

# Opracowanie pytań do obrony (EZI)

**KN EZI Team**

21 stycznia 2016

# Spis treści

|          |   |           |
|----------|---|-----------|
| <b>I</b> | <b>Zagadnienia specjalnościowe</b>                        | <b>5</b>  |
| <b>1</b> | <b>Sterowniki mikroprocesorowe i ich zastosowania</b>     | <b>6</b>  |
| 0.1      | Sterownik PLC . . . . .                                   | 6         |
| 0.2      | Regulator PID . . . . .                                   | 8         |
| 0.3      | Sterownik CNC . . . . .                                   | 9         |
| 0.4      | Kontroler PAC . . . . .                                   | 10        |
| <b>2</b> | <b>Lokalne sieci komputerowe</b>                          | <b>11</b> |
| <b>3</b> | <b>Bazy danych i ich zastosowania</b>                     | <b>12</b> |
| 1        | Bazy relacyjne . . . . .                                  | 12        |
| 2        | Z wikipedii (takie powtórzenie) . . . . .                 | 13        |
| 2.1      | Architektura Oracle . . . . .                             | 14        |
| <b>4</b> | <b>Przetwarzanie obrazów, metody i zastosowania</b>       | <b>16</b> |
| 1        | Algorytmy przetwarzania obrazów . . . . .                 | 16        |
| 1.1      | Operacje punktowe . . . . .                               | 17        |
| 1.2      | Histogramy . . . . .                                      | 17        |
| 1.3      | Operacje algebraiczne . . . . .                           | 17        |
| 1.4      | Binaryzacja . . . . .                                     | 17        |
| 1.5      | Filtracje . . . . .                                       | 17        |
| 1.6      | Decymacja i interpolacja . . . . .                        | 17        |
|          | . . . . .   | 18        |
| 1.7      | Wykrywanie krawędzi . . . . .                             | 18        |
|          | . . . . .   | 18        |
| 2        | Inne zastosowania . . . . .                               | 18        |
| <b>5</b> | <b>Miary i oceny dokładności algorytmów przybliżonych</b> | <b>19</b> |
| <b>6</b> | <b>Systemy operacyjne komputerów</b>                      | <b>21</b> |
| <b>7</b> | <b>Zadania optymalizacji i techniki ich rozwiązywania</b> | <b>25</b> |
| 0.1      | Sformułowanie zadania optymalizacji . . . . .             | 25        |
| 0.2      | Zadanie programowania liniowego PL . . . . .              | 28        |
| 0.3      | Zadanie programowania nieliniowego PN . . . . .           | 31        |
| 0.4      | Optymalizacja globalna . . . . .                          | 36        |
| 0.5      | Sformułowanie zadania optymalizacji . . . . .             | 37        |
| 0.6      | Rodzaje zadań . . . . .                                   | 37        |
| <b>8</b> | <b>Systemy dynamiczne, opisy własności</b>                | <b>41</b> |
| 1        | Podział układów . . . . .                                 | 41        |
| 2        | Transmitancja operatorowa układu . . . . .                | 42        |
| 3        | Charakterystyki . . . . .                                 | 42        |
| 3.1      | Czasowa . . . . .   | 42        |
| 3.2      | Skokowa . . . . .   | 42        |

|           |  |           |
|-----------|--|-----------|
| 3.3       | Impulsowa . . . . .  | 42        |
| 3.4       | Charakterystyka amplitudo-fazowa . . . . .   | 42        |
| 4         | Człony dynamiczne . . . . .  | 43        |
| 5         | Stabilność . . . . .   | 43        |
| 5.1       | Twierdzenia i Definicje . . . . .  | 43        |
| 5.2       | Kryteria stabilności . . . . .   | 43        |
| <b>9</b>  | <b>Programowanie w systemie operacyjnym Unix</b>   | <b>45</b> |
| 1         | Interpreter komend Bourne shell, podstawowe mechanizmy, pisanie skryptów, rozszerzenia POSIX . . . . .   | 45        |
| 1.1       | Programowanie w języku powłoki . . . . .   | 45        |
| 1.2       | Podstawowe mechanizmy . . . . .  | 46        |
| 1.3       | Wyrażenia regularne: sed, awk, grep . . . . .  | 47        |
| 1.4       | funkcje I/O niskiego poziomu, deskryptory, przegląd bibliotek systemu UNIX . . . .   | 47        |
| 1.5       | Programowanie procesów: tworzenie, atrybuty, dziedziczenie atrybutów, współdzielenie zasobów. Sygnały, obsługa sygnałów. Komunikacja przez potoki. . . . . | 48        |
| 1.6       | Komunikacja międzyprocesowa przez gniazda, kolejki komunikatów, i pamięć wspólną, semaforey . . . . .  | 48        |
| 1.7       | Programowanie w C . . . . .  | 49        |
| <b>10</b> | <b>Komputer, architektura i oprogramowanie</b>   | <b>51</b> |
| 0.1       | Architektury . . . . .   | 51        |
| 0.2       | Oprogramowanie . . . . .   | 52        |
| <b>II</b> | <b>Zagadnienia kierunkowe</b>  | <b>53</b> |
| <b>11</b> | <b>Programowanie strukturalne i obiektowe</b>  | <b>54</b> |
| <b>12</b> | <b>Fala elektromagnetyczna: typy, parametry, właściwości</b>   | <b>55</b> |
| <b>13</b> | <b>Tranzystory bipolarne i unipolarne: budowa, właściwości i zastosowania</b>  | <b>58</b> |
| 0.1       | Tranzystory bipolarne . . . . .  | 58        |
| 0.2       | Tranzystory unipolarne (polowe) . . . . .  | 60        |
| <b>14</b> | <b>Systemy ciągłe i dyskretne: klasyfikacja, opis</b>  | <b>66</b> |
| 0.1       | Sygnały . . . . .  | 67        |
| 0.2       | Dodatek . . . . .  | 67        |
| <b>15</b> | <b>Zmienna losowa: właściwości, opis</b>   | <b>70</b> |
| 0.1       | Zmienne losowe . . . . .   | 70        |
| 0.2       | Dystrybuanta i jej własności . . . . .   | 70        |
| 0.3       | Funkcja gęstości . . . . .   | 71        |
| 0.4       | Podstawowe parametry rozkładu zmiennej losowej . . . . .   | 71        |
| <b>16</b> | <b>Ciągła, dyskretna i szybka transformata Fouriera, widmo sygnału</b>   | <b>73</b> |
| 1         | Ciągła transformata Fouriera . . . . .   | 73        |
| 2         | Dyskretna transformata Fouriera (DFT) . . . . .  | 73        |
| 2.1       | Przeciek DFT . . . . .   | 74        |
| 3         | Szybka transformata Fouriera (FFT) . . . . .   | 75        |
| 4         | Widmo sygnału . . . . .  | 76        |
| <b>17</b> | <b>Modulacje analogowe i cyfrowe</b>   | <b>77</b> |
| 0.1       | Modulacje analogowe . . . . .  | 77        |
| 0.2       | Modulacje cyfrowe . . . . .  | 78        |
| <b>18</b> | <b>Wzmacniacze operacyjne: właściwości i zastosowania</b>  | <b>81</b> |

|  |           |
|--|-----------|
| <b>19 Mikroprocesory: budowa, zastosowania</b>               | <b>82</b> |
| 0.1 Architektury . . . . .                                   | 82        |
| 0.2 Budowa . . . . .   | 83        |
| 0.3 Mikroprocesor a mikrokontroler . . . . .                 | 83        |
| 0.4 Zastosowania . . . . .                                   | 83        |
| <b>20 Sieci komputerowe: budowa, protokoły, zastosowanie</b> | <b>85</b> |
| 0.1 Budowa . . . . .   | 85        |
| 0.2 Protokoły . . . . .                                      | 87        |
| 0.3 Zastosowania . . . . .                                   | 89        |

## Część I

# Zagadnienia specjalnościowe

# Rozdział 1

## Sterowniki mikroprocesorowe i ich zastosowania

Główną częścią takiego sterownika jest mikrokontroler pełniący funkcję jednostki wykonującej operacje logiczne i arytmetyczne.

### 0.1 Sterownik PLC

**Programowalny sterownik logiczny PLC** (ang. Programmable Logic Controller) uniwersalne urządzenie mikroprocesorowe przeznaczone do sterowania pracą maszyny lub urządzenia technologicznego. Sterownik PLC musi zostać dopasowany do określonego obiektu sterowania poprzez wprowadzenie do jego pamięci żadanego algorytmu działania obiektu - zaprogramowanie go. Cechą charakterystyczną sterowników PLC odróżniającą ten sterownik od innych sterowników komputerowych jest cykliczny obieg pamięci programu. Algorytm jest zapisywany w dedykowanym sterownikowi języku programowania. Istnieje możliwość zmiany algorytmu przez zmianę zawartości pamięci programu. Sterownik wyposaża się w odpowiednią liczbę układów wejściowych zbierających informacje o stanie obiektu i żądaniach obsługi oraz odpowiednią liczbę i rodzaj układów wyjściowych połączonych z elementami wykonawczymi, sygnalizacyjnymi lub transmisji danych. Układy I/O mogą być cyfrowe lub analogowe.

Zasada działania: przeglądanie wejść i w zależności od ich stanu oraz wprowadzonego programu użytkownika ustawianie stanu wyjść dwustanowych (zał/wył). PLC jest w swej istocie komputerem przemysłowym ze specjalnie zaprojektowaną jednostką centralną oraz rozbudowanymi układami wejść/wyjść do kontaktu z zewnętrznymi urządzeniami przemysłowymi (urządzeniami polowymi). Wcześniej działanie sterowników były realizowane przy pomocy rozbudowanej sieci elektrycznej przy zastosowaniu przekaźników.

#### Historia

Inżynierowie z Hydramatic Division firmy GM w roku 1968 sformułowali założenia dla sterownika programowalnego, którego główną zaletą miała być redukcja wysokich kosztów związanych z niewielką elastycznością systemów przekaźnikowych. W 1969 r. został opracowany pierwszy sterownik PLC.

#### Założenia sterownika

- realizacja na urządzeniach półprzewodnikowych,
- elastyczność komputera,
- praca w środowisku przemysłowym (wibracje, wysoka temperatura, zapylenie itp.),
- możliwość przeprogramowywania,
- łatwe programowanie i obsługa przez elektryków i techników.

#### Budowa

- jednostka centralna (CPU),

- zasilacz,
- pamięć,
- układy I/O,
- port komunikacyjny (np. PROFIBUS),
- gniazdo rozszerzenia.

### Języki programowania

- **drabinkowy LD** (ang. Ladder Diagram) - logika drabinkowa, a schemat podobny do klasycznego rysunku technicznego elektrycznego,
- **blokowy FBD** (ang. Function Block Diagram) - diagram bloków funkcyjnych, sekwencji linii zawierająca te bloki,
- **strukturalny ST** (ang. Structured Text) - tekst strukturalny - język zbliżony do Pascala;
- **lista instrukcji IL** (ang. Instruction List) lista instrukcji - rodzaj assemblera,
- **sekwencyjny SFC** (ang. Sequential Function Chart) sekwencyjny ciąg bloków - sekwencja bloków programowych z warunkami przejścia.

### Poziomy sygnałów cyfrowych

- 5, 12, 24, 48 VDC,
- 24, 128, 240 VAC.

### Poziomy sygnałów analogowych

- 0 ... 20 mA,
- 4 ... 20 mA (najczęściej),
- 0 ... 5, 10 V,
- -10 ... 10 V.

Źródłami sygnałów analogowych są: termometry, przepływomierze, wilgotnościomierze, mierniki nacisku, potencjometry.

Analogowe sygnały wejściowe sterują: zaworami analogowymi, szybkością obrotów silników, rejestratorami.

### Zadania PLC

- skanowanie wejść,
- wykonywanie instrukcji,
- ustawianie wyjść,
- samodiagnostyka.

## 0.2 Regulator PID

**Regulator PID** (ang. Proportional-Integral-Derivative controller) – regulator stosowany w układach regulacji składający się z trzech członów: proporcjonalnego, całkującego i różniczkującego. Najczęściej jego celem jest utrzymanie wartości wyjściowej na określonym poziomie, zwanym wartością zadaną. Jest najczęściej stosowanym regulatorem w przemyśle.

Właściwości dynamiczne idealnych regulatorów PID mogą być opisane za pomocą transmitancji operatorowej, będącej stosunkiem transformaty Laplace’a sygnału wyjściowego regulatora  $U'(s)$  i transformaty Laplace’a uchybu regulacji (sygnału wejściowego)  $E(s)$ :

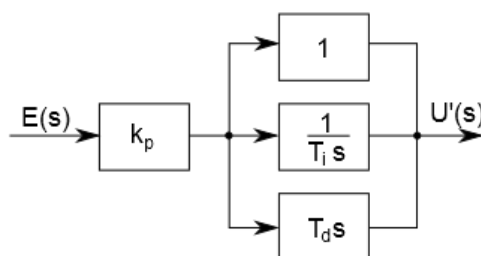
$$G_r(s) = \frac{U'(s)}{E(s)} = k_p \left( 1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right), \quad (1.1)$$

gdzie:

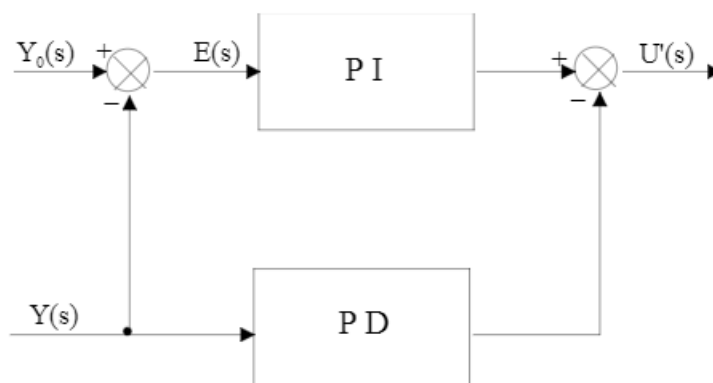
$k_p$  - wzmacnienie członu proporcjonalnego,

$T_i$  - czas całkowania (zdwojenia),

$T_d$  - czas różniczkowania (wyprzedzenia).



Rysunek 1.1: Schemat blokowy idealnego regulatora PID



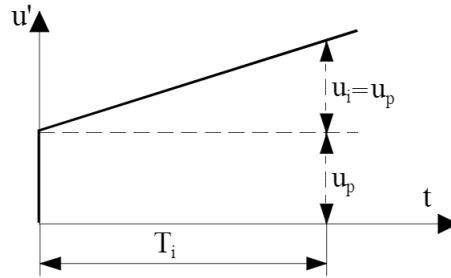
Rysunek 1.2: Schemat blokowy mikroprocesorowego regulatora PID

**Zakres proporcjonalności**  $X = (1/k_p) \cdot 100\%$  jest to procentowa część pełnego zakresu zmian wielkości wejściowej  $e$ , potrzebna do wywołania pełnej zmiany wielkości wyjściowej  $u'$  regulatora.

**Czas zdwojenia (całkowania)**  $T_i$  dotyczy regulatorów PI, których wielkość wyjściowa ma zarówno składową proporcjonalną  $u_p$ , jak i całkującą  $u_i$ . Czas zdwojenia określa się na podstawie znajomości charakterystyki czasowej skokowej regulatora. Jest to czas potrzebny na to, aby sygnał wyjściowy regulatora, będący wynikiem działania całkującego, stał się równy sygnałowi będącemu wynikiem działania proporcjonalnego (rysunek 1.3.).

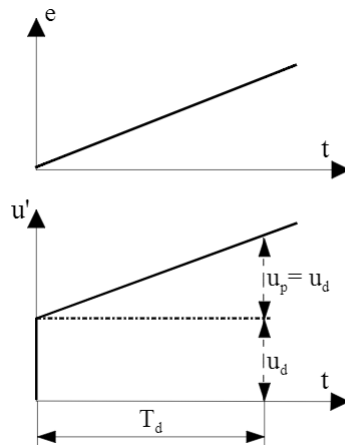
Zatem sygnał wyjściowy z regulatora PI (wypadkowy dla obu oddziaływań) po czasie  $T_i$  zwiększa dwukrotnie swoją wartość. Stąd pochodzi jego nazwa – czas zdwojenia.





Rysunek 1.3: Sposób wyznaczania czasu zdwojenia  $T_i$

**Czas wyprzedzenia (różniczkowania)**  $T_d$  dotyczy regulatorów PD, których wielkość wyjściowa ma zarówno składową proporcjonalną, jak i różniczkującą. Czas wyprzedzenia określa się na podstawie odpowiedzi regulatora na wymuszenie w postaci narastającego liniowo sygnału uchybu regulacji  $e$ . Jest to czas, po którym sygnał wyjściowy regulatora, związany działaniem proporcjonalnym  $u_p$ , zrówna się z sygnałem pochodzącym od działania różniczkującego  $u_d$  (rysunek 1.4.).



Rysunek 1.4: Sposób wyznaczania czasu wyprzedzenia  $T_d$

Wielkości  $k_p$  (w starszych typach regulatorów X),  $T_i$  i  $T_d$  noszą nazwę nastaw regulatora. Są to parametry nastawialne, które dobieramy tak, aby:

- uzyskać odpowiednie własności dynamiczne regulatora,
- uzyskać najlepszą jakość regulacji.

Transmitancja określona wzorem (1.1) dotyczy regulatora idealnego, którego w praktyce nie daje się zrealizować. W warunkach rzeczywistych w członie różniczkującym występuje inercja, określona stałą czasową  $T$ . Stała ta nie jest nastawialna. Zatem zamiast członu  $T_d \cdot s$  w transmitancji występuje człon  $T_d s / (Ts + 1)$ . Transmitancja operatorowa rzeczywistego regulatora PID ma wówczas postać:

$$G_r(s) = \frac{U'(s)}{E(s)} = k_p \left( 1 + \frac{1}{T_i \cdot s} + \frac{T_d \cdot s}{Ts + 1} \right), \quad (1.2)$$

### 0.3 Sterownik CNC

**Sterownik CNC** (ang. Computer Numerical Control) to typ sterownika mikroprocesorowego, który programuje się za pomocą tzw. G code'u. Sterowniki te używane są m.in. do kontroli takich urządzeń jak: frezarki, tokarki a w szerszym zastosowaniu do sterowania robotami fabrycznymi, które pracują w tzw. trybie taśmy montażowej np. automaty składające podzespoły samochodowe.

W nowoczesnych maszynach CNC stosuje się sterowniki pracujące na bazie komputera przemysłowego IPC w technologii „PC-based Automation”. W tej technologii sterownik CNC działa programowo,

a nie sprzętowo, tak jak to odbywało się w starego typu dedykowanych sterownikach. System operacyjny czasu rzeczywistego sterownika, realizuje funkcje PLC, HMI i sterowania ruchem, odpowiadając za funkcjonalność całej maszyny.

## 0.4 Kontroler PAC

**Sterowniki PAC** (ang. Programmable Automation Controller) zwane sterownikami automatyki są dużymi systemami sterownikowymi przeznaczonymi do obsługi złożonych procesów technologicznych. Wiele dotychczasowych sterowników PLC o największej wydajności zostało nazwanych przez producentów sterownikami PAC. Granica pomiędzy dużymi i wydajnymi sterownikami PLC a sterownikami PAC jest obecnie słabo widoczna. Współczesne procesy sterowania wykorzystują ogromne ilości sygnałów i danych, począwszy od analogowych i cyfrowych urządzeń wejścia/wyjścia, poprzez szybkie kamery wysokiej rozdzielczości, a kończąc na wieloosiowych sterownikach ruchu. Takie zastosowania, jak: szybka produkcja, monitoring pracy maszyn w czasie rzeczywistym, sterowanie precyzyjne czy sterowanie złożonym procesem, wymagają deterministycznych systemów akwizycji, zaawansowanej analizy i algorytmów przetwarzania danych. Wyższej klasy sterowniki PLC w pewnym stopniu spełniają te wymagania. Jednakże do wydajnego przetwarzania tych danych trzeba zastosować odpowiednie środki informatyczne, m.in. procesory zmiennoprzecinkowe i duże zasoby pamięci. Sterowniki PAC integrują tego rodzaju gotowe składniki sprzętowe w ramach systemu czasu rzeczywistego, oferując w ten sposób wydajną platformę dla inżynierów automatyki.

## Rozdział 2

# Lokalne sieci komputerowe

**Sieci lokalne LAN** (ang. Local Area Network), są sieciami prywatnymi obejmującymi pojedynczy budynek lub grupę budynków w obszarze o średnicy do kilku kilometrów. Sieci te są powszechnie używane do łączenia komputerów osobistych i stacji roboczych w biurach firmy i fabrykach w celu udostępnienia zasobów (np. drukarek) i wymiany informacji.

Sieci LAN od innych typów odróżniają trzy cechy:

- rozmiary,
- technologie transmisji,
- topologia.

Sieci LAN mają ograniczone rozmiary, co oznacza, że czas przesyłu w najgorszym przypadku jest ograniczony i z góry znany. Znajomość tych granic umożliwia użycie pewnych rozwiązań, które w innych przypadkach byłyby niemożliwe do zastosowania, a jednocześnie upraszcza zarządzanie siecią.

Sieci lokalne mogą korzystać z technologii transmisji opartej na kablu, do którego podłączone są wszystkie komputery. Tradycyjne sieci LAN charakteryzują się szybkością przesyłu od 10 Mb/s do 100 Mb oraz niskim opóźnieniem. Nowsze sieci lokalne działają z szybkością do 10 Gb/s.

W rozgłoszeniowych sieciach LAN możliwe są różnorodne topologie. Przykładem jest sieć magistralowa (zbudowana z kabla liniowego). W sieci magistralowej, w każdej chwili najwyżej jedno urządzenie jest nadrzędne i ma prawo do nadawania. Potrzebny jest zatem mechanizm arbitrażu który rozstrzyga konflikt gdy dwa lub więcej urządzeń chce nadawać jednocześnie. Mechanizm arbitrażu może być scentralizowany lub rozproszony. Na przykład sieć Ethernet jest siecią rozgłoszeniową opartą na magistrali ze zdecentralizowanym sterowaniem. Komputery w tej sieci mogą nadawać w dowolnej chwili; w razie kolizji dwóch lub więcej pakietów każdy komputer oczekuje losowy odstęp czasu i ponawia próbę. Drugim typem systemu rozgłoszeniowego jest pierścień. W pierścieniu każdy bit propaguje się samodzielnie bez czekania na resztę pakietu, do którego należy. Zazwyczaj każdy bit okrąży cały pierścień w czasie potrzebnym na transmisję kilku bitów, często zanim komplety pakiet zostanie wysłany.

# Rozdział 3

## Bazy danych i ich zastosowania

**Baza danych** – zbiór danych zapisanych zgodnie z określonymi regułami. W węższym znaczeniu jest to program (system) służący do zbierania, przetwarzania i organizowania informacji. Bazy danych pozwalają przechowywać dowolne informacje, na przykład informacje o ludziach, produktach czy zamówieniach. Dla systemów komputerowych mogą to być dane numeryczne, tekstowe, binarne, daty, lub inne dostępne typy w danym systemie. Można więc przechowywać grafikę, muzykę – np. jako pliki binarne. Kartoteka w bibliotece to również baza danych. Za bazę danych uważany może być plik tekstowy, arkusz kalkulacyjny, plik binarny. Gdy ilość danych rośnie, powstają nadmiarowe i niespójne dane. Dlatego wprowadza się różne systemy bazodanowe umożliwiające łatwiejsze zarządzanie dużą ilością danych – np. MS SQL, MySQL, Access.

Z uwagi na cenę danych współczesne systemy bazodanowe stanowią podstawą działalności wielu firm, co wymaga istnienia możliwie niezawodnych systemów baz danych.

**System zarządzania bazą danych** – oprogramowanie umożliwiające tworzenie oraz eksploatację bazy danych oraz jej użytkowników.

**baza danych** = dane + schemat bazy danych (np. relacyjny)

**system bazy danych** = baza danych + system zarządzania bazą

### Kryteria klasyfikacji baz danych

- wielkość
- liczba odwołań
- stopień ważności informacji
- struktura informacji
- implementacja komputerowa

Bazy danych można podzielić według struktur organizacji danych, których używają:

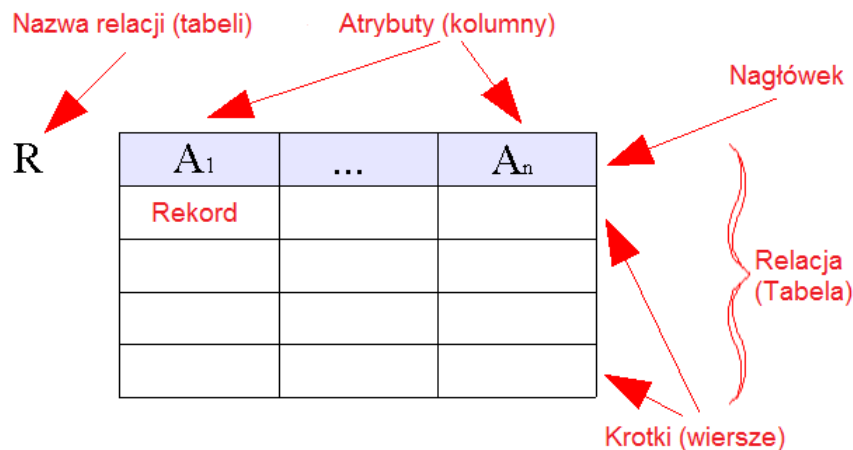
- Bazy proste: kartotekowe, hierarchiczne (struktura drzewa, np. katalogi w systemie)
- Bazy złożone: relacyjne, obiektowe, relacyjno-obiektowe, strumieniowe, temporalne, nierelacyjne (NoSQL), grafowe

Z wymienionych struktur, w praktyce zdecydowanie najczęściej używane są bazy relacyjne.

## 1 Bazy relacyjne

Relacyjny model danych został zaproponowany w latach 70. Zaimplementowany w latach 80.

Opiera się on (suprise) na pojęciu relacji. Dane grupowane są w relacje reprezentowane przez tablice. Czyli tablica STUDENCI zawierająca wiersze (krotki) z danymi studentów (imię, indeks – kolumny), o odpowiednich typach – całość tworzy relację. W modelu obiektowym relacji odpowiada obiekt.



Rysunek 3.1: Relacja

## Pojęcia

- klucz podstawowy (główny) - najkrótszy zbiór atrybutów (czyli kolumn) pozwalający na **jednoznaczną** identyfikację wiersza w tabeli. Max 1 na tabelę
- klucz obcy - kolumna zawierająca wartości wskazujące na *klucz główny* w innej tabeli, pozwala stworzyć związek pomiędzy tabelami. Typy związków: jeden do jednego, jeden do wielu, wiele do wielu (najczęściej realizowane za pomocą tabeli pośredniej)

Związki pomiędzy relacjami pozwalają uniknąć nadmiarowości danych i pozwalają zapewnić ich spójność. Jeżeli np. mamy tabelę studenci i samochody, relacja wiele do jednego (każdy samochód posiada klucz obcy czyli np. indeks studenta). Jeden student może posiadać więc wiele samochodów. Jeżeli chcemy sprawdzić czyj jest dany samochód to mamy klucz obcy studenta, a nie jesgo wszystkie dane - czyli wystarczy raz wpisać studenta, a zmiana jego danych następuje też w jednym miejscu. Czyli minimalna ilość danych zapewnia spójność. Zły przypadek - tabela samochody zawiera wszystkie dane studenta, on zmienia imię, trzeba aktualizować dane we wszystkich jego samochodach (i pewnie o którymś się zapomni).

Zasady projektowania schematów relacyjnych baz danych opisują **zasady normalne**, najważniejsze:

- 1NF atrybuty są atomowe (niepodzielne) - kolumna [imię,nazwisko] - należy rozdzielić na 2 osobne kolumny, jedna komórka zawiera jedną wartość, czyli nie chcemy kolumny imiona
- 2NF każda kolumna niekluczowa zależy funkcyjnie od całego klucza głównego (i tak nikt o to nie zapyta ani nikt nie wykuje, ale wstawiam)
- 3NF każda komórka w wierszu zależy jedynie od klucza głównego, [indeks], [imię], [płeć]  
Zakładając że indeks to klucz główny, a płeć jest jednak związana z imieniem to ta ka tabela nie spełni tego wymagania, należy wydzielić drugą tabelę imiona: [imię], [płeć] a studentowi zostawić indeks i klucz obcy do tabeli imiona

Do pracy z relacyjnymi bazami danych służy SQL - standardowy język zapytań, używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych. Konkretnie: selekcje, projekcje, złączanie, usuwanie, modyfikację dodawania danych, a także operacje na schematach bazy danych - tworzenie, modyfikowanie tabel.

