THEORETICAL ANALYSIS ON LORENZ ATTRACTOR

SAI KAUSHIK S

CED18I044

ABSTRACT

This project is designed to implement parallel programs for the simulation of objects following **Lorenz attractor** satisfying the Lorenz system.

Lorenz System is a system of differential equations, noted to have chaotic solutions for given parameters and the initial conditions. Lorenz attractor is the set of chaotic solution to the above system.

Using multi-dimensional arrays and multiple threads to increase the speed-up of the program. Generating high performance programs for the same of multi-core processor (OpenMP), cluster computers (MPI), and general-purpose graphic processing unit (Cuda C/C++).

Drawback of serial programs is the inability to run multiple objects to visually see the chaos with slight variations in the initial conditions.

INTRODUCTION

The Lorenz System is a system of differential equations, noted to have chaotic solutions for given parameters and the initial conditions, first studied by Edward Lorenz in 1962.

Lorenz attractor is the set of chaotic solutions of the Lorenz System. It is a part of an area of study, Chaos Theory.

The shape of the Lorenz attractor, when plotted graphically, represents that of a butterfly. The butterfly effect is a real-world implication of the Lorenz attractor.

This project is designed to implement parallel programs for the simulation of objects following Lorenz attractor satisfying the Lorenz system.

LITERATURE SURVEY – CHAOS THEORY

Chaos theory is a branch of mathematics focusing on the study of chaos—dynamical systems whose apparently random states of disorder and irregularities are actually governed by underlying patterns and deterministic laws that are highly sensitive to initial conditions.

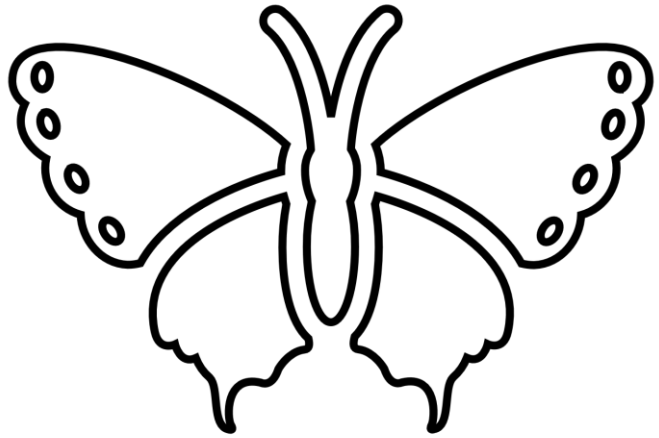
A commonly used definition, to classify a dynamical system as chaotic, it must satisfy three properties:

- It must be sensitive to initial conditions.
- It must be topologically transitive.
- It must have dense periodic orbits.

LITERATURE SURVEY

–

BUTTERFLY EFFECT



In chaos theory, the butterfly effect is the sensitive dependence on initial conditions in which a small change in one state of a deterministic nonlinear system can result in large differences in a later state.

It is metaphorically derived from the example of a tornado being influenced by minor perturbations such as a distant butterfly flapping its wings several weeks earlier.

LITERATURE SURVEY – ATMOSPHERIC CONVECTION

Atmospheric convection is the result of a temperature difference layer in the atmosphere. Different rate of temperature change with altitude within the air masses lead to instability. This instability with moist air masses causes thunderstorms, which are responsible for severe weather in the world.

HISTORY

Lorenz had been studying the simplified models describing the motion of the atmosphere, in terms of ordinary differential equations depending on few variables. In his 1962 article, he analyzed differential equations, for atmospheric convection, in a space of 12 dimensions, in which he numerically detects a sensitive dependence to initial conditions.

In 1963, with the help of Ellen Fetter, Lorenz simplified the above mathematical model to a system of differential equations, now known as the Lorenz equations.

LORENZ EQUATIONS

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

Here,

x is proportional to the rate on convection

y is proportional to the horizontal temperature variation

z is proportional to the vertical temperature variation

σ is proportional to the Prandtl number, ratio of momentum diffusivity to thermal diffusivity

ρ is proportional to the Rayleigh number, describes the behaviour of fluids when the mass density is non-uniform

β is proportional to certain physical dimensions

IMPLEMENTATION

We consider x, y, z as the coordinates of the 3-d cartesian plane.

