1 Абстракція

Для штучних супутників (ШС) Землі відомою є проблема електризації. Суть проблеми полягає в нерівномірній електризації поверхні супутника (наприклад, частини поверхні ШС, що знаходяться в сонячній тіні, зряджаються до високого від'ємного потенціала відносно оточуючого космісного простору) [1]. Через неоднакову освітленість діелектричних ділянок ШС виникає різниця потенціалів, яка веде до електричних пробоїв - вони руйнують поверхню супутника і ведуть до сбоїв в роботі радіоелектронних приладів.

Дана робота присвячена розробці програмного забезпечення для моделювання електризації тіл в космісному просторі, а також обчислення значень електричних потенціалів, що будуть накопичуватися на цих тілах або їх частинах.

Знаючи реальні дані (швидкість електронів, густину їх розподілу, швидкість тіла), проводиться серія випробувань (результати щоразу будуть різними за рахунок використання датчика псевдовипадкових чисел) і отримуються значення зміни потенціалу за проміжок часу.

Моделювання проводится за методом Монте-Карло [2]. Цей метод заснований на багатократних реалізація стохастичного процесу і застосовується в тих випадках, коли побудова точної математичної моделі неможлива або значно ускладнена. Прояв методів статистичного моделювання в різних областях прикладної математики, як правило, пов'язаний з необхідністю розв'язання якісно нових задач, що виникають з практичних потреб. Так було при створенні атомної зброї, при освоєнні космосу, моделюванні турбулентності. В якості означення методів Монте-Карло можна навести наступне: Методи Монте-Карло – це чисельні методи вирішення математичних задач (систем алгебраїчних, диференційних, інтегральних рівнянь) і пряме статистичне моделювання (фізичних, хімічних, біологічних, економічних, соціальних процесів) за допомогою отримання і перетворення випадкових чисел. Перша робота із застосуванням методу Монте-Карло була опублікована Холом в 1873 році при організації стохастичного процесу експериментального визначення числа π шляхом кидання голки на розкреслений аркуш паперу (сама задача теорії геометричних імовірностей про голку була сформульована Ж. Бюффоном ще в 1733 році, а розв'язок до неї був опублікований ним в 1777 році). Один з прикладів використання методу Монте-Карло –а використання ідеї Дж. Фон Неймана при моделюванні траєкторій нейтронів в лабораторії в Лос Аламосі в 1940-х роках. Метод названо на честь столиці князівства Монако, відомої своїми численними казино, основу яких складає рулетка — досконалий генератор випадкових чисел. Перша робота, де метод Монте-Карло викладався систематично, була опублікована в 1949 році Метрополісом і Уламом, де цей метод застосовувався для для розв'язання лінійних інтегральних рівнянь, що були пов'язані із задачею про проходження нейтронів через речовину [3].

В даній роботі проводиться пряме моделювання методом Монте-Карло. При такому підході виконується моделювання окремих частин фізичної системи, для прискорення розрахунків допускається застосування деяких фізичних наближень. В нашому випадку елементи фізичної системи — це штучний супутник і велика кількість елементарних частинок, для яких реалізується стохастичний процес їх руху і зіткнення із літальним апаратом. При цьому рух частинок розглядається як рівномірний прямолінійний, а зіткнення частинок між собою не враховуються, оскільки не мають значення для розв'язання задачі і не впливають на результати.

Елементарні частинки генеруються на сфері, що описується навколо тіла (літального апарату); центр сфери співпадає з центром тіла. Їх координати і напрям задаються випадковим чином, а швидкість – за допомогою нормального розподілу (для цього по-

трібно знати середню, максимальну і мінімальну швидкість – за правилом трьох сігм знаходяться параметри нормального розподілу). При цьому система має такі параметри: n – кількість частинок, присутніх в системі в кожну одиницю часу, R – радіус сфери, на якій будуть генеруватись частинки (між n і R існує залежність, оскільки кількість частинок визначається об'ємом сфери, всередині якої вони знаходяться) і Δt – часовий крок.

