, which can be shousen dependent on the system. MVC, layered architecture, Repository architecture, and pip and filter architecture.

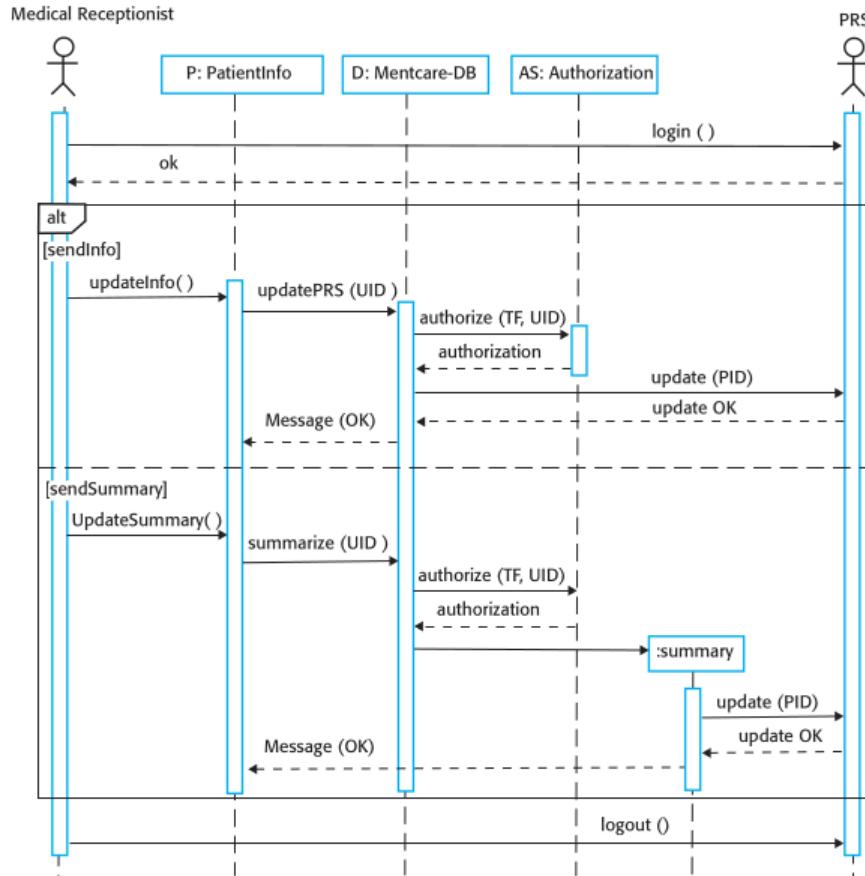


Figure 20.12: Scrum sprint cycle. From

Application architectures help us define commonalities with different system and thus design similar systems as such. Some of this include Transaction processing system, and Language processing system.

### 20.1.7 Software testing

*Validation* is whether the product is what it is suppose to be. *Verification* is if the product work properly.

#### Development testing

##### Unit testing

- Should test the different states of the object.
- A automated test consists of three parts: setup, call, and assertion.
- There are two strategies for determining which tests to create: Partition testing and Guideline-based testing.

##### Component testing

- Components are several interacting objects.
- Interface errors are the most common error in complex systems. These types of tests are best tested with component testing.

##### System testing

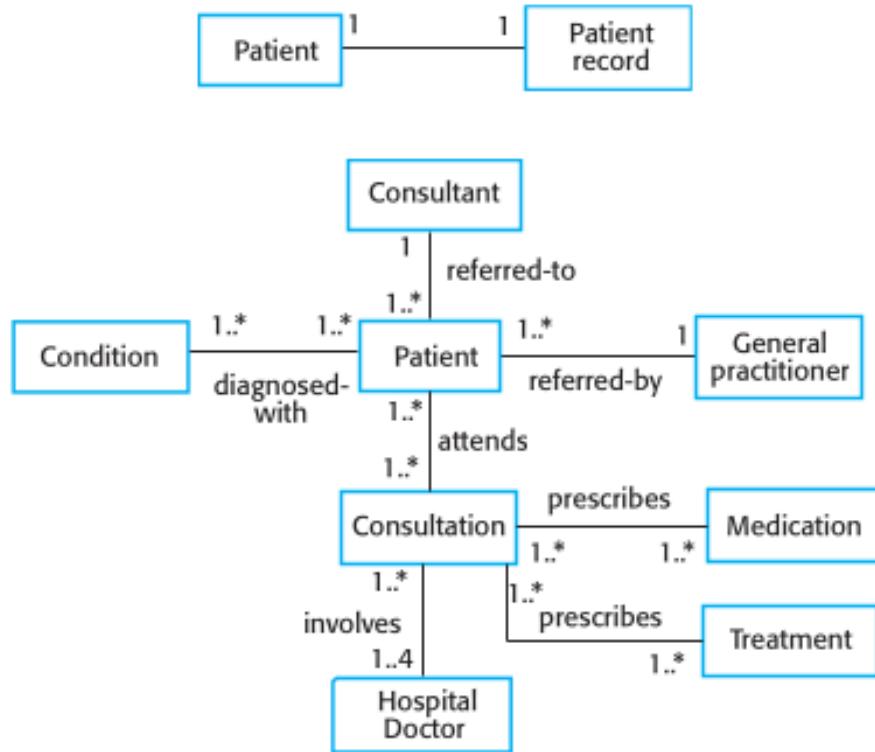


Figure 20.13: Scrum sprint cycle. From

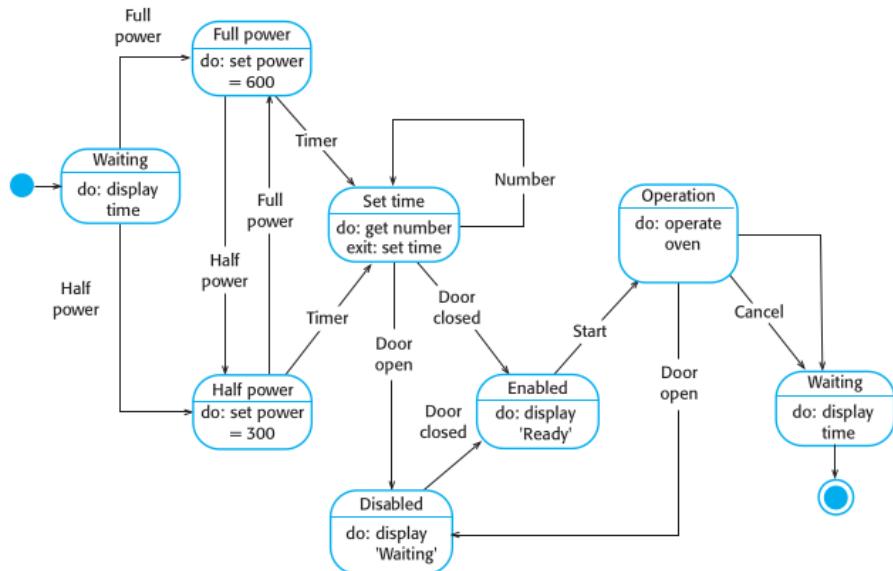


Figure 20.14: Scrum sprint cycle. From

- Testes the system as a hole and the diffrent functionality of the system.
- The tests are not just from one development team but can be a combination of diffrent teams.
- Often there exist a dedicated system tester.

- A sequens of operations makes up a system test.

### Test-driven development

Test-driven development (TDD) is a approach were the tests are first developed and then the implementation of the system. This approach has often better test coverage and is simpler to debugg.

### Release testing

Is the process of testing releases, with a focus on validation.

*Requirements-based testing*, is to validate and verify the requirements is fulfilled.

*Scenario testing*, test a use case of the system. It is generally good to test the combinations of requirements.

*Performance testing*, is good to test non-functional requirements of performance. Stress test can also be useful.

### User testing

- Alpha testing, a selected group test that work closely with the developers give feedback.
- Beta testing, a release that is made available for a large group.
- Acceptance testing, where the customer test if it is ready to be deployed in the customer environment.

## 20.2 Part 2 System Dependability and Security

## 20.3 Part 3 Advanced Software Engineering

## 20.4 Part 4 Software Management

## Chapter 21

# Introduction to Parallel Programming

### Resources

- <https://scrumguides.org/>
- <http://www.agilemodeling.com/artifacts/userStory.htm>
- <https://www.uml-diagrams.org/>

### 21.1 Intro

*Sequence* is the instructions that execute *sequentially*.

*Concurrent*: independent tasks. *Parallel*: execution at the same time

*speedup*:  $S = \frac{T_{old}}{T_{new}}$

Amdahl's Law

$$\text{Speedup} = S(n) = \frac{1}{1 - p + \frac{p}{n}}. \quad (21.1)$$

where  $p$  is the parallel fraction and  $1 - p$  is the sequential fraction.

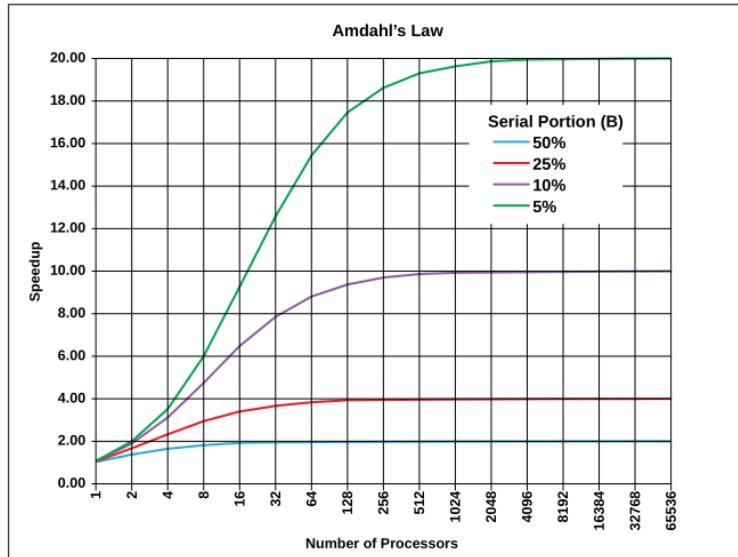


Figure 21.1: Amdahl's Law. From

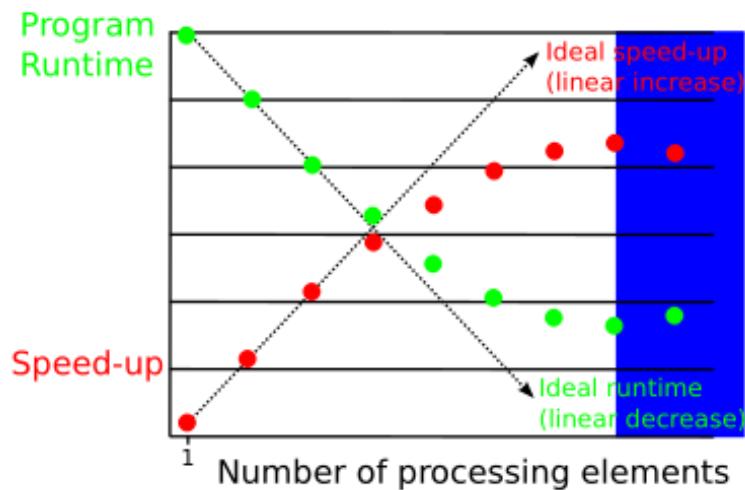


Figure 21.2: Parallel Slowdown. From

### 21.1.1 Why parallel computing?

- Smaller transistors = faster processors
- faster processors = increase power consumption
- increase power consumption = increase heat
- increase heat = unreliable processors

This is why the focus has been on more cores and making programs parallel will then utilize the resources of the physical processor better.

Parallelization of a serial program can either be done with manually rewriting the program so that more parts become parallel. It can also be done with a translation programs but this is very difficult to do such programs and it has a limited success rate.

If we were to have a addor for a sequens of numbers. A parralel approach would be to split the seqens up to diffrent parts and then let the **master** core add the results. Or let the cours be pared with eachother and add there results together themselves.

We devide the the work like a teacher does with exams with the TA's. But this require coordination.

- *Communication* between the cores
- *Load balance* to split the work evenly
- *Synchronization* make sure that one does not get to far ahead of the rest.

There are diffrent types of parallel systems those with

- *Shared-memmory*, this require coordinate the cores for examening and updating shared memory locations.
- *Distributed-memmory*, this require communication to send data between cores. Network?

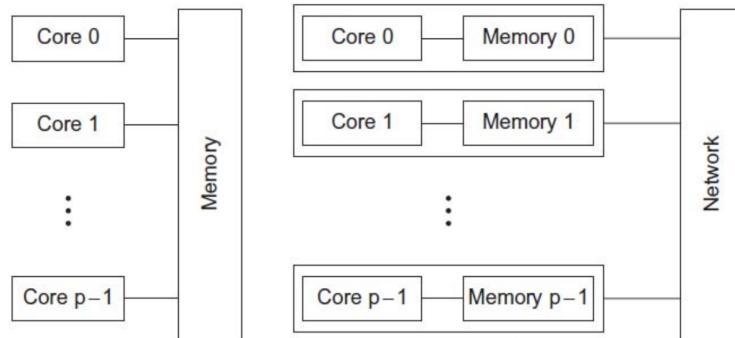


Figure 21.3: Types of parallel systems. From

### 21.1.2 Basic Concepts

#### Concurrent programing using shared memory

Good to know

- Most current multi-processor architectures used shared memory arhitectures.
- `printf` is thread-safe.
- *Deterministic* algorithm means that it will produce the same output every time. Like in a sequential algorithms, unless it depends on externam states.
- Concurrent programs are ofthen *non-deterministic*, since the output depends on scheduling desissions (*timing-sensitive*).
- *Race condition* is when the result depend on the timing of its execution.
- Shared memory doesnt scale well CPU-to-memory is a bottleneck.

