Metody numeryczne

Prof. dr hab. Ireneusz Grabowski

Instytut Fizyki UMK Toruń

WFAiIS

www.fizyka.umk.pl/~ig

Google: Irek Grabowski

2021

Irek Grabowski - Home

www.fizyka.umk.pl/~ig/jml/

110%

Szukaj

Irek Grabowski

Irek Grabowski home page

Main Menu

- Home
- Research
- Students&Teaching
- Contact
- News

Other

- Hobby

Login Form

Login

Hasło

Zapamiętać?

Zaloguj

Nie pamiętam hasła!

Home

.

prof. dr hab. Ireneusz Grabowski

Nicolaus Copernicus University Toruń Poland
Faculty of Physics, Astronomy and Informatics
Institute of Physics
Department of Quantum Physics

e-mail: ig@fizyka.umk.pl irek.grabowski@fizyka.umk.pl

News

new pictures: Peruvian Andes 2016Cordillera Blanca Tatry Orla Perć 2016 High Tatras May 2016 Hunagshan Mountains China 2015 High Tatra May 2014

My favorite trips - **Himalaya trekking 2012- Langtang - Cherkori 4948m** **Peruvian Andes 2016Cordillera Blanca**

Latest publications: **Google Scholar**

- S. Śmiga, O. Franck, B. Mussard, A. Buksztel, I. Grabowski, E. Luppi, J. Toulouse, Self-consistent double-hybrid density-functional theory using the optimized-effective-potential method *J. Chem. Phys.* **145** *(2016); 144102-1 - 144102-12*
- S. Śmiga, F. Della Sala, A. Buksztel, I. Grabowski, E. Fabiano, Accurate Kohn-Sham ionization potentials from scaled-opposite-spin second-order optimized effective potential methods, *J. Comput. Chem* **37** *2081 - 2090 (2016)*
- S. Śmiga, I. Grabowski, The correlation effects in density functional theory along the dissociation path. *Adv. Quantum Chem.* **73** *263 - 283 (2016)*

szukaj...

Popular

- Students main
- Tematy prac
- Licencjat
- inzynier
- Wspolpraca

Who's Online

Statistics

Użytkownicy: 3
Nowości: 38
Linki: 5
gości: 588819

www.fizyka.umk.pl/~ig/pictures/Peru2016/capture_20160919124301.html

09:53
03.10.2017

Czym się zajmuję na co dzień?

Rozwój (opracowywanie - wymyślanie, testowanie, obliczenia dla układów rzeczywistych) metod opisujących układy wieloelektronowe – atomy, cząsteczki, białka, układy biologiczne, ciało stałe (układy krystaliczne), ...

- Opis na poziomie elektronowym – przewidujemy własności atomów, cząsteczek, ..., enzymów, białek, kryształów, ..., oddziaływanie np. światła z cząsteczkami, ...

- Mechanika kwantowa, Równanie Schrodingera, chemia kwantowa,
- Kto wykorzystuje takie metody?: chemicy, biolodzy, fizycy, przemysł chemiczny, farmaceutyczny, nanotechnologia, ..., wojsko – ekologiczne paliwo dla rakiet balistycznych ;-)

Informatyka stosowana

- Bezpieczeństwo danych elektronicznych (także sieci LAN i Internet)
- Tworzenie i wdrażanie oprogramowania
- Metody numeryczne
- Administracja systemami sieciowymi

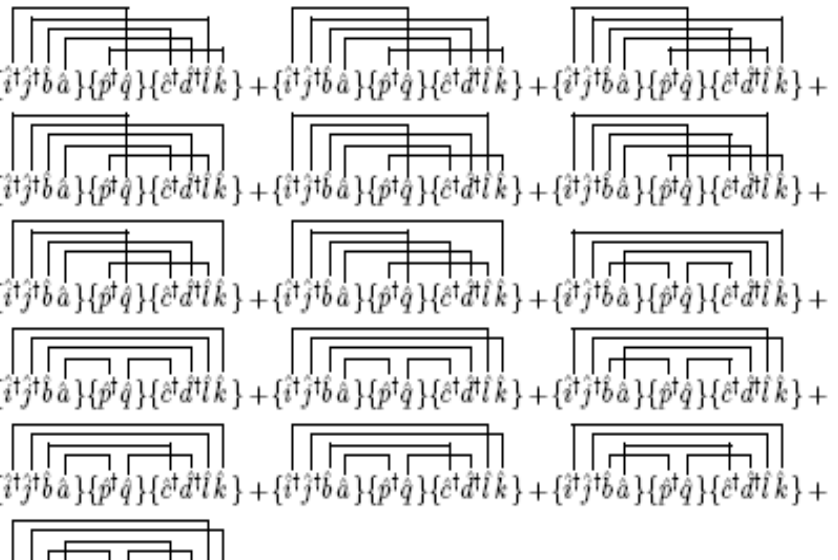
$$V'_{Dc\sigma}(\mathbf{r}) = \sum_{pq}\Big\{\sum_{ijab}\frac{2(ia|jb)-(aj|bi)}{\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b}\Big[2\sum_c\frac{(ca|jb)}{\varepsilon_i-\varepsilon_c}(ic|q) + 2\sum_k\frac{(ik|jb)}{\varepsilon_a-\varepsilon_k}(ka|q)\Big]\Big\}(\mathbf{X}^{-1}_{s\sigma})_{pq}g_p(\mathbf{r}). \quad (22)$$

$$V''_{Dc\sigma}(\mathbf{r}) = \sum_{pq}\Big\{\sum_{ijab}\frac{2(ia|jb)-(aj|bi)}{\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b}\Big[2\sum_{l\neq i}\frac{(la|jb)}{\varepsilon_i-\varepsilon_l}(il|q)+2\sum_{d\neq a}\frac{(id|jb)}{\varepsilon_a-\varepsilon_d}(da|q) - \frac{1}{2}\frac{(ia|jb)}{\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b}((ii|q)+(jj|q)-(aa|q)-(bb|q))\Big]\Big\}(\mathbf{X}^{-1}_{s\sigma})_{pq}g_p(\mathbf{r}) \quad (23)$$

$$V'_{Sc\sigma}(\mathbf{r}) = \sum_{pq}\Big\{\sum_{ia}\frac{f_{ia}}{\varepsilon_i-\varepsilon_a}\Big[2\sum_c\frac{(ci|q)}{\varepsilon_i-\varepsilon_c}f_{ca}+2\sum_k\frac{(ka|q)}{\varepsilon_a-\varepsilon_k} + 2\sum_{kc}\frac{(ck|q)}{\varepsilon_k-\varepsilon_c}(4(ia|ck)-(ic|ka)-(ik|ca))\Big]\Big\}($$

