Parallel Programming in Finite Difference Method to Solve Turing's Model of Spot Pattern*

Theodoret Putra Agatho†
Departement of Informatics
Universitas Atma Jaya Yogyakarta
Yogyakarta
190710233@students.uajy.ac.id

Pranowo
Departement of Informatics
Universitas Atma Jaya Yogyakarta
Yogyakarta
pranowo@uajy.ac.id

ABSTRACT

Turing's model is a model contains reaction-diffusion equation that capable to form skin patterns on an animal. In this paper, Turing's model was investigated, with the model improvisation by Barrio *et al.* [12], in parallel programming to shown its speed up impact. The parallel programming managed to speed up the process up to 8.9 times while retaining the quality of the result, compared to traditional programming.

CCS CONCEPTS

• Mathematics of Computing • Numerical Analysis • Parallel algorithms

KEYWORDS

Turing's model, Heun's method, finite difference, parallel programming

1 Introduction

Animal skins are different but contain a specific pattern at a certain level. For example, Zebra: each Zebra has different stripes, but it has stripes patterns in common which differentiate it (for example) from the Lion regardless both species have stripes patterns. This commonality raises the possibility of simulating those patterns in consistency even if randomness or similarity is at a certain level. Turing proposed that morphogens can be considered in reaction-diffusion systems, and skin pigments may be regarded as morphogens [1]. McNamara [2] defines Morphogens as a biological process that causes a tissue or organ to develop its shape by controlling the spatial distribution of cells during embryonic development. With these ideas, it can be assumed that: skin patterns can be formed by simulating the growth of the skin.

Turing's ideas have been studied extensively [3, 4, 7-11]. In those studies, different models and numerical methods have been employed: Sekimura *et al*. [3] used the Gierer-Meinhardt model to simulate the butterfly wing pattern. To solve the asymmetrical shape of the butterfly wing, the Finite volume method was used to transform partial equations in model into a distinct triangle grid. Shakeri and Dehghan [4] developed a hybrid finite volume spectral element method and applied it to the Turing system generated by the Schnakenberg model. The authors yielded low error compared to some approximate solutions presented in refs [5,6]. Kim *et al*. [9] studied an explicit time-stepping scheme for the pattern formation of Zebra in reaction-diffusion systems on evolving surfaces using the model proposed by Barrio *et al.* [12]. The numerical method used is based on a simple discretization scheme of the Laplace-Beltrami operator, which contains Euler's method and finite difference method. Numerical methods of finite difference for general models of chemotaxis systems were explored in depth by Smiley [11]. Sarra [10] explored the finite difference method by proposing a local RBF method. The author found it is well applicable to processes governed by reaction-diffusion type equations.

From the preceding discussion, it's clear that Turing's ideas are viable, and further research is possible. As has been recommended by refs [10]: the GPU usage in accelerating model progress in forming pattern, parallel programming by using GPU was proposed in this paper to solve the Turing model for generating spot patterns. Heun's method was used to solve semi-algebraic equations in the Turing model.

A spotted pattern is a pattern that has a dot-like shape scattered and forms the pattern. This pattern can be found in animals such as fish [12] or cats [13] family. Even though jaguars are spotted to have this pattern at the time they were cubs, as they grow into adults, the pattern turns into dark rosettes like [7].

The finite difference method was proposed to discretize the semi-algebraic of the mathematical model and the Heun's method to obtained algebraic results. The finite difference method is a simple discretize method but it's possible to be used in complex patterns [9-11] after further improvisation. The Euler's method was used by refs [9] but Heun's method was proposed as it's a generalized Euler's method [14], therefore an improved version [15]. The numerical solution of the model was implemented using GPU parallel programming to speedup calculation.

The paper was written as follows: The ideas are introduced in section 1. Section 2 is theory background which contains an explanation of basic knowledge in used equations and parallel programming. Model results are shown and discussed in section 3. In section 4, conclusions are drawn.

2 Theory Background

2.1 Turing's Model: Reaction & Diffusion

The reaction & diffusion system equation is written as follow, where the pair of u and v represent the activator and inhibitor, respectively:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f(u,v) \tag{1}$$

$$\frac{\partial v}{\partial t} = d\nabla^2 v + g(u,v) \tag{2}$$

2.2 Spot Pattern Equations

The equation proposed by Barrio *et al.* [12] was adopted, where the equation of the reaction-diffusion system was transformed to encompass the required features, included the boundary conditions. As a result, the steady state is at the point (0,0), and variables should be viewed as deviations from it [7]. The equations are as follow:

$$\frac{\partial u}{\partial t} = D\delta\nabla^2 u + \alpha u + v - r_2uv \; - \alpha r_3uv^2 \tag{3}$$

$$\frac{\partial v}{\partial t} = \delta\nabla^2 v + \gamma u + \beta v + r_2uv + \alpha r_3uv^2 \tag{4}$$

The model is interaction quadratic (r_3) and cubic (r_2) to u and v. A cubic interaction favors stripes, and a quadratic one produces spot patterns [12]. The D is the ratio of u to v and the δ is a scaling variable.