Модель тіла зчитується з файлу, в якому задаються координати вершин трикутних полігонів.

Висновок Про отримані результати <в процесі>

2 Постановка задачі

3 Опис розв'язання

3.1 Опис структур даних

Для представлення чисел з плаваючою крапкою введено тип real, який є альтернативним іменем для типу float (одинарна точність), а при недостачі точності чи інших потребах може бути легко замінений на double (подвійна точність).

3.1.1 Point

Тип потрібен для представлення точки в тривимірному просторі – в кожному об'єкті зберігаються три координати типу float. Також наявні методи для порівняння з іншими об'єктами цього ж типу і методи для додавання або віднімання вектора (виконується зсув точки на заданий вектор).

3.1.2 Vector

Тип введено для представлення вектора в тривимірному просторі. Клас Vector успадковано від класу Point, оскільки вектор теж однозначно задається трьома координатами – при необхідності може перетворений до батьківського класу. Серед методів можна назвати порівняння з об'єктами цього ж типу, множення вектора на константу і на вектор (в наявності як скалярний добуток, так і векторний, знаходження суми і різниці векторів, довжини вектора, косинуса кута між векторами, а також нормалізація вектора і приведення його до заданої довжини.

3.1.3 Locus

Базовий шаблонний клас для всіх підкласів, які представляють собою геометричний об'єкт, що задається набором точок. В якості параметра класу виступає кількість точок. Об'єкт класу містить лише масив заданої параметром довжини і має метод для виводу цього масиву в зручній для сприйняття формі. В перспективі до класу можуть бути додані методи, що працюють відразу з усіма точками, незалежно від їх кількості (наприклад, афінні перетворення).

3.1.4 Line

Клас успадковано від Locus з параметром 2, тобто містить в собі дві точки, а також для зручності направляючий вектор, котрий обчислюється при конструюванні об'єкта.

Серед методів можна відмітити метод отримання точки на прямій за коефіцієнтом, яким ця точка визначається в параметричних рівняннях прямої.

3.1.5 ThreePoints

Базовий клас для всіх класів, що представляють об'єкти, які можуть бути задані трьома точками (трикутник, площина, орієнтоана площина). Клас успадковано від Locus з параметром 3, тобто містить в собі три точки. Окрім оперетору присвоювання має ща метод для визначення нормалі (за допомогою векторного добутку векторів, утворнеих двома різними парами точок).

3.1.6 Triangle

Клас для представлення трикутника, успадкований від классу ThreePoints. Додано метод для знаходження центру мас трикутника.

3.1.7 Plane

Клас для представлення площини, успадкований від классу ThreePoints. Додано метод для визначення чи належить точка даній площині.

3.1.8 OrientedPlane

Клас для представлення орієнтоаної площини, успадкований від классу Plane. Містить вектор нормалі, який тепер повертається перевантаженим метод отрмання нормалі, якийй було визначено в класі ThreePoints. Нормаль обчислюється при конструюванні об'єкта як векторний добуток двох векторів: один з почтаком в першій точці, кінцем в другій, інший з початком в першій точці, кінцем в третій – тобто перший, другий вектори і нормаль утворюють праву трійку векторів; іншими словами з кінця нормалі, початок якої лежить в площині, видно три точки, якими задається площина, в порядку руху годинникової стрілки. Конструктор класу, що описується, приймає також логічний параметр, який визначає, чи буде видно точки в напрямі руху годинникової стрілки (за замовчуванням цей параметр істинний).

3.1.9 Particle

Клас, що описує елементарну частинку. Є успадкованим від класу Point. Додано поля для збереження вектора руху, а також поле типу real, що визначає час існування частинки в секундах (якщо значення від'ємне, час вважається не заданим).

3.1.10 Sphere

Клас для представлення сфери. Має поля для збереження точки – центру сфери, а також числа типу real – радіуса сфери.

