UNIVERSIDADE FEDERAL DO RIO GRANDE

PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

SIMULAÇÃO DA CURVA DE CRESCIMENTO DO MYCOBACTERIUM TUBERCULOSIS UTILIZANDO SISTEMAS MULTIAGENTES.

PABLO SANTOS WERLANG

Rio Grande

2013

PABLO SANTOS WERLANG

SIMULAÇÃO DA CURVA DE CRESCIMENTO DO MYCOBACTERIUM TUBERCULOSIS UTILIZANDO SISTEMAS MULTIAGENTES.

Dissertação apresentada à Universidade Federal do Rio Grande, como parte das exigências do programa de Pós-Graduação em Modelagem Computacional, para obtenção do título de Mestre.

Orientador: Prof. Dr. Adriano V. Werhli

Co-Orientadora: Profa. Dra. Diana F. Adamatti

Rio Grande

2013

W489s Werlang, Pablo Santos.
Simulação da curva de crescimento do mycobacterium tuberculosis utilizando sistemas multiagentes/ Pablo Santos Werlang – 2013.
86 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande/FURG, Programa de Pós-Graduação em Modelagem Computacional.

Orientador: Dr. Adriano V. Werhli.
Coorientadora: Drª. Diana F. Adamatti.

1. Mycobacterium tuberculosis. 2. Sistemas multiagentes. 3. Simulação computacional. 4 Curva de crescimento. I. Werhli, Adriano. II. Adamatti, Diana. III.Título.

CDU 519.67

Catalogação na fonte: Bibliotecária Alessandra Lemos CRB10/1530

PABLO SANTOS WERLANG

SIMULAÇÃO DA CURVA DE CRESCIMENTO DO MYCOBACTERIUM TUBERCULOSIS UTILIZANDO SISTEMAS MULTIAGENTES.

Dissertação apresentada à Universidade Federal do Rio Grande, como parte das exigências do programa de Pós-Graduação em Modelagem Computacional, para obtenção do título de Mestre.

BANCA EXAMINADORA

Prof. Dr. Adriano Velasque Werhli
Orientador - FURG

Prof.ª Dra. Karina dos Santos Machado
FURG

Prof.ª Dra. Andrea Von Groll
FURG

Prof. Dr. Antônio Carlos da Rocha Costa
FURG

Prof.ª Dra. Cecilia Dias Flores
UFCSPA

Rio Grande

2013

Dedico este trabalho aos meus pais,
Armindo e Rosane, que sempre me
apoiaram e me aconselharam durante minha
trajetória de vida, à minha irmã Paloma
pela sua amizade e alegria, à minha
namorada Cristine por sua compreensão e
companheirismo, e a todos meus amigos
pelos momentos felizes compartilhados.

AGRADECIMENTOS

Ao meu orientador, Adriano Werhli e co-orientadora, Diana Adamatti, por sempre estarem dispostos a me auxiliar na construção do meu conhecimento.

Ao curso de pós-graduação em modelagem computacional por me acolher em seu programa e tornar possível a conclusão de mais esta etapa em minha carreira acadêmica.

À todos professores do curso de Engenharia de computação da FURG, que me ajudaram na minha formação e me deram as bases necessárias para poder enfrentar o mestrado.

À equipe da Faculdade de Medicina da FURG, em especial à Andrea Von Groll, que disponibilizaram seu tempo, conhecimento e materiais de estudo dos quais este trabalho se baseia.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A tuberculose é uma doença infecciosa causada pelo bacilo *Mycobacterium tuberculosis*. Ainda hoje a tuberculose preocupa os profissionais da saúde, principalmente nos países em desenvolvimento. O estudo da curva de crescimento do *Mycobacterium tuberculosis* é muito importante e possibilita o estudo de suas características e o desenvolvimento de novos fármacos. Porém, a realização de testes experimentais com este bacilo é bastante demorado, levando pelo menos três semanas para mostrar algum resultado, e muitas vezes falham por causa de contaminação ou desidratação do meio. Portanto, é objeto de pesquisa o estudo de modelos que representem com fidelidade a curva de crescimento do *M. tuberculosis* e que obtenham estes resultados em um tempo menor.

Com este fim, existem diversos modelos matemáticos que descrevem curvas de crescimento de bactérias, e embora bastante precisos e úteis, carecem de uma correspondência clara com o ambiente real do qual tentam reproduzir.

Tomando uma outra direção, o presente trabalho descreve um modelo de curvas de crescimento de bactérias baseado em sistemas multiagentes. Este tipo de modelo possui por característica a modelagem do comportamento individual de agentes se utilizando de parâmetros que possuem alguma relação com variáveis observadas no ambiente real. O crescimento populacional dos agentes resulta em uma curva de crescimento que converge com o resultado obtido nas curvas experimentais.

Além disso, o modelo apresentado também conta com um mecanismo de inferência de parâmetros, que possibilita que através da entrada de valores de uma curva real e a aplicação de um método numérico possa encontrar os valores de parâmetros da simulação que resultarão em uma curva de crescimento o mais próximo possível da curva experimental.

Os resultados obtidos foram bastante satisfatórios e as curvas geradas através dos parâmetros automaticamente encontrados pelo método alcançaram um nível de similaridade bastante próximo às curvas reais, o que torna o modelo bastante útil para a verificação de hipóteses, uma vez que as simulações levem minutos e os testes de hipóteses horas, em oposição aos dias que levariam para realizar tais testes *in vitro*.

Palavras-chave: *Mycobacterium tuberculosis*, sistemas multiagentes, simulação computacional, curva de crescimento.

ABSTRACT

Tuberculosis is an infectious disease caused by *Mycobacterium tuberculosis*. Even today tuberculosis worries health professionals, especially in third world countries. The study of the growth curve of *Mycobacterium tuberculosis* is very important and allows the study of its characteristics and the development of new drugs. However, experimental tests with this bacillus is very time consuming, and therefore it is extremely useful to use a model which faithfully represents the growth curve of *M. tuberculosis* to obtain these results in a shorter time.

To this end, there are several mathematical models that describe growth curves, and although quite accurate and useful, they lack a clear correspondence with the actual environment which they try to reproduce.

Taking another direction, this paper describes a model developed based on multi-agent systems. This kind of model has the characteristic of individual behavior modeling using parameters that are related to the observed variables in the real environment. The population growth of the agents results in a growth curve that converges with the results obtained in the experimental curves.

Furthermore, the model also has a parameter inference engine, which in face of a real curve data and the application of a numerical method can find the parameter values of the simulation that in turn will result in a growth curve similar to the experimental curve.

The results obtained were satisfactory and curves generated using parameters found automatically by the method achieved a considerable degree of similarity with real curves. This makes the model useful for testing hypotheses in a matter of hours as opposed to days that would take to perform such tests *in vitro*.

Keywords: *Mycobacterium tuberculosis*, multi-agent systems, computer simulation, growth curve.

1. INTRODUÇÃO

A tuberculose é uma das doenças infecciosas mais antigas de que se tem notícia e que ainda hoje faz vítimas. Ela é causada pelo *Mycobacterium tuberculosis*, bactéria cujo comportamento é previsível e conhecido.

Um ponto muito importante no estudo do *M. tuberculosis* é sua curva de crescimento, pois através dela se pode determinar a resistência da bactéria a certos fármacos, aos diversos tipos de ambientes de cultura, e as características genéticas de diferentes cepas do *M. tuberculosis* (VON GROLL, 2010).

Existem diversos modelos que representam crescimento de populações da *M. tuberculosis*, porém os mais aceitos são determinísticos e carecem da devida representatividade dos componentes biológicos inerentes a esse tipo de sistema. Por esse motivo, tais modelos são pouco flexíveis quando se deseja testar novas hipóteses ou situações biológicas diversas pois não há uma correspondência clara de suas variáveis com o sistema biológico observado.

Uma alternativa é a utilização de Sistemas Multiagentes (SMA). Os SMA são uma poderosa e flexível ferramenta para a modelagem deste tipo de ambiente, pois com eles pode-se obter resultados que representam o comportamento de cada indivíduo, ao invés de uma média dos comportamentos do sistema (WOOLDRIDGE, 2009).

As variáveis descritas em um modelo utilizando SMA também possuem uma relação mais próxima com as variáveis observadas em experimentos reais, fato que torna a implementação e aprimoramento de um modelo SMA muito mais simples.

O uso de SMA tornariam testes de hipóteses utilizando *M. tuberculosis*, que normalmente demoram dias para serem concluídos, muito mais rápidos e baratos pois substituiriam o uso de material orgânico pelas simulações computacionais. Os resultados mais promissores *in silico* poderiam ser encaminhados para testes *in vitro* para confirmação das hipóteses.

1.1. OBJETIVOS

1.1.1. OBJETIVO GERAL

O objetivo deste trabalho é a implementação de um modelo capaz de refletir com precisão as curvas de crescimento do *M. tuberculosis* auxiliando diversas outras pesquisas relacionadas ao bacilo. Uma vez implementado, o modelo possibilita aos seus usuários realizar simulações relacionados ao bacilo em minutos ao invés de dias, que é o tempo que testes *in vitro* do *M. tuberculosis* despende. Esta melhoria na etapa de testes da curva de crescimento é possível desde que se conheça as características da população sendo modelada.

Um importante aspecto das simulações é obter resultados que mantenham o mesmo nível de precisão dos experimentos *in vitro*, porém em tempo muito inferior, e para isso é necessário que a modelagem da curva de crescimento seja fiel ao comportamento da curva real, e para tal é necessário conhecer os valores dos parâmetros da curva.

O modelo também possui por objetivo apresentar uma solução para este problema através da inferência ou estimativa de parâmetros, visando obter como resultado um meio automático de encontrar os valores dos parâmetros que modelam a curva de crescimento dada como entrada.

E por fim, o modelo possui por objetivo demonstrar que através da modelagem de populações do *M. tuberculosis*, se utilizando do conceito de *Quorum Sensing* um comportamento mais próximo ao observado nas curvas reais é obtido.

1.1.2. OBJETIVOS ESPECÍFICOS

A elaboração de um modelo no ambiente computacional NetLogo, onde as principais características comportamentais conhecidas das bactérias e um ambiente propício ao seu desenvolvimento é modelado.

O estudo de curvas de crescimento obtidas experimentalmente e de trabalhos relacionados do *M. tuberculosis* a fim de sua incorporação ao modelo criado.

A adaptação do modelo às especificidades do *M. tuberculosis* e ao meio de desenvolvimento observado nos experimentos reais.

Criação de um algoritmo que uma vez implementado no modelo busca automaticamente, através de um refinamento dos parâmetros, valores para cada um dos

parâmetros relevantes a fim da simulação resultante gerar curvas de crescimento próximas às curvas reais estudadas.

Demonstrar um estudo comparativo dos resultados obtidos com as curvas reais estudadas.

1.2. ORGANIZAÇÃO DO TRABALHO

Esta dissertação está dividida em seis capítulos. No capítulo 2 será apresentado o referenciais teóricos sobre o estudo da *Mycobacterium tuberculosis*, dos Sistema Multiagentes e Métodos Numéricos, bem como uma breve abordagem sobre trabalhos da área que são relacionados ao trabalho desenvolvido. No capítulo 3 é explicado o desenvolvimento do modelo como um todo. No capítulo 4 são apresentados os resultados gerados a partir do modelo desenvolvido. No capítulo 5 são realizadas conclusões sobre o trabalho, bem como sugestões para trabalhos futuros. No capítulo 6 são expostas as referências bibliográficas e no capítulo 7 um anexo onde consta o código-fonte que deu origem ao modelo desenvolvido.

2. REFERENCIAL TEÓRICO

2.1. TUBERCULOSE

A tuberculose é uma doença infectocontagiosa, de ocorrência mundial, causada pelo *Mycobacterium tuberculosis*, um bacilo álcool-ácido resistente, curvo, imóvel, intracelular e aeróbio facultativo (GOLDMAN e AUSIELLO, 2004). Foi descrito pela primeira vez por Robert Koch em 1882, por isso sendo também conhecido como Bacilo de Koch (BK).

Há registros da presença da tuberculose nas civilizações antigas do Egito, Grécia, Roma e Hindu, sendo descrita por Hipocrates (460A.C – 370A.C) como a doença mais disseminada dos tempos. Embora a descoberta do agente causador seja recente, estudos moleculares encontraram o bacilo em múmias egípcias datadas de 3000A.C. Nos séculos 17, 18 e 19 a tuberculose era conhecida como "Praga Branca", sendo a principal causa de morte na Europa. No século 19 foi considerada pandêmica, com altos índices de mortalidade em todos continentes (VON GROLL, 2010). Escritores como Edgar Alan Poe e Álvares de Azevedo, à época do romantismo, tiveram influência na aparência emagrecida e pálida dos doentes para a criação de seus personagens, reflexo da sentença de morte recebida pelos diagnosticados com a doença na época, motivo pelo qual ficou também conhecida como "mal do século".

Em 1890, o próprio Robert Koch desenvolveu a tuberculina, um extrato de glicerina com o bacilo, sem sucesso como medicamento, a substância foi adaptada por Von Piquet e até hoje é utilizada para a detecção precoce e assintomática da infecção (PPD – teste tuberculínico). Em 1906 Albert Calmette e Camille Guerin conseguiram atenuar o bacilo *Mycobacterium bovis* e criaram a primeira vacina de sucesso (BCG), utilizada desde 1926 até os dias atuais.

A primeira droga antituberculose desenvolvida foi a estreptomicina, em 1944, seguida pelo ácido para-aminossalicílico (PAS). Entretanto, cepas resistentes à estreptomicina logo apareceram, tornando necessária a terapia multidroga. Desde então, novas drogas foram introduzidas no mercado, atualmente sendo recomendado tratamento primário com três drogas (rifampicina, isoniazida e pirazinamida).

Entre as décadas de 60 e 80, a incidência e mortalidade da tuberculose diminuíram bastante, especialmente na Europa e EUA, sendo sua mortalidade relacionada a idosos e pessoas com resposta imune deficiente. Entretanto, com o advento do vírus HIV na década de

90, a incidência e mortalidade pela tuberculose voltaram a preocupar, especialmente nos países em desenvolvimento. Em 1993, a Organização Mundial de Saúde (OMS) estimou que um terço da população estaria contaminada pelo *Mycobacterium tuberculosis*, levando à implementação global de diretrizes para o diagnóstico e tratamento da tuberculose (GOLDMAN e AUSIELLO, 2004).

2.1.1. CURVA DE CRESCIMENTO

O estudo da curva de crescimento do *Mycobacterium tuberculosis* tem-se mostrado útil em um grande número de situações, como o estudo de mutações genéticas associadas a resistência a fármacos, resposta a condições físico-químicas, dentre outros.

Porém, um grande problema no estudo das curvas do *Mycobacterium tuberculosis* é que o bacilo possui uma taxa de crescimento extremamente lenta, e o processo de contagem da população é muito trabalhoso, despendendo até três semanas.

Recentemente, Von Groll (2010) padronizou um método para determinação da curva de crescimento baseado em um sistema que obtém o crescimento bacteriano através do monitoramento do consumo de oxigênio no meio líquido, por meio de um sensor de oxigênio que emite fluorescência. Este instrumento é denominado Tubo indicador de crescimento bacteriano, ou *Mycobacteria Growth Indicator Tube* (MGIT) em inglês. Von Groll utilizou o BACTEC MGIT 960 (MGIT Procedure Manual, 2006) e através dele obteve os resultados observados em (VON GROLL, 2010).

As etapas do crescimento bacteriano podem ser divididos em quatro fases (ver Figura 2.1):

- **Fase lag**: Nesta fase as bactérias estão passando por uma adaptação ao meio onde estão inseridas. Na fase lag, embora haja atividade metabólica, não há crescimento populacional significativo. Este fato é devido à necessidade das bactérias adaptarem seu metabolismo ao novo ambiente, que era diferente ao ambiente anterior que estavam inseridas antes do inoculo, e também ao fato que como a população inicial é baixa, o tempo necessário para atingir uma diferença populacional mínima para que possa ser capturada pelo instrumento de medição é alta
- **Fase log**: também chamada de fase exponencial, a fase log é caracterizada pelo rápido aumento da população. Nesta etapa as bactérias consomem o máximo dos

nutrientes do ambiente, liberam seus resíduos no ambiente, e liberam moléculas químicas sinalizadoras (*quorum sensing*[1]).

- **Fase estacionária**: Nesta etapa o crescimento populacional entra em estagnação. A concentração de bactérias no meio nesta etapa é alta, e elas entram em um estado de dormência onde poupam energia para aumentar o tempo de sobrevivência.
- **Fase de morte**: Nesta etapa as bactérias começam a morrer devido à existência de poucos nutrientes no ambiente para serem consumidos, alta concentração de resíduos tóxicos, falta de oxigênio e variações de pH.

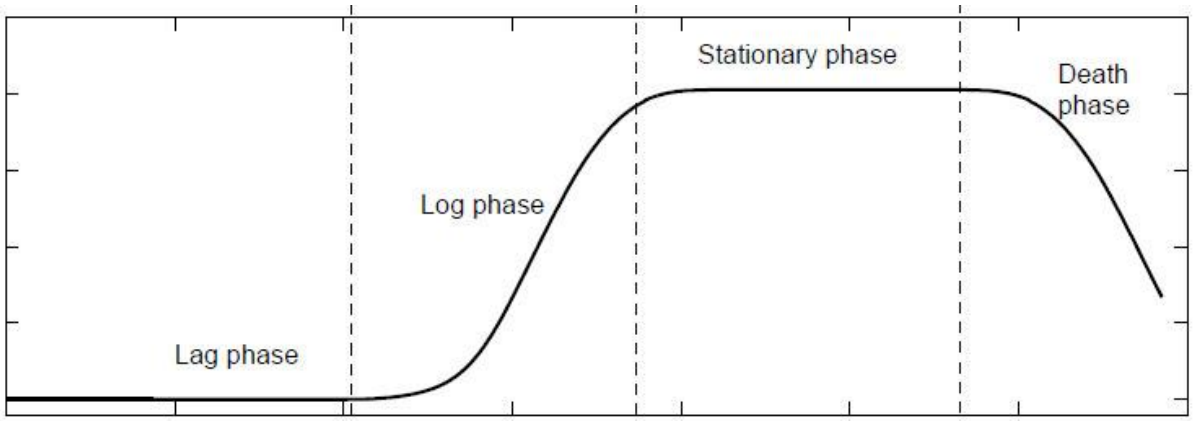


Figura 2.1 - Curva de crescimento e suas fases

2.2. SISTEMAS MULTIAGENTES

A área da computação denominada inteligência artificial visa resolver problemas ou desempenhar tarefas através do uso do conhecimento e possui a capacidade de aproveitar associações e inferência para trabalhar com problemas complexos que assemelham-se a problemas reais (REZENDE, 2005).