Zawiera on typy danych, umożliwia tworzenie funkcji, procedur składowanych.

## 2 Z wikipedii (takie powtórzenie)

Relacyjne bazy danych (jak również przeznaczony dla nich standard SQL) oparte są na kilku prostych zasadach:

1. Wszystkie wartości danych oparte są na prostych typach danych.

2. Wszystkie dane w bazie relacyjnej przedstawiane są w formie dwuwymiarowych tabel (w matematycznym żargonie noszących nazwę „relacji”). Każda tabela zawiera zero lub więcej wierszy (w tymże żargonie – „krotki”) i jedną lub więcej kolumn („atrybutów”). Na każdy wiersz składają się jednakowo ułożone kolumny wypełnione wartościami, które z kolei w każdym wierszu mogą być inne.
3. Po wprowadzeniu danych do bazy, możliwe jest porównywanie wartości z różnych kolumn, zazwyczaj również z różnych tabel, i scalanie wierszy, gdy pochodzące z nich wartości są zgodne. Umożliwia to wiązanie danych i wykonywanie stosunkowo złożonych operacji w granicach całej bazy danych.
4. Wszystkie operacje wykonywane są w oparciu o algebrę relacji, bez względu na położenie wiersza tabeli. Nie można więc zapytać o wiersze, gdzie ( $x=3$ ) bez wiersza pierwszego, trzeciego i piątego. Wiersze w relacyjnej bazie danych przechowywane są w porządku zupełnie dowolnym – nie musi on odzwierciedlać ani kolejności ich wprowadzania, ani kolejności ich przechowywania.
5. Z braku możliwości identyfikacji wiersza przez jego pozycję pojawia się potrzeba obecności jednej lub więcej kolumn niepowtarzalnych w granicach całej tabeli, pozwalających odnaleźć konkretny wiersz. Kolumny te określa się jako „klucz podstawowy” (ang. primary key) tabeli.

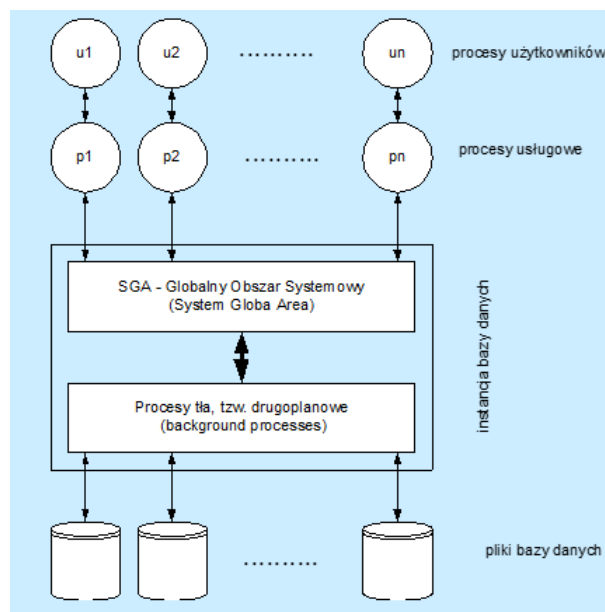
### Zastosowania

- systemy bankowe (bankomat)
- systemy masowej obsługi (hipermarket)
- rezerwacja biletów lotniczych
- telefonia komórkowa (sms)
- Dziekanat Wydziału Elektroniki
- toto-lotek
- policja (ewidencja przestępców, rejestr samochodów)
- rejestry sądowe, księgi wieczyste
- ankiety internetowe
- sklepy internetowe
- gry internetowe
- system audiotele
- biblioteka PWr.

## 2.1 Architektura Oracle

### Rodzaje plików

- pliki konfiguracyjne (init files) — najważniejszy z nich to plik parametrów (tzw. parameter file, w skrócie pfile) o nazwie init<SID>.ora, gdzie <SID> to unikalny (najczęściej 4-literowy) identyfikator instancji na danej maszynie, plik jest tekstowy i zawiera informacje o ustawieniach przy starcie bazy, a także o nazwie i położeniu plików kontrolnych
- pliki kontrolne (control files) — binarne, bardzo ważne, zawierają informacje o położeniu WSZYSTKICH plików bazy danych wraz z datą i godziną ich zamknięcia
- pliki danych (data files) — przechowują dane w postaci binarnej, z danych w nich zawartych korzysta się za pośrednictwem procesów serwera (tła)
- pliki dziennika powtórzeń (redo-log files) - rejestrują wszystkie operacje wykonane na bazie (na wypadek awarii), mogą być aktywne lub zarchiwizowane



Rysunek 3.2: Architektura systemu

- pliki śladu (trace files) — zawierają informacje o uszkodzeniach i o błędach procesów tła oraz procesów użytkowników
- pliki kodu (code files) — wszelkie kody źródłowe, skrypty SQL itp

**SGA** (ang. System Global Area). Składa się z:

- bufora danych, który: Zawiera dane załadowane z dysku, okresowo zapisane (Dirty blocks), przechowuje informacje o zmianach, ma strukturę cykliczną.
- bufora dziennika powtórzeń, który przechowuje informację o zmianach, ma strukturę cykliczną.
- obszaru współdzielonego

### Procesy tła

- DBWR — database writer — zapisuje DIRTY BLOCKS na dysk
- LGWR — log writer — zapisuje zmiany do loga
- CKPT — checkpoint — sygnalizuje tzw. punkt kontrolny + aktualizuje nagłówki plików kontrolnych
- SMON — system monitor — odtwarza instancję, scala wolne obszary na dysku
- PMON — process monitor — zwalnia zasoby procesu użytkownika, który się „zawiesił”
- ARCH — archiwizator — opcjonalny, spowalnia pracę, umożliwia skonfigurowanie serwera lustrzanego

### Stany bazy danych

- IDLE - nieczynna, pliki zamknięte, procesy tła nie działają
- NOMOUNT - stan po odczytaniu pfile-a, zainicjowaniu SGA i uruchomieniu procesów tła, stan służący do tworzenia nowej bazy danych
- MOUNT - stan po odczytaniu plików kontrolnych i otwarciu połączenia z plikami danych i log-ami
- OPEN - stan otwarcia bazy, dane są dostępne dla użytkowników (w wyjątkowych sytuacjach stan można uruchomić w tzw. trybie RESTRICTED, wtedy dostęp do bazy mają tylko administratorzy)

## Rozdział 4

# Przetwarzanie obrazów, metody i zastosowania

**Reprezentacja obrazu** Obraz cyfrowy powstaje poprzez kwantyzację i próbkowanie obrazu analogowego. Wynikiem jest uporządkowana siatka kwadratów, z których każdy zawiera informację o obrazie wejściowym jak np. średnia jasność w obszarze. Każdy z kwadratów siatki zwany jest pikselem, a ich liczba i wielkość są podstawowymi wartościami opisującymi matryce służące do akwizycji obrazu jak CCD lub CMOS.

Rozróżniamy obrazy jednobarwne - *monochromatyczne*, binarne, barwne.

Cyfrowy zapis obrazu polega na zdefiniowaniu przestrzeni barw w jakiej zapisujemy obraz (RGB, CMYK, inne), głębi obrazu - czyli bitów na kanał co określa ilość możliwych do przedstawienia barw. Przykładowo:

typowa kamera internetowa - ramka obrazu składa się z trzech kanałów - obrazów monochromatycznych, dla których zakres wartości przypisywanych pikselom należy do przedziału 0-255. Kanały reprezentują kolory z palety barw RGB, obraz końcowy powstaje poprzez złączenie wszystkich kanałów. Zapewnia to 24-bitową głębię kolorów (3 kanały po 8 bitów), tzn. pozwala reprezentować  $2^{24}$  kolorów. Tryb ten nazywany jest *true color*. 4 kanałem często używanym np. w formacie png może być kanał *Alpha* odpowiadający za (nie)-przezroczystość.

## 1 Algorytmy przetwarzania obrazów

Pojęcie **przetwarzania obrazów** można rozumieć dwójako. W wąskim sensie oznacza operacje, które przetwarzają jedne obrazy w inne - zarówno dane wejściowe jak i wyjściowe mają formę obrazu. W tym sensie zawierają się:

- akwizycja obrazów - wprowadzenie obrazów, zwłaszcza cyfrowych do systemów technicznych (sorki, brzydka definicja),
- reprezentacja obrazu i modelowanie - dobór reprezentacji obrazu w systemie tech., by odpowiadała ona charakterowi obrazu i przeznaczeniu reprezentacji, również modele matematyczne obrazów. Głębia, przestrzeń barw
- polepszanie obrazów - wytwarzanie obrazów subiektywnie ocenianych przez człowieka jako lepsze
- odzyskiwanie, restauracja obrazów - usuwanie zniekształceń (geometrycznych), zakłóceń (szumy, rozmycia)
- kompresja obrazów - bezstratna jak i stratna
- znakowanie wodne - umieszczanie i odczytywanie dodatkowych, często niewidocznych informacji
- prezentacja, wyświetlanie obrazów

Z tego punktu widzenia grafika komputerowa, animacje komputerowe stanowią oddzielną grupę metod.

W szerszym sensie do przetwarzania obrazów zalicza się **analizę obrazów**, w której dane wejściowe odpowiadają obrazom, natomiast dane wyjściowe mają inny charakter. Z analizą obrazów związane jest **rozpoznawanie wzorców** którego celem jest identyfikacja pewnych struktur w obrazach.



Trzecią kategorię metod przetwarzania stanowi **analiza obrazów**:

- analiza ogólnych cech obrazu lub jego fragmentu - np. analiza cech ilościowych wartości pikseli, barwy punktów
- ekstrakcja cech - np. krawędzi, struktur geometrycznych
- segmentacja - podział obrazu na spójne obszary o podobnych cechach (ruch, kolor)
- analiza tekstur - wydzielenie np. drewna, tkaniny i analiza cech
- analiza sceny i zdarzeń - określanie zależności przestrzennych i czasowych pomiędzy obiektami, analiza ruchu obiektów, zdarzeń itp.

## 1.1 Operacje punktowe

Należą do I grupy metod (obraz w obraz). Przekształcają każdy z pikseli wejściowych  $u(n_1, n_2)$  w piksel  $y(n_1, n_2)$  według:

$$y(n_1, n_2) = F[u(n_1, n_2)]$$

Gdzie  $n_1, n_2$ - indeks poszczególnego piksela, np. współrzędne x,y, trzeci parametr ( $n_3$ ) może być kanałem. Przykład - dodanie +70 do wartości każdego piksela przyciemni obraz kolorowy jako całość, przy obrazie binarnym zamiana 0 na 1 i odwrotnie - negatyw obrazu. Wszystkie piksele obliczane są niezależnie, co pozwala zrównoleglać te algorytmy. Przykłady:

- dodanie stałej C  $y(n_1, n_2) = u(n_1, n_2) + C$
- inwersja obrazu  $y(n_1, n_2) = 255 - u(n_1, n_2)$  (dla 8-bitowego obrazu)

Operacje punktowe pozwalają wykorzystywać tablicowanie, (*look-up-table* - LUT), zamiast wyliczać każdy piksel przechowujemy w pamięci zbiór wszystkich wartości, np. 40 zamieniamy na 70, 41 na 71... dzięki czemu wystarczy sprawdzić i wstawić wartość zamiast liczyć.

Wszystkie operacje punktowe można wykonywać z nasyceniem lub bez, dodanie 100 do 200 w 8 bitowej głębi (max 255) może dać czarny (255) przy nasyceniu (operacja nieliniowa!), lub 45 (czyli jasny) przy „zawinięciu” się wartości.

## 1.2 Histogramy

Wyliczenie histogramu pozwala „zobaczyć” ile pikseli jakiego koloru występuje w obrazie, np. dominują piksele czerwone nad zielonymi, albo jasne nad ciemnymi. Pozwala ocenić kontrast - równomierny histogram oznacza wiele detali w obrazie (dużo różnych pikseli), w innym przypadku obraz jest np. niedoświetlony. Histogram pozwala również opisać operacje punktowe. Dodanie stałej to nic innego jak przesunięcie histogramu.

## 1.3 Operacje algebraiczne

Operacje punktowe działające na wielu obrazach, przykładowo mając kilka identycznych obrazów, z różnym szumem można je dodać i uśrednić eliminując szum.

## 1.4 Binaryzacja

- progowanie, zmiana obrazu na binarnym, według jakiegoś progu, np.  $u < 100 \geq 0, u \geq 100 \geq 1$ .

## 1.5 Filtracje

Algorytmy usuwające zakłócenia, szumy. Wartość piksela obliczana jest na podstawie jego i otoczenia - o jego wielkości i kształcie decyduje maska, np. prostokątna o wymiarach 5x5. Przykładowe filtry:

- **uśredniający** - najprostszy, liczy średnią wartość pikseli pod maską, rozmywa obraz (ale usuwa szumy)
- **Gaussa** - rozmywający, ale według rozkładu normalnego. Czyli średnia ważona

- **bilateralny** - Gaussa z uwzględnieniem różnicy kolorów (jak zupełnie inne piksele to nie uśrednia), usuwa szum, zachowuje krawędzie więc bardzo przydatne przed np. wykrywaniem krawędzi
- **medianowy** - filtr nieliniowy, wartość piksela równa jest medianie pikseli pod maską, suwa zakłócenia impulsowe z obrazu (*pieprz i sól*), losowe bardzo jasne/ciemne piksele na obrazie

## 1.6 Decymacja i interpolacja

Zmniejszenie/zwiększenie rozdzielczości obrazu. Interpolację można opisać jako wstawienie zer w miejsca nowych próbek, a następnie wygładzenie obrazu filtrem dolnoprzepustowym.

## 1.7 Wykrywanie krawędzi

Rodzina algorytmów krawędziujących, np. na podstawie gradientu czyli skokowych zmian wartości pikseli. Przydatne przy algorytmach identyfikujących, pozwalają łatwiej ocenić kształt.

## 2 Inne zastosowania

Wszelkie algorytmy identyfikujące, twarz, rękę, samochód. Działają na podstawie wykrycia kształtu i/lub koloru. Okrągłe i cieliste - raczej głowa, kolorowe i kwadratowe - pewnie samochód. (sorki, chyba za późno to piszę).

## Rozdział 5

# Miary i oceny dokładności algorytmów przybliżonych

**Algorytmy aproksymacyjne** – algorytmy służące do znajdowania przybliżonych rozwiązań problemów optymalizacyjnych. Stosuje się je zwykle do rozwiązywania problemów, dla których nie są znane szybkie algorytmy znajdujące rozwiązanie dokładne, na przykład dla problemów NP-zupełnych.

Istotą algorytmu aproksymacyjnego, tym co odróżnia go od algorytmu heurystycznego, jest związana z każdym takim algorytmem informacja o koszcie zwracanego rozwiązania w stosunku do rozwiązania optymalnego. Mianowicie koszt rozwiązania zwróconego przez algorytm aproksymacyjny jest nie większy (w przypadku problemu minimalizacyjnego) albo nie mniejszy (w przypadku problemu maksymalizacyjnego) od rozwiązania optymalnego pomnożonego przez pewną stałą.

### Oszacowania jakości algorytmów aproksymacyjnych

Żałómy, że mamy do czynienia z problemem optymalizacyjnym, w którym każde potencjalne rozwiązanie ma dodatni koszt i, że chcemy znaleźć rozwiązanie prawie optymalne. Zależnie od problemu rozwiązanie optymalne może być zdefiniowane jako to o maksymalnym możliwym koszcie lub o minimalnym; zadanie może polegać na maksymalizacji albo na minimalizacji.

Mówimy, że algorytm aproksymacyjny dla danego problemu ma ograniczenie względne  $p(n)$ , jeśli dla dowolnych danych wejściowych rozmiaru  $n$  koszt  $C$  rozwiązania konstruowanego przez ów algorytm szacuję się z dokładnością do czynnika  $p(n)$  przez koszt  $C'$  rozwiązania optymalnego:

$$\max\left(\frac{C}{C'}, \frac{C'}{C}\right) \leq p(n)$$

Definicja ta stosuje się zarówno do problemów minimalizacji, jak i maksymalizacji. Dla problemu maksymalizacji  $0 < C \leq C'$ , a współczynnik  $C'/C$  określa ile razy koszt rozwiązania optymalnego jest większy od kosztu rozwiązania przybliżonego. Podobnie dla problemu minimalizacji  $0 < C' \leq C$ , a współczynnik  $C/C'$  określa ile razy koszt rozwiązania przybliżonego jest większy od kosztu rozwiązania optymalnego. Ponieważ zakładamy, że wszystkie rozwiązania mają dodatni koszt, współczynniki te są zawsze dobrze określone. Ograniczenie względne algorytmu aproksymacyjnego nigdy nie jest mniejsze niż 1, ponieważ nierówność  $C/C' < 1$  implikuje  $C'/C > 1$ . Ograniczeniem względnym algorytmu optymalnego jest 1, a algorytm aproksymacyjny o dużym ograniczeniu względnym może dać rozwiązanie znacznie gorsze niż optymalne.

Czasami wygodniej operować pojęciem błędu względnego. Dla dowolnych danych wejściowych błąd względny definiuję się jako:

$$\frac{|C - C'|}{C'}$$

gdzie, jak poprzednio  $C'$  jest kosztem rozwiązania optymalnego, a  $C$  to koszt rozwiązania uzyskanego za pomocą algorytmu aproksymacyjnego. Błąd względny jest zawsze nieujemny. Dla algorytmu aproksymacyjnego  $\epsilon(n)$  jest ograniczeniem błędu względnego, jeśli:

$$\frac{|C - C'|}{C'} \leq \epsilon(n)$$

Z powyższych definicji wynika, że ograniczenie błędu względnego można oszacować przez funkcję ograniczenia względnego

$$\epsilon(n) \leq p(n) - 1$$

(Dla problemu minimalizacyjnego jest to równość, podczas gdy dla problemu maksymalizacyjnego mamy  $\epsilon(n) = (p(n) - 1)/p(n)$ ; spełniona jest więc nierówność, ponieważ  $p(n) \geq 1$ ).

Dla wielu problemów skonstruowano algorytmy aproksymacyjne o stałym, niezależnym od  $n$  ograniczeniu względnym. W takich przypadkach będziemy pisać  $p$  lub  $\epsilon$  wskazując w ten sposób na niezależność od  $n$ .

Dla innych problemów informatykom nie udało się zaprojektować żadnego wielomianowego algorytmu aproksymacyjnego o stałym ograniczeniu względnym. Wszystko co można wówczas robić to potraktować ograniczenie względne jako funkcję rosnącą wraz z rozmiarem danych wejściowych. Przykładem takiego problemu jest problem pokrycia zbioru.

Dla niektórych problemów NP- zupełnych istnieją algorytmy aproksymacyjne za pomocą których można uzyskiwać coraz mniejsze ograniczenie względne (lub, równoważnie, malejący błąd względny) kosztem dłuższego czasu obliczeń. Inaczej mówiąc, zachodzi odwrotnie proporcjonalna współzależność między czasem obliczeń a jakością aproksymacji. Przykładem może być problem sumy podzbioru. Sytuacja jest na tyle ważna, że zasługuje na własną nazwę.

**Schemat aproksymacyjny** dla problemu optymalizacyjnego to algorytm aproksymacyjny, otrzymujący na wejściu nie tylko komplet danych opisujących problem, ale także wartość  $\epsilon > 0$  taką, że dla każdego ustalonego  $\epsilon$  schemat ten jest algorytmem aproksymacyjnym o ograniczeniu błędu względnego  $\epsilon$ . Mówimy, że schemat aproksymacyjny jest wielomianowym schematem aproksymacji, jeśli dla dowolnego ustalonego  $\epsilon > 0$  działa on w czasie wielomianowym ze względu na rozmiar  $n$  jego danych wejściowych.

Czas działania wielomianowego schematu aproksymacji nie powinien też rosnąć zbyt gwałtownie, kiedy zmniejsza się  $\epsilon$ . Najlepiej, gdyby zmniejszenie  $\epsilon$  o stały czynnik nie powodowało wzrostu czasu obliczeń potrzebnych do osiągnięcia dostatecznego przybliżenia o więcej niż stały czynnik. Innymi słowy, chcielibyśmy aby czas obliczeń było wielomianem ze względu na  $n$ , ale również, ze względu na  $1/\epsilon$ .

Mówimy, że schemat aproksymacji jest w pełni wielomianowym schematem aproksymacji, jeśli czas jego działania jest wielomianem zarówno ze względu na  $1/\epsilon$  jak i rozmiar  $n$  danych wejściowych, gdzie  $\epsilon$  jest ograniczeniem błędu względnego schematu. Czas działania schematu mógłby na przykład wynosić  $(1/\epsilon)^2 \cdot n^3$ . Dla takiego schematu dowolne zmniejszenie o stały czynnik ograniczenia błędu względnego dają się osiągnąć za pomocą odpowiedniego zwiększenia o stały czynnik czasu obliczeń.

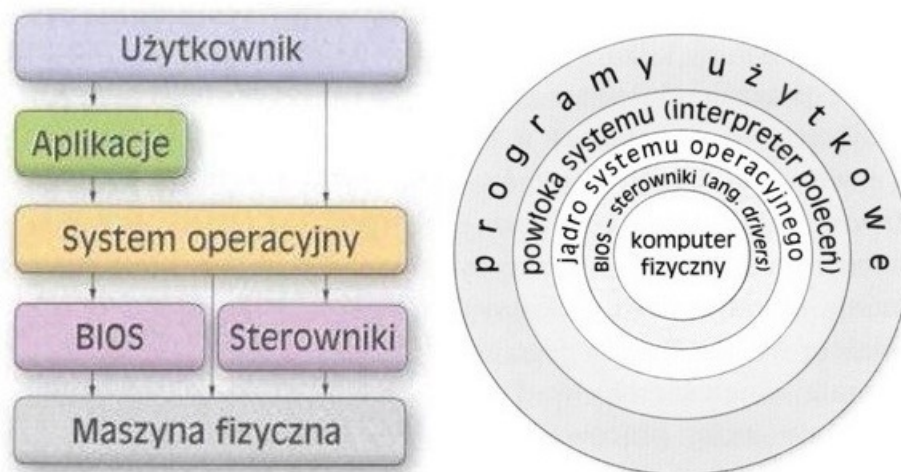
## Rozdział 6

# Systemy operacyjne komputerów

**System operacyjny** (ang. Operating System) jest środowiskiem programów tworzących główną platformę programową umożliwiającą działanie zainstalowanego w systemie oprogramowania. OS nadzoruje pracę wszystkich uruchomionych procesów oraz urządzeń komputera. Pomimo iż swą pracę wykonuje przeważnie w tle i sam w sobie nie tworzy z komputera w pełni funkcjonalnego narzędzia jednak bez niego komputer jest kompletnie bezużyteczny. System operacyjny zainstalowany na dysku twardym komputera decyduje jakie oprogramowanie może zostać uruchomione pod jego kontrolą, wpływa na bezpieczeństwo danych, realizuje połączenia do sieci nadzorując podrzędne systemy pracujące na innych komputerach, określa kompatybilność wobec innych systemów, decydując o funkcjonalności stabilności pracy. Niestety nie istnieje system idealny, każdy ma swoje zalety i wady.

### Budowa i zadania poszczególnych warstw OS

Podczas pracy OS komunikuje się zazwyczaj z elektroniką komputera poprzez BIOS i sterowniki choć jest możliwa również jego bezpośrednia komunikacja ze sprzętem co zostało pokazane na rysunku 6.1. System operacyjny można przedstawić w postaci modelu warstwowego.



Rysunek 6.1: Warstwowy model systemu

### Główne zadania stawiane przed systemami operacyjnymi to:

- zarządzanie zasobami komputera - systemy operacyjne starują oraz optymalizują wykorzystanie określonych urządzeń, które wchodzi w skład zestawu komputerowego; specjalne moduły zwane sterownikami, które składają się na system operacyjny udostępniają aplikacjom spójne metody programowania urządzeń (interfejsy), co gwarantuje współdziałanie każdego nowego urządzenia z oprogramowaniem (jeżeli producent dostarczy właściwy sterownik)

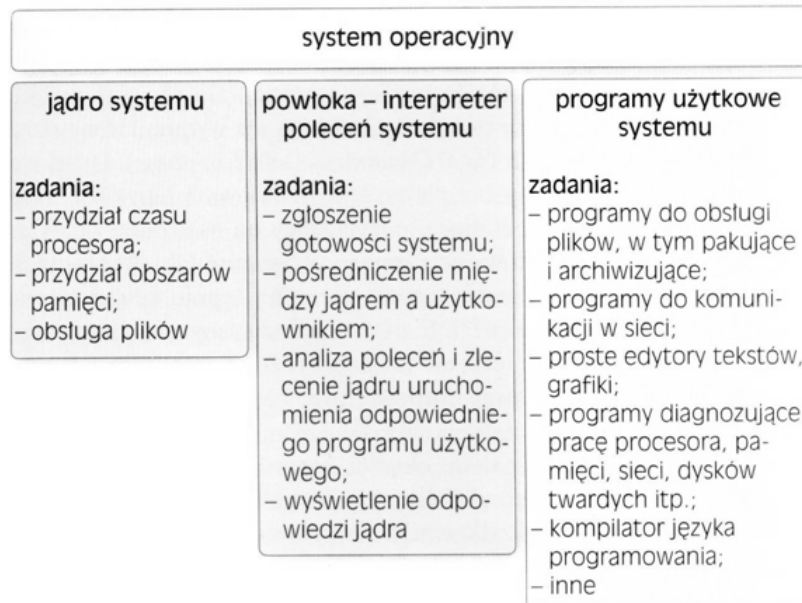
- gromadzenie oraz zarządzanie danymi - systemy operacyjne wyposażone są w moduły obsługujące system plików, czyli struktury umieszczone na dyskach, pomagające w logiczny sposób uporządkować dane, grupując je w katalogi i pliki
- maszyny wirtualne - systemy operacyjne udostępniają aplikacjom tzw. maszyny wirtualne, czyli uproszczone obrazy maszyn, na których pracują aplikacje; system udostępnia aplikacjom szczegóły dna tematu komputera oraz rozszerzenia ułatwiające pracę (np. zasoby udostępniane poprzez sieć aplikacje widzą tak, jakby znajdowały się one na dysku lokalnym; aplikacja, która korzysta z takiego zasobu nie obsługuje pracy sieciowej, więc aby umożliwić jej dostęp do zasobu, system operacyjny symuluje, jest zasób ten jest lokalny udostępniając go dla aplikacji)
- wielozadaniowość - na pojedynczym komputerze funkcjonować może wiele aplikacji w jednym czasie; każda aplikacja otrzymuje swoją maszynę wirtualną i może pracować jakby była pojedynczym programem działającym na maszynie; dzięki takiej właściwości systemów operacyjnych nie ma potrzeby przystosowywania aplikacji, by mogła dzielić się daną maszyną z innymi aplikacjami
- interakcja z użytkownikiem - rolę tę spełnia najbardziej zewnętrzna warstwa systemu operacyjnego zwana powłoką (ang. shell), która pozwala użytkownikowi uruchamiać aplikacje; w systemach graficznych do powłoki zaliczają się także typowe elementy interfejsu, z których korzysta aplikacja, jak kontrolki czy okna dialogowe
- komunikowanie się z innymi komputerami - jest to jeden z najistotniejszych elementów systemów operacyjnych; dzięki modułom obsługi sieci można uzyskać dostęp do Internetu, do dysków komputerów stojących w sąsiednim pomieszczeniu czy sieciowych urządzeń peryferyjnych jak drukarki czy skanery

**Najważniejszymi cechami, które decydują o użyteczności danego systemu operacyjnego są:**

- prostota instalacji oraz użytkowania systemu
- współpraca z innymi systemami, czyli możliwość odczytu i zapisu danych na partycjach z innych systemów a także współpraca oraz wymiana danych między komputerami w sieciach lokalnych i Internecie
- zgodność sprzętowa (instalację na konkretnej maszynie utrudnia czasami brak właściwych sterowników dla określonych urządzeń)
- wymiana danych, czyli możliwość przeglądania oraz wymiany dokumentów pomiędzy różnymi aplikacjami pracującymi pod kontrolą różnych systemów
- możliwość pracy sieciowej (wygoda podczas przeglądania zasobów sieciowych, wymiany protokołów, itp.)
- cena
- liczba aplikacji, które działają w danym systemie (nawet najlepiej pracujący system będzie niemal bezużyteczny, jeżeli oferta oprogramowania, które współpracuje z tym systemem będzie niewielka)
- lokalizacja (możliwość komunikacji użytkownika z systemem w ojczystym języku)

Schemat z rysunku 6.1. przedstawia ogólną budowę systemu operacyjnego. Podstawową częścią OS jest jądro systemu (kernel). Jądro odpowiedzialne jest za pracę systemu i wykonywanie wszystkich jego zadań. Jądro poprzez sterowniki i BIOS komunikuje się z elektroniką komputera. Aby użytkownik mógł komunikować się z jądrem system operacyjny posiada powłokę (Shell lub też interpreter). Powłoka systemowa jest to program pełniący rolę pośrednika pomiędzy całym systemem operacyjnym a użytkownikiem. Powłoki mogą być tekstowe lub graficzne.

