«Системні технології» 6 (149) 2023 «System technologies»
DOI 10.34185/1562-9945-6-149-2023-10
УДК 004.043, 004.021, 004.921

Р. Котенко, Л. Божуха

ДОСЛІДЖЕННЯ СТРУКТУР ДАНИХ ДЛЯ ЗАДАЧІ ОПТИМІЗАЦІЇ ПОШУКУ ПЕРЕТИНУ ТРИВИМІРНИХ ОБ'ЄКТІВ

Анотація. У роботі досліджуються підходи та алгоритми вирішення задачі оптимізації пошуку перетину з тривимірними об'єктами. Приділена увага вибору структури даних. Розроблено програмне забезпечення з набором структур даних (дерево октантів, kd дерева, ієрархія обмежувальних об'ємів, регулярна сітка) для оптимізації пошуку перети ну. Одержані результати аналізу роботи оптимізаційних структур даних задачі пошуку перетину тривимірних об'єктів. Проведено порівняльний аналіз швидкості побудови та пошуку перетину між тривимірними об'єктами. Програмне забезпечення написано на мові JavaScript у середовищі Visual Studio Code. Результати дослідження можуть бути викори стані при роботі з тривимірною графікою, моделюванням, 3D інженерією, обчислюваль ною геометрією та у вебдодатках з необхідністю реалізації модуля взаємодії з тривимір ними об'єктами.

Ключові слова: структури даних, дерево октантів, kd дерево, ієрархія обмежувальних об'ємів, регулярна сітка, тривимірна графіка, полігональна сітка, пошук перетину між мешами, оптимізація.

Постановка проблеми

Обробка великих обсягів тривимірних даних вимагає точних та ефективних методів пошуку перетину об'єктів у тривимірному просторі. Ця задача є особливо специфічною для сфери 3D інженерії, оскільки точність та швидкість пошуку перетину об'єктів мають критичне значення в процесах моделювання, проєктування та аналізу складних тривимірних систем.

У контексті оптимізації пошуку перетину в тривимірному просторі використовуються різні структури даних, наприклад Octree, KD-Tree, BVH (Bounding Volume Hierarchy) та Regular Grid. Кожна з цих структур має свої унікальні характеристики, які впливають на швидкодію та ефективність пошуку перетину об'єктів у тривимірному просторі. Ці структури відіграють ключову роль у покращенні продуктивності алгоритмів пошуку та дозволяють забезпечити швидкий доступ до тривимірних даних безпосередньо пов'язаних з пошуком перетину.

Аналіз останніх досліджень і публікацій

Пошук перетину полігональних сіток (мешів) є завданням знаходження перетину областей у тривимірному просторі, які представляють ці полігональні представлення об'єкта [1]. Підходи до пошуку перетину можуть бути різними залежно від складності мешів та вимог до точності результатів. У загальному вигляді, завдання виявлення перетину між мешами полягає у визначенні всіх трикутників одного мешу, які перетинають трикутники іншого мешу [2].

Інший підхід до пошуку перетину полягає в застосуванні алгоритмів променевого трасування (ray tracing), обходу граней мешу, алгоритми перетину для геометричних об'єктів тощо. Для складних мешів (велика кількість граней і вершин) можуть застосовуватися методи побудови обмежувальних об’ємів (bounding volumes), що дозволяють швидше відфільтровувати області, які точно не перетинаються, зменшуючи обчислювальну складність. Саме цей підхід буде використаний у цій роботі. Важливо враховувати, що пошук перетину може бути витратним за ресурсами обчисленням, особливо для складних моделей, тому ефективність та швидкодія цих алгоритмів важлива для реального часу в графіці, симуляціях чи програмах моделювання.

Мета дослідження

Метою роботи є розроблення програмного забезпечення з різними структурами даних. Трикутник вважається базовою складовою мешу, тому пошук перетину між трикутниками - це ключова операція для визначення перетину між мешами. Складніші геометричні структури складаються з великої кількості трикутників. Виявлення перетину трикутників є основою для визначення можливих зон перетину між складними тривимірними об'єктами, що представлені у вигляді мешів.

Викладення основного матеріалу дослідження

Алгоритм пошуку перетину між двома трикутниками у тривимірному просторі є процесом визначення існування хоча б однієї спільної точки (перетин трикутників). Основна ідея полягає в обчисленні перетину площин трикутників та подальшому визначенні областей перетину [3].