*Race condition*

What might go wrong when two concurrent calls of count()?

```
int x = 0;

int y = x+1;
printf("%d", z);
% ||| concurrent task denotation
int z = x+1;
printf("%d", y);
```

*Check-Then-Act* error pattern

```
if (check(x)) { act(x); }
```

If x changes after it has been change by another process or thread then it will still act but it is unwanted behaviour since the if statement is no longer true. This can be fixed by using a lock.

*Read-Modify-Write* error pattern

What might go wrong when two concurrent calls of count()?

```
int counter = 0;
count() {
    counter++;
}
```

count++ is not a atomic operation, thus it is possible that after counter is read it is concurrently modified and therefore will increase by one instead of two.

This is preventing with *synchronization* to ensure *mutual exclusion* meaning that no two tasks execute parts of their *critical sections* at the same time.

*Dekker's algorithm*

```
// P_0:                      // P_1:
flag_0 = true;              flag_1 = true;
while (flag_1) {             while (flag_0) {
    if (turn != 0) {        if (turn != 1) {
        flag_0 = false;      flag_1 = false;
        while (turn != 0) {  while (turn != 1) {
            // busy wait       // busy wait
        }
        flag_0 = true;        flag_1 = true;
    }
}
}
// critical section          // critical section
...
turn = 1;                   turn = 0;
flag_0 = false;              flag_1 = false;
```

There are some limitations:

- only two process
- busy waiting is not efficient when comparing to suspending processes

- Assumes that the concurrent execution of P0 and P1 is equivalent to some interleaving of their instructions (which is often not the case on modern hardware).

Locks can however cause problems for instance *deadlocks* where two (or more) tasks are waiting for each other.

#### Deadlock example

Consider the following algorithm for copying a file:

1. Open the source file for exclusive access. (Assume that this blocks until no other process has the file open. Once this call returns, other processes that attempt to open the file block until the file has been closed again by the current process.)
2. Open the destination file for exclusive access.
3. Copy data from source to destination.
4. Close the destination file.
5. Close the source file.

What can possibly go wrong? Consider concurrent calls `copy("A", "B")` and `copy("B", "A")`.

Another problem that can occur is *livelocks* where two (or more) tasks are waiting for each other, like deadlock, but the tasks keep changing their state, without making progress.

#### Livelocks example

Real-life example: two people meet in a narrow corridor. Each tries to be polite by moving aside to let the other person pass.

Problem like *Resource starvation* can also occur, when a task is waiting for a resource that keeps getting given to other tasks

#### Starvation example

For instance, a (bad) scheduling algorithm might never schedule a task as long as there is another task with higher priority.

Although the resource will eventually be granted to the task (*fairness*). The waiting time is unknown.

*Data race* is when two or more tasks try to access the same shared memory location. This might even result in the program crashing.

#### Efficency

Doesn't scale well means that the speedup converges. We talk then about *efficiency*:

$$E(n, p) = \frac{S(n, p)}{p} \quad (21.2)$$

If the efficiency decreases as the number of threads increases we call that *weakly scalable*.

#### Concurrent programming using Distributed memory

- Each processor has its own private memory.
- For communication to occur messages have to be sent.
- Advantages:
  - No data race
  - scales better since it can use any number of processors.

- Disadvantages:
  - High latency to access for data from other process
  - Expencive to create memory for each
  - no uniform address space
- Memory access
  - Uniform memory access (UMA) Shared memory. All access the same way independent on the processor.
  - symmetric multi-processor (SMP) Shared memory. Treats all processors equaly.
  - Non-uniform memory access (NUMA) Shared memory. Depends on the memory location.
- Message passing (send/receive)
  - Synchronous: sender is blocked untill receiver calls receive.
  - Asynchronous: The message is buffered unteill the receiver calls receive.
  - Direct Communication: Send message directly to a spesified process
  - Channels: send message via a channel

### moore's law

#### Load balance

If we split the tasks in the most even way if we dont know how much time it will take then we can use a counter that make everyone take a small task. Like a ticket counter like in Kjell och kompani.

#### Mutual exclusion

- Safety Properties: nothing bad ever happens, e.g. no one take the same ticket.
- Liveness Properties: something good happens eventualy, e.g. eventialy some get a ticket. Fareness

#### Flag protocol

- Alice protocol
  - Raise flag
  - Wait until Bob's flag is down
  - Unleash pet
  - Lower flag when pet returns
- Bob protocol
  - Raise flag
  - While Alice's flag is up
    - \* Lower flag
    - \* Wait for Alice's flag to go down
    - \* Raise flag
  - Unleash pet
  - Lower flag when pet returns

## 21.2 Threads and locks

### 21.2.1 PThreads

Posix Thread API is the standard thread API. A instead of a function. Threads are a more portability and ease of use.

#### Stack

- Stack: when we call another function and the other function calles. So the latest will return before the once before so the last alocated is removed first.
- Each tread has to have its own stack because otherwise the stack would be brocken. (logical)
- The stack is allocated at the beginng and it is devided to the threads.

#### Commands

- `pthread_create`, starts and create
- `pthread_join`, will just wait everyone else
- `pthread_exit`, will just exit the thread, it is caled from a thread
- `pthread_mutex_lock`, lock the lock
- `pthread_mutex_unlock`, unlock the lock
- `pthread_mutex_init`, can use `std::mutex` but this create and initialaze the lock.

### 21.2.2 Locks

Types of mutexes

- Normal, if you call the lock again it deadlock. Will just stop if it is called. Less overhead needs no counter or anything.
- Recursive (re-entrant), allow to call mutex multiple times.
- Error-check, it give error if called multiple times.

*Producer-Consumer* is posible with locks.

*Condition variables* are varibles which can be used to synchronization by sending mesage/signal to the verible. If not try get a message and signals when it is not locked. this is done with `pthread_cond_wait`. also signal, broadcast, init, destroy. We run *try-lock* first then we do conditional variables if that doesent work. Most of the time it should be free so it is faster.

*Attributes* can be set to change the default attributes for threads, mutex, conditional varibles. For instance it is possible to spesify the size of the stack for that is created for each thread. And also if mutex should be normal, recursive, or error-check.

*reader-writer locks* is a good option when it expected to be many read but few writes. It can only be one writers but many reader. It will also wake up all readers when we write the locks.

*Barriers* are used to make every one stop until every one is there and then carry on.

Most of the current devices are *MIMD* (Multiprocessors). There are other kinds of achitectures like SISD (Uniprocessor), and SIMD (Vector). In MIMD we have Memory Contention (can't access the same memory at the same time), Communication contention (accessing the bus), and communication latency (time it takes to communicate memmory or other processor).

### 21.2.3 Spin-lock

We keep trying until we get it.

#### TAS

Test-And-Set Lock, a atomic operation that set a value of a boolean. The return is the prior value.

```
class TASlock {
    AtomicBoolean state =
        new AtomicBoolean(false);

    void lock() {
        while (state.getAndSet(true)) {}
    }

    void unlock() {
        state.set(false);
    }
}
```

**Remember Cache hit and Cache miss.** The cash is not necessarily shared between processes thus when data is accessed from memory it will be stored in the cash of the processor, when another processor access the same data it will talk with the bus which will say that another process has the data and get it from there cash instead of the main memory. When modifying the data it will have to say to the other process that it is obsolete and then when the other process access the data again it will take the value from the first processor's cash. This is called *Cache Coherence*. *Cash coherence problem* if we modify in some cache what should we update to. *Write-Back Caches*, we need a protocol to write the data back to memory and to the other processors' cash. Write-Back caches, it is expensive to write it to memory so we wait until the end? we postpone it. For this protocol a *cache entry* has three states *Invalid*, *Valid*, *Dirty*.

#### TATAS

We need to write to the lock to read it, two or more can not write to the bus at the same time. That is why we Test-And-Test-and-Set, check if it is free first. It is faster since we avoid memory collisions since only one is writing to the bus at the same time so we will likely write at the same time and then we need to fix it. One can write but many can look at it.

When I want to write to the data I say that to the bus first and therefore everyone else will say there's data is invalid. Cash is cold when its nothing in it. TAS Only one has the lock and we continue spinning on it and get a cash miss every time.

```
class TATASlock {
    AtomicBoolean state =
        new AtomicBoolean(false);

    void lock() {
        while (true)
            while (state.get()) {}
            if (!state.getAndSet(true))
                return;
    }
}
```

#### Exponential backoff lock

Exponential backoff, if there is a conflict again then we increase the delay.

Exponential Backoff lock

```

public class Backoff implements lock {
    public void lock() {
        int delay = MIN_DELAY;
        while (true) {
            while (state.get()) {}
            if (!lock.getAndSet(true))
                return;
            sleep(random() % delay);
            if (delay < MAX_DELAY)
                delay = 2 * delay;
        }
    }
}

```

contention problem if contention is unlikely then TAS is the best since it is less operation

?

We give up the process and when we get an indication when it is unlocked. Sequential bottleneck, we sequentialise the parallel program.

#### 21.2.4 Queue Locks

When there is contention on the lock it is better to use a queue lock. Since it prevent threads spinning only one lock, instead you spin on the lock that is not already taken.

Sometimes a thread need to abort a lock, but with a queue this is difficult since it is no wait free, it requires cleaning up and can't immediate return. Have a queue where thread spin on the first false slot in the queue.

##### Anderson queue lock

A array of false values that when each thread sets the most recent element to true when it wants to take the lock. This require the number of threads to be defined in order to create a array of correct size.

```

public lock() {
    mySlot = next.getAndIncrement();
    while (!flags[mySlot % n]) {};
    flags[mySlot % n] = false;      Memmory needed:
}
public unlock() {
    flags[(mySlot+1) % n] = true;
}

```

$$O(L \cdot N) \quad (21.3)$$

where L is the number of locks and N is the number of threads.

##### CLH Queue Locks

The size of the queue is dynamic unlike anderson queue lock. The queue is represented with a linked list. Initially there is only one element in the queue, then when the a thread takes the lock, it creates a new element, but still point to the first element. The next thread creates a new element and point to the element created by the first thread. The second thread spins on the element it points to (actually a copy). Then when the first releases the lock the second acquire the lock and only the element that it created remains. There is also a tail that points to the next element.

Memory needed:

$$O(L + N) \quad (21.4)$$

where L is the number of locks and N is the number of threads.

This does not work on unchached NUMA architecture, i.e. no caches.

```
class CLHLock implements Lock {
    AtomicReference<Qnode> tail;
    ThreadLocal<Qnode> myNode
        = new Qnode();

    public void lock() {
        Qnode pred = tail.getAndSet(myNode);
        while (pred.locked) {}
    }
    public void unlock() {
        myNode.locked.set(false);
        myNode = pred;
    }
}
```

### MCS Locks

Unlike CLH that spins on predecessor's memory, MCS spins on a local memory only, which make it faster memory access. It works similarly but nowait there is a linked list that point to a memory location local the each thread.

```
class MCSLock implements Lock {
    AtomicReference tail;
    public void lock() {
        Qnode qnode = new Qnode();
        Qnode pred = tail.getAndSet(qnode);
        if (pred != null) {
            qnode.locked = true;
            pred.next = qnode;
            while (qnode.locked) {}
        }
    }

    public void unlock() {
        if (qnode.next == null) {
            if (tail.CAS(qnode, null)
                return;
            while (qnode.next == null) {}
        }
        qnode.next.locked = false;
    }
}
```

## 21.3 OpenMP

OpenMP creates and terminate threads automatically. We give compiling instructions to spesify where and how this should be done by using directives based on the pragma compiler directives. We can think of it like pthread under the hood.

### 21.3.1 Loops

#### Loop carried dependencies

Loops can only be parallel if there is no *loop carried dependencies*.

*Check-Then-Act* error pattern

```
int i,j,A[MAX];
j = 5;
for (i=0;i<MAX;i++){
    j+=2;
    A[i]=big(j);
}
```

Example of modifying loop to remove loop carried dependencies.

```
int i,j,A[MAX];
#pragma omp parallel for
for (i=0;i<MAX;i++){
    j = 5 + 2*(i+1);
    A[i]=big(j);
}
```

This can be modified to:

#### Scheduling

- static define a strict chunk size which does not change over runtime.
- dynamic similar to counter when load balancing.
- guided chunk size is predefined but changes during runtime.
- automatic let the compiler define how to load balance.
- runtime scheduling is defined during runtime.