**Systemy operacyjne podzielić można ze względu na:**



Rysunek 6.2: Składniki systemu operacyjnego i ich zadania

#### Sposób komunikacji systemu z użytkownikiem:

- systemy tekstowe - komunikacja przebiega przy pomocy komend wprowadzanych z linii poleceń (DOS, CP/M)
- systemy graficzne - komunikacja odbywa się przy pomocy graficznych symboli (okienek oraz ikon); obsługa systemu polega na manipulacji przy pomocy myszy bądź klawiatury symbolami odpowiadającym określonym zadaniom (MC Windows, MacOS)

#### Architekturę systemu:

- monolityczne - jednozadaniowe systemy posiadające najprostszą strukturę, gdzie w danym czasie może być realizowane tylko jedno zadanie
- warstwowe - posiadające hierarchiczną strukturę poleceń systemowych; możliwa jest realizacja wielu zadań jednocześnie (np. nadzorowanie procesu drukowania podczas edycji tekstu)
- klient/serwer - systemy posiadając bardzo rozbudowaną strukturę, nadzorujące podrzędne systemy zainstalowane na komputerach w sieci; systemy te postrzegają aplikacje jako klientów, korzystających z usług serwerów; aplikacja „klient” komunikuje się z serwerem przez jądro systemowe, natomiast każdy serwer działa we własnej, chronionej i wydzielonej przestrzeni adresowej w pamięci operacyjnej, gdzie jest odizolowany od innych zadań; systemy klient/serwer realizują swe zadania na trzy sposoby:
  - wszelkie aplikacje uruchamiane są na serwerze, a wyniki prezentowane u "klienta"
  - serwer dostarcza zasobów dla aplikacji, które uruchamiane są po stronie klienta
  - wszelkie komputery współdziałają ze sobą na zasadzie równy z równym (ang. peer to peer), wykorzystując wzajemnie swoje zasoby

#### Przykłady systemów:

- Windows
- Linux
- Unix

- OS X/ Mac OS X
- Android



## Rozdział 7

# Zadania optymalizacji i techniki ich rozwiązywania

Optymalizacja - znalezienie wariantu ocenianego według danego kryterium jako najlepszy spośród wariantów uznanych za dopuszczalne.

### 0.1 Sformułowanie zadania optymalizacji

Wektor zmiennych decyzyjnych  $x$ :

$$x = [x_1, x_2, \dots, x_n]^T,$$

gdzie:

$n$  - ilość zmiennych decyzyjnych.

Funkcja celu (funkcja kryterialna)  $f(x)$ :

$$f(x) : R^n \longrightarrow R^1$$

oraz  $m$  funkcji ograniczeń  $g_i(x)$ :

$$g_i(x) : R^n \longrightarrow R^1 \quad \text{dla } i = 1, \dots, m$$

Zadanie optymalizacji polega na znalezieniu wektora zmiennych decyzyjnych  $x$ , należącego do zbioru rozwiązań dopuszczalnych  $\mathbf{X}$  w postaci:

$$X = \{x | g_i(x) \leq 0, \quad i = 1, \dots, m\}$$

takiego, że dla wszystkich  $x \in X$

$$f(\hat{x}) \leq f(x).$$

Co jest równoznaczne zapisowi:

$$\min_{x \in X} f(x) = f(\hat{x}).$$

## Ogólna postać zadania optymalizacji

Dlaczego jako ogólne zapisaliśmy zadanie minimalizacji ?

Jak łatwo zauważyć, dla każdego  $D$  :

$$\max_{x \in D} f(x) = -\min_{x \in D} [-f(x)] ,$$

o ile to pierwsze maksimum istnieje.

Zatem wystarczy rozważać tylko jeden z dwu operatorów ekstremalizacji. Do systemu definicji przyjętych w matematyce, lepiej pasuje **minimalizacja** i w zasadzie wszystkie rozważania teoretyczne prezentowane w literaturze dotyczą minimalizacji.

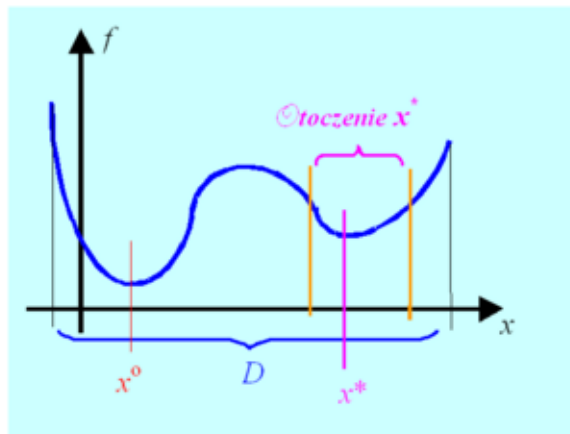
$D$  - zbiory dopuszczalne.

PO POLSKU:

Sformułowanie zadania optymalizacji polega na znalezieniu takiego zestawu wielkości zmiennych (z wektora zmiennych decyzyjnych), że będzie spełniał podane ograniczenia (na przykład że nie mogą być wartości ujemne), czyli należał do zbioru dopuszczalnego, oraz że będzie to rozwiązanie najlepsze według podanego kryterium, czyli funkcji celu (najmniejsze, największe, itp). Może też okazać się , że zadanie nie posiada rozwiązania , bądź też posiada nieskończenie wiele rozwiązań optymalnych.

Przypomnienie czym są ekstrema lokalne i globalne:

## Minima lokalne i minimum globalne



Przedstawiona na rysunku funkcja jednej zmiennej obok minimum globalnego w punkcie  $x^0$  ma minimum lokalne w punkcie  $x^*$ .

## Minima lokalne i minimum globalne

Dokładna definicja jest następująca:

Punkt  $x^* \in D$  nazywamy punktem minimum lokalnego funkcji  $f$  w tym zbiorze jeżeli istnieje takie jego otoczenie  $\mathcal{O}(x^*)$ , że

$$(\forall x \in \mathcal{O}(x^*) \cap D) f(x^*) \leq f(x).$$

Zauważmy, że zgodnie z podanymi definicjami - punkt minimum globalnego jest najlepszym spośród minimów lokalnych.

### Klasyfikacja zadań

Ze względu na ograniczenia:

- Zadania bez ograniczeń (dopuszczalne wszystkie warianty)
- Zadania z ograniczeniami (określono w zadaniu zbiór wartości dopuszczalnych)

Ze względu na postać:

- Zadania liniowe - oznacza że funkcja celu oraz układ ograniczeń są liniowe
- Zadania nieliniowe

Inne podziały:

- metody rozwiązania: numeryczne/analityczne
- ze względu na liczbę funkcji celu : a) zadania jednokryterialne (skalarna funkcja celu)/ zadania wielokryterialne (więcej niż jedno kryterium funkcja celu wektorowa)

- ze względu na typ zmiennych decyzyjnych: całkowitoliczbowe/zerojedynkowe/mieszane
- kilka innych

Najważniejszy podział to na zadanie liniowe lub nieliniowe, ponieważ zadanie nie może należeć jednocześnie do obu tych klas, co nie dotyczy pozostałych podziałów, np, możliwe jest zadanie bez ograniczeń nieliniowe.

## 0.2 Zadanie programowania liniowego PL

Dobrze wytłumaczone zadanie programowanie liniowe:

<http://www.cs.put.poznan.pl/jjozefowska/wyklady/bo/Rozdzial3.pdf> Warto przeanalizować przykład z kawą, wtedy wszystko jest jaśniejsze.

---

### 3.2.1. Ogólne sformułowanie problemu programowania liniowego

Sformułowanie problemu rozważanego w punkcie 3.1 można uogólnić na  $n$  zmiennych i  $m$  ograniczeń następująco:

zmaksymalizować (zminimalizować)

$$z = f(x_1, \dots, x_n) = \sum_{j=1}^n c_j x_j \quad (3.6)$$

przy ograniczeniach

$$g_i(x_1, \dots, x_n) = \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, \dots, m \quad (3.7)$$

Współczynniki  $c_j, b_i, a_{ij}, j = 1, \dots, n, i = 1, \dots, m$  są parametrami problemu, a  $x_j, j = 1, \dots, n$ , zmiennymi decyzyjnymi.

Przypadki szczególne rozwiązania zadania programowania liniowego (PL)

- Istnieje jedno rozwiązanie optymalne zadani PL.
- Zadanie PL jest zadaniem nieograniczonym – brak rozwiązania.
- W zadaniu PL zbiór rozwiązań dopuszczalnych jest pusty – brak rozwiązania.
- W zadaniu PL wszystkie zmienne lub ich część przyjmuje
- wartości rzeczywiste.
- Zadanie PL posiada nieskończoną liczbę rozwiązań.

**Metody rozwiązywania:**

- simpleks
- dwufazowy simpleks
- dualny simpleks
- metoda graficzna

## Metoda simpleks

Algorytm simpleks wymaga sprowadzenia zadania PL do postaci standardowej, w której ograniczenia mają postać równań, wszystkie zmienne są nieujemne, a także prawe strony ograniczeń są liczbami nieujemnymi. Czasem wymaga to dodania do macierzy pomocniczych zmiennych (w przypadku ograniczeń mniejszościowych lub większościowych).

Przyjmując  $m$  - liczba ograniczeń,  $n$  - liczba zmiennych decyzyjnych. Obliczenia simpleks wygodnie jest prowadzić w tablicy, która reprezentuje bazowe rozwiązanie dopuszczalne. Ma  $m+1$  wierszy i  $n$  kolumn. Kolumny to zmienne decyzyjne, wiersze odpowiadają zmiennym bazowym.

1. tworzy się postać początkową tablicy simpleks zawierającą rozwiązanie dopuszczalne. Jeśli dopuszczalne, następny krok.
2. Test optymalności (czy współczynniki w wierszu  $x_0$  są dodatnie?) jeśli tak, to jest rozwiązanie, jeśli nie - następny krok.
3. Wybór zmiennej wchodzącej do bazy (najmniejsza wartość w wierszu  $x_0$ )
4. wybór zmiennej usuwanej z bazy
5. Eliminacja Gaussa w celu uzyskania nowego rozwiązania bazowego dopuszczalnego. Powrót do kroku 1.

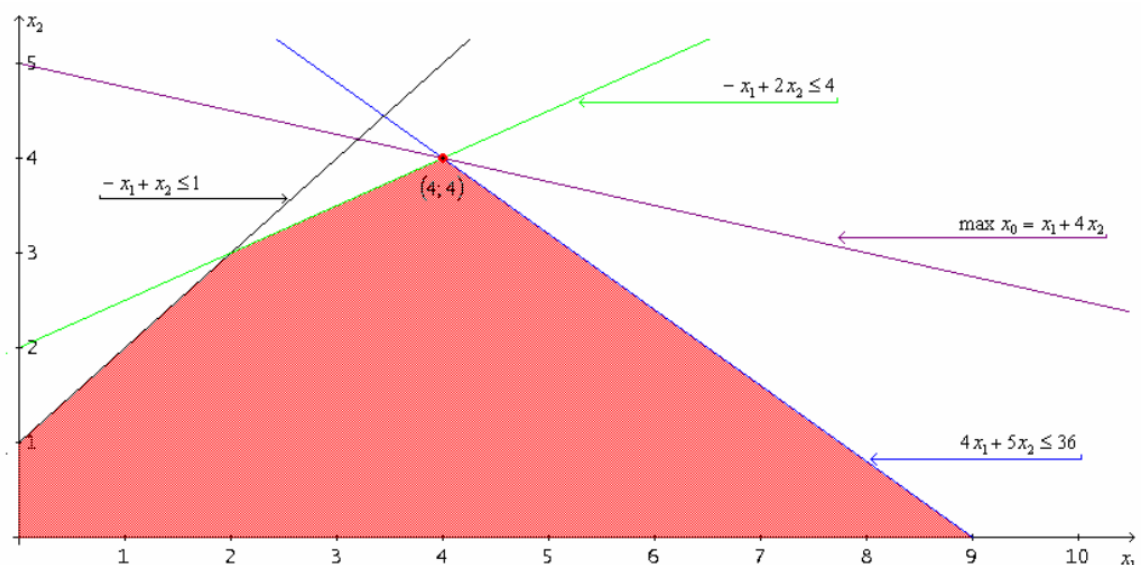
Tutaj jest rozwiązany przykład na str 3., warto to przejrzeć i samemu zrozumieć. [http://staff.iiar.pwr.wroc.pl/ewa.szlachcic/materialy%20dydaktyczne/eit%20technika%20optymalizacji/4w\\_to\\_proglin2.pdf](http://staff.iiar.pwr.wroc.pl/ewa.szlachcic/materialy%20dydaktyczne/eit%20technika%20optymalizacji/4w_to_proglin2.pdf)

## Metoda graficzna

Jest najprostsza, ale można ją zastosować jedynie do zadań o dwóch zmiennych decyzyjnych.

Funkcja celu:  $\max x_0 = x_1 + 4x_2$

Ograniczenia: 
$$\begin{cases} 1. -x_1 + 2x_2 \leq 4 \\ 2. -x_1 + x_2 \leq 1 \\ 3. 4x_1 + 5x_2 \leq 36 \end{cases}, x \geq 0$$



Rysujemy ograniczenia, zaznaczamy zbiór dopuszczalny, rysujemy funkcję celu i wybieramy najlepszy punkt optycznie.

## Metoda simpleksu dwufazowego

Ma dwie fazy:

1. należy znaleźć pierwsze rozwiązanie bazowe dopuszczalne poprzez rozwiązanie zadania pomocniczego (wprowadzenie pomocniczej funkcji celu.)
2. maksymalizacja funkcji celu  $x_0$  dla następnego rozwiązania bazowego dopuszczalnego wg algorytmu simpleks.

I faza zachodzi, gdy Krok 1 – nie ma możliwości stworzenia pierwszego rozwiązania bazowego dopuszczalnego.

### Zadanie pomocnicze

#### I faza metody dwufazowej simpleks – uzyskanie bazowego rozwiązania dopuszczalnego poprzez wprowadzenie pomocniczej funkcji celu

- Jeśli wektor  $b$  w początkowej tablicy simpleksowej ma przynajmniej jedną ujemną współrzędną, to tablica przedstawia niedopuszczalne rozwiązanie bazowe.
- Początkową niedopuszczalną tablicę simpleksową można przekształcić do tablicy dopuszczalnej wykorzystując algorytm simpleks.
- Cel – uzyskanie nieujemnych wartości zmiennych pomocniczych.
- Należy wybrać najmniejszą ujemną wartość zmiennej bazowej
$$r \Rightarrow \min_i \{y_{i0}, y_{i0} < 0 \text{ dla } i = 1, 2, \dots, m\}$$
- Wybrany wiersz  $s$  będzie stanowił **pomocniczą funkcję celu**, którą należy zmaksymalizować.
- Kolejne kroki metody simpleks powinny być prowadzone do uzyskania dopuszczalnej tablicy simpleks, tzn. takiej dla której spełniony jest warunek prymalnej dopuszczalności:

$$y_{i0} \geq 0 \text{ dla } i = 1, 2, \dots, m$$

- Koniec I fazy

Po tej fazie następuje II faza, czyli podstawowy simpleks (prymalny).

Szczególny przypadek jest wtedy, gdy po I fazie nadal nie ma rozwiązania dopuszczalnego. Wtedy brak rozwiązań.

przykład:

[http://staff.iiar.pwr.wroc.pl/ewa.szlachcic/materialy%20dydaktyczne/eit%20technika%20optymalizacji/5w\\_to\\_proglin3.pdf](http://staff.iiar.pwr.wroc.pl/ewa.szlachcic/materialy%20dydaktyczne/eit%20technika%20optymalizacji/5w_to_proglin3.pdf)

W tym miejscu, wszem i wobec oddaję honor pani dziekan, ponieważ jej przykłady obliczeniowe(!) są czytelne i łatwe do zrozumienia, w przeciwieństwie do internetowych, amen.

### Zadanie liniowego programowania całkowitoliczbowego PCL

Zagadnieniem liniowym całkowitoliczbowym nazywamy zadanie optymalizacji liniowej, w którym wszystkie zmienne lub ich część są liczbami całkowitymi.

$$\max_{x \in X} x_0 = c^T x \text{ dla } x \in X \subset R^n$$

dla każdego

$$1 \leq i \leq t \quad x_i \in \mathbb{Z}$$

Metody uproszczone:

- Przegląd zupełny zbioru rozwiązań dopuszczalnych  $X$
- Zaokrąglenie rozwiązania optymalnego zadania PL do zmiennych całkowitych

Zaokrąglenie do części całkowitych rozwiązania optymalnego zadania PL w celu zyskania rozwiązania zadania PCL. W ten sposób można uzyskać rozwiązanie PCL nie spełniające zbioru rozwiązań dopuszczalnych  $X$ . Należy sprawdzić miarę niedopuszczalności rozwiązania zaokrąglonego mierząc procentowe niespełnienie ograniczeń.

Metody złożone:

- Metoda odcięć Gomory'ego
  1. Krok 1 Znajdź rozwiązanie zadania PL (2). Idź do Kroku 2.
  2. Krok 2 Test na optymalność  
Jeśli rozwiązanie zadania jest całkowitoliczbowe, wówczas jest ono jednocześnie rozwiązaniem zadania (1) – Stop. W przeciwnym wypadku idź do Kroku 3.
  3. Krok 3 Odcinanie  
Dodaj odcięcie z odpowiednio dobraną wartością  $h$ . Dokonaj ponownej optymalizacji stosując algorytm simpleks. Może zaistnieć konieczność wykonania większej liczby kroków eliminacji, zgodnie z procedurą simpleks. Wróć do Kroku 2.

Odcięciem Gomory'ego nazywamy więc ograniczenie obszaru dopuszczalnego półprzestrzenią :

$$H = \{x \in \mathbb{R}^n \mid \sum_{i=1}^n \lfloor a_i \rfloor x_i \leq \lfloor b \rfloor\}$$

- Metoda podziału i ograniczeń  
Podstawą metody B&B jest przegląd drzewa rozwiązań. Wykorzystuje się fakt skończoności zbioru możliwych wartości zmiennych całkowitoliczbowych w przypadku ograniczonych zadań PCL.  
W każdym wierzchołku drzewa rozwiązuje się problem programowania liniowego z dodatkowym ograniczeniem na wybraną zmienną. Warunek całkowitoliczbowości zmiennych zostaje odrzucony.
- Metody przybliżone = metody heurystyczne

### 0.3 Zadanie programowania nieliniowego PN

$$\min_{x \in X} f(x) = f(\hat{x})$$

przy ograniczeniach:

$$X = \{x \mid g_i(x) \leq 0, i = 1, \dots, m\}$$

Zadanie programowania nieliniowego polega na znalezieniu wektora zmiennych decyzyjnych  $\hat{x}$ , należącego do zbioru rozwiązań dopuszczalnych  $X$  w postaci:

takiego, że dla

$$x \in X$$

$$f(\hat{x}) \leq f(x)$$

Przeważnie stosuje się metody przybliżone, tylko szczególne przypadki można rozwiązać analitycznie, na przykład sprowadzając do problemu liniowego. Metoda rozwiązywania zależy od postaci, jaką przyjmuje zadanie, nie ma uniwersalnej.

### Zadanie programowania nieliniowego bez ograniczeń

Twierdzenie. Niech  $f: X \subset \mathbb{R}^n \rightarrow \mathbb{R}^1$  będzie funkcją różniczkowalną. Jeśli minimalizuje funkcję  $f(x)$  tzn.

$$f(\hat{x}) \leq f(x), \quad \forall x \in X,$$

to

$$\nabla f(\hat{x}) = 0$$

Dowód: nie wprost.

Punkt  $\hat{x}$  jest nazywany punktem stacjonarnym.

Twierdzenie. Niech  $f: X \subset \mathbb{R}^n \rightarrow \mathbb{R}^1$  będzie funkcją wypukłą i różniczkowalną. Punkt  $\hat{x} \in X$  stanowi minimum globalne funkcji

$$f(x), \text{ tzn. } f(\hat{x}) \leq f(x),$$

dla każdego  $x \in X$  wtedy i tylko wtedy, gdy  $\hat{x}$  spełnia warunek

$$\nabla f(\hat{x}) = 0$$

Jest to warunek konieczny istnienia ekstremum lokalnego  $f(x)$  w punkcie  $\hat{x}$ .

Technika optymalizacji  
Dr inż. Ewa Szlachetka

Wydział Elektroniki studia I st.  
Kierunek Elektronika III r.

### Minimum globalne funkcji $f(x)$

**Twierdzenie:**

Jeśli  $f: X \subset \mathbb{R}^n \rightarrow \mathbb{R}^1$  będzie funkcją ściśle wypukłą i różniczkowalną, to wektor  $\hat{x} \in X$  spełniający warunek konieczny  $\nabla f(\hat{x}) = 0$  jest jedynym minimum globalnym funkcji  $f(x)$ .

Technika optymalizacji  
Dr inż. Ewa Szlachetka

Wydział Elektroniki studia I st.  
Kierunek Elektronika III r.

### Warunki wystarczające optymalizacji dla zadania bez ograniczeń

Funkcja  $f(x)$  jest funkcją ciągłą i dwukrotnie różniczkowalną. Posiada macierz drugich pochodnych (hesjan) - A

Macierz A posiada ciąg podwyznaczników głównych  $|A_i|$

$$|A_1| = \left| \frac{\partial^2 f}{\partial x_1^2}(x) \right|$$

$$|A_2| = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) \end{vmatrix}$$

$$|A_n| = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) & \dots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{vmatrix}$$

Technika optymalizacji  
Dr inż. Ewa Szlachetka

Wydział Elektroniki studia I st.  
Kierunek Elektronika III r.

### Warunki stacjonarności dla nieliniowej zadania optymalizacji bez ograniczeń cd.

**Twierdzenie:**

Założono, że  $\hat{x}$  jest punktem stacjonarnym funkcji  $f(x)$ . Wówczas zachodzą poniższe zależności:

1. Jeśli hesjan A jest dodatnio określony tzn:  $|A_i(x)| > 0$  dla  $i=1, \dots, n$  to funkcja  $f(x)$  ma minimum lokalne w tym punkcie
2. Jeśli hesjan A jest ujemnie określony tzn:  $(-1)^i |A_i(x)| > 0$  dla  $i=1, \dots, n$  to funkcja  $f(x)$  ma maksimum lokalne w tym punkcie
3. Jeśli hesjan A jest pół-dodatnio określony tzn:  $|A_i(x)| \geq 0$  dla  $i=1, \dots, n-1$  oraz  $|A_n(x)| = 0$  bądź hesjan pół-ujemnie określony  $(-1)^i |A_i(x)| \geq 0$  dla  $i=1, \dots, n-1$  oraz  $|A_n(x)| = 0$

to nie można rozstrzygnąć o typie ekstremum funkcji  $f(x)$  w tym punkcie

4. Jeśli nie są spełnione warunki 1 i 2 z nieostrych nierównościami (wówczas hesjan A nie jest określony) to funkcja  $f(x)$  nie ma ekstremum w punkcie  $\hat{x}$

Technika optymalizacji  
Dr inż. Ewa Szlachetka

Wydział Elektroniki studia I st.  
Kierunek Elektronika III r.



### Schemat algorytmu optymalizacji lokalnej bez ograniczeń

- (1) Wybierz punkt startowy  $\mathbf{x}^0 = \mathbf{x}^k$ .
- (2) Oblicz wartość funkcji  $f(\mathbf{x}^k)$  oraz jeżeli jest to wymagane to jej gradient  $\nabla f(\mathbf{x}^k)$  czy *hesjan*  $H = \nabla^2 f(\mathbf{x}^k)$
- (3) Zbadaj przyjęte kryterium zbieżności.
  - Jeśli kryterium jest spełnione to koniec algorytmu – uzyskano rozwiązanie optymalne  $\mathbf{x}^k$  i optymalną wartość funkcji celu  $f(\mathbf{x}^k)$
  - Jeżeli nie, to przejdź do (4)
- (4) Wyznacz ustalony kierunek poszukiwań :  $d^k$
- (5) Wykonaj minimalizację kierunkową wybraną metodą:

$$\mathbf{x}^{k+1} \in T(\mathbf{x}^k, d^k)$$

- (6) Podstaw  $\mathbf{x}^k \leftarrow \mathbf{x}^{k+1}$  oraz  $k \leftarrow k + 1$   
i przejdź do (2)

Technika optymalizacji  
Dr inż. Ewa Szlachcic

Wydział Elektroniki studia I st.  
Kierunek Elektronika III r.

### Nieliniowe zadanie optymalizacji statycznej z ograniczeniami

Znaleźć wektor rozwiązań optymalnych  $\hat{x}$ , taki, że :

$$f(\hat{x}) = \min_{x \in X} f(x)$$

gdzie:

$$X = \{x : g_i(x) \leq 0, \quad i = 1, \dots, m\}$$

$$f : X \subset \mathbb{R}^n \longrightarrow \mathbb{R}^1$$

oraz

$$g_i : X \subset \mathbb{R}^n \longrightarrow \mathbb{R}^1 \text{ dla każdego } i = 1, \dots, m$$

#### Ograniczenie aktywne

Ograniczenie nierównościowe, spełnione równościowo nazywa się ograniczeniem aktywnym. Dla  $\hat{x}$  ograniczenie przyjmuje postać:

$$g_i(\hat{x}) = 0$$

$$g_i(\hat{x} + \tau d) \leq 0, \text{ dla każdego } i \in A(\hat{x})$$

przy czym  $\tau \in [0, \sigma]$

### Przypadki:

1. Jedno ograniczenie aktywne – rozwiązanie optymalne na brzegu ograniczenia aktywnego.
2. Wszystkie ograniczenia aktywne - rozwiązanie optymalne na przecięciu ograniczeń aktywnych.
3. Brak ograniczeń aktywnych – rozwiązanie optymalne wewnątrz obszaru rozwiązań dopuszczalnych ( rozwiązanie optymalne zadania bez ograniczeń spełnia ograniczenia problemu.

W celu uwzględnienia ograniczeń można postąpić w poniższy sposób:

- dokonać transformacji zmiennych decyzyjnych
- dokonać transformacji funkcji celu wprowadzając funkcje kary

Funkcja kary charakteryzuje się tym, że w zbiorze rozwiązań dopuszczalnych  $X$  przyjmuje wartość równą zero lub bliską zero, a poza tym obszarem przyjmuje bardzo duże wartości.

Rodzaje funkcji kary:

- Metody zewnętrznej funkcji kary (metoda Couranta, metoda Schmita i Foxa)
- Metody wewnętrznej funkcji kary (metoda Rosenbrocka, metoda Carolla)
- Metody przesuwanej funkcji kary (metoda Powella).

Rozwiązanie zadania nieliniowego z ograniczeniami oznacza znalezienie takiego wektora rozwiązań optymalnych, że będzie on spełniał kryteria (to co w Matlabie było Tolx, TolCon...).

### Iteracyjne algorytmy optymalizacji:

- Algorytmy optymalizacji w kierunku
- Algorytmy optymalizacji bez ograniczeń
- Algorytmy optymalizacji z ograniczeniami

Algorytm optymalizacji lokalnej (czyli to co wyżej) - przemierzanie obszaru rozwiązań dopuszczalnych w poszukiwaniu ekstremum funkcji celu według iteracyjnego schematu.

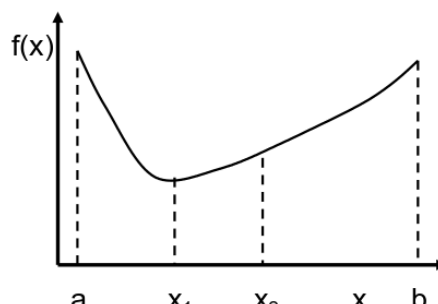
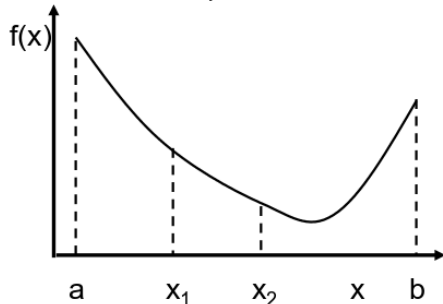
**Algorytmy bezgradientowe (minimalizacja w kierunku -wykorzystują tylko wartości funkcji)**

- Algorytm Nelder'a-Meade'a ( Matlab - funkcja fminsearch)
- Algorytm Gauss'a-Seidla (bardzo wolna zbieżność liniowa)
- Algorytm Powella
- złotego podziału

## Metoda złotego podziału

Jeżeli funkcja  $f(x)$  jest *unimodalna* (posiada tylko jedno minimum) w przedziale  $[a, b]$  to dla określenia podprzedziału, w którym leży punkt stacjonarny należy obliczyć wartość funkcji w *dwóch* punktach tego przedziału oprócz końców przedziału.