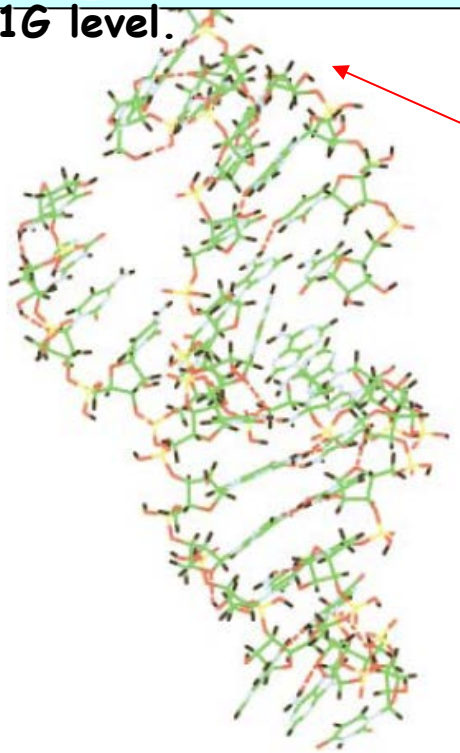
$$H\cdot\psi = E\cdot\psi$$

$\langle 0|\{\hat{i}^\dagger\hat{j}^\dagger\hat{b}\,\hat{a}\}\{\hat{p}^\dagger\hat{q}\}\{\hat{c}^\dagger\hat{d}^\dagger\hat{l}\,\hat{k}\}|0\rangle =$ (3.18)

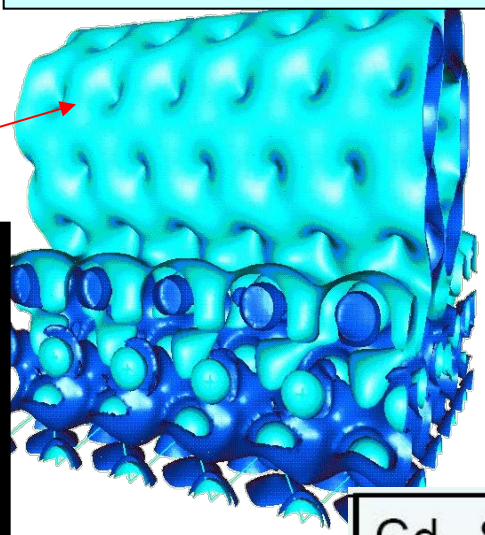


$$= \langle 0|\ [\ \delta_{iq}\delta_{jl}\delta_{bc}\delta_{ad}\delta_{pk}-\delta_{iq}\delta_{jl}\delta_{ac}\delta_{bd}\delta_{pk}-\delta_{iq}\delta_{jk}\delta_{bc}\delta_{ad}\delta_{pl}+\delta_{iq}\delta_{jk}\delta_{bd}\delta_{ac}\delta_{pl}+ \quad (3.19)$$
$$+\ \delta_{il}\delta_{jq}\delta_{bd}\delta_{ac}\delta_{pk}-\delta_{il}\delta_{jq}\delta_{bc}\delta_{ad}\delta_{pk}-\delta_{ik}\delta_{jq}\delta_{bd}\delta_{ac}\delta_{pl}+\delta_{ik}\delta_{jq}\delta_{bc}\delta_{ad}\delta_{pl}-$$
$$-\ \delta_{ik}\delta_{jl}\delta_{bp}\delta_{ad}\delta_{qc}+\delta_{ik}\delta_{jl}\delta_{bd}\delta_{ap}\delta_{qc}-\delta_{il}\delta_{jk}\delta_{bd}\delta_{ap}\delta_{qc}+\delta_{il}\delta_{jk}\delta_{bp}\delta_{ad}\delta_{qc}-$$
$$-\ \delta_{ik}\delta_{jl}\delta_{bc}\delta_{ap}\delta_{qd}+\delta_{il}\delta_{jk}\delta_{bc}\delta_{ap}\delta_{qd}+\delta_{ik}\delta_{jl}\delta_{bp}\delta_{ac}\delta_{qd}-\delta_{il}\delta_{jk}\delta_{bp}\delta_{ac}\delta_{qd}\ ]\ |0\rangle$$

·1026 atom fragment of RNA, calculated using linear scaling DFT at the LSDA/3-21G level.



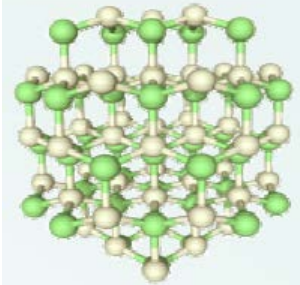
·Total electron density n(r) for a carbon nanotube on an Aluminum slab from an LDA simulation

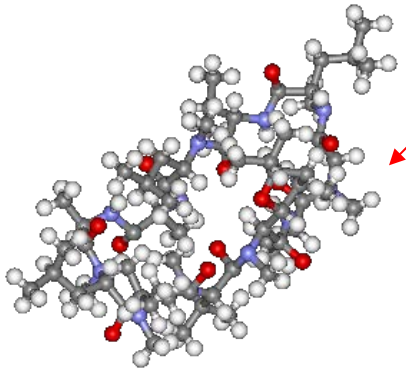


·Gaussian	BLYP
·NwChem	LDA
·Gamess	B3LYP
·Crystal	
·Wien2K	6-31G*
·Amber	augcc-PVDZ
·Q-Chem	
·VASP	

x1000 procesorów

$Cd_{45}Se_{45}$
1.5 nm





·Cyclosporin (DFT B3LYP):
·Basis: 6-31G*

·Accurate DFT calculations are required to accurately predict the electronic
·structure of CdSe quantum dots. (LDA, PBE)

Superkomputery w fizyce i życiu codziennym

www.top500.org

Porównanie mocy obliczeniowych komputerów

- Fugaku, Japan, Riken 455 PLOPS
- …
- Tianhe-2 (MilkyWay-2) 54,3 PFLOPS
- Titan - Cray XK7 – 27 PFLOPS
- Sequoia - 16,32 PFLOPS

K computer -10,51 PFLOPS

Prometeus z AGH -1,6 PFLOPS

IF UMK – 2 TFLOPS

IPad 2 - 1,6 Gflops

Core i7 965 XE to 70 Gflops

Plan wstępny (tylko wybrane zagadnienia -minimum)