3.1.11 Object3D

Клас, призначений для збереження координат полігонів тривимірних тіл. При конструюванні кожного об'єкта обчислюються і пишуться у дві відповідні точки (об'єкти Point) максимальні та мінімальні координати тіла за всіма осями. Клас успадковано від класу Sphere – центр шукається як середина відрізка, що сполучає дві вищеописані

точки, радіус – як половина довжини цього відрізка; таким чином виходить, що ці параметри задають сферу, описану навколо тіла. Це наслідування є корисним при перевірці, чи перетинає траекторія частинки тіло – спочатку перевіряється, чи перетинає пряма траекторії сферу, якщо так – виконується перевірка для кожного полігону циклічно. Як параметр констуктора об'єкта приймається також вектор, що задає напрям руху тіла (за замовчуванням співпадає з віссю OX).

3.1.12 GenerativeSphere

Клас, призначений для генерації елементарних частинок. Є нащадком класу Sphere. По суті являє собою сферу, центр якої співпадає з центром тіла, а радіус має перевищувати радіус тіла. Полями класу є генератори випадкрвих числел, за допомогою яких для кожної частинки генерується випадковим чином швидкість (один генератор для кожного типу частинок – іонів та електронів), а також посилання на об'єкт, що представляє собою власне тіло (типу Object3D). В класі наявні методи для генерації частинок наступним чином:

1. частинки, що в початковий момент часу свого існування лежать всередині сфери і траекторія яких не обов'язково перетинає тіло;

2. частинки, що в початковий момент часу свого існування лежать на сфері і траекторія яких перетинає тіло;

3.2 Опис геометричних функцій

3.2.1 Перевірка, чи знаходиться точка всередині трикутника

Перевірка виконується для точки, що знаходиться в площині трикутника.

Спосіб 1

Виконується перевірка, чи не співпадає точка з однією з вершин трикутника. Після цього шукаються кути між всіма парами векторів, початок яких знаходится в даній точці, а кінець співпадає з вершиною трикутнка. Очевидно, що якщо всі кути будуть тупими, тобто всі косинуси від'ємними, то точка лежамтие всередині трикутника. Також очевидно, що якщо три або два кути виявляться гострими, то точка лежить за межами трикутника. Інакше за допомогою стандартної функції знаходження арккосинуса шукаємо суму всіх кутів. Неважко побачити, що для точок, які не належать трикутнику, ця сума буде менше за 2π.

Спосіб 2

Для кожної пари вершин трикутника виконаємо перевірку: проведемо через них пряму і визначимо, чи лежить точка і третя вершина в одній і тій самій півплощині відносно цієї прямої. Для цього знайдемо проекцію третьої вершини на пряму і косинус кута, вершиною якого є дана проекція, а сторони проходять через задану точку і третю вершину трикутника. Якщо синус від'ємний, то кут є тупим, тобто точка і третя вершина трикутника лежить в різних півплощинах – перевірка не виконалась. Якщо для кожної пари вершин ця перевірка виконається, то точка лежатиме всередині трикутника. Інакше – ні.

3.2.2 Пошук точки перетину прямої і площини

Як відомо, пряма в просторі (тут і далі мається на увазі тривимірний Евклідів простір) може бути задана трьома параметричними рівняннями

$$\begin{cases} x = A_x - k(B_x - A_x) \\ y = A_y - k(B_y - A_y) \\ z = A_z - k(B_z - A_z) \end{cases}, \tag{1}$$

де A, B – точки, через які прохведено пряму. В цьому випадку кожна її точка задається єдиним значенням коефіцієнта. В обох описаних нижче способах ми знаходимо коефіцієнт точки перетину прямої і площини.

Спосіб 1

В канонічне рівняння площини, проведеної через 3 точки A, B і C

$$\begin{vmatrix} x - A_x & y - A_y & z - A_z \\ B_x - A_x & B_y - A_y & B_z - A_z \\ C_x - A_x & C_y - A_y & C_z - A_z \end{vmatrix} = 0 \tag{2}$$

підставимо координати з рівнянь 1. Отримаємо лінійне рівняння відносно кофіцієнта k. Розв'язавши його, отримаємо коефіцієнт шуканої точки перетину.