**Jeżeli dla  $a < x_1 < x_2 < b$  zachodzi  $f(x_2) < f(x_1)$  i  $f(x_2) < f(b)$  - to minimum znajduje się pomiędzy  $x_1$  i  $b$ .**



**Jeżeli dla  $a < x_1 < x_2 < b$  zachodzi  $f(a) > f(x_1)$  i  $f(x_1) < f(x_2)$  - to minimum znajduje się pomiędzy  $a$  oraz  $x_2$ ;**

**Te obserwacje stanowią podstawę zawężania przedziału, w którym zawarte jest minimum.**

Technika optymalizacji  
Dr inż. Ewa Szlachcic

Wydział Elektroniki studia I st.  
Kierunek Elektronika III r.

- aproksymacji kwadratowej

**Algorytmy gradientowe (minimalizacja w kierunku - wykorzystują wartości funkcji oraz co najmniej wartość pochodnej kierunkowej w kierunku poszukiwań):**

- Algorytm największego spadku (zbieżność liniowa)
- Algorytm Newtona (zbieżna kwadratowo ale kosztowna i nie zawsze stabilna)
- Algorytm Zangwilla
- Algorytm Fletchera-Reeves'a
- Algorytm Polaka-Ribiera
- Algorytm Fletchera-Powella-Davidona

Najefektywniejsze są tzw. metody quasi-newtonowskie, w których w kolejnych iteracjach konstruuje się przybliżenie odwrotności hesjanu.

Przykłady metod quasinewtonowskich:

- Metoda Broyden-Fletcher-Goldfarb-Shanno (BFGS)  
Właściwości metody: -Brak operacji odwracania macierzy hesjanu  
-Blisko rozwiązania optymalnego – dobra zbieżność  
-Na początku obliczeń – zbieżność słaba – słabe jest przybliżenie hesjanu  
-Przybliżenie hesjanu w każdej iteracji polepsza się

[chmielowski.eu/POLITECHNIKA/Dydaktyka/OPTYdoktoranci/Wyklady/W1.doc](http://chmielowski.eu/POLITECHNIKA/Dydaktyka/OPTYdoktoranci/Wyklady/W1.doc)  
[smurf.mimuw.edu.pl/external\\_slides/MO/MO\\_1\\_Przyklady\\_zadan\\_optymalizacji/MO\\_1\\_Przyklady\\_zadan\\_optymalizacji.html](http://smurf.mimuw.edu.pl/external_slides/MO/MO_1_Przyklady_zadan_optymalizacji/MO_1_Przyklady_zadan_optymalizacji.html)

## 0.4 Optymalizacja globalna

Do tej grupy należą stochastyczne iteracyjne algorytmy przeszukiwania przestrzeni rozwiązań :

- metody przeszukiwania lokalnego
- metody przeszukiwania populacyjnego.

SCHEMAT ALGORYTMU:

1. Wygeneruj początkowy zbiór rozwiązań  $W$  i oceń każde z nich.
2. Wygeneruj i oceń zbiór nowych kandydatów  $W'$  drogą losowych zmian u wybranych osobników z  $W$ .
3. Zastąp pewne osobniki z  $W$  osobnikami z  $W'$  i wróć do kroku 2, o ile nie jest spełniony warunek zatrzymania algorytmu.

Są dwie metody generowania nowych osobników:

- Kolejni kandydaci są niewielkimi modyfikacjami po przednich kandydatów (mutacja). dotyczy metod przeszukiwania lokalnego.
- Nowy kandydat powstaje w drodze rekombinacji pewnych cech dwóch lub więcej poprzedników (krzyżowanie). Dotyczy przeszukiwania populacyjnego

Metody przeszukiwania lokalnego – algorytm największego wzrostu

1. Wygeneruj i oceń początkowe „aktualne rozwiązanie”  $s$ .
2. Zmodyfikuj  $s$  otrzymując  $s'$  i oceń  $s'$ .
3. Jeżeli  $s'$  jest lepsze niż  $s$ , podstaw  $s \leftarrow s'$
4. Wróć do kroku 2, chyba że jest spełniony warunek zatrzymania algorytmu.

W tym algorytmie wykorzystano pomysł (1!): nowe potencjalne rozwiązania generowane są drogą niewielkich modyfikacji aktualnego rozwiązania.

Podstawowa różnica uwidacznia się w kroku 3. Czasem można zaakceptować rozwiązanie  $s'$  nawet wtedy gdy jest ono gorsze niż  $s$ . Pozwala to uniknąć (choć nie zawsze) stabilizacji aktualnego rozwiązania w lokalnym optimum.

Teoria i metody optymalizacji  
Dr inż. Ewa Szlachdź

Wydział Elektroniki studia II st.  
kier. Automatyka i Robotyka

Metody przeszukiwania populacyjnego

- W populacyjnych metodach przeszukiwania stosuje się zamiast pojedynczego rozwiązania aktualnego populację (zazwyczaj różnych) rozwiązań aktualnych.
- Nowe rozwiązania uzyskiwane są drogą wyboru z populacji „rodziców” i odpowiedniego modyfikowania ich.
- Tutaj pomysł (2) odgrywa główną rolę:  
*Nowy kandydat powstaje w drodze rekombinacji pewnych cech dwóch lub więcej poprzedników.*

Teoria i metody optymalizacji  
Dr inż. Ewa Szlachdź

Wydział Elektroniki studia II st.  
kier. Automatyka i Robotyka

### Techniki heurystyczne

Algorytmy lokalnego poszukiwania – klasa algorytmów, w których rozwiązanie problemu jest iteracyjne poprawiane poprzez przeglądanie sąsiedztwa rozwiązania.

Metaheurystyki (naśladowanie rzeczywistości):

- symulowanie wyżarzanie  
Na początku działania algorytmu temperatura (patametr) jest wysoka, dzięki czemu algorytm może bardzo często zmieniać konfigurację rozwiązania, niejednokrotnie wybierając rozwiązanie gorsze. Wraz z kolejnymi iteracjami algorytmu temperatura spada i wybierane są częściej rozwiązania lepsze. Pod koniec pracy algorytmu, temperatura jest na tyle niska, że prawdopodobieństwo wyboru gorszego rozwiązania jest bliskie zeru. Algorytm zachowuje się wówczas, jak typowy algorytm iteracyjny i stara się maksymalnie ulepszyć rozwiązanie.
- przeszukiwanie TABU  
Jest to metoda pozwalająca uniknąć niebezpieczeństwa wielokrotnego powracania do tego samego rozwiązania.  
Algorytm jest wyposażony w pamięć dotychczas odwiedzanych punktów.  
Ponowne odwiedzenie punktów znajdujących się w pamięci tabu jest zakazane.

- systemy mrówkowe

W prawdziwym świecie, mrówki poruszają się w sposób losowy; gdy znajdują pożywienie, wracają do swojej kolonii pozostawiając ślad składający się z feromonów. Gdy inna mrówka natknie się na ten ślad, przestaje poruszać się w sposób losowy i podąża za śladem w kierunku pożywienia. Gdy mrówki odnajdują najbliższy położony pokarm wracają szybciej zostawiając szybciej feromony i jest ich coraz więcej, gdyż reszta podąża za częstszym zapachem. W ten sposób na koniec większość mrówek chodzi drogą najkrótszą.

- algorytmy ewolucyjne Idea algorytmów genetycznych polega na naśladowaniu ewolucji gatunków. Podczas iteracji kod genetyczny czyli chromosomy ulegają krzyżowaniu lub selekcji tworząc w ten sposób potomstwo, mogą one również podlegać mutacji z pewnym prawdopodobieństwem. Algorytmy genetyczne umożliwiają utrzymanie równowagi pomiędzy szerokim badaniem przestrzeni a wykorzystaniem wcześniejszych wyników. W każdej następnej populacji znajdują się osobniki lepsze, zgodnie z funkcją celu. Osobniki gorsze podlegają eliminacji.
- optymalizacja rojem cząstek Każda cząstka reprezentuje potencjalne rozwiązanie. Cząstki poruszają się przez wielowymiarową przestrzeń poszukiwań. Pozycja jest aktualizowana według własnego doświadczenia oraz doświadczenia sąsiadów, poprzez dodanie wektora prędkości do ich aktualnego stanu przemieszczania się. Aktualna prędkość zależy od, poprzedniej prędkości, dążenia do swojego najlepszego poprzedniego położenia oraz dążenia do globalnego lub lokalnego najlepszego wyniku sąsiada. Każda cząstka jest zbieżna do punktu pomiędzy własnym najlepszym rozwiązaniem a najlepszym rozwiązaniem globalnym.

## 0.5 Sformułowanie zadania optymalizacji

Wektor zmiennych decyzyjnych  $x$ :

$$x = [x_1, x_2, \dots, x_n]^T,$$

gdzie:

$n$  - ilość zmiennych decyzyjnych.

Funkcja celu (funkcja kryterialna)  $f(x)$ :

$$f(x) : R^n \longrightarrow R^1$$

oraz  $m$  funkcji ograniczeń  $g_i(x)$ :

$$g_i(x) : R^n \longrightarrow R^1 \quad \text{dla } i = 1, \dots, m$$

Zadanie optymalizacji polega na znalezieniu wektora zmiennych decyzyjnych  $x$ , należącego do zbioru rozwiązań dopuszczalnych  $X$  w postaci:

$$X = \{x | g_i(x) \leq 0, \quad i = 1, \dots, m\}$$

takiego, że dla wszystkich  $x \in X$

$$f(\hat{x}) \leq f(x).$$

Co jest równoznaczne zapisowi:

$$\min_{x \in X} f(x) = f(\hat{x}).$$

## 0.6 Rodzaje zadań

**Zadanie programowania liniowego PL**

$$\max f(x) = c^T x$$

przy ograniczeniach:

$$A_1 x \leq b_1$$

$$A_2 x \geq b_2$$

$$x \geq 0$$

$$\dim x=n, \dim c = n$$

Macierze  $A_1, A_2$  odpowiadają za współczynniki w  $m_1 \times m_2$  ograniczeniach  
Wektory  $b_1, b_2$  odpowiadają za prawe strony ograniczeń

Przypadki szczególne rozwiązania zadania programowania liniowego (PL)

- Istnieje jedno rozwiązanie optymalne zadani PL.
- Zadanie PL jest zadaniem nieograniczonym – brak rozwiązania.
- W zadaniu PL zbiór rozwiązań dopuszczalnych jest pusty – brak rozwiązania.
- W zadaniu PL wszystkie zmienne lub ich część przyjmuje
- wartości rzeczywiste.
- Zadanie PL posiada nieskończoną liczbę rozwiązań.

**Metody rozwiązywania:**

- simpleks
- dwufazowy simpleks
- dualny simpleks

**Zadanie liniowego programowania całkowitoliczbowego PCL**

Zagadnieniem liniowym całkowitoliczbowym nazywamy zadanie optymalizacji liniowej, w którym wszystkie zmienne lub ich część są liczbami całkowitymi.

$$\max_{x \in X} c^T x \quad \text{dla } x \in X \subset R^n$$

dla każdego

$$1 \leq i \leq t \quad x_i \in \mathbb{Z}$$

**Metody uproszczone:**

- Przegląd zupełny zbioru rozwiązań dopuszczalnych X
- Zaokrąglenie rozwiązania optymalnego zadania PL do zmiennych całkowitych

**Metody złożone:**

- Metoda odcięć Gomory'ego
- Metoda podziału i ograniczeń
- Metody przybliżone = metody heurystyczne

**Zadanie programowania nieliniowego PN**

$$\min_{x \in X} f(x) = f(\hat{x})$$

przy ograniczeniach:

$$X = \{x | g_i(x) \leq 0, i = 1, \dots, m\}$$

Zadanie programowania nieliniowego polega na znalezieniu wektora zmiennych decyzyjnych  $\hat{x}$ , należącego do zbioru rozwiązań dopuszczalnych  $X$  w postaci:  
takiego, że dla

$$x \in X$$

$$f(\hat{x}) \leq f(x)$$

**Nieliniowe zadanie optymalizacji statycznej bez ograniczeń** Funkcja celu  $f(x)$ :

$$f(x) : R^n \longrightarrow R^1$$

Zadanie optymalizacji polega na znalezieniu wektora zmiennych decyzyjnych  $x$ , należącego do zbioru rozwiązań dopuszczalnych  $R^n$  takiego, że dla każdego  $x \in R^n$

$$f(\hat{x}) \leq f(x)$$

Co jest równoznaczne zapisowi:

$$\min_{x \in R^n} f(x) = f(\hat{x})$$

#### **Iteracyjne algorytmy optymalizacji:**

- Algorytmy optymalizacji w kierunku
- Algorytmy optymalizacji bez ograniczeń
- Algorytmy optymalizacji z ograniczeniami

Algorytm optymalizacji lokalnej - przemierzanie obszaru rozwiązań dopuszczalnych w poszukiwaniu ekstremum funkcji celu według iteracyjnego schematu.

#### **Algorytmy bezgradientowe:**

- Algorytm Nelder'a-Meade'a ( Matlab - funkcja `fminsearch`)
- Algorytm Gauss'a-Seidla
- Algorytm Powella

#### **Algorytmy gradientowe:**

- Algorytm największego spadku
- Zmodyfikowany algorytm Newtona ( Matlab – wersja metody QuasiNewton - funkcja `fminunc`)
- Algorytm Zangwilla
- Algorytm Fletchera-Reeves'a
- Algorytm Polaka-Ribiera
- Algorytm Fletchera-Powella-Davidona

#### **Nieliniowe zadanie optymalizacji statycznej z ograniczeniami**

Znaleźć wektor rozwiązań optymalnych  $\hat{x}$ , taki, że :

$$f(\hat{x}) = \min_{x \in X} f(x)$$

gdzie:

$$X = \{x : g_i(x) \leq 0, \quad i = 1, \dots, m\}$$

$$f : X \subset R^n \longrightarrow R^1$$

oraz

$$g_i : X \subset R^n \longrightarrow R^1 \text{ dla każdego } i = 1, \dots, m$$

### Ograniczenie aktywne

Dla  $\hat{x}$  ograniczenie przyjmuje postać:

$$g_i(\hat{x}) = 0$$

$$g_i(\hat{x} + \tau d) \leq 0, \text{ dla każdego } i \in A(\hat{x})$$

przy czym  $\tau \in [0, \sigma]$

### Przypadki:

1. Jedno ograniczenie aktywne – rozwiązanie optymalne na brzegu ograniczenia aktywnego.
2. Wszystkie ograniczenia aktywne - rozwiązanie optymalne na przecięciu ograniczeń aktywnych.
3. Brak ograniczeń aktywnych – rozwiązanie optymalne wewnątrz obszaru rozwiązań dopuszczalnych  
( rozwiązanie optymalne zadania bez ograniczeń spełnia ograniczenia problemu.



# Rozdział 8

## Systemy dynamiczne, opisy własności

**Układ dynamiczny**, to model matematyczny rzeczywistego zjawiska przyrody, którego przyszłe zachowanie (stan, ewolucja) jest wyznaczone jednoznacznie przez stan początkowy i sygnał sterujący. Najczęściej jest opisany pewnym wektorowym równaniem różniczkowym - zwanym równaniem stanu układu.

### 1 Podział układów

#### Ze względu na charakter sygnałów

- **Układy ciągłe** - wszystkie sygnały (wejściowe i wyjściowe) są funkcjami ciągłymi w czasie i mogą przybierać dowolną wartość z obszaru swojej zmienności. Układy te opisuje się zwykle równaniami różniczkowymi.
- **Układy dyskretny** - układ jest dyskretny, jeżeli przynajmniej jeden jego sygnał ma charakter dyskretny, tzn. przyjmuje tylko określone wartości dla określonych argumentów. Układy takie opisuje się zwykle równaniami różnicowymi.

#### Ze względu na charakter układu

- **Układy statyczne** (bezinercyjne) - wyjście w danej chwili czasu zależy tylko od wejścia (brak stanu nieustalonego). Układy te składają się tylko z elementów rozpraszających energię i opisuje się je równaniami algebraicznymi.
- **Układy dynamiczne** - układy, w których wyjście nie jest jednoznaczna funkcja wejścia i zależy dodatkowo od charakteru procesu przejściowego (inercyjności) i stanu układu w chwili początkowej. Układy te zawierają elementy magazynujące energię. Opisuje się je równaniami różniczkowymi lub różnicowymi

#### Ze względu na liniowość układu

- Układy liniowe - można je opisać za pomocą liniowych równań algebraicznych, różniczkowych lub różnicowych. **Układy liniowe spełniają zasadę superpozycji**
- Układy nieliniowe - układ zawierający przynajmniej jeden element nieliniowy jest układem nieliniowym.

*W praktyce każdy układ fizyczny jest nieliniowy. Model liniowy powstaje na podstawie przybliżenia zakładającego liniowość fizycznego zjawiska i stałość parametrów lub w wyniku linearyzacji wyznaczonej nieliniowej charakterystyki. Linearyzacje przeprowadza się zwłaszcza wtedy, gdy działanie procesu ogranicza się do niewielkiego obszaru wokół pewnego ustalonego punktu pracy.*

## 2 Transmitancja operatorowa układu

**Transmitancja operatorowa** (funkcja przejścia,  $G(s)$ ) - stosunek transformaty Laplace'a sygnału wyjściowego do transformaty Laplace'a sygnału wejściowego układu przy zerowych warunkach początkowych:

$$G(s) = \frac{Y(s)}{U(s)}$$

Transmitancja określa ogólne własności stacjonarnego układu liniowego o jednym wejściu i jednym wyjściu, niezależne od rodzaju wymuszenia. Dla układu wielowymiarowego o  $r$  wejściach o  $m$  wyjściach można określić  $m \times n$  transmitancji wiążących każde wyjście z każdym wejściem.

## 3 Charakterystyki

### 3.1 Czasowa

**Charakterystyka czasowa** układu liniowego jest jego odpowiedź na określone wymuszenie przy założeniu zerowych warunków początkowych. Sygnałem wejściowym może być impuls Diraca  $\delta(t)$ , skok jednostkowy  $1(t)$  (funkcja Heaviside'a) lub sygnał narastający liniowo (rampa).

*Należy zaznaczyć, że zarówno impuls Diraca, jak i skok jednostkowy są sygnałami niemożliwymi do generacji z uwagi na skończoną szybkość zmian wartości sygnałów (na przykład skończoną prędkość przestawiania zaworu przez siłownik) lub niebezpieczeństwo wywołania dynamicznego wejścia obiektu rzeczywistego w zakres nieliniowości*

### 3.2 Skokowa

**Charakterystyka skokowa** (odpowiedzią skokowa)  $h(t)$  układu liniowego jest jego odpowiedź na wymuszenie skokiem jednostkowym  $u(t) = 1(t)$  przy zerowych warunkach początkowych. Jest ona bardzo ważnym elementem teorii sterowania, ponieważ opisuje właściwości dynamiki układu w zależności jedynie od jego parametrów i pojedynczej wartości opisującej skok (amplitudy).

*Charakterystyka skokowa umożliwia proste wyznaczenie współczynnika wzmocnienia obiektu statycznego, równego stosunkowi wartości ustalonej odpowiedzi skokowej do wartości sygnału wejściowego.*

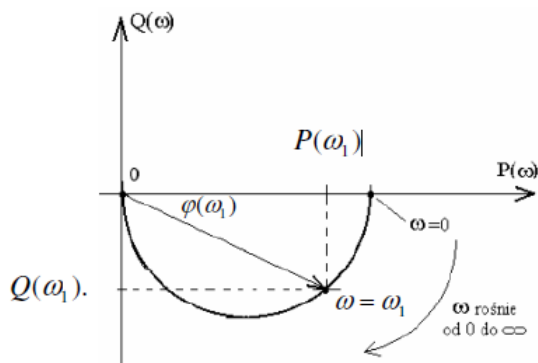
### 3.3 Impulsowa

**Charakterystyka impulsowa** (odpowiedzią impulsowa)  $g(t)$  układu liniowego jest jego odpowiedź na wymuszenie impulsem Diraca  $u(t) = \delta(t)$  przy zerowych warunkach początkowych. Podobnie jak charakterystyka skokowa opisuje również właściwości dynamiki układu w zależności od jego parametrów.

**C** charakterystyka impulsowa umożliwia w prosty sposób określenie, czy obiekt jest астатyczny – w tym przypadku wartość  $g(t)$  w stanie ustalonym jest niezerowa.

### 3.4 Charakterystyka amplitudo-fazowa

Charakterystyka amplitudowo-fazowa (wykresem Nyquista) nazywa się wykres transmitancji widmowej na płaszczyźnie zmiennej zespolonej, przy czym na osi rzędnych (osi liczb rzeczywistych) odłożona jest wartość  $P(\omega)$ , a na osi odciętych (osi liczb urojonych) wartość  $Q(\omega)$



## 4 Człony dynamiczne

Tutaj właściwie jest Rozdział 3(dokładnie 9 stron) z książki Greblickiego do przepisania, więc bezsensu byłoby to kopiować.

## 5 Stabilność

Stabilność jest jedną z najważniejszych własności, której wymaga się od systemu dynamicznego.

### 5.1 Twierdzenia i Definicje

**Definicja stabilności** Jeśli przy zerowym pobudzeniu i każdym warunku początkowym

$$\lim_{t \rightarrow \infty} y(t) = 0$$

to system nazywamy stabilnym

System jest stabilny wtedy i tylko wtedy, gdy wszystkie bieguny jego transmitancji leżą w lewej półpłaszczyźnie płaszczyzny liczb zespolonych.

**Granica stabilności** System znajduje się na granicy stabilności wtedy i tylko wtedy :

$$\text{Res}_1 \leq 0, \text{Res}_2 \leq 0, \dots, \text{Res}_m \leq 0,$$

przy czym bieguny transmitancji, dla których zachodzą równości, są co najwyżej jednokrotne.

### 5.2 Kryteria stabilności

**Twierdzenie o znaku współczynników** Jest to warunek konieczny.

Niech  $a_m > 0$ . Jeśli system jest stabilny to

$$a_0 > 0, a_1 > 0, \dots, a_{m-1} > 0$$

**Kryterium Routha-Hurwitza** Jest to warunek wystarczający.

$$\Delta_1 \neq 0, \Delta_2 \neq 0, \dots, \Delta_m \neq 0$$

to żaden z pierwiastków wielomianu  $M(s)$  nie leży na osi  $j\omega$ . Z twierdzenia wynika oczywisty wniosek, że jeśli wszystkie  $\Delta$  są większe od zera to system jest stabilny.

**Kryterium Hurwitza** Niech  $a_m > 0$ . System jest stabilny wtedy i tylko wtedy gdy :

$$\Delta_1 > 0, \Delta_2 > 0, \dots, \Delta_m > 0$$

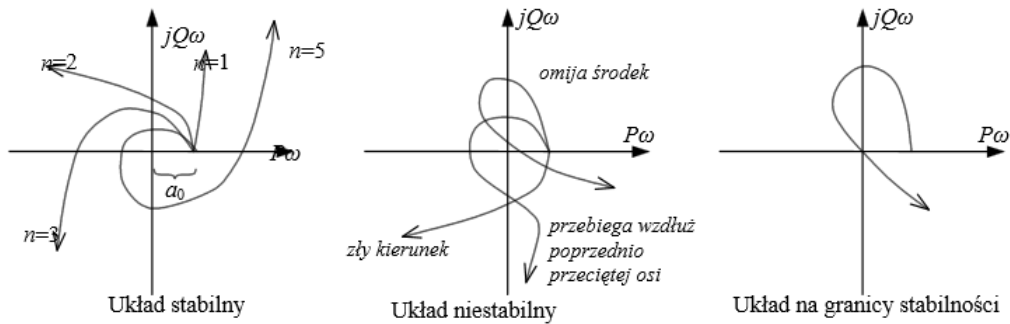
**Warunek konieczny i wystarczający**

**Kryterium Michajłowa** Niech  $a_m > 0$ . System jest stabilny wtedy i tylko wtedy, gdy:

$$M(j\omega) \neq 0 \text{ dla wszystkich } \omega \in [0, \infty)$$

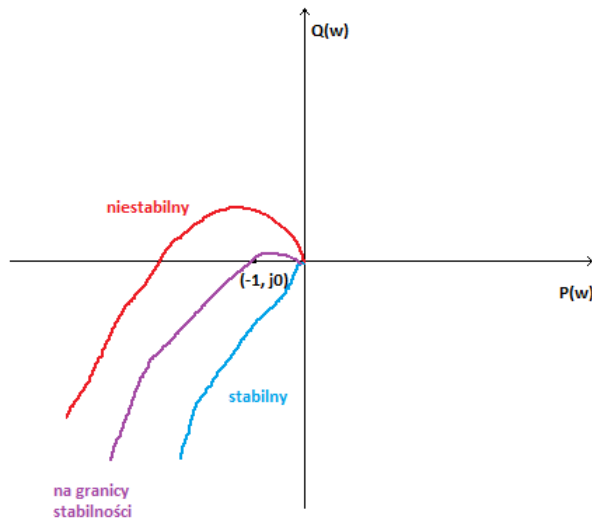
oraz

$$\Delta \arg M(j\omega) = m \frac{\pi}{2}$$



Układ automatycznej regulacji jest stabilny, gdy zmiana argumentu funkcji  $N(j\omega)$  przy zmianie pulsacji  $\omega$  od zera do  $+\infty$  wynosi  $n \frac{\pi}{2}$ , gdzie  $n$  oznacza stopień równania charakterystycznego.

**Kryterium Nyquista** Układ zamknięty jest stabilny, jeżeli logarytmiczna charakterystyka amplitudowa układu otwartego posiada wartość ujemną dla pulsacji odpowiadającej przesunięciu fazowemu  $-\pi$



**DODATEK** Gdyby ktoś potrzebował większego przypomnienia z kryteriów : <http://home.agh.edu.pl/pau-tom/pliki/wyklady/przykladowe/08.pdf> + warto przypomnieć sobie rozwiązywanie równań różniczkowych

## Rozdział 9

# Programowanie w systemie operacyjnym Unix

### 1 Interpreter komend Bourne shell, podstawowe mechanizmy, pisanie skryptów, rozszerzenia POSIX

#### 1.1 Programowanie w języku powłoki

**Powłoka** (ang. shell) jest programem umożliwiającym pracę z systemem UNIX. Jej nazwa wywodzi się z graficznej prezentacji systemu, w której jądro i procesy obsługi wejścia i wyjścia są otoczone właśnie przez powłokę. Jej funkcją jest odseparowanie użytkownika od bezpośredniego dostępu do jądra systemu i umożliwienie łatwej i wszechstronnej pracy. Podstawowym zadaniem powłoki jest przyjmowanie poleceń użytkownika i uruchamianie stosownych programów. Do dalszych funkcji powłoki należy konstruowanie parametrów dla poleceń oraz zarządzanie sposobem wykonywania programów procesów. Następnym, bardziej skomplikowanym krokiem jest tworzenie funkcji i procedur powłoki(skryptów) pozwalających na łączenie wielu poleceń w jedno,umożliwiając pisanie całych programów dla powłoki.

#### Powłoki UNIX

##### Bourne shell (sh)

- nie umożliwia ona operowania na liczbach całkowitych bez tworzenia nowego procesu,
- plikiem wykonywalnym powłoki na większości systemów Unix jest /bin/sh,
- podstawowa powłoka w każdym systemie typu Unix.

##### Bash

W rozwinięciu Bourne-Again Shell (born again shell) - wywodzi się od powłoki Bourne'a sh i był pisany w 1987 r. Jest nadal rozwijany.

- umożliwia wykonanie obliczeń za pomocą wyrażeń w podwójnych nawiasach ((...)) oraz składni \$[...].

##### Korn shell (ksh)

- kompatybilna wstecz z sh,
- zawiera wiele elementów z csh np. historię komend.

## C shell (csh)

- o składni języka C,
- pochodzi od sh,
- ulepszenia: aliasy i historia komend,
- jej następcą jest tcsh,
- rzadko wykorzystywana.

## Polecenia powłoki

Wydanie polecenia w powłoce wiąże się z podaniem ciągu znaków. Składa się z następujących elementów:

**<nazwa polecenia> -<opcje> <parametry>**

Nazwą polecenia jest nazwa jakiegokolwiek programu, funkcji powłoki lub funkcji wewnętrznej powłoki. Opcje to najczęściej zestaw pojedynczych liter określających sposób wykonania programu. Parametry to informacje czego dotyczy wykonanie.

