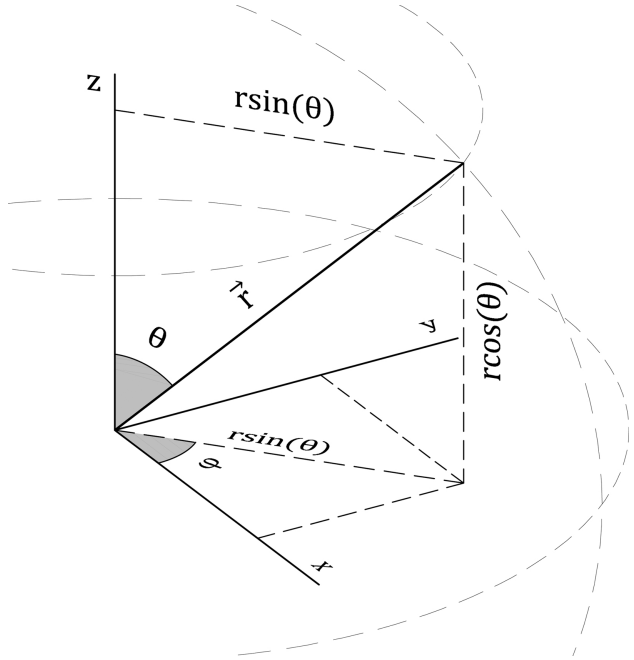



**Figure 5:** Spherical coordinates, where the radial distance is $r$, the polar angle is $\theta$, and the azimuth angle is $\phi$.

$$L_{n+l}^{2l+1}(\rho) = \sum_{k=0}^{n-l-1} (-1)^{k+1} \frac{[(n+l)!]^2}{(n-l-1-k)!\,(2l+1+k)!k!}\rho^k \quad (22)$$

In the angular part, we use the following relation:

$$Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)\,(l-|m|)!}{4\pi\,(l+|m|)!}} P_l^{|m|}(\cos(\theta))\, e^{im\phi} \quad (23)$$

where the constant $m$ is an integer, it represents the **magnetic quantum number** and can assume the values $m = \{-l, ..., +l\}$. The term $P_l^{|m|}(\cos(\theta))$ comes from the next two relations (24) and (25), respectively known as polynomials of Legendre associated and Legendre polynomials.

$$P_l^{|m|}(x) = (-1)^{|m|}\left(1-x^2\right)^{|m|/2} \frac{d^{|m|}}{dx^{|m|}} P_l(x) \quad (24)$$

$$P_l(x) = \frac{1}{2^l l!}\frac{d^l}{dx^l}\left(x^2-1\right)^l \quad (25)$$

At this point, we have all the terms of the equation (19) available, and we can proceed by calculating the probability density numerically. Moreover, as an alternative, at least for the first quantum numbers, it is also possible to perform the calculations by the wave functions present in Table 1. Therefore, whichever path we choose, we must obtain the following probability density:

**Table 1:** First normalized wave functions of the hydrogen atom. This is an adaptation of the table 6.1 from [31].

| $n$ | $l$ | $m$ | $\psi_{n,l,m}(r,\theta,\phi)$ |
|---|---|---|---|
| 1 | 0 | 0 | $\frac{1}{\sqrt{\pi}a_0^{3/2}} e^{-r/a_0}$ |
| 2 | 0 | 0 | $\frac{1}{4\sqrt{2\pi}a_0^{3/2}}\left(2-\frac{r}{a_0}\right) e^{-r/2a_0}$ |
| 2 | 1 | 0 | $\frac{1}{4\sqrt{2\pi}a_0^{3/2}} \frac{r}{a_0} e^{-r/2a_0} cos\ \theta$ |
| 2 | 1 | $\pm1$ | $\frac{1}{8\sqrt{3\pi}a_0^{3/2}} \frac{r}{a_0} e^{-r/2a_0} sin\ \theta\ e^{\pm i\phi}$ |
| 3 | 0 | 0 | $\frac{1}{81\sqrt{3\pi}a_0^{3/2}}\left(27-18\frac{r}{a_0}+2\frac{r^2}{a_o^2}\right) e^{-r/2a_o}$ |
| 3 | 1 | 0 | $\frac{1}{81\sqrt{3\pi}a_0^{3/2}}\left(6-\frac{r}{a_0}\right) \frac{r}{a_0} e^{-r/3a_0}\ cos\ \theta$ |
| 3 | 1 | $\pm1$ | $\frac{1}{81\sqrt{3\pi}a_0^{3/2}}\left(6-\frac{r}{a_0}\right) \frac{r}{a_0} e^{-r/3a_0}\ sin\ \theta\ e^{\pm i\phi}$ |
| 3 | 2 | 0 | $\frac{1}{81\sqrt{6\pi}a_0^{3/2}} \frac{r^2}{a_0^2} e^{-r/3a_0}\ \left(3\ cos^2\ \theta-1\right)$ |
| 3 | 2 | $\pm1$ | $\frac{1}{81\sqrt{\pi}a_0^{3/2}} \frac{r^2}{a_0^2} e^{-r/3a_0}\ sin\ \theta\ cos\ \theta\ e^{\pm i\phi}$ |
| 3 | 2 | $\pm2$ | $\frac{1}{162\sqrt{\pi}a_0^{3/2}} \frac{r^2}{a_0^2} e^{-r/3a_0}\ sin^2\ \theta\ e^{\pm 2i\phi}$ |

$$|\psi_{nlm}(r,\theta,\phi)|^2 = |R_{nl}(r)|^2 |Y_m^l(\theta,\phi)|^2 \quad (26)$$

Note that in the equation (23) there is an exponential term that carries the imaginary number $i$, making $\psi_{nlm}(r,\theta,\phi)$ non-measurable, but $|Y_m^l(\theta,\phi)|^2$ does not contain the imaginary number [8] and independent of $\phi$. So, $|\psi_{nlm}(r,\theta,\phi)|^2$ is also real and is a function only of the coordinates $r$ and $\theta$.

Now we got $|\psi_{nlm}(r,\theta,\phi)|^2$ already explained, and we can determine its domain and the range to apply the MCM, as we have done before. So, the first thing we need to do is to be able to switch from cartesian to a spherical coordinate system. To achieve this, we can extract this information from Figure 5 to get the relation (27) below.

$$\begin{aligned} r &= \sqrt{x^2+y^2+z^2} \\ \theta &= arc\ cos\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right) \end{aligned} \quad (27)$$

For the domain of equation (26), we know that $\theta \in [0,\pi]$, and $r \in [0,+\infty)$. Since the radius $r$ has no upper limit, it is impossible to search the electron in the entire space. Alternatively, we will check all volume by an arbitrary box which contains the location most likely to find the single electron.

Therefore, we place a box of dimensions $\Delta x$, $\Delta y$, $\Delta z$, centralized at the origin of the coordinate system. Then we use the relation (27) to evaluate $r_L = (1/2)\sqrt{\Delta x^2+\Delta y^2+\Delta z^2}$. This $r_L$ is the magnitude of a vector that goes from the origin to any corner and will be used as the upper limit of the domain.

In respect of range, $|\psi_{nlm}(r,\theta,\phi)|^2$ is lower bounded by 0, and it reaches different top values depending on the quantum numbers. Unfortunately, finding a generic function that calculates the global maximum for each set

[8] The product between a number and its conjugate always gives a non-negative real number: $z \cdot z^* = |z|^2$

involving all of them is no easy task. A first alternative would be to find a solution analytically for each case, but this process becomes increasingly tedious as quantum numbers grow. Another option would be to use some searching process to seek global maximum numerically [41], after all, we have the computer available. Instead, let us proceed more smoothly.

The simplest way is to split the space [9] into a $2D$ regular grid, over the $r$ and $\theta$ "axis", and for each node (grid point) we calculate $|\psi_{nlm}(r,\theta,\phi)|^2$. Then we update a variable that stores the highest value $M$ found, and then increase it to ensure it is above the global maximum of this function. The Figures 6, 7, and 8 exemplify this method.

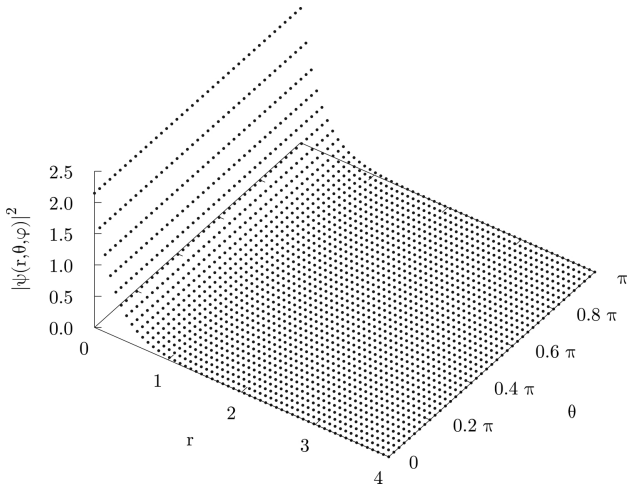



**Figure 6:** The probability density as a function of $r$ and $\theta$ for the hydrogen atom, with quantum numbers $n = 1$, $l = 0, m = 0$. The points are projections of nodes in the function $|\psi(r,\theta,\phi)|^2$.

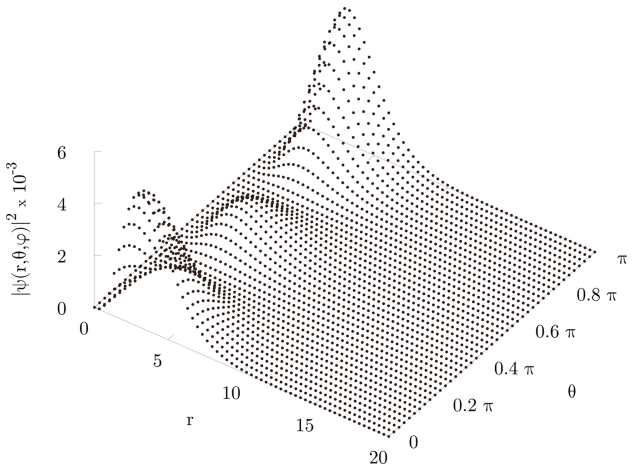



**Figure 7:** The probability density as a function of $r$ and $\theta$ for the hydrogen atom, with quantum numbers $n = 3$, $l = 2, m = 0$. The points are projections of nodes in the function $|\psi(r,\theta,\phi)|^2$.

[9] For example, $r$ and $\theta$ starts at 0 and reaches $r_L$ and $\pi$ respectively, so if the length $r_L$ is divided by 5 and angle $\pi$ by 4, then we have 30 nodes ($30 = (5+1) \times (4+1)$) to check.

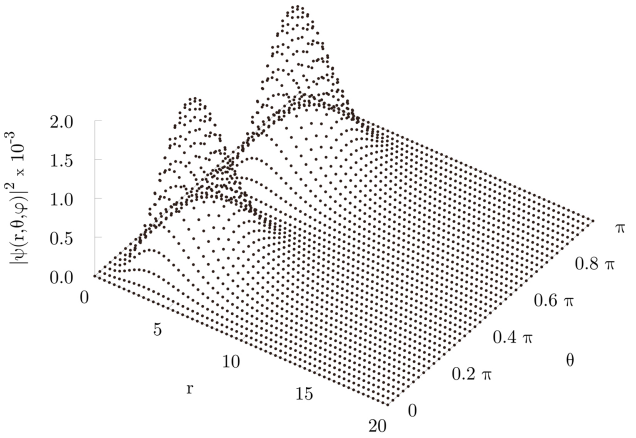



**Figure 8:** The probability density as a function of $r$ and $\theta$ for the hydrogen atom, with quantum numbers $n = 3$, $l = 2, m = 1$. The points are projections of nodes in the function $|\psi(r,\theta,\phi)|^2$.

This procedure may not be elegant, but it is practical. If the reader is uncomfortable about this, do the same with the problem in $1D$, then he will realize that the method will discard points above the function $|\psi_n(x)|^2$ in any way.

The final comment should be made about the Legendre associated polynomials presented in the equation (24). Some environments operate with algebraic or symbolic computations to interpret equations like that, but to reproduce this method in several programming languages, we must soften this step. In this case, it is better to use suitable libraries for this purpose, because the implementation of simpler algorithms may cause problems of numerical instability [42]. A good choice for the C++ programming language is the *BOOST* library [43] and the *SHTOOLS* [44] for FORTRAN and Python.

Finally, with all this information gathered, we have the conditions to represent the orbitals of the hydrogen atom. For that, we will adapt the algorithm seen in the examples of the confined particle in boxes, presented in the following.

1. Set the quantum numbers $n \in [1, +\infty)$, $l \in [0, n-1]$, $m \in [-l, l]$.
2. Set the box dimensions $\Delta x$, $\Delta y$ e $\Delta z$.
3. Estimate an upper limit for $M$ above of $|\psi_{nlm}(r,\theta,\phi)|^2$.
4. Set an amount *p-points* to plot.
5. Start an integer *i-counter* and reset it to zero.
6. Generate a random $x \in [-\Delta x/2, \Delta x/2]$, $y \in [-\Delta y/2, \Delta y/2]$ e $z \in [-\Delta z/2, \Delta z/2]$.
7. Evaluate $r$ and $\theta$ as the functions of $x$, $y$ e $z$.
8. Generate a random $w \in [0, M]$.
9. If $w \leq |\psi_{nlm}(r,\theta,\phi)|^2$, them $x$, $y$ and $z$ are stored in the list, and add $+1$ to $i$.
10. Repeat all steps from the $6th$ until $i$ equals $p$.
11. Make a graph with the coordinates from the list.

To estimate the upper limit $M$, we calculate $r_L$ to set up the grid, then we scan it to get an approximate global

maximum of $|\psi_{nlm}(r,\theta,\phi)|^2$, and finally, we add 5% over this value found. To make this grid, we suggest that use nearly 50 nodes per Angström along of the $r$ radius, and 160 nodes over the $\theta$ angle.

For plotting the orbitals in $2D$, turn the box into a rectangle by choosing the null value for some side. For example, if the interest is in the $x-y$ plane, then $\Delta z = 0$, and so on. We show some orbitals in Figures 9, 10 and 11.