- Błędy w obliczeniach numerycznych
- Znajdowanie miejsc zerowych równań nieliniowych.
  - metoda bisekcji
  - metoda stycznych
  - metoda siecznych
- Różniczkowanie numeryczne
  - wzory dwupunktowe, trójpunktowe i pięciopunktowe
- Całkowanie numeryczne
  - metoda trapezów
  - metoda Simpsona
- Rozwiązywanie układów równań liniowych
  - metoda eliminacji Gaussa
  - metoda LU
- Metody iteracyjne rozwiązywania zagadnień numerycznych.
- Interpolacja
- Równania różniczkowe
- Zagadnienie własne

Literatura

- Majchrzak E., Mochnacki B.*: Metody numeryczne. Podstawy teoretyczne, aspekty praktyczne i algorytmy*, Wydawnictwo Politechniki Śląskiej, wyd. IV, Gliwice 2004.
- Legras J.: Praktyczne metody analizy numerycznej, WNT, Warszawa 1974
- Wanat K.: Algorytmy numeryczne, Wyd. Dir, Gliwice 1993
- Bjorck A., Dahlquist G.: Metody numeryczne, PWN, Warszawa 1987
- FORTUNA Z., MACUKOW B., WĄSOWSKI J., *Metody numeryczne, WNT,* Warszawa, 2003
- J. Stoer, “Wstęp do metod numerycznych”
- D. Kincaid, W. Cheney *Analiza numeryczna*, Wydawnictwa Naukowo-Techniczne, Warszawa 2006.
- Każda książka, która ma w tytule „Metody numeryczne”
- Google: numerical recipes, metody numeryczne
- http://www.nr.com/
- Google
- http://www.fizyka.umk.pl/~ig/DYDAKTYKA/metody_numeryczne_2018_19/

Zaliczenia

Zaliczenie przedmiotu - egzamin pisemny

Ćwiczenia – zaliczenie na podstawie kolokwiów

Obecność na ćwiczeniach obowiązkowa

Ocena z ćwiczeń liczy się do oceny końcowej przedmiotu z wagą **0,4**

Na ćwiczeniach programowanie w dowolnym języku programowania – C, FORTRAN, MATLAB (zależy od prowadzącego)

i w dowolnym środowisku

Visual C++ … Express – darmowe środowisko do C++

Prezentacja pełni tylko funkcję pomocniczą

- Obecność na wykładach nieobowiązkowa, ale wskazana
- Mogą się pojawić prace domowe
- http://www.fizyka.umk.pl/~ig/DYDAKTYKA/metody_numeryczne_2021_22/

Motywacja

Metody numeryczne są jedną dziedzin matematyki stosowanej, które maja powszechne zastosowanie w praktyce w bardzo wielu dziedzinach – matematyka, chemia, ekonomia, inżynieria, itp... Wykorzystywane są wówczas, gdy badany problem nie ma w ogóle rozwiązania analitycznego (danego wzorami), lub korzystanie z takich rozwiązań jest uciążliwe ze względu na ich złożoność (analityczne rozwiązania unikatowe) lub koszty numeryczne.

Otrzymywane tą drogą (wykorzystania metod numerycznych) wyniki są na ogół przybliżone ale dokładność obliczeń może być z góry określona i dobiera się ją zależnie od potrzeb.

Motywacja

Metody numeryczne – metody rozwiązywania problemów matematycznych za pomocą operacji na liczbach (metody arytmetyczne) (te operacje wykonywane są przez komputery).

2+2, 3/2, 3/2.0, …

Metody numeryczne to sztuka doboru spośród wielu możliwych procedur takiej, która jest „najlepiej” dostosowana do rozwiązania danego zadania - uwzględniająca dokładność wyników, sprzęt, oprogramowanie, czas, koszty, itp.

https://www.wolframalpha.com

Motywacja

Obecnie dostępnych jest wiele programów komputerowych (np. Matlab, Mathematica, …) zawierających dużą liczbę gotowych procedur numerycznych, oraz gotowe rozwiązania(procedury), które są dostępne w postaci pojedynczych instrukcji, procedur lub bibliotek (BLAS, LAPACK i ATLAS).

```
DOUBLE PRECISION FUNCTION DNRM2 ( N, X, INCX )

 * .. Scalar Arguments .. INTEGER INCX, N

 * .. Array Arguments .. DOUBLE PRECISION X( * )

 * ..

* DNRM2 returns the euclidean norm of a vector via the functionname, so that

* DNRM2 := sqrt( x'*x ) *
```

