Є.Ф. Вінниченко
Чернігівський педагогічний університет
ім.Т.Г. Шевченка

Деякі особливості використання математичних програмних засобів на уроках математики

Використання комп'ютера в учбовому процесі, і, зокрема, педагогічних програмних засобів при вивченні математики, ґрунтовно увійшло в повсякденне життя. Усе більше навчальних закладів оснащується сучасними комп'ютерними класами, з'являються нові та вдосконалюються вже існуючі програми, розробляються відповідні методики. Але серед захоплених відгуків про використання комп'ютерних навчальних програм різних типів, іноді зустрічаються і критичні оцінки. Однак з точки зору методики навчання краще, коли учень самостійно припускається помилок та виправляє їх при традиційному безкомп'ютерному навчанні, ніж коли він сліпо вірить в безпомилковість комп'ютера при розв'язуванні математичних задач.

Як відомо, ідеального програмного засобу не існує, кожен із них містить як переваги, так і недоліки. Одна справа, якщо це недоліки, які стосуються інтерфейсу, зручності введення даних та одержання результатів. Інша справа, якщо вони впливають на правильність одержаного результату. Розробники програмного забезпечення, методисти, досвідчені вчителі найчастіше знають і про самі недоліки, і про їх причини, та, якщо це можливо, намагаються їх усувати. Але не кожен вчитель (особливо, якщо він не є фахівцем з інформатики), а тим більше учень, здатен виділити ці недоліки, щоб обходити їх у своїй роботі. Вкажемо на деякі з них, з якими, на наш погляд, найчастіше зустрічаються користувачі при роботі з математичними програмами.

Найбільш відомими програмними продуктами для комп'ютерного супроводу розв'язування різноманітних задач з математики є Derive, Eureka, MathCad, MathLab, Mathematika, MapleV, Gran1, Gran-2D, Gran-3D, Advanced Grapher, Математика+ тощо. Однак не всі з них доцільно використовувати в школі. Так, наприклад, програма Mathematika є досить складним професійним програмним засобом, тільки для ознайомлення з яким потрібно проводити окремий цикл занять. Тому, для аналізу недоліків ми вибрали програми, що найбільш часто застосовуються в школі при вивченні курсу алгебри та початків аналізу: Gran1, Derive, Advanced Grapher, Математика+.

Помилкові результати, які одержує користувач внаслідок користування математичними програмами, можна поділити на дві основні групи:

- *недоліки програмного забезпечення*. Помилки цієї групи пов'язані з алгоритмами опрацювання даних, що використали розробники відповідного ПЗ;
- *помилки користувача*. Вони пов'язані з неадекватним застосуванням ПЗ або низьким рівнем знань та навичок користувача в конкретній предметній галузі.

Деякі з помилок можуть містити елементи обох груп. Послідовно розглянемо на прикладах кожен із типів таких помилок, розпочавши з недоліків власне програм.

Не зважаючи на великі досягнення комп'ютерної архітектури, все більшу складність та потужність сучасних ЕОМ, все ж комп'ютерна математика суттєво відрізняється від звичайної. Комп'ютер практично ніколи не оперує точними даними, якщо говорити про дійсні числа, – всі числа, що зберігаються в пам'яті, задані з певною точністю (наприклад, 14 знаків). Зокрема, в окремих програмних продуктах піднесення до степеня a^b відбувається з використанням основної логарифмічної тотожності $a^b = e^{b\ln a}$. Звичайно, що в цьому випадку повинно бути $a > 0$. Тому в деяких програмних засобах реалізовано алгоритми аналізу основи та показника степеня і, коли це потрібно, обчислення проводяться дещо інакше. Якщо ж основою або показником степеня виступає деякий вираз, то спочатку обчислюється значення цього виразу, а вже потім обчислення відбувається за вищевказаним принципом.

Таке попереднє спрощення виразу, звичайно, робиться не тільки для обчислення степеня, та воно може зіграти злий жарт із користувачем. Розглянемо функцію $y = \log_x x$ та побудуємо її графік. Якщо врахувати означення логарифма, то дану функцію можна подати так: $y = 1, x > 0, x \neq 1$ (рис.1). Введемо функцію $y = \log_x x$ в програму Derive у вигляді $y = \log(x, x)$. Побудувавши графік цієї функції, бачимо, що вираз в Derive спочатку було максимально спрощено, тобто замінено дану функцію "тотожною" $\log_x x = 1$. При цьому було безпідставно розширено область визначення функції, а за основу взято від'ємне число (рис.2). Звичайно, побудований графік не відповідає шуканому.