## 1.2 Podstawowe mechanizmy

### Globbering

Mechanizm polegający na dopasowywaniu  $* ? [a - zA - Z]$  do nazw plików.

### Procesy i podprocesy

Wykonywanie programów powoduje utworzenie oddzielnego procesu, który jest podprocesem interpretera poleceń. Proces ma dostęp do standardowych urządzeń wejść i wyjść: stdin, stdout, stderr, oraz środowiska, które jest zestawem zmiennych z przypisanymi wartościami.

### Potoki

Potoki to równoległe uruchomienie dwóch lub więcej poleceń z szeregowym połączeniem ich wejść i wyjść.

### Prawa dostępu

Mechanizm w systemach uniksowych, mający na celu umożliwić określenie uprawnień odczytu, edycji i uruchamiania dla poszczególnych użytkowników. Ma on na celu zapewnienie bezpieczeństwa, stabilności i kontroli prywatności w systemach wielodostępnych. Prawa dostępu przydzielane są dla kategorii: user, group, other. Dla każdej z tych kategorii możliwe są trzy prawa dostępu, opisane literowo (rwx) lub liczbowo trzema bitami: read, write, execute.

### Pisanie skryptów

Skrypty to pliki zawierające zestaw poleceń interpretera. Skrypt można uruchomić jawnie wywołując interpreter poleceń podając nazwę skryptu jako argument.

### rozszerzenia POSIX

POSIX (ang. Portable Operating System Interface for Unix) – przenośny interfejs dla systemu operacyjnego Unix. Jest zbiorem norm, mających zestandaryzować cechy i interfejsy, które powinien mieć i zapewniać system UNIX. Standaryzuje między innymi: operatory arytmetyczne, lokalizacje, klasy mechanizmu globbingu, polecenia zagnieżdżone.

## 1.3 Wyrażenia regularne: sed, awk, grep

### sed

sed (od ang. stream editor) – edytor strumieniowy, służy do przetwarzania tekstu. Edytor sed wczytuje bieżący wiersz pliku do wewnętrznego bufora celem manipulowania tekstem. Wynik jest wysyłany na standardowe wyjście. Oryginalny plik nie jest nigdy zmieniany.

Składnia uruchamiania sed: sed opcje 'polecenie' nazwa\_pliku Wybrane polecenia: wyświetlenie linii, opuszczenie pustych linii, zakończ, skasowanie linii, zapisanie linii do pliku.

### awk

uniwersalny filtr programowalny.

- czyta wiersz z wejścia, dzieli na pola (słowa) dostępne jako:1,2, ...
- wykonuje program składający się z par warunek - akcja.
- par warunek-akcja może być wiele i w każdej może brakować warunku (domyślnie: prawda) albo akcji (domyślnie: wyświetlenie wiersza
- zmienne zachowują wartości pomiędzy wierszami

Jego główną funkcją jest wyszukiwanie i przetwarzanie wzorców w plikach lub strumieniach danych.

### grep

grep – program który służy do wyszukiwania w tekście i wyodrębniania linii zawierających ciąg znaków pasujący do podanego wyrażenia regularnego.

użycie: \$ grep opcje wzorec plik/pliki

przykładowe opcje:

-w – wyszukuje tylko całe słowa;

-x – wyszukuje tylko całe linie;

-c – wyświetla liczbę znalezionych linii;

## 1.4 funkcje I/O niskiego poziomu, deskryptory, przegląd bibliotek systemu UNIX

### funkcje I/O niskiego poziomu

Funkcje open, read, write, zwane wbudowanym systemem we/wy niskiego poziomu, są głównym systemem operacji na plikach i urządzeniach w Unixie.

Odwołują się do otwartych plików przez numery deskryptorów.

### Deskryptory

To małe liczby przyporządkowane kolejno otwieranym plikom. Programy uzyskują najniższy wolny deskryptor. Jeśli inny proces otworzy ten sam plik otrzyma własny (inny lub ten sam) numer deskryptora. Operacja odczytu może przebiegać jednocześnie dla wielu procesów. Jeśli procesy otworzą jednocześnie plik do zapisu, nadpiszą się wzajemnie, konieczna jest synchronizacja w postaci blokad.

### przegląd bibliotek systemu UNIX

- stdio Działa podobnie jak operacje I/O niskiego poziomu
- string
- crypt
- ndbm tworzenie własnych baz danych dla dużej ilości danych i dużej przenoszalności

## 1.5 Programowanie procesów: tworzenie, atrybuty, dziedziczenie atrybutów, współdzielenie zasobów. Sygnały, obsługa sygnałów. Komunikacja przez potoki.

### Programowanie procesów

Procesy mają: kod programu w pamięci, środowisko (zmienne i wartości), zestaw dalszych informacji np. tablicę otwartych plików, priorytet.

Tworzenie procesów poprzez klonowanie istniejącego procesu funkcją `fork()` (procesy potomek i rodzic). Podproces dziedziczy lub współdzieli pewne atrybuty i zasoby procesu nadrzędnego. Proces kończy pracę normalnie (funkcja `exit()` lub koniec `main()`) lub anormalnie (sygnał lub funkcja `abort()`).

Przykładowe atrybuty procesu: PID, PPID, UID, terminal, blokady, maska sygnałów, ograniczenia zasobów.

### Sygnały, obsługa sygnałów.

Sygnały są mechanizmem asynchronicznego powiadamiania procesów o zdarzeniach.

Generuje je:

- błędy i wyjątki sprzętowe
- pewne klawisze terminala
- funkcja `kill`
- mechanizmy softwarowe

Reakcja procesu możliwa: ignorowanie, wywołanie funkcji, śmierć procesu.

Tradycyjna obsługa procesów: zadeklarowanie innej niż domyślna obsługi sygnału ma charakter jednorazowy (zadziała inaczej tylko raz).

### Komunikacja przez potoki

Potok to narzędzie szeregowej, jednokierunkowej komunikacji o własnościach:

- Operacje odczytu i zapisu na potoku jak na zwykłym pliku
- Dostępne jako dwa lub jeden deskryptor pliku (oddzielne dla końca zapisu i odczytu)
- ma ograniczoną pojemność
- jeśli pusty lub ponad pojemność, proces obsługi wisi i kontynuuje gdy to możliwe.

## 1.6 Komunikacja międzyprocesowa przez gniazdko, kolejki komunikatów, i pamięć wspólną, semaforey

### Komunikacja międzyprocesowa przez gniazdko

Gniazdko do deskryptory plików umożliwiające dwukierunkową komunikację w ramach jednego systemu lub przez sieć komputerową. Do korzystania z gniazdek potrzebne jest tworzenie struktury adresowej. Gniazdko posiada również port. Mogą działać jako klient serwer albo broadcast (do wszystkich w sieci).



## kolejki komunikatów POSIX

Mają własności:

- Dwukierunkowe narzędzie komunikacji w jednym z trybów (O\_RDONLY, O\_WRONLY, O\_RDWR)
- stały rozmiar komunikatu
- nie przekazują komunikatu jako strumień bajtów, ale jako całe jednostki
- komunikaty posiadają priorytety

## semafony

Semafor jest zmienną kontrolującą przydział danego zasobu. Semafor wykonuje dwie operacje. P - zajęcie zasobu sygnalizowane zmniejszeniem wartości zmiennej o 1, V - zwolnienie zasobu, zwiększenie zmiennej o 1.

Istnieją semafony system V IPC, oraz POSIX, choć te pierwsze są kłopotliwe w użyciu.

## 1.7 Programowanie w C

Proces kompilacji, pliki obiektowe, pliki biblioteczne, i dynamiczne biblioteczne, linkowanie

### Proces kompilacji

**preprocesor** Przygotowanie do kompilacji, np. wykonywanie dyrektyw preprocesora (`#include`, `#define`), wstawianie definicji i makr.

**Kompilacja i optymalizacja** Proces zmieniający kod języka programowania, na język pośredni kompilatora, a następnie Assemblera. Kompilator sprawdza poprawność kodu źródłowego, i jeżeli nie wystąpią błędy, tworzy plik binarny, np. z rozszerzeniem `.o` lub `.obj`. Dokonywana jest optymalizacja wykonywanych procedur. W trakcie kompilacji kod jest również dostosowywany do wybranej architektury i środowiska.

**Asemlacja** Kod jest tłumaczony na język maszynowy.

**Linkowanie - konsolidacja** Polega na zebraniu wszystkich obiektów do razem i złączeniu ich w wykonywalny program, na przykład dodanie plików nagłówkowych albo referencji. Etap ten to wiązanie ze sobą funkcji, obiektów, itp. Pliki binarne, powstałe po kompilacji kodu źródłowego, łączone są z plikami binarnymi bibliotek: własnych, dostarczonych wraz z kompilatorem, lub innych.

Kompilacja przebiega w kilku etapach i prowadzi do wytworzenia pliku obiektowego.

### Pliki wykonywalne/obektowe, biblioteki dynamiczne

Plik obiektowy składa się z:

- nagłówek  
Nagłówek umieszczany jest zawsze na początku pliku obiektowego w celu ustalenia dalszego sposobu postępowania z tym plikiem.
- kod maszynowy i dane  
Kod i dane binarne przechowują instrukcje zrozumiałe dla procesora oraz dane wykorzystywane przez te instrukcje.
- tablica symboli  
Tablica symboli zawiera między innymi opisy funkcji i zmiennych eksportowanych przez plik obiektowy oraz opisy funkcji i zmiennych, do których ten plik się odwołuje.
- informacje o relokacji  
określają miejsca w kodzie binarnym, które należy zmodyfikować podczas konsolidacji.
- informacje dla programu uruchomieniowego

W systemach Unix formatem pliku obiektowego jest Executable and Linkable Format (ELF), lub przestarzałe a.out.

**Pliki wykonywalne** to plik, który może być uruchomiony bezpośrednio w środowisku systemu operacyjnego. Najczęściej zawiera binarną reprezentację instrukcji konkretnego typu procesora. Plikami wykonywalnymi są skrypty powłoki.

**Biblioteka dynamiczna** rodzaj biblioteki, która łączona jest z programem wykonywalnym dopiero w momencie jego wykonania. Dane z bibliotek dynamicznych mogą być współdzielone przez różne programy jednocześnie. Biblioteki są ładowane do pamięci tylko raz, nawet jeśli są równocześnie współużytkowane.

**Biblioteka statyczna** rodzaj biblioteki funkcji, która łączona jest z programem w momencie konsolidacji

# Rozdział 10

## Komputer, architektura i oprogramowanie

**Komputer** [ang. < łac. computare ‘rozważać’, ‘obliczyć’], elektroniczna maszyna cyfrowa, urządzenie elektroniczne przeznaczone do przetwarzania informacji (danych) przedstawionych w postaci cyfrowej, sterowane programem zapisanym w pamięci. Pojęcie komputera obejmuje obecnie zarówno komputery zaprogramowane na stałe, używane jako automaty sterujące, np. w urządzeniach gospodarstwa domowego, jak i komputery uniwersalne, dające się dowolnie zaprogramować.

### 1 Architektury

#### Model von Neumanna a model współczesnego komputera

John von Neumann opublikował w 1945 propozycję opracowania nowego komputera. Zasadniczą nowością była koncepcja przechowywania programu w pamięci. Komputer miał składać się z czterech bloków funkcjonalnych:

- pamięć przechowująca program do wykonania i dane dla niego;
- jednostka arytmetyczno-logiczna zawierająca rejestry AC (accumulator), MQ (multiplier-quotier), MBR (memory buffer register);
- jednostka sterująca zawierająca licznik programu PC (program counter), rejestr adresowy MAR (memory address register) i pomocnicze rejestry IBR (instruction buffer register), IR (instruction register);
- urządzenia wejścia-wyjścia

Składniki współczesnego komputera:

- (mikro)procesor, zawierający co najmniej jedną jednostkę arytmetyczno-logiczną, jednostkę sterującą i rejestry;
- pamięć operacyjna;
- urządzenia wejścia-wyjścia (klawiatura, mysz, karta graficzna, pamięci dyskowe itp.);
- układ bezpośredniego dostępu do pamięci (ang. DMA – direct memory access);
- układ przerwań.

Urządzenia wejścia-wyjścia same też mogą być skomplikowanymi układami mikroprocesorowymi i zawierać w swoim wnętrzu mikroprocesor, pamięć operacyjną, układy wejścia-wyjścia itd. Układ DMA jest specjalizowanym procesorem odciażającym procesor główny przy operacjach przesyłania dużych bloków danych między pamięcią operacyjną a układami wejścia-wyjścia. Komunikacja pomiędzy poszczególnymi składnikami odbywa się za pomocą szyn (zwanymi też magistralami): danych, adresowych, sygnałów sterujących.

## Architektury typu Princeton i Harvard

Architektura typu Princeton posiada wspólną hierarchię pamięci programu i danych, jak opisano to w modelu von Neumanna. Architektura typu Harvard polega na rozdzieleniu pamięci programu od pamięci danych. Stosowana jest dla zwiększenia wydajności w pamięciach podręcznych oraz w systemach wbudowanych (np. sterownikach urządzeń AGD, gdzie kod programu nie zmienia się przez całe życie urządzenia lub zmienia się rzadko)

**Pamięć operacyjna** Pamięć operacyjna (ang. internal memory, primary storage) – pamięć komputerowa adresowana i dostępna bezpośrednio przez procesor, a nie za pośrednictwem urządzeń wejścia-wyjścia. W pamięci tej mogą być umieszczane rozkazy procesora (program) dostępne bezpośrednio dla jego jednostek wykonawczych.

**Urządzenia wejścia-wyjścia** Urządzenie wejścia-wyjścia, urządzenie we/wy, urządzenie I/O służy do komunikacji systemu komputerowego z jego użytkownikiem lub innym systemem przetwarzania danych. Urządzenie wejścia-wyjścia służy często do zamiany wielkości fizycznych na dane przetwarzane przez system lub odwrotnie. Np. mysz komputerowa przetwarza ruch ręki, odbiornik GPS aktualne położenie geograficzne, a monitor komputera przetwarza dane komputerowe na obraz.

**DMA - Direct Memory Access** DMA jest to metoda, która umożliwia urządzeniom I/O wysłanie oraz odbieranie danych bezpośrednio do lub z pamięci operacyjnej, pomijając CPU w celu przyspieszenia tych operacji oraz odciążenia procesora który uczestniczy w procesie w niewielkim stopniu. Ma on za zadanie odpowiednio programować kontroler DMA który zajmuje się transferem danych.

## System przerwań

**Przerwanie** sygnał powodujący zmianę przepływu sterowania, niezależnie od aktualnie wykonywanego programu. Pojawienie się przerwania powoduje wstrzymanie aktualnie wykonywanego programu i wykonanie przez procesor kodu procedury obsługi przerwania (ang. interrupt handler). Procedura ta wykonuje czynności związane z obsługą przerwania i na końcu wydaje instrukcję powrotu z przerwania, która powoduje powrót do programu realizowanego przed przerwaniem.

- Przerwania sprzętowe (maskowalne, niemaskowalne)
- Przerwania programowe
- Praca krokowa
- Wyjątki
- Tablica przerwań

## 1.1 Oprogramowanie

**Oprogramowanie** (ang. software) – całość informacji w postaci zestawu instrukcji, zaimplementowanych interfejsów i zintegrowanych danych przeznaczonych dla komputera do realizacji wyznaczonych celów. Celem oprogramowania jest przetwarzanie danych w określonym przez twórcę zakresie.

### Rodzaje oprogramowania

- oprogramowanie systemowe - realizujące funkcje konieczne dla działania systemu
- oprogramowanie do tworzenia oprogramowania
- biblioteki programistyczne
- oprogramowanie użytkowe

## Część II

# Zagadnienia kierunkowe

## Rozdział 11

# Programowanie strukturalne i obiektowe

**Programowanie strukturalne** – jest to „podparadygmat” programowania proceduralnego. Opiera się na tworzeniu programów z kilku dobrze zdefiniowanych funkcji takich jak instrukcja warunkowa if-then-else i pętle, za to bez skoków (go to). Proponowane jest używanie tylko trzech struktur sterujących:

- **Sekwencja lub konkatencja** – wykonywanie instrukcji w określonej kolejności.
- **Wybór** – wykonywanie jednej z kilku instrukcji zależnie od stanu programu. Przykładem jest if-then-else i switch/case.
- **Iteracja** - przetwarzanie instrukcji tak długo, jak długo spełniony (lub niespełniony) jest dany warunek. Np. while, for. Te struktury stosuje się do „małych” bloków programu złożonych z elementarnych instrukcji tj. podstawień, wywołań procedur/funkcji, instrukcji IO itd... Duże bloki powinny być rozbite na mniejsze (funkcje, procedury) tak aby rozumieć poszczególne fragmenty bez rozumienia całości. Podział ten ma również wpływ na jakość kodu, ponieważ procedura/funkcja może być używana wielokrotnie bez niepotrzebnego powielania kodu.

**Programowanie obiektywne** – W programowaniu obiektywnym program to zbiór porozumiewających się ze sobą obiektów, czyli jednostek zawierających określone dane i umiejących wykonywać na nich określone operacje. Najważniejsze są tu dwie cechy: po pierwsze, powiązanie danych (czyli stanu) z operacjami na nich (czyli poleceniami) w całość, stanowiącą odrębną jednostkę — obiekt; Programowanie obiektywne ułatwia, pisanie, konserwację, testowanie i wielokrotne użycie programów lub ich fragmentów.

Podstawowe założenia paradygmatu obiektywego:

- **Abstrakcja** – każdy obiekt w systemie służy jako model abstrakcyjnego „wykonawcy” który może wykonywać pracę (metody), opisywać swój stan, zmieniać swój stan oraz komunikować się z innymi obiektami bez ujawniania jego implementacji.
- **Hermetyzacja** – ukrywanie implementacji, enkapsulacja. Zapewnia, że obiekt nie może zmienić stanu wewnętrznego innych obiektów w nieoczekiwany sposób. Każdy obiekt prezentuje innym obiektom swój interfejs który określa dopuszczalne metody współpracy.
- **Polimorfizm** – wielopostaciowość, referencje mogą dotyczyć obiektów różnego typu. Wywołanie metody dla referencji spowoduje zachowanie odpowiednie dla typu obiektu danej referencji. Jeśli dzieje się to w trakcie wykonywania programu to nazywa się to wiązaniem dynamicznym.
- **Dziedziczenie** – porządkuje i wspomaga polimorfizm. Umożliwia definiowanie i tworzenia specjalizowanych obiektów na podstawie bardziej ogólnych (od ogółu do szczegółu). Dzięki temu nie powiela się kodu oraz można redefiniować poszczególne parametry bez redefiniowania parametrów ogólnych.

Języki wspierające programowanie obiektywne m.in.: C#, Java, C++, JavaScript, Objective-C

Źródło : materiały PW

## Rozdział 12

# Fala elektromagnetyczna: typy, parametry, właściwości

**Pole elektryczne** – każdy ładunek wytwarza w przestrzeni wokół siebie pole elektryczne, siła elektrostatyczna działająca na dowolny ładunek jest wywołana polem elektrycznym, wytworzonym przez inne ładunki w miejscu w którym znajduje się rozważany ładunek.

**Pole magnetyczne** – wytwarzany przez magnes, jest to pole wektorowe tak jak w przypadku pola elektrycznego. W elektromagnesie pole magnetyczne wytwarzane jest za pomocą przepływu prądu przez cewkę nawiniętą na metalowy rdzeń z żelaza. O sile pola decyduje wartość natężenia prądu. Magnesy trwale zawierają cząstki elementarne (elektrony) które wytwarzają własne pole magnetyczne. W niektórych materiałach pola magnetyczne elektronów sumują się wytwarzając wokół nich wypadkowe pole magnetyczne.

**Fala elektromagnetyczna** – jest to rozchodząca się z prędkością  $c$  fala pola elektrycznego i magnetycznego. Fale elektromagnetyczne można podzielić na:

- Fale stojące (punkty o jednakowej fazie nie przemieszczają się) np. wnęka rezonansowa
- Bieżące (punkty o jednakowej fazie poruszają się) – rozchodzące się wzdłuż linii przesyłowej lub wolnej przestrzeni

### Cechy

- Wektory  $\vec{E}$  i  $\vec{B}$  są zawsze prostopadłe do kierunku rozchodzenia się fali. Zatem fale elektromagnetyczne są falami poprzecznymi.
- Iloczyn wektorowy  $\vec{E} \times \vec{B}$  zawsze wyznacza kierunek rozchodzenia się fali.
- Natężenie pola elektrycznego i indukcja pola magnetycznego zmieniają się zawsze sinusoidalnie. Wektory pól zmieniają się z taką samą częstotliwością a ich oscylacje są zgodne w fazie.

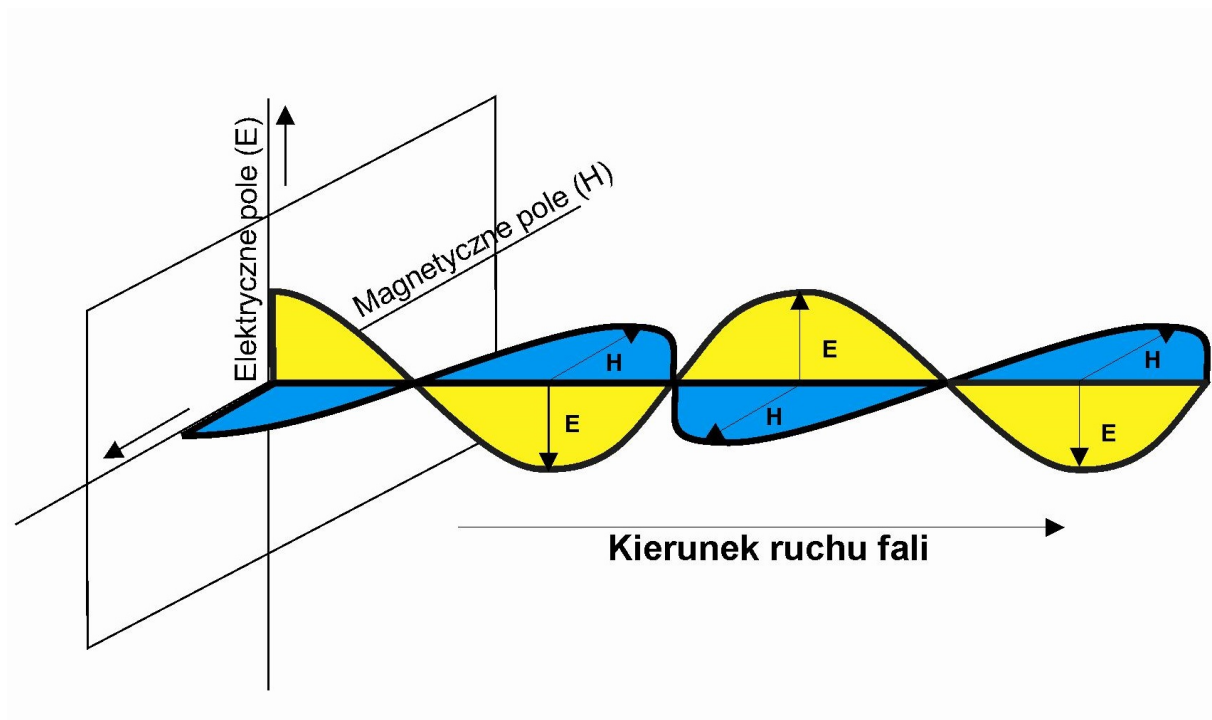
Rozważając powyższe cechy, przyjmując, że fala rozchodzi się w dodatnim kierunku osi OX. Wektor natężenia pola elektrycznego wykonuje oscylacje równoległe do osi OY a wektor indukcji równoległe do osi OZ (w prawoskrętnym układzie współrzędnych). Można zapisać następujące równania:

$$E(x, t) = E_m \cdot \sin(kx - \omega t) - \text{składowa elektryczna}, \quad (12.1)$$

$$B(x, t) = B_m \cdot \sin(kx - \omega t) - \text{składowa magnetyczna}, \quad (12.2)$$

gdzie:

$E_m$  i  $B_m$  - amplitudy fali pola elektrycznego i magnetycznego,  
 $\omega$  - częstotliwość kątowa wyrażona przez wektor falowy ( $\omega = ck$ ),  
 $k$  - stała propagacji ( $k = 2\pi/\lambda$ ),  
 $\lambda$  - długość fali ( $\lambda = cT$ ),



Rysunek 12.1: Fala elektromagnetyczna

$T$  - okres drgań.

Prędkość fali elektromagnetycznej:

$$c = \frac{\omega}{k} = \frac{1}{\sqrt{\mu_0 \epsilon_0}} = 3.0 \cdot 10^8 \frac{m}{s}, \quad (12.3)$$

gdzie:

$\mu_0$  - przenikalność magnetyczna próżni ( $4\pi \cdot 10^{-7} \text{ H/m}$ ),

$\epsilon_0$  - przenikalność elektryczna próżni ( $8.854 \cdot 10^{-12} \text{ F/m}$ ).

**Energia fali elektromagnetycznej** – energia pojedynczego kwantu jest zależna tylko od częstotliwości fali  $f$  i wynosi:

$$E = hf, \quad (12.4)$$

gdzie:

$h$  - stała Plancka ( $6.626 \cdot 10^{-34} \text{ J} \cdot \text{s}$ ).

**Wektor Poyntinga** – strumień energii przenoszonej przez falę elektromagnetyczną w każdym punkcie przestrzeni określa wektor Poyntinga zdefiniowany, jako:

$$\vec{S} = \frac{1}{\mu_0} \vec{E} \times \vec{B}. \quad (12.5)$$

**Pęd i ciśnienie fali elektromagnetycznej** – biegnąca fala elektromagnetyczna niesie ze sobą pęd równy:

$$\vec{p} = \frac{W}{c} \hat{k}, \quad (12.6)$$

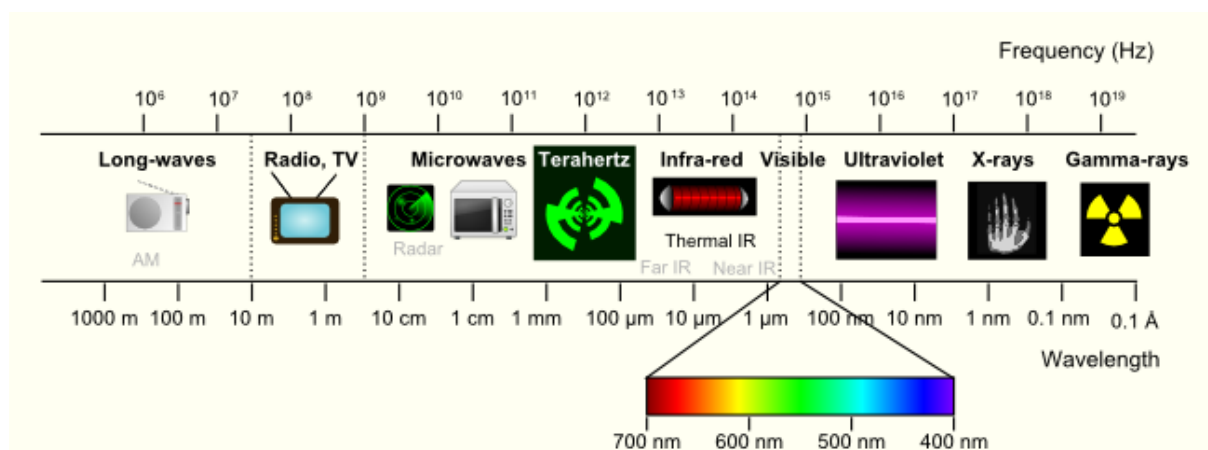
gdzie:

$W$  - energia niesiona przez falę,

$c$  - prędkość światła,

$\hat{k}$  - wektor jednostkowy w kierunku rozchodzenia się fali.