$\sigma = 10, \rho = 28, \beta = 8/3.$
The stating points, x_0, y_0, z_0

Compute $\mathrm{d}\,x/\mathrm{d}\,t, \mathrm{d}\,y/\mathrm{d}\,t, \mathrm{d}\,z/\mathrm{d}\,t$

Multiply by $\mathrm{d}t$, the time since execution, to get $\mathrm{d}x, \mathrm{d}y, \mathrm{d}z$.

Add this to the previous coordinate to get the current coordinate.

To observe the chaos in the system, we start multiple objects at the same time with the same start points, to observe their behavior.

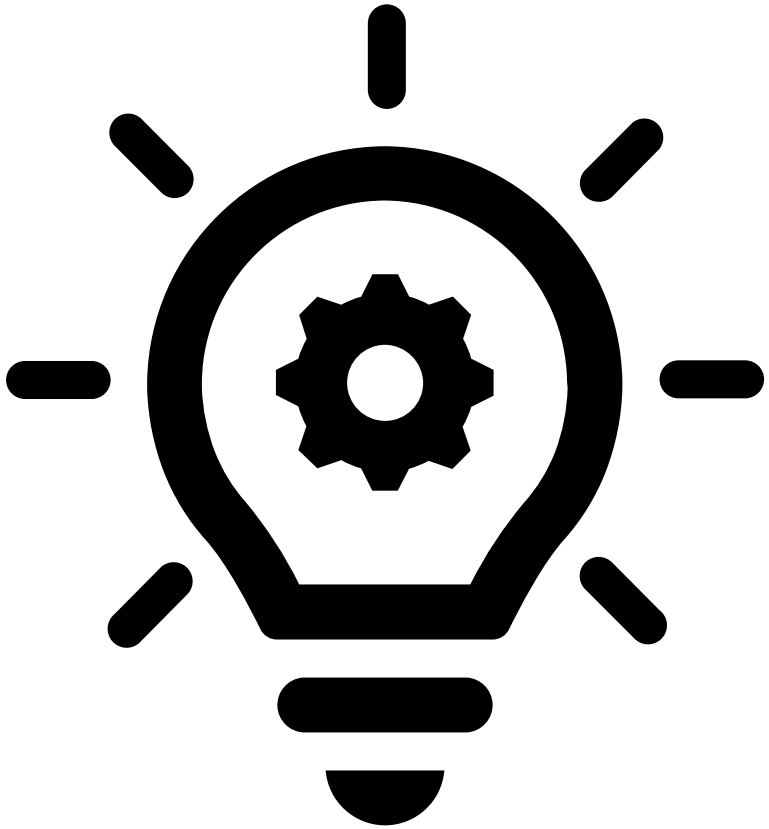
```
for (j = 0; j < n; j++)
{
    xnew = rk4vec(t[j], m, x + j * m, dt, lorenz_rhs);
    for (i = 0; i < m; i++)
    {
        x[i + (j + 1) * m] = xnew[i];
    }
    free(xnew);
}
```

DRAWBACKS

In the critical part of the serial algorithm of Lorenz attractor, to generate the value of the point at (j + 1) ⋆ m, we need the value at j ⋆ m.

Since the current point depends on the position of the previous value, the main challenge is to remove the dependency to achieve high speed up when multiple threads are used.

Multiple objects cannot be run at the same time in a serial code. So, to observe the chaos in the systems, a parallel program is necessary.