#### Reduction

```
double ave = 0.0, A[MAX]; int i;
for (i=0;i<MAX;i++){
    ave += A[i];
}
ave = ave/MAX;
```

*Reduction* is very common. This is a form of loop carried dependency. Since it depends on the previous iterations since you read and then update it so the final value is a summation of the iterations in sequence.

`reduction (op: list)` Op is the type of operation, e.g. “+”, and list is the variable that will change. This will create local instances of the variable and then add them together afterwards. The initial value is “0” for the “+” operations of the list. If it is times then it needs to be “1” since otherwise it would always be 0.

### 21.3.2 Synchronization

#### Barrier

OpenMP puts barriers for you, e.g. when there are two loops at a row. *Nowait*: don't put a barrier here. For those cases where it is no barrier by default you need to put a *barrier*. *master*: only the master thread does this, there are no synchronizations so we have to put a barrier afterwards if we want the other threads to wait until the master is done. *single*: only the first thread that reaches it will do the task. *Workshare* construct like *single* and *master* the compiler always puts a barrier afterwards. Another one is *section* which means that one will do this and others will do the other section.

## Looks

### 21.3.3 Environment Variables

- OMP\_NUM\_THREADS: set the number of thread to use. No garante.
- OMP\_STACKSIZE: limit the stack size to prevent system crash.
- OMP\_WAIT\_POLICY ACTIVE|PASSIVE: To spin on locks or put thread to sleep.
- OMP\_PROC\_BIND TRUE|FALSE: Bind thread to a spesific process for instance if you want to make sure it access the same cach and not have to access cach fether away in the cach hiracy.

### 21.3.4 Data Environment

Private on the stack. Shared on the heap.

- SHARED Every thread has the same version.
- PRIVATE Every thread has there local version. Un enitalised.
- FIRSTPRIVATE Same as private but initialized to the value given.
- LASTPRIVATE Save the last value of the variable.
- DEFAULT(PRIVATE|SHARED|NONE) Set the default.

## 21.4 MPI

Message Passing Interface (MPI) a library specification. Many implementations with subtle diffrences between implementations. As the name sugest data is chared by sending messages since processes do not share memory like theads does. MPI allowes distrebution of work accross multiple machines.

MPI when there is a error from MPI the program will crash. The MPI suppervices wil crash so all MPI programs runing will crash.

Processes can be collected in a *group*. Messages that are sent contain a *tag*, which sets the *context* of the message in order to achieve point-to-point communication. Groups and context together formes a *communicator*. Each process as a id called *rank* starting from 0.

There is a number for MPI data types that can be used to send messages.

```
MPI_Init
// Everything in between these two will run with MPI
MPI_Finalize
```

when we cal the exec program we spesify all other process so we can execute for multiple machines. Int tag the data, so we can determin wheter or not to process it by the reciver.

### 21.4.1 Basic functions

```
MPI_COMM_SIZE // number of processes involved in a communicator
MPI_COMM_RANK // rank of the calling process in the communicator
MPI_Send // blocking send
MPI_Recv // blocking receive
MPI_BCAST // broadcast to all other in the communicator
MPI_REDUCE // combine all data from all the processes to
```

### 21.4.2 Deadlock with MPI

Deadlock may still happen.

`Send(1) Send(0)` There is no space for we need to wait until it is space at the receiver.  
`Recv(1) Recv(0)`

Relying on the buffer to execute properly is unsafe since there might be deadlock, if not implemented correctly. If we want to debug we can run MPI\_Ssend instead. The extra “s” stands for synchronous which guarantees that it will block until it is received.

Non-blocking send/recv

```
Isend(1) Irecv(0)
Irecv(1) Isend(0)
Waitall Waitall
```

### 21.4.3 Broadcast and reduce

*tree-structure communication. Reduce* Need to specify a interval of .. *Allreduce* all processes do the reduction and get the value.

When using reduce you specify the number a value operator like MPI\_SUM or MPI\_LOR that does operation to all the results in a group.

MPI comes with support with measure performance/time MPI\_Wtime. There is also barriers (MPI\_Barrier). We should have a barrier before we measure time so one process does not finish before the other and measure the time.

### 21.4.4 Other functions

```
MPI_Comm_split // Create new communicator
MPI_Scatter // Similar to broadcast but send chunks of the data not the same
MPI_Gather // Similar to reduce but process 0 does all the work
MPI_Sendrecv // Both send and recv
```



# Chapter 22

## Real Time Systems

## 22.1 RTOS

A realtime OS needs to be *Determinism*, we want the ability to control the sequence of execution. *Responsiveness*, i.e. there should be short interrupt latency as well as fast context switching. It should also have a *small footprint*, i.e. use little storage. Some other requirements are *Support for timely concurrent processing*, which allow real-time, multi-tasking, and synchronization. Also, *User control over OS policies*, which allow CPU scheduling and memory management.

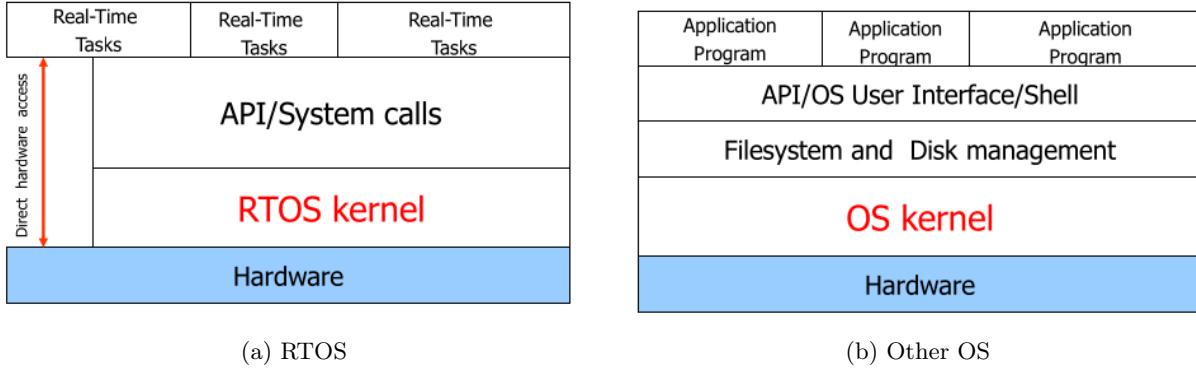


Figure 22.1: Realtime OS and other OS, ???

### 22.1.1 Time management

The hardware has a timer which interrupts the processor at a fixed rate (i.e. *Time Interrupt*). Each time interrupt is called a system *tick*. For each time interrupt routine the cyclic task will start again when a period has past.

### 22.1.2 Task management

Timing constrained, time-driven.

Periodic tasks described with 3 parameters (C,D,T) where

- C = resource budget
- D = deadline
- T = period (e.g. 20ms, or 50HZ)

Often D=T, but it can be D<T or D>T

*Time-driven* means that the task is depending on time, e.g. periodic tasks. There is also *Event-driven*, which are instead dependent on a interrupt.

- C = resource budget
- D = deadline
- $T_{min}$  = minimum interarrival time

It is not the end of the world the deadline is greater than the period since it want relay effect the system in a negative way. But it is easier to conceptualize and work with if the deadline is within the period.

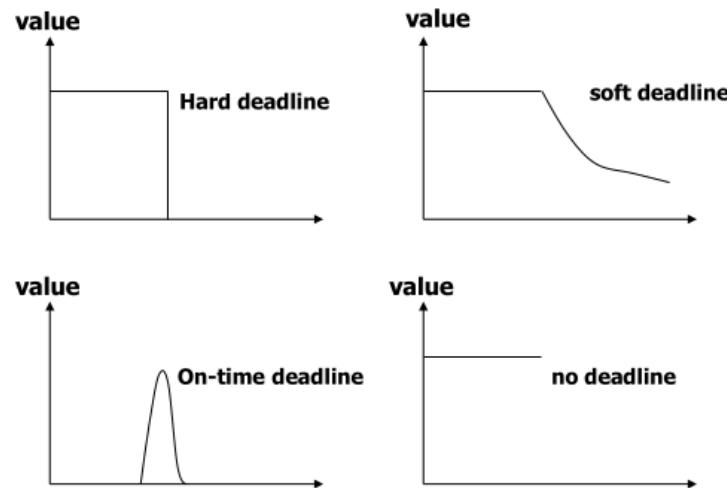


Figure 22.2: Timing Constraints, ??

Task states

- Ready
- Running
- Waiting/blocked/suspended ...
- Idling
- Terminated

*TCB* (Task Control Block) is the meta data of the task.

### 22.1.3 Interrupt handling

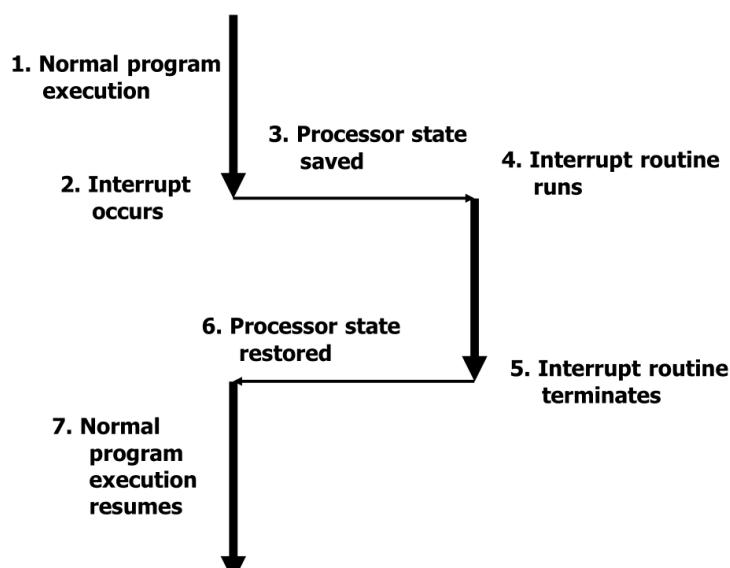


Figure 22.3: Handling an Interrupt, ??

### 22.1.4 Memory management

Memory is handled tru the standard method with Block-based, Paging, hardware mapping for protection. For some cases there are no virtual memory, i.e. Lock all pages in main memory. Several embedded RTS does not have memory protection since it is not needed for a correctly design system.

### 22.1.5 Exception handling

*Exceptions* e.g missing deadline, running out of memory, timeouts, deadlocks, divide by zero, etc. Watch-dog we create a monitor that looks at some result that is a seperate tasks.

### 22.1.6 Task scheduling

Most important. Sort the READY queue acording to

- Priorities (HPF)
- Execution times (SCF)
- Deadlines (EDF)
- Arrival times (FIFO)

Classes of scheduling algorithms

- Preemptive vs non preemptive
- Off-line vs on-line
  - Static vs dynamic: Static has less overhead than dynamic since a dynamic scheduling algorithm like EDF has to constantly check which deadline is the closes. However dynamic like EDF is the optimal scheduling algorithm for single core preemptive processor. Static is less flexible as it can not change during runtime to better fit the task set. Dynamic doesn't need to save a static schedule or priority ordering of the task set, but static need until the hyper period.

Interrupt it is a vector of bits so 8 bit has 8 interrupts and the first has the highest priority and the last is the lowest.

### 22.1.7 Task synchronization

Synchronization primitives can be used such as *spinlock* and *semaphores*.

Potential synchronization issues that can occurs are *deadlock*, *livelock*, *starvation*, which can be caused by critical sections. Also *priority inversion* a higher-priority job can block a medium-priority job.

### 22.1.8 Types of tasks

- Sporadic tasks: it reoccurs in random instances and has hard deadlines.
- Asynchronous periodic tasks: instead have  $O_i$  aka offset/ the arrival time,  $C_i$ ,  $D_i$ ,  $T_i$ .
- synchronous periodic tasks: All tasks arrive at he same time.

SAS: the hardest sequence that is a set of sporadic tasks is synchronous and then release as early as possible. Meaning that to analyze sporadic task we treat them as synchronous periodic tasks.

## 22.2 ADA

Standard for automotives **AUTOSAR**.

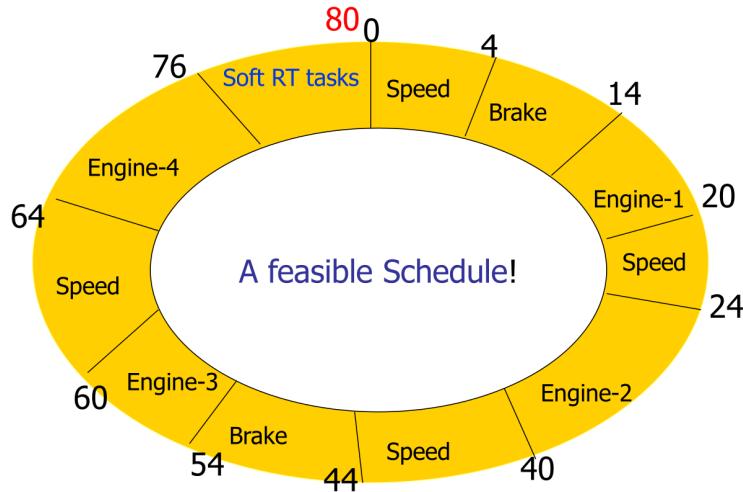


Figure 22.4: Handling an Interrupt, ??