Rysunek 12.2: Widmo promieniowania elektromagnetycznego

## Rozdział 13

# Tranzystory bipolarne i unipolarne: budowa, właściwości i zastosowania

### 0.1 Tranzystory bipolarne

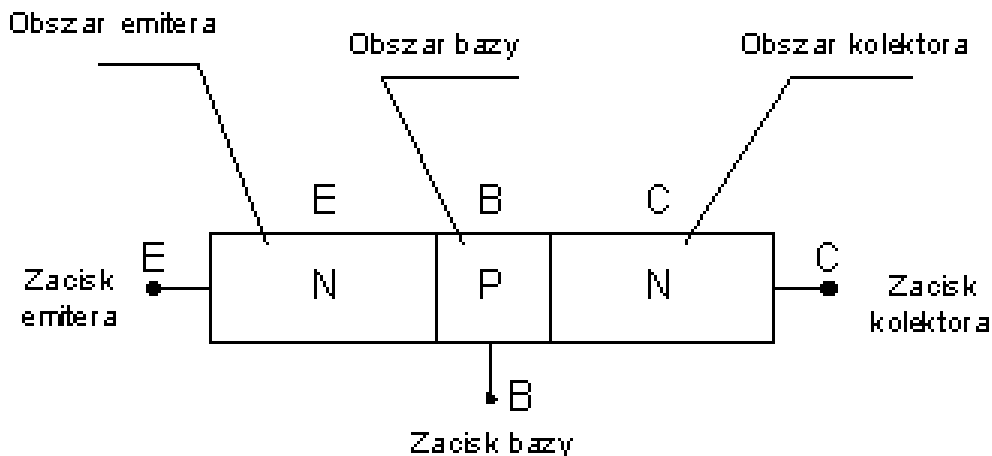
**Tranzystor bipolarny** – półprzewodnikowy element elektroniczny o trzech elektrodach. Jest to najważniejszy przykład elementu aktywnego, czyli urządzenia, które może wytwarzać na wyjściu sygnał o mocy większej niż moc sygnału wejściowego. Służy również do przełączania sygnału.

#### Budowa

Tranzystor bipolarny składa się z trzech obszarów półprzewodnika o przeciwnym typie przewodnictwa, co powoduje powstanie dwóch złączy: p-n i n-p.

**p** – półprzewodnik niedomiarowy gdyż przeważają nośniki typu dziurowego

**n** – półprzewodnik nadmiarowy gdyż przeważają nośniki typu elektronowego



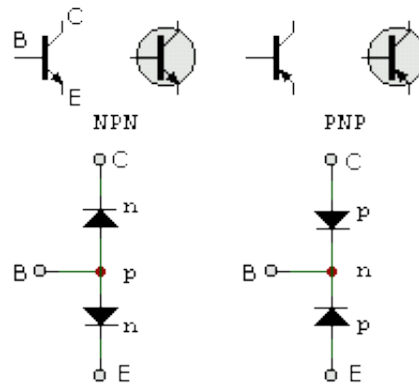
Rysunek 13.1: Budowa tranzystora bipolarnego (NPN)

Diodowy schemat zastępczy nie odzwierciedla w pełni jego działania, daje jednak pewien pogląd na napięcia występujące między elektrodami.

Główną cechą charakterystyczną tranzystora jest to, że prąd kolektora  $I_C$  jest proporcjonalny do prądu bazy  $I_B$ . Stosunek:

$$\beta = \frac{I_C}{I_B}$$

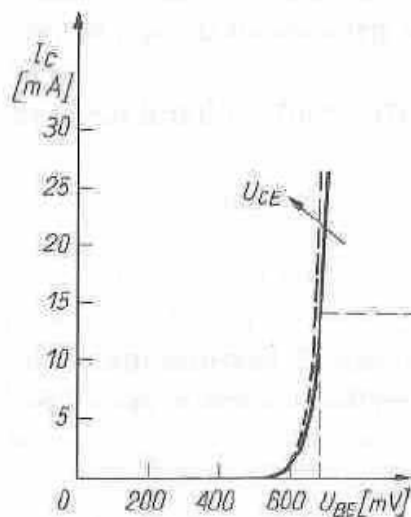
wielkosygnalowym współczynnikiem wzmocnienia prądowego.



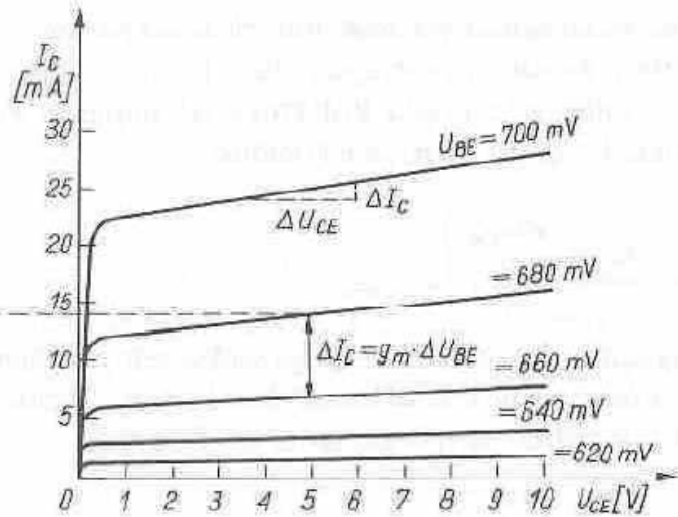
Rysunek 13.2: Diodowe schematy zastępcze tranzystorów bipolarnych

Dalsze rozważania dotyczą tranzystorów typu pnp, w przypadku npn należy zmienić znak wszystkich prądów i napięć na przeciwny.

Po doprowadzeniu  $U_{BE}$  i dokonaniu pomiaru  $I_C$  w funkcji napięcia wyjściowego  $U_{CE}$ . Stopniowe zwiększanie napięcia wejściowego daje charakterystykę wyjściową.



Rys. 4.5. Charakterystyka przejściowa



Rys. 4.6. Charakterystyka wyjściowa

Rysunek 13.3: Charakterystyki: przejściowa i wyjściowa tranzystora bipolarnego

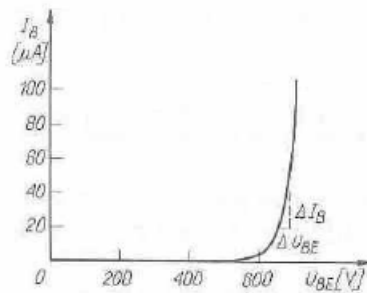
Na podstawie charakterystyki wyjściowej można zauważyć, że powyżej pewnego napięcia prąd kolektora prawie nie zależy od  $U_{CE}$ . Napięcie, przy którym następuje zagięcie charakterystyk nosi nazwę napięcia nasycenia kolektor-emiter  $U_{CEsat}$ . Drugą ważną właściwością jest fakt, że do wywołania względnie dużej zmiany prądu kolektora wystarczy niewielka zmiana napięcia wejściowego. Ta zmiana, czyli odstęp między charakterystykami, silnie rośnie przy zwiększaniu prądu kolektora. Zmianę tą można dobrze zauważyć na charakterystyce przejściowej z rysunku 13.3.

Tranzystor w przeciwieństwie do lampy elektronowej nie da się sterować bezprądowo, widać to na powyższej charakterystyce. Wzmocnienie prądowe nie jest stałe, lecz zależy od prądu kolektora.

Prąd kolektora w pierwszym przybliżeniu jest proporcjonalny do prądu bazy. Widać to na rysunku 4.8. Stosunek  $I_C$  do  $I_B$  nazywa się statycznym współczynnikiem wzmocnienia prądowego.

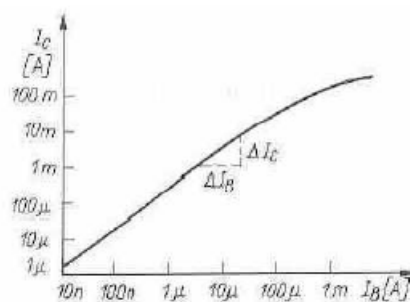
Podsumowując, właściwości tranzystorów bipolarnych:

- prąd kolektora jest proporcjonalny do prądu bazy
- powyżej pewnego napięcia prąd kolektora prawie nie zależy od  $U_{CE}$



Rys. 4.7. Charakterystyka wejściowa

Rysunek 13.4: Charakterystyka wejściowa tranzystora bipolarnego



Rys. 4.8. Wykres typowej zależności prądu kolektora od prądu bazy dla tranzystora małej mocy

- do wywołania względnie dużej zmiany prądu kolektora wystarczy niewielka zmiana napięcia wejściowego
- tranzystorów bipolarnych nie da się sterować bezprądowo
- wzmacnienie prądowe nie jest stałe, lecz zależy od prądu kolektora.

#### Zastosowania:

**Wzmacniacz** – tranzystor pracujący w stanie aktywnym może być wykorzystany do budowy układu będącego wzmacniaczem natężenia prądu elektrycznego. Małe zmiany prądu elektrycznego płynącego w obwodzie bazy powodują duże zmiany prądu płynącego w obwodzie kolektora. W zależności od konstrukcji układu można uzyskać wzmacnienie prądu, napięcia lub obu tych wielkości.

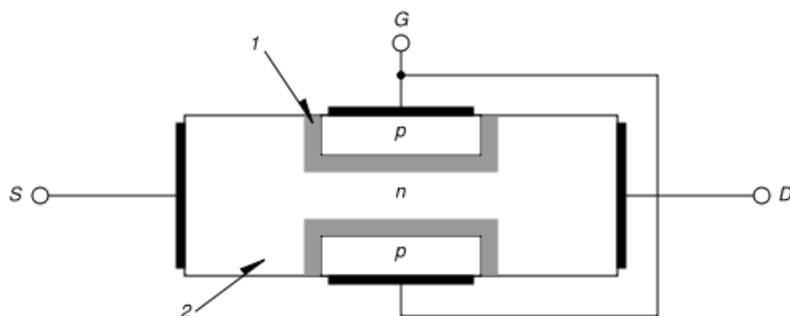
**Przełącznik** – przy pracy tranzystora jako przełącznik wykorzystuje się przejście między stanem nasyconym (tranzystor włączony) a zatkany (tranzystor wyłączony). Taki tryb pracy tranzystora jest stosowany w niektórych układach impulsowych oraz cyfrowych.

## 0.2 Tranzystory unipolarne (polowe)

**Tranzystory unipolarne (polowe)** – są elementami półprzewodnikowymi, trójelektrodowymi, które w przeciwieństwie do normalnych tranzystorów bipolarnych są sterowane polem elektrycznym tzn. bez poboru mocy. Działanie tranzystorów polowych opiera się na sterowaniu przepływem prądu przez kanał za pomocą pola elektrycznego wytwarzanego przez napięcie doprowadzane do elektrody nazywanej bramką. Nie ma tu żadnych przewodzących złącz, więc do bramki nie wpływa ani z niej nie wypływa prąd. Podobnie jak w przypadku tranzystorów bipolarnych, istnieją tranzystory polowe o dwóch różnych rodzajach przewodnictwa: z kanałem typu n (przewodnictwo elektronowe) oraz z kanałem typu p (przewodnictwo dziurowe). Tranzystory polowe mogą być wykonane z dwoma różnymi rodzajami bramek (mamy więc tranzystory złączowe i z izolowaną bramką) oraz mogą różnić się sposobem domieszkiowania materiału półprzewodnikowego tworzącego kanał (tranzystory ze zubożaniem i ze wzbogacaniem kanału).

## Budowa JFET ( tranzystora polowego złączowego)

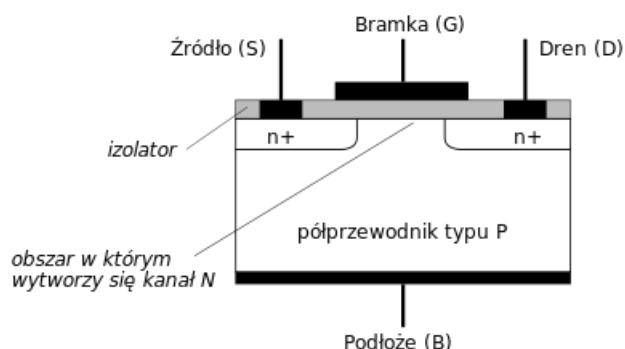
Budowa tranzystora JFET została przedstawiona na rysunku 13.5.



Rysunek 13.5: Budowa JFET

## Budowa MOSFET (tranzystora polowego z izolowaną bramką)

Najważniejszą cechą tranzystora polowego jest brak prądu bramki. Wynikająca z tego duża wartość rezystancji wejściowej, przekraczająca  $10^{14} \Omega$  jest podstawą wielu zastosowań. Budowa MOSFET'a została ukazana na rysunku 13.6.



Rysunek 13.6: Budowa MOSFET

## Rodzaje tranzystorów polowych

Klasyfikacja pod względem polaryzacji napięć wejściowych i wyjściowych (ze źródłem dołączonym do masy).



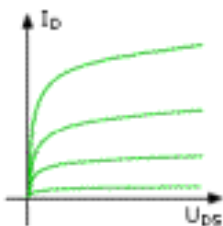
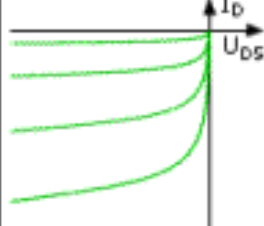
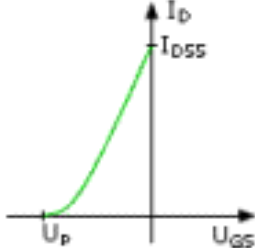
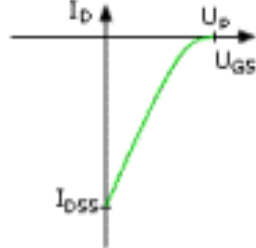
## Właściwości tranzystorów polowych

Jak wynika z powyższych rysunków w normalnym użyciu znajdują się pięć typów tranzystorów polowych. Jednakże, niekoniecznie trzeba pamiętać właściwości każdego z nich gdyż wszystkie zachowują się podobnie.

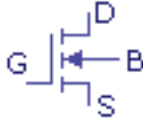
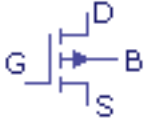
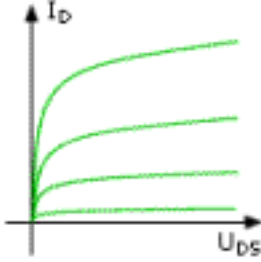
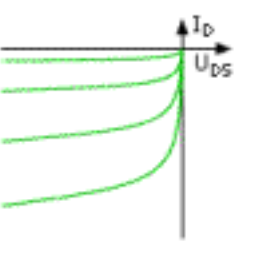
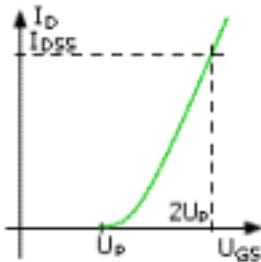
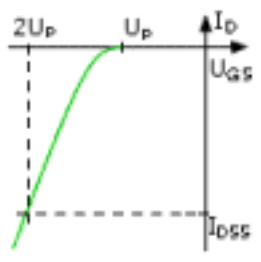
Po pierwsze, dla źródła dołączonego do masy, każdy tranzystor polowy jest wprowadzany w stan przewodzenia („włączany”) przez taką zmianę napięcia bramki, aby wartość tego napięcia dążyła do wartości napięcia zasilającego dren w warunkach aktywnej pracy tranzystora.

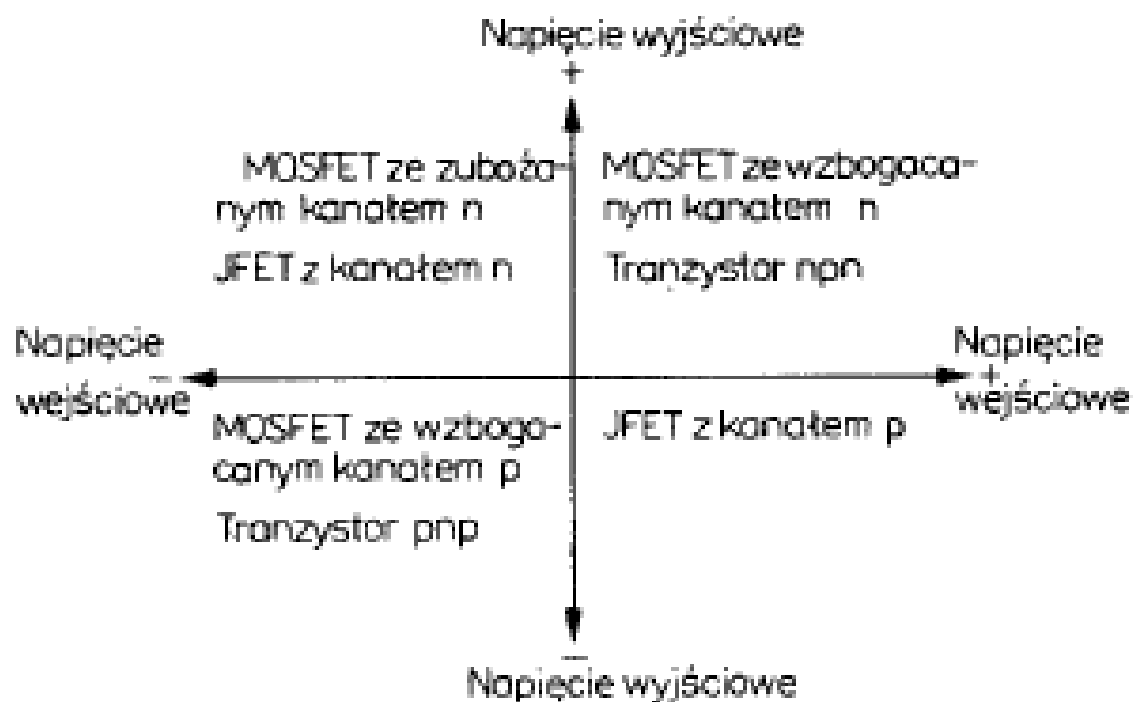
Po drugie, ze względu na prawie symetryczną konstrukcję tranzystora polowego, zarówno dren jak i źródło mogą pracować, jako rzeczywiste źródło (wyjątek stanowią tranzystory MOS mocy, w których podłoże jest połączone ze źródłem wewnątrz obudowy).

Podobnie jak tranzystor n-p-n, tranzystor polowy charakteryzuje się dużą przyrostową wartością impedancji obwodu drenu. Przejawia się to stałą wartością prądu drenu, niezależną od wartości napięcia  $U_{DS}$  jeśli tylko wartość tego napięcia jest większa od powiedzmy, 1 V. Im większa jest wartość napięcia

|   |  |
|---|--|
| złączowe  |  |
| kanał typu n  | kanał typu p   |
|    |     |
|   |   |
|  |  |
| Wzmacniacze zbudowane z elementów dyskretnych. Analogowe układy scalone.            | Wzmacniacze zbudowane z elementów dyskretnych. Analogowe układy scalone.             |

|  |  |
|--|--|
| z izolowaną bramką   |  |
| z kanałem zubożanym  |  |
| kanał typu n   | kanał typu p   |
|  |  |
|  |  |
|  |  |
| Wzmacniacze w.cz. zbudowane z elementów dyskretnych. Cyfrowe układy scalone. | Wzmacniacze w.cz. zbudowane z elementów dyskretnych. Cyfrowe układy scalone. |

| z kanałem wzbogaczonym   |   |
|--|---|
| kanał typu n   | kanał typu p  |
|   |    |
|   |   |
|  |  |
| Wzmacniacze mocy zbudowane z elementów dyskretnych. Cyfrowe układy scalone.        | Wzmacniacze mocy zbudowane z elementów dyskretnych. Cyfrowe układy scalone.         |

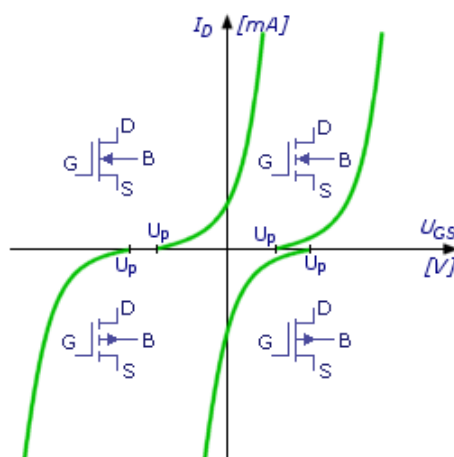




między bramką a źródłem tym większa jest wartość prądu drenu. W obszarze nasycenia, czyli w normalnym obszarze pracy FET-a prąd drenu nie jest szybkozmienna funkcją napięcia  $U_{GS}$ . Prąd drenu jest proporcjonalny do  $(U_{GS} - U_T)^2$ , gdzie  $U_T$  jest napięciem progowym czyli napięciem  $U_{GS}$  dla którego zaczyna płynąć prąd drenu.

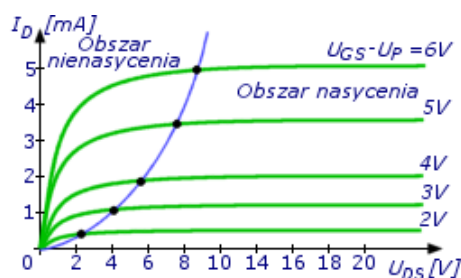
### Podstawowe charakterystyki:

**Przejęciowa** – zależność prądu drenu ( $I_D$ ) od napięcia bramka-źródło ( $U_{GS}$ ) przy stałym napięciu dren-źródło ( $U_{DS}$ ).



Rysunek 13.7: Charakterystyka przejściowa tranzystorów unipolarnych

**Wyjęciowa** – zależność prądu drenu ( $I_D$ ) od napięcia dren-źródło ( $U_{DS}$ ), przy stałym napięciu bramka-źródło ( $U_{GS}$ ). Cały obszar charakterystyki wyjściowej można podzielić na dwie części: obszar nasycenia i obszar nienasycenia (liniowy). Na rysunku 13.8 obszary te są rozdzielone niebieską linią, której kształt przypomina parabolę.



Rysunek 13.8: Charakterystyka wyjściowa tranzystora unipolarnego

W zakresie liniowym (nienasycenia) tranzystor unipolarny zachowuje się jak rezystor półprzewodnikowy. Prąd  $I_D$  ze wzrostem napięcia  $U_{DS}$  wzrasta w przybliżeniu liniowo. W zakresie nasycenia napięcie  $U_{DS}$  bardzo nieznacznie wpływa na prąd drenu, natomiast bramka zachowuje właściwości sterujące.

## Rozdział 14

# Systemy ciągłe i dyskretne: klasyfikacja, opis

**System** można rozumieć jako pewien blok, który odpowiada na zadany sygnał wejściowy (pobudzenie) jednym lub więcej sygnałami wyjściowym.



**Systemy ciągłe** – wszystkie sygnały (wejściowe i wyjściowe) są funkcjami ciągłymi w dziedzinie  $x$  (np. czasie) i mogą przybierać dowolną wartość z obszaru swojej zmienności. Układy te opisuje się zwykle równaniami różniczkowymi.

**Systemy dyskretne** – układ jest dyskretny, jeżeli przynajmniej jeden jego sygnał ma charakter dyskretny, tzn. przyjmuje tylko określone wartości dla określonych argumentów. Układy takie opisuje się zwykle równaniami różnicowymi.

W systemie dyskretnym, przetwarzanie obejmuje operacje arytmetyczne przeprowadzane na sygnale wejściowym  $x[n]$ , w wyniku, których na wyjściu systemu otrzymuje się sygnał wyjściowy  $y[n]$  w postaci ciągu liczb. W większości przypadków systemy czasu dyskretnego są systemami o jednym wejściu i jednym wyjściu.

Systemy można klasyfikować ze względu na następujące własności:

- **Liniowość** – spełnia zasadę superpozycji tj. odpowiedź systemu liniowego na sumę sygnałów wejściowych jest równa sumie odpowiedzi systemu na poszczególne sygnały składowe.
- **Przyczynowość** – wyjścia zależą od wejść bieżących i przeszłych, ale nie od wejść przyszłych. Układ taki nie wykazuje reakcji, nim nie nastąpi jego pobudzenie.
- **Stacjonarność** – jeżeli dowolne przesunięcie czasu  $q$  dla sygnału wejścia  $u(t+q)$  powoduje takie samo przesunięcie dla sygnału wyjścia  $y(t+q)$ .

### Właściwości systemów

- Addytywność

$$f(x_1(t) + x_2(t)) = f(x_1(t)) + f(x_2(t))$$

$$f(x_1(n) + x_2(n)) = f(x_1(n)) + f(x_2(n))$$

- Jednorodność

$$f(kx(t)) = kf(x(t))$$

$$f(kx(n)) = kf(x(n))$$

- Liniowość

$$f(k_1x_1(t) + k_2x_2(t)) = k_1f_1(x_1(t)) + k_2f_2(x_2(t))$$

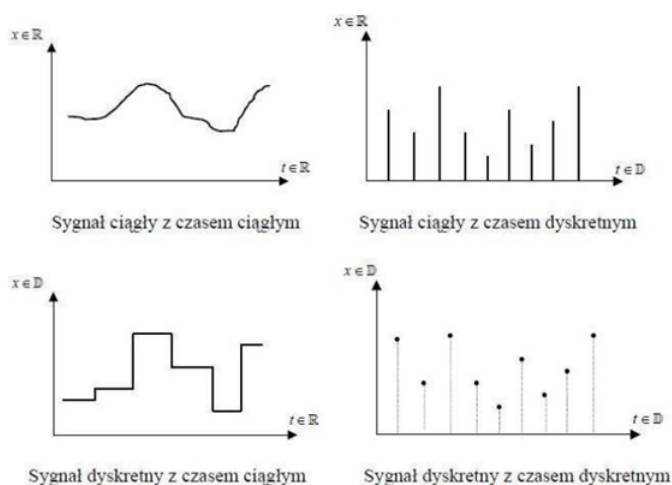
$$f(k_1x_1(n) + k_2x_2(n)) = k_1f_1(x_1(n)) + k_2f_2(x_2(n))$$

## 0.1 Sygnały

**Sygnał** – proces zmian pewnej wielkości fizycznej lub stanu obiektu fizycznego w czasie lub w przestrzeni. Sygnał generalnie przekazuje jakąś informację (tj. jest nośnikiem informacji). Sygnał może być również syntetyzowany do celów komunikacji.

**Klasyfikacja ze względu na:**

- **Dziedzinę** – sygnały ciągłe (określone dla wszystkich  $x \in [a, b]$ ) i dyskretne określone dla wybranych punktów. Poza tymi punktami sygnały są nieokreślone.
- **Przeciwdziedzinę** (zbiór wartości funkcji) Zbiór ten może być ciągły (sygnał ciągły w amplitudzie) lub dyskretny (albo skończony gdy liczba wartości przyjmowanych przez funkcję jest równa  $N$ ).



Rysunek 14.1: Wykresy sygnałów

Sygnał dyskretny w czasie powstaje w wyniku próbkowania sygnałów ciągłych z określonym krokiem próbkowania.

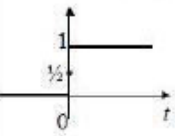
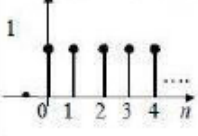
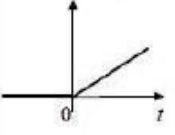
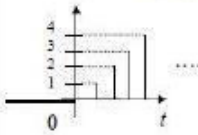
## 0.2 Dodatek

Tematyka systemów jest bardzo rozbudowana – stąd dodatek.

Systemy mogą również być:

- **Dynamiczne** system dynamiczny to taki, w którym zmiana w jednej części wpływa na pozostałe; największy dynamiczny system fizyczny to Wszechświat.
- **Statyczne** system statyczny jest niezmienny w czasie, może być abstrakcyjny lub fizyczny,

TABLICA 4. PODSTAWOWE SYGNAŁY CZASU CIĄGŁEGO I DYSKRETNEGO

| Sygnały ciągłe   | Sygnały dyskretne   |
|--|---|
| Delta Diraca $\delta(t)$   | Delta Kroneckera $\delta[n]$ – impuls jednostkowy<br>$\delta[n] = \begin{cases} 1 & \text{dla } n = 0 \\ 0 & \text{dla } n \neq 0 \end{cases}$ $\delta[n] = 1[n] - 1[n-1]$                                  |
| Skok jednostkowy $1(t)$<br> $1(t) = \begin{cases} 1 & \text{dla } t > 0 \\ 1/2 & \text{dla } t = 0 \\ 0 & \text{dla } t < 0 \end{cases}$                          | Dyskretny skok jednostkowy $1[n]$<br> $1[n] = \begin{cases} 1 & \text{dla } n \geq 0 \\ 0 & \text{dla } n < 0 \end{cases}$ |
| Rzeczywisty sygnał wykładniczy<br>$x(t) = e^{at}, a \in \mathbb{R}$  | Dyskretny sygnał wykładniczy<br>$x[n] = a^n, a \in \mathbb{R}$<br>$a \in (0,1)$ - wykładniczy malejący<br>$a > 1$ - wykładniczy rosnący   |
| Sygnał rampowy $r(t)$<br> $r(t) = \begin{cases} t & \text{dla } t \geq 0 \\ 0 & \text{dla } t < 0 \end{cases}$ $r(t) = t \cdot 1(t)$ $1(t) = \frac{d}{dt} r(t)$ | Dyskretny sygnał rampowy $r[n]$<br> $r[n] = \begin{cases} n & \text{dla } n \geq 0 \\ 0 & \text{dla } n < 0 \end{cases}$ |

Praktyczne wskazówki:

Na niestacjonarność wskazują

- jakiegokolwiek niejednostkowe stałe związane z argumentem czasu np.  $u(2t)$ ,  $u(-t)$ ,  $u[2n]$ ,  $u[-n]$
- jakiegokolwiek współczynniki będące funkcjami czasu w równaniu różniczkowym lub różnicowym

Przykłady:

Systemy ciągłe:

Stacjonarne

$$\frac{d}{dt} y(t) = \sin(u(t))$$

Niestacjonarne

$$\frac{d}{dt} y(t) = (\sin(t) - 1)y(t) + u(t)$$

Systemy dyskretne:

$$y[n] = \frac{1}{3} (u[n-1] + u[n] + u[n+1])$$

$$y[n] = \frac{1}{n} y[n-1] + u[n]$$

**Systemy nieprzyczynowe** - zwane wyprzedzającymi, to takie, w których wartość sygnału wyjściowego w badanej chwili zależy także od przyszłych wartości sygnału na wejściu. Przykładami takich systemów są:

- systemy, w których zmienną niezależną nie jest czas (np. systemy cyfrowego przetwarzania obrazów),
- systemy w których uśredniamy dane zebrane w pewnym okresie czasu (ceny akcji na giełdzie, dane demograficzne, sygnały meteorologiczne), i w których interesuje nas określenie wolnozmiennych trendów w danych, zawierających także szybkozmiennie (często przypadkowe) fluktuacje.