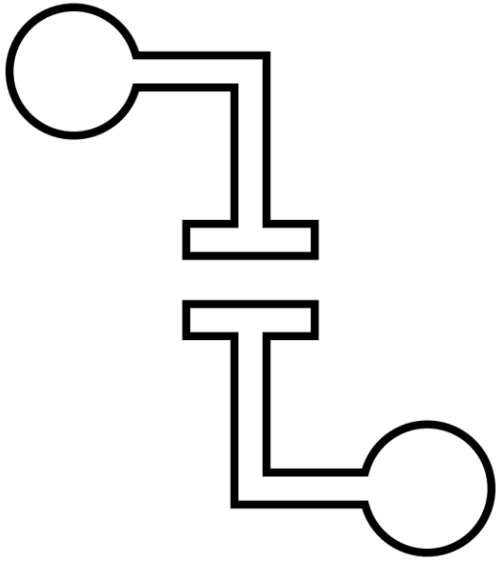


POSSIBLE SOLUTION

Multi-dimensional arrays to store the output coordinates, with multiple entries to the n-threads assigned for the program. Also, parallelize the execution of the different objects to get the synchronized output of the flow of the coordinates.

CONCLUSION

On perusal of the implementation and codes, the current parallelization factor is very low, on accounts on dependencies with previous values. This gives us a lot of room to apply parallelization techniques, to be learnt over the period of the course.



THEORETICAL ANALYSIS ON LORENZ ATTRACTOR

High Performance Computing Project Report

By,

SAI KAUSHIK SUDHAKARAN

CED18I044

Department of Computer Science and Engineering,

Indian Institute of Information Technology Design and Manufacturing

Kancheepuram, Chennai

August 2021

ACKNOWLEDGEMENT

I would like to express my gratitude toward staff of Department of Computer Science and Engineering of Indian Institute of Information Technology Design and Manufacturing Kancheepuram for providing me a great opportunity to complete a project on Parallelizing Lorenz Attractor.

My sincere thanks go to **Dr. Noor Mahammad SK, Assistant Professor, Department of Computer Science and Engineering.** Your valuable guidance and suggestions helped me in various phases of the completion of this project. I will always be thankful to you in this regard.

I am ensuring that this project was done by me and not copied from anywhere.

ABSTRACT

This project is designed to implement parallel programs for the simulation of objects following **Lorenz attractor** satisfying the Lorenz system.

Lorenz System is a system of differential equations, noted to have chaotic solutions for given parameters and the initial conditions. Lorenz attractor is the set of chaotic solution to the above system.

Using multi-dimensional arrays and multiple threads to increase the speed-up of the program. Generating high performance programs for the same of multi-core processor (OpenMP), cluster computers (MPI), and general-purpose graphic processing unit (Cuda C/C++).

Drawback of serial programs is the inability to run multiple objects to visually see the chaos with slight variations in the initial conditions.

TABLE OF CONTENT

INTRODUCTION

The Lorenz System is a system of differential equations, noted to have chaotic solutions for given parameters and the initial conditions, first studied by Edward Lorenz in 1962 [1].

Lorenz attractor is the set of chaotic solutions of the Lorenz System. It is a part of an area of study, Chaos Theory.

The shape of the Lorenz attractor, when plotted graphically, represents that of a butterfly. The butterfly effect is a real-world implication of the Lorenz attractor.

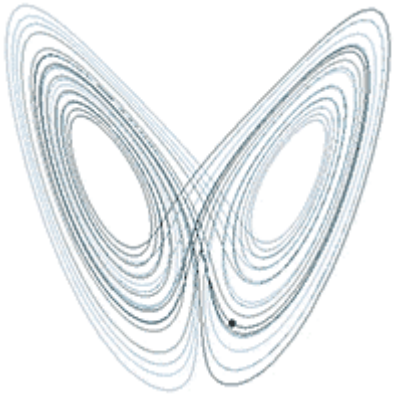


Figure 1: *Lorenz Attractor Animation*

This project is designed to implement parallel programs for the simulation of objects following Lorenz attractor satisfying the Lorenz system.

LITERATURE SURVEY

CHAOS THEORY

Chaos theory is a branch of mathematics focusing on the study of chaos—dynamical systems whose apparently random states of disorder and irregularities are actually governed by underlying patterns and deterministic laws that are highly sensitive to initial conditions [2].

A commonly used definition, to classify a dynamical system as chaotic, it must satisfy three properties:

- It must be sensitive to initial conditions.
- It must be topologically transitive.
- It must have dense periodic orbits.

BUTTERFLY EFFECT

In chaos theory, the butterfly effect is the sensitive dependence on initial conditions in which a small change in one state of a deterministic nonlinear system can result in large differences in a later state. [3]

It is metaphorically derived from the example of a tornado being influenced by minor perturbations such as a distant butterfly flapping its wings several weeks earlier.