**Worst-Case Response Time (WCRT).** A task might not finish its execution due to *preempty*, i.e. remove the task from executing and then let the another task get the computing resources. If we have a cold cash then it will take longer, and therefore need to be taken into consideration when calculating WCET.

A job is a execution of a task. A arraying time, C computing time, D deadline.

The delay will cause a delay for at least a specified time interval. The delay until causes a delay until an absolute wake-up time.

For tasks ADA let you specify entries which is defined in the a select statement. It is often you have the select statements in a loop as it should be possible to call a task entry multiple time. However, eventhough it is a loop it doesn't contently run over and over again. It instead wait until someone calls the entry.

## 22.3 Scheduling

**Classical scheduling theory** (e.g., in operations research) generally deals with *finite* processes (job-shop, flow-shop &c.) to *optimize* some metric.

**Real-time scheduling theory** generally deals with *infinite* processes (control loops &c.) to *guarantee* a safety specification.

- Task models: Formalisms to specify workload and timing constraints.
- Scheduling algorithms: Run-time strategies for scheduling workload.
- Analysis: Offline methods for proving timing safety.

A job  $j_i$  is geven by a triple  $(A_i, C_i, D_i) \in \mathcal{N}^3$ , where

- $A_i$  is the arrival time (or release time),
- $C_i$  is the worst-case execution time (WCET), and
- $D_i$  is the deadline.

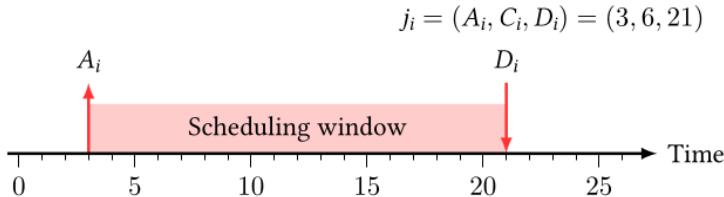


Figure 22.5: Scheduling window, ??

We are **assuming** 1. All jobs are independent, 2. A single processor, and 3. Fully preemptive or non-preemptive scheduling.

Context switching

- *Preemptive*: allow jobs to be paused and resumed later.
- *Non-preemptive*: don't allow context switching, the job who has CPU time will run until the end.

Important definitions

- *Schedulability*: iff the scheduling algorithm schedules the set of jobs without deadline misses.
- *Feasibility*: iff there exists a scheduling algorithm that is schedulable.
- *Optimality*: iff all sets of feasible schedules are also schedulable.

Test definitions:

- *Sufficient test*: iff by passing the test means that the system is schedulable.
- *Necessary test*: iff by failing the test means that the system is unschedulable.
- *Exact test*: iff sufficient and necessary.

Schedulability test we compute that the response time is within all deadlines, then it is schedulable. EDF can be used to test since if it is not feasible with EDF it is the jobs are not schedulable. Preemptive EDF, the first argument is when the task arrives.

- *Implicit deadlines*: if  $D_i = T_i$  for all  $\tau \in \mathcal{J}$ . Is when the deadline and period are the same.
- *Constrained deadlines*: if  $D_i \leq T_i$  for all  $\tau \in \mathcal{J}$ .
- *Arbitrary deadlines*: if  $D_i$  and  $T_i$  are unrelated.

Other terminology:

- *Priority ordering*: tells us who has the highest priority.
- *Critical instant schedule*: shows what tasks have the resources, and it is used to determine if deadlines are met.
- *Utilization*: see equation. executing time / period ( $C_i/T_i$ ).
- *Higher periodicity*:  $hp\{\mathcal{J}\}$ .

### 22.3.1 Scheduling algorithms

A static schedule is a schedule for a hyper period which will then repeat over and over again. A static priority is a priority that won't change.

- Dynamic priority: the priority changes during run time
  - EDF: The task with the earliest deadline is the task with the highest priority.

- Fixed Priority (FP): A unique priority is set before run time.
  - DM: Deadline-Monotonic ordering (DM). Tasks with shorter relative deadlines get higher priorities. It is *optimal priority ordering* for synchronous periodic task sets with *constrained deadlines* executed on a single preemptive processor.
  - RM: Rate-Monotonic ordering (RM). Tasks with shorter periods get higher priorities. It is *optimal priority ordering* for synchronous periodic task sets with *implicit deadline* executed on a single preemptive processor.

	cons.DL dynamic static	impl.DL EDF $U \leq 1$ DM	anbitus,DL EDF dbf ?

### 22.3.2 Schedulability tests and analysis

#### Schedulability test (Jackson)

$\mathcal{J} = j_1, \dots, j_n$  is ordered with non-decreasing deadline and the arrival time is zero. Then,  $\mathcal{J}$  is EDF schedulable iff

$$\forall i, 1 \leq i \leq n : \sum_{k=1}^i C_k \leq D_i$$

#### WCRT and schedule

Worst case Response time for constrained deadline with preemptive FP-scheduling on a single processor:

$$R_i = C_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (22.1)$$

#### Draw for a schedule

Draw a schedule for FP with RM schedule algorithm.

Task	$C_i$	$T_i$	$D_i$
$\tau_1$	2 ms	10 ms	10 ms
$\tau_2$	4 ms	15 ms	15 ms
$\tau_3$	10 ms	35 ms	35 ms

Table 22.1: A number of periodic tasks with  $D_i = T_i$

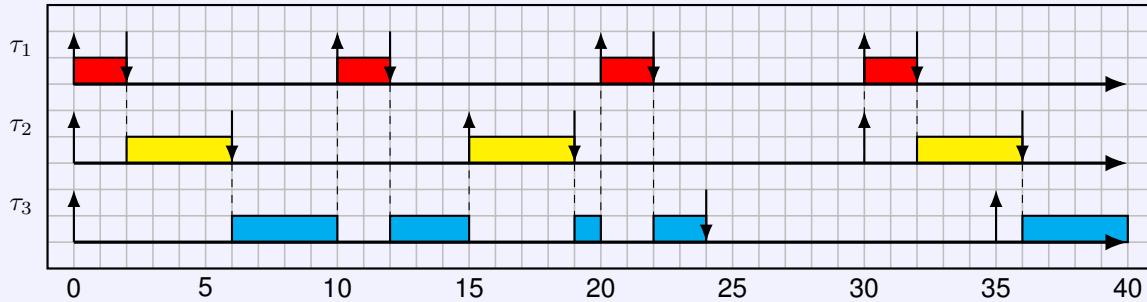


Figure 22.6: Critical instant schedule for the task set in 22.1

### WCRT analysis

Calculate WCRT of set  $\mathcal{J} = \{(1, 4, 4), (2, 5, 5)\}$

$$R_i = c_i + \sum_{t_j \in hp(t_i)} \left\lceil \frac{R_i}{T_j} \right\rceil * C_j$$

first

$$R_1^1 = c_1 + 0 = 1$$

second  $hp(t_2) = t_1$

$$R_2^1 = c_2 = 2$$

$$R_2^2 = c_2 + 2[\frac{2}{4}]1 = 2 + 1 = 3$$

$$R_2^3 = 2 \left\lceil \frac{3}{4} \right\rceil 1 = 2 + 1 = 3$$

reached a fixed point

It might go to infinity. In the works case it will get a fixed time or if it just increases by one  $R_i \leq D_i$  the task with the lowest priority is the works case scenario so we only need to calculate it for that one.

EDF us absolute deadline and DM use relative deadline.

### Utilization

The utilization of a single task is the fraction between the execution time and the period of the task:

$$U(\tau_i) = \frac{C_i}{T_i} \quad (22.2)$$

Tasks with implicit deadlines is FP schedulable with RM priority ordering on a single preemptive processor if:

$$U(\mathcal{J}) = \sum_{\tau_i \in \mathcal{J}} U(\tau_i) = \sum_{i=1}^n \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \quad (22.3)$$

where U is utilization, n is the number of tasks. When n goes to infinity the utilization will be bounded by 0.69. This is a *sufficient* test but not *necessary*, i.e. if it is true then it is schedulable but if it false it might still be schedulable.

$$\lim_{n \rightarrow \infty} n(2^{1/n} - 1) = \ln(2) \approx 0.69 \quad (22.4)$$

For EDF it the utilization is instead bounded by 1. This is a *exact* test, i.e. we dont know that it is schedulable if it is within the bound.

$$U(\mathcal{J}) \leq 1 \quad (22.5)$$

It is also true for RM priority is scheduleble if:

$$\prod_{\tau_i \in \mathcal{J}} (U(\tau_i) + 1) \leq 2 \quad (22.6)$$

Test utilization bound for RM

$$t_1 = (1, 3, 3) \text{ and } t_2 = (3, 5, 5)$$

$$\begin{aligned} U(\mathcal{J}) &= 1/3 + 3/5 \leq n(2^{\frac{1}{n}} - 1) = 2(2^{\frac{1}{2}}) - 2 \\ 8/15 &\approx 0.53 \leq 2\sqrt{2} - 2 \approx 2.83 - 2 = 0.83 \end{aligned}$$

On multicoreprocessor then sporadic task set with implicit deadline on  $m$  preemptive processors using EDF, than a sufficient bound is

$$U(\mathcal{J}) \leq \frac{m+1}{2} \quad (22.7)$$

### Demand Bound Function

Demand bound function (DBF) is often used to analyses DBF since it does not have a utilization bound like RM and EDF.

$$dbf(\mathcal{J}, t_1, t_2) = \sum_{j_i \in \mathcal{J}} dbf(j_i, t_1, t_2) \quad (22.8)$$

To create a DBF graph you first create a DBF graph for each task in the set. Let the x axis be the each period and the y axis is the execution time.

A job set  $\mathcal{J}$  is feasible on a single preemptive processor iff

$$\forall t_1, t_2 \text{ such that } 0 \leq t_1 \leq t_2 : dbf(\mathcal{J}, t_1, t_2) \leq t_2 - t_1 \quad (22.9)$$

if we set  $t_1$  to be 0 then dbf is bounded by  $t$ , which has a derivative of 1.

$$\forall t, \text{ such that } 0 \leq t \leq t_2 : dbf(\mathcal{J}, t) \leq t \quad (22.10)$$

However, there is no need to test for all  $t$  it enough to check until the slope  $U(\mathcal{J})$  and the line  $t$  crosses.

$$0 \leq t \leq HP(T) + \max_{\tau_i \in \mathcal{J}} D_i \quad (22.11)$$

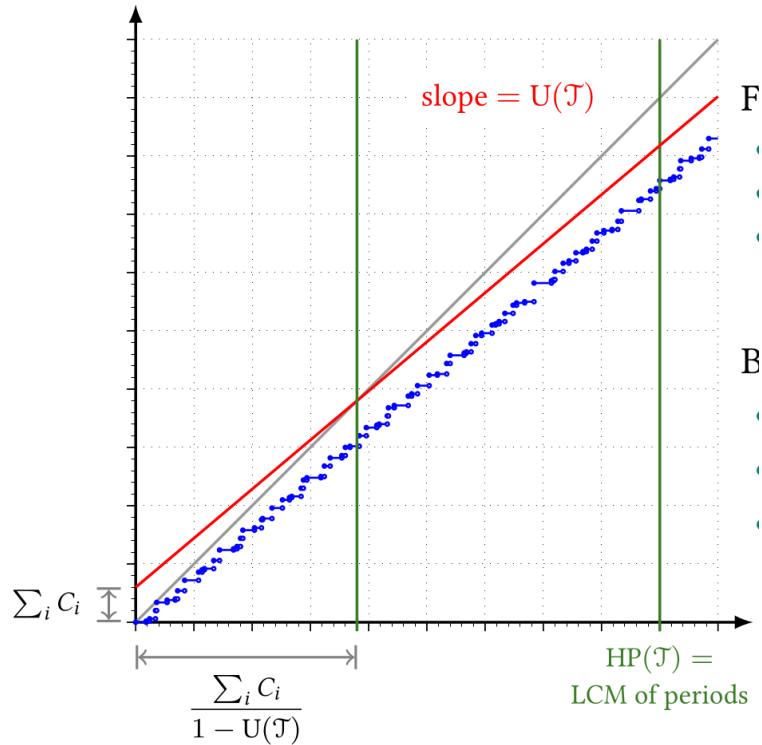


Figure 22.7: DBF graph

The hyper period is calculated by finding the least common multiple. This only work for a shedable processor.

#### Hyper period

Let task set  $\mathcal{J} = \{(2, 7, 8), (4, 8, 9), (3, 10, 10)\}$

$$8 = 2 \cdot 2 \cdot 2$$

$$9 = 3 \cdot 3$$

$$10 = 5 \cdot 2$$

Then the hyper period is:

$$2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 5 = 4 \cdot 9 \cdot 10 = 36 \cdot 10 = 360$$

#### dbf

Then draw it out with will be stepwise function with define length of each lines. each step is one unit in between (the y-axis)

Don't need to test all, there is a bound only up to the Hyper period plus max relative deadline

If all deadlines are implicit it is a lot easier whe need just to check that the total utilization is less then 1 see the formula above for utilization.