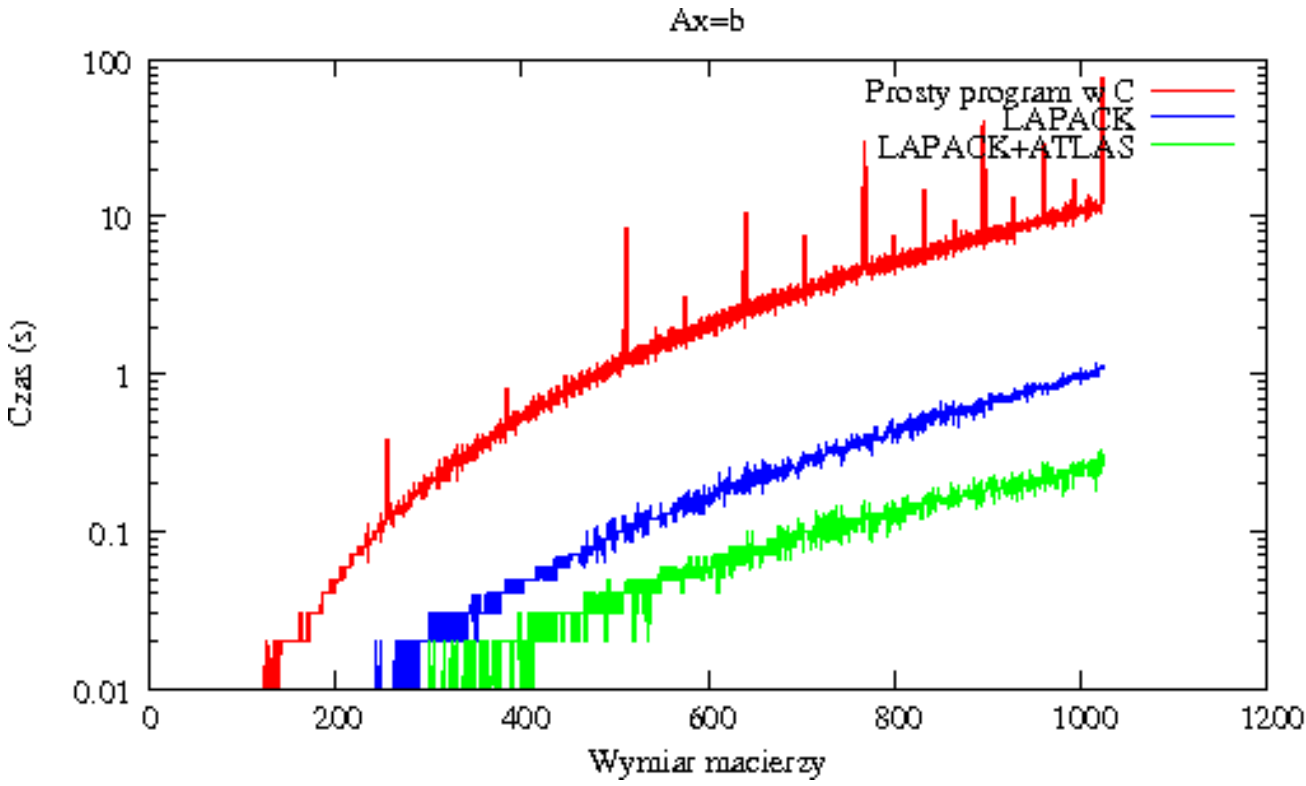
Motywacja

Jednak każdy użytkownik tego typu oprogramowania specjalistycznego natrafia na problemy związane z wyborem odpowiednich procedur (często do rozwiązania tego samego zadania można zastosować różne algorytmy), interpretacji błędów, dokładności wyników, rozwiązywania zagadnień niestandardowych, problemów z otrzymaniem poprawnego rozwiązania, czasu obliczeń, dostępności zasobów, czy wreszcie rozumienia i interpretacji tekstu instrukcji.

Ważna jest także umiejętność formułowania modeli matematycznych analizowanych zjawisk, które pozwalają określić poszukiwane parametry lub zależności między nimi. W takich przypadkach wymagana jest znajomość zagadnień analizy numerycznej.

Motywacja

Porównanie czasu działania kodu w C, implementującego algorytm rozkładu LU z czasem działania procedury DGESV z LAPACKa, niezoptymalizowanej zoptymalizowanej (ATLAS) na daną architekturę.



Motywacja

Najczęściej metody numeryczne stosuje się w:

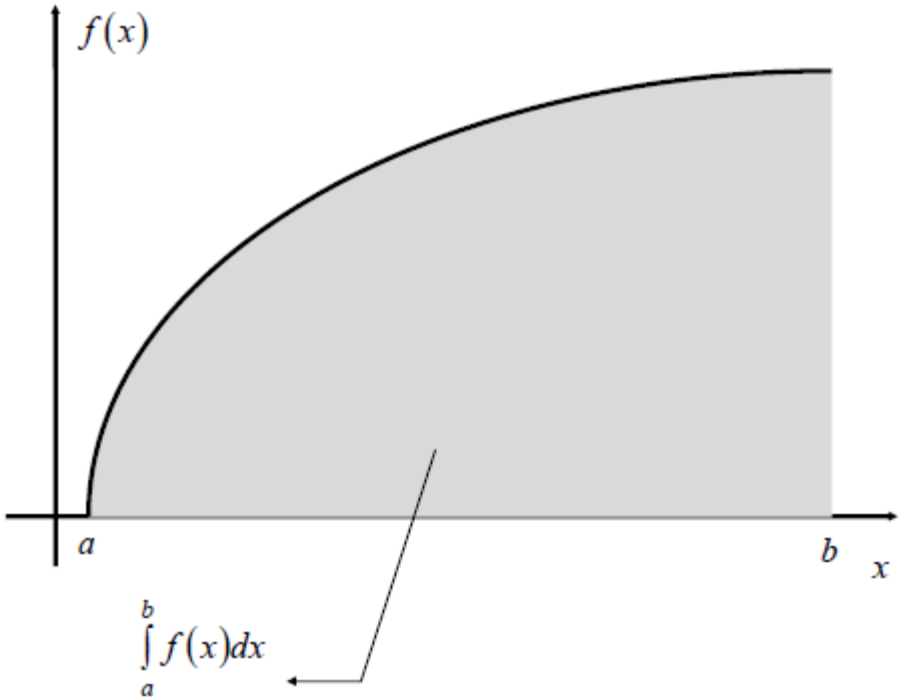
- rachunku macierzowym (obliczanie macierzy odwrotnej, wyznacznika, wartości własnych),
- rozwiązywaniu równań (metoda bisekcji, cięciw, stycznych, Newtona, itd.),
- rozwiązywaniu układów równań (metody dokładne i iteracyjne dla układów równań liniowych i nieliniowych)
- aproksymacji funkcji jednej lub wielu zmiennych,
- interpolacji funkcji (interpolacja wielomianowa, Lagrange'a, Newtona, Czebyszewa, trygonometryczna, itd.)
- całkowaniu całek pojedynczych i podwójnych(metoda prostokątów, trapezów, Simpsona, kwadratur i kubatur Gaussa),
- różniczkowaniu (metoda Lagrange'a, Newtona),
- rozwiązywaniu równań różniczkowych zwyczajnych (wzór Eulera).

Przykłady

- całkowanie numeryczne (wynikiem jest liczba!)