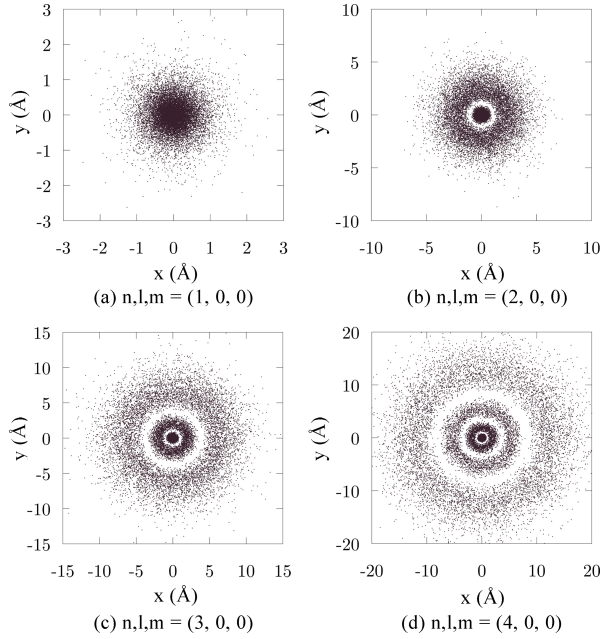



**Figure 9:** The simulated hydrogen atom with fifteen thousand points.

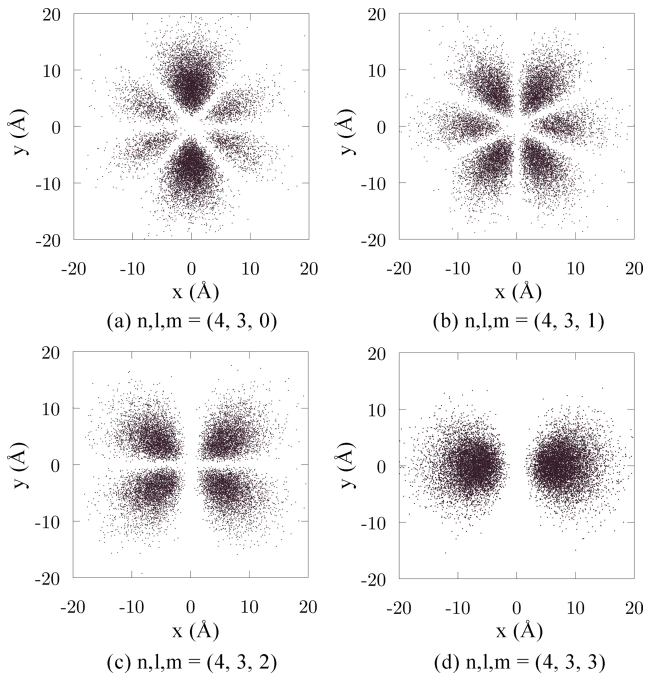



**Figure 10:** The simulated hydrogen atom with fifteen thousand points.

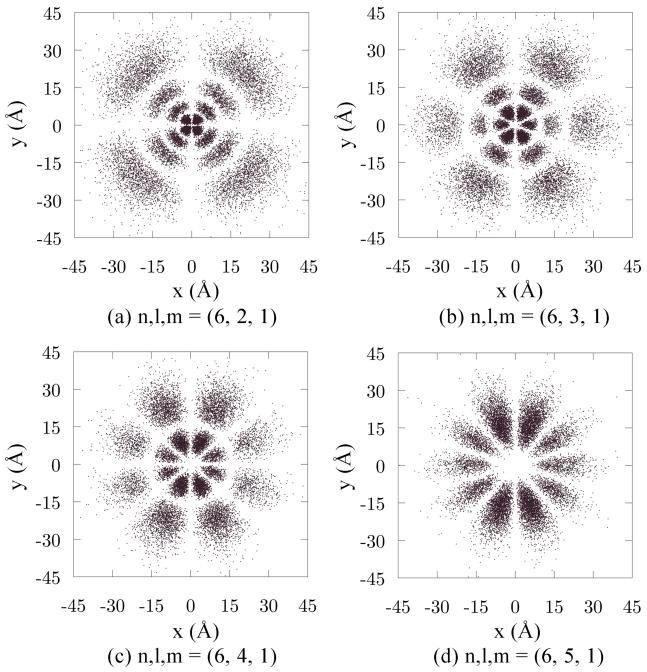



**Figure 11:** The simulated hydrogen atom with fifteen thousand points.

## 6. Conclusion

This paper shows in a heuristic way that it is easy to represent the orbitals of the hydrogen atom using basic quantum mechanics and coding skills.

We also made all source code available at a third-party platform [45], for any uses under the MIT license.

## Acknowledgment

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil* (CAPES) - Finance Code 001.

## References

[1] A. Lakhtakia, *Models and Modelers of Hydrogen* (World Scientific Publishing Company, Singapura 1996).
[2] H. Fischler and M. Lichtfeldt, International Journal of Science Education **14**, 181 (1992).
[3] D.T. Cromer, J. Chem. Educ. **45**, 626 (1968).
[4] B. Coto, A. Arencibia and I. Suárez, Computer Applications in Engineering Education **24**, 765 (2016).
[5] B.G. Moore, J. Chem. Educ. **77**, 785 (2000).
[6] M. Goto and V.M. Aquino, Semina: Ciências Exatas e Tecnológicas **13**, 255 (1992).
[7] V.M. Aquino, V.C. Aguilera-Navarro, M. Goto and H. Iwamoto, American Journal of Physics **69**, 788 (2001).
[8] E.A. Silver, Journal of the Operational Research Society **55**, 936 (2004).
[9] M. Shlomo and S. Mordechai, *Applications of Monte Carlo Method in Science and Engineering* (IntechOpen, London, 2011).

[10] D.P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics, Second Edition* (Cambridge University Press, New York, 2005).
[11] V.L. Líbero, Rev. Bras. Ens. Fis. **22**, 346 (2000).
[12] R. da Silva and J.R.D. de Felício, Rev. Bras. Ens. Fis. **24**, 103 (2002).
[13] E. Arashiro, J.R.D. de Felício and U.H.E. Hansmann, Physical Review E **73**, 040902 (2006).
[14] F.M. Ruziska, E. Arashiro and T. Tomé, Physica A: Statistical Mechanics and its Applications **489**, 56 (2018).
[15] N.R.S. Ortega, C.F.S. Pinheiro, T. Tomé and J.R.D. de Felício, Physica A: Statistical Mechanics and its Applications **255**, 189 (1998).
[16] F. Keesen, A. Castro e Silva, E. Arashiro, C.F.S. Pinheiro, Ecological Modelling **344**, 38 (2017).
[17] A. Castro e Silva and A.T. Bernardes, Physica A: Statistical Mechanics and its Applications **352**, 535 (2005).
[18] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart and W.S. Stornetta, IEEE Transactions on Software Engineering **18**, 103 (1992).
[19] F. James, Reports on Progress in Physics **43**, 1145 (1980).
[20] R.E. Caflisch, Acta Numerica **7**, 1 (1998).
[21] H. Gould, J. Tobochnik and W. Christian *An introduction to computer simulation methods: applications to physical systems-3rd Edition* (Pearson Addison Wesley, 2007).
[22] R.Y. Rubinstein, *Simulation and the Monte Carlo Method* (Wiley, Hoboken, 1981).
[23] Y. Badis, *Computational Physics* (WORLD SCIENTIFIC, Singapore, 2017).
[24] D.P. Kroese, T. Taimre and Z.I. Botev, *Handbook of Monte Carlo Methods* (John Wiley & Sons, Hoboken, 2011).
[25] J.K.S. William and L. Dunn, *Exploring Monte Carlo Methods* (Elsevier, Cambridge, 2011).
[26] M. Lee, *C++ Programming for the Absolute Beginner, 2nd Edition* (Course Technology Press, Boston, 2009).
[27] https://ieeexplore.ieee.org/document/977250.
[28] https://www.academie-sciences.fr/pdf/dossiers/Broglie/Broglie_pdf/CR1923_p507.pdf.
[29] https://www.nature.com/articles/112540a0.
[30] V.V. Raman and P. Forman, Historical Studies in the Physical Sciences **1**, 291 (1969).
[31] A. Beiser, *Concepts of Modern Physics* (McGraw-Hill, Boston, 2002).
[32] I. Levine, *The Hydrogen Atom* (Pearson, Boston, 2014).
[33] D.J. Griffiths *Introduction to Quantum Mechanics* (Cambridge University Press, Cambridge, 2017).
[34] N. Zettili, *Quantum Mechanics: Concepts and Applications* (Wiley, Hoboken, 2009).
[35] J. Von Neumann, National Bureau of Standards Applied Mathematics **3**, 36 (1951).
[36] G.D.W.W. Bauer, in *University Physics with Modern Physics* (McGraw-Hill, New York, 2011).
[37] https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Quantum_Mechanics/05.5%3A_Particle_in_Boxes/Particle_in_a_1-Dimensional_box.
[38] https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Quantum_Mechanics/05.5%3A_Particle_in_Boxes/Particle_in_a_2-Dimensional_box.
[39] https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Quantum_Mechanics/05.5%3A_Particle_in_Boxes/Particle_in_a_3-Dimensional_box.
[40] L. Pauling, *Introduction to quantum mechanics: with applications to chemistry* (Dover Publications, New York, 1985).
[41] J. Nocedal and S. Wright *Numerical Optimization* (Springer Science & Business Media, Berlin, 2006).
[42] W. Press, *Numerical recipes in C : the art of scientific computing* (Cambridge University Press, Cambridge, 1992).
[43] https://www.boost.org/doc/libs/1_56_0/libs/math/doc/html/index.html.
[44] https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2018GC007529.
[45] https://github.com/pedrohflobo/hydrogen_orbitals_by_Monte_Carlo_Method.

## O Modelo Baseado em Agentes

Modelos são uma forma matemática, física ou conceitual de representar um fenômeno que é difícil de se observar diretamente. Eles são desenhados para explicar ou prever comportamentos objetos reais ou sistemas e são aplicados em um amplo intervalo de áreas científicas como física, ecologia e economia. Os modelos são aproximações de objetos e sistemas, e por isto devemos enfatizar que eles possuem as suas limitações. E isto, muitas vezes implica em uma constante demanda por aprimoramento.

Na história dos modelos atômicos por exemplo, o modelo de Bohr deu um passo adiante em relação ao de Rutherford, porque ao introduzir a teoria quântica, explicou com sucesso, como cargas negativas podem permanecer orbitando ao redor de cargas positivas. Porém, ele não é tão preciso quanto ao de Schrödinger, que além de incorporar os todos avanços de modelos atômicos anteriores, este modelo considera o conceito de dualidade partícula-onda. Por isto, ao analisarmos um modelo, devemos nos ater a sua proposta e suas limitações.

A proposta de modelos variam. O modelo planetário Kepleriano foi construído a partir de dados experimentais coletados por Brahe, e pelo menos para Kepler o objetivo era validar a hipótese do heliocentrismo. Já os modelos preditivos como os climáticos por exemplo, são baseados em dados de fenômenos passados, e são mesclados com análises matemáticas para que sirvam para prever ocorrências futuras. No entanto, por causa de sua natureza estocástica ou por simplificações convenientes, os modelos deste tipo não se comprometem a descrever o futuro ou mesmo os fenômenos com exatidão. Outro tipo de modelo e que será mostrado adiante é classificado como modelo baseado em agentes (MBAs), e que é usado para simular interações entre agentes autônomos, para estudar seus efeitos no sistema quando visto na macroescala. Os MBAs são construídos através de simples regras e a partir delas sugere a explicação de fenômenos emergentes.

Uma das aplicações típicas de MBAs é no estudo de formação de nuvens de pássaro e cardumes, que possivelmente se formam como uma estrutura de defesa contra predadores. Apesar de percebermos a sincronia do grupo como um todo, sugerindo que um indivíduo se comporta como líder, provavelmente são as interações locais entre pássaros ou peixes

que produz o efeito coletivo. Motivado por esta hipótese, Couzin [9] desenvolveu um MBAs propondo três regras simples, com parâmetros ajustáveis e que se aplicam somente aos agentes do mesmo grupo. Neste modelo, Couzin mostrou como reproduzir fielmente o comportamento de bandos e cardumes na ausência de predadores. Porém, posteriormente ao adicionar uma quarta regra, Couzin [10] simulou a dispersão dos agentes, tipicamente quando sob ataque de predadores. Neste caso, esta perturbação adicional se propaga em toda a rede, criando locais vazios ao redor do agressor, inclusive promovendo a separação momentânea do grupo.

No caso do trabalho de nossa autoria, a ser apresentado na sequência, também nos inspiramos na dinâmica de perseguição entre predadores e presas, para entender se seria possível emergir o comportamento de manada, a partir de indivíduos que quase não interagem entre si. Neste caso, negligenciamos efeitos de estruturas sociais de grupos de animais, assim como a personalidade ou qualquer outra particularidade dos indivíduos. Na prática, nos estabelecemos uma rede que representa o ambiente e sobre esta criamos campos potenciais que servem de base para locomoção de indivíduos. Ou seja, dependendo de algumas condições, através destes campos, os predadores tenderão a se aproximar das presas, enquanto estas tenderão a se afastar dos predadores.

Além deste modelo propor as simplificações já mencionadas, possivelmente a maior diferença desde MBAs com a realidade é o espaço métrico que utilizamos. No mundo real, quando não é necessário cálculos de agrimensura, as distâncias geralmente são obtidas de acordo com a métrica euclideana, enquanto utilizamos a Manhattan neste modelo. Isto é devido ao fato de ser muito mais fácil implementar desta forma, porque naturalmente o computador trabalha com variáveis discretizadas.

O MMC também está presente neste modelo, porque toda vez que um agente se movimenta, a sua posição de destino é determinada aleatoriamente. Além disso, como será visto adiante, realizamos cálculos de curvas que representam médias que usamos como formas de avaliar a formação de *clusters*. A medida em que o número de amostras aumentam, mais delineadas ficam estas curvas médias, por consequência da atuação do MMC.

Na sequência apresentaremos o trabalho intitulado “Formação de *clusters* em um

modelo de perseguição e fuga" e que foi publicado no evento 8º MCSul. Em seguida apresentaremos outros resultados deste trabalho.

## 4.1 Formação de clusters em um modelo de perseguição e fuga


# Formação de clusters em um modelo de perseguição e fuga

Lobo, P.H.F[1]
Arashiro, E.[2]
Lazo, M.J.[3]
Mendonça, S.M.[4]

**Resumo:** Predadores procurando por presas, assim como presas fugindo de seus predadores são capazes de exibir comportamentos coletivos. Inspirados por este tipo de efeito, foi desenvolvido um modelo estocástico baseado em agentes, com o objetivo de estudar a movimentação e a formação de *clusters* de seus indivíduos. Neste projeto, foi montada uma rede virtual discretizada bidimensional, com condições periódicas de contorno, onde se movimentam dois tipos de agentes, sendo que um deles se locomove como presa e é programado para se afastar do segundo tipo, que se comporta como predador. Nesse modelo, como o interesse principal foi estudar os padrões formados pela dinâmica de perseguição e fuga, as proporções destes agentes são mantidas fixas em cada simulação. As regras de movimentação são baseadas em uma caminhada aleatória assimétrica, que faz com que os dois tipos de agentes, com seus respectivos comportamentos, executem um movimento browniano quando muito afastados. Porém a dinâmica de perseguição começa a ficar mais intensa quando estes dois se aproximam. Para analisar as condições nas quais os *clusters* emergem, foram variadas as concentrações dos dois tipos de agentes e um parâmetro $\sigma$, que atua como um mediador, amplificando ou atenuando as "forças" de atração/repulsão entre os indivíduos. Foram realizadas simulações que revelaram padrões de movimentação de agentes inicialmente posicionados de forma aleatória na rede, assim como a contagem do número médio de *clusters* ao longo do tempo.