**System jest odwracalny** , jeżeli jest możliwe znalezienie takiego systemu, który włączony z nim kaskadowo da na wyjściu sygnał wejściowy.

### Rozszerzone definicje

- **Sygnał ciągły** – wartości  $f(x)$  sygnału są istotne dla każdego argumentu  $x$  z pewnego przedziału, np. dla każdej chwili czasu  $t$
- **Sygnał dyskretny** – określony dla dyskretnych wartości argumentu  $x$  – gdzie znana jest wartość ilości argumentów  $x$
- **Sygnał analogowy** – jest przebiegiem konkretnej wielkości fizycznej, np. napięcia, natężenie prądu, ciśnienia, temperatury etc. dla konkretnej lub nieskończonej dziedziny argumentu  $x$  (np. w czasie).
- **Sygnał cyfrowy** – sygnał określony dla dyskretnych wartości  $x$  i  $f(x)$ . Znaczenie tego terminu może odnosić się do:
  - wielkości fizycznej, która z natury jest dyskretna (np. liczba błysków lampy w ciągu godziny)
  - wielkości pierwotnie ciągłej i analogowej, która została spróbkowana i skwantowana (np. sygnał na wyjściu komparatora napięcia kontrolującego pewien proces w określonych chwilach)
  - każdej reprezentacji jednego z powyższych, w tym (najczęściej) w postaci ciągu liczb zapisanych w pamięci maszyny cyfrowej (np. plik komputerowy typu WAV).

# Rozdział 15

## Zmienna losowa: właściwości, opis

### 0.1 Zmienne losowe

**Zmienna losowa**  $X$  to taka funkcja  $X : \Omega \rightarrow \mathbb{R}$ , dla którego dla dowolnego borelowskiego zbioru  $B \subset \mathbb{R}$  zbiór:

$$\{\omega : X(\omega) \in B\} = \{X \in B\} \in \mathcal{F}, \quad (15.1)$$

tzn. zbiór  $X \in B$  jest zdarzeniem losowym.

Innymi słowy, jest to taka funkcja  $X$  na zbiorze zdarzeń elementarnych o wartościach liczbowych, dla której określone są (teoretycznie) prawdopodobieństwa przyjmowania przez  $X$  wartości z każdego dowolnego zakresu.

Zmienna losowa to funkcja przekształcająca wynik eksperymentu losowego na liczbę rzeczywistą.

Pojęcie zmiennej losowej jest bardzo użyteczne, pozwala na abstrahowanie od postaci przestrzeni zdarzeń a operowanie wyłącznie na liczbach.

Wyobraźmy sobie rzut kostką sześciocienną. Jest to przykład eksperymentu losowego, czyli eksperymentu, dla którego wiemy jakie sytuacje mogą się wydarzyć, ale nie wiemy która się wydarzy. Możliwe wyniki eksperymentu są najróżniejsze, np. kostka będzie turlała się przez 2 minuty i wypadnie sześć oczek, kostka spadnie ze stołu i wypadnie jedno oczko itp.

Zmienna losowa to funkcja przedstawiająca wynik eksperymentu w postaci liczby rzeczywistej. Rozważając eksperyment rzutu kostką nie jest dla nas istotne co działo się z kostką, istotne jest jedynie ile oczek wypadło. Dlatego naturalnym wyborem zmiennej losowej jest funkcja określająca liczbę oczek które wypadły w wyniku rzutu kostką.

Typy zmiennych losowych:

- **ciągła** - zmienna przyjmuje dowolne wartości z określonego przedziału (w szczególności cały zbiór liczb rzeczywistych),
- **skokowa (dyskretna)** - zmienna przyjmuje dowolne wartości ze zbioru przeliczalnego (np. zbiór liczb całkowitych z określonego przedziału),
- **osobliwa** - zmienna losowa, której rozkład skupiony jest na nieprzeliczalnym zbiorze o długości 0 (np. na zbiorze Cantora), tzn. prawdopodobieństwo tego, że zmienna ta przyjmuje wartość z tego zbioru, wynosi 1, przy czym  $P(X = x) = 0$  dla każdego  $x \in \mathbb{R}$ ,
- dowolna zmienna losowa albo jest jednego z trzech powyższych typów, albo ma rozkład **mieszany** składający się z rozkładów tych typów.

### 0.2 Dystrybuanta i jej własności

**Dystrybuanta** zmiennej losowej  $X$  jest to funkcja zdefiniowana następująco:

$$F(x) = P(X \leq x). \quad (15.2)$$

(czytamy: „Dystrybuenta dla konkretnej wartości zmiennej losowej tj. dla  $X=x$  ) jest równa prawdopodobieństwu tego, że zmienna losowa  $X$  będzie przyjmowała wartości nie większe niż konkretna wartość  $x$ .”)

Własności dystrybuenty:

1.  $0 \leq F(x) \leq 1$  dla każdego  $x \in \mathbb{R}$ ,
2.  $F(x)$  jest niemalejąca,
3. jest lewostronnie ciągła,
4.  $\lim_{x \rightarrow -\infty} F(x) = 0$ ,  $\lim_{x \rightarrow +\infty} F(x) = 1$ .

Dystrybuenta zmiennej losowej *ciągłej*  $X$  jest to funkcja podana wzorem:

$$F(x) = P(X < x) = \int_{-\infty}^x f(x) dx. \quad (15.3)$$

Dystrybuenta zmiennej losowej *skokowej*  $X$  jest to funkcja podana wzorem:

$$F(x) = \sum_{x_i < p_i} p_i. \quad (15.4)$$

### 0.3 Funkcja gęstości

**Funkcja gęstości zmiennej losowej ciągłej** to funkcja  $f(x)$  określona na zbiorze liczb rzeczywistych i spełniająca następujące warunki:

1.  $f(x) \geq 0$  - jest określona nieujemnie,
2.  $\int_{-\infty}^{+\infty} f(x) dx = 1$ .

**Mediana**  $M$  to taka wartość zmiennej losowej  $x$  dla której dystrybuenta wynosi  $1/2$ .

$$F(x_{0.5}) = P(x < x_{0.5}) = 0.5 \quad (15.5)$$

Gęstość prawdopodobieństwa to szansa przyjęcia konkretnej wartości przez zmienną losową. Dystrybuenta to szansa przyjęcia przez zmienną losową wartości nie większej od argumentu.

### 0.4 Podstawowe parametry rozkładu zmiennej losowej

**Wartość oczekiwana**  $E(X)$  (nadzieja matematyczna)

$$E(x) = m. \quad (15.6)$$

Wartość  $m$  jest to taka wartość zmiennej losowej  $X$ , wokół której skupiają się wyniki wielokrotnych realizacji tej zmiennej. Innymi słowy, oczekuje się (ma się nadzieję), że wielokrotne realizacje zmiennej losowej  $X$  będą skupiały się wokół liczby  $m$ . Wartość oczekiwana należy do miar położenia.

Dla zmiennych losowych **ciągłych** wzór wygląda następująco:

$$E(x) = \int_{-\infty}^{+\infty} x f(x) dx. \quad (15.7)$$

Dla zmiennych losowych **dyskretnych** wzór wygląda następująco:

$$E(x) = \sum_{i=1}^n p_i \cdot x_i. \quad (15.8)$$

**Wariancja** należy do miar rozproszenia. Można ją wyliczyć wykorzystując wzór:

$$V(x) = E[X - E(X)]^2 = E(X^2) - E(X)^2 \quad (15.9)$$

**Odchylenie standardowe** informuje jak szeroko wartości jakiejś wielkości są rozrzucone wokół jej średniej.

$$\sigma = \sqrt{V(x)}. \quad (15.10)$$



## Rozdział 16

# Ciągła, dyskretna i szybka transformata Fouriera, widmo sygnału

**Ogólnie** W dziedzinie przetwarzania sygnałów **transformacja** Fouriera używana jest do przetworzenia funkcji  $x(t)$  ciągłej lub dyskretniej w dziedzinie czasu na funkcję  $X(s)$  ciągłą lub dyskretną w dziedzinie częstotliwości. Ocena wyrażenia  $X(s)$  pozwala określić zawartość częstotliwościową dowolnego sygnału.

## 1 Ciągła transformata Fouriera

**Transformata Fouriera** - wynik operatora liniowego (czyli jednorodny  $[cf(x) = f(cx)]$  i addytywny  $[f(a+b) = f(a) + f(b)]$ ) jakim jest **transformacja** Fouriera. Dzięki liniowości możemy transformować sygnały rzeczywiste, w innym przypadku byłyby to jedynie sygnały zawierające pojedynczy przebieg sinusoidalny.

Dla funkcji  $f(x)$  transformatę Fouriera definiujemy następująco:

$$F(s) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i x s} dx,$$

a transformację odwrotną

$$f(x) = \int_{-\infty}^{+\infty} F(s) e^{2\pi i x s} ds,$$

## 2 Dyskretna transformata Fouriera (DFT)

W praktyce pomiary pozwalają otrzymać dane o charakterze dyskretnym, stąd też zdefiniowana jest dyskretna transformata Fouriera (dyskretne przekształcenie Fouriera, DFT), używana w cyfrowym przetwarzaniu sygnałów:

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi i n m}{N}}, \quad 0 \leq k \leq N-1$$

gdzie  $x(n)$  to dyskretny ciąg wartości ciągłej zmiennej  $x(t)$ , próbkowanej w dziedzinie czasu,

$N$  – liczba próbek wejściowych w dziedzinie czasu.

DFT wyznacza zawartość widmową sygnału wejściowego w  $N$  równomiernie rozłożonych punktach na osi częstotliwości.  $N$  określa ile próbek wejściowych jest wymaganych do policzenia DFT, jaka jest rozdzielczość wyników w dziedzinie częstotliwości (odległość pomiędzy prążkami w Hz) oraz jaki jest czas przetwarzania wymagany do obliczenia  $N$ -punktowej DFT. Ze wzoru na DFT wynika bowiem, iż złożoność obliczeniowa algorytmu wynosi  $O(N^2)$  (to dużo).

## DFT

- jest procedurą matematyczną używaną do wyznaczenia zawartości harmonicznej lub częstotliwościowej sygnału dyskretnego.
- Dokładne wartości częstotliwości analizowanych przebiegów (czyli wartości prążków na osi x) zależą od częstotliwości próbkowania  $f_s$  i liczby próbek  $N$ . Przykładowo dla sygnału próbkowanego z  $f_s = 500\text{Hz}$  i 16-punktowej DFT, podstawowa częstotliwość sinusoid wynosi  $f_s/N = 31,25\text{Hz}$ . Czyli na otrzymanym wykresie wyznaczone są wartości prążków o częstotliwościach analizy: 0 Hz, 31.25, 62.6, 93.75 ... 468.75Hz
- Wyjściowe wartości amplitud prążków DFT są równe amplitudzie wejściowej danej składowej (sinusoidy) \*  $N/2$ .
- jeżeli ciąg wejściowy  $x(n)$  jest rzeczywisty, to zespolone wartości wyjściowe DFT dla argumentów  $m \geq (N/2)$  są nadmiarowe w stosunku do wartości wyjściowych dla argumentów od  $m = 0$  do  $m = (N/2) - 1$ .  $m$ -ta wartość wyjściowa DFT będzie miała taką samą amplitudę jak  $(N - m)$ -ta wartość wyjściowa DFT. Kąt fazowy  $m$ -tej wartości będzie równy kątowi fazowemu  $(N - m)$ -tej wartości wyjściowej DFT, ze znakiem ujemnym.  
Jeżeli ciąg wyjściowy DFT jest rzeczywisty, to  $X(m)$  jest sprzężeniem  $X(Nm)$ , lub:

$$X(m) = X^*(N - m)$$

gdzie \* oznacza sprzężenie.

Podsumowanie:

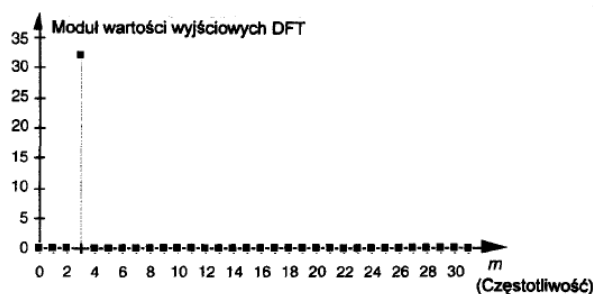
- każdy człon wyjściowy DFT jest sumą iloczynów wejściowego ciągu w dziedzinie czasu z ciągami reprezentującymi przebieg sinusoidalny i cosinusoidalny, ponieważ można napisać wzór na DFT z użyciem współrzędnych prostokątnych:

$$X(m) = \sum_{n=0}^{N-1} x(n) [\cos(\frac{2\pi nm}{N}) - i \sin(\frac{2\pi nm}{N})]$$

- dla rzeczywistych sygnałów wejściowych  $N$ -punktowa DFT daje w wyniku jedynie  $N/2$  członów niezależnych,
- DFT jest operacją liniową,
- amplitudy składowych wyjściowych DFT są wprost proporcjonalne do  $N$ ,
- rozdzielczość DFT w dziedzinie częstotliwości wynosi  $f_s/N$

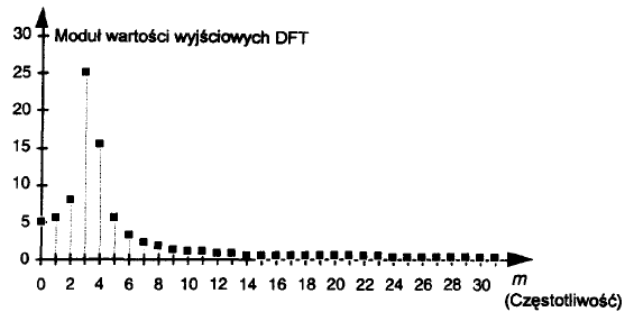
### 2.1 Przeciek DFT

Jeżeli sygnał wejściowy zawiera częstotliwość która nie należy do częstotliwości ze zbioru  $mf_s/N$  (czyli jest gdzieś pomiędzy prążkami wynikowymi) to energia tej składowej ujawni się w pewnym stopniu przy wszystkich  $N$  wartościach wyjściowych (czyli prążkach).



Rysunek 16.1: Wynik 64 punktowej DFT sygnału o częstotliwości 3 okresów w analizowanym oknie 64 próbek.

Łopatologicznie: 8 punktowa DFT przy próbkowaniu 8000Hz daje prążki w punktach 0Hz, 1000Hz, 2000Hz... 7000Hz, jak sygnał wejściowy ma składową 1500Hz to ta „rozlewa” się na wszystkie prążki.



Rysunek 16.2: Wynik DFT dla sygnału o częstotliwości 3.4 okresów w analizowanym oknie 64 próbek.

**Przeciek** jest nie do uniknięcia przy wyznaczaniu DFT rzeczywistego ciągu czasowego o skończonej długości. W celu zminimalizowania skutków przecieku można użyć funkcji okna (jednego z miliona) na wartościach wejściowego ciągu czasowego. Przeciek widma powodowany jest poprzez próbkowanie sygnału o niepełnej liczbie cykli. Wtedy na początku i końcu przedziału próbkowania mogą powstać nieciągłości, które w DFT objawiają się istnieniem wysokich składowych (których nie było w sygnale wejściowym) - nieciągłości te tworzą listki boczne sygnału próbkowanego. Jest to bowiem próbkowanie z użyciem okna prostokątnego. Poprzez zastosowanie innych okien minimalizowane są nieciągłości w punktach końcowych przedziału próbkowania, co w efekcie minimalizuje przeciek widma.

Jak polepszyć rozdzielczość DFT? Zwiększyć  $N$ , nawet gdy nie mamy więcej próbek, to zawsze można dodać zera i liczyć dla takiego sygnału. Poprawa rozdzielczości DFT poprzez uzupełnianie zerami.

### 3 Szybka transformata Fouriera (FFT)

Wyznaczenie DFT o tysiącach punktów jest nieefektywne czasowo, tzn. wymaga olbrzymiej mocy obliczeniowej, z tego powodu w 1965 roku opublikowano artykuł przedstawiający wydajny algorytm implementujący DFT znany dzisiaj jako szybkie przekształcenie Fouriera - FFT. Algorytm przetransformował dziedzinę cyfrowego przetwarzania sygnałów. Z wielu zaproponowanych algorytmów najpopularniejszym jest algorytm FFT o podstawie 2, tzn. liczba punktów w transformacji wynosi  $N=2^k$ , gdzie  $k$  jest liczbą naturalną.

Zalety FFT o podstawie 2 i metody wykorzystania:

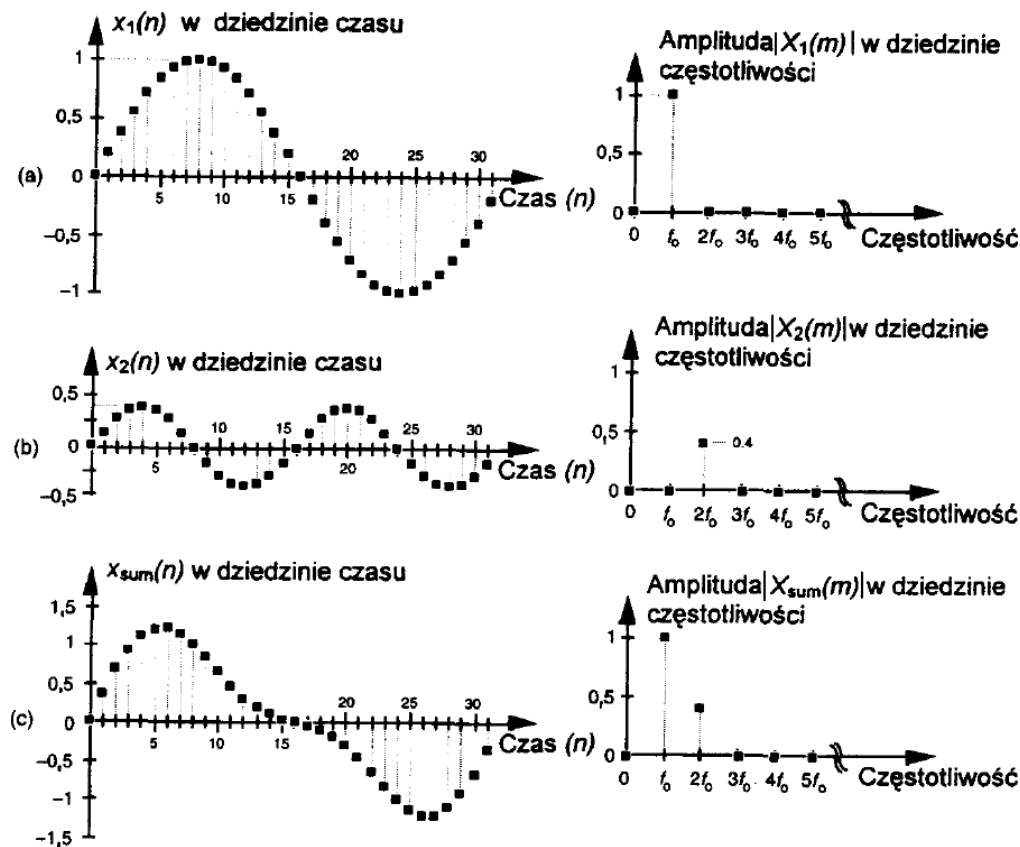
- Redukcja ilości mnożeń z  $N^2$  do około  $N/2 * \log_2 N$ , dla  $N = 512$  to 200 razy mniej mnożeń zespolonych, dla  $N = 8192$  redukcja 1000 krotna.
- FFT nie przybliża DFT, ale daje dokładny wynik DFT, z zachowaniem wszystkich właściwości DFT!
- Jeżeli nie mamy wpływu na ilość pobieranych próbek, a nie jest to liczba będąca potęgą 2, to należy dodać zera do najbliższej potęgi 2, nigdy nie obcinać.
- Poprawianie wyników FFT - do wykrycia sygnałów w obecności szumu, można wykorzystać uśrednianie kolejnych procedur FFT

**Opis algorytmu** Obliczenie  $N$ -punktowej FFT można przyspieszyć przez policzenie FFT  $N/2$  punktowej, którą można przyspieszyć przez policzenie  $N/4$  FFT itd. Za każdym razem obliczenie drugiej połowy FFT jest jedynie kwestią kilku dodawań i przemnożenia przez -1 więc znacząco zmniejsza się ilość operacji (dodatkowo, dodawanie i zmiana znaku jest tańsze od mnożenia)

Warto nauczyć się chociaż jak narysować „motylki” FFT, nie ma sensu tego przepisywać tylko od strony 8 rozdziału 4 z książki Lyonsa.

## 4 Widmo sygnału

Określenie składowych częstotliwościowych sygnałów dyskretnych w dziedzinie czasu, dokonywane jest w *dziedzinie częstotliwości*. Zobrazowane  $X(m)$  (czyli wartości prążków sygnału) zwane są widmem sygnału.  $x(n)$  – ciąg czasowy sygnału  $X(m)$  – ciąg widmowy  $X$  zmiennej  $m$



Rysunek 16.3: Widma różnych sygnałów. Proszę zwrócić uwagę na liniowość operacji DFT.  $f_0$  - częstotliwość podstawowa prążków =  $\frac{f_s}{N}$

**Widmo** - przedstawienie sygnału w dziedzinie częstotliwości lub pulsacji, otrzymane przy pomocy transformacji Fouriera. Z wykresu tego można przykładowo odczytać, jakie składowe harmoniczne wchodzi w skład danego sygnału, czy sygnał ma ograniczone pasmo, jaka jest szerokość jego pasma, czy zawiera składowe wolnozmiennne (o małych częstotliwościach) oraz szybkozmiennne (o wielkich częstotliwościach).

Ponieważ transformata Fouriera  $F(j\omega)$  jest w ogólności funkcją o wartościach zespolonych, zatem przy wykonywaniu wykresów widma wygodne jest niezależne przedstawianie modułu  $M(\omega) = |F(j\omega)|$  oraz argumentu  $\theta(\omega) = \arg F(j\omega)$ . Wykres **widma amplitudowego** (wykres modułu) pokazuje, jakie są amplitudy składowych widmowych sygnału o różnych częstotliwościach. Wykres **widma fazowego** (wykres argumentu) pokazuje, jakie są fazy tych składowych.

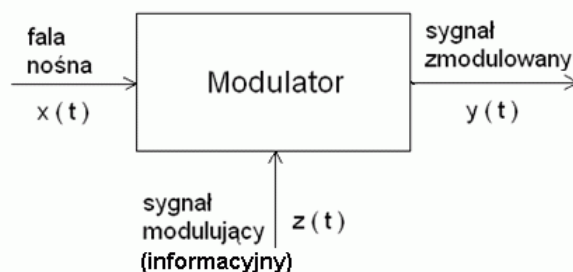
## Rozdział 17

# Modulacje analogowe i cyfrowe

**Modulacja** - celowa lub samorzutna zmiana parametrów sygnału.

Na wyjściu źródła informacji sygnał oryginalny jest przeważnie dolnopasmowy. Jednak kanały radiowe i telekomunikacyjne są zawsze środkowoprzepustowe. Aby zamienić sygnał oryginalny na sygnał środkowopasmowy musimy zastosować metodę zwaną modulacją. Modulacja jest to uzmiennienie standardowego przebiegu  $x(t)$  nazywanego sygnałem lub falą nośną przez sygnał  $z(t)$  zwany sygnałem modulującym. W wyniku tego jako sygnał wyjściowy uzyskujemy sygnał zmodulowany  $y(t)$ . Schemat blokowy procesu modulacji został przedstawiony na rysunku 17.1. Inaczej mówiąc modulacja to zmiana parametrów fali nośnej, która może być np. sinusoidalna lub prostokątna. W przypadku modulowania fal sinusoidalnych zmianom może ulegać amplituda, częstotliwość lub faza. Natomiast dla fal prostokątnych szerokość, amplituda lub ilość impulsów. W zależności od charakteru zmian parametru modulowanego rozróżniamy modulacje analogowe (zmiany o charakterze ciągłym) oraz cyfrowe (zmiany o charakterze skokowym).

### 0.1 Modulacje analogowe



Rysunek 17.1: Schemat blokowy układu modulacji

**Modulacja amplitudy AM** (ang. Amplitude Modulation) ma miejsce, gdy w procesie modulacji następują zmiany amplitudy sygnału nośnego proporcjonalne do chwilowej wartości sygnału modulującego. Uzyskujemy ją poprzez zmianę parametru sygnału o częstotliwości nośnej. W tym przypadku jest to amplituda sygnału nośnego  $Y_0$ , która zmienia się pod wpływem sygnału modulującego  $z(t)$ . Uzyskujemy w ten sposób sygnał zmodulowany nadający się np. do transmisji drogą radiową. Proces modulacji jest realizowany w urządzeniu zwanym modulatorem. Modulację amplitudy AM charakteryzuje współczynnik głębokości modulacji  $m$ , zwany często głębokością modulacji. Definiujemy go jako stosunek amplitudy przebiegu modulującego  $M$  do amplitudy sygnału nośnego. Podawany jest on zazwyczaj w procentach.

$$m = M/Y_0. \quad (17.1)$$

**Modulacja częstotliwości FM** (ang. Frequency Modulation) ma miejsce, gdy w procesie modulacji zmianie ulega częstotliwość sygnału nośnego  $x(t)$  pod wpływem sygnału modulującego  $z(t)$ . Częstotliwość

chwilowa zmienia się według wzoru:

$$f(t) = f_0 + k \cdot z(t), \quad (17.2)$$

gdzie :

$z(t)$  - przebieg modulujący,

$k$  - stała proporcjonalności.

Częstotliwość sygnału nośnego  $f_n$  zmienia się w pewnym zakresie, a różnica pomiędzy najniższą i najwyższą chwilową wartością częstotliwości fali nośnej nazywana jest dewiacją częstotliwości  $\Delta f$ . Stosunek dewiacji częstotliwości do częstotliwości sygnału nośnego:

$$m_f = \frac{\Delta f}{f_n}. \quad (17.3)$$

nazywamy współczynnikiem modulacji częstotliwości lub wskaźnikiem dewiacji częstotliwości. W zależności od wartości wskaźnika rozróżniamy szerokopasmową modulację częstotliwości (wskaźnik większy od jedności) oraz wąskopasmową modulację częstotliwości (wskaźnik mniejszy od jedności). Modulacja częstotliwości FM jest stosowana nie tylko do przesyłania sygnału radiowego w zakresie fal ultrakrótkich, ale również do transmisji sygnału w telewizji satelitarnej i sygnału dźwiękowego w wielu systemach telewizji naziemnej. Modulacja częstotliwości FM jest bardzo podobna do modulacji fazy PM. Gdy zmieniamy częstotliwość sygnału nośnego to zmiana ulega również faza. Identycznie odbywa się to w drugą stronę, gdy zmieniamy fazę sygnału nośnego, zmiana ulega również jego częstotliwość. Jednak modulacje FM i PM nie są sobie równoważne. Gdy odbiornik FM jest używany do demodulacji sygnału PM sygnał audio jest zniekształcony. Dzieje się tak dlatego, gdyż związek pomiędzy częstotliwością a fazą jest nieliniowy.