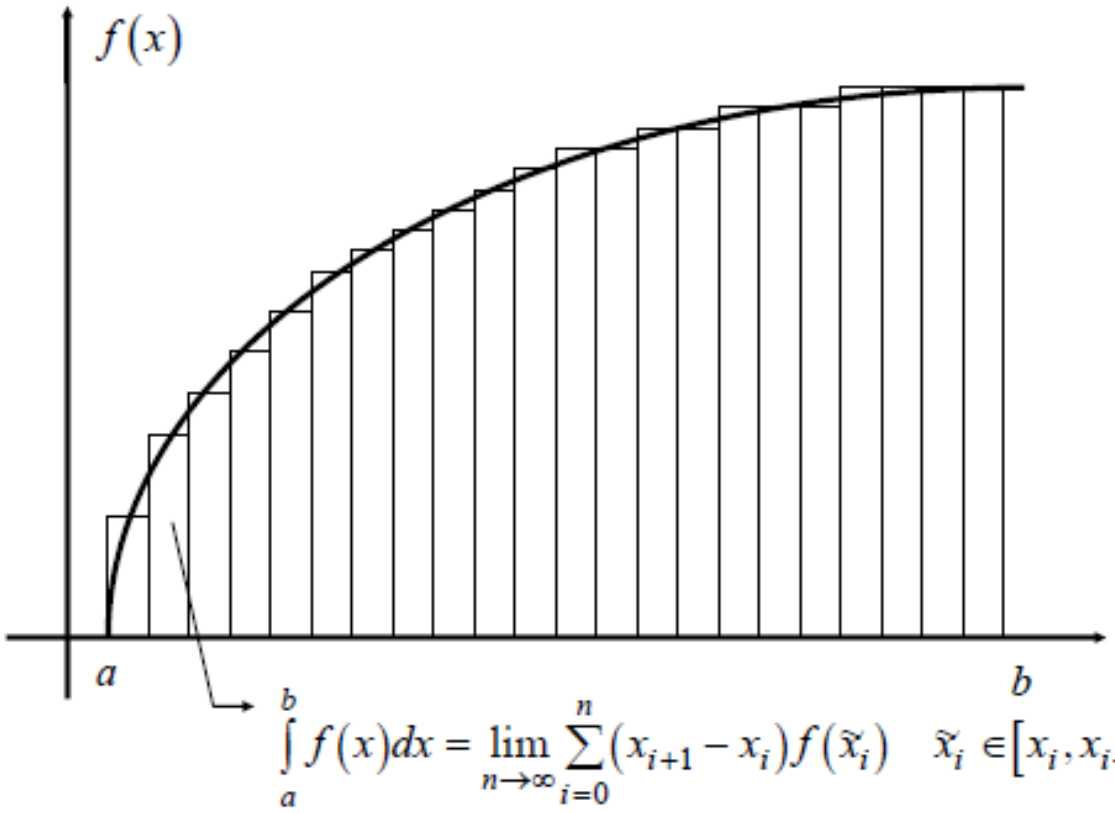
$$\int_a^b f(x)\,dx$$

$$\int_0^{2\Pi} \frac{\sqrt[4]{\sin(3x)}}{\ln(2x+\sin x)\arctan x}$$



Przykłady

- całkowanie numeryczne



$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=0}^{n} (x_{i+1} - x_i) f(\tilde{x}_i) \quad \tilde{x}_i \in [x_i, x_{i+1}]$$

Przykłady

- różniczkowanie numeryczne

$f(x)=\ln(3x)\sin 4x$ dla $x=0.0072$

$f`(x_0)= ?$ (liczba!)

$f``(x_0)= ?$

$f``(x_0)= ?$

Przykłady

• rozwiązywania układów równań liniowych w przypadku większej liczby równań i niewiadomych (np. miliony równań)

$$\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}$$

$$Ax = b, \qquad \det A \neq 0$$

Przykłady

- znajdowanie miejsc zerowych wielomianów stopnia większego niż 2

(korzystanie ze wzorów na pierwiastki równań stopnia 3 i stopnia 4 jest niepraktyczne, dla równań stopnia wyższego niż 4 wzorów już nie ma)

- znajdowanie miejsc zerowych równań nieliniowych i rozwiązywanie układów takich równań

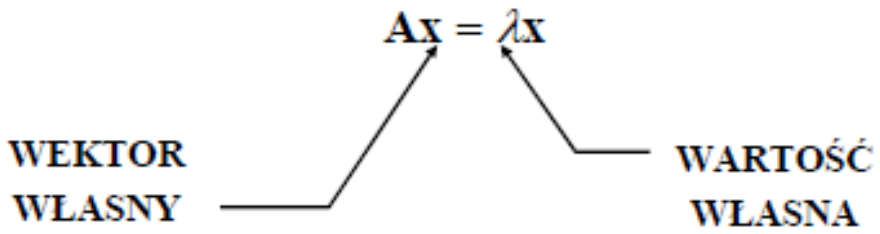
$$\sin 2x + 2\operatorname{tg} x - 4x^2 = 0$$

Przykłady

- rozwiązywania równań różniczkowych i układów takich równań

  y`-3xy=sin3x

- znajdowanie wartości i wektorów własnych



Przykłady

$$\hat{H}\Psi = E\Psi$$

$$\downarrow \text{Hartree-Fock}$$

$$\begin{pmatrix} F_{11} & F_{12} & \cdots & F_{1N} \\ F_{21} & F_{22} & \cdots & F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ F_{1N} & F_{2N} & \cdots & F_{NN} \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1N} & c_{2N} & \cdots & c_{NN} \end{pmatrix} = \begin{pmatrix} \varepsilon_{11} & 0 & \cdots & 0 \\ 0 & \varepsilon_{11} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \varepsilon_{11} \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1N} & c_{2N} & \cdots & c_{NN} \end{pmatrix}$$

Przykłady

- Metody iteracyjne
  - Ciąg kolejnych przybliżeń $\{x_i\}$.
  - n-punktowy wzór iteracyjny

$$x_{i+1} = F_i(x_i, x_{i-1}, \ldots, x_{i-n+1}).$$

Przykład:

Metoda iteracji prostej

Jedną z najprostrzych metod iteracyjnych jest metoda iteracji prostej. Polega ona na przejściu od układu równań liniowych

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

do równoważnego układu (mającego te same rozwiązania)

$$\boldsymbol{Q}\boldsymbol{x} = (\boldsymbol{Q} - \boldsymbol{A})\boldsymbol{x} + \boldsymbol{b}$$

Znając powyższe równanie wyznaczamy ciąg przybliżeń rozwiązania $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ ze wzoru

$$\boldsymbol{Q}\boldsymbol{x}^{(k)} = (\boldsymbol{Q} - \boldsymbol{A})\boldsymbol{x}^{(k-1)} + \boldsymbol{b} \quad (k \geq 1),$$

gdzie $\boldsymbol{x}^{(0)}$ jest danym przybliżeniem początkowym.

Etapy rozwiązywania problemu

Realny problem wymagający zastosowania metod numerycznych – kroki:

1. Odpowiednie sformułowanie zadania (metoda)
2. Metoda numeryczna + analiza błędu
3. Algorytm
4. Implementacja
5. Testy numeryczne (na modelach i w znanych przypadkach, tzw. benchmark calculations)
6. Optymalizacja (sprzęt, metoda, algorytm)
7. Realistyczne obliczenia

Analiza problemu za pomocą komputera

Diagram : rodzaje błędów

i miejsca ich powstawania

Problem inżynierski, naukowy

błąd przybliżenia zagadnienia

Model matematyczny

błąd modelu

Metoda numeryczna

błąd metody numerycznej

Algorytm numeryczny i implementacja

Błędy: danych wejściowych, obcięć, zaokrągleń,, programowania, ….