**Palavras-chave:** modelo baseado em agentes, dinâmica estocástica, *clusters* emergentes, perseguição.

[1]Bacharel em Física, Programa de Pós-Graduação em Física, Universidade Federal do Rio Grande, pedrohflobo@gmail.com

[2]Doutor em Física Aplicada à Medicina e Biologia, Programa de Pós-Graduação em Física, Universidade Federal do Rio Grande, earashiro@furg.br

[3]Doutor em Física, Programa de Pós-Graduação em Modelagem Computacional, Universidade Federal do Rio Grande, matheuslazo@furg.br

[4]Graduanda em Física Bacharelado com ênfase em Física Médica, Universidade Federal do Rio Grande, suziellim@gmail.com

# 1 Introdução

O modelo de Lotka-Volterra clássico (Lotka, 1926; Volterra, 1990 apud Arashiro et al., 2008), também conhecido como modelo presa-predador, foi o primeiro a descrever matematicamente como a população de presas e predadores variam com o passar do tempo. Ele consiste em duas equações diferenciais ordinárias acopladas para a densidade de presas e predadores, e trata-se de uma analogia às leis de ação das massas (Schnakenberg, 1977). No entanto, este modelo não leva em consideração, pelo menos de forma explicita, a estrutura espacial do ambiente onde as espécies coexistem (Arashiro, Rodrigues, de Oliveira, & Tomé, 2008). Portanto, quando é de interesse saber detalhes da movimentação e a disposição espacial das espécies, devemos abordar o tema de outra forma, e uma alternativa corriqueira, é através de modelos baseados em agentes (MBAs).

Os MBAs são uma classe de modelos computacionais utilizados para simular ações, comportamentos e interações entre indivíduos ou grupos, com o objetivo de explorar o impacto dos agentes no sistema quando visto como um todo (Clarke, 2014). Além disto, eles têm aplicações em áreas interdisciplinares que envolvem estudos de comportamento organizacional (Secchi, 2015), comportamento do consumidor (Garifullin, Borshchev, & Popkov, 2007), tráfego de veículos (Huynh, Cao, & Wickramasuriya, 2014), evacuação (Kirchner & Schadschneider, 2002) e epidemias (Perez & Dragicevic, 2009). Na biologia eles são empregados também para descrever distribuições espaciais de indivíduos presentes em um ecossistema, com a adição inclusive de uma dinâmica populacional (Satulovsky & Tomé, 1994; Keesen, e Silva, Arashiro, & Pinheiro, 2017; Ruziska, Arashiro, & Tomé, 2018; Argolo et al., 2016; Boccara, Roblin, & Roger, 1994).

Motivados também por aplicações na área da ecologia e inspirados por comportamentos de bando que observamos em grandes grupos de animais como os gnus, por exemplo, propomos um modelo baseado em agentes (MBA) com o objetivo principal de estudar efeitos que advém da dinâmica de perseguição e fuga que ocorrem entre predadores e presas.

No entanto, salientamos que neste trabalho embora tenhamos nos inspirado no modelo de Lotka-Volterra, não se trata de um modelo presa-predador, uma vez que o objetivo aqui é estudar efeitos de bandos, e este não considera morte e

nascimento de nenhum indivíduo de ambas espécies. Além disto, por uma questão de simplificação, ao contrário do modelo clássico, que é determinístico, abordamos o problema fundamentando em princípios estocásticos. Isto é oportuno para compensar eventuais variedades internas do sistema, como diferenças entre os indivíduos ou mesmo no ambiente, típicas de sistemas complexos. Isto garante uma evolução única de uma mesma rede para cada número pseudo-aleatório fornecido como entrada.

# 2 Fundamentação teórica

## 2.1 Modelos Baseados em Agentes

Seria conveniente neste momento definir "agentes" antes de aplicarmos ao nosso propósito, mas infelizmente, apesar do seu uso comum, não existe uma versão precisa e universalmente aceita. (Chen, 2012) faz algumas comparações entre diferentes definições encontradas na literatura, em seguida, argumenta que existem duas características de agentes que são acordados por pessoas de área afins: autonomia e habilidade social. Isto os tornam capazes de se comportar de forma independente, mas com flexibilidade para reagir ao ambiente e interagir com outros indivíduos.

Isto significa que na prática, podemos interpretar agentes como indivíduos autônomos inseridos em um ambiente, capazes de interagir com os demais e que executam um conjunto de regras para os quais foram projetados. Em ambiente de simulação computacional por exemplo, os agentes em seus respectivos ambientes podem representar veículos nas estradas, moradores em um prédio ou mesmo vírus em seus hospedeiros.

## 2.2 A rede

Para que possamos utilizar o computador como ferramenta de trabalho, devemos realizar algumas adaptações necessárias da realidade a ser simulada. A primeira delas é propormos uma discretização no espaço e tempo. Isto é realizado ao inserirmos os agentes em uma rede bidimensional, composta por uma quantidade finita de células iguais, por onde se locomovem "saltando" de uma célula a outra em intervalos iguais de tempo.

Existem várias maneiras de elaborar a rede, e para nosso propósito, utilizaremos

uma das mais simples e comuns. Ela possui formato retangular, construída com células quadradas e admite condições periódicas de contorno nas bordas. Isto significa que a superior se comunica com a inferior, assim como as laterais esquerda e direita se conectam. Ou seja, podemos representar esta rede como a superfície de um toroide.

Além disso, o deslocamento é baseado na geometria do táxi, isto é, o agente só pode migrar de uma célula central para uma das a células vizinhas ao norte, sul, leste ou oeste. Em outras palavras, este tipo configuração admite movimentação somente sobre a vizinhança de *Neumann* de intervalo $r = 1$.

Como nosso modelo opera com uma grande quantidade de agentes, é necessário inserirmos uma origem do sistema de coordenadas global para mapearmos a rede posteriormente. Sendo assim, iremos etiquetar as células por suas posições genericamente representadas pelo par $(i, j)$. A escolha da célula origem, assim como a sequência numérica que representam os eixos de coordenadas são arbitrários, desde que $i$ e $j$ sejam valores discretizados em intervalos regulares. Por conveniência trabalharemos com a representação matricial. Isto é: $i$ aumenta de cima para baixo, enquanto $j$ cresce da esquerda para a direita, assumindo valores inteiros $i = \{1, 2, ..., m\}$ e $j = \{1, 2, ..., n\}$.

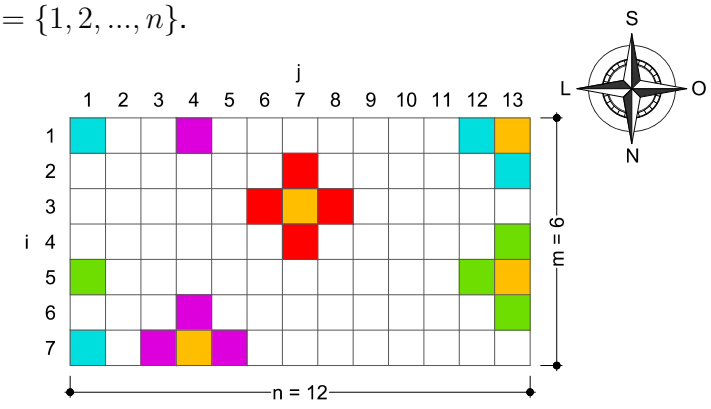



Figura 1: Algumas células destacadas em laranja com suas respectivas vizinhanças de *Neumann* em diversas cores. Esta rede considera condições periódicas de contorno nas bordas.

## 2.3 Os agentes

Embora todo organismo vivo possa ser único quando o examinamos em níveis de detalhamento mais elevados (Elsasser, 1981), simplificaremos ao considerarmos que todos os indivíduos de uma mesma classe são indistinguíveis. Este procedimento é frequentemente observado em MBAs que lidam com "agentes biológicos simulados". Deste modo, é denominado então como "*x*-agente" quando a intenção é fazer referência

aos membros das duas únicas classes presentes neste projeto. Entretanto, quando for conveniente fazer menção a presas e predadores de maneira específica, aqui denotados por $H$ e $P$ respectivamente, substituiremos estes em alguns índices representados genericamente pelo referido *'x'*.

Esta distinção é necessária porque os membros de cada uma dessas classes se comportam de maneiras diferentes ao se movimentarem. Isto porque, embora sejam semelhantes, eles compartilharem somente dois dos três objetivos explicitadas a seguir:

Classe predador ($P$):

i. Procurem se **aproximar das presas**.
ii. Não migrem para células ocupadas.
iii. Procurem se manter em movimento.

Classe presa ($H$):

i. Procurem se **afastar dos predadores**.
ii. Não migrem para células ocupadas.
iii. Procurem se manter em movimento.

## 2.4 O campo e a movimentação

O conjunto formado por cada classe de agentes produzem separadamente dois campos escalares, genericamente chamados de $C^x$ e é determinado através da equação (1):

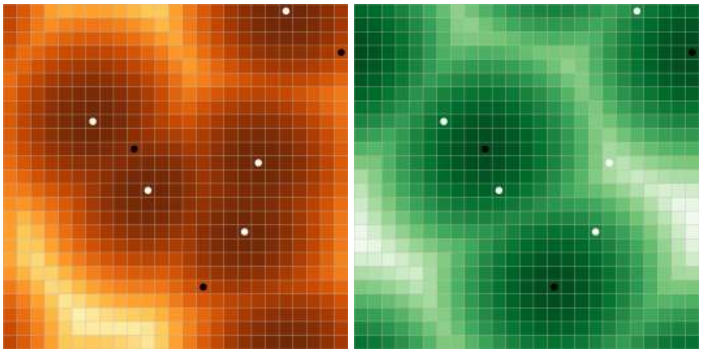
$$C^x_{i,j} = \sum_{k=1}^{K} \exp\left[ -\frac{\left( r_i^{x,k} + r_j^{x,k} \right)^2}{2\sigma^2} \right] \tag{1}$$

onde:

$$\begin{aligned} r_i^{x,k} &= \quad min\left( \left| i - i^{x,k} \right|, m - i + i^{x,k} \right) \\ r_j^{x,k} &= \quad min\left( \left| j - j^{x,k} \right|, n - j + j^{x,k} \right) \end{aligned} \tag{2}$$

A equação (1) nada mais é do que um somatório sobre gaussianas bidimensionais com desvio padrão $\sigma$, elaboradas na geometria do táxi. O numerador no argumento da exponencial corresponde ao quadrado da distância de *Manhattan*, calculado entre uma célula de coordenada $(i, j)$ e um *k-ésimo x*-agente situado em ($i^{x,k}$,$j^{x,k}$). Note que $r_i^{x,k}$ e $r_j^{x,k}$ explicitados em (2), tratam-se de distâncias ao longo dos eixos $i$ e $j$, adaptadas para rede $m \times n$ com periodicidade nas bordas. Além do mais, embora estes estados $C^x_{i,j}$ sejam capazes de mapear a rede inteira, é necessário calcular apenas os valores correspondentes a vizinhança imediata, que são as quatro posições para as quais os agentes efetivamente possuem uma probabilidade de se deslocarem. Por isso é conveniente isolar tais estados de células em uma tupla

$T^x = \{C^x_{i-1,j}, C^x_{i,j+1}, C^x_{i+1,j}, C^x_{i,j-1}\}$.



(a) (b)

Figura 2: Campo $C^H$ gerado por $5$ presas (pontos brancos) em (a) e $C^P$ produzidos por $3$ predadores (pontos pretos) em (b). Em ambos, os casos eles foram obtidos através da equação (1), sendo que os locais mais escuros e mais claros representam os maiores e menores valores dos campos respectivamente.

Este campo $C^x$ produzido é proposto desta forma para mapear gradientes das "zonas de ameaça e de predação " associadas aos agentes, que são na prática, difíceis de determinar e são responsáveis por orientá-los durante a dinâmica de perseguição. Por isto, o termo $\sigma$ é flexível, para fazermos ajustes em cada caso. Já o somatório é inserido porque os animais agrupados podem ser mais facilmente percebidos pelos seus "inimigos", tornando o campo mais intenso em locais mais concentrados.

Para promover o deslocamento dos agentes na rede, procuramos criar regras baseadas no primeiro objetivo das duas classes. Isto é, do ponto de vista dos predadores, que procuram perseguir as presas, calculamos a probabilidade $P^P_n$ de um agente migrar para cada uma das *n*-células da sua vizinhança ortogonal, tomando uma relação de proporção direta a $T^H_n$. No caso das presas, como elas tendem a se afastar dos predadores, estabelecemos a probabilidade $P^H_n$ por uma relação de proporção inversa a $T^P_n$.

É necessário considerar também, que dois ou mais indivíduos, devido ao segundo objetivo, não podem coexistir no mesmo espaço ao mesmo tempo. Além disso, como todos eles são atualizados simultaneamente, se uma célula fica vaga, nenhum outro poderá ocupá-la a menos que demais agentes, quando possível, já tenham se movimentado. Em outras palavras, no mesmo quadro, aqui considerado como atualização completa da rede, não é permitida a permanência em regiões já visitadas. Por isto, vamos considerar um estado $u$, no qual é $u_{i,j} = 1$, quando se a célula em

questão está vazia e $u_{i,j} = 0$ se está ocupada. Da mesma maneira, consideremos um estado $w$, no qual $w_{i,j} = 0$ se algum agente esteve nela no atual quadro, e $w = 1$ caso contrário. Com isto determinamos os conjuntos $S = \{u_{i-1,j}, u_{i,j+1}, u_{i+1,j}, u_{i,j-1}\}$ e $R = \{w_{i-1,j}, w_{i,j+1}, w_{i+1,j}, w_{i,j-1}\}$.

Por fim, através da combinação da equação 1 com os termos $T^x$, $S$ e $R$, determinamos a probabilidade de um agente migrar para cada uma das quatro células de sua vizinhança ortogonal, obtida através das relações 3 e 4 apresentadas em sequência. Explicitamente, estas probabilidades são dadas pelo conjunto $P^x = \{p^x_{i-1,j}, p^x_{i,j+1}, p^x_{i+1,j}, p^x_{i,j-1}\}$.

$$P_n^P = \begin{cases} S_n R_n T_n^H \left( \sum_{n=1}^{4} S_n R_n T_n^H \right)^{-1} & \text{, se } \sum_{n=1}^{4} S_n R_n \neq 0 \\ 0 & \text{, caso contrário} \end{cases} \tag{3}$$

$$P_n^H = \begin{cases} \frac{S_n R_n}{T_n^P} \left( \sum_{n=1}^{4} \frac{S_n R_n}{T_n^P} \right)^{-1} & \text{, se } \sum_{n=1}^{4} S_n R_n \neq 0 \\ 0 & \text{, caso contrário} \end{cases} \tag{4}$$

Note que a probabilidade calculada através das equações anteriores, determina se os agentes poderão se mover. Desta forma, executamos a regra baseada no terceiro e último objetivo comum às duas classes. Sendo assim, é conveniente agora explicar como todas estas equações trabalham juntas na etapa a seguir:

## 2.5 O algoritmo

Inicialmente, é utilizado como entrada o valor fixado de $\sigma$, as dimensões $m$ e $n$ da rede, assim como a quantidade de presas $H$ e predadores $P$. Em seguida os agentes são inseridos em posições aleatórias, sem que mais de um ocupe a mesma célula, com suas devidas posições temporárias coincidindo com as localizações de seus agentes.

Na sequência, inicia-se um *loop*, que consiste em selecionar uma das duas classes, depois um dos seus membros e atualizar sua posição na rede. Ou seja, quando um agente é escolhido, é calculada a probabilidade $P^x$ dele se deslocar para sua vizinhança. E caso ele se mova, é memorizado de onde ele veio, como uma posição temporária, para que nenhum outro agente a ocupe no mesmo quadro.