Спосіб 2

У відоме векторне рівняння площини

$$\bar{n} \cdot \overline{AX} = 0 \tag{3}$$

(де $\bar{n}$ – нормаль площини, P – задана точка на площині, а X – довільна точка площини) підставляємо замість X точку з координатами, взятими з параметричного рівняння прямої 1. Отримаємо лінійне рівняння відносно кофіцієнта k. Розв'язавши його, отримаємо коефіцієнт шуканої точки перетину.

3.2.3 Пошук проекцій

Пошук проекції точки на пряму

Щоб точка на прямій була проекцією заданої точки, необхідно виконання двох умов:

1. Її координати мають задовольняти рівняння прямої 1;
2. Вектор з точки до її проекції має бути перпендикулярним направляючому вектору прямої: $\overline{PP'} \cdot \bar{n} = 0$.

Підставляючи замість координат проекції P' координати з рівнянь 1, отримуємо коефіцієнт точки перетину.

Пошук проекції точки на площину Шукана точка – точка перетину прямої (направляючий вектор якої дорівнює нормалі площини; пряма проходить через задану точку) та заданої площини. Пошук перетину прямої і площини описаний в 3.2.2.

3.2.4 Пошук відстаней

Відстань між двома точками

Знаходиться за відомою формулою

$$\rho(x, y) = \sqrt{\sum_{i=1}^{3} (x_i - y_i)^2} \tag{4}$$

Відстань між точкою та площиною

Знаходимо проекцію точки на площину і підставляємо в формулу 4.

3.2.5 Перевірка, чи перетинає пряма сферу

Неважко побачити, що у випадку, коли пряма перетинає сферу, проекція центру сфери на пряму не буде лежати зовні сфери. Тобто, необхідно лише знайти відстань (див. 3.2.4) між центром сфери і його проекцією (див. 3.2.3) на задану пряму та переконатись, що ця відстань не перевищує радіус сфери.

3.2.6 Поворот точки навколо прямої

Знайдемо проекцію точки на пряму – точку P'. Введемо двохмірну систему координат: одиничнй вектор (орт) осі абсцис $\bar{i}$ співпадатиє з вектором $\overline{P'P}$, а для осі ординат $\bar{j}$ буде одночасно перпендикулярним до осі абсцис та напрявляючого вектора прямої (знайдемо його як векторний добуток направляючого вектора і першої осі); також вісь ординат нормалізуємо і домножимо на довжину осі абсцис, щоб система була декартовою. Очевидно, що шукана точка знаходиться саме в отриманій площині. Як відомо, на площині координати повороту точки навколо початку координат задаються наступною формулою:

$$\begin{cases} x' = x \cdot cos\alpha - y \cdot sin\alpha \\ y' = x \cdot sin\alpha + y \cdot cos\alpha \end{cases}$$

Також очевидно, що початкова точка в новій системі матиме координати (0;1), отже формула повороту для неї перепишеться як

$$\begin{cases} x' = -y \cdot sin\alpha \\ y' = y \cdot cos\alpha \end{cases}$$

Тепер залишилось перейти від двохмірних коорднат назад до трьохмірних – для цього необхідно до координат точки P' додати зміщення $x'\bar{i} + y'\bar{j}$.

Література

[1] А. М. Капулкин, В. Г. Труш, Д. В. Красношапка: *Исследование плазменных нейтрализаторов для снятия электростатических зарядов с поверхности высокоорбитальных космических аппаратов.* ДНУ, 1994.

[2] И. М. Соболь: *Метод Монте-Карло.* «Наука», Москва, 1968.

[3] О.М. Белоцерковский, Ю.И. Хлопков: *Методы Монте-Карло в прикладной математике и вычислительной аэродинамике.*