На жаль, не кожна математична програма містить функцію знаходження логарифма за будь-якою основою. В таких випадках потрібно користуватись формулою переходу $\log_b a = \dfrac{\ln a}{\ln b}$. Заміна функції $y = \log(x, x)$ на тотожною $y = \ln(x)/\ln(x)$ в програмі Derive залишає графік без змін. В деяких інших (наприклад, GRAN1 або Advanced Grapher) попереднього спрощення виразу не відбувається, отже значення функції обчислюються згідно введеного в програму виразу шляхом підстановки відповідного значення *x*. Саме тому графіки функцій, побудовані в цих програмах розташовані лише в додатній півплощині відносно осі *y* (рис.3). Відсутність точок розриву при $x = 0$ та $x = 1$ пояснюється іншими причинами. Єдиним програмним засобом, за яким правильно побудовано графік даної функції, була програма Математика+.

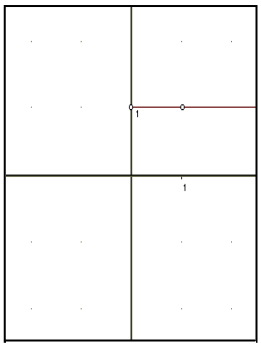

Рис. 1.

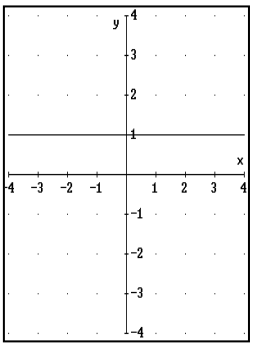

Рис. 2.

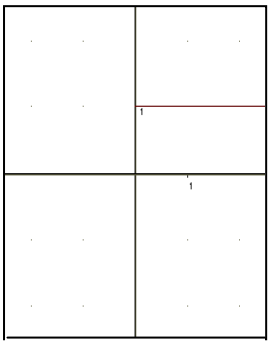

Рис. 3.

В школі для побудови графіка функції, як відомо, поступають наступним чином: будують таблицю залежності значень функції від значень аргументу, а потім відмічають відповідні точки на координатній площині та з'єднують їх плавною кривою. Майже аналогічно відбувається побудова графіків функціональних залежностей і в деяких математичних програмах: спочатку проводиться табулювання потрібної функції на заданому відрізку з певним кроком, який залежить від кількості точок побудови, одержані в процесі табулювання точки відображаються на координатній площині та послідовно з'єднуються відрізками. Чим більшу кількість значень аргументу на заданому відрізку взяти (тобто, чим меншим буде крок табулювання), тим більш точний графік можна одержати. Але такий алгоритм побудови працює правильно лише в тому випадку, коли функція є неперервною на заданому відрізку, оскільки для неперервної функції можна наближено заміняти відрізок дуги на відрізок прямої, що сполучає кінці цієї дуги. Якщо ж функція має точки розриву, то робити таку заміну в околі точки розриву не можна. Саме цим пояснюється відсутність точок розриву в попередньому прикладі. В окремих випадках точка розриву може співпадати з вузлом табулювання, тоді програма при побудові графіка функціональної залежності в околі такої точки графік не будує (тобто відображається точка розриву).

Наведемо ще декілька прикладів, коли наявність точок розриву суттєво впливає на побудову графіків функцій в комп'ютерних програмах. Розглянемо функцію $y = \frac{1}{|x|}$, графік якої, отриманий за допомогою програми GRAN1, подано на рис.4. Якщо, користуючись цією ж програмою, побудувати графік даної функції на відрізку $[-5;5]$, використовуючи лише 25 точок побудови, то одержимо графік, зображений на рис.5, що суттєво відрізняється від правильного.