Os agentes são entidades criadas para resolver determinados problemas ou cumprir certas tarefas, respeitando o conjunto de informações que eles possuem e que definirão seus

[1] Quorum sensing é um processo de comunicação de uma espécie bacteriana onde sinalização através da produção de moléculas bioquímicas ocorre. Estas bactérias detectam a presença das moléculas sinalizadoras no ambiente após um certo limiar de concentração, e então mudanças na regulação dos genes ocorrem. Estas mudança se manifestam de diferentes maneiras dependendo da espécie em questão, variando desde alteração no nível de virulência da infecção, até ativação de características como bioluminescência (SIFRI, 2008).

comportamentos. Conforme (WOOLDRIDGE, 2009), um agente é um sistema de computador situado em um ambiente, e é capaz de ações autônomas em seu ambiente para fim de atingir seus objetivos estabelecidos.

Os Sistemas Multiagentes (SMA) foram criados através da pesquisa de inteligência artificial distribuída, que aborda questões sobre a resolução de problemas em ambientes onde a distribuição, física ou não, é inerente. Em um sistema multiagentes, um conjunto independente de agentes com características diferentes compartilham um meio, e através deste meio as ações de um irão interferir nas ações/estados dos outros. Estas interações possuem caráter cooperativo ou competitivo, dependendo do modelo em questão.

Segundo (REZENDE, 2005), "o foco de pesquisa (de sistemas multiagentes) reside nos modelos para conceber agentes, suas organizações e interações de modo genérico, para que possam ser instanciadas num caso particular quando um determinado problema é colocado para a sociedade de agentes e esta de deve resolvê-lo".

Segundo a autora, ao abordar a resolução de um problemas através da modelagem de SMA, deve-se criar o ambiente e as regras de interação entre os agentes e dos agentes para com o ambiente, e uma vez o modelo criado, o comportamento do sistema como um todo deverá emergir da interação dos agentes entre si e com o ambiente.

Quando se modela uma situação da vida real através da perspectiva dos sistemas multiagentes, certas etapas, descritas a seguir, podem ser seguidas (FROZZA, 1997 apud ADAMATTI, 2007, p. 30):

- Decompor o fenômeno em um conjunto de elementos autônomos;
- Modelar cada um dos elementos como um agente definindo seu conhecimento, funções, comportamento e modos de interação;
- Definir o ambiente dos agentes;
- Definir quais agentes possuem a capacidade de ação e comunicação.

Os modelos baseados em agentes são especialmente úteis pois oferecem alto grau de modularidade. Um problema muito complexo, grande ou imprevisível pode ser razoavelmente abordado quando confrontado por partes menores e específicas. Um agente pode fazer o papel deste módulo, e o problema em questão poderá ser resolvido utilizando a interação entre esses agentes em um ambiente colaborativo (SYCARA, 1998).

Quando junta-se o conceito de um modelo baseado em agentes com simulações computacionais, obtêm-se o que chama-se de simulação multiagente (do inglês Multi-Agent-Based Simulation, MABS).

Em uma simulação multiagente utiliza-se um problema ou alguma situação presente na vida real. O modelo final é o resultado da interação entre os agentes que o formam, e estes agentes são a representação individual de cada elemento do problema apresentado (ADAMATTI, 2007).

As simulações multiagente são especialmente úteis nas áreas da medicina, ciência da computação, ciências sociais, dentre outros. Existem diversos softwares de simulação multiagente que se propõe a solucionar problemas da área e demonstrar teorias, como por exemplo modelos de disseminação de doenças, comportamento de grandes massas, resolução de problemas computacionais evolutivos, controle de tráfego, etc.

Os propósitos das simulações multiagente podem ser divididas em três grandes áreas (DROGOUL e FERBER, 1992):

- Testar hipóteses sobre a emergência de estruturas sociais dos comportamentos e interações de cada indivíduo. Isto é feito testando as condições mínimas em nível micro que são necessárias para observar estas estruturas a nível macro.
- Construir teorias para contribuir para o desenvolvimento da compreensão geral de sistemas etológicos, sociológicos e psico-sociológicos, relacionando comportamentos com propriedades estruturais e organizacionais.
- Integrar diferentes teorias parciais de várias disciplinas (por exemplo, sociologia, etnologia, etologia, psicologia cognitiva) em um framework diferente, provendo ferramentas que permitem a integração de diferentes estudos.

No presente trabalho, segue-se a linha do primeiro propósito, onde o objetivo é criar um modelo utilizando sistemas multiagentes que ajude na compreensão e no estudo das curvas de crescimento do *Mycobacterium tuberculosis*, determinando características quantitativas da população através da análise comparativa dos resultados obtidos experimentalmente e na simulação.

2.2.1. A LINGUAGEM NETLOGO

O modelo desenvolvido busca simular o comportamento de uma colônia de bactérias do tipo *Mycobacterium tuberculosis*. Para tal, foi feito a escolha de se utilizar o software de simulação multiagente NetLogo. Este software foi desenvolvido com o propósito de facilitar o desenvolvimento de modelos que possuem o uso de sistemas multiagentes como foco (NetLogo 5.0.4 User Manual, 2013). O NetLogo trabalha com uma linguagem de programação voltada ao uso de sistemas multiagente, o que torna os códigos escritos simples e legíveis. O NetLogo também proporciona uma interface gráfica amigável, que mostra visualmente os agentes presentes no ambiente, além de facilitar o controle de variáveis e parâmetros através da adição de barras deslizáveis, botões de ação, e áreas de plotagem de gráficos.

A criação de modelos no NetLogo pode ser dividida em duas partes. A interface e o código. Na Figura 2.2 pode-se observar que existem três abas na parte superior da janela do software. Ao escolher a aba interface, o usuário possui diversas opções para criação de parâmetros, botões, plotagens e monitores do ambiente, além de um monitor que serve para visualizar os agentes dispostos no ambiente, ou seja, a simulação em si. Este monitor especial pode ser observado na Figura 2.2.

Para adicionar um botão na interface, basta escolher o item *button* na caixa destinada a adição de itens.

Uma vez que o botão seja adicionado ao espaço de interface do modelo, uma janela aparecerá. Nesta janela, o campo *Display Name* deverá ser preenchido com o texto que aparecerá como rótulo do botão, e no campo *commands* deverá conter todos os comandos que serão executados no evento do botão ser clicado. Normalmente adiciona-se uma chamada de função neste campo. A caixa de seleção *forever* deverá ser marcada caso seja desejável que os comandos executados no evento do clique do botão sejam continuamente executados em repetição uma vez o botão seja clicado pela primeira vez.

Os *slider* adicionam a possibilidade da utilização de parâmetros facilmente ajustáveis através de barras deslizáveis no modelo. Na janela de configuração do *slider*, é necessário inserir o nome da variável global que este *slider* irá representar no modelo. Há também os campos para inserção do valor máximo e mínimo mostrado na barra deslizante.

Os *switches*, ou chaves, são ferramentas que uma vez adicionados no modelo utilizam uma variável global que recebe os valores verdadeiro ou falso para controlar algum

evento na simulação. Os parâmetros, botões e chaves após criados ficam dispostos no ambiente conforme demonstra a Figura 2.3.

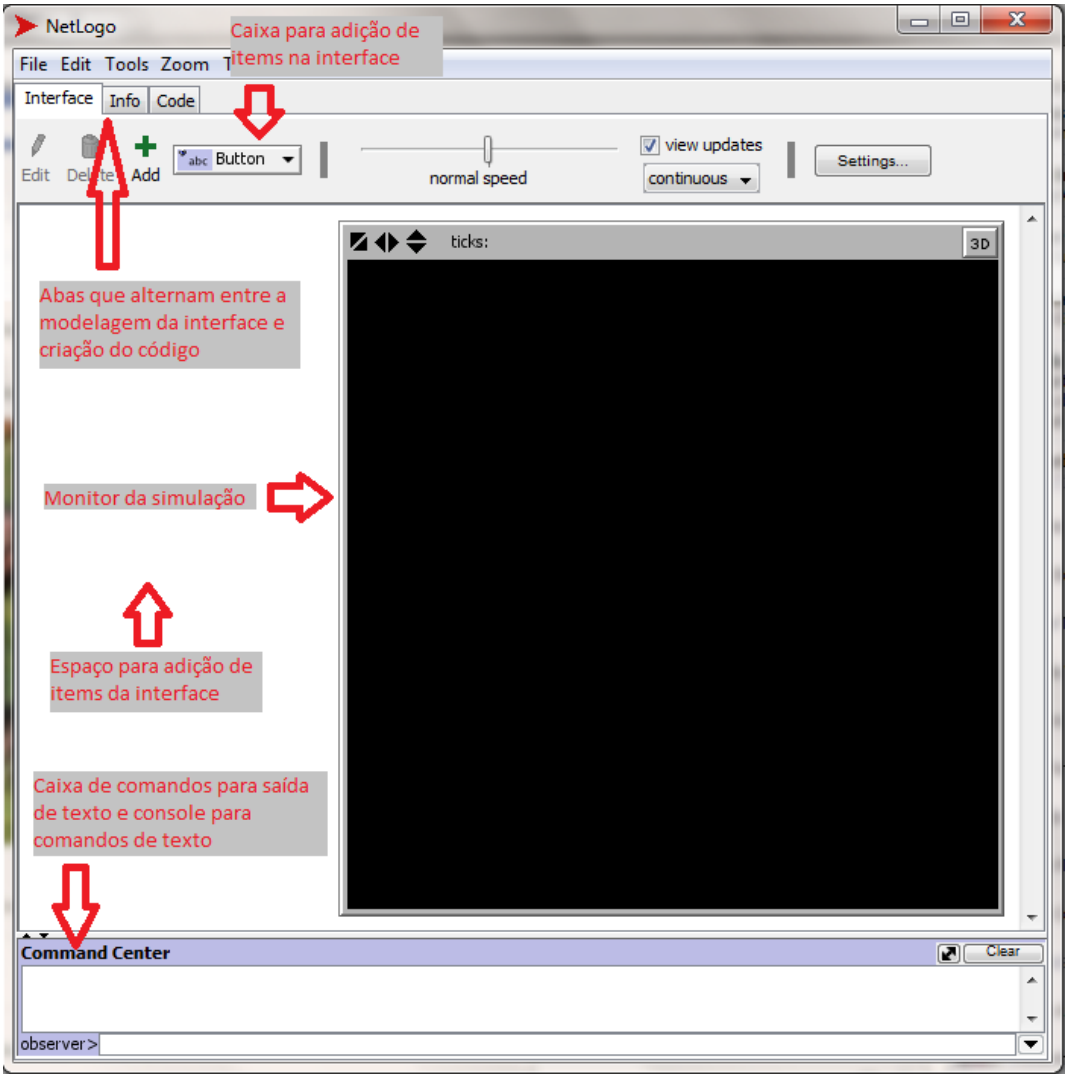


Figura 2.2 - Tela inicial do NetLogo. As setas e caixas de textos indicam os elementos principais da aba Interface.

Os monitores são ferramentas que servem para observar textualmente valores da simulação. Estes valores podem ser conteúdos de variáveis ou operações mais complexas. O valor mostrado no *monitor* automaticamente é atualizado.

Através dos *plots* é possível visualizar o estado de variáveis ao longo do tempo na simulação. Gráficos em função do tempo da simulação são gerados para a função do eixo y escolhida. Ao criar um *plot* é possível escolher os rótulos dos eixos x e y do gráfico, bem como escolher quantas séries de dados serão apresentadas (nomeadas *pens* no NetLogo). Para cada série, é possível escolher qual cor a representará, bem como o que esta série deve

representar. Este deve ser um valor retornado por um comando do NetLogo. Exemplos de monitores e de um gráfico pode ser observado na Figura 2.4.

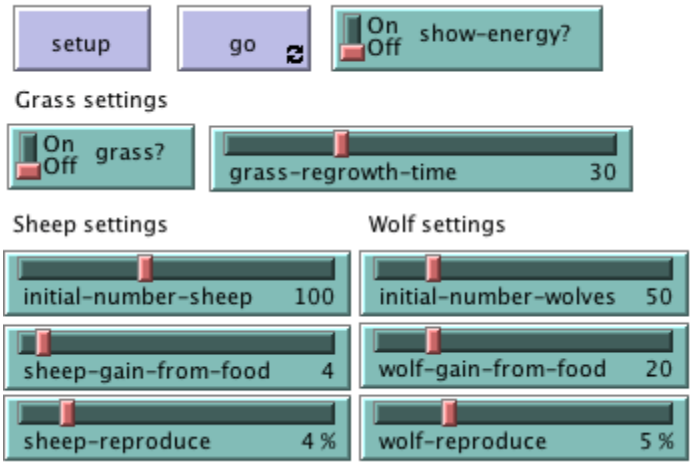


Figura 2.3 - Exemplos de botões (*buttons*), chaves (*switches*) e parâmetros (sliders) retirados de um modelo de exemplo do NetLogo.
(NetLogo 5.0.4 User Manual, 2013)

Após inserir os elementos da interface do modelo, torna-se necessário escrever o código que irá fazer com que os agentes desempenhem suas atividades. Na aba *code*, apresentada na Figura 2.2, é possível inserir toda a programação do modelo, seguindo a sintaxe da linguagem. Tutoriais sobre como criar e editar modelos podem ser encontrados em (NetLogo 5.0.4 User Manual, 2013).

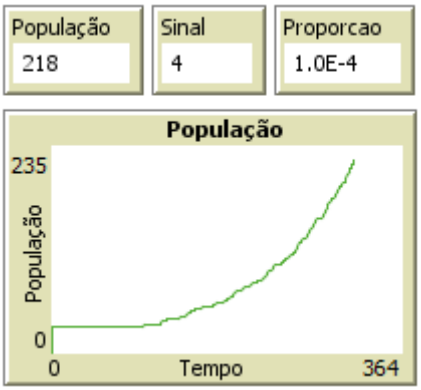


Figura 2.4 - Exemplos de monitores (*monitors*) e gráfico (*plot*) no ambiente NetLogo.

2.3. MÉTODOS NUMÉRICOS

No universo de problemas da matemática e física, muitos deles possuem soluções bastante complexas e particulares, resolvidos analiticamente através de equações diferenciais. Exemplos destes problemas podem ser encontrados em diversas áreas da mecânica dos fluidos, aerodinâmica, resistência dos materiais, termodinâmica, dentre outros.

Dentre a maioria destes problemas práticos, uma solução não exata, porém com um alto grau de aproximação é o suficiente.

Tendo em vista estes problemas, os métodos numéricos surgiram como uma maneira de resolver problemas complexos da matemática através de soluções aproximadas.

Os métodos numéricos computacionais se valem de algoritmos iterativos para encontrar a solução destes problemas. Estes algoritmos buscam, em um universo definido e finito de possibilidades, uma solução aproximada para o problema, e na medida que o número de iterações for aumentando, a solução se torna cada vez mais refinada até o ponto que esta solução satisfaça o erro mínimo aceitável (FREITAS, 2000).

2.3.1. O ERRO MÁXIMO ACEITÁVEL

O conceito de erro está diretamente ligado aos métodos numéricos. A definição de erro neste contexto ocorre pela diferença do resultado obtido e o resultado esperado.

Conforme mencionado anteriormente, o algoritmo utilizado para a resolução do problema realiza uma série de iterações que progressivamente refinam o espaço de busca e aproximam a solução encontrada esperada.

Ao fim de cada iteração um resultado é obtido. Com este resultado consegue-se obter o erro. Este erro poderá ser um erro absoluto ou erro relativo:

- Erro absoluto: é quando obtem-se o erro fazendo a diferença entre o resultado esperado e o obtido.
- Erro relativo: Obtem-se o erro relativo calculando a porcentagem de desvio que o resultado obtido teve do esperado.

Por exemplo: Se o resultado esperado for:

$R_e = 100$ e o resultado obtido $R_o = 102{,}5$, tem-se um erro absoluto:

$$e_a = |100 - 102{,}5| = 2{,}5$$

e um erro relativo:

$$e_r = {}^{e_a}/_{R_e} = {}^{2,5}/_{100} = 0{,}025$$

Definindo-se o maior erro aceitável para o caso da aplicação que está sendo modelada, o algoritmo poderá ser executado sucessivamente até que o erro máximo seja satisfatório. Em cada nova iteração com espaço de busca reduzido, o erro deverá ser conferido e as execuções deverão encerrar quando for encontrado uma solução que satisfaça o critério do erro máximo aceitável.

Tabela 2.1 - Exemplos de cálculo de erros.

Resultado obtido	**Resultado esperado**	**Erro absoluto**	**Erro relativo**
7,56	7,45	0,11	0,0148
8,91	8,9	0,01	0,0011
101,7	105	3,3	0,0314
1,51	1,59	0,08	0,0503
10	13,88	3,88	0,2795

2.3.2. DETERMINAÇÃO DOS ZEROS DE FUNÇÕES

Para a resolução de um problema são necessárias duas coisas principais. A compreensão e a metodologia para a resolução do mesmo.

Uma das classe de problemas solucionáveis através de métodos numéricos é a determinação dos zeros de uma função matemática. Este tipo de aplicação tem uma grande utilidade, como, por exemplo a obtenção dos instantes onde níveis de tensão são nulos em um circuito na área da engenharia elétrica, até os valores de força aplicados à atuadores para que um sistema seja estável na área da robótica.

No presente trabalho, a determinação dos zeros de uma função é útil pois da determinação de um ponto exato de uma curva matematicamente conhecida, pode-se chegar a um resultado com um grau de erro tão pequeno quanto o desejado através dos métodos numéricos.

Sendo uma função $f(x)$, os zeros desta função são todos os pontos de x para $f(x) = 0$.

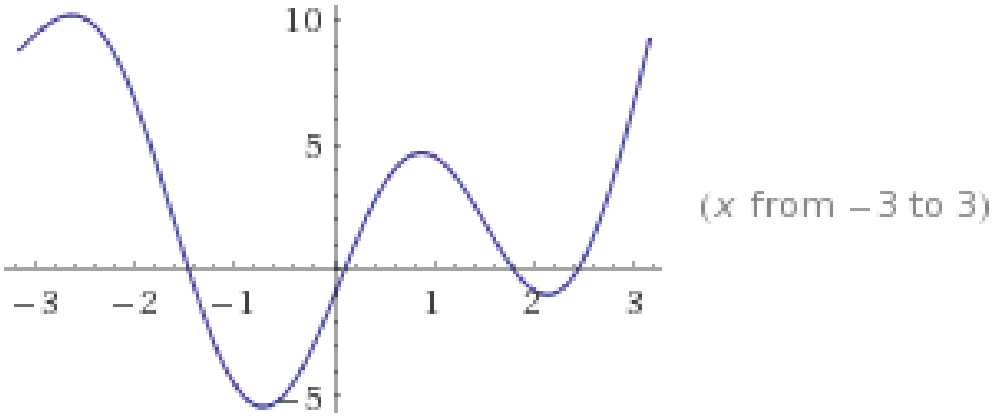


Figura 2.5 - Gráfico da função $f(x) = x^2 + 5sen(2x) - 1$.
Esta função possui quatro pontos onde $f(x) = 0$.
(Wolfram Alpha, 2013).