The hyperparameters values of the first stage: "initial distributions" from refs [7] were used as the hyperparameters value. The hyperparameters value as follow:

$$D = 0.45,\quad \delta = 6,\quad \alpha = 0.899,\quad \beta = -0.91,$$
$$r_2 = 2,\quad r_3 = 3.5,\quad \gamma = -\alpha$$

While the initial values of u and v were initialized randomly 0 and 1.

2.3 Heun's Method and Finite Difference

Consider following Turing's model equation, from simplified Barrio equations (3,4):

$$\frac{\partial u}{\partial t} = D\delta\nabla^2 u + f(u,v) \tag{5}$$

$$\frac{\partial v}{\partial t} = \delta\nabla^2 v + g(u,v) \tag{6}$$

Where,

$$f(u,v) = \alpha u + v - r_2uv \; - \alpha r_3uv^2 \tag{7}$$

$$g(u,v) = \gamma u + \beta v + r_2uv + \alpha r_3uv^2 \tag{8}$$

The Heun's method with assigned finite difference, the first orders (ku_1, kv_1) are follow:

$$ku_1 = D\delta\left[\frac{(u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j})}{\Delta x^2} + \frac{(u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1})}{\Delta y^2}\right] + f(u^n, v^n) \tag{9}$$

$$kv_1 = \delta\left[\frac{(v^n_{i+1,j} - 2v^n_{i,j} + v^n_{i-1,j})}{\Delta x^2} + \frac{(v^n_{i,j+1} - 2v^n_{i,j} + v^n_{i,j-1})}{\Delta y^2}\right] + g(u^n, v^n) \tag{10}$$

Where n is time step, i is space step of y, and j is space step of x. u and v are updated temporary for the requirements of the second orders (ku_2, kv_2), as follows:

$$u^{n+\frac{1}{2}} = u^n + \Delta t(ku_1) \tag{11}$$

$$v^{n+\frac{1}{2}} = v^n + \Delta t(kv_1) \tag{12}$$

Then, the second orders (k_2) are obtained as follows:

$$ku_2 = D\delta\left[\frac{\left(u^{n+\frac{1}{2}}_{i+1,j} - 2u^{n+\frac{1}{2}}_{i,j} + u^{n+\frac{1}{2}}_{i-1,j}\right)}{\Delta x^2} + \frac{\left(u^{n+\frac{1}{2}}_{i,j+1} - 2u^{n+\frac{1}{2}}_{i,j} + u^{n+\frac{1}{2}}_{i,j-1}\right)}{\Delta y^2}\right] + f(u^{n+\frac{1}{2}}, v^{n+\frac{1}{2}}) \tag{13}$$

$$kv_2 = \delta\left[\frac{\left(v^{n+\frac{1}{2}}_{i+1,j} - 2v^{n+\frac{1}{2}}_{i,j} + v^{n+\frac{1}{2}}_{i-1,j}\right)}{\Delta x^2} + \frac{\left(v^{n+\frac{1}{2}}_{i,j+1} - 2v^{n+\frac{1}{2}}_{i,j} + v^{n+\frac{1}{2}}_{i,j-1}\right)}{\Delta y^2}\right] + g(u^{n+\frac{1}{2}}, v^{n+\frac{1}{2}}) \tag{14}$$

Within both orders are known, u and v are updated as follows:

$$u^{n+1} = u^n + \frac{\Delta t}{2}(ku_1 + ku_2) \tag{15}$$

$$v^{n+1} = v^n + \frac{\Delta t}{2}(kv_1 + kv_2) \tag{16}$$

2.4 Parallel Programming

In this paper, parallel programming was done with Compute Unified Device Architecture (CUDA) from NVIDIA, which allows the usage of Graphics Processing Units (GPU).