Błędy obliczeń numerycznych

- błędy zaokrągleń w czasie obliczeń ^

  - skończona liczba cyfr (bitów) w reprezentacji numerycznej

- błędy obcięcia ^

  - rozwinięcia w szeregi i procesy iteracyjne - w praktyce muszą być skończone

- błędy metody

  - przybliżone rozwiązania problemu

- błędy modelu

- błędy danych wejściowych (gdy wykorzystujemy dane zaokrąglone, pochodzące np. z wcześniejszych obliczeń lub gdy dane wejściowe są wynikiem pomiarów wielkości fizycznych obarczonych błędami pomiarowymi)

Błędy obliczeń numerycznych

- pomyłki (błąd człowieka)
- niestabilność numeryczna problemu
- złe uwarunkowanie numeryczne
- ograniczenia sprzętowe (CUDA Compute Unified Device Architecture -Nvidia)
- Ograniczenia oprogramowania (kompilatory, biblioteki, języki programowania, programy komercyjne)

Błędy w obliczeniach numerycznych

Zaokrąglenia

1. nieskończone reprezentacje dziesiętne

   1/3=0.333333333333333....

2. nieskończone reprezentacje binarne

   skończonych reprezentacji dziesiętnych

$(0.1)_{10} = (0.0001100110011001100110011001100 \ldots)_2$

0.09999999 ...

3. **Operacje na skończonej liczbie bajtów**

Ej, jesteś dobry z matmy nie?

No

To patrz... jak pokroję ciasto na 3 części to każda będzie 0,333 całości tak?

Zgadza się

Ok, a jak pomnożę 0,333 przez 3 to mam 0,999

Gdzie się podziało 0,001?

Zostało na nożu

A ok dzięki

Błędy w obliczeniach numerycznych

błędy obcięcia

pewne wielkości wyliczane są zawsze

z nieskończonych sum, szeregów czy iteracji

np:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

przykład unikania utraty dokładności

$$f(x) = x - \sin(x)$$

dla $x \approx 0$ $x \approx \sin(x)$ (bliskie wartości)

$$f(x) = \frac{x^3}{3!} - \frac{x^5}{5!} + \frac{x^7}{7!} - \ldots$$

Uwarunkowanie zadania jest cechą metod numerycznych, która określa możliwość uzyskania poprawnych wyników przy stosowaniu dowolnych danych wejściowych z odpowiednio zdefiniowanego zbioru. Jeśli analizowany algorytm służy do rozwiązania zadania $y = w(x)$, *to stopień uwarunkowania zadania można* mierzyć za pomocą ilorazu $(w(x + \delta x) - w(x)) / wx$.

Mówi się, zatem, że zadanie jest *dobrze uwarunkowane lub źle uwarunkowane. W pierwszym przypadku zadanie jest stabilne względem danych* wejściowych,

W przypadku złego uwarunkowania zadania, możliwość uzyskania poprawnego rozwiązania w sposób kluczowy zależy od wartości danych wejściowych.

problemy źle uwarunkowane

(np. przy zadanej dokładności numerycznej)

układ równań

$$6.025\,x + 109.774\,y = 4.12$$

$$14.054\,x + 256.125\,y = 10.50$$

ma rozwiązanie: $x = -250.15$ $y = 13.77$;

jeśli dysponujemy dokładnością do 2 miejsc po kropce

i wyniki pośrednich operacji zaokrąglamy do 2 miejsc po kropce, to rozwiązaniem jest

$$x = -44.33 \qquad y = 2.47 \quad !!!$$

Zadania źle uwarunkowane

Definicja 1. *Jeśli niewielkie względne zmiany danych powodują duże względne odkształcenia wyników, to zadanie nazywamy źle uwarunkowanym.*

Przykład 1. *Zbadajmy uwarunkowanie zadania obliczania wartości funkcji* $f(x) \neq 0$ *w punkcie* x. *W tym celu zaburzymy dane* x, $\tilde{x} = x(1+\delta) = x + x\delta$. *Oszacujmy względną zmianę wyniku.*

$$\frac{|f(\tilde{x}) - f(x)|}{|f(x)|} = \frac{|f(x + x\delta) - f(x)|}{|f(x)|} \approx \frac{|f'(x)x\delta|}{|f(x)|} = cond(x)|\delta|,$$

$cond(x) = |f'(x)x|/|f(x)|$ *jest wskaźnikiem uwarunkowania zadania. Jeśli* $cond(x)$ *jest mały (*$|f'(x)|$ *jest mała), to zadanie jest dobrze uwarunkowane.*

Definicja 2. *Wielkości charakteryzujące wpływ zaburzeń danych na odkształcenia wyników nazywamy wskaźnikami uwarunkowania.*

Niestabilność numeryczna procesu

Mówiąc nieformalnie, proces numeryczny jest niestabilny jeśli niewielkie błędy, popełnione w początkowym (lub pośrednim) stadium procesu kumulują się w kolejnych stadiach, powodując poważną utratę dokładności obliczeń.

Przykład 6. *Rozważmy ciąg liczb rzeczywistych zdefiniowany za pomocą rekurencyjnego związku:*

$$\begin{cases} x_0 = 1 \quad x_1 = \frac{1}{3} \\ x_{n+1} = \frac{13}{3}x_n - \frac{4}{3}x_{n-1} \quad (n \geqslant 1). \end{cases}$$

Powyższy związek generuje ciąg $x_n = \left(\frac{1}{3}\right)^n$.