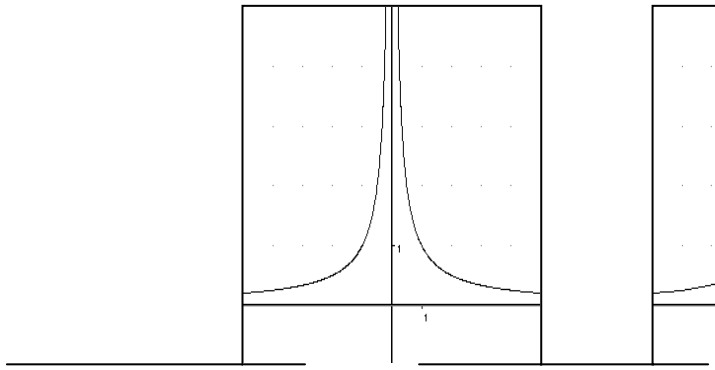

Рис. 4.

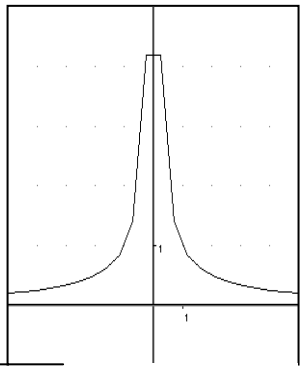

Рис. 5.

Як вже вказувалось, збільшення кількості точок побудови, веде до виправлення графіка. Однак такий шлях спрацьовує не завжди. Візьмемо функцію $y = \frac{1}{\sin\frac{1}{x}}$, яка має в околі точки $x = 0$ безліч точок розриву. Тому збільшення кількості точок побудови не дає помітного ефекту, оскільки на правильність побудови впливає вже не тільки крок табулювання, а і роздільна здатність монітору. Дещо покращити зображення графіка можна, наприклад, в програмі GRAN1, обравши послугу "Графік точками". При цьому графік буде складатись лише з окремих точок (вузлів табулювання), які не будуть з'єднуватись між собою (рис.6). В програмі Advanced Grapher існує послуга, за допомогою якої можна заборонити програмі з'єднувати дві послідовні точки, якщо абсолютне значення приросту функції на цьому відрізку перевищує наперед задане число. Але в кожному з цих випадків графік все одно одержується з невеликою точністю.

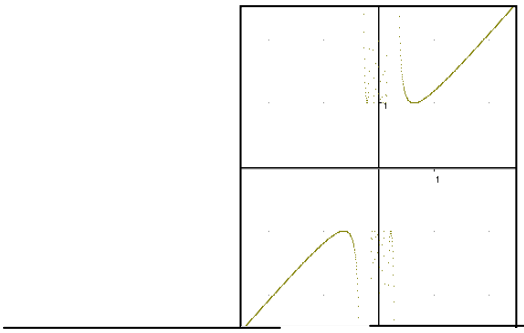

Рис. 6.

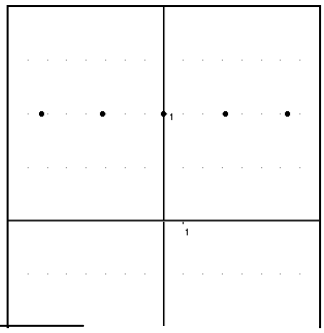

Рис. 7.

Якщо побудова графіків неперервних функцій, або таких, що мають велику кількість точок розриву, все ж таки можлива, то перед дискретними (кусково-сталими) функціями (з дискретними множинами значень) проаналізовані комп'ютерні програми залишаються безсилими. Розглянемо функцію $y = 1 + \sqrt{\cos x - 1}$ та знайдемо для неї область допустимих значень: $\cos x - 1 \geq 0$, $\cos x \geq 1$, $x = \pi n, n \in Z$. Отже, областю визначення даної функції буде множина точок $x = \pi n$, де n – ціле число. При цьому значення підкореневого виразу, а, відповідно, і кореня дорівнює 0. Тому функція в цих точках набуває вигляду $y = 1$ (рис.7). Побудувати графік такої функції в досліджених програмах неможливо. Наприклад в програмі Gran1 це пов'язано з тим, що при табулюванні функції жоден з вузлів не набуває точного значення числа π, а значить буде знаходитись поза областю визначення функції.