Першим кроком є первинна перевірка можливого перетину шляхом визначення методами побудови обмежуючих об'ємів для кожного трикутника та подальшої перевірки цих просторових фігур на перетин. Такий підхід дозволяє швидко відфільтрувати трикутники, які точно не перетинаються.

Стратегія розділення тривимірного простору на менші області для полегшення оброблення та швидкого доступу до об'єктів у цьому просторі. Вибір

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

стратегії допомагає оптимізувати пошук, взаємодію та візуалізацію об'єктів у тривимірному середовищі [4]. У роботі проаналізовані підходи розбиття простору:

1) ієрархічна структура дерева (Octree) з різними типами вузлів (листові, внутрішні) використовується для швидкого пошуку та оброблення інформації (роботи з даними) об'єктів у тривимірному просторі, швидкої локалізації області простору з визначеними об'єктами;

2) бінарне дерево (K-dimensional tree) використовує розділення простору за різними вимірами та ефективно використовується для пошуку точок;

3) ієрархічна структура (Bounding Volume Hierarchy, BVH) з групуванням об'єктів в обмежувальні об'єми (паралелепіпеди, сфери) та використовується для організації та прискорення пошуку перетинів між об'єктами у тривимірному просторі [5].

4) структура сітки (Regular Grid) використовує розділення простору на рівні області та дозволяє ефективно здійснювати пошук перетину лише в необхідних областях (наявна велика ймовірність перетину мешів).

Процес побудови таких структур даних включає в себе вибір площини розділення для розбиття даних на дві півплощини. Вибір цієї площини використовує методи центру (вибір точки посередині діапазону значень), медіани (вибір медіани діапазону значень) та Surface Area Heuristic (SAH, евристична оптимізація).

Для розроблення програмного забезпечення обрано TypeScript як мову програмування з урахуванням її переваг у відношенні до статичної типізації, що сприяє виявленню помилок на етапі розроблення та полегшує обслуговування програми в подальшому. Node.js був обраний як середовище виконання для забезпечення можливості запуску програми на сервері, що дозволяє використовувати його потужності для обробки тривимірних даних. Бібліотека THREE.js визначена для реалізації графічної частини програмного продукту для забезпечення потужних можливостей візуалізації тривимірних об'єктів та легкого інтегрування з TypeScript. У якості середовища розроблення програмного забезпечення обрано Visual Studio Code.

Для перевірки роботи програмного забезпечення були обрані тривимірні об’єкти (рис. 1).

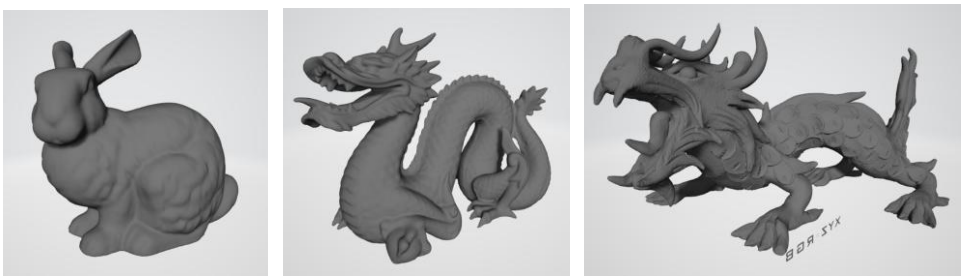

Рисунок 1 – Тривимірні об’єкти:
(а) Stanford Bunny (~70 000 примітивів), (б) Stanford Dragon
(~870 000 примітивів), (с) Stanford XYZRGB Dragon (~7 200 000 примітивів)

Заміри швидкості побудови структур за різних параметрів

Для обраних фігур була виконана побудова структур за різних параметрів глибини, та типів розподілу.