Quando todos os indivíduos forem verificados, estas posições temporárias serão

descartadas, e será contabilizado o número de *clusters*, repetindo o *loop anterior* até que seja alcançada uma quantidade desejável de quadros. No final, temos para cada quadro o número de *clusters*, armazenado usualmente em forma de arquivos com extensão ".dat" ou em um banco de dados (*MongoDB*, *MariaDB*, *MySQL*, *etc*).

Desta maneira, podemos utilizar os procedimentos anteriores para produzirmos resultados suficientes para realizar um tratamento estatístico, e completar o ciclo através do último *loop*. Isto consiste em determinamos o número médio de *clusters* em cada quadro, acompanhados das medidas de dispersão, os quais nos limitaremos aos desvio padrão neste trabalho. Ou seja, a saída desta etapa é um arquivo com o número médio de *clusters* para cada quadro. A figura 3 na sequência, ilustra todo o algorítimo com mais detalhes.

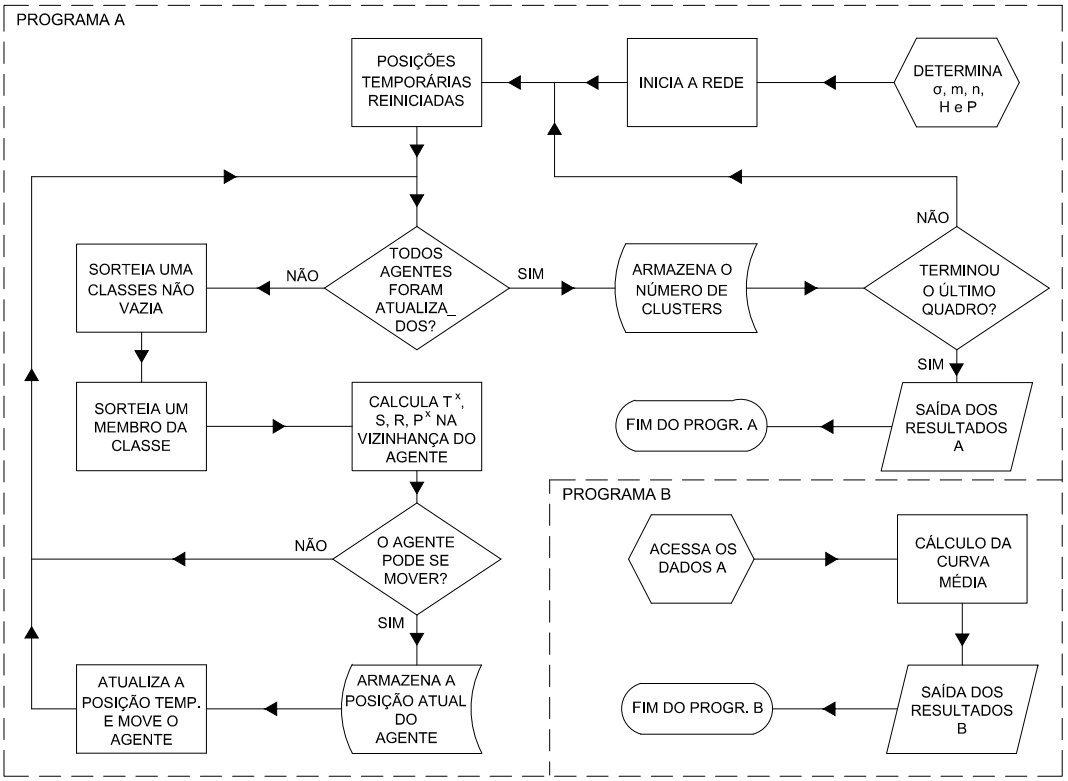



Figura 3: Algoritmo em forma de diagrama.

## 3 Simulações

Para realizar as simulações, em um primeiro ensaio, as concentrações dos dois tipos de agentes são mantidos constantes, enquanto o valor de $\sigma$ é variado. O objetivo nesta etapa é investigar se este termo é capaz de influir na "atração/repulsão" produzida entre

predadores e presas.

Posteriormente, em um segundo teste, é mantido fixo o valor de $\sigma$, assim como o número de presas e são realizadas algumas concentrações diferentes de predadores. Com isso, espera-se verificar sob quais condições é possível formar grupos ou rebanhos de predadores ou presas.

Por fim, verificamos a evolução do número médio de *clusters* $z$ formados por presas, influenciado pelos termos $\sigma$ e $H$. Nas simulações, são preservadas em cada ensaio as condições iniciais como dimensões da rede, estabelecidas inicialmente por $H$, $P$ e $\sigma$. Para tal verificação, realizamos um grande número de ensaios, nos quais o gerador de números aleatórios garante uma evolução distinta do MBA para cada simulação.

Para realizar esta contagem, consideramos um *cluster* como o número de agentes de mesma classe, conectadas através de suas vizinhanças, mesmo quando isolados, aqui considerados como *clusters* unitários.

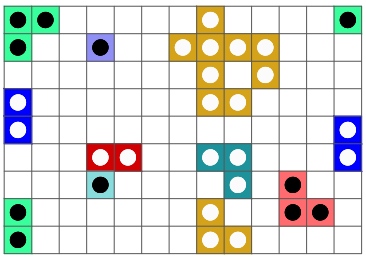


Figura 4: *Clusters* de diversos tamanhos formados por predadores e presas, representados por círculos pretos e brancos respectivamente. Nesta imagem, as cores diferenciadas nas células representam seus *clusters*.

## 4 Resultados

Inicialmente, fizemos alguns testes rápidos, que nos revelou na prática, como os membros das duas classes se comportam. Como previsto, os predadores perseguiram as presas e houve formação de *clusters* em algumas casos (ver figura 5).

Em seguida, para iniciar a investigação das condições necessárias para formar grupos, adotamos um valor fixo para $\sigma$ e arbitramos $[H]$ relativamente alto, quando comparado com $[P]$. Neste caso, foi observado que as presas passaram a formar rebanhos, enquanto os predadores tenderam a ficar isolados (figura 6a). Em seguida, quando invertemos estas quantidades, verificamos que as poucas presas foram cercadas por vários predadores (figura 6b).

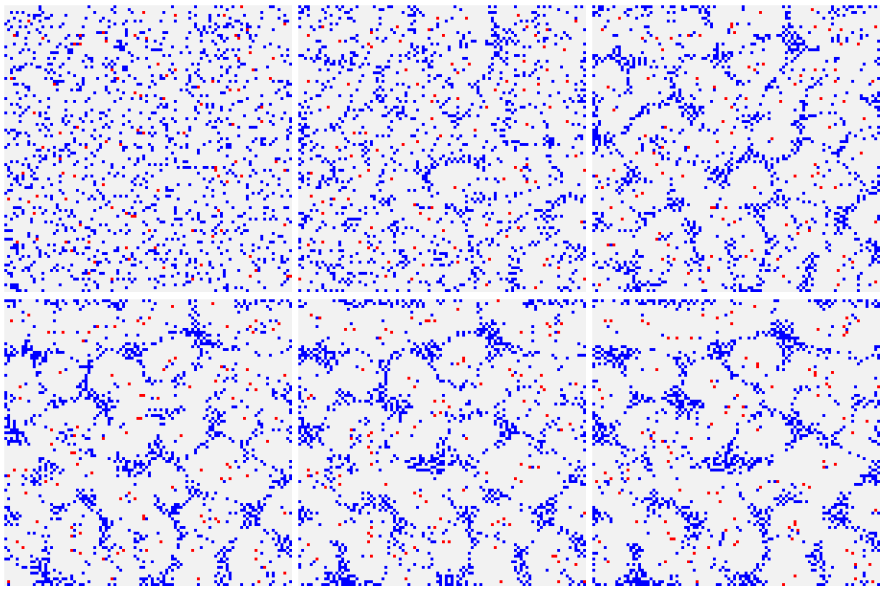
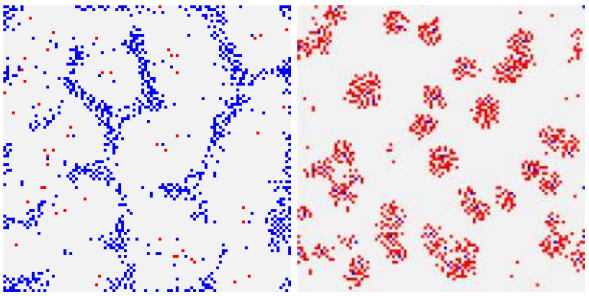


Figura 5: Evolução temporal a partir do início da simulação, para uma rede $100 \times 100$, a cada 4 quadros com $\sigma = 1.5$. O espaço foi populado com $[H] = 12.0\%$ de presas em azul e $[P] = 1.0\%$ de predadores em vermelho.

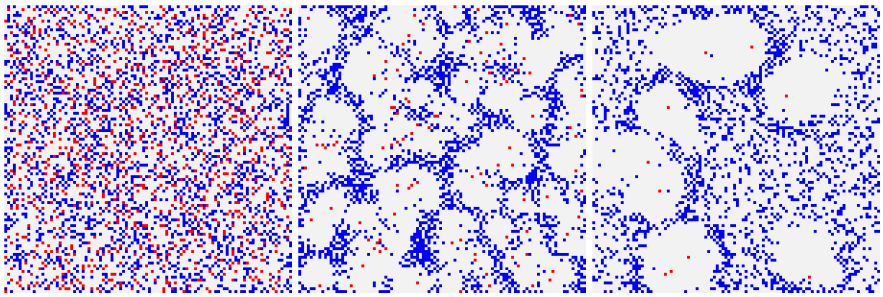


(a) (b)

Figura 6: Redes de dimensões $100 \times 100$, utilizando $\sigma = 1.5$ e evoluída para $30$ quadros. Em (a) ela é composta por $[P] = 0,5\%$ de predadores em vermelho e $[H] = 10,0\%$ de presas em azul. Já em (b) a rede é populada por $[P] = 10,0\%$ de predadores e $[H] = 0,5\%$ de presas.

Ao variarmos a concentração $[P]$, mantendo $\sigma$ e $[H]$ constante, verificamos que a diferença entre a quantidade de membros das duas classes favoreceu a formação de *clusters* (figura 7). Na presença de muitos predadores (figura 7a), é possível que o excesso deles tenha "pressionado" as presas por todos os lados, diminuindo os espaços para movimentação e consequentemente, dificultando a formação de grandes *clusters*. Por outro lado, se há poucos predadores, (figura 7c), a ação do campo produzido por eles pouco influencia nas presas mais distantes, fazendo com que elas passem a executar uma caminhada aleatória tradicional nestes locais. Ou seja, a tendência é criar rebanhos

somente nas proximidades dos predadores.



(a) $[P] = 10.0\%$ (b) $[P] = 1.0\%$ (c) $[P] = 0.1\%$

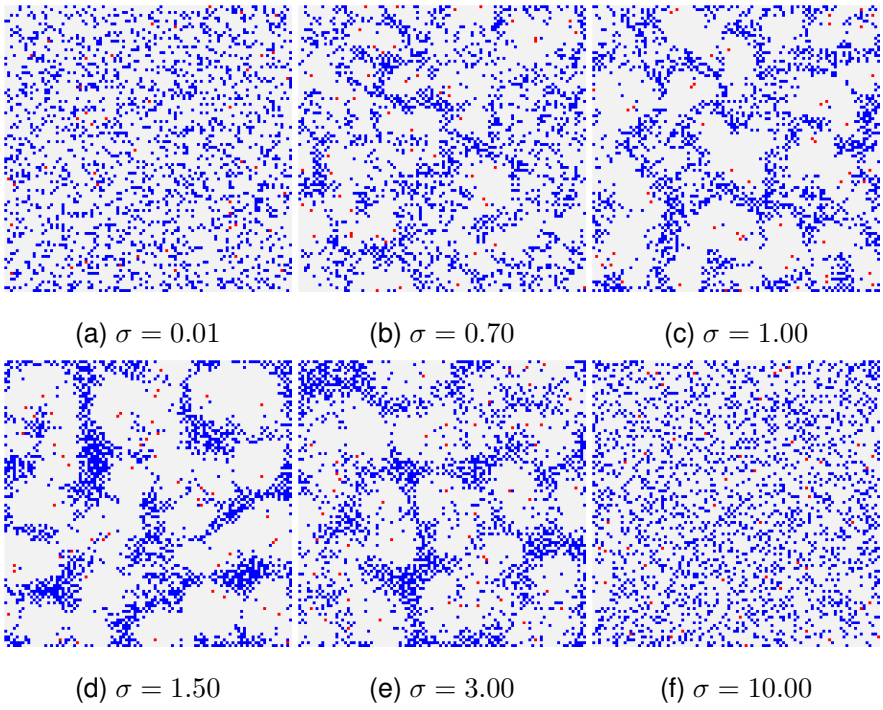
Figura 7: Diferentes simulações de concentrações de predadores $[P]$ em vermelho, para uma rede $100 \times 100$, evoluída a $30$ quadros. A concentração de presas inseridas é $[H] = 20.0\%$ em azul, para $\sigma = 1.5$.

Quando as concentrações $[H]$ e $[P]$ são mantidas fixas, porém é variado o valor de $\sigma$, observamos que nos extremos de um intervalo de $\sigma = (0, 10]$, as presas passam a não se agrupar (ver figuras 8a e 8f). Entretanto, para valores intermediários, notamos a formação de grupos que parecem mais compactos quando $\sigma = 1.5$ (ver figura 8d). Já para $\sigma \approx 0$, os agentes produzem campos em forma de picos. Ou seja, eles são intensos nas suas origens e praticamente nulos em suas vizinhanças. Porém, quando $\sigma$ é suficientemente grande, aqui representado por $\sigma = 10.0$, também não se unem porque os agentes produzem campos homogêneos. Isto implica na probabilidade equilibrada deles se movimentarem para qualquer sentido, executando movimentos brownianos.

Na sequência, para quantificarmos os números de *clusters* formados com o passar do tempo, variando $[H]$ e $\sigma$, realizamos algumas simulações, cujo os resultados serão apresentados nas próximas imagens.

De acordo com as figuras 9 e 10, entendemos que após uma pequena instabilidade, os números médios de *clusters* ($< z >$) diminuíram. Como o número de agentes é sempre constante, então isto significa que houve no início uma grande quantidade de pequenos *clusters* que se fundiram. O mesmo efeito é observado na maioria dos casos, quando a concentração de $H$ é a mesma para vários ensaios com diferentes valores de $\sigma$ (ver figura 11).

Nesta circunstância, escolha de $\sigma = 1.5$ entre todos os simulados, foi a que aparentemente mais favoreceu a formação de rebanhos. Porém, para $\sigma = 10.0$, houve um aumento no número de *clusters* unitários, ou seja, neste caso, a rede não apresentou



(a) $\sigma = 0.01$ (b) $\sigma = 0.70$ (c) $\sigma = 1.00$

(d) $\sigma = 1.50$ (e) $\sigma = 3.00$ (f) $\sigma = 10.00$

Figura 8: Simulações evoluídas para $30$ quadros sobre uma rede de $100 \times 100$, com valores diversos de $\sigma$. O espaço foi populado com $[H] = 20.0\%$ de presas em azul, e $[P] = 1.0\%$ de predadores em vermelho.