З особливостями алгоритмів побудови графіків пов'язаний недолік побудови графіків неявно заданих залежностей. Найпростішим алгоритмом побудови графіка такої залежності є послідовний перебір всіх точок екрану, що відповідає координатній площині, перетворення координат екрану в координати площини та перевірка на належність відповідної точки графіку функціональної залежності. Але такий підхід потребує великої кількості математичних обчислень. І хоча швидкодія сучасних комп'ютерів дозволяє це робити, ще кілька років тому побудова графіка за таким алгоритмом потребувала значного часу (іноді кількох хвилин). Саме тому розробникам програмного забезпечення довелося знаходити інші алгоритми побудови неявно заданих залежностей. В різних програмах вони досить різні, але їх точність в окремих випадках не завжди задовільна. На рисунках 8-10 зображені фрагменти графіка неявно заданої функціональної залежності $\sin(\sin x^2 + \cos y^2) = 0$, побудовані в програмах GRAN1, Advanced Grapher та Derive.

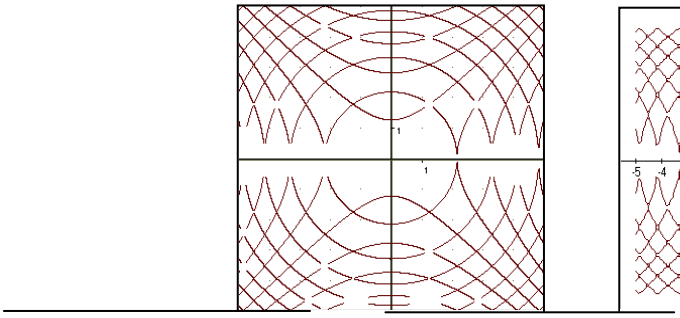

Рис. 8.

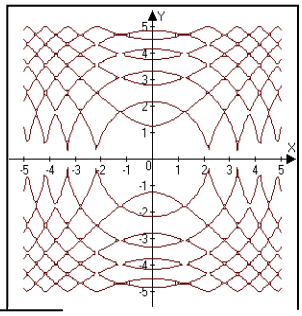

Рис. 9.

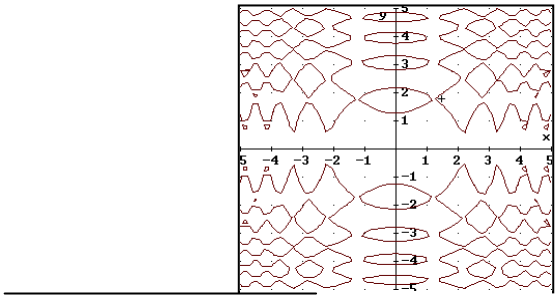

Рис. 10.

В деяких програмах передбачено розв'язування різноманітних рівнянь шляхом символьних перетворень, але в певних випадках програма показує, що рівняння не має розв'язків, хоча насправді вони є, або виводить лише окремі корені рівнянь, без врахування, наприклад, періоду функції. Докладні приклади можна знайти в [1].

Помилки, що можна віднести до другої групи, суттєво відрізняються від попередніх. Найчастіше вони пояснюються відсутністю у користувача достатніх знань користувача з математики або незадовільним рівнем знань та навичок з математики, а також користування як конкретною програмою, так і комп'ютером взагалі.

Однією з особливостей використання комп'ютерних програм для підтримки курсу математики

є запис математичних виразів, що вводяться до пам'яті комп'ютера. Як відомо, запис певного виразу в зошиті учня найчастіше відрізняється від запису того ж виразу для занесення в програму. Окремі учні про це забувають та намагаються комп'ютерний запис виразу подати однаково із записом в зошиті. Для прикладу розглянемо вираз $\frac{\sqrt[3]{\cos^2 4x+1}}{e^x}+\frac{\sqrt{x}}{5x-2}$, запис якого практично в будь-якому програмному засобі можна подати наступним чином: $(\cos(4*x)\text{^}2+1)\text{^}(1/3)/\exp(x)+\text{sqrt}(x)/(5*x-2)$.

Наведемо основні помилки, яких припускаються учні при запису такого виразу.