Para utilizar-se alguma técnica para encontrar os zeros de uma função, é necessário definir os limites do espaço de busca. Para isso é necessário fazer uma análise prévia do gráfico da função em questão e então escolher dois pontos a e b. Estes pontos devem ser tais que $f(a) < 0$ e $f(b) > 0$. Desta maneira, pode-se garantir que o ponto x em que se encontra o zero da função ($f(x) = 0$) está entre a e b.

2.3.3. O MÉTODO DA BISSECÇÃO

Com os limites do espaço de busca definidos, o próximo passo para obtenção do zero da função é o refinamento do espaço de busca. Para realizar tal feito existem diversos algoritmos que implementam cada método numérico.

Dentre os métodos numéricos usados para obtenção de zero de funções está o método da bissecção, ou dicotomia.

O método da bissecção consiste em calcular a média x_0 dos pontos a e b e utilizá-lo como valor de x na função. Caso o resultado $f(x_0) > 0$, o ponto b receberá o valor de x_0, e caso $f(x_0) < 0$, o valor de x_0 irá substituir o valor do ponto a.

Este processo deverá ser repetido até que a diferença entre $f(a)$ e $f(b)$ seja menor que o erro e estipulado.

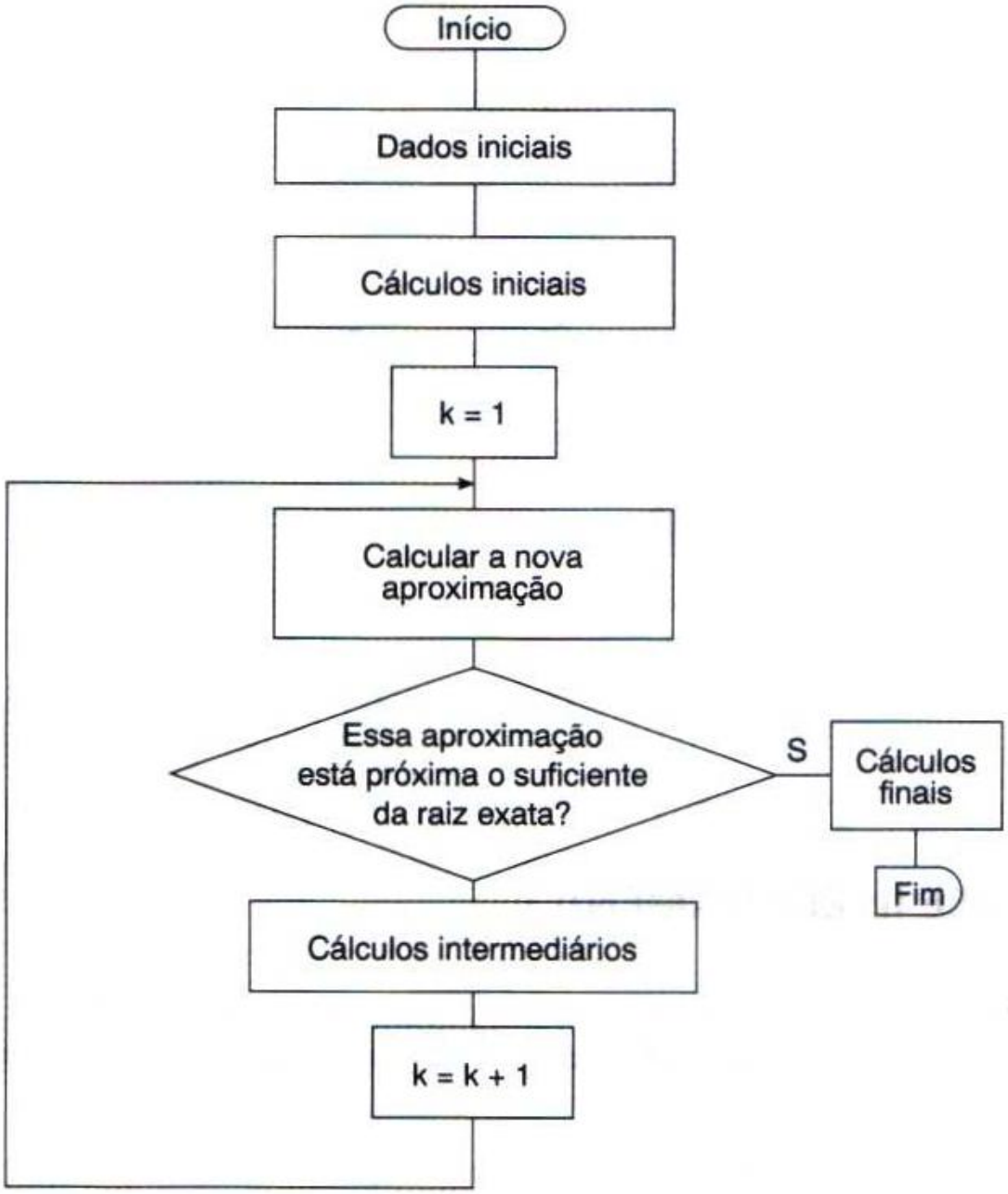


Figura 2.6 - Fluxograma demonstrando os passos a serem seguidos para obtenção do zero da função.

(RUGGIERO e LOPES, 1997).

Demonstração de uso do método:

Encontrar o zero da função $x^3 + 2x - 27$, para um erro máximo de $e = 0{,}02$

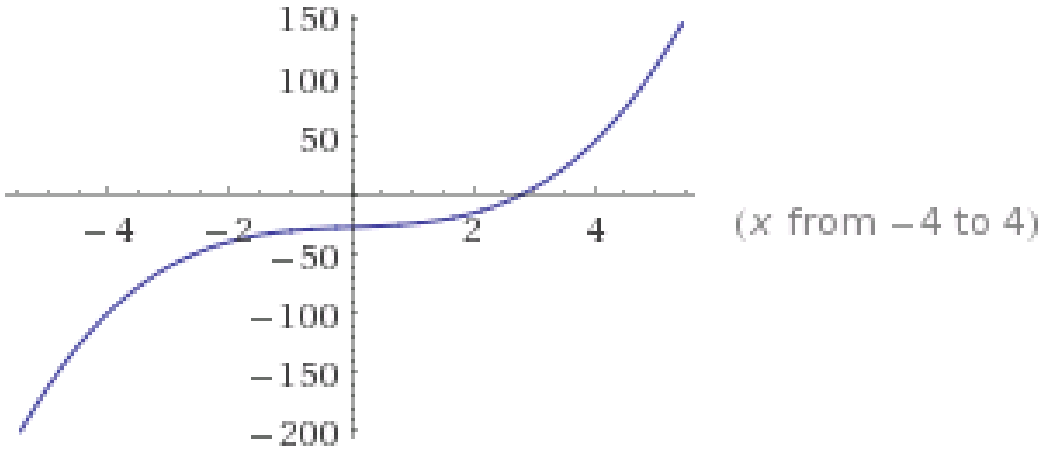


Figura 2.7 - Gráfico da função $x^3 + 2x - 27$.

(Wolfram Alpha, 2013).

Sabendo que o zero da função está entre 2 e 3, define-se:

$a = 2$

$b = 3$

$x_0 = \frac{2+3}{2} = 2{,}5$

$f(x_0) = f(2{,}5) = (2{,}5)^3 + 2(2{,}5) - 27 = -6{,}375$

$b - a = 1$

$a = x_0 = 2{,}5$

$b = 3$

$x_1 = \frac{2{,}5+3}{2} = 2{,}75$

$f(x_1) = f(2{,}75) = (2{,}75)^3 + 2(2{,}75) - 27 = -0{,}703125$

$b - a = 0{,}5$

$a = x_1 = 2{,}75$

$b = 3$

$x_2 = \frac{2{,}75+3}{2} = 2{,}875$

$f(x_2) = f(2{,}875) = (2{,}875)^3 + 2(2{,}875) - 27 = 2{,}5136$

$b - a = 0{,}25$

$a = 2{,}75$

$b = x_2 = 2{,}875$

$x_3 = \frac{2{,}75 + 2{,}875}{2} = 2{,}8125$

$f(x_3) = f(2{,}8125) = (2{,}8125)^3 + 2(2{,}8125) - 27 = 0{,}8723$

$b - a = 0{,}125$

$a = 2{,}75$

$b = x_3 = 2{,}8125$

$x_4 = \frac{2{,}75 + 2{,}8125}{2} = 2{,}7812$

$f(x_4) = f(2{,}7812) = (2{,}7812)^3 + 2(2{,}7812) - 27 = 0{,}0752$

$b - a = 0{,}0625$

$a = 2{,}75$

$b = x_4 = 2{,}7812$

$x_5 = \frac{2{,}75 + 2{,}7812}{2} = 2{,}7656$

$f(x_5) = f(2{,}7656) = (2{,}7656)^3 + 2(2{,}7656) - 27 = -0{,}3160$

$b - a = 0{,}0312$

$a = x_5 = 2{,}7656$

$b = 2{,}7812$

$x_6 = \frac{2{,}7656 + 2{,}7812}{2} = 2{,}7734$

$f(x_6) = f(2{,}7734) = (2{,}7734)^3 + 2(2{,}7734) - 27 = -0{,}1209$

$b - a = 0{,}0156 < e$

Desta maneira, pode-se concluir que com um erro menor que 0,02, o ponto de x que corresponde ao zero da função $f(x) = x^3 + 2x - 27$ é $x = 2{,}7734$.

Para resolver computacionalmente um problema através do método da bissecção, é necessário a implementação de um algoritmo usando alguma linguagem de programação. Na

Figura 2.8 o algoritmo em português estruturado. Note que os valores das variáveis *a*,*b* e *e* foram informadas arbitrariamente para corresponder o exemplo mostrado anteriormente.

```
VARIAVEIS
      numerico a,b,e,x,fx
INICIO
      a=2
      b=3
      e=0.02
      FACA ENQUANTO b-a > e
            x=(a+b)/2
            fx=x^3+2*x-27
            SE fx<0 ENTAO
                  a=x
            SENAO
                  b=x
            FIM SE
      FIM ENQUANTO
FIM
```

Figura 2.8 - Algoritmo exemplificando o uso do método da bissecção.

2.4. TRABALHOS RELACIONADOS

A simulação computacional de organismos simples data de 1984, com os trabalhos de Christopher Langton (LANGTON, 1984). Desde então, diversos trabalhos vêm surgindo que abordam esta temática.

A abordagem do assunto vida computacional basicamente se divide em dois campos. Segundo Langton (1997, p.1):

> "Em vida artificial, há uma grande divisão intelectual, similar à que ocorre na Inteligência Artificial, entre sistemas "projetados" para cumprir alguma tarefa complexa através de qualquer meio que o desenvolvedor conseguir bolar, mesmo que estes sejam distantemente relacionados ao modo como os sistemas naturais funcionam, e sistemas criados para modelar precisamente sistemas biológicos e criados para testar hipóteses biológicas".

No segundo campo, os recursos computacionais são utilizados para criar organismos artificiais dotados de características inerentes aos seres vivos. O objetivo é criar modelos

computacionais que resultem em indivíduos dotados de comportamentos que emergiram automaticamente. Estes comportamentos então são estudados e comparados aos dos organismos reais existentes. Este ramo de estudo chama-se "Vida Artificial" (ou do inglês Artificial Life) (LANGTON, 1997).

No primeiro campo criam-se modelos inspirados nos organismos reais, e a partir das regras de comportamento observadas, criam-se os modelos que visam obter respostas o mais próximo possível das encontradas nos organismos originais.

Enquanto no segundo campo as pesquisas são direcionadas a compreender melhor o funcionamento da vida, no primeiro os problemas são de ordem mais prática e geralmente são concentrados na resolução de algum problema específico.

Ainda neste primeiro ramo, pode-se destacar duas correntes principais. Uma delas busca encontrar os resultados esperados através de simulações com modelos matemáticos, enquanto a outra realiza simulações computacionais se utilizando de modelos baseados em sistemas multiagentes.

Os modelos matemáticos se utilizam de funções sigmoidais para modelar o comportamento de crescimento populacional. As funções sigmoidais tem por característica possuírem uma fase lag, uma exponencial e uma estacionária.

Um comparativo entre as diversas funções sigmoidais aplicadas ao crescimento bacteriano é abordado em (ZWIETERING, JONGENBURGER, *et al.*, 1990), onde os autores concluem que a função de Gompertz atinge um resultado mais preciso na maioria dos casos.

Em (BUCHANAN, WHITING e DAMERT, 1996) os autores propõe um modelo para ajuste de curvas de crescimento de bactérias chamado *three-phase linear*, e os compara aos modelos matemáticos de Gompertz e de Baranyi. O modelo divide as fases de crescimento das bactérias em três fases distintas (lag, exponencial e estacionária) e possui somente quatro parâmetros. Os autores se utilizaram de dados experimentais do *Escherichia coli* para realizar suas comparações.

Os modelos que utilizam sistemas multiagentes se preocupam com o comportamento individual dos seres virtuais (chamados agentes), ao invés de se preocupar com a curva de crescimento e com a população como um todo. O fato da curva de crescimento resultante destes modelos se tornarem semelhantes às curvas reais é uma característica emergente destes sistemas, ao invés de uma característica modelada.

Um exemplo de característica emergente pode ser observado em (KRAWCZYK, DZWINEL e YUEN, 2003), onde os agentes modelados representam bactérias do Anthrax (*bacillus anthracis*). O modelo proposto pelos autores usa uma grade 2D retangular para

representar o ambiente. Os agentes consomem nutrientes do ambiente e quando atingem um certo estado de saúde estão prontos para se reproduzir, que é um processo análogo à divisão celular. Os agentes também podem morrer, caso não hajam nutrientes a serem consumidos. Outra característica interessante, e que teve papel crucial na característica emergente do modelo é que os agentes possuem orientação horizontal, vertical e diagonal. Os resultados apresentados pelos autores mostram que as colônias que se desenvolverem melhor foram as que apresentaram uma geometria circular, formando uma "estrutura de camadas".

O modelo desenvolvido em (KREFT, BOOTH e WIMPENNY, 1998) também se utiliza de agentes. Os autores criaram agentes com regras comportamentais genéricas para as culturas bacterianas, com o intuito do modelo apresentado servir como base para o estudo do crescimento em diferentes aplicações. Os agentes presentes no modelo absorvem nutrientes e com isso ganham massa celular bem como produzem resíduos no ambiente. Quando atingem um certo limiar, os agentes se reproduzem (imitando o processo de divisão celular). Cada agente gasta uma quantidade fixa de energia a fim de manter suas atividades metabólicas, e caso o consumo de nutrientes seja menor que o necessário para esta manutenção, o agente perderá massa celular. Após certo limiar, o agente morre, e sua massa celular é reintegrada no ambiente com uma porcentagem de aproveitamento como parte do substrato.

Um outro trabalho, que destaca a convergência comportamental de agentes presentes em modelos multiagente, descrito em (CLAUS e BOUTILIER, 1998), relata as dinâmicas de um modelo multiagente utilizando aprendizagem por reforço. Neste trabalho os autores apresentam um SMA onde os agentes interagem em um jogo cooperativo. O foco do trabalho é comparar o tempo de convergência das ações dos agentes que se utilizam do algoritmo Q-learning quando expostos a um jogo cooperativo com diversas possibilidades de ações. Os autores verificam que quando submetidos à estratégias conjuntas (JAL - Join-Action Learner) os agentes convergem para os mesmo resultados em uma velocidade não tão superior que quando submetidos à estratégias individualistas (IL - Independent Learner). Os autores explicam que o Q-learning não é tão robusto em estratégias JAL quanto é em estratégias IL.

O trabalho apresentado em (TERANO, 2006) aborda o problema da exploração de espaços de busca de parâmetros em modelos baseados em agentes. O autor relata sobre os problemas clássicos encontrados na maioria dos trabalhos relacionados a modelagem multiagente, dentre eles a dificuldade da obtenção dos valores de parâmetros que correspondam a uma função objetivo conhecida. O autor então propõe que a criação do modelo deva ser feita a partir da função objetivo, e através da técnica de Simulação Inversa, o modelo que melhor representa a situação que busca-se modelar emerge das simulações.

No método da Simulação Inversa deve-se estipular uma grande quantidade de parâmetros que modelem o ambiente, e então submeter este ambiente a testes de modo a modificá-lo, visando uma adaptação a uma função objetivo.

O autor se utiliza de Algoritmos Genéticos para fazer a Simulação Inversa e adaptar o ambiente ao objetivo. Desta maneira esta busca por força bruta que a princípio é inviável torna-se plausível. Por fim o modelo resultante é analisado a fim de classificar os parâmetros que são relevantes para a modelagem. O trabalho apresentado se utiliza de métodos numéricos ao invés de técnicas semelhantes às apresentadas nos trabalhos relacionados por questões de velocidade de convergência. Um método numérico que encontre zero de funções normalmente irá convergir mais rápido que um método estocástico.

3. MODELO PROPOSTO

O modelo proposto possui o objetivo de criar um ambiente de simulação computacional capaz de reproduzir as curvas de crescimento do *M. tuberculosis* bem como realizar testes para encontrar parâmetros capazes de reproduzir curvas de crescimento dadas como entrada. Para tal, os conhecimentos abordados no Capítulo 2 se fizeram presentes no desenvolvimento da simulação.

3.1. A SIMULAÇÃO

A simulação transcorre em intervalos de tempo chamados *ticks*. A cada *tick* cada um dos agentes tem a oportunidade de realizar uma ou mais ações, podendo ser: deslocar-se, absorver nutrientes, secretar resíduos, liberar moléculas sinalizadoras ou reproduzir-se. O fato de em cada momento da simulação existir uma quantidade diferente de agentes ativos simultaneamente, bem como o poder computacional da máquina em que a simulação está sendo executada variar, faz com que o tempo real de cada *tick* não possa ser calculado com precisão. Por conta disso, existe a necessidade de uma unidade de medida de tempo relativo ao transcorrido dentro da simulação, ao invés de uma unidade real de tempo.

A simulação conta com agentes que representam as bactérias *Mycobacterium tuberculosis*. Estes agentes possuem características individuais que são modeladas como variáveis do agente. Estas características são essenciais para que os agentes representem seu papel no ambiente e interajam com ele, da mesma maneira que uma bactéria interagiria com seu meio.

A seguir estão descritas todas as características dos agentes:

- **energia**: descreve o quão saudável o agente está.
- **consumo**: determina o quanto o agente está consumindo dos nutrientes do ambiente a cada intervalo de tempo. Também determina a taxa com que ele consome a sua energia.
- **adaptação**: determina o tempo que o agente levará para fazer a transição da fase lag para a fase log.