ATMOSPHERIC CONVECTION

Atmospheric convection is the result of a temperature difference layer in the atmosphere. Different rate of temperature change with altitude within the air masses lead to instability. This instability with moist air masses causes thunderstorms, which are responsible for severe weather in the world. [4]

HISTORY

Lorenz had been studying the simplified models describing the motion of the atmosphere, in terms of ordinary differential equations depending on few variables. In his 1962 article, he analysed differential equations, for atmospheric convection, in a space of 12 dimensions, in which he numerically detects a sensitive dependence to initial conditions. [5]

In 1963, with the help of Ellen Fetter, Lorenz simplified the above mathematical model to a system of differential equations, now known as the Lorenz equations.

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

Here, x is proportional to the rate on convection

y is proportional to the horizontal temperature variation

z is proportional to the vertical temperature variation

σ is proportional to the Prandtl number, ratio of momentum diffusivity to thermal diffusivity

ρ is proportional to the Rayleigh number, describes the behaviour of fluids when the mass density is non-uniform

β is proportional to certain physical dimensions

ANALYSIS

If ρ is negative, there is only one equilibrium point, the origin. All orbits converge to the origin, which is a global attractor.

The corresponding system is stable only when

$$\rho < \sigma\left(\frac{\sigma + \beta + 3}{\sigma - \beta - 1}\right)$$

which can hold only positive for ρ if $\sigma > \beta + 1$, $\sigma > 0$ and $\beta > 0$.

Lorenz used the values $\sigma = 10$, $\rho = 28$, $\beta = 8 / 3$. The Lorenz system has chaotic solutions (but not all are chaotic). Almost all initial points tend to an invariant set, the Lorenz attractor.

System behaviour exhibited during different values of the constants, σ, ρ, β.