**Modulacja fazy PM** (ang. Phase Modulation) ma miejsce, gdy w procesie modulacji zmiane ulega chwilowa wartość fazy sygnału nośnego  $x(t)$ , która zmienia się zgodnie z zależnością:

$$\Theta(t) = \omega t + k \cdot z(t), \quad (17.4)$$

gdzie:

$z(t)$  - przebieg modulujący,

$k$  - stała proporcjonalności.

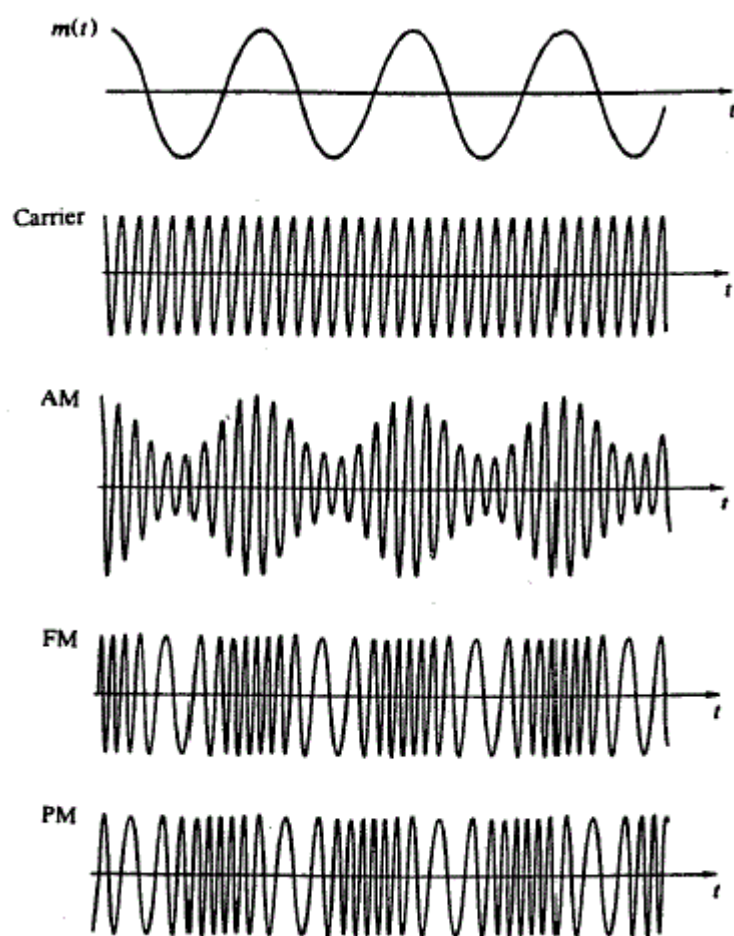
Modulacja częstotliwości pozwala na stosowanie prostszych modulatorów i demodulatorów sygnału. Dlatego też modulację fazy stosujemy głównie do transmisji cyfrowej. Charakterystyczny dla modulacji fazy jest współczynnik zwany dewiacją fazy, który definiujemy w następujący sposób:

$$\Delta\phi_{PM} = k \cdot z_{max}. \quad (17.5)$$

## 0.2 Modulacje cyfrowe

**Modulacja cyfrowa** to proces zmiany analogowego sygnału nośnego przez binarny sygnał modulujący, który z łatwością możemy przesłać np. drogą radiową. Sygnałem nośnym w modulacji cyfrowej jak i analogowej jest przebieg sinusoidalny, jednak w modulacjach cyfrowych sygnał modulujący to strumień elementów binarnych. Podobnie jak w modulacjach analogowych tak i w cyfrowych zmianom ulega amplituda, częstotliwość lub faza przebiegu nośnego. Ponieważ cyfrowy sygnał modulujący zmienia się skokowo, to zmiany w przebiegu zmodulowanym również są skokowe. Celem modulacji jest dopasowanie właściwości sygnału wyjściowego zmodulowanego do parametrów kanału transmisyjnego.

**Modulacja ASK** (ang. Amplitude Shift Keying) zwana kluczowaniem amplitudy ma miejsce, gdy występuje zmiana amplitudy sygnału nośnego w zależności od cyfrowego sygnału modulującego. Proces kluczowania amplitudy jest zbliżony do modulacji amplitudy AM, a w zasadzie jest to szczególny rodzaj modulacji AM. Różnica polega na tym, iż sygnał modulujący jest sygnałem cyfrowym. Cechą charakterystyczną kluczowania amplitudy jest to, że dzięki cyfrowemu sygnałowi modulującemu w czasie trwania stanu wysokiego występuje pełna amplituda sygnału zmodulowanego natomiast w stanie niskim jest ona wytłumiona. Podobnie jak modulacja AM, kluczowanie amplitudy jest liniowe, czułe na zakłócenia atmosferyczne i zniekształcenia.



Rysunek 17.2: Modulacje

**Modulacja FSK** (ang. Frequency Shift Keying), zwana kluczkowaniem częstotliwości, jest szczególnym przypadkiem modulacji częstotliwości FM. Sygnałem modulującym jest sygnał cyfrowy. Modulacja FSK polega na przypisaniu odpowiedniej częstotliwości sygnału nośnego każdemu z dwóch stanów sygnału modulującego. Przejście z jednej częstotliwości może odbywać się z ciągłością fazy lub bez niej. Ta modulacja wyróżnia się stałą amplitudą chwilową, niezależną od sygnału modulującego. Ta właściwość jest bardzo przydatna przy transmisji sygnałów przez kanały, gdzie występują zmiany amplitudy transmitowanego sygnału na skutek nieliniowości tego kanału. Zaletą tej modulacji jest również odporność na zakłócenia impulsowe oraz zniekształcenia tłumieniowe i opóźnieniowe. Dlatego też modulacja FSK jest częściej stosowana niż modulacja ASK.

**Modulacja PSK** (ang. Phase Shift Keying) zwana kluczkowaniem fazy to jeden z rodzajów modulacji PM. Ma ona miejsce, gdy przy stałej amplitudzie i częstotliwości harmonicznego sygnału nośnego występuje przesunięcie fazy w zależności od stanu informacji pierwotnej. Sygnały przy modulacji PSK są zawsze transmitowane w systemach koherentnych, czyli faza sygnału nadawanego znana jest po stronie odbiorczej. Podobnie jak w przypadku modulacji FSK, modulacja PSK charakteryzuje się stałą w czasie amplitudą chwilową, co odróżnia ją od modulacji ASK, gdzie amplituda chwilowa zmienia się. Powoduje to większe narażenia na zniekształcenia nieliniowe.

**Modulacja QPSK** (ang. Quadrature Phase Shift Keying) zwana modulacją kwadraturową to jeden z rodzajów modulacji fazy. Może być ona traktowana jako klasyczna modulacja 4 - wartościowa PSK nośnej o amplitudzie  $A$ , bądź jako złożenie dwóch dwuwartościowych modulacji amplitudy BASK (ang. Binary Amplitude Shift Keying) o amplitudzie  $A/\sqrt{2}$  i ortogonalnych nośnych  $\sin 2\pi f_0 t$  oraz  $\cos 2\pi f_0 t$ . Zastosowano w niej kodowanie na czterech ortogonalnych przesunięciach fazowych sygnału nośnego. Jej zaletą jest zwiększenie efektywności wykorzystania pasma, przy jednoczesnym braku negatywnego wpływu na bitową stopę błędów. Modulację QPSK definiujemy:

$$x = A \cos(2\pi f t + \Theta), \quad 0 < t < T, \quad i = 1, 2, 3, 4 \quad \text{oraz} \quad \Theta = \frac{\pi(2i - 1)}{4} \quad (17.6)$$

Wyjściowe fazy sygnału to  $\pi/4, 3\pi/4, 5\pi/4, 7\pi/4$ . Częstotliwość nośnej jest wybierana jako całkowita wielokrotność szybkości nadawania znaku, dlatego faza sygnału jest jedną z wymienionych powyżej faz.

**Modulacja MSK** (ang. Minimum Shift Keying) jest szczególnym przypadkiem modulacji FSK, gdy częstotliwości  $f_1$  i  $f_2$  są równe odpowiednio  $f_0 - 1/(4T_b)$  i  $f_0 + 1/(4T_b)$ . MSK może być również traktowana jako zmodyfikowana forma modulacji OQPSK (ang. Offset Quadrature Phase Shift Keying), w której elementy odpowiadające składowej synfazowej i kwadraturowej są odpowiednio ukształtowane, a następnie wymnożone przez przebiegi nośne. Przez ukształtowanie rozumiemy zastąpienie impulsu prostokątnego impulsem sinusoidalnym. Modulację MSK definiujemy jako:

$$s(t) = s_I(t) \cos(2\pi f_0 t) + s_Q(t) \sin(2\pi f_0 t). \quad (17.7)$$

Modulacja MSK zalicza się do klasy modulacji z ciągłą fazą CPM (ang. Continuous Phase Modulation).

#### Zastosowania:

- transmisja danych binarnych w kanale o wąskim paśmie,
- łączność modemowa, faksowa,
- łączność radiowa (telemetria, zdalne sterowanie),
- systemy bezprzewodowe,
- telefonia cyfrowa (GSM, UMTS, TETRA, ...),
- łączność satelitarna.



## Rozdział 18

# Wzmacniacze operacyjne: właściwości i zastosowania

Materiały: wzmacniacze\_operacyjne.pdf

## Rozdział 19

# Mikroprocesory: budowa, zastosowania

### 0.1 Architektury

**Architektura harwardzka** W tej architekturze pamięć programu przechowująca rozkazy została oddzielona od pamięci danych (dwie różne magistrale). W trakcie pobierania argumentów wykonywanej właśnie instrukcji można równocześnie zacząć pobieranie następnego słowa rozkazowego (pre-fetch).

**Architektura von Neumanna** Architektura zakłada, że podział przestrzeni adresowej na pamięć programu i pamięć danych jest czysto umowny. W ten sposób jest uproszczony dostęp do obu pamięci, gdyż wykonywany jest za pomocą tych samych instrukcji. Główną wadą jest wydłużenie wykonywania cyklu.

**Architektura super-harwardzka** Pamięć programu i pamięć danych są oddzielone od siebie, ale wykorzystują one wspólną magistralę: danych i adresową.

**CISC** (ang. Complex Instruction Set Computing) to architektura konwencjonalnych procesorów. Charakteryzuje się ona znaczną liczbą elementarnych rozkazów i trybów adresowania przy niewielkiej liczbie rejestrów uniwersalnych. Jest ona bardzo wolna w porównaniu do RISC-ów. Na tej architekturze były oparte pierwsze procesory z rodziny x86. Dziś w tej rodzinie nie spotka się procesora CISC, ale jest ona zastąpiona przez wewnętrzny RISC. Jednak pozostała zgodnie z zasadą "kompatybilności w tył" mała liczba rejestrów, co jest główną bolączką projektantów procesorów x86.

**RISC** (ang. Reduced Instruction Set Computing/Computer) wywodzi się z Berkeley (1985). Koncepcja tego typu architektury jest oparta na ograniczeniu liczby krótkich (maksymalnie dwusłowych) rozkazów mających niewiele formatów i trybów adresowania. Procesor RISC posiada wiele rejestrów uniwersalnych (nawet powyżej 100). Działanie procesora przyspieszają dodatkowe ulepszenia, np. przetwarzanie potokowe, czy pamięć podręczna. Instrukcje wykonują proste operacje, dzięki czemu mogą to robić szybko - każda jednostka wykonawcza jest w stanie wykonać dokładnie jedną instrukcję w jednym cyklu zegara. Lista instrukcji zawiera tylko kilka rozkazów umożliwiających dostęp do pamięci: załadowanie (load), zapis (store) oraz instrukcje semaforowe. Wszystkie pozostałe rozkazy operują wyłącznie na rejestrach - to również upraszcza budowę układu. Instrukcje kodowane są w prosty sposób - każdy rozkaz ma taką samą długość (np. 32 bity).

**VLIW** (ang. Very Long Instruction Word) – nazwa architektury mikroprocesorów z bardzo długim słowem instrukcji. Obecnie procesory VLIW są oparte na architekturze RISC, zazwyczaj z czterema lub maksymalnie ośmioma jednostkami obliczeniowymi. Po normalnej kompilacji programu, kompilator VLIW porządkuje kod na ścieżki, które wprost nie posiadają jakichkolwiek zależności. Następnie są one dzielone na cztery lub więcej części (jeden dla każdej jednostki obliczeniowej CPU) i pakowane razem w większe instrukcje z dodatkową informacją odnośnie do jednostki, na której ma być wykonywana. Rezultatem tego jest pojedynczy wielki opcode - kod operacji, stąd nazwa „Very Long”.

## 0.2 Budowa

Jeśli chodzi o budowę fizyczną, mikroprocesor to nic innego jak krzemowa płytka z milionem tranzystorów, które blokują lub umożliwiają przepływ prądu. Z tranzystorów budowane są bramki logiczne, a te z kolei są łączone w bardziej rozbudowane układy.

**ALU** (ang. Arithmetic Logic Unit) - wykonuje podstawowe operacje arytmetyczne (dodawanie, odejmowanie, dzielenie oraz mnożenie oraz logiczne (OR, AND, XOR, NOT) oraz przesunięcia bitowe. ALU współpracuje z roboczym rejestrem zwanym akumulatorem (lub wieloma akumulatorami), który przechowuje jeden z operandów (argumentów) wykonywanej operacji oraz wyniku tej operacji.

**CU** (ang. Control Unit) - dekoduje zawartość rejestru rozkazów i generuje odpowiednie sygnały sterujące zapewniające prawidłowy przebieg operacji zdefiniowanej kodem rozkazu.

**Rejestry** (ang. Register) - komórki pamięci do przechowywania tymczasowych wyników obliczeń, adresów lokacji w pamięci RAM itp. Rejestry są najszybszym rodzajem pamięci.

- **Rejestr instrukcji IR** (ang. Instruction Register) - przechowuje aktualnie wykonywaną instrukcję.
- **Licznik rozkazów PC** (ang. Program Counter) - przechowuje adres w pamięci, gdzie przechowywany jest kolejny rozkaz do pobrania. Rozkazy są przechowywane w postaci kodów binarnych.
- **Akumulator A** (ang. Accumulator) - przechowuje argument (operand) do operacji ALU lub wynik operacji.
- **Wskaźnik stosu SP** (ang. Stack Pointer) - wskazuje na szczyt stosu (adres ostatniej zajętej komórki stosu).
- **Rejestr flagowy** - przechowuje informacje dotyczące operacji ALU np. flaga przeniesienia lub pożyczki CF (ang. Carry Flag), flaga parzystości PF (ang. Parity Flag), flaga przepełnienia OF (ang. Overflow Flag) itp.

**Magistrale** (ang. Bus) - wewnętrzne szyny łączące.

- **szyna danych** (ang. data bus) - magistrala komunikacyjna wykorzystywana do przesyłania właściwych danych,
- **szyna adresowa** (ang. address bus) - łączy CPU z pamięcią. Określa pod jaki adres mają zostać wysłane dane szyną danych. Szerokość magistrali (liczba linii) określa maksymalną pojemność pamięci systemu (przestrzeń adresową)
- **szyna sterująca** (ang. control bus) - zapewnia regulację dostępu do szyny adresowej i szyny danych.

## 0.3 Mikroprocesor a mikrokontroler

Na system mikroprocesorowy składa się mikroprocesor, układy wejścia/wyjścia, pamięć programu i danych, szyny adresowe, szyny danych ..., system operacyjny. Mikrokontroler to pojedynczy układ scalony zawierający kompletny system, zdolny do samodzielnego wykonywania operacji arytmetycznych i logicznych oraz do sterowania układami i elementami zewnętrznymi. Typowy uC zawiera CPU, pamięć programu (często FLASH), pamięć danych RAM, układu wejścia/wyjścia, wewnętrzne źródło taktowania, interfejsy komunikacyjne oraz inne układy peryferyjne np. kontroler przerwań, timery, ADC, DAC, DMA.

## 0.4 Zastosowania

- elektronika przemysłowa - sterowniki PLC,
- elektronika powszechnego użytku - telefony komórkowe, zegarki, komputery
- telekomunikacja - routery, switche etc.,
- technika samochodowa - piloty, sterowniki świateł, lusterek, radia, kontrolery wtrysku,

- medycyna - ciśnieniomierze, EKG, USG, termometry, mierniki poziomu cukru we krwi,
- automatyka budynków - sterowniki klimatyzacji i rolet.

## Rozdział 20

# Sieci komputerowe: budowa, protokoły, zastosowanie

**Sieć komputerowa** - wzór wzajemnie połączonych komputerów, które mogą pracować samodzielnie i komunikować się z innymi komputerami.

### 0.1 Budowa

#### Składniki sieci komputerowych

- **hosty** - komputery wykorzystywane przez użytkowników,
- **serwery** - stale działające komputery o dużej mocy obliczeniowej świadczące usługi hostom (udostępnianie plików, baz danych itp.),
- **medium transmisyjne** - nośnik informacji (kable miedziane, światłowody lub fale radiowe),
- **sprzęt sieciowy** - koncentratory, przełączniki, routery, karty sieciowe, modemy, punkty dostępu,
- **oprogramowanie** - programy komputerowe zainstalowane na urządzeniach sieciowych.

#### Model odniesienia OSI (ang. Open System Interconnection)

- **warstwa aplikacji** - określa w jaki sposób aplikacje działają ze sobą,
- **warstwa prezentacji** - dodaje podstawowe formatowanie do prezentacji danych,
- **warstwa sesji** - zarządza przebiegiem komunikacji pomiędzy dwoma komputerami,
- **warstwa transportowa** - sprawdza poprawność wysyłanych danych,
- **warstwa sieci** - adresuje wiadomości wewnątrz i pomiędzy sieciami,
- **warstwa łącza danych** - określa sposób uzyskiwania dostępu do fizycznego medium,
- **warstwa fizyczna** - przesyła dane przez fizyczne medium.

**Budowa (topologia)** warunkowana jest przez zastosowanie sieci. Najprostsze komunikowanie się sieci można zrealizować przez jedynie połączenie komputerów, w innych przypadkach używa się urządzeń kierujących ruchem.

**Topologia magistrali (szyny, linii)** - połączone jednym, współdzielonym medium.

#### Zalety:

- Brak koncentratorów/przełączników
- Awaria węzła nie powoduje paraliżu sieci

**Wady:**

- Awaria kabla powoduje paraliż sieci
- Ograniczona możliwość rozbudowy
- Niska przepustowość
- Obsługuje tylko jeden kanał transmisyjny

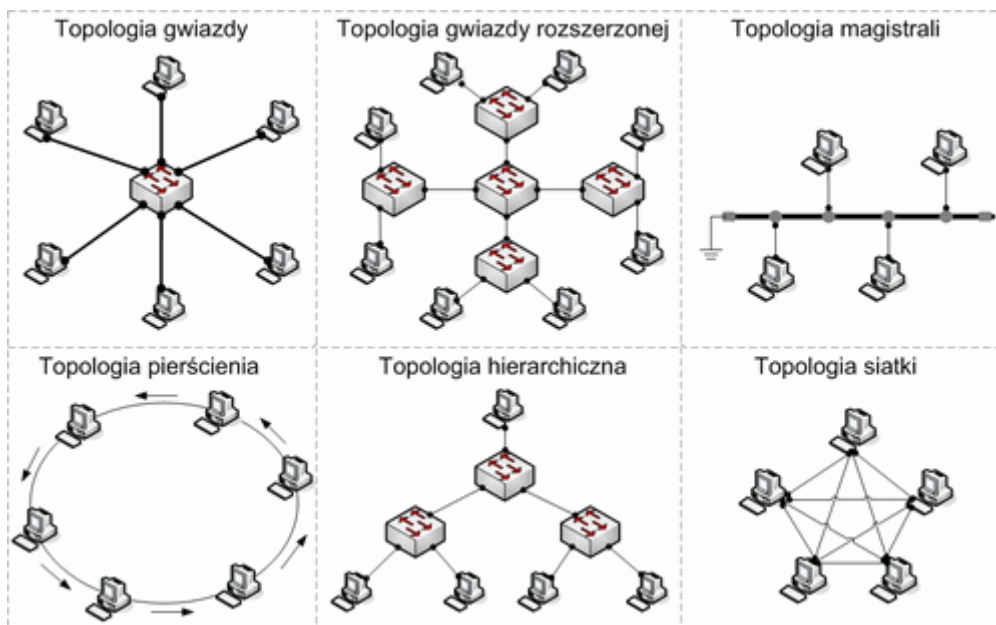
**Topologia gwiazdy** - posiada punkt centralny (switch, koncentrator) i gwiazdźście połączone do niego komputery.

**Zalety:**

- Bardzo łatwa rozbudowa sieci
- Awaria węzła nie powoduje paraliżu sieci
- Wysoka przepustowość

**Wady:**

- Ograniczenie odległości stacji roboczej od koncentratora
- Uszkodzenie koncentratora powoduje całkowity paraliż sieci



Rysunek 20.1: Topologie sieci komputerowych

**Topologia pierścienia** - komputery połączone są za pomocą jednego nośnika informacji w układzie zamkniętym - okablowanie nie ma żadnych zakończeń (tworzy krąg).

**Zalety:**

- Niskie koszty budowy

### Wady:

- Niska przepustowość
- Trudna do rozbudowy
- Ciężka lokalizacja uszkodzeń
- Uszkodzenie jednej stacji powoduje paraliż sieci

### Rozszerzone topologie

- **Topologia siatki**
- **Topologia gwiazdy rozszerzonej** – posiada punkt centralny (podobnie do topologii gwiazdy) i punkty poboczne (jedna z częstszych topologii fizycznych Ethernetu)
- **Topologia podwójnego pierścienia** – poszczególne elementy są połączone pomiędzy sobą odcinkami tworząc dwa zamknięte pierścienie
- **Topologia siatki** – oprócz koniecznych połączeń sieć zawiera połączenia nadmiarowe; rozwiązanie często stosowane w sieciach, w których wymagana jest bezawaryjność

### Podstawowe urządzenia kierujące ruchem w sieci

- switch - urządzenie łączące segmenty sieci komputerowej pracujące głównie w drugiej warstwie modelu ISO/OSI (łącza danych), jego zadaniem jest przekazywanie ramki między segmentami sieci z doбором portu przełącznika, na który jest przekazywana.
- router - urządzenie sieciowe pracujące w trzeciej warstwie modelu OSI. Służy do łączenia różnych sieci komputerowych. Na podstawie informacji zawartych w pakietach TCP/IP jest w stanie przekazać pakiety z dołączonej do siebie sieci źródłowej do docelowej, rozróżniając ją spośród wielu dołączonych do siebie sieci. Proces kierowania ruchem nosi nazwę trasowania, routingu lub routowania.

## 0.2 Protokoły

**Protokół komunikacyjny** - zbiór reguł i kroków postępowania, które są wykonywane podczas komunikacji co najmniej dwóch urządzeń ze sobą.

### Protokoły warstwy aplikacji

**FTP** (ang. File Transfer Protocol) umożliwia dwukierunkowe przesyłanie plików binarnych i tekstowych, korzysta przy tym z protokołu TCP. FTP działa w oparciu o zasadę klient-serwer i korzystanie z usługi polega na użyciu interaktywnej aplikacji. Technologia FTP zapewnia ochronę stosując hasła dostępu.

**HTTP** (ang. Hypertext Transfer Protocol) odpowiada za przesyłanie dokumentów hipertekstowych w sieci WWW, która jest najszybciej rozwijającą się i najczęściej używaną częścią internetu. Przy jego pomocy przebiega komunikacja między klientami i serwerami sieci Web.

**HTTPS** (ang. HyperText Transfer Protocol Secure) to szyfrowana wersja protokołu HTTP. Zamiast używać w komunikacji klient-serwer niezaszyfrowanego tekstu, szyfruje go za pomocą protokołu SSL. Zapobiega to przechwytywaniu i zmienianiu przesyłanych danych. HTTPS działa domyślnie na porcie nr 443 w protokole TCP.

**DNS** (ang. Transmission Control Protocol) jest systemem tłumaczenia internetowych nazw domenowych na adresy IP. System DNS jest rozproszoną bazą danych obsługiwana przez wiele serwerów, z których każdy posiada tylko informacje o domenie, którą zarządza, oraz o adresie serwera nadrzędnego. Na najwyższym poziomie znajdują się tzw. główne serwery nazw (root level servers), które znajdują się w Stanach Zjednoczonych i podłączone są do szybkich sieci szkieletowych Internetu. Przechowują one adresy serwerów nazw dla domen najwyższego poziomu, np. .com, .edu, .org, oraz domen krajowych, np. .pl, .de, .uk. Adresy serwerów głównych muszą być znane każdemu innemu serwerowi nazw. Wewnątrz każdej domeny można tworzyć tzw. subdomeny, np. wewnątrz domeny .pl utworzono wiele domen regionalnych jak .waw.pl, .lodz.pl itp, oraz funkcjonalnych jak .com.pl, .gov.pl lub .org.pl, należących do firm, organizacji lub osób prywatnych.

**SMTP** (ang. Simple Mail Transfer Protocol) odpowiada za przesyłanie poczty elektronicznej pomiędzy komputerami pracującymi w sieci.

**POP3** (ang. Post Office Protocol v 3) pozwalający na odbiór poczty elektronicznej ze zdalnego serwera do lokalnego komputera poprzez połączenie TCP/IP.

**TELNET** (ang. Terminal emulation) - umożliwia użytkownikowi zdalny dostęp do innego komputera: zalogowanie się hoście internetowym i wykonywanie poleceń. Wysyłane dane nie są szyfrowane. Aktualnie częściej wykorzystywany jest protokół SSH (ang. Secure SHell) z szyfrowaniem danych.

## Protokoły warstwy transportowej

**TCP** (ang. Transmission Control Protocol) działa w warstwie transportowej w *trybie połączeniowym*. Gwarantuje dostarczenie danych do odbiorcy. Połączenia TCP są połączeniami wirtualnymi, rozpoznawanymi po adresach i portach urządzeń docelowych i źródłowych. Połączenia takie charakteryzują się możliwościami sterowania przepływem, potwierdzaniem odbioru, zachowywaniem kolejności danych, kontrolą błędów i przeprowadzaniem retransmisji. Odbiorca po odebraniu danych zobowiązany jest do przesłania do nadawcy potwierdzenia odebrania danych. Jeżeli potwierdzenie nie nadejdzie w określonym czasie, to nadawca wysyła dane ponownie. Segmenty TCP składają się z nagłówka i danych.

**UDP** (ang. User Datagram Protocol) działa w warstwie transportowej w *trybie bezpołączeniowym*. Protokół ten nie gwarantuje dostarczenia danych do odbiorcy. Jeżeli pakiet nie dotrze do odbiorcy, lub dotrze uszkodzony, UDP nie podejmie żadnych działań zmierzających do retransmisji danych, a zapewnienie niezawodności pozostawi warstwie wyższej. Protokół wykorzystywany jest do szybkiego przesyłania danych w niezawodnych sieciach.

## Protokoły warstwy sieci

**IP** (ang. Internet Protocol) zapewnia usługę dostarczania pakietów danych z jednego punktu sieci do drugiego. Nie analizuje zawartości pakietu, ale wyszukuje ścieżkę prowadzącą do jego miejsca przeznaczenia. Protokół ten wykorzystuje adresy sieciowe komputerów zwane adresami IP. Są to 32-bitowa liczba zapisywana jako sekwencje czterech ośmiobitowych liczb dziesiętnych (mogących przybierać wartość od 0 do 255), oddzielonych od siebie kropkami. Adres IP dzieli się na dwie części: identyfikator sieciowy (network id) i identyfikator komputera (host id). Istnieje kilka klas adresowych, o różnych długościach obydwu składników. Obowiązujący obecnie sposób adresowania ogranicza liczbę dostępnych adresów, co przy bardzo szybkim rozwoju Internetu jest dla niego istotnym zagrożeniem. W celu ułatwienia zapamiętania adresów wprowadzono nazwy symboliczne, które tłumaczone są na adresy liczbowe przez specjalne komputery w sieci, zwane serwerami DNS.

**ARP** (ang. Address Resolution Protocol) odpytuje wszystkie komputery w sieci, czy mają potrzebny mu adres IP i prosi o przesłanie odpowiadającego mu adresu fizycznego MAC. Aby ograniczyć ruch w sieci budowana jest dynamiczna tablica ARP, w której zapisywane są pary adres IP adres MAC komputerów z którymi został nawiązany kontakt. Tablica ta ma ograniczony rozmiar. Jeśli tablica ARP przepełni się, to jest z niej usuwany najstarszy wpis.



### 0.3 Zastosowania

- współdzielenie zasobów np. plików, drukarek,
- komunikacja np. poczta e-mail, telefonia,
- sieci przemysłowe,
- bezprzewodowe sieci czujników,