- **sense**: determina se o agente detectou no ambiente uma quantidade suficiente de moléculas sinalizadoras. Uma vez que esta detecção seja positiva, o agente entra em um estado de consumo reduzido.

Os agentes possuem um espaço para realizarem suas atividades. Este ambiente é compartilhado por todos os agentes, que embora autônomos, interagem dentro deste espaço. É neste ambiente comum que os agentes irão encontrar os nutrientes necessários para sua sobrevivência, bem como e onde irão depositar os resíduos de suas atividades metabólicas.

Cada pequeno trecho no ambiente é chamado *patch*, e cada um destes trechos possui suas próprias características. Estas características são os nutrientes e os resíduos, que são respectivamente o recurso que os agentes consomem para produzirem energia e os resíduos que os agentes produzem e depositam no ambiente após metabolizar os nutrientes.

Na etapa de adaptação presente no crescimento bacteriano, estes micro-organismos precisam de um certo tempo para ajustar suas funções metabólicas ao ambiente em que eles estão inseridos, consumindo nutrientes para realizar esta adaptação, ao invés de consumí-los para a reprodução.

Os agentes do modelo proposto passam por uma etapa análoga, onde existe um parâmetro chamado *laglim*, que define quanto tempo (em *ticks*) os agentes irão demorar até terminarem a etapa de adaptação, isto é, realizarem a transição da fase lag para a fase log.

Após este período, os agentes se reproduzem de maneira semelhante ao processo de divisão celular, onde o novo agente recém criado é uma cópia exata do agente que deu origem a ele.

Para o processo de reprodução acontecer, o agente precisa possuir uma quantidade de energia mínima. No processo esta energia é dividida entre os dois seres, metade para o agente que deu origem, e a outra metade para o novo agente.

Portanto, o fator limitante para a reprodução dos agentes é a quantidade de energia que eles conseguem absorver do ambiente através da metabolização dos nutrientes. Estes eventos representam a fase log.

A cada *tick*, os agentes consomem do ambiente, mais especificamente do *patch* em que estão, uma determinada quantidade de nutrientes, dado pelo consumo do agente. Quanto maior for o valor do consumo do agente, maior será a quantidade de nutrientes que este agente conseguirá absorver do *patch* a cada *tick*.

Após metabolizar os nutrientes absorvidos, o agente adquire energia e secreta resíduos. A energia absorvida, além de ser guardada para ser utilizada na reprodução, também

serve para manter as funções vitais do agente. A quantidade de energia necessária para manter o agente é dada pelo valor da variável *enconsumo*. Caso o agente venha a absorver menos energia que o mínimo necessário para manter seu metabolismo, ele começará a perder energia, e caso acabe esgotando este recurso por completo, ele acabará morrendo.

Os resíduos que o agente secreta no ambiente são chamados de *sujeira*, e como o depósito de *sujeira* no ambiente faz com que a concentração de nutrientes neste mesmo *patch* reduza, o acúmulo de resíduos em um *patch* faz com que em futuros *ticks*, os agentes consigam absorver menos nutrientes daquele mesmo trecho. Esta característica do modelo faz com que os agentes tenham mais dificuldade de sobreviver em um ambiente muito saturado.

A cada *tick*, o agente também possui uma chance de secretar uma molécula sinalizadora, dado pela variável *taxasinal*. Esta molécula química sinalizadora é utilizada pela bactéria para perceber a densidade populacional e "comunicar-se" com outras em um processo chamado *Quorum Sensing* (SIFRI, 2008). Este processo é utilizado para coordenar e otimizar o comportamento e atividade metabólica de várias espécies bacterianas.

Quando o agente detectar que a concentração da molécula sinalizadora no ambiente for alta, ele entrará em um estado de consumo reduzido, onde ele gradualmente consome menos nutrientes, gera menos resíduos e se reproduz menos.

A detecção da concentração é um meio indireto de se estimar o quão populoso o ambiente está no momento. Esta medida indica quando a população deverá parar de crescer. Caso não pare, os agentes irão sofrer falta de energia por causa da escassez de nutrientes, e consequentemente irão morrer.

Esta detecção da concentração é feita comparando o valor da variável *loglim* com a proporção entre o total de moléculas sinalizadoras pelo tamanho total do ambiente. Caso esta proporção seja maior que o valor de *loglim*, o agente entrará no estado de consumo reduzido.

Ao entrar no estado de consumo reduzido, o agente irá gradualmente reduzir seu consumo (e consequentemente gerar menos resíduos e obter menos energia) em uma taxa por *tick*, dada pelo valor da variável *decaiconsumo*. Quando os agentes entrarem neste estado e eventualmente pararem de se reproduzir, a população irá estagnar, e o crescimento entrará na fase estacionária.

Embora o consumo do agente reduza drasticamente quando no estado de consumo reduzido, existe um mínimo de energia que precisa ser consumido a fim de manter as funções básicas do agente, e qualquer valor de consumo abaixo disso é impossível, ou seja, ele morre. Este valor é dado pela variável *consumomin*.

A cada *tick* os agentes também movimentam-se aleatoriamente pelo ambiente em busca de *patches* que possuam uma concentração de nutrientes mais alta, pois desta maneira eles conseguem explorar todos os *patches* de sua vizinhança, maximizando sua obtenção de energia.

O modelo desenvolvido possui diversas variáveis, algumas delas já explicadas anteriormente. Através da intervenção do usuário, que poderá configurar os diferentes parâmetros do ambiente, resultados próximos aos experimentos reais poderão ser obtidos.

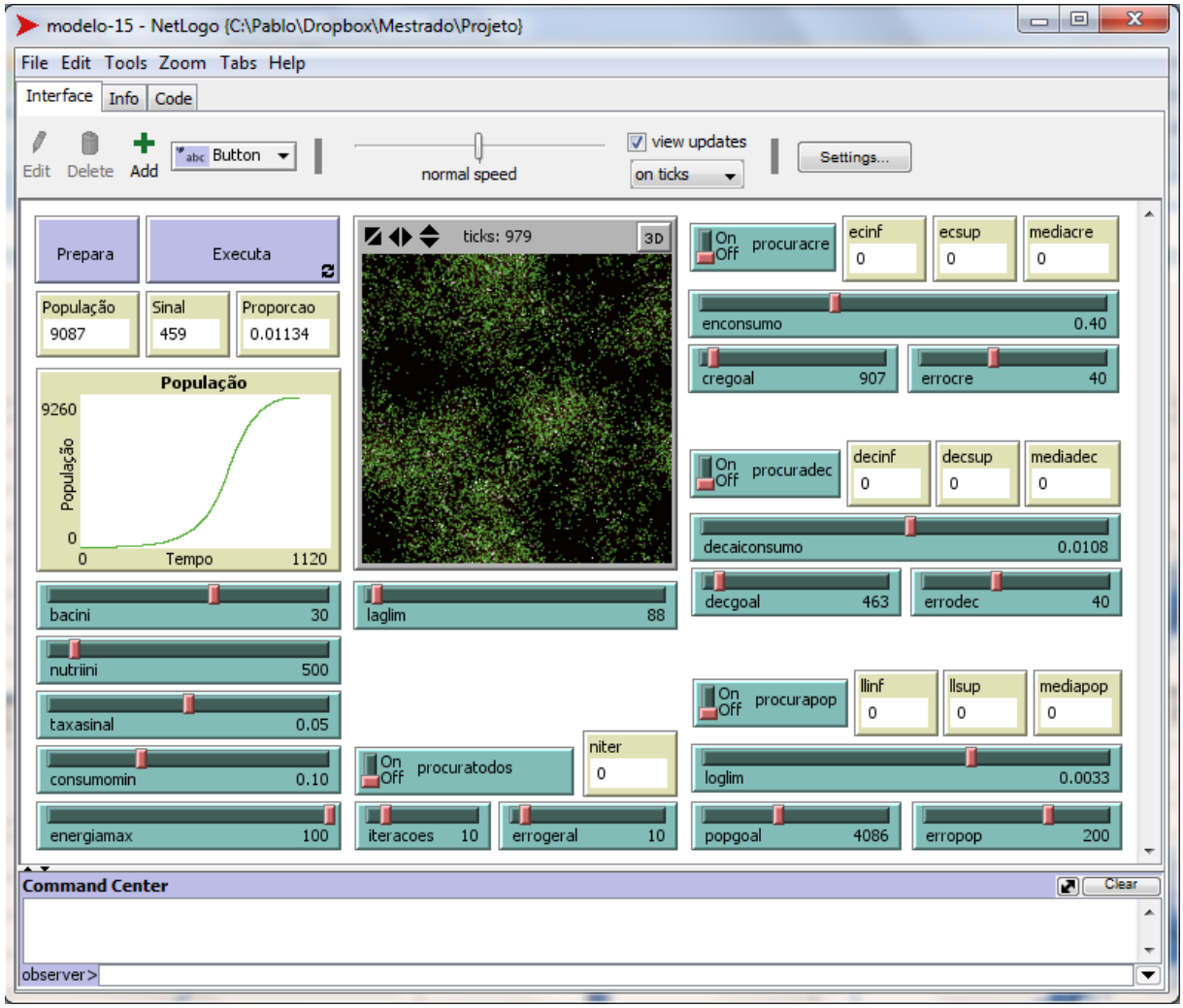


Figura 3.1 - A interface do modelo criado usando NetLogo.

Os parâmetros iniciais que podem ser configurados pelo usuário são os seguintes:

- ***bacini*** - números inicial de agentes
- ***nutriini*** - nutrientes que cada *patch* do ambiente possuirá
- **laglim** - tempo de adaptação dos agentes

- **loglim** - limiar onde os agentes irão entrar no estado de consumo reduzido
- **decaiconsumo** - taxa gradual de decaimento do consumo ao entrar no estado de consumo reduzido
- **consumomin** - consumo mínimo de energia
- **taxasinal** - chance do agente secretar uma molécula sinalizadora
- **enconsumo** - quantidade de energia gasta por *tick* para manter as funções vitais do agente
- **energiamax** - quantidade de energia máxima que um agente pode possuir.

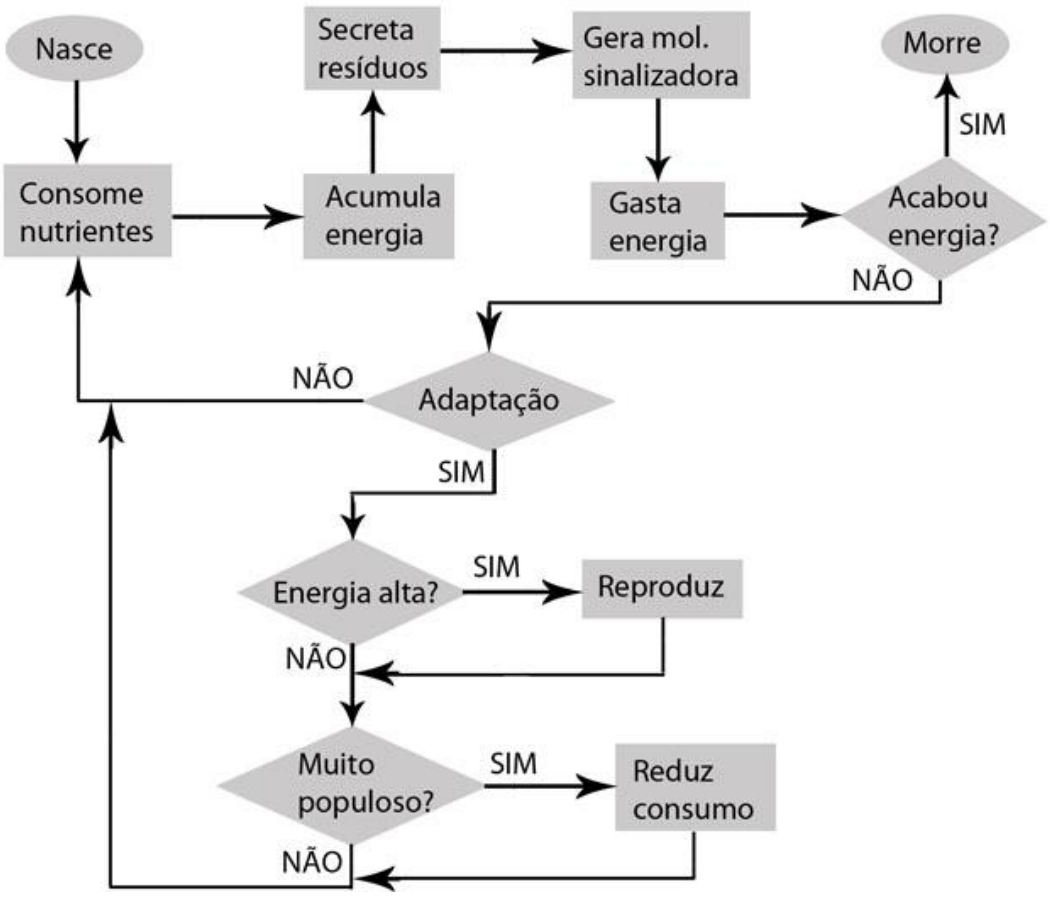


Figura 3.2 - Fluxograma do ciclo de vida dos agentes

Na Figura 3.2, o ciclo de vida dos agentes é apresentado, explicitando de maneira gráfica os processos de decisão e as ações que eles tomam a cada *tick*.

O agente inicia o ciclo consumindo nutrientes do *patch* em que ele está situado. Por conta disso, o agente ganhará uma determinada quantidade de energia, então após metabolizar essa energia, ele secretará resíduos (*sujeira*) no ambiente. Logo em seguida, o agente irá

secretar a molécula sinalizadora (com uma chance dada por *taxasinal*). Após todas estas tarefas desempenhadas, parte da energia reserva do agente será consumida. Caso a energia do agente acabe, ele morrerá. Caso contrário, ele seguirá executando seu ciclo de vida.

Quando o período de adaptação do agente terminar, ele então criará um novo agente (processo de reprodução), caso ele tenha bastante energia em sua reserva.

Em algum momento o agente perceberá que o ambiente está com uma concentração muito grande de moléculas sinalizadoras por causa do *quorum sensing*, e neste instante ele reduzirá a quantidade do seu consumo, que embora não cause nenhum efeito imediato, fará com que o agente no próximo ciclo absorva menos energia do ambiente, fazendo com que o número de ciclos gastos até a próxima reprodução aumente.

3.2. IMPLEMENTAÇÃO DA SIMULAÇÃO

A partir do modelo desenvolvido foi possível realizar a simulação. A seguir, trechos do código fonte são apresentados a fim de ampliar a compreensão sobre o desenvolvimento do presente trabalho, bem como criar um vínculo entre a teoria e o modelo desenvolvido na prática. O código desenvolvido é apresentado integralmente no anexo do presente trabalho.

Para realizar as tarefas apresentadas na seção anterior e ilustradas na Figura 3.2, cada agente passa pelas seguintes etapas:

```
1     set proporcao count patches with [any? sinal-here] / count
patches
2     ask bac [
3       ifelse adaptacao > laglim [
4         move
5         metaboliza
6         reproduz
7       ]
8       [
9         adapta
10      ]
11      if proporcao > loglim [set sense true]
12    ]
```

Figura 3.3 - Ciclo geral de vida do agente.

A Figura 3.3 descreve a rotina que cada agente executará a cada *tick*. A condição presente na linha 3 mostra que até que os agentes tenham terminado a etapa de adaptação, eles

não poderão executar outras atividades. Após isso, eles irão mover-se (linha 4), consumir e absorver energia, gerar resíduos (linha 5) e então reproduzir-se (linha 6).

Caso a proporção de moléculas sinalizadores pelo tamanho do ambiente seja maior que a especificada em *loglim*, os agentes entrarão em estado de consumo reduzido, conforme demonstrado na linha 11.

No processo de metabolização o agente irá lidar com todas questões envolvendo a energia que o mantém vivo. A implementação da função que detalha este processo pode ser observado na Figura 3.4, e seu funcionamento é descrito a seguir.

```
1     to metaboliza
2       if sense = true [
3         set consumo consumo * (1 - decaiconsumo)
4       ]
5
6       if nutrientes > 0 [
7         set nutrientes nutrientes - consumo
8       ]
9
10      sinaliza
11
12      set sujeira sujeira + consumo
13      if energia < energiamax and nutrientes > 0 [
14        set energia energia + (nutrientes / (nutrientes + sujeira))
* consumo
15      ]
16
17      if energia <= 0 [die]
18
19      ifelse consumo < consumomin [
20        set energia energia - enconsumo * consumomin
21      ]
22      [
23        set energia energia - enconsumo * consumo
24      ]
25    end
```

Figura 3.4 - Função que resolve a etapa de metabolização dos agentes.

Os agentes irão reduzir o seu consumo de energia caso eles já tenham entrado em estado de consumo reduzido, conforme demonstrado nas linhas 2 e 3. O valor desta redução é gradual, e vai depender da variável *decaiconsumo*.

Os agentes absorvem nutrientes do *patch* onde ele está localizado também de acordo com o seu consumo atual, conforme analisado na linhas 7.

Os agentes possuem uma chance de gerar uma molécula sinalizadora, para indicar o quão populoso está o ambiente. Esta etapa está demonstrada na linha 10, e a os detalhes sobre a implementação da função *sinaliza* pode ser visto no anexo do trabalho.

O agente gera resíduos através de sua atividade metabólica, e estes resíduos ficam depositados no *patch* onde ele está situado, esta etapa pode ser observada na linha 12.

Na medida que o agente consome nutrientes do ambiente, ele gera energia para garantir sua sobrevivência e gerar outros agentes. A quantidade de energia que este agente gera é determinada pelo seu consumo, bem como pela saturação de nutrientes que o *patch* que ele está situado possui, conforme pode ser visto na linha 13.

Caso o agente não possua mais energia, ele morre (linha 17).

Cada agente possui uma quantidade mínima de energia para realizar suas funções vitais básicas, independente do quanto ele consome do ambiente. O gasto de energia que o agente tem para se manter vivo vai depender do quão ativo está seu metabolismo no momento (consumo). Porém o gasto mínimo deste agente nunca será menor que seu consumo mínimo (linhas 19-24).

```
1     to reproduz
2       if any? neighbors with [not any? bac-here] and energia >=
energiamax[
3         set energia random energiamax
4         hatch 1 [
5           move-to one-of neighbors with [not any? bac-here]
6           set energia energiamax - energia
7           set sense false
8         ]
9       ]
10    end
```

Figura 3.5 - Função responsável pela reprodução dos agentes.

Conforme observado na Figura 3.5, toda vez que um agente atingir em sua reserva de energia uma quantidade grande o suficiente (linha 2), ele irá criar um outro agente igual à ele ao seu redor. Na etapa de criação a energia total é dividida entre o agente progenitor e o novo agente, conforme observado nas linhas 3 e 6.

3.3. INTERPRETAÇÃO DE VALORES

Para realizar a validação e ajuste do modelo foram obtidos dados experimentais de (VON GROLL, 2010). As curvas de crescimento utilizadas são referentes a cepas que diferem em sua origem geográfica bem como em sua resistência aos fármacos Isoniazida (INH), Rifampicina (RIF), Etambutol (EMB) e Estreptomicina (STM).