In GPU execution, tasks in the program are spread to threads, respectively. Each thread is organized into each block in grids and executed at the same time or in parallel. Blocks and grids can be considered as the addresses to organize threads in physical storage. In parallel programming, calculations were organized into threads, grids, and blocks to allow calculations conducted in parallel and achieve the result as if in serial execution. Further details regarding programming in CUDA C can be found in ref Cheng *et al.* [16].

Within the equations are established by precede discussions, the GPU algorithm was programmed within the equations order as in Fig. 2.1. A solid box denotes sequences cooperation between CPU and GPU while calculation process in GPU denoted by dotted line box.

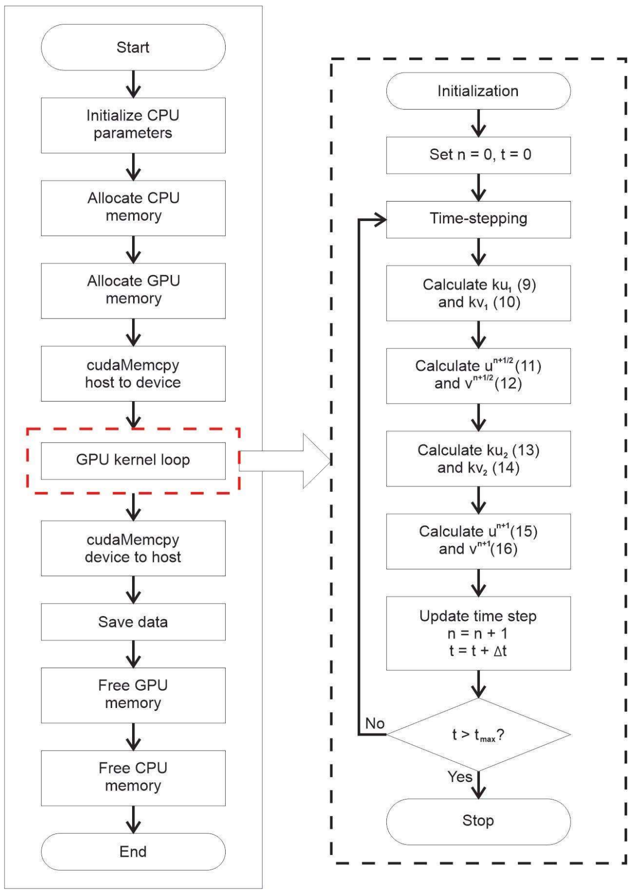
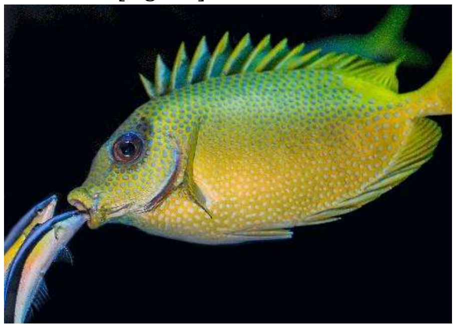
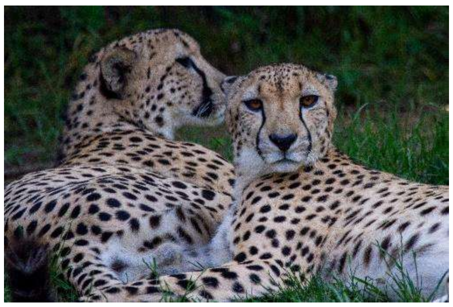

Start
Initialize CPU parameters
Allocate CPU memory
Allocate GPU memory
cudaMemcpy host to device
GPU kernel loop
cudaMemcpy device to host
Save data
Free GPU memory
Free CPU memory
End
Initialization
Set n = 0, t = 0
Time-stepping
Calculate ku_1 (9) and kv_1 (10)
Calculate $u^{n+1/2}$(11) and $v^{n+1/2}$(12)
Calculate ku_2 (13) and kv_2 (14)
Calculate u^{n+1}(15) and v^{n+1}(16)
Update time step n = n + 1 t = t + Δt
No
$t > t_{max}$?
Yes
Stop

Figure 2.1: **Parallel programming of Turing’s model flowchart.**

3 Result and Discussion

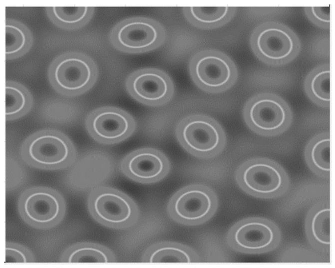
The program was run on a computer with NVIDIA GeForce GTX 1650 8GB and Intel Core i7 Gen 9th. The model managed to achieve similar result with spot pattern which was generated by Liu *et al.* [Liu] at the first stage, while GPU managed to form an identical result [Fig 3.2. a] to CPU result [Fig 3.2. b]. The formed patterns [Fig 3.2.] were also comparable to spot patterns on animals [Fig 3.1.].