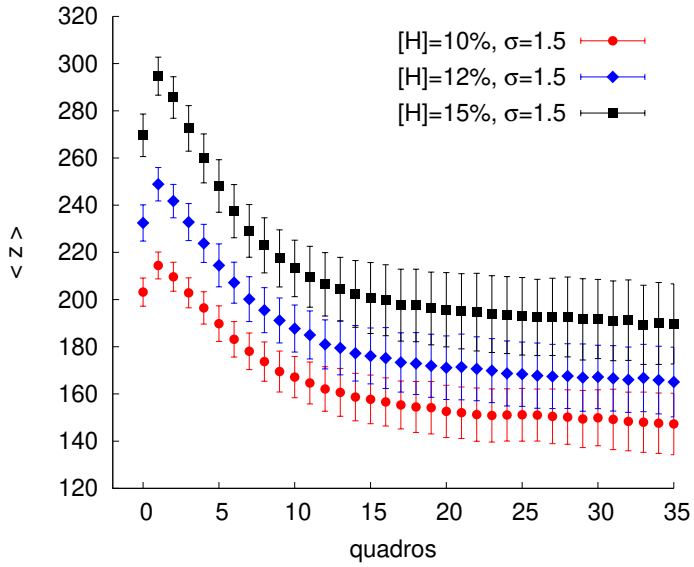



Figura 9: Número médio de *clusters* formados ao longo do tempo para uma rede de dimensões $50 \times 50$ com a utilização de um valor fixo de $\sigma = 1.5$. Para cada quadro as médias foram obtidas realizando-se $500$ ensaios e a concentração de predadores foi mantida a $[P] = 1\%$ em relação ao tamanho da rede.

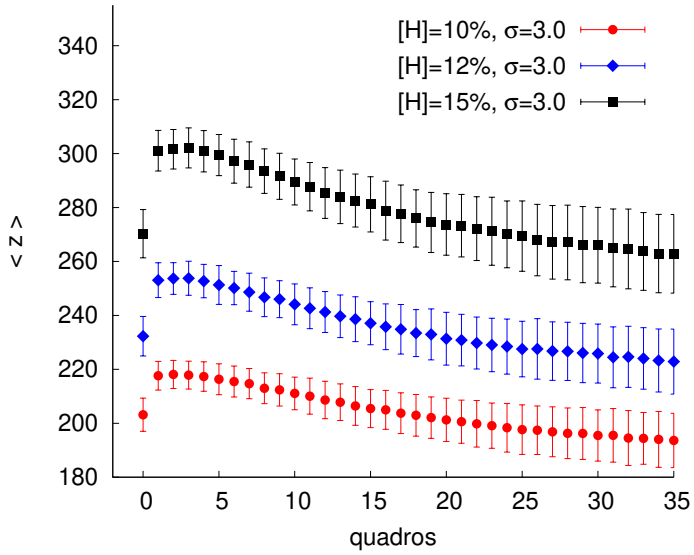



Figura 10: Número médio de *clusters* formados ao longo do tempo para uma rede de dimensões $50 \times 50$ com a utilização de um valor fixo de $\sigma = 3.0$. Para cada quadro, as médias foram obtidas realizando-se $500$ ensaios e a concentração de predadores foi mantida a $[P] = 1\%$ em relação ao tamanho da rede.

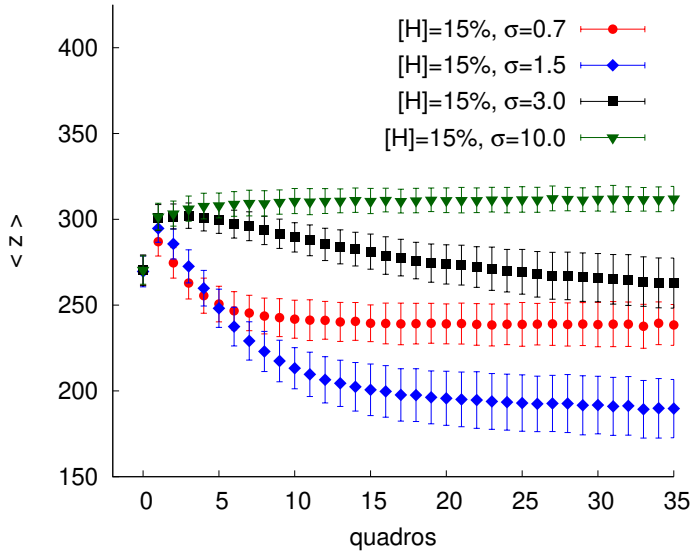



Figura 11: Comparação entre números médios de *clusters* formados ao longo do tempo para diferentes valores de $\sigma$ adotados. Nestas simulações, para cada quadro as médias foram obtidas realizando-se $500$ ensaios. A concentração de predadores e presas foram mantidas constantes com $[P] = 1\%$ e $[H] = 15\%$ em relação a rede de tamanho $50 \times 50$.

tendência a formar grandes grupos de presas. Este resultado coincide com o observado na figura 8f.

## 5 Conclusão e considerações finais

As simulações revelaram que este MBA é capaz de reproduzir a formação de bandos de animais, frequentemente observados na natureza, utilizando regras simples focadas na interação entre presas e predadores. Neste contexto, foi possível observar as condições que determinam o fenômeno de emergência de *clusters* de tamanhos variados, em ambiente de simulação computacional. Ou seja, isto é realizado ao controlarmos as concentrações de presas e predadores, assim como o desvio padrão $\sigma$, que atua como mediador de intensidade com que cada um dos membros destas classes interagem com seus "inimigos".

Um efeito interessante verificado, é que os agentes executam o tempo inteiro uma caminhada aleatória assimétrica, especialmente quando membros de classes opostas estão próximos, mas a medida em que se afastam, eles passam a exibir um típico movimento browniano. Isto é conveniente porque, é razoável propor que a dinâmica de perseguição e fuga se intensifica a medida que estes "inimigos" se aproximam.

## Agradecimentos

Pedro Henrique F. Lobo agradece a CAPES pelo suporte financeiro para realização deste projeto e a Mariana Gauterio Tavares pela revisão.

## Referências

Arashiro, E., Rodrigues, a. L., de Oliveira, M. J., & Tomé, T. (2008). Time correlation function in systems with two coexisting biological species. *Physical Review E*, *77*(6), 061909. doi: 10.1103/PhysRevE.77.061909

Argolo, C., Barros, P., Tomé, T., Arashiro, E., Gleria, I., & Lyra, M. L. (2016). Threshold of coexistence and critical behavior of a predator-prey stochastic model in a fractal landscape. *Journal of Statistical Mechanics: Theory and Experiment*, *2016*(8), 083204.

Boccara, N., Roblin, O., & Roger, M. (1994, Dec). Automata network predator-prey model with pursuit and evasion. *Phys. Rev. E*, *50*, 4531–4541. doi: 10.1103/PhysRevE.50.4531

Chen, L. (2012, jun). Agent-based modeling in urban and architectural research: A brief literature review. *Frontiers of Architectural Research*, *1*(2), 166–177. doi: 10.1016/j.foar.2012.03.003

Clarke, K. C. (2014). Handbook of Regional Science. In M. M. Fischer & P. Nijkamp (Eds.), *Journal of regional science* (Vol. 54, pp. 1217–1231). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-23430-9

Elsasser, W. M. (1981). Principles of a new biological theory: A summary. *Journal of Theoretical Biology*, *89*(1), 131 - 150. doi: https://doi.org/10.1016/0022-5193(81)90182-X

Garifullin, M., Borshchev, A., & Popkov, T. (2007). Using Anylogic and Agent-Based Approach To Model Consumer Market. *EUROSIM 2007, Ljubljana, Slovenia.*, 1–5.

Hanson, J. (2012). A visit to taxicab geometry. *International Journal of Mathematical Education in Science and Technology*, *43*(8), 1109-1123. doi: 10.1080/0020739X.2012.662291

Huynh, N., Cao, V., & Wickramasuriya, R. (2014). An Agent Based Model for the Simulation of Road Traffic and Transport Demand in A Sydney Metropolitan Area. *8th International Workshop on Agents in Traffic and Transportation (ATT-2014)*, 1–7. doi: 10.13140/2.1.4023.2961

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998, Mar 01). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, *1*(1), 7–38. doi: 10.1023/A:1010090405266

Keesen, F., e Silva, A. C., Arashiro, E., & Pinheiro, C. (2017). Simulations of populations of sapajus robustus in a fragmented landscape. *Ecological Modelling*, *344*, 38 - 47. doi: https://doi.org/10.1016/j.ecolmodel.2016.11.003

Kirchner, A., & Schadschneider, A. (2002). Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications*, *312*(1-2), 260–276. doi: 10.1016/S0378-4371(02)00857-9

Lotka, A. J. (1926). Elements of physical biology. *Science Progress in the Twentieth Century (1919-1933)*, *21*(82), 341–343.

Perez, L., & Dragicevic, S. (2009). An agent-based approach for modeling dynamics of contagious disease spread. *International Journal of Health Geographics*, *8*(1), 1–17. doi: 10.1186/1476-072X-8-50

Ruziska, F. M., Arashiro, E., & Tomé, T. (2018). Stochastic dynamics for two biological species and ecological niches. *Physica A: Statistical Mechanics and its Applications*, *489*, 56 - 64. doi: https://doi.org/10.1016/j.physa.2017.07.016

Satulovsky, J. E., & Tomé, T. (1994, Jun). Stochastic lattice gas model for a predator-prey system. *Phys. Rev. E*, *49*, 5073–5079. doi: 10.1103/PhysRevE.49.5073

Schnakenberg, J. (1977). G. Nicolis und I. Prigogine: Self-Organization in Nonequilibrium Systems . From Dissipative Structures to Order through Fluctuations. J. Wiley & Sons, New York, London, Sydney, Toronto 1977. 491 Seiten, Preis: £ 20.-, $ 34.- . *Berichte der Bunsengesellschaft für physikalische Chemie*, *82*(6), 672–672. doi: 10.1002/bbpc.197800155

Secchi, D. (2015, mar). A case for agent-based models in organizational behavior and team research. *Team Performance Management: An International Journal*, *21*(1/2), 37–50. doi: 10.1108/tpm-12-2014-0063

Volterra, V. (1990). *Leçons sur la théorie mathématique de la lutte pour la vie*. Paris: J. Gabay.

## Resultados complementares

A partir de agora faremos uma reavaliação, mas de outra perspectiva, verificando a influência do termo $\sigma$ e concentrações de presas e predadores. Além disso, realizamos simulações com a intenção de observar o efeito em diferentes tamanhos de rede.

Sendo assim, tudo que foi apresentado anteriormente se aplica nesta parte a seguir. Porém, devemos adicionar o termo "perímetro" no contexto de MBAs, porque este será útil futuramente. Um *cluster* formado por uma das classes de agentes, possui uma borda delimitada por agentes mais externos. Nós consideramos como "perímetro" o número de células externas a esta borda, da sua a vizinhança de Neumann imediata. Logo, um *cluster* unitário possui perímetro de 4 unidades, assim um *cluster* de dois agentes possui perímetro de 6 unidades.

Inicialmente simulamos algumas redes quadradas de tamanhos variados, com a intenção de determinarmos o tempo (ou quadro) no qual cada dimensão atinge o estado estacionário. A melhor forma que encontramos de compararmos redes de diferentes dimensões foi calcular a razão entre o perímetro médio de *clusters* em relação ao número de células da rede, para situações que de antemão sabíamos que *clusters* se formariam.

Para exemplificarmos, uma rede $30 \times 30$ possui 900 células. Desta forma, especificamente para este tamanho de rede, além de fixamos o valor de $\sigma$ e mantermos as concentrações de presas e predadores contantes, em cada quadro dividimos a média de perímetros de *clusters* de presas por 900.

De acordo com o resultado observado na figura 1, nós consideramos que para redes de dimensões $50 \times 50$ e menores, as simulações atingiram o estado estacionário antes do quadro = 1000. Este resultado é importante, porque nos permite determinar um tamanho de rede ideal para se trabalhar a partir de agora. Redes pequenas demais não são convenientes de se utilizar, porque a inserção de poucos agentes implica em um drástico aumento de concentrações, enquanto redes grandes exigem muito esforço computacional e portanto, muito no tempo de simulação. Por isto, para as próximas simulações, adotaremos o redes de dimensões $50 \times 50$ por considerarmos como um tamanho

intermediário e adequado.

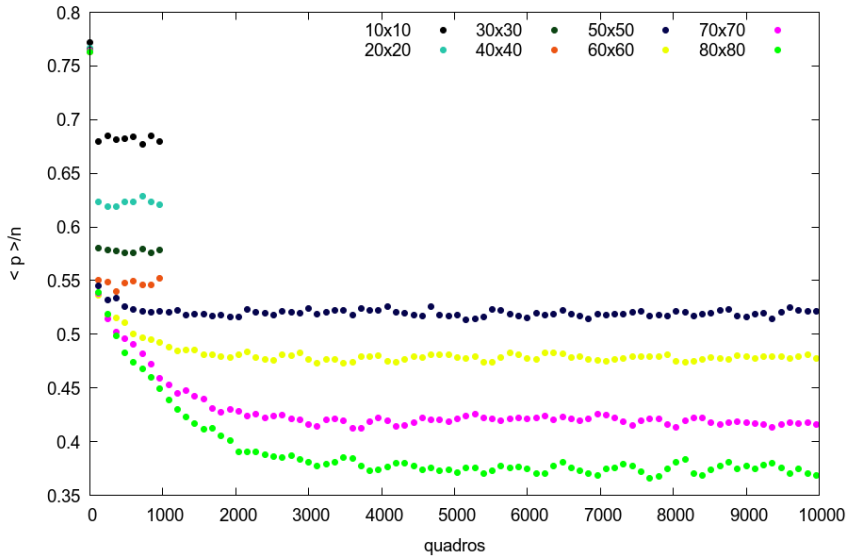


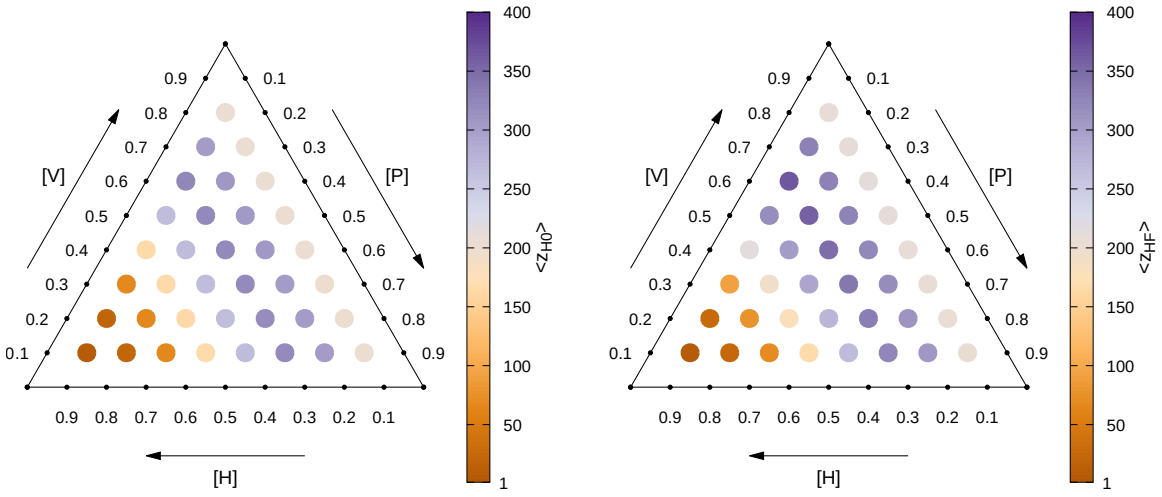

**Figura 1:** Evolução temporal para diferentes tamanhos de rede, da razão entre o perímetro médio de presas e seus respectivos número células. Nesta simulação foram utilizadas 200 amostras e para $\sigma = 3.00$, $[H] = 30\%$ e $[P] = 1\%$.