Tabela 3.1 - As cinco cepas usadas como exemplo nas simulações, sua origem e padrões de suscetibilidade. (VON GROLL, 2010).

Identificação das cepas	**Origem**	**Suscetibilidade**			
		INH	**RIF**	**EMB**	**STM**
GC 02-2761	Bangladesh	S	S	S	S
GC 03-0850	Bangladesh	R	R	R	R
GC 01-2522	Georgia	S	S	S	S
GC 03-2922	Georgia	R	S	S	R
H37Rv	ATCC	S	S	S	S

O processo de incubação das cepas foi detalhado no trabalho de (VON GROLL, 2010), e os gráficos resultantes detalham a relação de unidades de crescimento (Growth Units, GU) por hora.

Porém, quando as curvas de crescimento experimentais são obtidas através do MGIT (*Mycobacteria Growth Indicator Tube*) a informação que estas curvas apresentam é o produto de diversos fatores do ambiente de cultura das bactérias. Sendo assim, é muito difícil extrair informações isoladamente destas curvas. Por exemplo, não há a informação de qual a taxa que as bactérias reduzem seu consumo, uma vez que o estado de saturação do ambiente é atingido, não há como coletar essa medida. Ou ainda, qual a proporção de moléculas sinalizadoras do ambiente é necessária para que a colônia entre em estado de consumo reduzido.

Tendo isto em vista, é necessário realizar a inferência das informações relevantes destas curvas e através da observação dos resultados obtidos. Com estes resultados, estimar os valores dos parâmetros do modelo que resultarão em uma curva de crescimento semelhante à curva que está tentando modelar.

Primeiramente, é importante ressaltar que até mesmo a tarefa de estudar e analisar a curva real em busca dos padrões de comportamento desejados, isto é, padrões que o modelo

apresente como resultado em uma simulação, deve ser feita com cuidado para evitar interpretações errôneas e por consequência uma modelagem errada.

Para evitar este tipo de equívoco, são definidos padrões que podem ser observados tanto na curva simulada quanto na curva real, e a partir destes padrões é possível realizar comparações e estimar se as duas curvas possuem um nível de semelhança entre si.

Tendo em vista o tempo computacional necessário para realizar simulações multiagentes, torna-se pouco prático a simulação de uma colônia bacteriana representando cada bactéria através de um agente. E mesmo que não fosse este o caso, como as curvas de crescimento experimentais provenientes do MGIT expressam resultados em unidades de crescimento (*Growth Units*) torna-se necessário um número arbitrário de equivalência, que relaciona o GU com o número de agentes da simulação. Portanto, decidiu-se por usar a relação de um agente para cada cinco unidades de crescimento.

Pela mesma lógica, uma equivalência também é obtida para a grandeza referente ao tempo da simulação. Como a curva experimental demonstra resultados na ordem de dias, e conforme visto anteriormente a simulação trabalha com a unidade relativa de tempo *ticks*, foi usado a equivalência de 100 *ticks* da simulação para cada dia do experimento.

Sendo assim, analisando a Figura 3.6 pode-se, por exemplo, concluir que como auge da população, 2678 agentes, foi atingido no *tick* 1788, esta simulação equivaleria a um experimento que demonstrasse a mesma curva de crescimento e que possuísse pico de população de 13390 GU passados 17,88 dias.

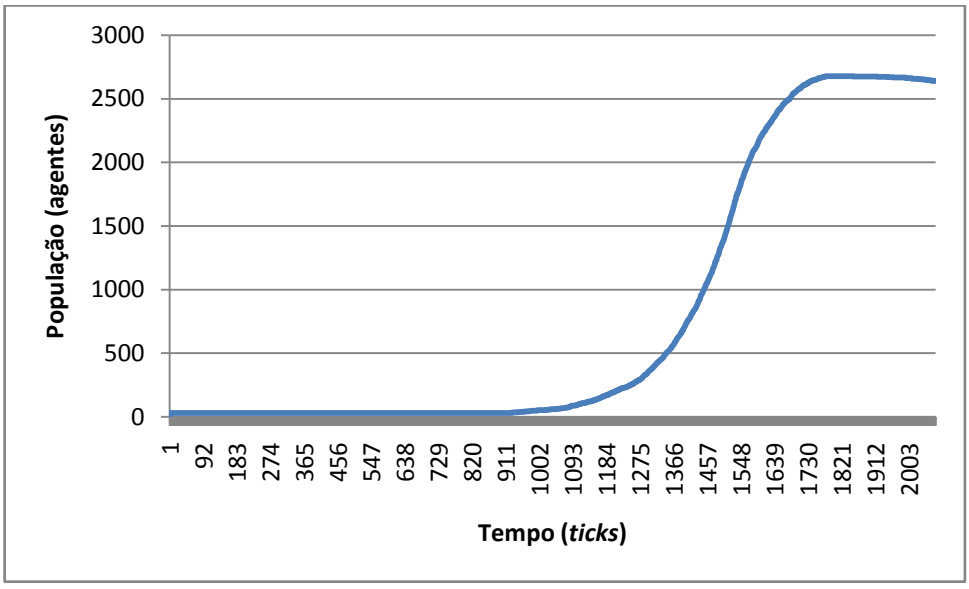


Figura 3.6 - Resultado de uma simulação.

Diante do conhecimento dos três parâmetros principais que ditam a forma da curva de crescimento e a definição de equivalência das unidades dadas como resultado tendo sido feita, deve-se ainda compreender quais os valores de *enconsumo*, *decaiconsumo* e *loglim* das curvas reais.

Embora não hajam valores para estas variáveis nos resultados do MGIT, o resultado que a variação destes três parâmetros causam na curva de crescimento é conhecido. Tendo isto em vista, pode-se fazer um estudo do impacto que a variação destes parâmetros causará na curva, e através da comparação de resultados determinar uma estimação da semelhança que a curva obtida na simulação possui com a curva obtida experimentalmente.

3.3.1. A FUNÇÃO OBJETIVO DO *ENCONSUMO*

O parâmetro *enconsumo* é um dos três parâmetros que modelam a forma da curva de crescimento. Quanto maior seu valor, maior é a taxa de crescimento, isto é, com mais rapidez os agentes irão se reproduzir. Este comportamento pode ser observado na Figura 3.7.

Este comportamento emergente se deve ao fato de que o *enconsumo* é a quantidade percentual de energia da variável *consumo* que o agente usará para manter suas funções vitais. Caso o valor de *enconsumo* seja 0.37, por exemplo, significa que 37% do valor de *consumo* será gasto de energia todos os *ticks* para a manutenção da existência do agente.

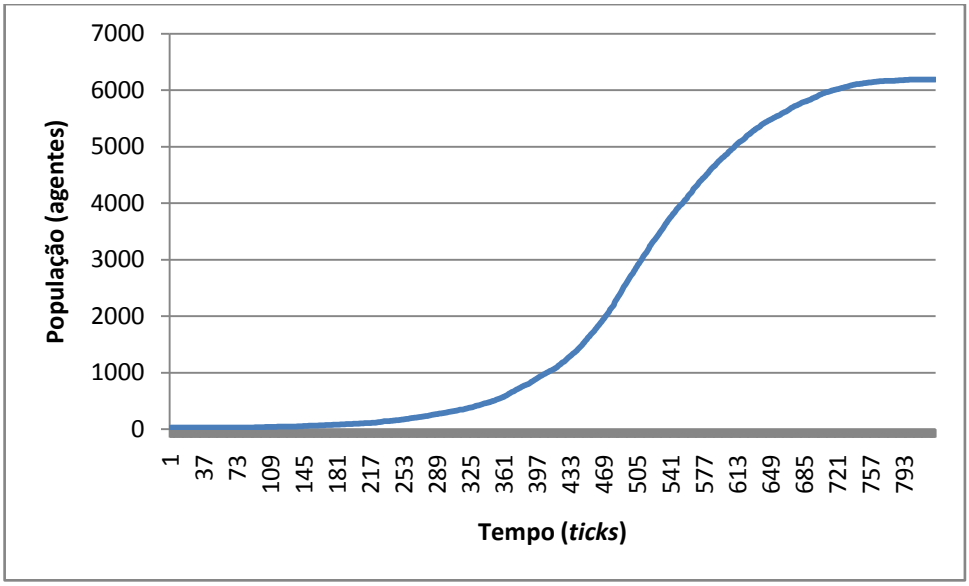
Isto acarreta em uma alteração da taxa de reprodução da população, pois é modelado que cada agente só sofrerá o processo de divisão quando obtiver em sua reserva uma quantidade considerável de energia, no caso 100.

Portanto, quanto mais energia absorvida do ambiente sobra para acumular mais rápido a população vai crescer.

Sabendo isso, cria-se a necessidade de uma medida que expresse o quão rápido a curva está crescendo.

Como a velocidade de crescimento pode ser expressa em tempo, e a unidade de tempo da simulação são os *ticks*, define-se que o tempo que a população levou para crescer é dado diretamente pelo número de *ticks* que se passaram do princípio até o fim da simulação.

(a)



(b)

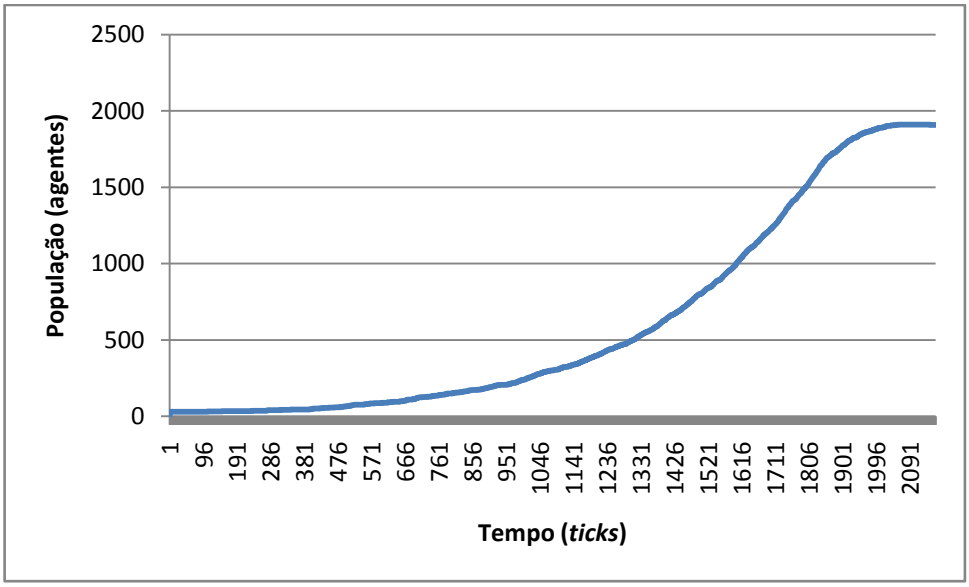


Figura 3.7 - Exemplo de curva de crescimento comparativa para o parâmetro *enconsumo*. Foram usados os valores 0,28 em (a) e 0,83 em (b) para *enconsumo* e valores iguais para todos os outros parâmetros. Nota-se que (b) cresce mais suavemente enquanto o crescimento de (a) é mais rápido.

3.3.2. A FUNÇÃO OBJETIVO DO *DECAICONSUMO*

Outro parâmetro de importância na modelagem da curva de crescimento é o *decaiconsumo*. Este parâmetro define o quão brusco será a queda no crescimento da população uma vez que o a proporção de agentes por moléculas sinalizadoras definida por *loglim* seja atingida, conforme visto na Figura 3.8.

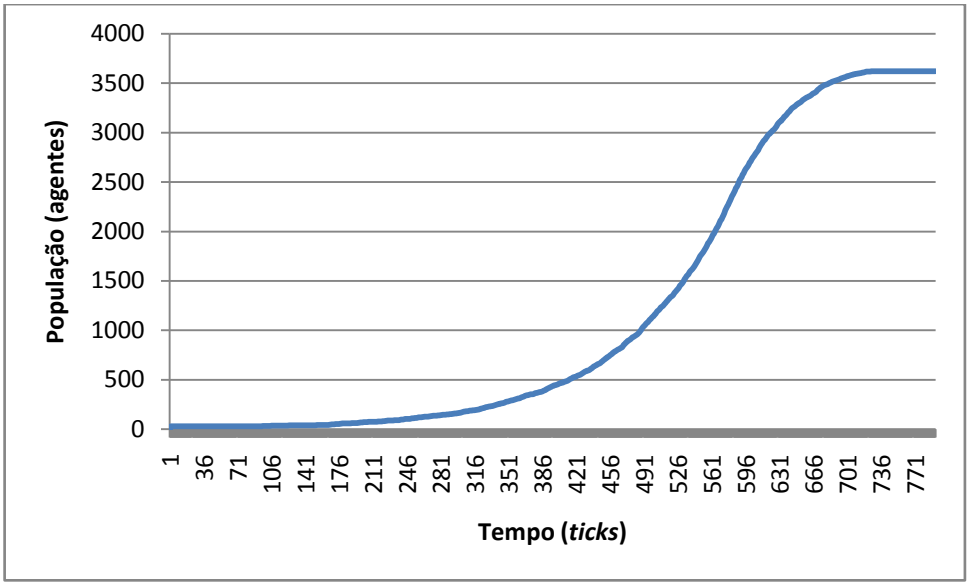
Conforme visto, a variável *consumo* define o quanto o agente consumirá de nutrientes do ambiente e o quanto de sua energia será consumida por *tick*. Quando a proporção de moléculas sinalizadoras por agentes (*loglim)* é atingida, este consumo o agente entra em um estado de consumo reduzido, gradualmente reduzindo o valor da variável *consumo*. A taxa com que este decaimento ocorre é dado por *decaiconsumo*, e quanto maior o valor deste parâmetro mais rapidamente a população de agentes irá estagnar.

Uma vez que se saiba que existe uma diferença no tempo de decaimento entre as curvas, cria-se a necessidade da medição deste tempo para futuras comparações e controle.

Define-se o tempo que a curva levou para decair pela diferença de tempo entre o ponto em que esta curva obteve a maior diferença de população com relação ao instante anterior e o ponto em que a reta atingiu a estagnação.

Embora pelas curvas de crescimento apresentadas anteriormente já fosse visível, pode-se perceber mais claramente através dos gráficos acima que o tempo de decaimento da segunda curva é muito maior que da primeira, ou seja, o número de *ticks* que demora para a primeira curva chegar do ponto máximo de diferença de crescimento populacional até o ponto de estagnação é muito menor que o da segunda curva.

(a)



(b)

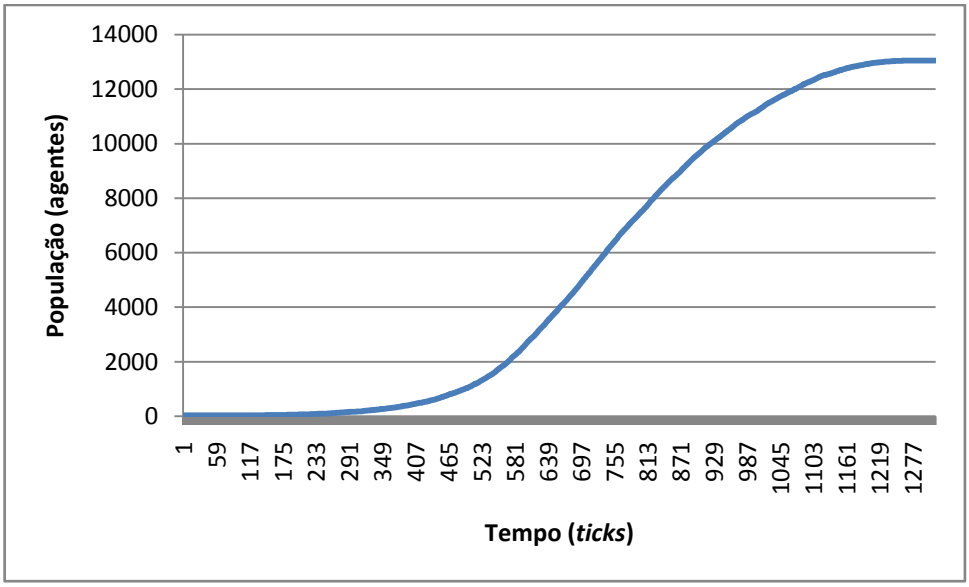
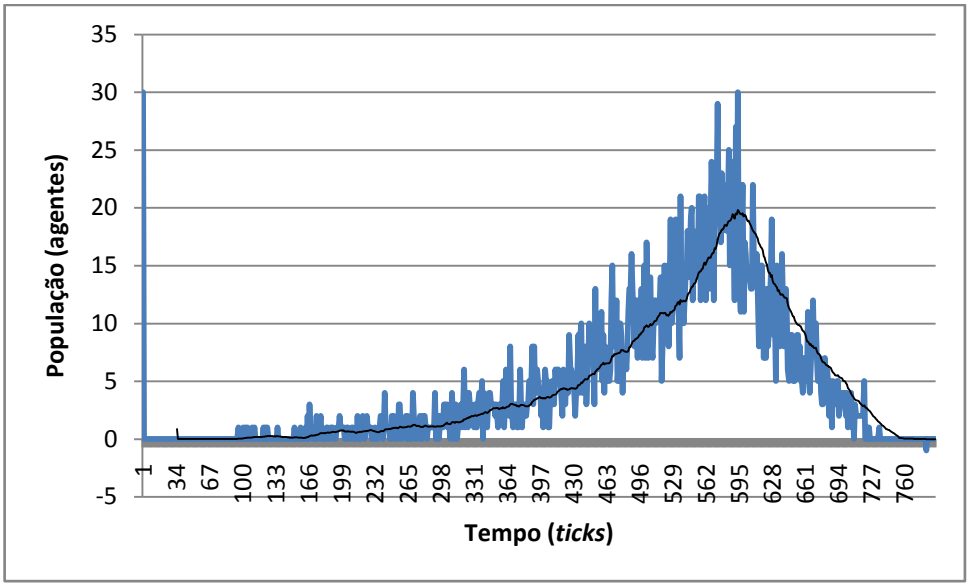


Figura 3.8 - Exemplo de curva de crescimento comparativa para o parâmetro *decaiconsumo*. Foram utilizados os valores 0,0194 em (a) e 0,0044 em (b) para *decaiconsumo* e valores iguais para todos os outros parâmetros. Nota-se que (a) possui uma velocidade de decaimento muito superior a (b). O ponto de transição da etapa de crescimento para etapa de decaimento é muito mais clara (a) do que em (b).