Таблиця

Середній час побудови **octree** (у мс)

Макс. глибина	Bunny	Dragon	XYZRGB Dragon
1	80	1050	13700
2	180	2100	24300
3	280	3300	32300
4	340	4500	45100
5	500	5800	55800
6	850	7600	67700

Таблиця 2

Середній час побудови **kd-дерева** (центр/медіана/ SAH) (у мс)

Макс. глибина	Bunny	Dragon	XYZRGB Dragon
2	70/80/700	1100/1010/10100	12300/8800/95300
4	150/170/1400	2040/2100/20500	21000/22000/223400
6	230/250/2200	3250/3200/31000	30900/30500/302400
8	310/330/3100	4210/4300/40000	43200/41400/395600
10	400/440/4000	5350/5500/52000	48000/51200/480300
12	480/480/4800	6400/6200/61000	59700/62200/547200
14	540/480/4900	7600/8700/76200	67800/73200/725600

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Для вісімкового дерева здійснювалась побудова за різних значень максимальної глибини, а для kd-дерева та BVH – за різних значень глибини та типу розподілу вузла (по центру, по медіані, з використанням SAH) (табл. 1-3).

Таблиця 3

Середній час побудови **BVH** (центр/медіана/ SAH) (у мс)

Макс. глибина	Bunny	Dragon	XYZRGB Dragon
2	120/100/720	1300/1400/10700	7500/8400/87200
4	160/160/1490	2200/2400/20600	14700/16200/165300
6	230/260/2200	3400/3500/31700	22300/24600/261500
8	310/330/3000	4300/4700/42300	31400/30900/312400
10	400/420/3900	5100/5900/54400	39100/39930/386500
12	460/470/4200	6400/7100/64700	48100/47200/457800
14	450/460/4300	7600/8300/76500	56200/58800/564300

Заміри швидкості пошуку перетину за різних параметрів

Використовуючи побудовані структури були здійснені заміри швидкості пошуку перетину (табл. 4-6).

Таблиця 4

Середній час пошуку перетину **octree** (у мс)

Макс. глибина	Bunny	Dragon	XYZRGB Dragon
1	1,5	14	170
2	0,7	5	70
3	0,4	1,5	24
4	0,15	0,7	15
5	0,1	0,4	3
6	0,1	0,3	1,5

Таблиця 5

Середній час пошуку перетину **kd-дерева** (центр/медіана/ SAH)(у мс)

Макс. глибина	Bunny	Dragon	XYZRGB Dragon
2	1,8/1,5/1,5	25/15/15	320/280/290
4	1,6/1,3/1,3	14/9/9	90/110/109
6	1/0,8/0,7	8/5/4	50/70/80
8	0,5/0,4/0,4	3/2,4/2,5	30/28/31
10	0,3/0,3/0,2	1,7/1,3/1,1	18/18/17
12	0,2/0,2/0,15	0,8/0,8/0,5	8/2,7/4
14	0,15/0,18/0,15	0,5/0,3/0,3	3,2/1,9/1,3

Таблиця 6

Середній час пошуку перетину BVH (центр/медіана/ SAH)(у мс)

Макс. глибина	Bunny	Dragon	XYZRGB Dragon
2	1,9/1,8/1,9	24/25/24	230/200/210
4	1,5/1,4/1,5	17/16/15	140/120/115
6	0,9/0,9/0,9	7/5/5	35/25/31
8	0,6/0,6/0,6	1,8/2,1/2,2	19/12/13
10	0,3/0,25/0,3	1,4/0,9/0,8	7/5/5
12	0,15/0,2/0,2	0,6/0,35/0,4	2,8/2,5/2,4
14	0,15/0,15/0,15	0,2/0,2/0,15	1,4/1,1/1,1

Отже, середня швидкість пошуку перетину для одного й того ж об’єкту при використанні різних методів є приблизно однаковою. Проте можна бачити деяке збільшення швидкості якщо порівняти kd-дерево з розподілом по центру та BVH з використанням SAH. Але таке збільшення є важливим у випадках, коли перетин з об’єктом потрібно шукати велику кількість разів протягом певного часу. Такий приріст швидкості можна отримати лише витративши набагато більше часу на первинну побудову, що скоріш за все не буде мати сенсу для вебдодатків, але буде важливим для отримання більшої швидкості пошуку перетину для інших цілей.

Тому оптимальними рішеннями, що не потребують багато часу для побудови та мають достатню швидкість пошуку перетину є структура даних , яка

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

представлена вісімковим деревом, kd-деревом (розділення по центру та за медіаною) та ієрархією обмежувальних об’ємів (розділення по центру та за медіаною).