Dla $n = 0$ *i* $n = 1$ *oczywiste. Załóżmy równoważność jest spełniona dla* $n \leqslant m$. *Równoważność dla* $n = m + 1$ *wynika*

$$\frac{13}{3}x_m - \frac{4}{3}x_{m-1} = \frac{13}{3}\left(\frac{1}{3}\right)^m - \frac{4}{3}\left(\frac{1}{3}\right)^{m-1} = \left(\frac{1}{3}\right)^{m-1}\left(\frac{13}{9} - \frac{4}{3}\right) = \left(\frac{1}{3}\right)^{m+1}$$

Niestabilność numeryczna procesu

Poniżej 15 iteracji algorytmu (w arytmetyce `single`)

$$\begin{cases} x_0 = 1 \quad x_1 = \frac{1}{3} \\ x_{n+1} = \frac{13}{3} x_n - \frac{4}{3} x_{n-1} \quad (n \geqslant 1). \end{cases}$$

$x_0 = 1.0000000$		$x_8 = 0.0003757$	(0 cyfr)
$x_1 = 0.3333333$	(7 cyfr)	$x_9 = 0.0009437$	
$x_2 = 0.1111112$	(6 cyfr)	$x_{10} = 0.0035887$	
$x_3 = 0.0370373$	(5 cyfr)	$x_{11} = 0.0142927$	
$x_4 = 0.0123466$	(4 cyfr)	$x_{12} = 0.0571202$	
$x_5 = 0.0041187$	(3 cyfr)	$x_{13} = 0.2285939$	
$x_6 = 0.0013857$	(2 cyfr)	$x_{14} = 0.9143735$	
$x_7 = 0.0005131$	(1 cyfr)	$x_{15} = 3.657493$	(błąd względny 10^8)

Niedokładność x_n przenosi się na x_{n+1} z mnożnikiem $13/3$.

Zatem niedokładność x_1, rzędu 10^{-8}, przenosi się na x_{15} z wielkim mnożnikiem $(13/3)^{14} \approx 10^9$.

Niestabilny algorytm

W wielu przypadkach algorytm zastosowany do rozwiązania zadania dobrze uwarunkowanego (stabilnego) może być niestabilny. Stabilność numeryczna algorytmu odnosi się do możliwości uzyskania określonej dokładności obliczeń.

Algorytm jest stabilny numerycznie, gdy zwiększając dokładność obliczeń można z dowolną dokładnością określić dowolne z istniejących rozwiązań.

Błąd bezwzględny i względny

Niech $\tilde{x}$ będzie oszacowaniem wartości x (*dokładnej*)

- Błąd bezwzględny

$$\Delta x = |\tilde{x} - x|$$

- Błąd względny

$$\delta x = \frac{|\tilde{x} - x|}{x}$$

Reprezentacja liczb

- Reprezentacja stałopozycyjna
  - w praktyce tylko dla liczb całkowitych
- Reprezentacja zmiennopozycyjna (zmiennoprzecinkowa)
  - mantysa i cecha, liczba $=$ mantysa $* 2^{\text{cecha}}$
    $18.5 = 0.185 * 10^2 = 0.100101 * 2^{101}$
  - rozdzielczość i skończoność komputerowych liczb rzeczywistych
  - typy 4-bajtowe (3 bajty mantysy i 1 bajt cechy), 6-bajtowe, 8-bajtowe (podwójnej precyzji)
  - należy być ostrożnym przy dodawaniu małej liczby zmiennoprzecinkowej do dużej (mała może zniknąć)
  - wartości, które nie są poprawnymi liczbami (NAN - not a number)

Rodzaje zmiennych c++

int - liczby całkowite
float - liczby zmiennoprzecinkowe pojedynczej precyzji
double - liczby zmiennoprzecinkowe podwójnej precyzji

•Typy zmiennoprzecinkowe

Do przechowywania liczb ułamkowych używamy zmiennych typu zmiennoprzecinkowego (ang. floating point type). W typach zmiennoprzecinkowych ważniejszym parametrem od zakresu jest tzw. precyzja, która określa dokładność zapamiętywania liczb.

float

32 bitowe liczby zmiennoprzecinkowe o pojedynczej precyzji.
Precyzja 7-8 cyfr.
Każda zmienna float zajmuje w pamięci komputera 4 bajty.

double

64 bitowe liczby zmiennoprzecinkowe o podwójnej precyzji.
Precyzja 15 cyfr.
Każda zmienna double zajmuje w pamięci komputera 8 bajtów.

Komputer działa na 0 i 1

- 1 **bit** – miejsce na zapamiętanie znaku 0 bądź 1
- 1 **bajt** = 8 bitów – pozwala zapamiętać 2^8 różnych wartości, a więc np. liczby 0-255
- 1 **kB** (kilobajt) = 2^{10} B = 1024 B
- 1 **MB** (megabajt) = 2^{20} B = 1048576 B
- 1 **GB** (gigabajt) = 2^{30} B = 1073741824 B
- 1 **słowo** to ilość informacji przetwarzanej przez procesor w jednym cyklu (procesor 32 bitowy = procesor o słowie 32 bitowym)

Efektywność numeryczna algorytmu

Analiza szybkości danego algorytmu (metody) — polega na szacowaniu liczby operacji zmienno-przecinkowych niezbędnych do zakończenia obliczeń komputerowych.

Złożoność obliczeniowa algorytmu jest związana z liczbą operacji numerycznych, które prowadzą do uzyskania wyniku. Jest zrozumiałe, że spośród różnych algorytmów, które zapewniają poprawne rozwiązanie, należy wybierać te, które charakteryzują się małą złożonością obliczeniową. Jest to szczególnie istotne np. układach sterowania, gdzie pełny cykl obliczeń numerycznych musi być wykonany w czasie określonym przez okres pomiędzy kolejnymi pomiarami wielkości wejściowych.

Oprogramowanie

Oprogramowanie — procedury, programy, biblioteki numeryczne; powinny być **łatwe w użyciu**, **niezawodne** i **szybkie**. Różne paradygmaty programowania:

- programowanie przy użyciu bibliotek (ogólnie dostępnych lub komercyjnych); zapewnia niezawodność i szybkość, ale wymaga od użytkownika wiedzy w zakresie sposobu posługiwania się stosownymi procedurami (dotyczy to zwłaszcza sposobu wprowadzania danych i kolejności wywoływania podprogramów).
- Rozwiązywanie danego zagadnienia przy pomocy środowisk obliczeniowych lub pakietów programów typu: MATLAB, MATHCAD, MATHEMATICA, DERIVE, REDUCE.
- Programowanie blokowe — polega na składaniu programu komputerowego z gotowych podprogramów zwanych blokami (znanych w literaturze angielskiej pod nazwą *templates* lub *recipies*).

Program komputerowy

Każdy program komputerowy powinien robić to, do czego jest przeznaczony.

Pisząc program trzeba więc zacząć od precyzyjnego przedstawienia problemu oraz scharakteryzowania dopuszczalnych danych wejściowych oraz oczekiwanych wyników jako funkcji danych wejściowych.

Jak można programować komputery?

- pisanie ciągów bitów
- symboliczne języki programowania
- algorytmiczne języki programowania wysokiego poziomu

Program komputerowy

Program komputerowy to algorytm napisany w języku programowania.