### 22.3.3 Jitter

Jitter can be due to Tick-driven scheduler, since the timing-interupps accures every 10th ms so we could be unlocky and have 10ms jitter.

varying execution time that start another task is another source of jitter

Jitter, not exact or other tasks

$$R_i = w_i + j_i$$

$$w_i = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{w_i + j_j}{T_j} \right\rceil C_j$$

an optimistic view, so we can see the maximum allowed jitter

## 22.4 Workload Models

Several real-time systems contain functionality of different *importance*, or *criticality*. Such systems are sometimes called mixed-criticality systems.

In such system let every task  $\tau_i$  have two WCET estimates  $C_i^{HI}$ , pesimistic estimation, and  $C_i^{LO}$ , not pesimistic estimation.

$$C_i^{LO} \leq C_i^{HI} \quad (22.12)$$

Were the criticality level  $\chi_i \in \{LO, HI\}$ , i.e { Low criticality, High criticality}.

Criticality

Let,

$$\begin{aligned} \mathcal{J} &= \{\tau_1, \tau_2, \tau_3\} \\ &= \{(1, 1, HI, 4, 4), (1, 2, HI, 3, 4), (3, 3, LO, 9, 10)\} \end{aligned}$$

Then the critical instance schedule will use only the HI value when locking at WCET for a task and only on the LO for task with low criticality. See Figure 22.8 and Figure 22.9.

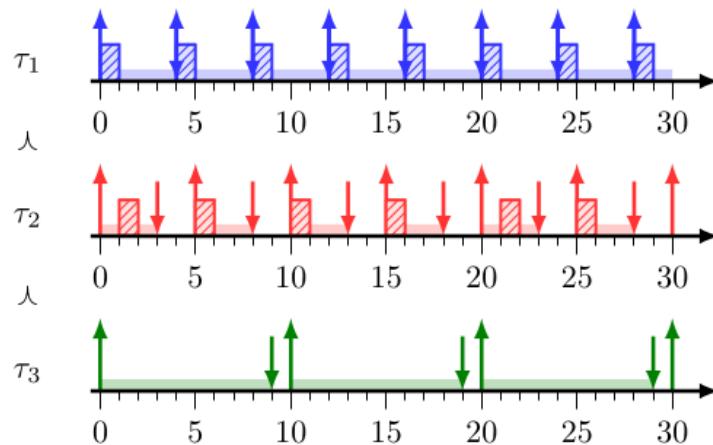


Figure 22.8: Timing Constraints, ??

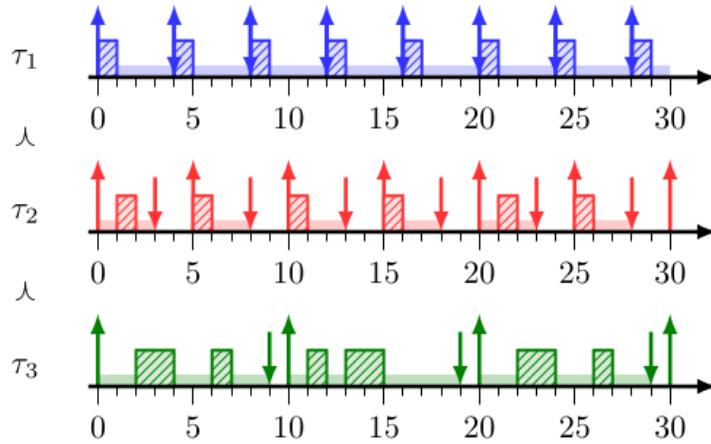


Figure 22.9: Timing Constraints, ??

**Correctness criteria for mix-criticality.** We say that scheduling is correct iff

1. All jobs meet their deadlines when all actual execution times stay below the  $C_i^{LO}$ 's in the current run of the system.
2. Jobs from high-criticality tasks meet their deadlines when all actual execution times stay below the  $C_i^{HI}$ 's in the current run of the system.

Schedulable test

$$R_i = C_i^{\chi_i} + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{\chi_i} \quad (22.13)$$

Not all sets of tasks are sporadic.

- The general multiframe (GMF) Task Model: cyclic
- The Recurring Branching (RB) Task model: different branches
- The Digraph real-time model (DRT): different branches and circles

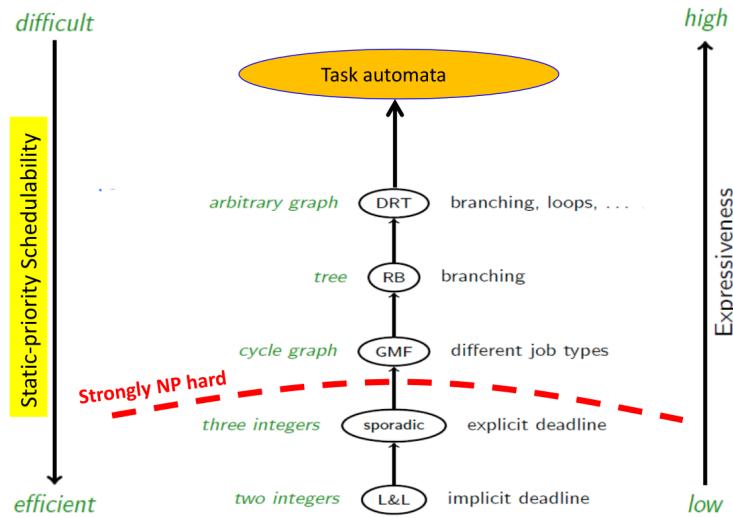


Figure 22.10: Hierarchy of Models

Demand Bound Function (DBF).  $C_{max}$  add up every execution demand, i.e. the first element of the demand pair for all elements in the task model.

## 22.5 Synchronization

### 22.5.1 Blocking

Blocking occurs when one or more lower priority tasks use a semaphore that a higher priority task wants, thus worsen the WCRT.

If we allow blocking, then the worst case execution time is:

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (22.14)$$

Un-bounded priority inversion occurs when a lower priority task has taken the semaphore and it is not allowed to run since a higher priority task is running, which is lower priority task than the task who wants the semaphore.

To solve priority inversion there are a few resource access protocols.

- Highest Priority Inheritance
  - Non preemption protocol (NPP)
- Basic Priority Inheritance Protocol (BIP)
  - POSIX (RT OS standard) mutexes
- Immediate Priority Inheritance
  - Highest Locker's priority Protocol (HLP)
  - \* Ada95 (protected object) and POSIX mutexes
- Priority Ceiling Protocols (PCP)
  - The general one

### 22.5.2 Non preemption protocol (NPP)

The one who successfully grabs the semaphore will get the highest priority. Meaning that it will not be preempted as no other task has higher priority. However, lower priority task will then block higher priority tasks.

### 22.5.3 Basic Priority Inheritance Protocol (BIP)

The one who tries to get a semaphore that another already have taken will switch priorities.

- A gets semaphore S
- B with higher priority tries to lock S, but can't since A has it
- B transfers its priority to A, so that A runs with B's priority.

image code

```
P(scb):
    Disable-interrupt;
    If scb.counter>0 then {scb.counter - -1;
                            scb.holder:= current-task
                            add(current-task.sem-list,scb)}
    else
        {save-context( );
        current-task.state := blocked;
        insert(current-task, scb.queue);
        save(scb.holder.priortiry);
        scb.holder.priority := current-task.priority;
        schedule();
        load-context() }
    Enable-interrupt

V(scb):
    Disable-interrupt;
    Restore current-task.priority (with "its original priority")
    If not-empty(scb.queue) then
        { next-to-run := get-first(scb.queue);
        scb.holder := next-to-run;
        next-to-run.state := ready;
        insert(next-to-run, ready-queue);
        save-context();
        schedule();
        load-context() }
    else scb.counter ++1;
    Enable-interrupt
```

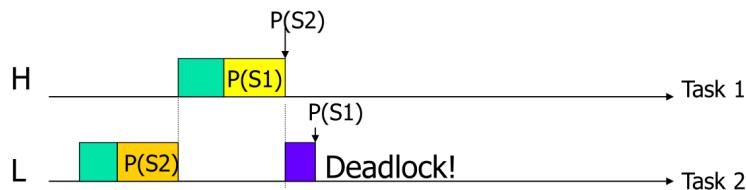


Figure 22.11: Potential deadlock with BIP scheduling

### 22.5.4 Immediate Priority Inheritance and HLP

The task that grabs the semaphore will directly get the priority which is assigned to that semaphore.

	Priority	Share Semaphors	Ceiling of semaphors
Task 1	H	S3	$C(S1)=M$
Task 2	M	S1, S	$C(S2)=L$
Task 3	L	S1, S2	$C(S3)=H$
Task 4	Lower	S2, S	$C(S)=M$

Figure 22.12: HLP example of ceiling table

```

P(scb):
    Disable-interrupt;
    If scb.counter>0 then
        { scb.counter -= 1;
          current-task.priority := Ceiling(scb) }
    else
        { save-context();
          current-task.state := blocked
          insert(current-task, scb.queue);
          schedule();
          load-context() }
    Enable-interrupt

V(scb):
    Disable-interrupt;
    Restore current-task.priority
    If not-empty(scb.queue) then
        next-to-run := get-first(scb.queue);
        next-to-run.state := ready
        insert(next-to-run, ready-queue);
        save-context();
        schedule(); /* dispatch invoked*/
        load-context();
    end then
    else scb.counter +=1;
    end else
    Enable-interrupt

```

### 22.5.5 Priority Ceiling Protocols (PCP)

PCP is an extension of PIP and HLP.

	NPP	BIP	HLP	PCP
Bounded Priority Inversion	yes	yes	yes	yes
Deadlock free	yes	no	yes	yes
Un-necessary blocking	yes	no	yes/no	no
Blocking time calculation	easy	hard	easy	easy
Number of blocking	1	> 1	1	1
Implementation	easy	easy	easy	hard

## 22.6 Multiprocessor scheduling

### 22.6.1 Multiprocessor Scheduling of Task Graphs

The response time will be bounded with graham's bound

$$R \leq \text{len}(G) + \frac{\text{vol}(G) - \text{len}(G)}{m}$$

Where  $\text{len}(G)$  is the length of the longest path,  $\text{vol}(G)$  is the total workload, and  $m$  is the number of processors. This is somewhat logical bound since we know that  $R \geq \text{len}(G)$  and to get the work case we need to add the workload that could theoretically be scheduled before the jobs in the longest path.

Utilization bound

- Global Scheduling: formula?
- Partitioned Scheduling: 50% for each queue?
  - First Fit manner: highest priority first.
- Partitioned Scheduling with Task Splitting:
  - Lakshmanan's algorithm: 65% for each queue
  - breadth-first partitioning algorithm: 69% for each queue (same as RM)
  - preassigning heavy task:  $\frac{C_i/T_i}{M} \leq N(2^{1/N} - 1)$

Heavy task is tasks with utilization higher than 0.41.

Advantages and disadvantages for scheduling algorithms

- Global Scheduling:
  - (+) Supported by most multiprocessor os.
  - (-) No optimal algorithm
  - (-) Poor resource utilization for hard RT.
  - (-) Experiences scheduling anomalies
- Partitioned scheduling:
  -
- Partitioned scheduling with task splitting:
  - (+) Better resource utilization.
  - (-) Task splitting requires more preemption and migration.

### 22.6.2 Multiprocessor Scheduling of Task Graphs

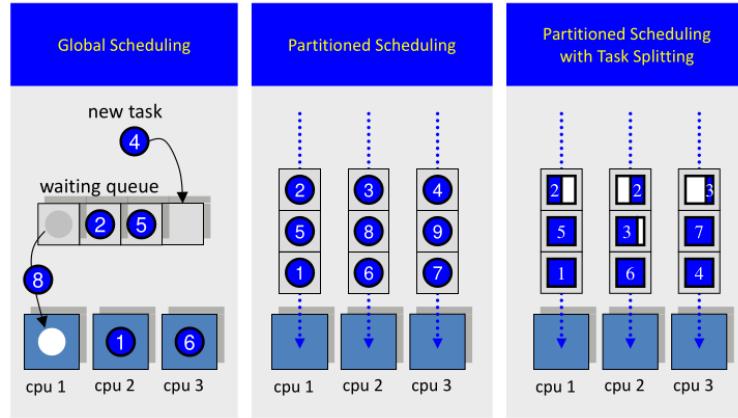


Figure 22.13: Multiprocessor Scheduling of sequential tasks, ??

EDF is not optimal, Fixed priority suffers from Dhall's anomali. Dhall's anomali happens for instance when there is 3 tasks that are scheduled on 2 cpu's and the shortest jobs will be scheduled first thus the longer job will be scheduled after a shorther job thus worsening the WCET.

There is also Richard's anomali, increasing the number of processors might make the schedule wors.