A partir de agora, iremos verificar a influência de várias combinações de quantidade de agentes na rede, por isto nós varremos o espaço procurando alterar as concentrações dos dois tipos de agentes e introduzimos uma nova concentração $[V]$ que chamamos de "concentração de vazios", fazendo referência aos espaços da rede desocupados. Para calcularmos o número médio de *clusters* de $V$, em cada quadro, após todos os agentes da classe $P$ e $H$ se movimentem na redes, nos contabilizamos todos os espaços desocupados para obter os *clusters*, assim como fizemos com as outras classes.

Para mostrarmos simultaneamente as variáveis $[H]$, $[P]$ e $[V]$, produzimos um triângulo de concentrações, o qual expressa valores de maneira idêntica ao triângulo de Maxwell. O procedimento é análogo ao que gera o espectro cromático visível, e nós apenas substituímos as intensidades que representam as cores primárias aditivas (*red*, *green*, *blue*), por estas concentrações de agentes.

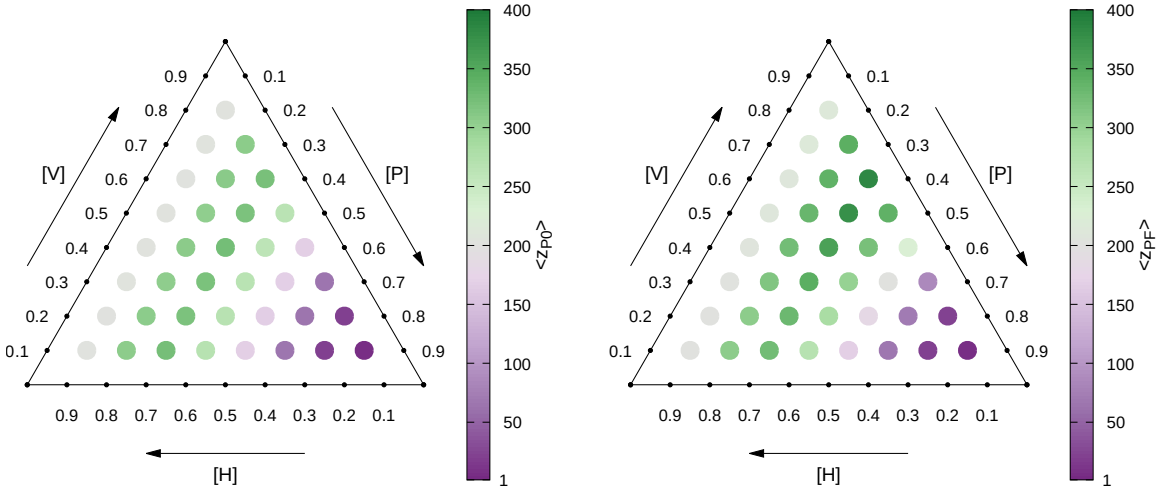
As figuras 2 e 3 a seguir expressam o número médio de *clusters*, para 36 concentrações distintas de ambos os tipos de agentes, no início da simulação e no seu estado estacionário. Estas imagens são adequadas para representar as situações quando a menor concentração de agentes é de pelo menos 10%, ao contrário dos casos anteriores simulados próximos a 1%. Lembramos também que um aumento no número de *clusters* representa uma diminuição

dos seus tamanhos médios, e portanto, nesta escala, os locais próximos a 400 são os mais populados com pequenos *clusters*, enquanto se forma um único grande rebanho em regiões representado por 1.




(a) (b)

**Figura 2:** Triângulos de concentrações para número médio de *clusters* de presas inicial (quadro = 0) em (a) e final (quadro = 1000) em (b). A rede utilizada possui dimensões $50 \times 50$ e adotamos $\sigma = 3.0$. Os termos $[H]$, $[P]$ e $[V]$ representam concentrações de presas, predadores e "vazios" respectivamente. Nesta simulação foram utilizadas 200 amostras.




(a) (b)

**Figura 3:** Triângulos de concentrações para número médio de *clusters* de predadores inicial (quadro = 0) em (a) e final (quadro = 1000) em (b). A rede utilizada possui dimensões $50 \times 50$ e adotamos $\sigma = 3.00$. Os termos $[H]$, $[P]$ e $[V]$ representam concentrações de presas, predadores e "vazios" respectivamente. Nesta simulação foram utilizadas 200 amostras.

Comparando o lados esquerdos e direitos das figuras 2 e 3, é difícil averiguar se houve formação de *clusters* com o passar do tempo, uma vez que estes expressam valores absolutos e as diferenças são muito sutis. Sendo assim, prosseguiremos calulando em cada

caso, a razão entre o número médio de *clusters* no estado estacionário, pelo número inicial médio de *clusters* ($< z_F > / < z_0 >$). Esta abordagem nos permite comparar as situações as quais formam rebanhos de presas e predadores, uma vez que a deposição de agentes na rede no tempo inicial é sempre uniforme. Os resultados estão presentes na figura 4 a seguir.

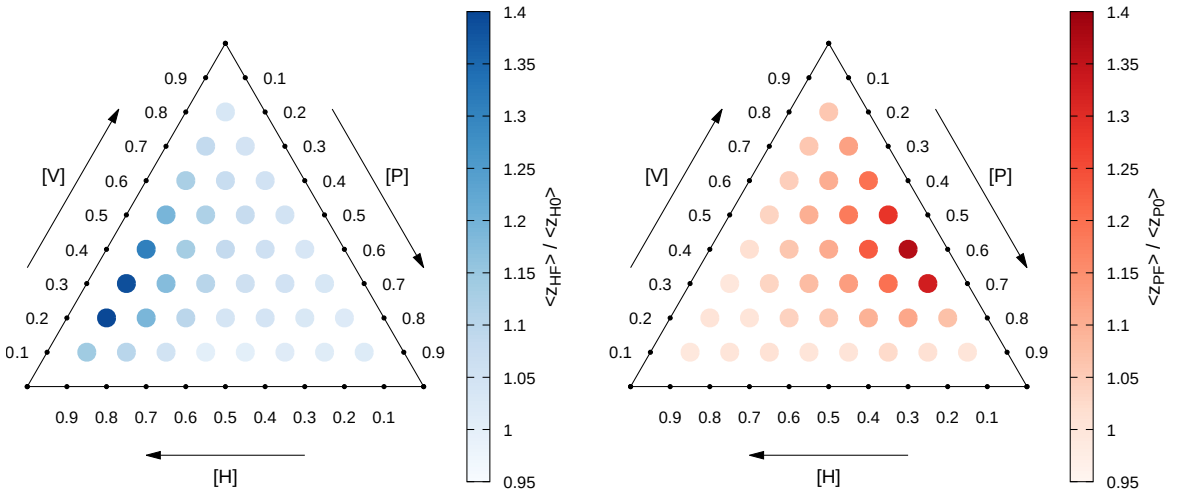
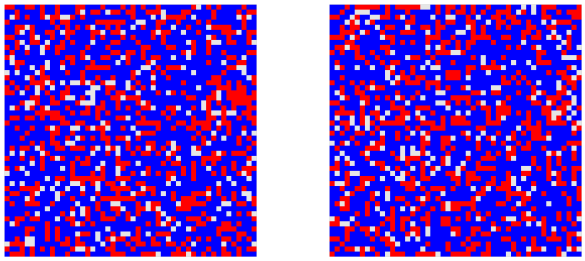



**Figura 4:** Triângulos de concentrações para a razão entre o número médio de *clusters* no estado estacionário (quadro = 1000), pelo número inicial médio (quadro = 0) de *clusters* ($< z_F > / < z_0 >$). Em (a) é referente a presas e em (b) predadores. A rede utilizada possui dimensões $50 \times 50$ e adotamos $\sigma = 3.0$. Os termos $[H]$, $[P]$ e $[V]$ representam concentrações de presas, predadores e "vazios" respectivamente. Nesta simulação foram utilizadas 200 amostras.

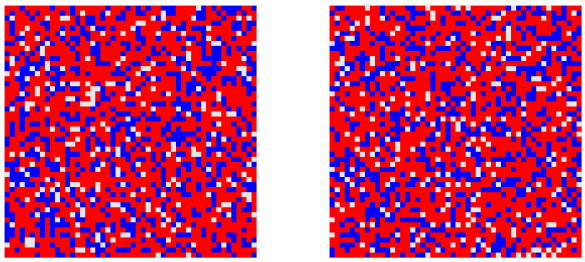
De acordo com as figuras 4a e 4b, os pontos mais claros revelaram que o número médio de *clusters* permaneceu aproximadamente constante, enquanto os pontos mais escuros indicaram um aumento mais expressivo. Em outras palavras, como $(< z_F > / < z_0 >) \geq 1$, em média, para presas e predadores, nenhuma destas 36 situações culminou no surgimento de grandes *clusters* por efeito da dinâmica de perseguição e fuga, inclusive favoreceu a dispersão de agentes em alguns casos (pontos escuros). Portanto, há outro motivo que justifica a formação de grandes rebanhos previstos nas figuras 2b e 3b.

Ao observarmos a figura 2, na situação em que $([H], [P], [V]) = (0.8, 0.1, 0.1)$, houve uma grande quantidade de presas inseridas na rede, e por isto formaram-se várias conexões entres elas, produzindo grandes *clusters*. Analogamente, o mesmo efeito ocorre com os predadores observados na figura 3 para $([H], [P], [V]) = (0.1, 0.8, 0.1)$. As figuras 5 e 6 a seguir ilustram estes cenários.



(a) (b)

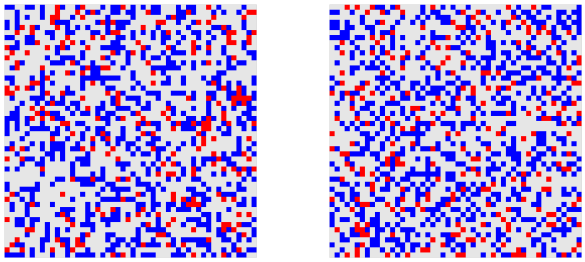
**Figura 5:** Rede $50 \times 50$ no quadro $= 0$ em (a) e no quadro $= 1000$ (estado estacionário) em (b), para concentrações $[H] = 60\%$ e $[P] = 30\%$, para $\sigma = 3.00$, utilizando uma única amostra. Os pontos em azuis e vermelhos representam presas e predadores respectivamente.



(a) (b)

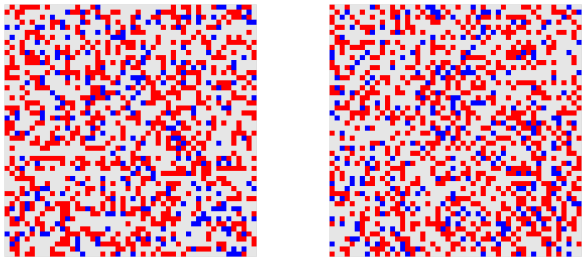
**Figura 6:** Rede $50 \times 50$ no quadro $= 0$ em (a) e no quadro $= 1000$ (estado estacionário) em (b), para concentrações $[H] = 30\%$ e $[P] = 60\%$, Para $\sigma = 3.00$, utilizando uma única amostra. Os pontos em azuis e vermelhos representam presas e predadores respectivamente.

Ao reduzirmos as concentrações $[H]$ e $[P]$ para valores mais moderados e assim facilitar a locomoção de agentes, observamos que tanto para presas e predadores, a rede permaneceu ocupada por vários *clusters* pequenos como visto nas figuras 7 e 8 a seguir.



(a) (b)

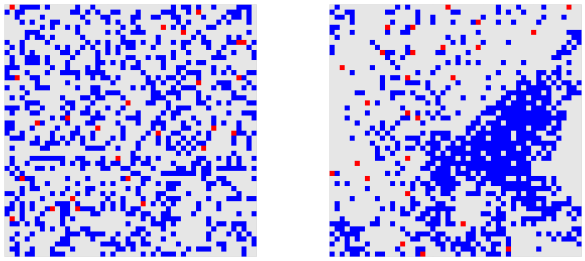
**Figura 7:** Rede $50 \times 50$ no quadro $= 0$ em (a) e no quadro $= 1000$ (estado estacionário) em (b), para concentrações $[H] = 30\%$ e $[P] = 10\%$, para $\sigma = 3.00$, utilizando uma única amostra. Os pontos em azuis e vermelhos representam presas e predadores respectivamente.



(a) (b)

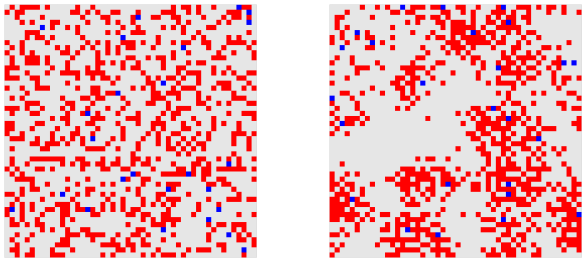
**Figura 8:** Rede $50 \times 50$ no quadro $= 0$ em (a) e no quadro $= 1000$ (estado estacionário) em (b), para concentrações $[H] = 10\%$ e $[P] = 30\%$, para $\sigma = 3.00$, utilizando uma única amostra. Os pontos em azuis e vermelhos representam presas e predadores respectivamente.

Ao diminuirmos a menor das concentrações de 10% para 1%, notamos que há tendência a formar *clusters* como visto nas figuras 9 e 10 na sequência. Isto sugere que entre estes dois há um valor que satura a rede dificultando a criação de grandes rebanhos.



(a) (b)

**Figura 9:** Rede $50 \times 50$ no quadro $= 0$ em (a) e no quadro $= 1000$ (estado estacionário) em (b), para concentrações $[H] = 30\%$ e $[P] = 1\%$, para $\sigma = 3.00$, utilizando uma única amostra. Os pontos em azuis e vermelhos representam presas e predadores respectivamente.



(a) (b)

**Figura 10:** Rede $50 \times 50$ no quadro $= 0$ em (a) e no quadro $= 1000$ (estado estacionário) em (b), para concentrações $[H] = 1\%$ e $[P] = 30\%$, para $\sigma = 3.00$, utilizando uma única amostra. Os pontos em azuis e vermelhos representam presas e predadores respectivamente.