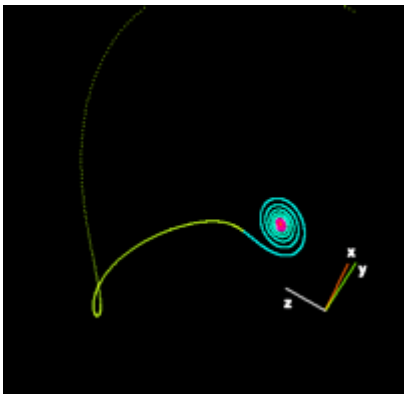


Figure 2: *$\sigma = 10, \rho = 14, \beta = 8 / 3$*

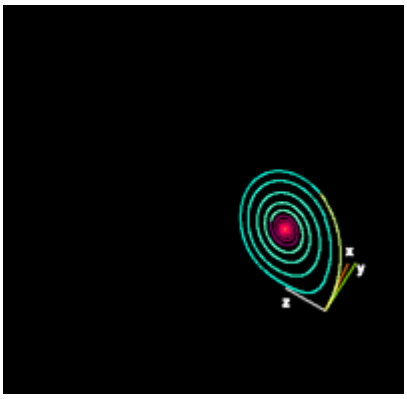


Figure 3: *$\sigma = 10, \rho = 13, \beta = 8 / 3$*

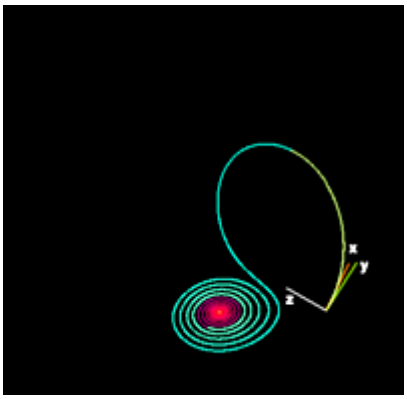


Figure 4: *$\sigma = 10, \rho = 15, \beta = 8 / 3$*

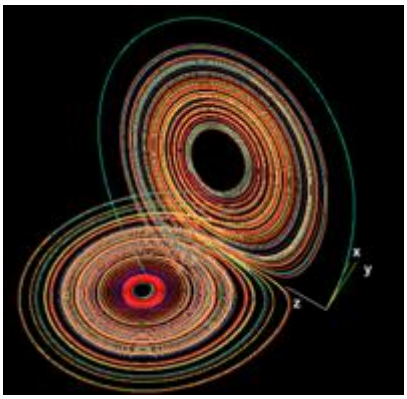
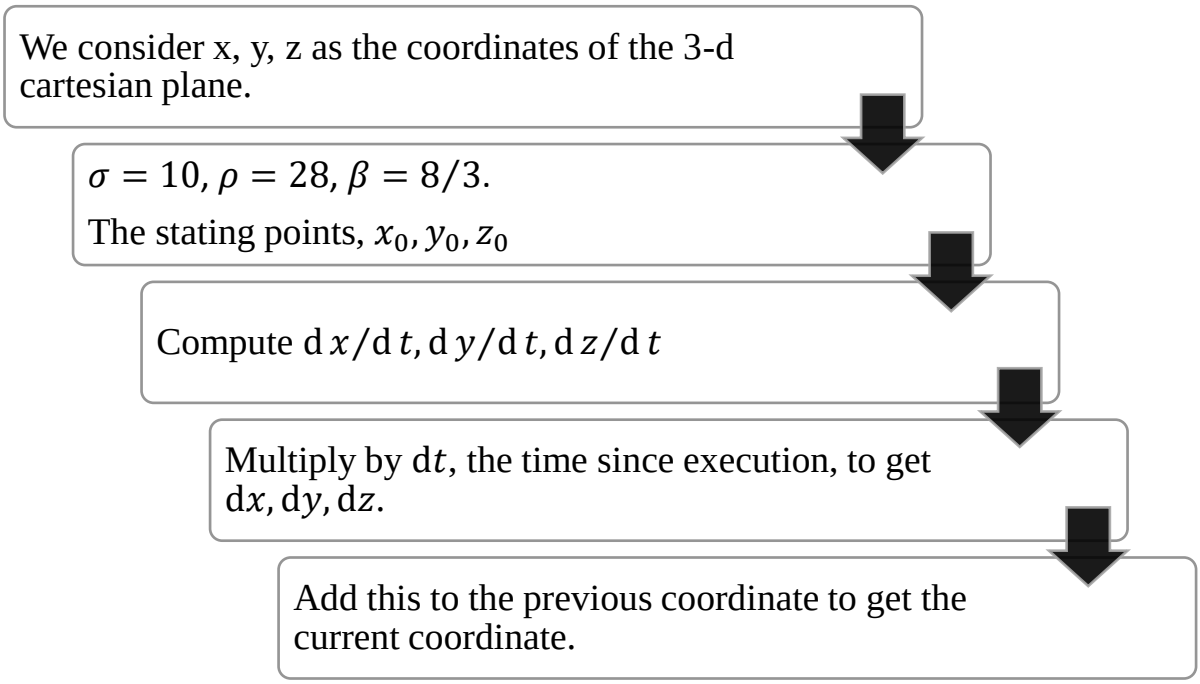


Figure 5: *$\sigma = 10, \rho = 28, \beta = 8 / 3$*

IMPLEMENTATION



To observe the chaos in the system, we start multiple objects at the same time with the same start points, to observe their behaviour.

Theoretically, due to delays due to scheduling of the initialization of the objects, they will have very small difference in the initial conditions. Due to which, the initial path of all the objects will remain the same, after which, they should diverge and create their own path but still inherit the butterfly shape.

DRAWBACKS

In the critical part of the serial algorithm of Lorenz attractor, to generate the value of the point at (j + 1) * m, we need the value at j * m. [6]