У випадку якщо все ж таки необхідна максимальна швидкість пошуку перетину, то оптимальними рішеннями є структура даних , яка представлена kd-деревом (розділення з використанням SAH) або ієрархією обмежувальних об’ємів (розділення з використанням SAH).

Заміри швидкості побудови структур в залежності від розміру мешу

Для оцінки і порівняння швидкості побудови структур було обрано три варіанти мешу з різною кількістю полігонів: ~ 16 тис. трикутників - малий меш; ~ 260 тис. трикутників - середній меш; ~ 1 млн. трикутників - великий меш.

При використанні визначених полігональних сіток проведено певну кількість запусків алгоритму побудови дерева октантів, kd-дерева, ієрархії обмежувальних об’ємів та сітки для з’ясування середньої швидкості побудови структур.

Побудова деревоподібних структур виконувалася на таких параметрах: максимальна глибина: для octree - 10, для kd-дерев - 30; кількість трикутників на вузол - 20. Дані параметри побудови дерев забезпечують максимальну швидкість пошуку перетину (табл. 7). Для побудови сітки параметр розміру був встановлений на - 20 комірок.

Результати спостереженнь показують, що деревоподібні структури мають більшу ефективність у порівнянні з сіткою. Серед усіх структур kd-дерево - найшвидша структура при побудові. При опрацюванні великих тривимірних об’єктів ієрархія має дещо більший час побудови у порівнянні з kd-деревом, це може бути спричинено тим, що при кожному розподілу вузла виконується перерахування обмежувального об’єму. Структура сітки у свою чергу є найповільнішою структур для яких проводились заміри.

Таблиця 7

Середній час побудови структур (у мс)

Структура	Малий меш	Середній меш	Великий меш
Octree	70	1260	5750
KD-Tree	54	1030	5550
BVH	54	1090	7880
Grid	1010	24110	101000

Для більш точної перевірки сітки були проведені додаткові заміри швидкості на різних значеннях розміру сітки для малого мешу.

Структура сітки є більш чутливою до параметрів побудови і при належному їх встановленні можна досягти непоганих результатів, при цьому не втрачаючи у ефективності.

Заміри швидкості пошуку перетину об'єктів в залежності від розміру мешу

Для замірів використовувались деревоподібні структури з оптимальними параметрами побудови - 20 трикутників на вузол і максимальна глибина 30 та 10 для бінарних дерев та октодерева відповідно (табл.8).

Таблиця 8

Середній час пошуку перетину (у мс)

Структура	Малий меш	Середній меш	Великий меш
Octree	60	110	281
KD-Tree	25	69	190
BVH	29	92	268

Отже, структура даних Octree у порівнянні з другими структурами має більший час пошуку. При чому kd-дерево найшвидша серед усіх структур з великий відривом. У свою чергу ієрархія обмежувальних об'ємів хоч і є швидшою за дерево октантів, але не має таких результатів як і KD-Tree. Причиною такої швидкості пошуку перетину може бути те, що BVH часто має перетин вузлів і так, як вузли перетинаються під час пошуку додатковий час витрачається на обхід одних і тих самих полігонів.

Результати проведених експериментів показують, що швидкість пошуку перетину між мешами з використанням сітки менша, ніж октодерева, kd-дерева та BVH. Але при цьому швидкість менша не надто сильно (у випадку великого мешу у два рази).

Висновки

Досліджено технології пошуку перетину із тривимірними об'єктами та сформовані проблеми, які можуть виникати під час виконання цієї операції за певних умов. Однією з таких складностей є швидкість пошуку перетину при великих наборах примітивів, з яких складаються об'єкти.

Досліджено результати використання складних структур даних. Реалізовано вище зазначені структури з можливістю їх налаштування різними вхідними параметрами (максимальна глибина, мінімальна кількість трикутників у листовому вузлі та тип розподілу). Проведено заміри швидкості роботи реалізованих структур для різних за розміром об'єктів та за різних параметрів побудови структур.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Найшвидшою при побудові та при пошуку перетину є kd-дерево. Ця структура мала найкращі результати як на малих так і на великих об'ємах даних.

Другою по ефективності є ієрархія обмежувальних об'ємів. Дана структура схожа на KD-Tree, але через особливість у вигляді перерахування об'ємів для кожного вузла потребує більше часу при побудові. Також під часу перевірки швидкості пошуку перетину дана структура показала гірші результати, ніж kd-дерево, незважаючи на те, що меш був розподілений на більш щільні підпростори, що у свою чергу мало б збільшити ефективність.