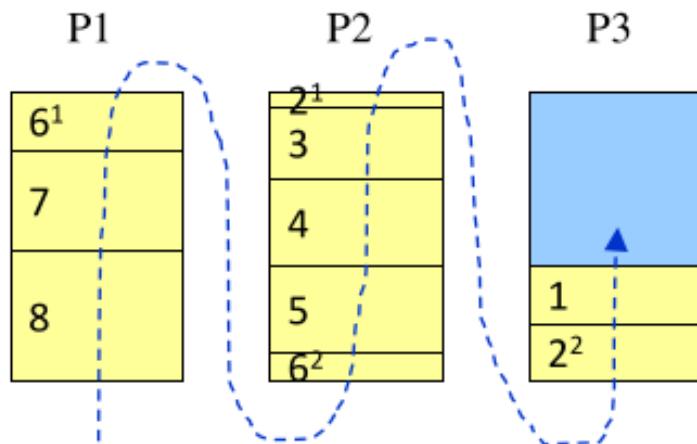


Figure 22.14: Lakshmanan's Algorithm, ??

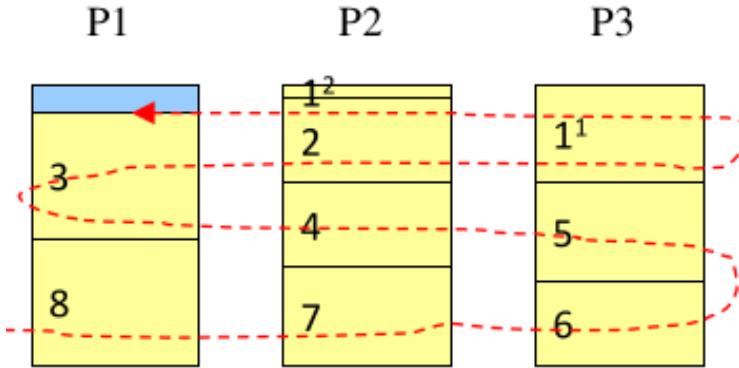


Figure 22.15: breadth-first partitioning algorithms, ??

There is also a solution to assigning tasks to queues with the heavy tasks first. Meaning that if the task is significantly larger then the others it will be assigned to a queue first. It is called pre-assigning the heavy tasks.

## 22.7 UPPAAL

- ! (shouting I want to do something)
  - ? (does someone want to do something) If we shout someone needs to answer.
  - E (some execution of our system)
  - A (one possible execution)
  - <> (some point in the execution)
- (all parts of the execution)

There will always be a deadlock if all tasks can reach a done state therefore to check if no deadlock occurs we need to check that there is no state where there is a deadlock and all tasks have reached their done state.

```
A[] not (deadlock and prod_done==false and cons_done==false and buff_done==false)
```

## 22.8 Real time communication (RTC)

Can bus is one of the most common forms of real time communication, since of the low cost and dependability. It is a shared broadcast meaning that you don't send a message to a specific node but broadcasts it and those who are interested will listen. The arbitration mechanism shown in Figure 22.16, allows the node with the highest priority to send first. The priority is dependent on the identifier. It is therefore important not to have the same identifier or the behavior will be unexpected.

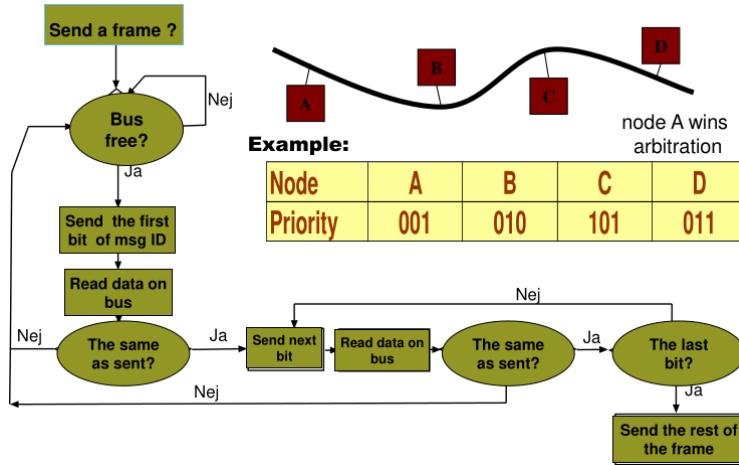


Figure 22.16: CAN Arbitration Mechanism, ??

SOF, Start Of Frame	Identifier	RTR, Re- move Transi- tion Re- quest	Control	Data	CRC, Cyclic Re- dund- ancy Check	CRC DEL, CRC De- lim- iter	ACK, Ac- knowl- edge	ACK DEL, Ac- knowl- edge De- lim- iter	EOF, End of Frame	IFS, Inter Frame Space
1 bit	11 bits	1 bit	6 bits	0-8 bytes	15 bits	1 bit	1 bit	1 bit	7 bits	3-, min 3 bits

Table 22.2: Note that the field priority/idenfitier

The maximum size is:

$$64\text{bits} + 47\text{bits} + 24\text{bits} = 135\text{bits}$$

if the message is sent 1Mbit/sec then the max transmition time for one message is 135 microseconds.

Task "comp" to produce a message at A:

$$X_{comp} = C_{comp} + \sum_{j \in hp(comp)} \left\lceil \frac{(X_{comp} + J_j)}{T_j} \right\rceil C_j$$

$$R_{comp} = X_{comp}^* + J_{comp}$$

Task "send" to transmit the message at A:

$$J_{send} = R_{comp} - C_{comp}$$

$$Y_{send} = C_{send} + B_{send} + \sum_{j \in hp(send)} \left\lceil \frac{(Y_{send} + J_j)}{T_j} \right\rceil C_j$$

$$R_{send} = Y_{send}^* + J_{send} = Y_{send}^* + R_{comp}$$

Task "dest" to consume the message at B:

$$\begin{aligned} J_{dest} &= R_{send} - C_{send} \\ Z_{dest} &= C_{dest} + \sum_{j \in hp(dest)} \left\lceil \frac{(Z_{dest} + J_j)}{T_j} \right\rceil C_j \\ R_{dest} &= Z_{dest}^* + J_{dest} = Z_{dest}^* + R_{send} \end{aligned}$$

$C_{send} = B_{send} = 135$  micro sec if message size is 135 bits

# Chapter 23

## Advanced Software Design

Resources

- 

### 23.1 Domain Model

A Domain Model is a static visual representations of the concepts from a specific domain, i.e. from the relevant system. If the domain is Spotify a domain element could be playlist and playSong.

### 23.2 Class Diagram

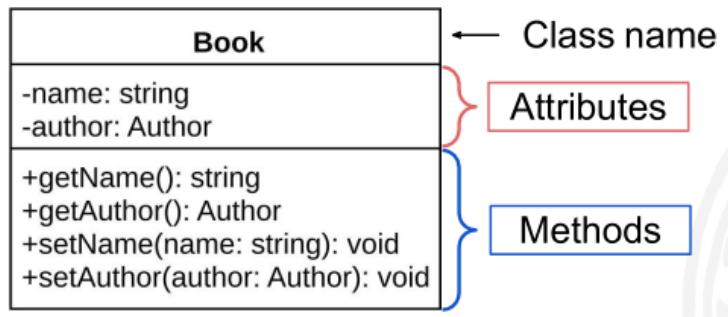


Figure 23.1: Class Structure. + stands for public and – for private. From ??

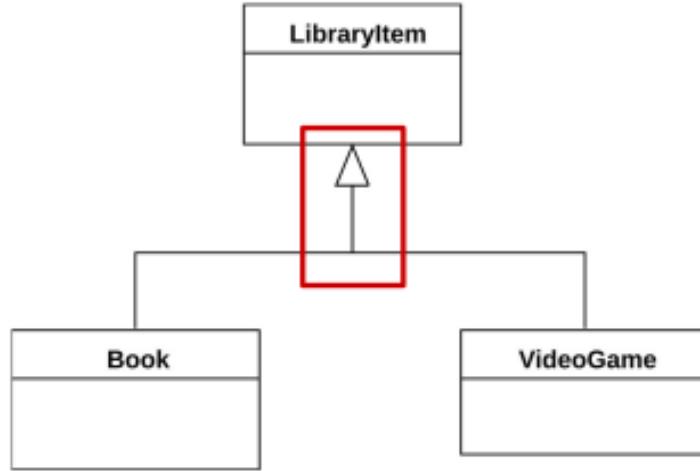


Figure 23.2: Class relationship represent “is a”, i.e. inheritance. From ??

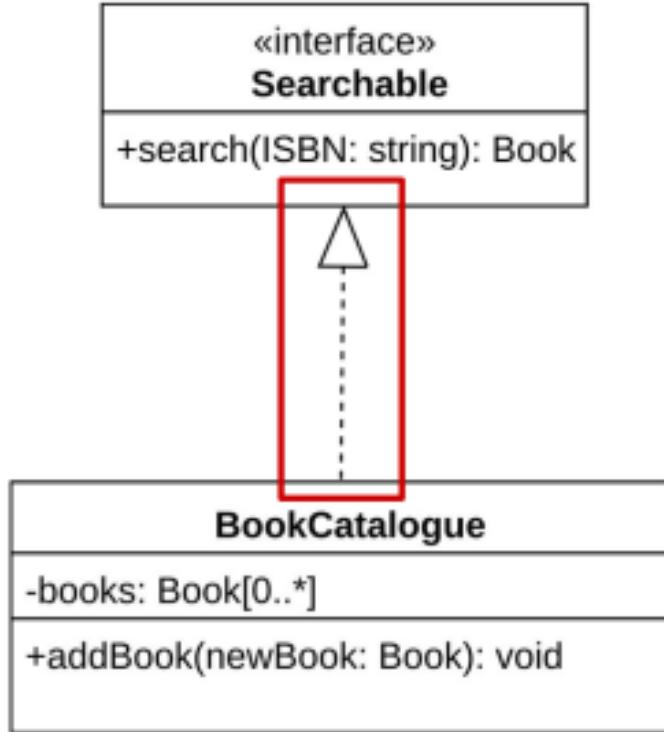


Figure 23.3: Class relationship represent “is a”, i.e. inheritance. From ??

#### Class diagram notations

- Basic Association: an arrow to a class from another class
- Multiplicities: 1, 1..\*, 0..\*. Describes one-to-one, many-to-many, or one-to-many relationships
- Attribute names and visibilities: The attribute used to identify the object?
- Aggregation: refers to “has a”.

- Composition: More strict than Aggregation.
- Dependency: Cannot exist unless another class exists.
- Qualifier (or: Key): The attribute used to identify the object.

Other

- Enumeration: defines a type.
- Constraints: defines a constraint on a class attribute.

## 23.3 GRASP

1. **Low coupling:** Not dependent on too many elements.
2. **High cohesion:** No element has too many responsibilities, i.e. not too many associations.
3. **Information expert:** The classes have the information to fulfil its responsibility.
4. **Pure fabrication:** We create classes just for the sake of increasing cohesive.
5. **Polymorphism:** Classes that allow for varying types.
6. **Indirection:** Create an object in between two classes, so that there are not directly coupled.
7. **Protected variations:** We design the architecture so that changes in objects or systems do not have a significant impact on the rest of the system. This is for example done with, data encapsulation, interfaces, polymorphism, standards, virtual machines, or operating systems.
8. **Creator:** Assign responsibilities to a class to create artifacts (objects or systems).
9. **Controller:**

## 23.4 Design patterns

### 23.4.1 Factory method

-Define an interface for creating an object -Subclasses decide which class to instantiate

### 23.4.2 Abstract factory

-Define an interface for creating families of related/dependent objects -Concrete classes are not specified

### 23.4.3 Builder

-Separate the construction of complex objects from its representation -The same process can create different representations

### 23.4.4 Object pool

### 23.4.5 The Observer



## Chapter 24

# Programming Embedded Systems

### 24.1 Intro, SDK

*Embedded system* is a system in which the computer (generally a microcontroller or microprocessor) is included as an integral part of the system. **Constraints**

- Timing
- Cost
- Weight
- Power
- Memory
- Computation power

*SDK* stands for software development kit and it provides an interface to abstract the specific hardware.

A clear and precise description of the problem that a function solves is called a *contract* for the function.  
<http://www.cs.ecu.edu/karl/2310/Javanotes/contract1.html>

*Frama-C* is an open-source extensible and collaborative platform dedicated to source-code analysis of C software. The Frama-C analyzers assist you in various source-code-related activities, from the navigation through unfamiliar projects up to the certification of critical software. <https://frama-c.com/>

*Zephyr* is a small real-time operating system (RTOS)[6] for connected, resource-constrained and embedded devices (with an emphasis on microcontrollers) supporting multiple architectures and released under the Apache License 2.0. [https://en.wikipedia.org/wiki/Zephyr\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Zephyr_(operating_system))

### 24.2 States Machines

*Bare metal* is a system with a very simple application. It doesn't have any specific timing requirements and is considered to be a single loop application. Purely interrupt-based system.

*RTOS* has multiple, independent execution threads. Usually needs more memory and it is more elaborate data exchange between tasks.

*State machines* A state machine shows the dynamic flow to states depending on values from previous states or user inputs.

When designing a state machines we need to first figure out the states of the system. Then what are the state change triggers. After that we want to define what happens when these triggers happens, what is the next state.