(a)



(b)

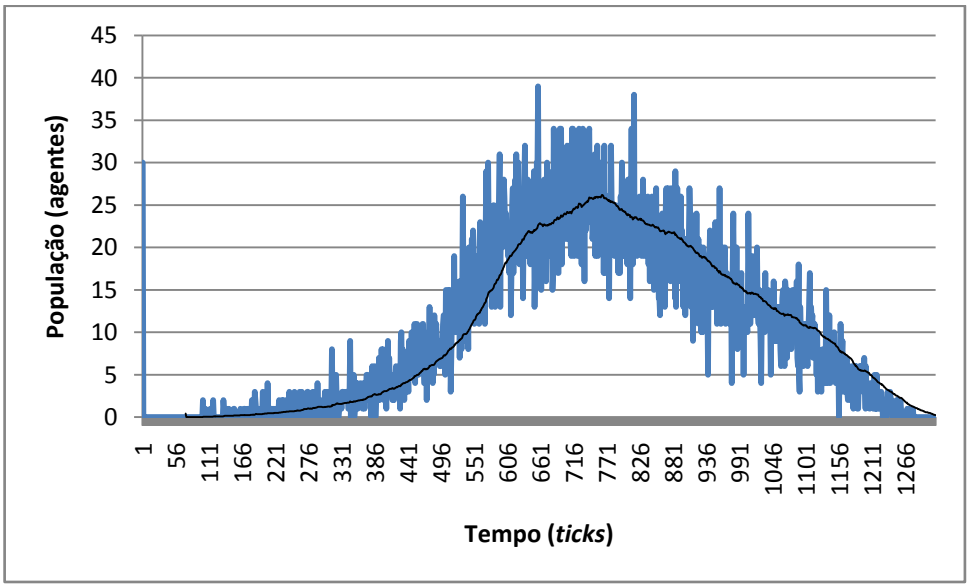


Figura 3.9 - Variação de população em cada *tick* das simulações referentes à Figura 3.8. O item (a) refere-se à variação de população da Figura 3.8 (a), enquanto (b) refere-se à variação de população da Figura 3.8 (b). A linha azul refere-se aos valores absolutos da variação de população, enquanto a linha preta refere-se à linha de tendência da série. É demonstrado que na simulação de (a) a variação de população na etapa de decaimento é mais acentuada do que em (b). O decaimento começa no momento seguinte ao ponto máximo da curva.

3.3.3. A FUNÇÃO OBJETIVO DO *LOGLIM*

Por fim, o *loglim* é o terceiro parâmetro que tem relação direta com a forma da curva de crescimento. O aumento do seu valor faz com que a população máxima da curva aumente.

Conforme visto anteriormente, cada agente possui uma chance dada por *taxasinal* de expelir uma molécula sinalizadora. Esta molécula serve para alertar os agentes sobre o ambiente estar superpopuloso, e caso este alerta não ocorra, os agentes continuarão a se multiplicar e consumir os nutrientes do ambiente desenfreadamente, resultando em uma morte prematura de todos agentes presentes na simulação. Para que isto não ocorra, todos agentes têm a capacidade de entrar em um estado de consumo reduzido, que conforme explicado anteriormente, faz com que este agente consuma menos nutrientes, gaste menos energia, e se reproduza menos. Esta detecção é dada pela proporção de moléculas sinalizadoras pelo número de agentes no ambientes. Uma vez este limiar atingido, os agentes entrarão no estado de consumo reduzido. Este limiar é dado por *loglim*.

Portanto, como *loglim* é o limiar de população que os agentes param de se reproduzir, quanto maior este limiar, mais agentes haverão no ambiente até que eles comecem a consumir menos e a curva comece seu decaimento.

Nota-se na Figura 3.10 que embora as curvas tenham o mesmo formato, cada uma delas possui diferentes pontos máximos onde a população entrou em estagnação.

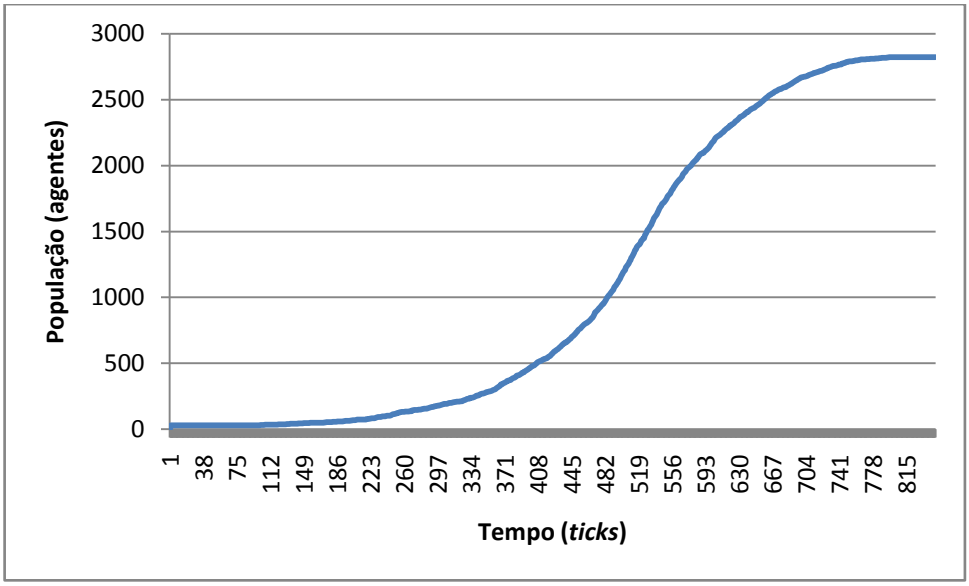
Diferente dos outros dois parâmetros onde a função objetivo era baseada no tempo transcorrido, no parâmetro *loglim* temos por objetivo o máximo de população alcançada em toda a simulação.

3.4. ESTIMAÇÃO DE PARÂMETROS

Uma vez que a simulação foi criada, através dela é possível representar qualquer curva de crescimento, bastando saber os valores a serem preenchidos nos parâmetros *decaiconsumo*, *enconsumo* e *loglim*. Porém, nem sempre é sabe-se quais valores para estes parâmetros resultam na curva desejada. Neste caso é utilizada a estimativa de parâmetros.

Conforme abordado no Capítulo 2, a utilização de métodos numéricos é bastante eficiente para solução de problemas onde pode-se estimar se o resultado está próximo do objetivo e tem-se uma maneira de realizar iterações entre os diferentes parâmetros do ambiente de simulação.

(a)



(b)

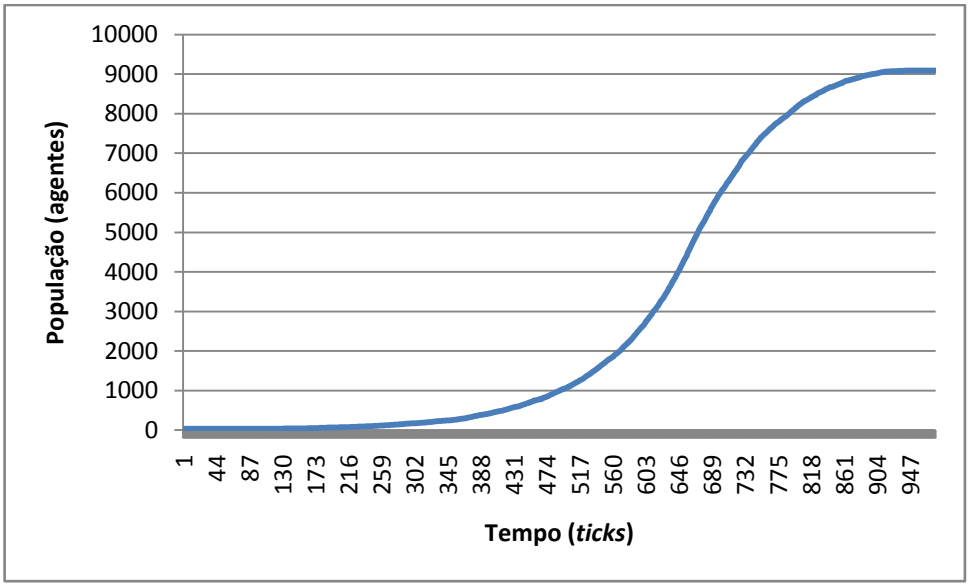


Figura 3.10 - Curva de crescimento comparativa para o parâmetro *loglim*.

Foram utilizados os valores 0,0011 em (a) e 0,0033 em (b) para *loglim* e valores iguais para todos os outros parâmetros. Percebe-se que o topo de população em (a) é muito menor que (b). Este fenômeno se deve ao fato que em (a) a proporção de moléculas sinalizadoras necessárias para fazer a os agentes entrarem no estado de consumo reduzido é menor que em (b), fazendo com que os agentes tenham menos tempo para se reproduzir.

A literatura apresenta diversos exemplos, como visto em (TERANO, 2006), de modelos multiagente que se utilizam de algoritmos evolutivos para encontrar valores de parâmetros para funções objetivo. Porém, optou-se pela utilização de um método numérico para resolver o problema apresentado por principalmente dois fatores, mencionados a seguir:

- O problema proposto possui um função objetivo bem definida, podendo ser expressada matematicamente por uma curva, onde a alteração de valores de parâmetros acarretam em mudanças previsíveis na curva resultante.
- A aplicação de um método numérico demanda um esforço computacional muito menor que a aplicação de um algoritmo evolutivo. E como no presente trabalho o número de agentes presentes simultaneamente na simulação é muito elevado, é de vital importância que se utilize um algoritmo de otimização que não demande muitos recursos computacionais.

No caso trabalhado, o método numérico é aplicado pois o objetivo da simulação é conhecido, onde deseja-se encontrar na simulação os valores de crescimento, decaimento e topo de população iguais, ou pelo menos consideravelmente próximos aos mesmos valores obtidos na curva obtida experimentalmente.

Para usar a ferramenta numérica escolhida, o software criado para representar o modelo proposto possui alguns parâmetros que são necessários para a aplicação do método numérico:

- **cregoal** - Define quantos *ticks* a simulação resultante deverá possuir para ser considerada igual à curva experimental. Este parâmetro caracteriza o tempo de crescimento da simulação.
- **decgoal** - Define o tempo, em *ticks*, que a simulação levará do instante em que todos os agentes estiverem entrado na fase de consumo reduzido até a simulação entrar na etapa de estagnação.
- **popgoal** - Define o número máximo de agentes que a simulação atingirá em seu auge.
- **errocre** - Tempo máximo de diferença de crescimento que a simulação poderá ter em relação à curva real dada como objetivo.

- **errodec** -Tempo máximo de diferença de decaimento que a simulação poderá ter em relação à curva real dada como objetivo
- **erropop** - Diferença máxima entre o número de agentes máximos obtidos na simulação em relação à curva real dada como objetivo.
- **iteracoes** - Número simulações que serão usadas como universo da base de dados para calcular a média dos parâmetros obtidos. As comparações de *cregoal*, *decgoal*, e *popgoal* são realizadas com base nas médias de diversas simulações. O número de simulações usadas é definido neste parâmetro.
- **errogeral** - Define o quão a curva gerada na simulação pode estar distante da curva dada como objetivo para ser considerada suficientemente semelhante. Uma vez cada um dos três parâmetros tenha sido encontrado com base nos seus respectivos erros, a curva final é verificada como um todo em comparação à curva objetivo.

Como já foram definidas as equivalências das unidades e a maneira de mesurar cada um dos parâmetros de curvas dadas como resultado de uma simulação, é possível encontrar os parâmetros que resultam em uma curva muito próxima a uma curva dada como objetivo através de sucessivas simulações.

No modelo proposto neste trabalho, uma vez que seja escolhida uma curva experimental para ser usada como objetivo, basta calcular qual o tempo que esta curva leva para crescer, qual o tempo de decaimento dela, e qual o máximo de população que esta curva atingiu utilizando as métricas previamente discutidas.

Uma vez encontrados estes valores, deve-se convertê-los para *ticks* e número de agentes, e então definir estes valores como o objetivo de simulação. Estes valores convertidos das curvas experimentais serão definidos como valores objetivo uma vez que sejam atribuídos aos parâmetros *cregoal, degoal* e *popgoal.*

Os parâmetros *cregoal*, *decgol* e *popgoal* definem qual o valor de crescimento, decaimento e topo de população que a simulação deverá atingir por objetivo. Uma vez que as simulações resultem em valores muito próximos a estes parâmetros, a estimativa de parâmetros encerra. Enquanto isso não ocorrer, novas simulações são realizadas.

Uma vez definidos os objetivos de crescimento, decaimento e topo de população, *cregoal*, *decgoal* e *popgoal*, respectivamente, deve-se definir os valores dos erros aceitáveis. Ao escolher os valores de *errocre, errodec* e *erropop* está sendo informado para a simulação

uma importante informação. Estas variáveis de erro carregam a informação que ditará quando que uma simulação alcançou um máximo/mínimo local.

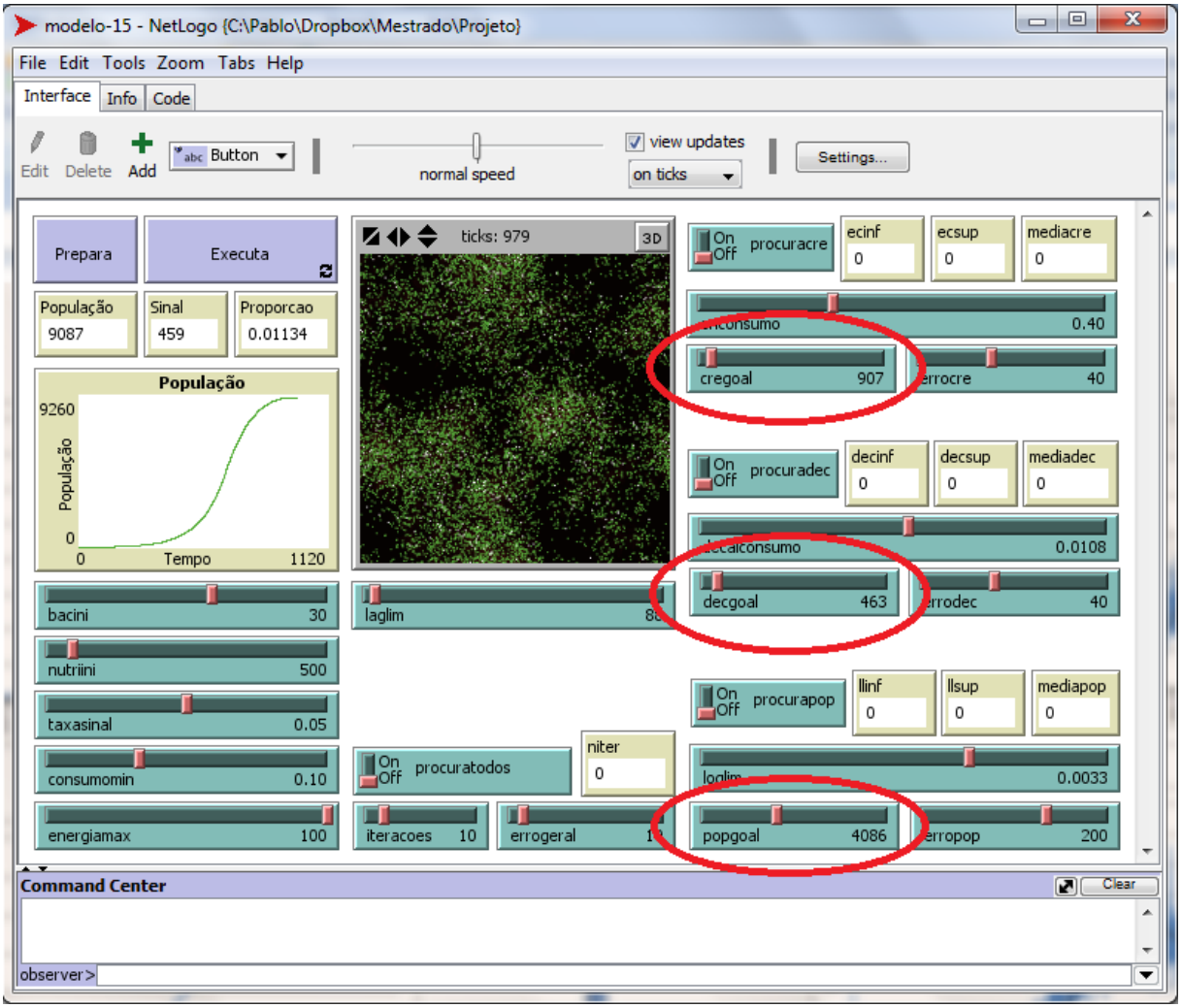


Figura 3.11 - Captura de tela da simulação.

Destacam-se *cregoal*, *decgoal* e *popgoal*, parâmetros que definem o objetivo do crescimento, decaimento e população, respectivamente.

Uma vez que uma simulação é concluída, são calculados os valores resultantes de crescimento, decaimento e topo de população e então estes valores são comparados aos valores obtidos na simulação anterior.

Caso a diferença entre o resultado obtido anteriormente e o resultado atual for maior que as respectivas variáveis de erro, significa que a margem de erro definida a simulação ainda não convergiu para os valores obtidos em *enconsumo*, *decaiconsumo* e *loglim*, e portanto os valores dos parâmetros precisam ser refinados.

Para realizar o refinamento, o método numérico é aplicado, e os valores atuais de *enconsumo, decaiconsumo* e *loglim* são modificados dependendo do resultado do crescimento, decaimento e topo de população obtido na simulação.

Inicialmente os limites de busca para os valores são os apresentados na Tabela 3.2:

Tabela 3.2 - Limites inferiores e superiores de refinamento dos parâmetros *enconsumo, decaiconsumo* e *loglim*.

Variável	**Limite inferior**	**Limite superior**
enconsumo	0,1	0,9
decaiconsumo	0,001	0,02
loglim	0,001	0,05

Caso o resultado da simulação possua o tempo crescimento maior que *cregoal*, a próxima simulação terá seu valor de *enconsumo* ajustado para ser o ponto médio entre o valor atual e o valor do limite inferior (pois o novo *enconsumo* deve resultar em um crescimento mais rápido), bem como o limite superior é ajustado para receber o valor atual de *enconsumo.* Caso contrário, o novo valor de *enconsmo* será o ponto médio entre o valor atual e o limite superior, com como limite inferior é ajustado para receber o valor atual de *enconsumo*.

Desta maneira há uma garantia de que a cada vez que a simulação chega no fim, e os parâmetros são refinados, o espaço de busca é reduzido pela metade.

A mesma lógica se aplica para os parâmetros *decaiconsumo* e *loglim*. Caso o valor do resultado simulado de decaimento seja superior a *decgoal*, o novo valor de *decaiconsumo* será ajustado para o ponto médio entre o valor atual e o seu limite superior (pois o novo *decaiconsumo* deverá resultar em um decaimento mais rápido) enquanto o limite inferior recebe o valor atual de *decaiconusmo*. Caso o valor simulado seja menor, o novo valor de *decaiconsumo* será ajustado para o ponto médio entre o valor atual e o limite inferior.