На третьому місці посідає дерево октантів. Ця структура досить за алгоритмом побудови тому, що розділ вузлів завжди відбувається у середині об'єму розділяючи вузол на вісім дочірніх вузлів. Через таку структуру, розподіл відбувається не рівномірно, що впливає на швидкість пошуку.

Підхід з розділенням простору з використанням сітки посідає четверте місце, але під час проведення замірів було з'ясовано, що недостатньо обрати максимальні значення параметрів, щоб досягти максимальної ефективності, як у випадку деревоподібних структур. Дана структура потребує більш точного використання параметрів побудови, що в результаті приведе до збільшення ефективності.

Створено додаток для відображення вхідних тривимірних об'єктів з візуалізацією обраних структур даних.

ЛІТЕРАТУРА

1. Introduction to Polygon Meshes. URL: https://www.scratchapixel.com /lessons/3d-basic-rendering/introduction-polygon-mesh (дата звернення 23.05.2022).

2. Brian Curless. Ray-triangle intersection. URL:https://courses.cs. washington.edu/courses/csep557/10au/lectures/triangle_intersection.pdf (дата звернення 23.05.2022).

3. Hamzah Asyrani Sulaiman, Abdullah Bade. Bounding Volume Hierarchies for Collision Detection. URL: https://www.researchgate.net /publication/224829148_Bounding_Volume_Hierarchies_for_Collision_Detection (дата звернення 23.05.2022).

4. what-when-how — In Depth Tutorials and Information. URL:http://what-when-how.com/advanced-methods-in-computer-graphics/collision-detection-advanced-methods-in-computer-graphics-part-6/ (дата звернення 23.05.2022).

5. Ingo Wald. On fast Construction of SAH-based Bounding Volume Hierarchies. URL: https://www.sci.utah.edu/~wald/Publications/2007 /ParallelBVHBuild/fastbuild.pdf (дата звернення 23.05.2022).
6. Полігональна сітка в комп'ютерній графіці. URL: https://termin.in.ua/polihonal-na-sitka/ (дата звернення 09.01.2024).
7. Finding Intersecting Mesh. URL:https://www.sandia.gov/files/cubit/15.5/help_manual/WebHelp/mesh_generation/mesh_quality_assessment/find_intersecting_mesh.htm (дата звернення 09.01.2024).
8. A Fast Triangle-Triangle Intersection Test. URL: https://fileadmin.cs.lth.se/cs/Personal/Tomas_Akenine-Moller/pubs/tritri.pdf (дата звернення 09.01.2024).
9. Spatial Partition. URL:https://gameprogrammingpatterns.com/spatial-partition.html (дата звернення 09.01.2024).
10. How octree work. URL: https://castle-engine.io/vrml_engine_doc /output/xsl/html/section.how_octree_works.html (дата звернення 09.01.2024).
11. Fast kd-Tree Construction for 3D-Rendering Algorithms Like Ray Tracing. URL: https://www.researchgate.net/publication/228573134_Fast_kd Tree_Construction_for_3D-Rendering_Algorithms_Like_Ray_Tracing (дата звернення 09.12.2024).
12. Spatial Splits in Bounding Volume Hierarchies. URL: https://www.nvidia.in/docs/IO/77714/sbvh.pdf (дата звернення 09.12.2024).

REFERENCES

1. Introduction to Polygon Meshes. URL: https://www.scratchapixel.com /lessons/3d-basic-rendering/introduction-polygon-mesh (date of application 23.05.2022).
2. Brian Curless. Ray-triangle intersection. URL:https://courses.cs. washington.edu/courses/csep557/10au/lectures/triangle_intersection.pdf (date of application 23.05.2022).
3. Hamzah Asyrani Sulaiman, Abdullah Bade. Bounding Volume Hierarchies for Collision Detection. URL: https://www.researchgate.net /publication/224829148_Bounding_Volume_Hierarchies_for_Collision_Detection (date of application 23.05.2022).
4. what-when-how — In Depth Tutorials and Information. URL:http://what-when-how.com/advanced-methods-in-computer-graphics/collision-detection-advanced-methods-in-computer-graphics-part-6/ (date of application 23.05.2022).