(a)


(b)

(c)

Figure 3.1: **Spot patterns are spotted on a rabbitfish (a) [17], cheetahs (b) [18], and cats(c) [19].**


(a)

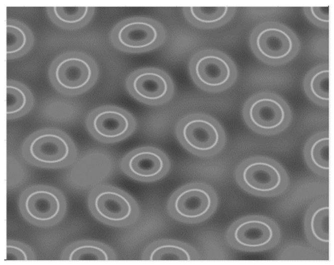
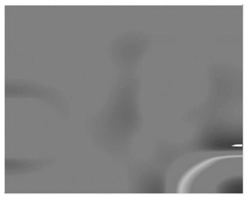
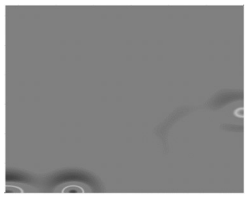

(b)

Figure 3.2: **Patterns generated using finite difference method and Heun’s method with GPU (a) and CPU (b) in 128 x 128 grids.**


(a)


(b)

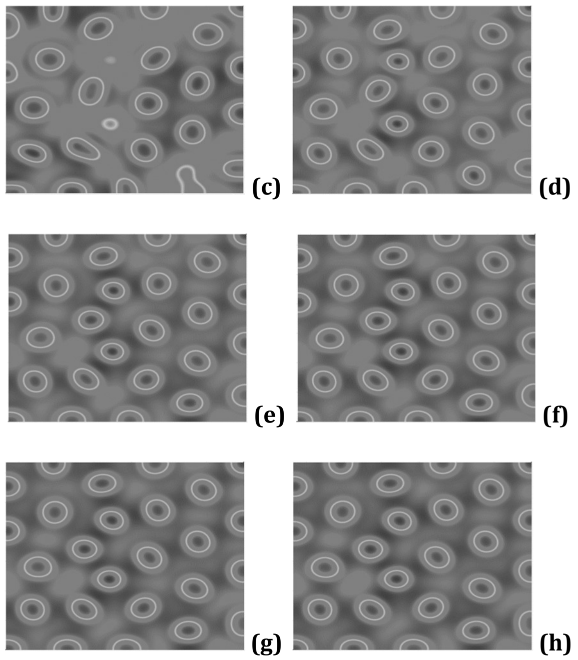


Figure 3.3: **Evolution of patterns generated using finite difference method and Heun's method in 128 x 128 grids: 3750th (a), 7500th (b), 11250th (c), 15000th (d), 18750th (e), 22500th (f), 26250th (g), and 30000th (h) iterations.**

Pattern processing times were noted in the table [Tab 3.1.]. GPU achieved 3.5472 times (in the smallest size: 128 x 128) to 8.9280 times (in the largest size: 1024 x 1024) faster than CPU. As the grid size gets larger, the time required to execute gets longer; this fact is true on GPU and CPU, but the acceleration of GPU also increases.

Grid Size	GPU Time (ms)	CPU Time (ms)	Accel (times)
128 x 128	11355	40279	3.5472
256 x 256	26828	182649	6.8081
512 x 512	91682	770620	8.4054
1024 x 1024	359311	3207949	8.9280

Table 3.1: **Time spend of GPU and CPU for each grid size.**

4 Conclusion

Turing's model to generate spot pattern purposed by Barrio *et al* [12] was used to investigate parallel programming impact in speed of processing model. In this paper, model managed to yield resemble pattern with patterns are spotted on fish [Fig 3.1.a], cheetah [Fig 3.1.b], and cat [Fig 3.1.c]. The implementation of parallel programming to the model managed to achieve 3.5472 and 8.9280 times speed up compare to traditional computing while maintaining the quality of result.

ACKNOWLEDGMENTS

This paper is part of learning progress in curriculum "Program Merdeka Belajar – Kampus Merdeka" and a part of PDUPT research.