Quanto ao parâmetro *loglim*, o novo valor será ajustado para o lado do limite inferior quando o resultado do topo de população da simulação for maior que o valor de *popgoal*, e para o lado do limite superior quando for menor.

A Figura 3.12 demonstra a lógica utilizada na aplicação do método numérico na escolha dos parâmetros.

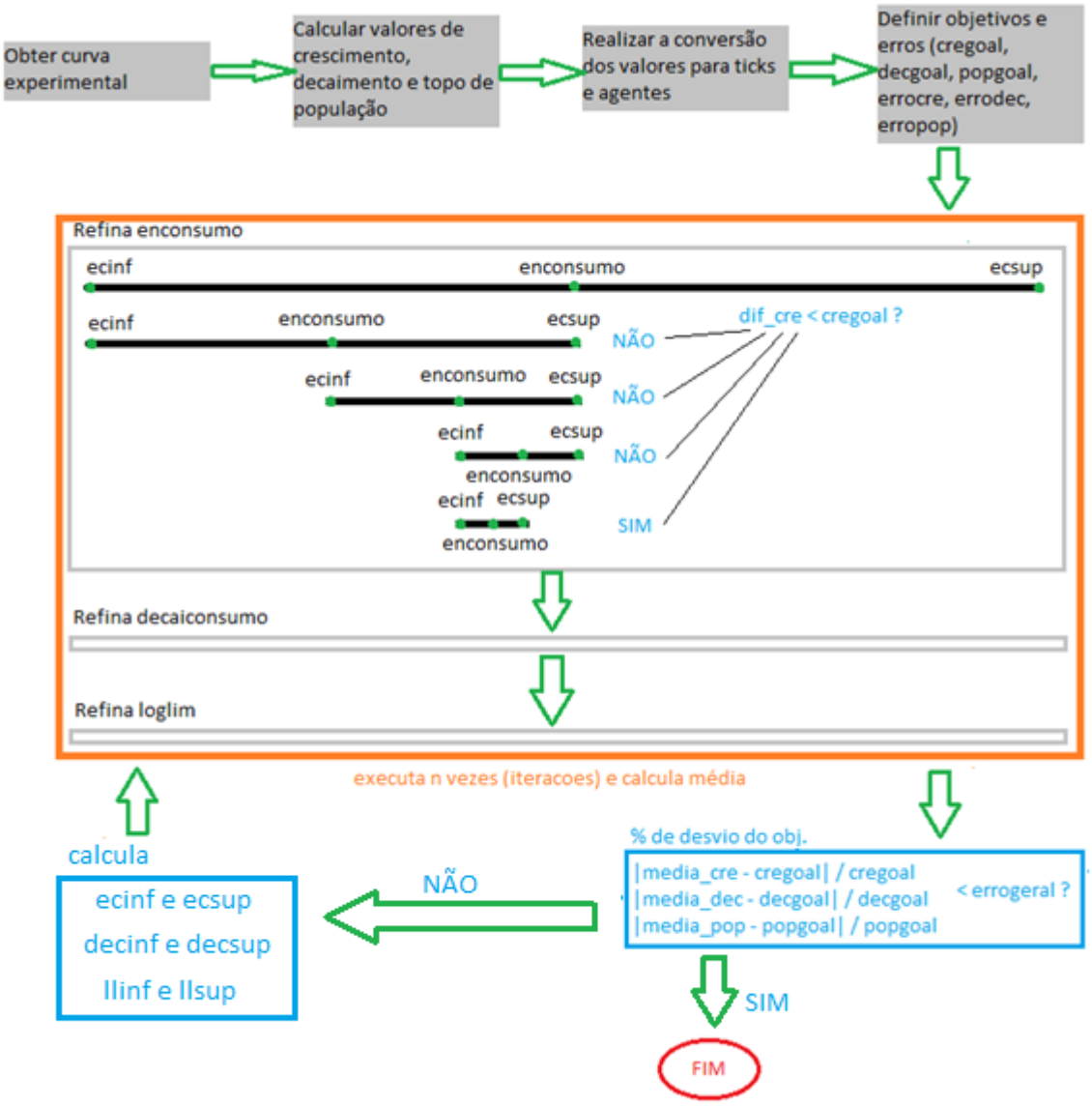


Figura 3.12 - Etapas do processo de estimativa de parâmetros

Após a convergência dos parâmetros, o valor de *enconsumo*, *decaiconsumo* e *loglim* são acumulados e então o processo de refinamento é iniciado novamente. Esta etapa se repete um número de vezes referente ao valor do parâmetro *iterações*. Após todos os valores convergentes serem calculados e acumulados a média deles é realizada para evitar que mínimos locais sejam considerados como os resultados finais.

Quando as três médias tiverem sido calculadas, então é verificado se os valores de crescimento, decaimento e topo de população estão próximos da curva objetivo. Caso a diferença percentual seja menor que o valor de *errogeral* então a simulação termina e os valores dos parâmetros *enconsumo*, *decaiconsumo* e *loglim* são os valores finais encontrados.

Caso a diferença percentual dos parâmetros encontrados seja maior que *errogeral* então novas simulações são feitas. Os valores de limite inferior e superior de varredura dos parâmetros a ser encontrados (*llinf*, *llsup*, *ecinf*, *ecsup*, *decinf* e *decsup*) na nova simulação não deverão começar dos valores mínimos e máximos na nova simulação. Estes valores irão variar de acordo com o quão próximo a curva resultante da última simulação chegou da curva objetivo, ou seja, o quão próximo os valores de crescimento, decaimento e topo de população chegaram dos valores objetivo *cregoal*, *decgoal* e *popgoal*, respectivamente.

Os novos limites serão calculados com base na porcentagem de desvio que a curva obtida teve da curva objetivo. O novo valor de limite inferior será dado pelo valor do respectivo parâmetro que está se tentando obter (*enconsumo*, *decaiconsumo* ou *loglim*) subtraído de um valor, enquanto o novo limite superior será uma adição.

O valor a ser subtraído ou adicionado é calculado analisando o maior valor e o menor valor que o parâmetro poderá assumir (na prática, o início e o fim do valor indicado no respectivo *slider*). O valor será igual a porcentagem do erro obtido multiplicado pela diferença da possível variação do *slider.* Um exemplo para melhor compreensão é demonstrado a na Figura 3.13.

```
popgoal = 1024
loglim = 0.00453
erro obtido = 4.5%
valor mínimo do slider loglim = 0.001
valor máximo do slider loglim = 0.05

variação do slider = 0.05 - 0.001 = 0.049
erro obtido x variação = 0.045 x 0.049 = 0.002205

novo valor de llinf = 0.00453 - 0.002205 = 0.002325
novo valor de llsup = 0.00453 + 0.002205 = 0.006735
```

Figura 3.13 - Exemplo de variação do limite inferior e superior de um *slider* após o término de uma simulação.

Através deste mecanismo, é garantido que cada vez que uma nova simulação é iniciada, os valores a ser buscados serão cada vez mais próximos dos valores que levarão à curva objetivo. Caso a curva resultante se afaste da curva objetivo, o erro será maior e por consequência a nova simulação terá um espaço maior de busca.

4. RESULTADOS OBTIDOS

O modelo proposto foi submetido a testes a fim de verificar sua eficiência. A validação dos resultados obtidos pelo modelo possuem duas etapas, descritas a seguir:

- Verificação do modelo como ferramenta capaz de reproduzir curvas de crescimento bacteriano, reproduzindo curvas obtidas experimentalmente *in vitro* através da modelagem de parâmetros que refletem características de agentes.

- Validação do processo de estimação de parâmetros, que através de uma função objetivo verificada nas curvas reais, tenta obter automaticamente os valores dos parâmetro da simulação que resultam na curva real.

Tendo em vista que a proposta do modelo é abordar tanto a modelagem do sistema multiagente quanto o mecanismo de estimação de parâmetros, os resultados de ambas etapas são apresentados a seguir.

4.1. RESULTADOS DA MODELAGEM DO AMBIENTE

Conforme apresentado em 3.3 - Interpretação de valores, as cinco cepas obtidas experimentalmente foram utilizadas como referência para comparação com as curvas geradas na simulação. Foram realizadas 10 simulações de cada uma das populações de agentes e o resultado apresentado é composto da média destas simulações.

Para a realização da calibragem da simulação com as curvas de crescimento reais foram testados diferentes valores em todos os parâmetros da simulação. Algumas variáveis dos agentes provaram ter um grau de sensibilidade maior que outras no que diz respeito aos resultados observados na curva de crescimento resultante, ao mesmo tempo que outras variáveis mostraram ter um papel bem claro e distinto na simulação, como na inclinação da curva ou no topo de população.

A Tabela 4.1 apresenta parâmetros que são comuns a todas as populações.

Tabela 4.1 - Parâmetros com valores em comum para todas populações de agentes.

Variável	**Descrição**	**Valor**
bacini	Número inicial de agentes	30 agentes
nutriini	Nutrientes em cada *patch*	100 unidades
laglim	Tempo de adaptação (fase lag)	250 *ticks*
consumomin	Mínimo consumo possível	0.1 unidades
taxasinal	Chance do agente produzir uma molécula sinalizadora	0.05 (5% por *tick*)
decaiconsumo	Taxa de redução do consumo gradual no estado de consumo reduzido	0.03 (3% por *tick*)

Além dos parâmetros gerais, que são compartilhados por todas as populações, existem os específicos, que são os parâmetros que darão as características diferenciadas para cada população, permitindo a reprodução aproximada das cepas. Os valores utilizados são apresentados na Tabela 4.2.

Tabela 4.2 - Parâmetros com valores variáveis entre as populações de agentes.

Variável	**Descrição**	**GC 03-0850**	**GC 02-2761**	**GC 01-2522**	**GC 03-2922**	**H37Rv**
loglim	Limiar onde os agentes entrarão no estado de consumo reduzido	0.02	0.02	0.018	0.018	0.017
enconsumo	Quantidade de energia gasta por *tick* para manutenção das funções vitais do agente	0.01	0.01	0.03	0.03	0.04

Diante dos resultados obtidos e apresentados nas Figura 4.1 e Figura 4.2, pode-se concluir sobre o comportamento resultante que a simulação irá apresentar diante da variação dos valores de algumas variáveis do modelo. Foram destacadas três variáveis especificamente

que quando alteradas, produzem um efeito bastante visível e destacável na curva de crescimento simulada. Estes três parâmetros são o *enconsumo*, *decaiconsumo* e *loglim*.

A alteração do valor de *decaiconsumo* faz com que o fim da curva de crescimento resultante se torne mais ou menos drástico. Um valor alto para esta variável fará com que os agentes entrem em estado de estagnação mais rapidamente. Isto acontece porque uma vez no estado de consumo reduzido, os agentes começarão a reduzir a quantidade de nutrientes absorvida a cada *tick* até o limite dado por *consumomin*, fazendo com que eles se reproduzam cada vez menos até entrarem em um estado de estagnação, conhecido como fase estacionária do crescimento.

O valor da variável *loglim* determinará o tamanho final da população. Este fato ocorre porque, uma vez que os agentes alcancem uma proporção de população por quantidade de moléculas sinalizadoras no ambiente igual ao valor de *loglim*, eles entrarão no estado de consumo reduzido, fazendo com que eventualmente entrem na fase estacionária. Então pode-se concluir que quanto menor o valor de *loglim*, menos agentes serão necessários para este limiar ser alcançado, logo, menor a população final destes agentes.

A variável *enconsumo* é diretamente relacionada à inclinação da curva de crescimento dos agentes, pois sendo esta a variável que representa a quantidade de energia necessária para manter as funções básicas dos agentes, quanto maior for este valor, menos energia por *tick* restará para reprodução, o que irá afetar a velocidade com que a população cresce.

As diversas tentativas de ajuste dos parâmetros permitiram que o comportamento das curvas de crescimento reais pudessem ser reproduzidas com um nível de semelhança relativamente próximo ao esperado.

Na Figura 4.1- (b) pode-se observar a curva de crescimento gerada a partir dos dados experimentais obtidos em *in vitro* contendo 5 cepas. Enquanto na Figura 4.1 - (a) observa-se a curva de crescimento gerada a partir da média entre 10 simulações feitas no modelo em desenvolvimento. Cada simulação utilizou 5 populações de agentes com diferentes parâmetros calibrados para resultarem em uma curva de crescimento semelhante à vista na Figura 4.1 - (b).

Na Figura 4.2 a curva de crescimento de cada uma das 5 cepas foi representada em um gráfico diferente, juntamente com a curva de crescimento da simulação que representa a respectiva cepa. Deste modo, pode-se comparar para cada uma das cepas, o quão próximo o resultado da curva simulada está da curva real.

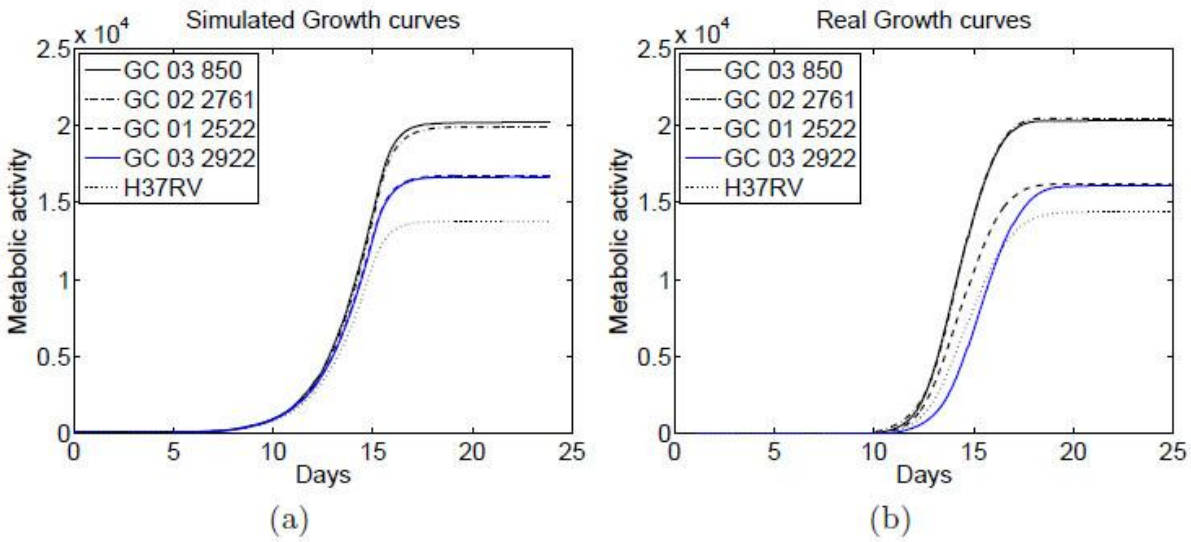


Figura 4.1 - Curvas simuladas (a) e reais (b) de todas as cepas e populações de agentes. (WERLANG, FAGUNDES, *et al.*, 2013)

Para todos os cinco gráficos da Figura 4.2 foram usados uma linha pontilhada para representar a curva real e uma linha contínua para representar a média das curvas simuladas. A área cinza ao redor da curva simulada representa o desvio padrão das 10 simulações.

A partir do modelo desenvolvido, o artigo intitulado "Multi-Agent-Based Simulation of *Mycobacterium tuberculosis* growth" que foi submetido e aprovado no 14th International Workshop on Multi-Agent-Based Simulation (MABS) (WERLANG, FAGUNDES, *et al.*, 2013).

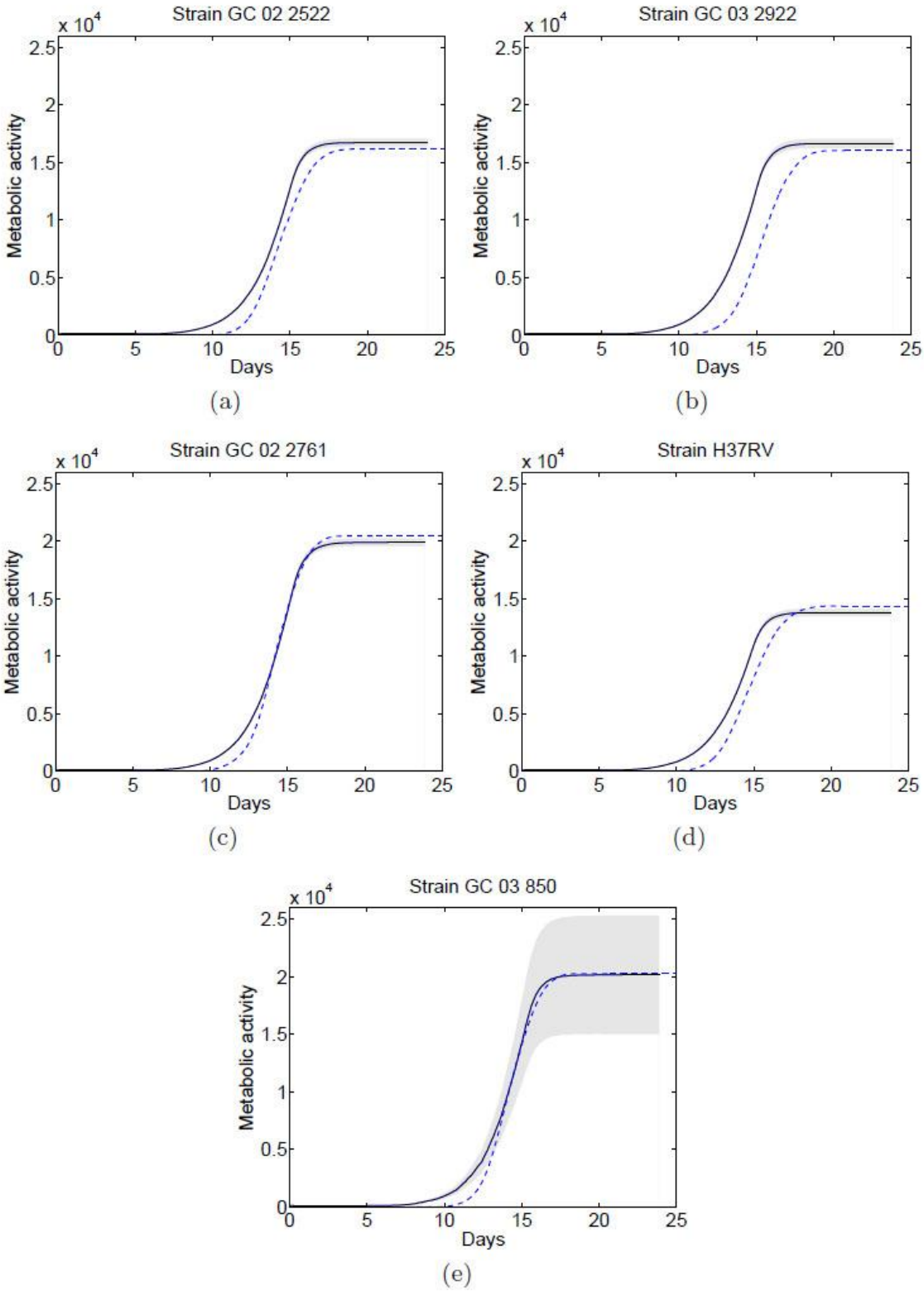


Figura 4.2 - Comparação entre curva real e simulada. Cada gráfico representa uma população com a sua respectiva cepa. A linha pontilhada representa a curva real, a linha contínua representa a média das simulações, e a área cinza o desvio padrão das 10 simulações. (WERLANG, FAGUNDES, *et al.*, 2013)