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

5. Ingo Wald. On fast Construction of SAH-based Bounding Volume Hierarchies. URL: https://www.sci.utah.edu/~wald/Publications/2007 /ParallelBVHBuild/fastbuild.pdf (date of application 23.05.2022).
6. Polygonal grid in computer graphics. URL: https://termin.in.ua/polihonal-na-sitka/ (date of application 09.12.2024).
7. Finding Intersecting Mesh. URL:https://www.sandia.gov/files/cubit/15.5/help_manual/WebHelp/mesh_generation/mesh_quality_assessment/find_intersecting_mesh.htm (date of application 09.12.2024).
8. A Fast Triangle-Triangle Intersection Test. URL: https://fileadmin.cs.lth.se/cs/Personal/Tomas_Akenine-Moller/pubs/tritri.pdf (date of application 09.01.2024).
9. Spatial Partition. URL:https://gameprogrammingpatterns.com/spatial-partition.html (дата звернення 09.12.2024).
10. How octree work. URL: https://castle-engine.io/vrml_engine_doc /output/xsl/html/section.how_octree_works.html (date of application 09.01.2024).
11. Fast kd-Tree Construction for 3D-Rendering Algorithms Like Ray Tracing. URL: https://www.researchgate.net/publication/228573134_Fast_kd Tree_Construction_for_3D-Rendering_Algorithms_Like_Ray_Tracing (date of application 09.12.2024).
12. Spatial Splits in Bounding Volume Hierarchies. URL: https://www.nvidia.in/docs/IO/77714/sbvh.pdf (date of application 09.12.2024).

Received 19.12.2023.
Accepted 21.12.2023.

Study of data structures for the optimization problem of searching the intersection of three-dimensional objects

In the context of optimizing intersection search in three-dimensional space, various data structures are used, such as Octree, KD-Tree, BVH (Bounding Volume Hierarchy), and Regular Grid.

Approaches to finding the intersection may be different depending on the complexity of the meshes and the requirements for the accuracy of the results.

For complex meshes (a large number of faces and vertices), the methods of building bounding volumes can be used, which allow you to quickly filter out areas that do not exactly intersect, reducing the computational complexity. It is this approach that will be used in this work.

The purpose of the work is to develop software with various data structures.

Three-dimensional objects were selected to test the software: Stanford Bunny (~70,000 primitives), Stanford Dragon (~870,000 primitives), Stanford XYZRGB Dragon

(~7,200,000 primitives). For the selected shapes, the construction of structures was performed with different parameters of depth and types of distribution.

To evaluate and compare the speed of construction of structures, three versions of the mesh with different number of polygons were chosen: ~ 16 thousand triangles - small mesh; ~ 260 thousand triangles - average mesh; ~ 1 million triangles - a large mesh.

Construction of tree-like structures was performed with the following parameters: maximum depth: for octree - 10, for kd-trees - 30; the number of triangles per node is 20. These tree construction parameters ensure the maximum speed of intersection search. To build the grid, the size parameter was set to - 20 cells.

For a more accurate check of the grid, additional velocity measurements were made at different values of the grid size for a small mesh.

Technologies for searching for intersections with three-dimensional objects have been studied and problems that may arise during this operation under certain conditions have been identified. One of these difficulties is the speed of finding an intersection with large sets of primitives that make up objects.

Keywords: data structures, octant tree, kd-tree, bounding volume hierarchy, regular mesh, 3D graphics, polygonal mesh, mesh intersection search, optimization.

Котенко Роман Вікторович – магістр, спеціальність "Інженерія програмного забезпечення", кафедра математичного забезпечення ЕОМ, факультет прикладної математики, Дніпровський національний університет імені Олеся Гончара.

Божуха Лілія Миколаївна – кандидат фізико-математичних наук, доцент кафедри математичного забезпечення ЕОМ, Дніпровський національний університет імені Олеся Гончара.

Kotenko Roman - master, speciality "Software Engineering", department of Mathematical Support of Calculating Machines, Faculty of Applied Mathematics, Oles Honchar Dnipro National University.

Bozhukha Liliia - candidate of physico-mathematical sciences, associate professor, Department of Mathematical Support of Calculating Machines, Oles Honchar Dnipro National University.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)