REFERENCES

[1] A. M. Turing. 1952. The Chemical Basis of Morphogenesis. *Philos. Trans. R. Soc. Lond. Ser. B. Biol.* Vol. 237, No. 641 (Aug. 1952), 37-72. DOI: https://doi.org/10.1098/rstb.1952.0012

[2] L. M. McNamara. 2017. Comprehensive Biomaterials II, 2.10 Bone as a Material, 202-227. Elsevier. DOI: https://doi.org/10.1016/B978-0-12-803581-8.10127-4

[3] T. Sekimura, A. Madzvamuse, A. J. Wathen, and P. K. Maini, 2000. A model for colour pattern formation in the butterfly wing of Papilio Dardanus. In *Proceedings of the Royal Society B Biological Sciences,* 7 May, 2000, London, Royal Society. DOI: https://doi.org/10.1098/rspb.2000.1081

[4] F. Shakeri and M. Dehghan. 2011. The finite volume spectral element method to solve Turing models in the biological pattern formation. *Comput. Math. With Appl.* 62, 12 (Dec. 2011), 4322-4336. DOI: https://doi.org/10.1016/j.camwa.2011.09.049

[5] A. Madzvamuse, A. J. Wathen, P. K. Maini, 2003. A moving grid finite element method applied to a model biological pattern generator. *J. Comput. Phys* 190, 2 (Sept. 2003), 478-500. DOI: https://doi.org/10.1016/S0021-9991(03)00294-8

[6] J. Zhu, Y. -T. Zhang, S. A. Newman, M. Alber, 2009. Application of discontinuous Galerkin methods for reaction-diffusion systems in developmental biology. *J. Sci. Comput.* 40, (July 2009), 391-418. DOI: https://doi.org/10.1007/s10915-008-9218-4

[7] R. T. Liu, S. S. Liaw, and P. K. Maini, 2005. Two-stage Turing model for generating pigment patterns on the leopard and the jaguar. *Phys. Rev. E* 74, 1, 011914 (July 2006). DOI: https://doi.org/10.1103/PhysRevE.74.011914

[8] J. H. E. Cartwright. 2001. Labyrinthine Turing Pattern Formation in the Cerebral Cortex. *J. Theor. Biol.* 217, 1 (March 2002), 97-103. DOI: https://doi.org/10.1006/jtbi.2002.3012

[9] H. Kim, A. Yun, S. Yoon, C. Lee, J. Park, J. Kim, 2020. Pattern formation in reaction-diffusion systems on evolving surfaces. *Comput. Math. With Appl.* 80, 9 (Nov. 2020), 2019-2028. DOI: https://doi.org/10.1016/j.camwa.2020.08.026

[10] S. A. Sarra. 2012. A local radial basis function method for advection-diffusion-reaction equations on complexly shaped domains. *App. Math. and Comp.* 218 (2012), 9853-9865. DOI: https://doi.org/10.1016/j.amc.2012.03.062

[11] M. W. Smiley. 2008. An efficient implementation of a numerical method for a chemotaxis system. *J. Comput. Math.* 86, 2 (Dec. 2008), 219-235. DOI: https://doi.org/10.1080/00207160701864475

[12] R. A. Barrio, C. Varea, J. L. Aragon, 1999. A Two-dimensional Numerical Study of Spatial Pattern Formation in Interacting Turing Systems. *Bull. Math. Biol.* 61, (May 1999), 483-505. DOI: https://doi.org/10.1006/bulm.1998.0093

[13] W. L. Allen, I. C. Cuthill, N. E. Scott-Samuel, R. Baddeley, 2010. Why the leopard got its spots: relating pattern development to ecology in felids. In *Proceedings of the Royal Society B Biological Sciences,* 20 Oct., 2010, London, Royal Society. DOI: https://doi.org/10.1098/rspb.2010.1734

[14] J. C. Butcher. 2008. *Numerical Methods for Ordinary Differential Equations* (2nd. ed.). John Wiley & Sons, Chichester, West Sussex, UK.

[15] A. H. Workie. 2020. New Modification on Heun's Method Based on Contraharmonic Mean for Solving Initial Value Problems with High Efficiency. *J. Math.* 2020, Article 6650855 (Dec. 2020). DOI: https://doi.org/10.1155/2020/6650855

[16] J. Cheng, M. Grossman, T. McKercher, 2014. *Professional CUDA C Programming*. John Wiley & Sons, Indianapolis, Indiana, US.

[17] Unsplash. 2018. Unsplash: The internet's source for visuals. Retrieved from https://unsplash.com/photos/j25BolXb0mk.

[18] Unsplash. 2021. Unsplash: The internet's source for visuals. Retrieved from https://unsplash.com/photos/fkLgfKJZQCE.

[19] Unsplash. 2017. Unsplash: The internet's source for visuals. Retrieved from https://unsplash.com/photos/ZBvCYHOo-ww.