1. *Пропуск знака множення*. Наприклад, замість $\cos(4*x)$ записується $\cos(4x)$. Це пояснюється звичкою опускати знак множення при запису виразів у зошиті. Щоправда, деякі програми (Derive, MathCad тощо) дозволяють не писати знак множення, але це може привести до іншої помилки, яку буде розглянуто нижче.
2. *Пропуск дужок*. Найчастіше така помилка пов'язана з незнанням пріоритету виконання операцій, простою неуважністю учня або, як і в попередньому випадку, звичкою опускати окремі знаки при запису виразу в зошиті. Наприклад, замість $\cos(4*x)$ пишуть $\cos 4x$ або $\cos 4*x$, а замість $x/(5*x-2)$ – вираз $x/5*x-2$. Оскільки обчислення кореня, крім квадратного, в комп'ютерних програмах не передбачено, то для задання виразу $\sqrt[3]{\cos^2 4x+1}$ потрібно скористатись властивістю кореня, тобто $\sqrt[3]{\cos^2 4x+1}=(\cos^2 4x+1)^{\frac{1}{3}}$. І тут учень може пропустити дужки, задавши вираз так: $(\cos(4*x)\text{^}2+1)\text{^}1/3$, що, фактично, відповідає виразу $\frac{\cos^2 4x+1}{3}$.
3. *Неправильне використання дії піднесення до степеня*. І знов звичка може відіграти негативну роль. Математичний запис виразу $\cos^2 4x$ передбачає спочатку запис піднесення косинуса до квадрату, а вже потім запис аргументу функції. Аналогічно учні намагаються зробити і в комп'ютерному записі – $\cos\text{^}2(4*x)$. Досить часто зустрічається інша помилка, яку можна віднести до цього ж виду – неправильне використання функції експоненти. Далеко не всі учні одразу розуміють, що запис функції $\exp(x)$ вже означає піднесення числа e до степеня x, тому намагаються записати $\exp\text{^}x$ або $\exp\text{^}(x)$, що є помилкою
4. *Недостатнє знання правил запису виразів в конкретному програмному продукті*. Хоча всі програмні продукти схожі між собою в способах задання виразів, все ж існують певні відмінності, які можуть вплинути на результат. Як вже вказувалось, в окремих програмах дозволяється пропускати знак множення, а для задання виразів необов'язково попередньо оголошувати прості змінні (такі як *x, y, k, m* тощо), інші змінні навпаки підлягають обов'язковому попередньому оголошенню (наприклад, *y5, xx, xz*). Якщо цього не зробити, то може трапитись ситуація, коли при запису у виразі змінних *x1, x2, xx*, наприклад в програмі Derive, в самій програмі їх буде сприйнято як $x\cdot 1$, $x\cdot 2$, $x\cdot x$, тобто фактично x, $2x$ та x^2 відповідно.

Аналогічна ситуація може трапитись, якщо використовувати неправильний запис для математичної функції. Так, квадратний корінь в одних програмах позначають функцією *SQR*, а в інших – *SQRT*. Використовуючи ту ж саму програму Derive для обробки виразу із коренем, потрібно знати, що в цьому випадку використовується саме функція *SQRT*. Запис *SQR*(*x*) буде трактуватись програмою як добуток змінних *s, q, r, x*.

Ще один недолік, на який потрібно звернути увагу, можна віднести до комбінованого типу. Розглянемо функцію $y=x^{\frac{2}{6}}$ та побудуємо її графік, користуючись програмою GRAN1. Для цього введемо вираз для функції як *x*^(2/6). Одержимо графік, поданий на рис.11. Запропонуємо учням скористатись властивістю степеневої функції $x^{\frac{a}{b}}=(x^a)^{\frac{1}{b}}=(x^{\frac{1}{b}})^a$, запишемо відповідні вирази (*x*^2)^(1/6) *і* (*x*^(1/6))^2 та побудуємо графіки (рис.12, рис.13). Як бачимо, графіки суттєво різняться. Насправді ж графіком функції $y=x^{\frac{2}{6}}$ є крива, зображена на рис.13. Практично аналогічні результати одержуються при користуванні й іншими програмами, хоча потрібно зауважити, що програма MathCad одразу малює правильний графік, а програма Derive за умови, коли параметр Branch встановлено в Principal.

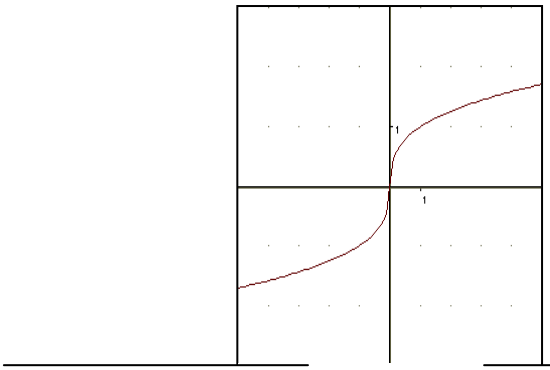

Рис. 11.