Since the current point depends on the position of the previous value, the main challenge is to remove the aforementioned dependency to achieve high speed up when multiple threads are used. Multiple objects cannot be run at the same time in a serial code. So, to observe the chaos in the systems, a parallel program is necessary.

```
        for (int i = start; i < end; i++)
        {
            coord = differential(coord);
            glBegin(GL_POINTS);
            glColor3f(1, 1, 1);
            glVertex3f(coord.x, coord.y, coord.z);
            glEnd();
            glFlush();
            glutSwapBuffers();
        }

Point differential(Point curr)
{
    Point diff;
    // 4 artih ops, 5 reads, 1 write
    diff.x = (sigma * (curr.y - curr.x)) * dt + curr.x;
    op += 4;
    // 5 arith ops, 6 reads, 1 write
    diff.y = (curr.x * (rho - curr.z) - curr.y) * dt + curr.y;
    op += 5;
    // 5 artih ops, 6 reads, 1 write
    diff.z = (curr.x * curr.y - beta * curr.z) * dt + curr.z;
    op += 5;
    // 1 arith op, 2 reads, 1 write
    diff.t = curr.t + dt;
    op += 1;
    return diff;
}
```

Figure 6: *Critical part of the algorithm*

CODE BALANCING

Current processor: Ryzen 7 5800H, 8 cores, 3.2 GHz

No. of flops supported by the system for double precision

= No of cores * Clock frequency * Double precision calculations

= 8 * 3.2 GHz * 4 flops/sec = 102.4 Gflops/sec

Each loop contains 15 floating point arithmetic operations.

For input size of 1000000, no. of floating-point operations

= 15 * 1000000 = 15 Mflops

The processor can handle a higher input size within 1 second.

No. of words used & modified

= 23 (19 reads, 4 writes) * 2 (Double datatype uses 8 bytes / 2 words) * 1000000

= 46M words

Code balance (B_C)

= No. of words used & modified / No. of flops in the code

= 46M / 15Mflops = 3.0667

Computational intensity

= 1 / Code balance (B_C)

= 1 / 3.0667 = 0.326

Machine balance (B_M)

= Memory bandwidth (B_{max}) / No. of flops of the system (P_{max})

= 3.2 GHz / 102.4 Gflops = 0.03125

Light speed of the loop (l)

= $min\left(1, \frac{B_M}{B_C}\right)$

= $min\ (1, 0.03125\ /\ 3.0667)$ = $min(1, 0.01019)$

= 0.01019

Performance of the loop (P)

= $min\left(P_{max}, \frac{B_{max}}{B_C}\right)$

= $min\left(102.4 \times\ 10^9, \frac{3.2 \times 10^9}{3.0667}\right)$ = $min(3.2 \times\ 10^9, 1.0434 \times\ 10^9\ )$

= 1.0434 Gflops/sec

POSSIBLE SOLUTION

Multi-dimensional arrays to store the output coordinates, with multiple entries to the n-threads assigned for the program. Also, parallelize the execution of the different objects to get the synchronized output of the flow of the coordinates.

SOFTWARES

- **C/C++**, programming language
- **OpenMP**, API for shared-memory parallel programming
- **MPI**, High performance Message Passing library
- **Cuda C/C++**, API for utilizing CUDA-enabled GPU for computation
- **Qt**, GUI library for displaying the output graph animations

APPLICATIONS

Lorenz equations are used to:

- Study atmospheric convection, used in weather predictions.
- Study thermal convection in the lithosphere
- Investigate the behaviour of dynamic systems that cannot be explained.

CONCLUSION

On perusal of the implementation and codes, the current parallelization factor is very low, on accounts on dependencies with previous values. This gives us a lot of room to apply parallelization techniques, to be learnt over the period of the course.

This problem statement was chosen as the student was intrigued by YouTube videos describing the physics of everyday events which showed the butterfly effect and chaos theory and how the said phenomena is used in one of the common real-world application, weather prediction.

REFERENCES

[1] Wikipedia, “Lorenz system - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Lorenz_system.

[2] Wikipedia, “Chaos theory - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Chaos_theory.

[3] Wikipedia, “Butterfly effect - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Butterfly_effect.

[4] Wikipedia, “Atmospheric convection - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Atmospheric_convection.

[5] E. Lorenz, “The statistical prediction of solutions of dynamic equations.,” Meteorological Society of Japan, Tokyo, 1962.

[6] J. Burkardt, “LORENZ_ODE - The Lorenz System,” Florida State University, [Online]. Available: https://people.sc.fsu.edu/~jburkardt/c_src/lorenz_ode/lorenz_ode.html.