Apesar das figuras 4a e 4b nos fornecer uma noção geral do efeito de várias

configurações de concentrações de agentes, elas podem dificultar a distinção de valores em algumas regiões. Por isto, faremos uma nova inspeção para alguns casos particulares. De acordo com a figura 11 a seguir, ao fixarmos $[H] = 30\%$ observamos que ao longo do tempo, a medida que aumentamos algumas concentrações de predadores, em média nota-se uma diminuição no número *clusters* de presas formados. O mesmo comportamento é esperado quando analisamos o número médio relativo *clusters* de predadores ao longo do tempo (ver figura 12).

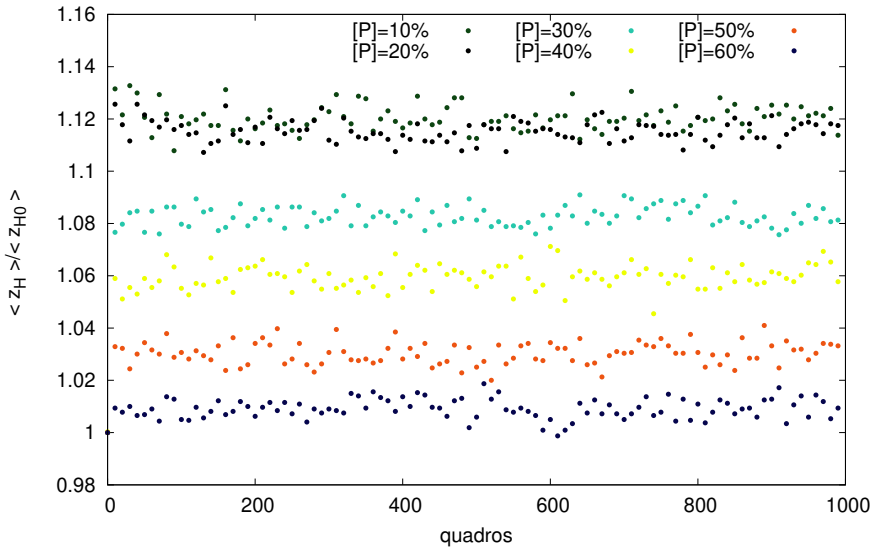



**Figura 11:** Número médio de *clusters* relativo de presas ao longo do tempo, para diferentes concentrações de predadores. Nesta simulação foram utilizadas 200 amostras, para $\sigma = 3.00$ e mantivemos a concentração de presas $[H] = 30\%$, para uma rede de tamanho $50 \times 50$.

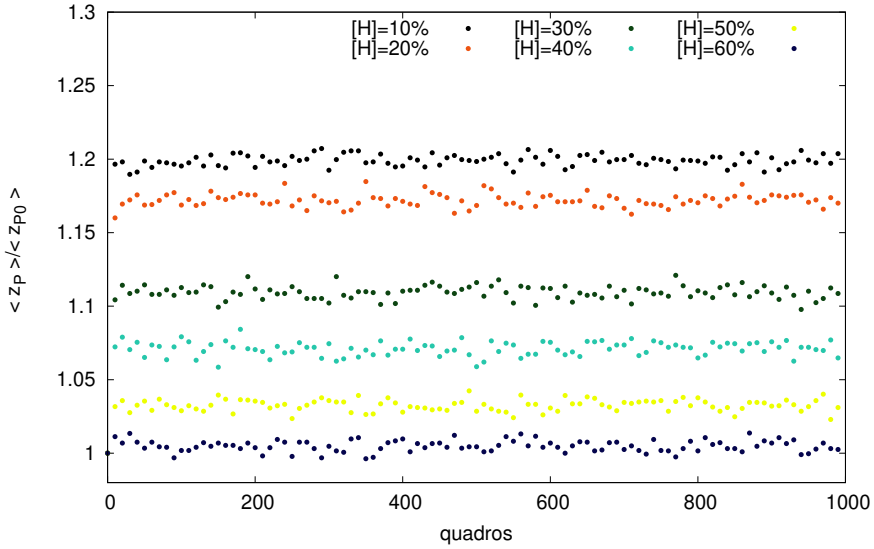



**Figura 12:** Número médio de *clusters* relativos de predadores ao longo do tempo, para diferentes concentrações de presas. Nesta simulação foram utilizadas 200 amostras, para $\sigma = 3.00$ e mantivemos a concentração de predadores $[P] = 30\%$, para uma rede de tamanho $50 \times 50$.

Para este mesmo conjunto de concentrações, ao considerarmos as médias de *clusters* relativo de presas no estado estacionário, observamos na figura 13 que em geral, o aumento na concentração de predadores implica na formação de menos clusters *clusters* de presas. O mesmo resultado é observado se fizermos a mesma análise do ponto de vista dos predadores (ver figura 14).

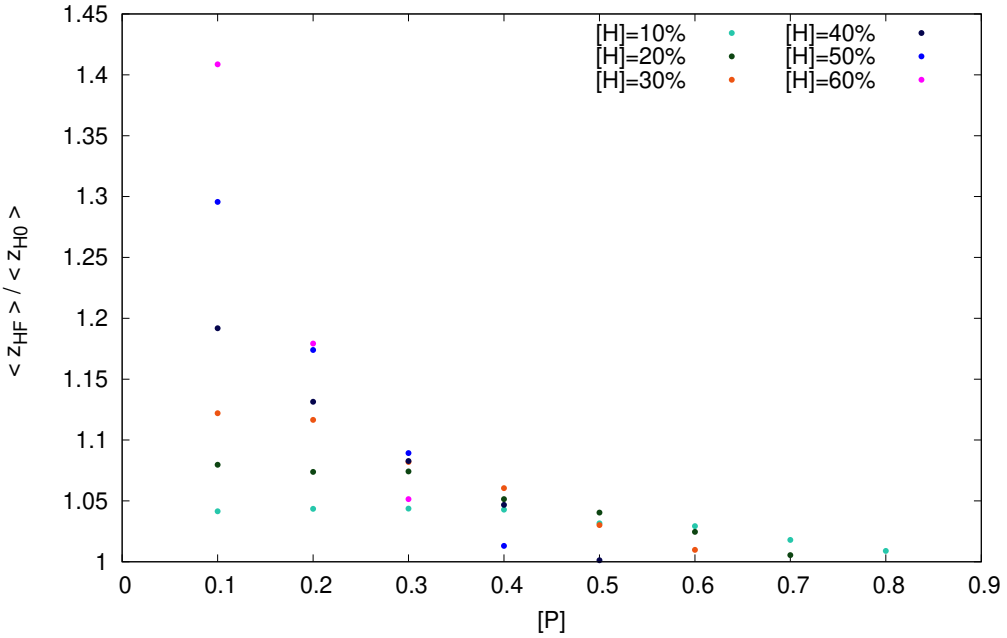



**Figura 13:** Número médio de *clusters* de presas em função de $[H]$ para uma rede $50 \times 50$ no estado estacionário (quadro = 1000). Nesta simulação foram utilizadas 200 amostras.

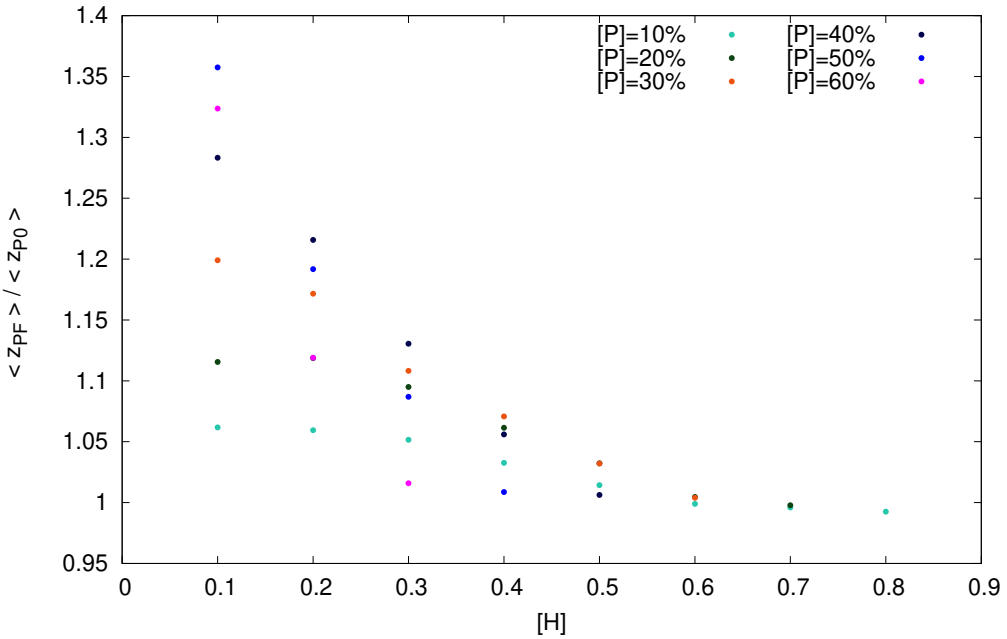



**Figura 14:** Número médio de *clusters* de predadores em função de $[P]$ para uma rede $50 \times 50$ no estado estacionário (quadro = 1000). Nesta simulação foram utilizadas 200 amostras.

A última análise que iremos apresentar é referente ao desvio padrão da gaussiana. Inicialmente calculamos médias de *clusters* de presas ao longo do tempo para vários valores de $\sigma$. Os resultado está presente na figura 15 a seguir e este mesmo estudo já foi apresentado no artigo, porém para uma evolução temporal curta. Ou seja, de antemão já sabemos o porquê de $\sigma \approx 0$ e $\sigma = 10$ não formar *clusters* grandes de presas.

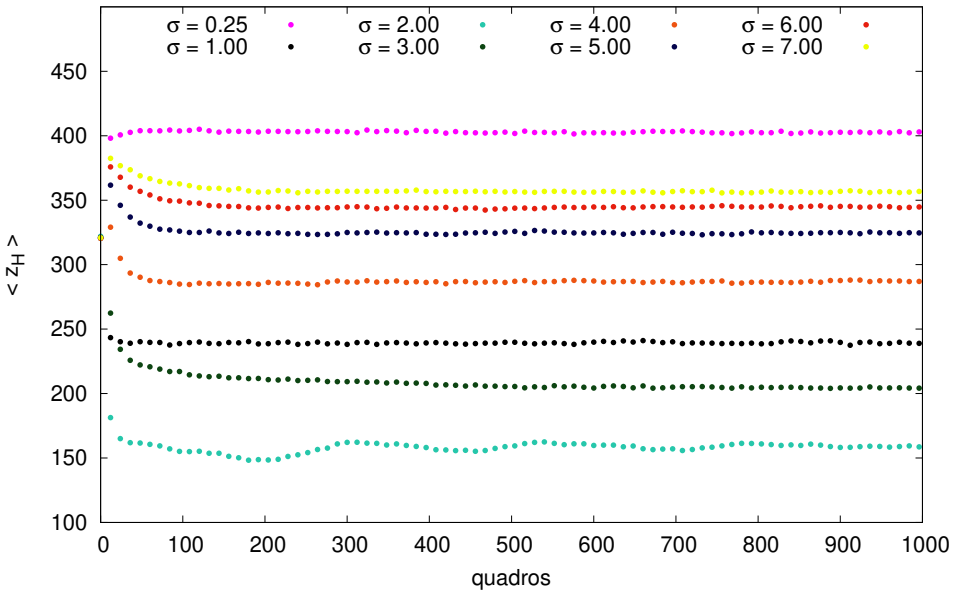



**Figura 15:** Número médio de *clusters* de presas ao longo do tempo para diferentes valores de $\sigma$, em uma rede $50 \times 50$, para $[P] = 1\%$ e $[H] = 30\%$. Nesta simulação foram utilizadas 200 amostras.

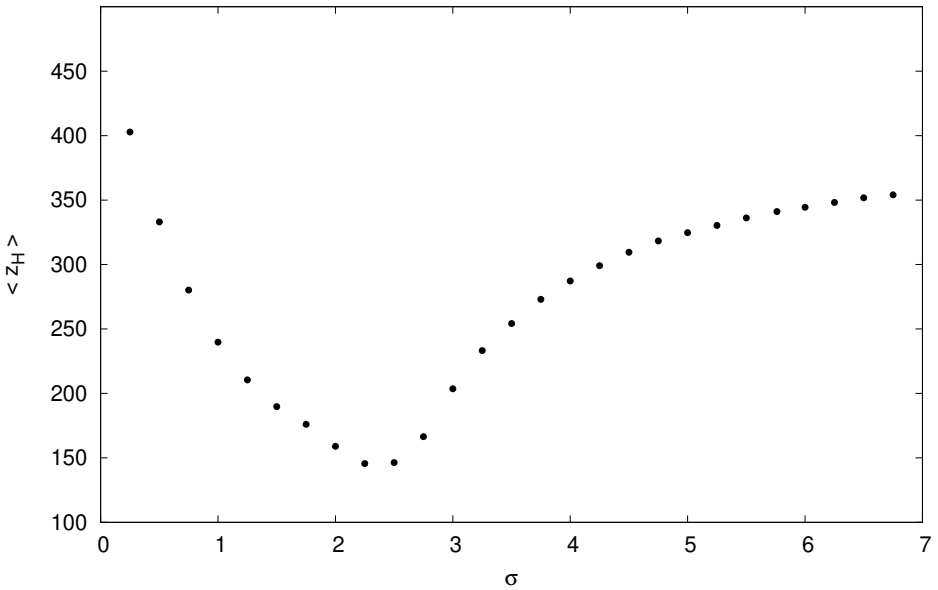



**Figura 16:** Número médio de *clusters* de presas em função do $\sigma$ para uma rede $50 \times 50$ no estado estacionário (quadro = 1000), para $[P] = 1\%$ e $[H] = 30\%$. Nesta simulação foram utilizadas 200 amostras.

Em algum ponto entre estes dois extremos, sabemos que há uma mudança de comportamento que leva a formação rebanhos de presas. Sendo assim, acrescentamos um novo gráfico (ver figura 16), que avalia no estado estacionário, a média de *clusters* de presa em função de $\sigma$ e concluímos que no intervalo $\sigma \in [2, 3]$, o número de *clusters* atinge o valor mínimo. Ou seja, é a situação que produz em média os maiores rebanhos.

## Conclusão

Quanto ao artigo envolvendo o modelo de Schrödinger, reforçamos que os pontos que esboçam o átomo de hidrogênio, representam uma área "abaixo" função densidade de probabilidade (versão independente do tempo) e que os locais mais concentrados são regiões com maior probabilidade de se encontrar o elétron. Sendo assim, o método de Neumann apenas converte uma distribuições uniformes nesta mesma função.

Em relação ao MBAs desenvolvido, mostramos que simples regras de interações são capazes de formar grupos de presas e predadores. Este trabalho revelou que tanto o desvio padrão da gaussiana $\sigma$, quanto as diferentes concentrações das duas classes de agentes, foram fatores decisivos para formação de grupos de diversos tamanhos. Inclusive enfatizamos que, apesar dos triângulos de concentrações médias com valores absolutos, indicarem a presença de grandes *clusters* em alguns casos, a "formação" destes ocorrem especialmente porque a rede estava saturada. Os grandes *clusters* que advém da dinâmica de perseguição entre presas e predadores emergem quando há uma concentração mínima de um tipo de agente, e a rede é "dopada" minimamente com (aproximadamente 1%) de agentes do segundo tipo.