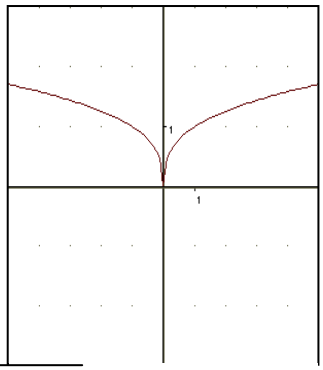

Рис. 12.

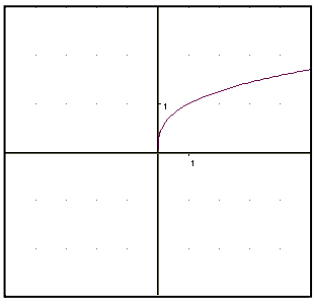

Рис. 13.

Така невідповідність зазвичай спантеличує переважну більшість учнів. А разом з тим справа тут дуже проста – незнання учнями означення степеня. За означенням степінь з раціональним показником розглядається лише для додатних значень основи, тобто графік функції можна будувати тільки для $x > 0$, і лише для окремих значень показника область визначення можна розширити [3]. Як правило, учні це пам'ятають лише поки вивчається власне означення та властивості степеневої функції. Після цього, коли починається розв'язування прикладів або вивчення іншої теми, умова $x > 0$ поступово відходить на другий план і з часом забувається. З іншого боку не в кожній комп'ютерній програмі передбачено аналіз виразу на відповідність області побудові і області допустимих значень, як це робиться у великих професійних пакетах. Так, за алгоритмом програма повинна будувати графік функції з дробовим показником і при $x \le 0$, бо в протилежному випадку не можна було б побудувати, наприклад, графік функції кубічного кореня $y = \sqrt[3]{x}$, оскільки ввести вираз, що відповідає цій функції можна лише застосовуючи властивість $\sqrt[3]{x} = x^{\frac{1}{3}}$. Таким чином, основний контроль за правильністю вхідних даних та області побудови графіка функції лягає на користувача.

Як показують наведені приклади, комп'ютер з відповідними (навіть професійними) програмними засобами не є засобом стовідсотково правильного розв'язування математичних задач. Дійсно, за програмою комп'ютер може спростити вираз, розв'язати рівняння, побудувати графік функції тощо, але довіряти цим відповідям можна не завжди. Це, однак, не означає, що потрібно відмовлятись від використання відповідних програм, навіть якщо вони і містять певні недоліки, оскільки "переваги однієї системи важать більше, ніж аналогічні недоліки ряду систем, бо ці переваги завжди є реальними, а вказані недоліки можна подолати" [2]. Зменшити кількість таких недоліків можна і необхідно. Для цього розробники програмного забезпечення повинні вносити корективи у відповідні алгоритми опрацювання даних, в чому їм повинні допомогти вчителі та методисти, тримаючи з розробниками зв'язок та повідомляючи їх про знайдені недоліки, висувати свої пропозиції щодо вдосконалення програмного забезпечення. Користувачам же необхідно більш глибоко вивчати як можливості і особливості використання відповідних програм та правила роботи з ними, так і саму предметну галузь, в якій він працює. Як відомо, окремі учні на певному етапі навчання починають вважати, що математику можна не вчити, бо приклади за них зробить комп'ютер. Але розуміння того, що комп'ютер в цьому не ідеальний, не тільки може додати учням впевненості ("я самостійно можу розв'язувати приклади, які комп'ютеру не під силу"), але і стимулюватиме їх до вивчення предмету.

ЛІТЕРАТУРА

1. Жалдак М.І. Комп'ютер на уроках математики. Посібник для вчителів. – К.: Техніка, 1997. – 304 с.
2. Основи нових інформаційних технологій навчання: Посібник для вчителів / Машбиць Ю.І., Гокунь О.О, Жалдак М.І. та ін. / За ред. Машбиця Ю.І. / Інститут психології ім. Г.С. Костюка АПН України. – К.:ІЗМН, 1997. – 264 с.
3. Алгебра і початки аналізу: Підруч. Для 10-11 кл. серед. шк. / А.М.Колмогоров, О.М.Абрамов, Ю.П.Дудніцин та ін.; За ред. А.М.Колмогорова. – К.:Освіта, 1994. – 350 с.