*State centric state machine* basically a big if-else statements that determine exactly what happens when we are in a state and how we get out. The main issue with state centric state machine is that it becomes difficult to maintain. It is also continuous computing, even if there is no update. And using a get event function does not allow for debouncing.

### 24.2.1 Debouncing

A button press is not a clean input. If it is not handled each button press will be registered as several button presses. There is a solution, however, *Debouncing*.

- Hardware debouncing: using a schmitt trigger.
- software debouncing: See figure ???. There is a delay between sensing an input, so if input comes each ms it will not count if the delay is 100ms.

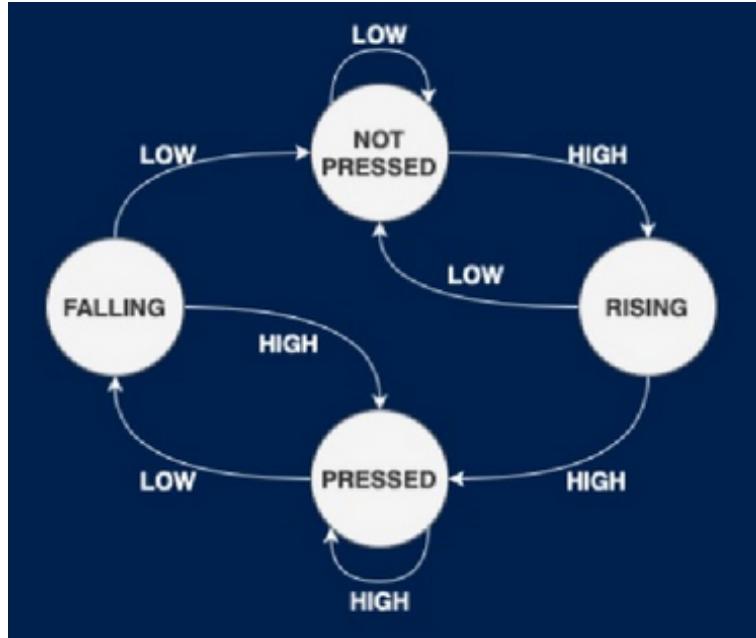


Figure 24.1: software debouncing

### 24.2.2 Hardware Interrupts

Examples of interrupts

- Clock interrupts
- Other internal interrupts: error like division by zero
- External interrupts: input from pins

All interrupts should be specified in the processor datasheet.

Interrupt service routine (ISR) is a software routine that the hardware invokes when there is a interrupt. These interrupts can be stored in global/static variables or in a thread safe queue to handle them when there is time.

*Deferred Interrupt Handling* is when the interrupt handler just record the event and try to do as little as possible so that the priority of the task, which will react on the event, is the one who decides when to run.

### 24.2.3 Table driven state machine

Instead of state machine with a bunch of switch cases of if statements, it is possible to create a table from the current state describes what happens when an event happens. This makes it easier to update and requires less code.

State	GO	STOP	TIMEOUT	None
●	●	●	●	●
●	●	●	●	●
●	●	●	●	●

Figure 24.2: Table driven state machine

## 24.3 Zephyr

The reason why we use realtime operating systems is that we want deterministic behavior and is particular important safety critical systems.

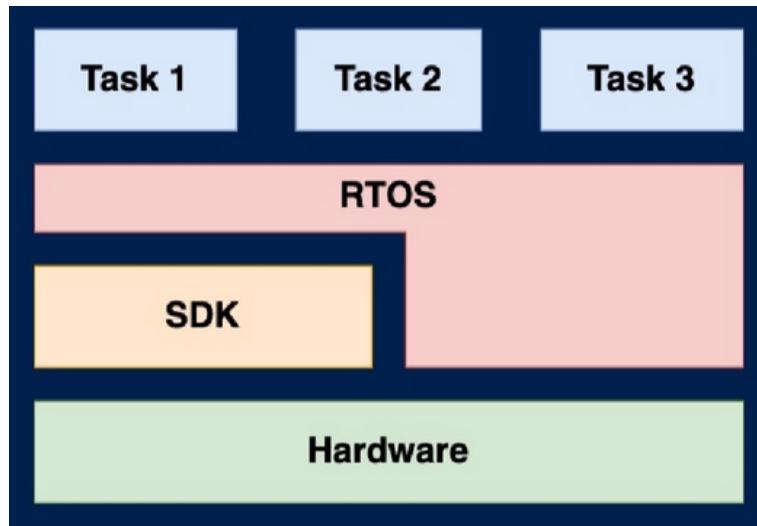


Figure 24.3: RTOS architecture

### 24.3.1 Zephyr threads

A thread is a kernel objects that is used when the application is to complex to be performed by an ISR. The number of threads that can be created is limited by RAM.

Spawning a thread statically:

```
K_THREAD_DEFINE(thread_id, stack_size,
```

```
entry_point_function, arg1, arg2, arg3,
priority, start_delay, execution_mode);
```

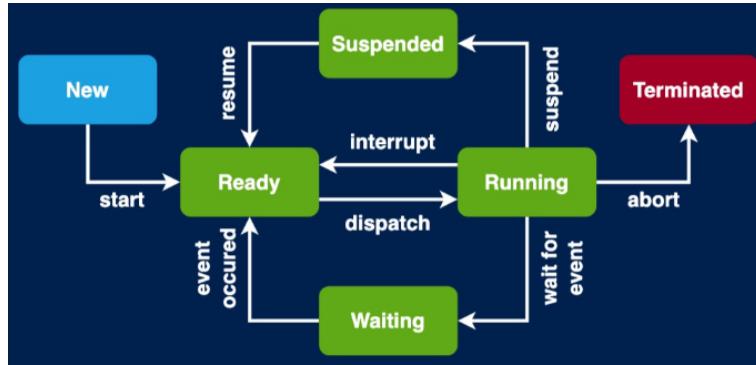


Figure 24.4: Thread states

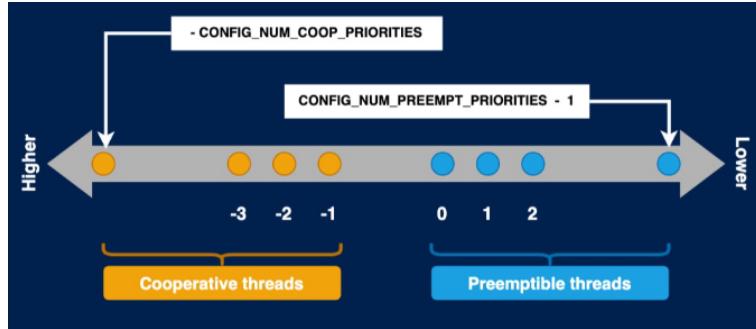


Figure 24.5: Thread priorities. Cooperative threads are non-preemptible, when the highest priority task has been dispatched then it won't be preempted even if a higher priority task is in the ready queue. Preempted threads have a lower priority and are allowed to be preempted.

### 24.3.2 Communication

Communication between threads can be either shared memory (not stack since each thread has its own stack) or message passing.

#### Shared memory and the use of locks

Global and/or static variables need to be carefully managed. However, this is not a problem if we use cooperative threads, otherwise we have to use locks. Using locks can result in priority inversion (a higher priority task is pre-empted by a lower priority task) or deadlock (also known as deadly embrace). Semaphores can also be used, they are more useful for sending signals. Semaphores are taken/given unlike mutex which are locked/unlocked.

#### Reentrant function

Function which executes correctly even when called by more than one task. It must follow certain rules:

- Must not use any global variables that can change.
- Must not call non-reentrant functions.
- Must not use hardware in a non-atomic way.

### Message passing

Message passing is more high-level way of communicating and is often considered. The implementation is dependent on the RTOS.

#### 24.3.3 Zephyr architecture

Code can be reused since the hardware is abstracted by having *arch*, *soc*, and *board*. If we want to create our own board, we can just add it in *boards*.

### Device tree

The OS should be able to run on different HW platforms, which is possible through the use of device tree. A device tree represent the hardware configuration int o special data structure. The HW layout is specified in a .dts file and can be extended with the use of .overlay file. The .overlay file writes over the .dts file to get a more specific layout of the HW. Linux compiles the .dts file into a binary representation used during boot-up. For zephyr it compiles into a header file.

### Other

ISR should be as minimal as possible, it should use semaphore to unlock tasks which do the actual work.

**Volatile**, whenever a variable is shared between two tasks, or a task and an ISR. C qualifier volatile: We tell the compiler that it cannot rely on previously read values. C compilers usually do some optimization however if it is volatile it can not do so.

## 24.4 Specification

Types of requirements (FURPS+)

- Functionality
  - Features, capabilities, security
- Usability
  - Human-computer interaction
  - Documentation
- Reliability
  - Frequency of failure
  - Error recovery
- Performance
  - Response time
- Supportability
  - Adaptability, Configurability
  - Maintenance
- Many others
  - Adaptability
  - User interface
  - Licencing

*Function contracts* describes the intended behavior as a function between the *caller* and the *callee*. It consists of a pair of a set of *preconditions* (assumptions) and a set of *postconditions* (guarantees).

A formal specification is not expressed in nature language, but instead is formally specified with logic.

*FOL properties* can be *satisfiable*, *valid*, *unsatisfiable*, *invalid*, and can be solved with *satisfiability modulo theories* (SMT) solver.

*Assertion* an alternative to contracts to test if a preconditions or postconditions holds true. However, there are some issues with assertions, they mix specification and implementation, and mixes responsibilities of the caller and the callee.

*Observers* is separate from the controller, it will check that the controller is behaving as expected.

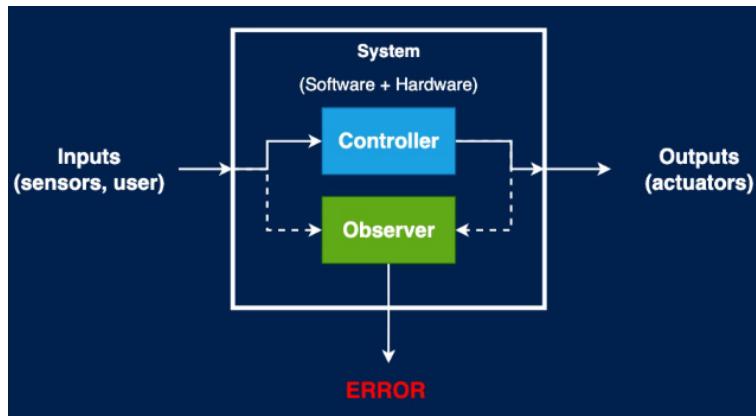


Figure 24.6: Observers

## 24.5 Memory Management

*Harvard architecture* a system that has separate memory and buses for code (ROM) and data (RAM).

### 24.5.1 Read only memory (ROM)

- Store code and constants
- Usually difficult to write
- Usually Flash memory
- Also exists PROM, UV-EPROM, EEPROM
- Pico Board has 2MB external Flash

### 24.5.2 Random access memory (RAM)

- Stack and heap
- Usually SRAM or DRAM
- Usually Flash memory
- Also exists PROM, UV-EPROM, EEPROM
- Pico RP2040 has 265kB on-chip SRAM divided into 6 separate modules to allow 6 parallel accesses.

### 24.5.3 Read - Modify - Write

Often we want to change just a single bit in a register. This require reading a word modify the word using bit operations then write it back, and this is not atomic.

#### Solution 1 - BIT-BANDING

0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---

Then if we want to access the second bit we have a alias for it to a non physical memory address.

#### Solution 2 - Bit Set and Clear Register

When set is 1 and the register is either 1 or 0 the result will be 1. When set is 0 the register wont change.

Reg	0	0	0	0	1	1	1	1
Set	0	1	1	0	1	0	0	0
Result:	0	1	1	0	1	1	1	1

When CLR is 1 and the register is either 1 or 0 the result is 0. When CLR is 0 the register wont change.

Reg	0	1	1	0	1	1	1	1
CLR	1	1	0	0	1	0	0	0
Result:	0	0	1	0	0	1	1	1

### 24.5.4 Manage memory

At compile time (statically)

- Compiler/linker creates different segments
  - Code (.text)
  - Read-only data (.rodata)
  - Read-write data (.data)
  - zero-initialized read-write data (.bss). It is more effective to have it in a separate segment than in read-write, since we don't have to assign the value to each variable.

There are also possible to do it at startup or at runtime (dynamic). At runtime (dynamic).

- Problems:
  - Possibly insufficient memory during runtime
  - Fragmentation
  - Implementation of malloc and free can be of substantial size

There are different modes:

- Processor mode
  - Thad mode, for executing applications
  - Handler mode, will automaticl enter this mode when there is exception handing and then will return to thred mode whenever it is done.
- Software execution modes
  - Unprivileged mode, limited access, e.g. cannot change process state.
  - Privileged mode, has access to all resources.