Verificamos também que é possível construir MBAs como este, em que estados de células atuam como campo escalar, guiando agentes que transitam na rede. Outra constatação realizada, é que neste modelo, a não ser pelo fato de indivíduos não poderem ocupar simultaneamente a mesma célula, não há interação entre agentes iguais. Isto é, a presença do agente de uma classe praticamente influencia apenas o comportamento dos demais da outra classe, e a intensidade deste efeito transmitido, varia com a distância entre eles.

## Referências

[1] GOTO, M.; AQUINO, V. M. de. Uma aplicação ingênua do método de Monte Carlo: visualização de orbitais atômicos. *Semina: Ciências Exatas e Tecnológicas*, Universidade Estadual de Londrina, v. 13, n. 4, p. 255, dec 1992. ISSN 1676-5451.

[2] MOBLEY, R. K. Fluid power dynamics. In: _____. [S.l.]: Butterworth-Heinemann, 1999. (Plant Engineering Maintenance Series), cap. 2. ISBN 9780750671743,0750671742.

[3] ADLER, J. Bootstrap percolation. v. 171, p. 453–470, 1991.

[4] GUEGUEN, J. D. Y. Transport properties of rocks from statistics and percolation. *Mathematical Geology*, v. 21, n. 1, p. 1–13, 1989. ISSN 1573-8868.

[5] DUNN, J. K. S. W. L. Introduction. In: _____. *Exploring Monte Carlo Methods*. [S.l.: s.n.], 2012. p. 7–20. ISBN 9780444515759.

[6] DUNN, J. K. S. W. L. The basis of monte carlo. In: _____. *Exploring Monte Carlo Methods*. [S.l.: s.n.], 2012. p. 20–46. ISBN 9780444515759.

[7] DEVORE, J. L. Statistical intervals based on a single sample. In: _____. *Probability and Statistics for Engineering and the Sciences*. 9. ed. [S.l.]: Cengage Learning, 2015. cap. 7. ISBN 1305251806, 9781305251809.

[8] DEVORE, J. L. Appendix tables. In: _____. *Probability and Statistics for Engineering and the Sciences*. 9. ed. [S.l.]: Cengage Learning, 2015. cap. A-1. ISBN 1305251806, 9781305251809.

[9] COUZIN, I. D. et al. Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, v. 218, n. 1, p. 1 – 11, 2002. ISSN 0022-5193. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022519302930651>.

[10] COUZIN, I. D.; KRAUSE, J. Self-organization and collective behavior in vertebrates. *Advances in the Study of Behavior*, v. 32, p. 1–75, 2003. ISSN 0065-3454.

**Anexo 1:** Programa referente a lei dos grandes números (tabela 1).

```
////////////////////////////////////////////////////////////////////////////
//FileName: lawoflargenumbers.cpp                                         //
//FileType: C++ Source file                                               //
//Author : Pedro Henrique Fernandes Lobo                                  //
//Created On : 7/04/2019 09:56:39 AM                                      //
//Last Modified On : 8/04/2019 11:53:23 AM                                //
//Copyrights (c) : Pedro Henrique Fernandes Lobo,                         //
//                 Distributed under the MIT software license, see        //
//                 http://www.opensource.org/licenses/mit-license.php     //
//Warning: You might want to compile in this way "g++ -std=c++11          //
//         lawoflargenumbers.cpp -o lawoflargenumbers.out"                //
//Description : This program evaluates the integral $\int_{-1}^{1}        //
//              \frac{1}{\sqrt{1-x^2}}dx$ with the Monte Carlo            //
//              Integration technique and compares the results with its   //
//              analytical solution $\pi$.                                //
////////////////////////////////////////////////////////////////////////////

#include <iostream>
#include <math.h>
#include <iomanip>
#include <random>
#include <limits>

int requestTheSampleSize();
double getTheOpenedLowerBoundOfDistribution(const double lowerBound);
double evaluateZbar(const int sampleSize,
                    std::uniform_real_distribution<double> &uniformDist,
            std::mt19937_64 &generator);

int main(){

    const double a = -1.0; // lower limit of integral
    const double b = 1.0;  // upper limit of integral

    std::random_device rd;
    std::mt19937_64 generator(rd());
    double openedLowerBound = getTheOpenedLowerBoundOfDistribution(a);
    std::uniform_real_distribution<double> uniformDist(openedLowerBound,b);
    int sampleSize = requestTheSampleSize();
    double zBar = evaluateZbar(sampleSize, uniformDist, generator);
    double rn = (b - a) * zBar;
    double relativeError = std::abs(1.0 - rn / M_PI);

    std::cout << "Sample size [N]: " << sampleSize << std::setprecision(5)
              << std::fixed << ", R_N: " << rn << ", Relative error: "
              << relativeError << std::endl;

    return 0;
}

int requestTheSampleSize(){

    float inputVar = 0.0;
    bool inputFail = true;

    while(inputFail){
```

```
57          std::cout << "Enter the sample size [N]: ";
58          std::cin >> inputVar;
59          inputFail = std::cin.fail();
60          if(inputVar < 2 || fmod(inputVar, 1) != 0.0){
61              std::cout << "Error: [N] must be an integer"
62                        << " greater than 1.\n";
63              inputFail = true;
64          }
65          std::cin.clear();
66          std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
67          '\n');
68      }
69
70      int sampleSize = (int)inputVar;
71      return sampleSize;
72  }
73
74  double getTheOpenedLowerBoundOfDistribution(const double lowerBound){
75
76      /* The lower bound is shifted by adding the lowest double,
77          because the std::uniform_real_distribution bound is closed
78          at this point. See https://en.cppreference.com/w/cpp/numeric/
79          random/uniform_real_distribution for more details. */
80
81      const double lowestDouble = std::numeric_limits<double>::min();
82      return lowerBound + lowestDouble;
83  }
84
85  double evaluateZbar(const int sampleSize,
86                      std::uniform_real_distribution<double> &uniformDist,
87                      std::mt19937_64 &generator){
88
89      // zBar is: $\bar{z} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}
90      //                     {\sqrt{1-{x_i}^2}}$
91
92      double xi;
93      double sumZ_xi = 0.0;
94
95      for(int i = 0; i < sampleSize; i++){
96          xi = uniformDist(generator);
97          sumZ_xi += 1.0 / sqrt(1.0 - pow(xi, 2.0));
98      }
99
100     return sumZ_xi / sampleSize;
101 }
```

**Anexo 2:** Programa referente ao teorema do limite central (tabela 2).

```
////////////////////////////////////////////////////////////////////////////
//FileName: limitcentraltheorem.cpp                                       //
//FileType: C++ Source file                                               //
//Author : Pedro Henrique Fernandes Lobo                                  //
//Created On : 12/04/2019 03:06:21 PM                                     //
//Last Modified On : 17/04/2019 11:01:29 PM                               //
//Copyrights (c) : Pedro Henrique Fernandes Lobo,                         //
//                 Distributed under the MIT software license, see        //
//                 http://www.opensource.org/licenses/mit-license.php     //
//Warning: You might want to compile in this way "g++ -std=c++11          //
//         limitcentraltheorem.cpp -o limitcentraltheorem.out sample.h    //
//         sample.cpp". Also this program requires the Boost C++          //
//         library. See https://www.boost.org/ for more detais.           //
//Description : This program estimates the integral $\int_{-1}^{1}        //
//              \frac{1}{\sqrt{1-x^2}}dx$ with the Monte Carlo            //
//              Integration technique and provides confidence             //
//              coefficient and confidence interval.                      //
////////////////////////////////////////////////////////////////////////////

#include <iostream>
#include <math.h>
#include <iomanip>
#include <random>
#include <limits>
#include "sample.h"

int getAmountOfDecimalDigits(double value);
int requestTheSampleSize();
double requestTheLambda();
double getTheOpenedLowerBoundOfDistribution(const double lowerBound);
double evaluateZ_xi(std::uniform_real_distribution<double> &uniformDist,
            std::mt19937_64 &generator);

int main(){

    const double a = -1.0; // lower limit of integral
    const double b = 1.0;  // upper limit of integral

    int sampleSize = requestTheSampleSize();
    float lambda = requestTheLambda();

    std::random_device rd;
    std::mt19937_64 generator(rd());
    double openedLowerBound = getTheOpenedLowerBoundOfDistribution(a);
    std::uniform_real_distribution<double> uniformDist(openedLowerBound,b);

    Sample s;
    s.setLambda(lambda);

    for(int i = 0; i < sampleSize; i++){
        s.addVariable(evaluateZ_xi(uniformDist, generator));
    }

    double marginOfError = (b - a) * s.getMarginOfError();
    double rn = (b - a) * s.getSampleMean();

```

```
57      std::cout << std::fixed
58                << std::setprecision(getAmountOfDecimalDigits(marginOfError))
59                << "confidence interval: " << rn << " +/- " << marginOfError
60                << "; confidence coefficient: " << std::setprecision(3)
61                << s.getConfidenceCoefficient()
62                << "; sample size: " << sampleSize
63                << "; lambda: " << lambda << std::endl;
64
65      return 0;
66  }
67
68  int getAmountOfDecimalDigits(double value){
69
70      int decimalDigitsAmount = 0;
71      while(value < 1.0){
72          value *= 10;
73          decimalDigitsAmount++;
74      }
75
76      return decimalDigitsAmount;
77  }
78
79  int requestTheSampleSize(){
80
81      float inputVar = 0.0;
82      bool inputFail = true;
83
84      while(inputFail){
85          std::cout << "Enter the sample size [N]: ";
86          std::cin >> inputVar;
87          inputFail = std::cin.fail();
88          if(inputVar < 2 || fmod(inputVar, 1) != 0.0){
89              std::cout << "Error: [N] must be an integer"
90                        << " greater than 1.\n";
91              inputFail = true;
92          }
93          std::cin.clear();
94          std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
95          '\n');
96      }
97
98      int sampleSize = (int)inputVar;
99      return sampleSize;
100 }
101
102 double requestTheLambda(){
103
104     double lambda = -1.0;
105     bool inputFail = true;
106
107     while(inputFail){
108         std::cout << "Enter the lambda value: ";
109         std::cin >> lambda;
110         inputFail = std::cin.fail();
111         if(lambda < 0.0 || inputFail){
112             std::cout << "Error: lambda must be a real "
113                       << "greater or equal than 0.0.\n";
114             inputFail = true;
```

```
115         }
116         std::cin.clear();
117         std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
118         '\n');
119     }
120 
121     return lambda;
122 }
123 
124 double getTheOpenedLowerBoundOfDistribution(const double lowerBound){
125 
126     /* The lower bound is shifted by adding the lowest double,
127         because the std::uniform_real_distribution bound is closed
128         at this point. See https://en.cppreference.com/w/cpp/numeric/
129         random/uniform_real_distribution for more details. */
130 
131     const double lowestDouble = std::numeric_limits<double>::min();
132 
133     return lowerBound + lowestDouble;
134 }
135 
136 double evaluateZ_xi(std::uniform_real_distribution<double> &uniformDist,
137             std::mt19937_64 &generator){
138 
139     // z_xi is: $z(x_i) = \frac{1}{\sqrt{1-{x_i}^2}}$
140     double xi;
141     xi = uniformDist(generator);
142     return 1.0 / sqrt(1.0 - pow(xi, 2.0));
143 }
```

```
1  ////////////////////////////////////////////////////////////////////////
2  //FileName: sample.h                                                  //
3  //FileType: C++ Source file                                           //
4  //Author : Pedro Henrique Fernandes Lobo                              //
5  //Created On : 13/04/2019 04:06:32 AM                                 //
6  //Last Modified On : 13/04/2019 04:59:00 AM                           //
7  //Copyrights (c) : Pedro Henrique Fernandes Lobo,                     //
8  //                 Distrbuited under the MIT software license, see    //
9  //                 http://www.opensource.org/licenses/mit-license.php //
10 //Warning: This class requires the Boost C++ library.                 //
11 //         See https://www.boost.org/ for more detais.               //
12 //Description : This program evaluates the mean and variance of normal //
13 //              distributions. The boost library can provide the mean  //
14 //              and standard deviation, but we calculate separately for//
15 //              educational purposes.                                  //
16 ////////////////////////////////////////////////////////////////////////
17 
18 #ifndef SAMPLE_H
19 #define SAMPLE_H
20 
21 #include <boost/math/distributions/normal.hpp> // for normal_distribution
22 
23 class Sample{
24 public:
25     Sample();
26     ~Sample();
27     double getSampleMean();
28     double getMarginOfError();
```

```
    double getConfidenceCoefficient();
    void addVariable(double x);
    void setLambda(double zScore);
    double getSampleVariance();

private:

    int sampleSize;
    double sampleMean;
    double sumTermInVariance; // $\sum_{i=1}^N(x - \bar{x})^2$
    double lambda;
    double sampleVariance;
    void updateSampleMean(double x);
    void updateSampleVariance(double x);

};
#endif
```

```
////////////////////////////////////////////////////////////////////////
//FileName: sample.cpp                                                //
//FileType: C++ Source file                                           //
//Author : Pedro Henrique Fernandes Lobo                              //
//Created On : 8/04/2019 01:08:21 PM                                  //
//Last Modified On : 09/04/2019 00:01:49 PM                           //
//Copyrights (c) : Pedro Henrique Fernandes Lobo,                     //
//                 Distributed under the MIT software license, see    //
//                 http://www.opensource.org/licenses/mit-license.php //
//Description : Implementation of sample.h methods                    //
////////////////////////////////////////////////////////////////////////

#include "sample.h"

Sample::Sample(){

    sampleSize = 0;
    sampleMean = 0.0;
    sumTermInVariance = 0.0;
    lambda = 0.0;
    sampleVariance = 0.0;
}

Sample::~Sample(){

}

double Sample::getSampleMean(){

    return sampleMean;
}

double Sample::getMarginOfError(){

    return lambda * sqrt(sampleVariance / sampleSize);
}

double Sample::getConfidenceCoefficient(){

    boost::math::normal distribution(0.0, 1.0);
```

```
41      return cdf(distribution, lambda) - cdf(distribution, -lambda);
42  }
43
44  void Sample::addVariable(double x){
45
46      sampleSize++;
47      updateSampleMean(x);
48      updateSampleVariance(x);
49  }
50
51  void Sample::updateSampleVariance(double x){
52
53      // Welford's method for computing sample variance.
54
55      double previousSampleMean;
56
57      if(sampleSize > 1){
58          double dummyVar = sampleSize * sampleMean - x;
59          previousSampleMean = dummyVar / (sampleSize - 1);
60      }
61      else{
62          previousSampleMean = sampleMean;
63      }
64
65      double delta = x - previousSampleMean;
66      double delta2 = x - sampleMean;
67      sumTermInVariance += delta * delta2;
68      sampleVariance = sumTermInVariance / (sampleSize - 1);
69  }
70
71  void Sample::updateSampleMean(double x){
72
73      sampleMean += (x - sampleMean) / (1.0 * sampleSize);
74  }
75
76  void Sample::setLambda(double lambda){
77
78      this -> lambda = lambda;
79  }
80
81  double Sample::getSampleVariance(){
82
83      return sampleVariance;
84  }
```