Program – zbiór poleceń wybranych zgodnie z dopuszczalnymi przez dany język programowania regułami.

Pogram tłumaczy języki symboliczne na języki maszynowe – jest uruchamiany i wykonywany na konkretnym komputerze (w konkretnym środowisku) w konkretnym celu, przetwarzając dane wejściowe i dając konkretny wynik.

Języki programowania

Język programowania, zbiór zasad składni, instrukcji, dzięki którym powstaje kod źródłowy programu.

Języki maszynowe i symboliczne języki programowania: zbiór abstrakcyjnych definicji i reguł syntaktycznych, które potrafimy przełożyć na kod maszynowy.

Programy tłumaczące języki symboliczne na języki maszynowe to translatory

Translacja programu na kod maszynowy

Interpretery - każde polecenie jest na bieżąco zamieniane na kod maszynowy, łatwa praca interakcyjna. Interpreter - realizuje translację instrukcji naprzemiennie z ich wykonywaniem; przy zastosowaniu interpretatora każde wykonanie programu jest związane z jego ponowną translacją (np. Basic, SQL, php, html – przeglądarka internetowa).

Kompilatory - cały program przekładany jest na kod maszynowy, duża szybkość wykonywania programu – kompilacja programu

Konsolidacja (linkowanie):

Polega na scaleniu binarnych fragmentów programu w jedną całość i dołączeniu procedur systemowych, procedur wejścia/wyjścia, funkcji matematycznych z bibliotek systemowych i innych elementów koniecznych do działania programu.

Algorytm

Definicja programu komputerowego jest ściśle związana z pojęciem algorytmu.

Algorytm – sformalizowany sposób zapisu i prezentacji metody rozwiązania określonego problemu

Algorytm – zbiór określonych reguł postępowania, które realizowane zgodnie z ustalonym porządkiem umożliwiają rozwiązanie określonego zadania. Jego ważną cechą jest możliwość wykorzystania tego samego algorytmu do rozwiązania całej grupy podobnych problemów.

Przykłady algorytmów:

- Obliczenie pierwiastków równania kwadratowego
- Znajdowanie liczby największej w danym zbiorze liczb
- Gotowanie jajka na miękko

Algorytm – sposób przetwarzania danych wejściowych na dane wyjściowe (wyniki) w skończonej liczbie kroków.

Algorytm definiuje:

abstrakcyjne obiekty, na których wykonywane są działania, reprezentowane przez:

- odpowiednie struktury danych;
- operacje realizujące cel algorytmu;
- kolejność wykonywania działań.

Cechy algorytmów

- **Ogólność -** algorytm przeznaczony jest do rozwiązywania określonej klasy zadań, a nie tylko pojedynczego szczególnego przypadku
- **Skończoność** - możliwość uzyskania rozwiązania problemu w skończonej liczbie kroków.
- **Określoność -** tzn. jednoznaczność wszystkich wykonywanych w nim operacji
- **Efektywność** - czas potrzebny na wykonanie tego algorytmu (najczęściej liczony w umownych jednostkach np. ilości wykonywanych operacji w zależności od danych wejściowych) lub zapotrzebowanie algorytmu na pamięć.

Etapy konstrukcji algorytmów

- określenie stanu wyjściowego
- ustalenie dziedziny dopuszczalnych operacji
- określenie celów pośrednich, niezbędnych do realizacji celu głównego
- budowa procedur realizujących cele pośrednie
- powiązanie procedur w jedną całość
- formalizacja algorytmu
- weryfikacja algorytmu

Tworzenie oprogramowania - narzędzia

Narzędzia konieczne do tworzenia oprogramowania:

- edytory narzędziowe (mogą to być zwykłe edytory tekstu)
- kompilatory i interpretery
- debuggery : programy, ułatwiające śledzenie i weryfikację poprawności wykonywania się danego programu
- programy wspomagające i tworzące środowisko pracy

Etapy rozwiązywania problemów z wykorzystaniem komputera

- specyfikacja problemu,
- określenie danych wejściowych,
- określenie celu (wyniku końcowego),
- analiza problemu i wybór modelu,
- synteza algorytmu prowadzącego do rozwiązania,
- przedstawienie algorytmu

- analiza poprawności rozwiązania,
- ocena efektywności algorytmu (złożoności obliczeniowej),
- kodowanie algorytmu w postaci instrukcji języka programowania
- zapis programu do pliku,
- kompilacja i usuwanie usterek,
- utworzenie wersji wykonywalnej,
- wykonanie programu w komputerze,
- testowanie i analiza wyników.

Visual C++ 2010 Express – darmowe środowisko do C++

// new project - empty

// source files - cpp

Ćwiczenia – program w c++

```
#include <iostream>
#include"conio.h"
#include <math.h>
// standard c++
using namespace std;
int main()
{
// komentarz
Polecenia;
}
```

Ćwiczenia – program w c++

```
int main()
{
    int ib=1;            // deklaracja i inicjalizacja zmiennej
    cout << "aaaaa" << endl;            // średniki, wydruk na ekran
    cout << ib << endl;         // przejście do nowej linii
    int a=0;
    // petla
    for (int i=0; i<=2; i++)
    {
        cout << "podaj a" << endl << "a=";
        cin >> a;        // wczytanie wartości na zmienną a
        cout << a << endl;
    }
  _getch();      // „przytrzymanie ekranu”
    return 0;
}
```

Ćwiczenia – program w c++

```
#include <iostream>
…
double f(double);      // deklaracja funkcji o nazwie f
double fp(double);
int main()
{
            double wynik,h,x;
            cout.precision(20);
……
          wynik=h*f(x)   // wywołanie funkcji i obliczenie wart. zmiennej wynik
}

double f(double x) // definicja funkcji  f
{
            return sin(float(x));
}
```

Ćwiczenia – program w c++

```
int main()   // Instrukcja warunkowa
{
    int wiek;
     cout << "podaj liczbe" << endl;
    cin >> wiek;
    if( wiek >= 18 && wiek <=45 ) // and
    {
        cout << "masz wiecej niz 18 a mniej niz 45" << endl;
    }
    else
    {
      cout << "masz mniej niz 18 lub wiecej niz 45" << endl;
     }
    getch();
    return 0;
}
```

Ćwiczenia – program w c++

```
// pętla for
    for (int i=0; i<=2; i++)
    {
       n=n/(2*i);
       cout  << „numer iteracji” << i << endl;
       cout  << „wartość n” << n << endl;
    }
// pętla while
```

break

continue