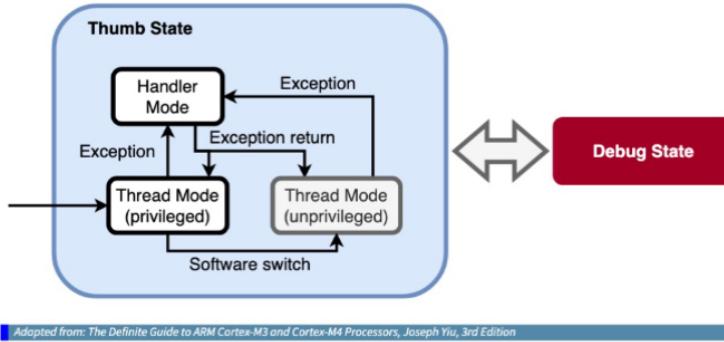


Figure 24.7: ARM Operation modes

*Memory protection unit* allows access rules to be set up for privileged access and user program access.

## 24.6 Debugging

From failure to fault

1. Verify the failure, determine correct behavior
2. Isolate and minimize (shrink)
3. Eyeball the code, where could the fault be?
4. Devise and run experiments to test your hypothesis
5. Repeat 3 & 4 until you understand what's wrong
6. Create a regression test

### 24.6.1 Problem minimization 1: inputs

Decrease the inputs.

- Generalization of greedy binary search
- Basic idea
  - Divide input into chunks (initially 2)
  - Remove a chunk, does the test still fail?
  - If yes, continue without it
  - If no, increase granularity (\*2)
  - Stop when culling away doesn't help anymore and number of chunks is the length of the input

### Delta Debugging algorithm

Use the delta debugging algorithm to find the input which gives the error. Whenever 1, 7, and 8 is in the input there is an error.

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

Initially divide it up into two chunks.

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

No fail increase granularity.

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Failed, remove the chunk which did not fail.

1	2	5	6	7	8
1	2	5	6	7	8
1	2	5	6	7	8

Failed, remove the chunk which did not fail.

1	2	7	8
1	2	7	8
1	2	7	8

No fail increase granularity.

1	2	7	8
1	2	7	8
1	2	7	8

Found that 1, 7, and 8 is causing the issue.

### 24.6.2 Problem minimisation 2: Slicing

Decrease the code.

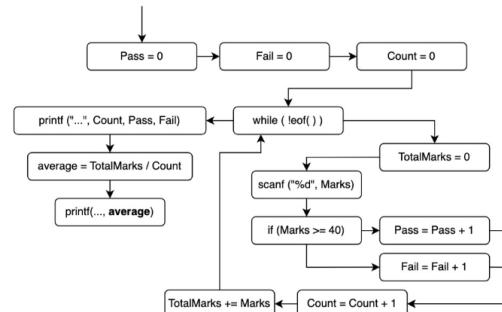
- Central concept of debugging
- Main idea
  - Give a Program P and some occurrence of variable x
  - Remove all statements that do not affect x
  - end up with simplified version of P' of P
  - P' only contains statements imports of value of x
- *Static backward slicing*: determine what causes the changes to the final value.

```

1 Pass = 0;
2 Fail = 0;
3 Count = 0;
4 while (!feof()) {
5     TotalMarks=0;
6     scanf("%d", Marks);
7     if (Marks >= 40) { Pass = Pass + 1; }
8     if (Marks < 40) { Fail = Fail + 1; }
9     Count = Count + 1;
10    TotalMarks = TotalMarks + Marks;
11 }
12 printf("Out of %d, %d passed and %d failed\n", Count, Pass, Fa
13 average = TotalMarks/Count;
14 printf("The average was %d\n", average);

```

(a) CFG example code



(b) CFG

Figure 24.8: Control Flow Graph (CFG)

- Simple logging

- printf

- serial output

- LEDs

- ...

- Logging frameworks

- log4c

- Zephyr logging

- ...

- Using a debugger

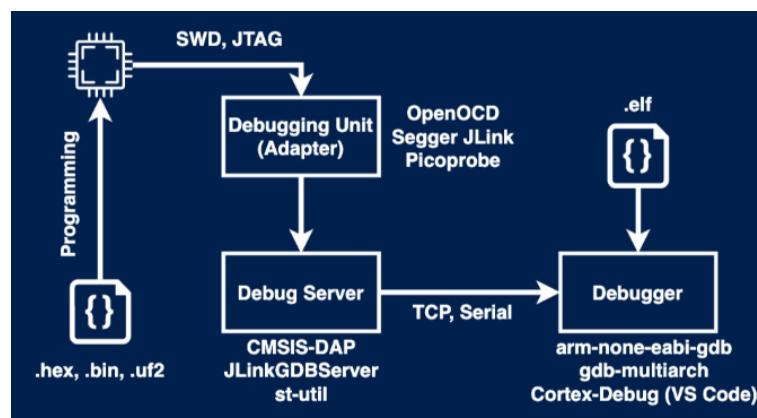


Figure 24.9: Debugging setup

## 24.7 Optimization

Algorithm	WC Runtime	AC Runtime
Bubble-sort	$O(n^2)$	$O(n^2)$
Insertion-sort	$O(n^2)$	$O(n^2)$
Quick-sort	$O(n^2)$	$O(n \log(n))$
Merge-sort	$O(n \log(n))$	$O(n \log(n))$
Tim-sort	$O(n \log(n))$	$O(n \log(n))$

Different data structures has different complexity for different operations, e.g. a array has a faster read time than a linked list.

*Space - Time - Tradeoff* is when we have to choose what is more important memory or performance. Since using little memory could worsen the performance and focusing on performance could worsen the memory usage. For example with a look-up table with frequently calculated values instead are precalculated values in a table. This cost more space but there is less computation involved.

Individual bits can be stored in different ways either.

- Packed: bits are grouped together
- Padded: reserve whole byte/word for each bit

For instance bit fields in c where we packed a number of bits in a struct.

*Alignment* is how we align data in memory. On *fully aligned* architectures, we need to have all variables aligned within a word. On *self aligned* architectures, addresses of n-byte types have to be multiples of n. The compiler is not allowed to re-order the fields. (a word is 2 bytes).

To know what we should optimize we use a *profiler* that measures the time spent in individual functions, blocks, ... We can also measure the performance, i.e. *instrument* the code. We can also experiment with removing or duplicate code sections.

Profilers:

- Instrumentation based
  - Profiler add statements to capture time at various code locations
  - Affects overall measured time (at times drastically)
- Sampling based
  - Profiler stops program periodically to record current program counter. So the current executed routine can be determined.
  - This is less precise
- Simulation based
  - Usually quite slow
  - Needs cycle-accurate simulator for the platform
  - often works with simplified assumptions about the hardware
- In-circuit/tracing
  - Needs hardware support
  - Often implemented by internally sampling

Some common thing that we can optimize to eliminate bottlenecks is:

- Choose a better suited algorithm or data structure
- Perform low-level optimizations

- Optimization is related to refactoring
  - We want to improve the design
  - Modifications should not impact functionality of the systems
- Use faster instructions: Using bit operators is faster instead of for instance multiplication.
- Use right arithmetic data-types: Using generalized int, float, double can have significant impact on computation, memory allocations depending on microC used. Use fixed sizes that are better suited for operation/purpose. For instance, uint8\_t instead of int when doing a for loop that has a few iterations (< 256).
- Loop optimizations: If we have a loop that does similar operation every iteration or that iterations might not be that big, we can ask the compiler to optimize the loop by unrolling the loop for instance (loop iterations has to be known at compile time). Unrolling a loop decreases conditional checks because we are executing sequentially instead of jumping back and doing condition checks for the loop.
- Sub-expression elimination: the process of eliminating a process that occurs at multiple instances. For instance, when an expression is calculated at multiple places, it is maybe better to just calculate it once, save in memory and use it everywhere it is needed instead of recalculating the expression.
- Inlining: the process of replacing a subroutine or function call at the call site with the body of the subroutine or function being called. This eliminates call-linkage overhead and can expose significant optimization opportunities.
- Algebraic simplifications: Simplifying your calculations to maybe do less calculations and still get the same/similar/close-enough result.

## 24.8 Testing

Testing in form of *dynamic* verifications means that we verify by running the code. *Static* verification means that we inspect the code without running it.

- Unit testing: test the low-level unit design with individual software units such as functions, classes, tasks, ...
- Integration testing: test the architectural design with a focus on integration/interaction between different modules/units.
- System testing: testing the system level specification.
- Acceptance testing: test the software with respect to user requirements.
- Orthogonal: Regression testing.
- Test set: a collection of test cases for a particular unit
- Test suite: is a collection of test sets

*Oracles* is a mechanism for determining whether a test has passed or failed.

Construction of test suites

- Black box (Closed box) testing: The test are derived from the external description without any knowledge of the implementation.
- White box (Glass box) testing: Make test from the source code to test for instance all branches, conditions, statements, ...

Code *coverage* refers to how much of the code has been covered when all tests have run.

- Input domain modelling: All possible inputs.

- Input space partitioning: Partition input domain into regions.

Structural coverage

Common notions in CFGS

- Execution path: Path through CFG that start at entry point.
- Path condition: the condition for which the path is taken.
- Feasible execution path: If a path condition can be satisfied.
- *Statement coverage*: with all tests every node in the CFG is executed at least one.
- *Branch coverage*: with all tests every edge in the CFG is taken at least once.
- *Path coverage*: with all tests every possible path in the CFG is executed at least once
- *Decision coverage*: with at least one test where d evaluates to true and one where d evaluates to false.
- *Condition Coverage*: for each condition c in program p evaluates as least once to true and once to false.

#### Modified condition decision coverage (MC/DC)

Let X be the condition ( $A \&& (B \mid C)$ )

Test	A	B	C	X
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

All pairs where A changes values and the output is different but B and C is the same.

Test = {1, 5}, {2, 6}, {3, 7}

All pairs where B changes values and the output is different but A and C is the same.

Test = {2, 4}

All pairs where C changes values and the output is different but A and B is the same.

Test = {3, 4}

To have tests that cover at least one pair of each boolean variable:

Test = {2, 3, 4, 6}

*Regression testing*: test which runs when new features or when we refactor code to prevent the introduction of new bugs.

## 24.9 Verification

*Program verifier* takes a program specification and program code and verifies it, i.e. proves the programs correctness. If it finds a counter examples it proves that the code is incorrect. It can also be inconclusive were no conclusion can be drawn.

*Abstract interpretation* techniques based on fixed-point computation. Widely used by compilers. Use specification to create a mathematical structure for the prof.

*Model checking* a technics based on (systematic) state-space exploration.

*Heuristic Bug finders* focus on implicit specifications.

### 24.9.1 Transition systems

*transition systems* is a concept used to study the computation. A program can be converted into a number of possible state the variables can be in. A state space is noted as  $S$  and the initial state  $I$  were  $I \subseteq S$ , and transitions  $\rightarrow \subseteq S \times S$ .

The goal is to identify if there is a set of  $Err \subseteq S$  error states. The system is safe if there is no path  $s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_n$  with  $s_0 \in I$  and  $s_n \in Err$ .

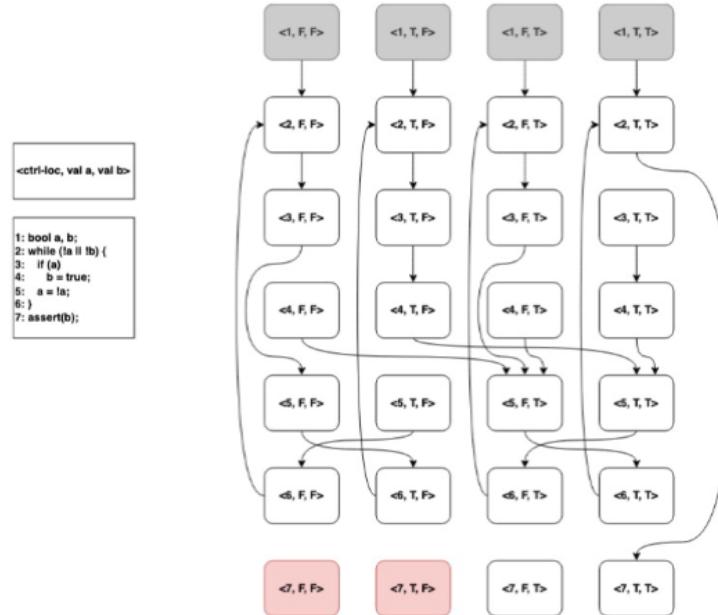


Figure 24.10: Transition Systems

*Explicit state model checking*: explicitly construct graph  $(S, I, \rightarrow)$  and check reachability of error states. There are a few languages that does support it like java with *Java Path Finder*.

### 24.9.2 Deductive Verification

Deductive software verification aims at formally verifying that all possible behaviors of a given program satisfy formally defined, possibly complex properties, where the verification process is based on some form of logical inference, i.e., “deduction”.

*Hoare logic* has the goal to provide a formal system for reasoning about program correctness.

## Hoare logic example

```
/*
PRE: a > 0
POST: ret > 0
*/
int f(int a) {
    /* { a > -1 } => is implied by PRE */
    int x = a;
    /* { x > -1 } */
    x = 2 * x;
    /* { x > -2 } */
    x = x + 2;
    /* { x > 0 } */
    return x;
}
```

Start at the end and work your way up.