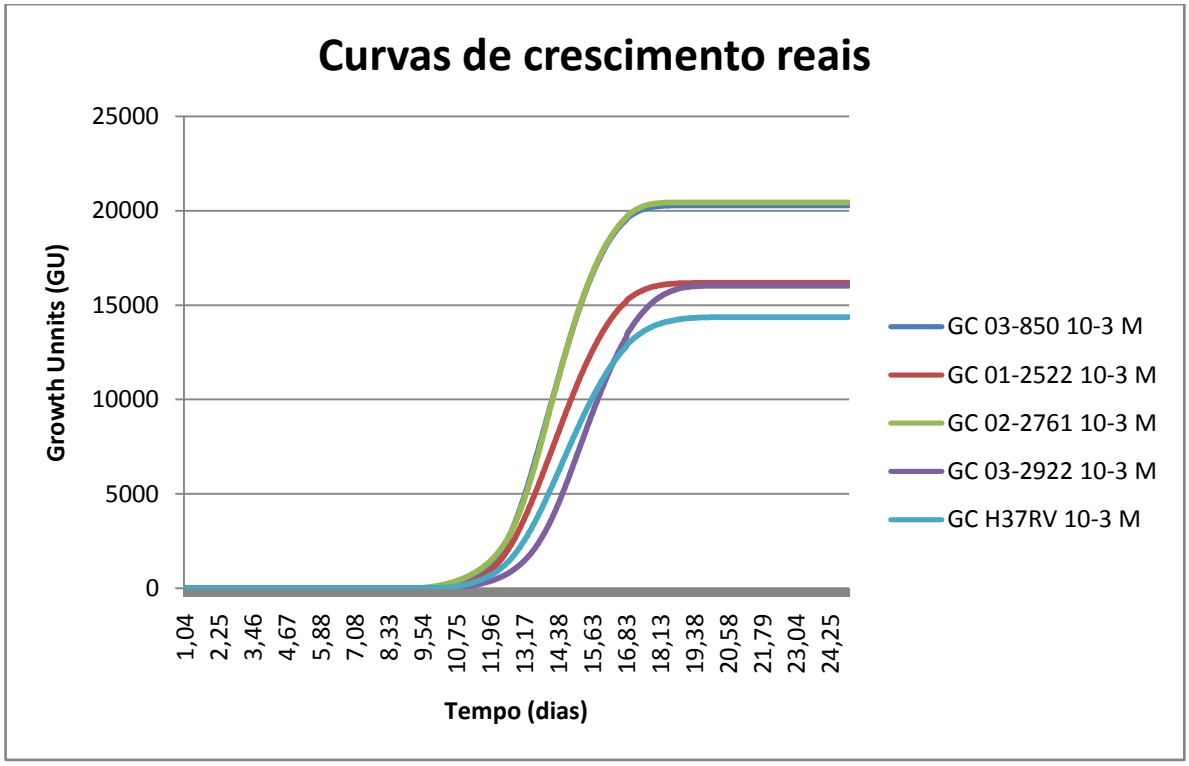


Figura 4.3 - Curvas de crescimento obtidas experimentalmente.

4.1. RESULTADOS DO MECANISMO DE ESTIMATIVA DE PARÂMETROS

Após a obtenção dos resultados preliminares, o mecanismo de estimação de parâmetros, apresentado em 3.4 - Estimação de parâmetros, foi implementado e submetido a testes para verificar sua validade. Nesta etapa, foram usadas as mesmas curvas de crescimento que serviram de base para validação dos resultados apresentados em 4.1 - Resultados da modelagem do ambiente.

Através da análise dos dados das curvas apresentadas na Figura 4.3, pôde-se medir os valores das características das curvas. Estes valores são apresentados na Tabela 4.3.

Os conceitos envolvidos na obtenção dos valores de lag, crescimento, decaimento e população máxima foram abordados em 3.3.1, 3.3.2, 3.3.3.

As características consideradas objetivo para as curvas simuladas foram obtidas e apresentadas na Tabela 4.3.

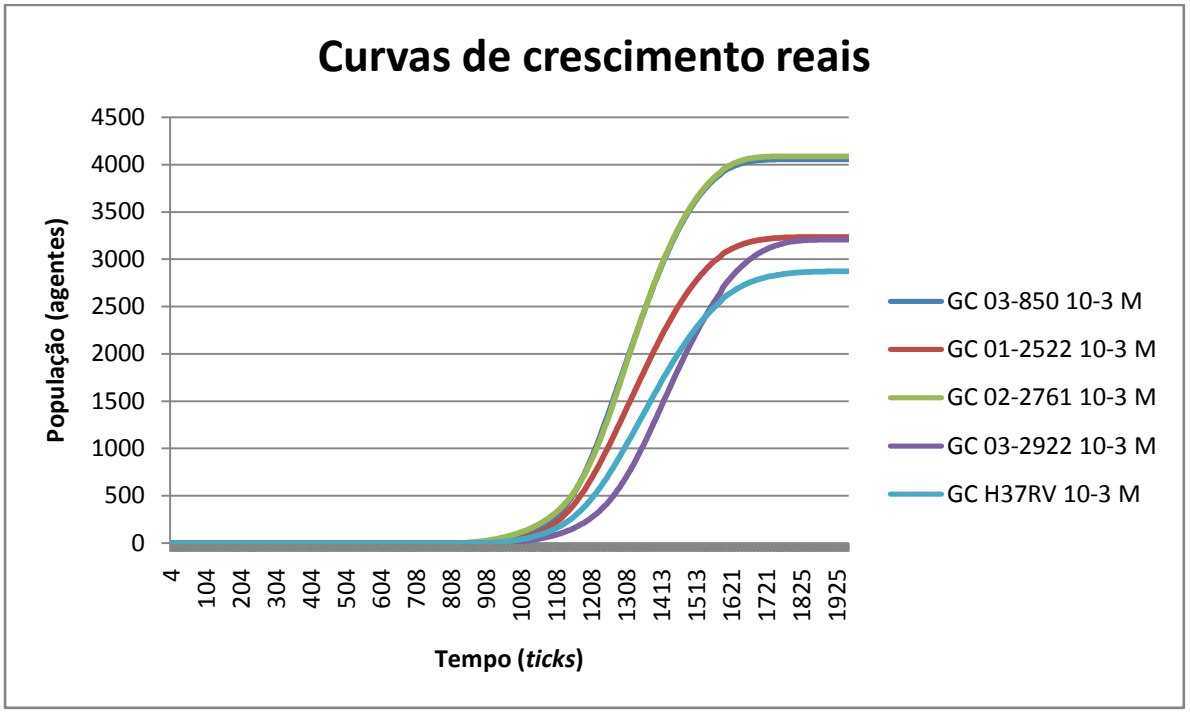


Figura 4.4 - Curvas de crescimento da Figura 4.3 representadas através dos valores de população e tempo equivalentes às unidades utilizadas no modelo.

Tabela 4.3 - Tempo de lag, crescimento, decaimento e topo de população atingido pelas curvas experimentais das cinco cepas analisadas.
O tempo está apresentado em *ticks* e a população em número de agentes, conforme equivalência adotada e discutida em 3.3 - Interpretação de valores.

	GC 03-850 10-3 M	**GC 01-2522 10-3 M**	**GC 02-2761 10-3 M**	**GC 03-2922 10-3 M**	**GC H37RV 10-3 M**
Lag	833	875	817	908	871
Crescimento	959	979	937	1000	1083
Decaimento	479	562	454	491	591
População	4056	3236	4087	3206	2872

A partir dos dados que foram atribuídos aos parâmetros *laglim*, *cregoal*, *decgoal* e *loglim*, os valores de *enconsumo*, *decaiconsumo* e *loglim* foram obtidos para cada curva através do algoritmo de estimativa de parâmetros discutido em 3.4 - Estimação de parâmetros

utilizando uma tolerância à erros de 3% (valor do parâmetro *errogeral*). Os valores resultantes das iterações são apresentados na Tabela 4.4.

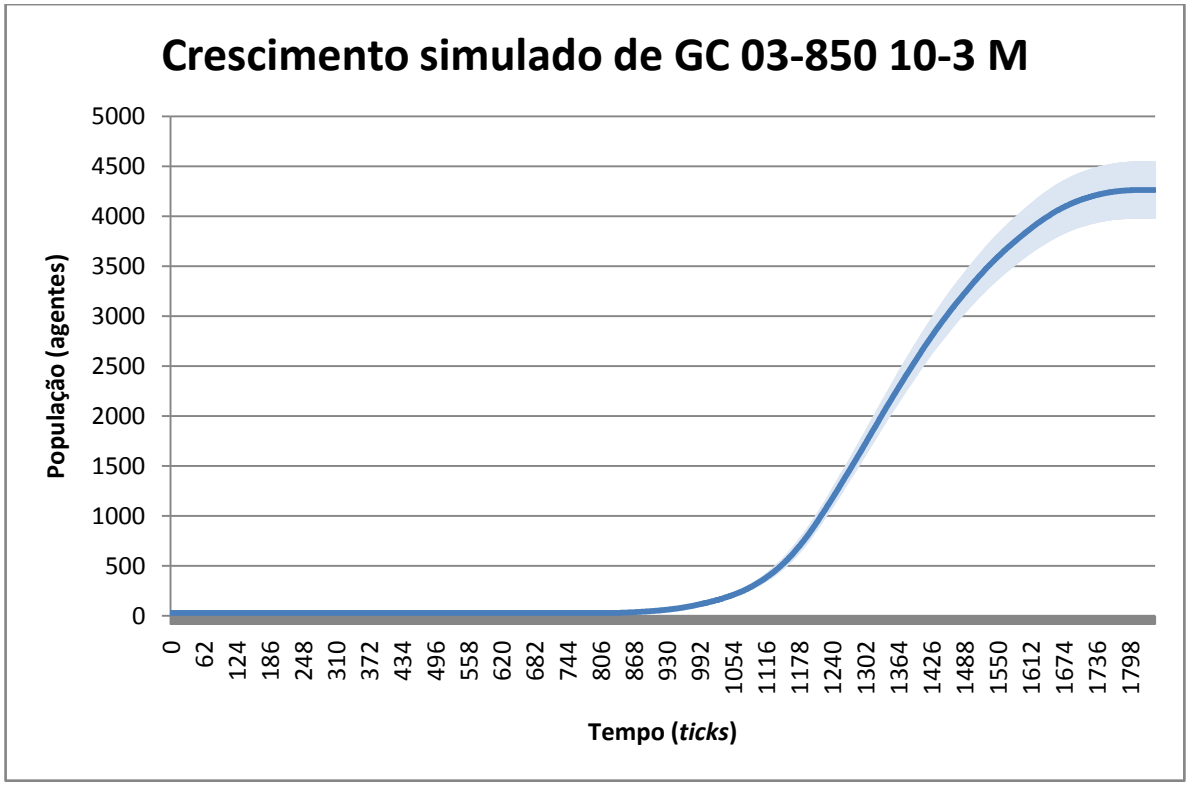
Tabela 4.4 - Valores dos parâmetros *enconsumo*, *decaiconsumo* e *loglim* obtidos através da estimativa de parâmetros

As curvas de crescimento apresentadas são referentes às cepas GC 03-850 10-3 M, GC 01-2522 10-3 M, GC 02-2761 10-3 M, GC 03-2922 10-3 M e GC H37RV 10-3 M.

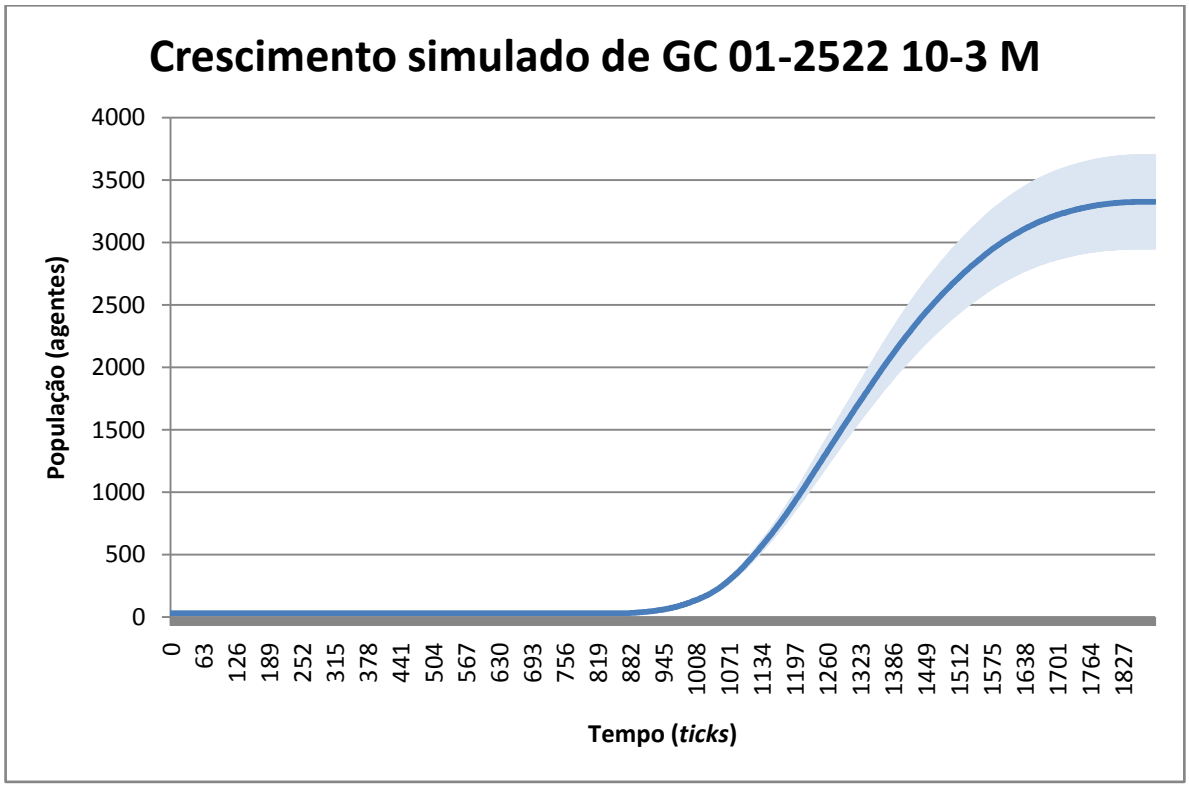
Cepa	***enconsumo***	***decaiconsumo***	***loglim***
GC 03-850 10-3 M	0.3806204379	0.0050882290	0.0044603745
GC 01-2522 10-3 M	0.1925366385	0.0047286468	0.0011570323
GC 02-2761 10-3 M	0.4641542156	0.0060620457	0.0085007324
GC 03-2922 10-3 M	0.4509999999	0.0049939664	0.0047623291
GC H37RV 10-3 M	0.5016147276	0.0040455557	0.0038394029

A fim de verificar a validade dos valores dos parâmetros encontrados, foram realizadas 50 simulações para cada uma das cinco populações utilizando os dados apresentados na Tabela 4.4 como entrada.

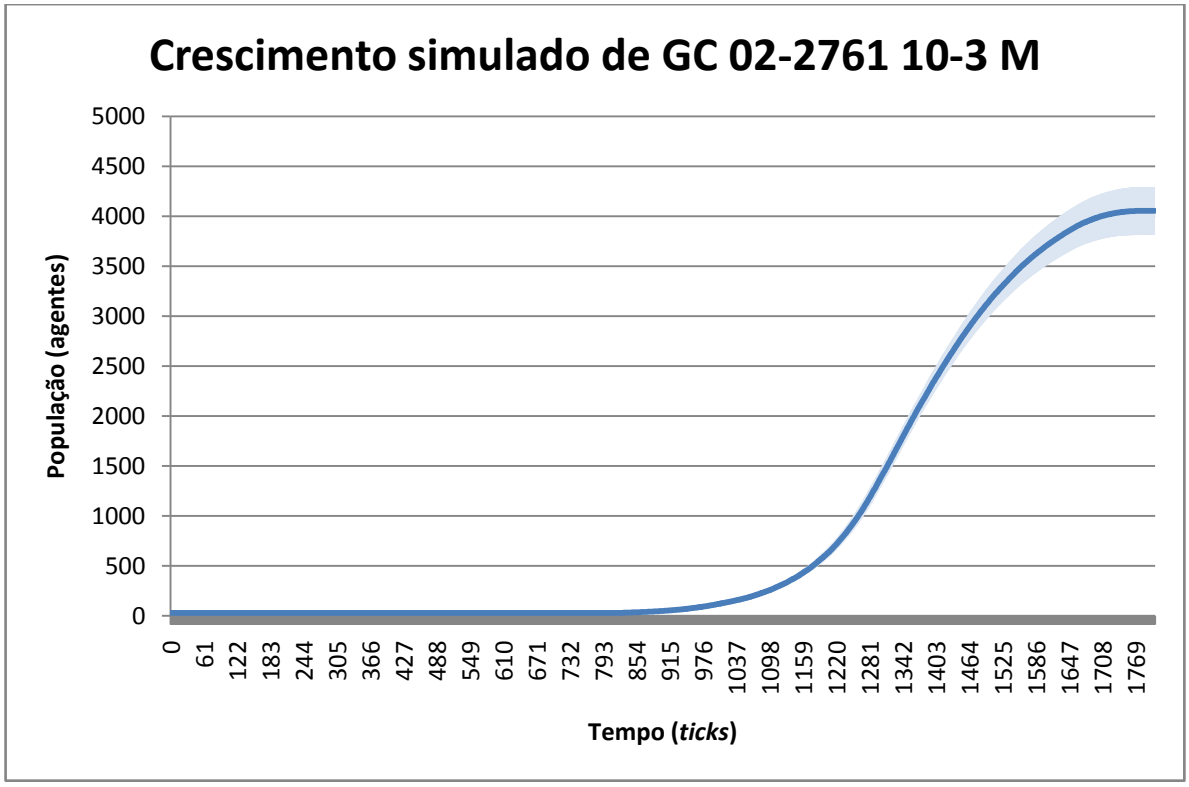
(a)


Crescimento simulado de GC 03-850 10-3 M
População (agentes)
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
62
124
186
248
310
372
434
496
558
620
682
744
806
868
930
992
1054
1116
1178
1240
1302
1364
1426
1488
1550
1612
1674
1736
1798
Tempo (ticks)

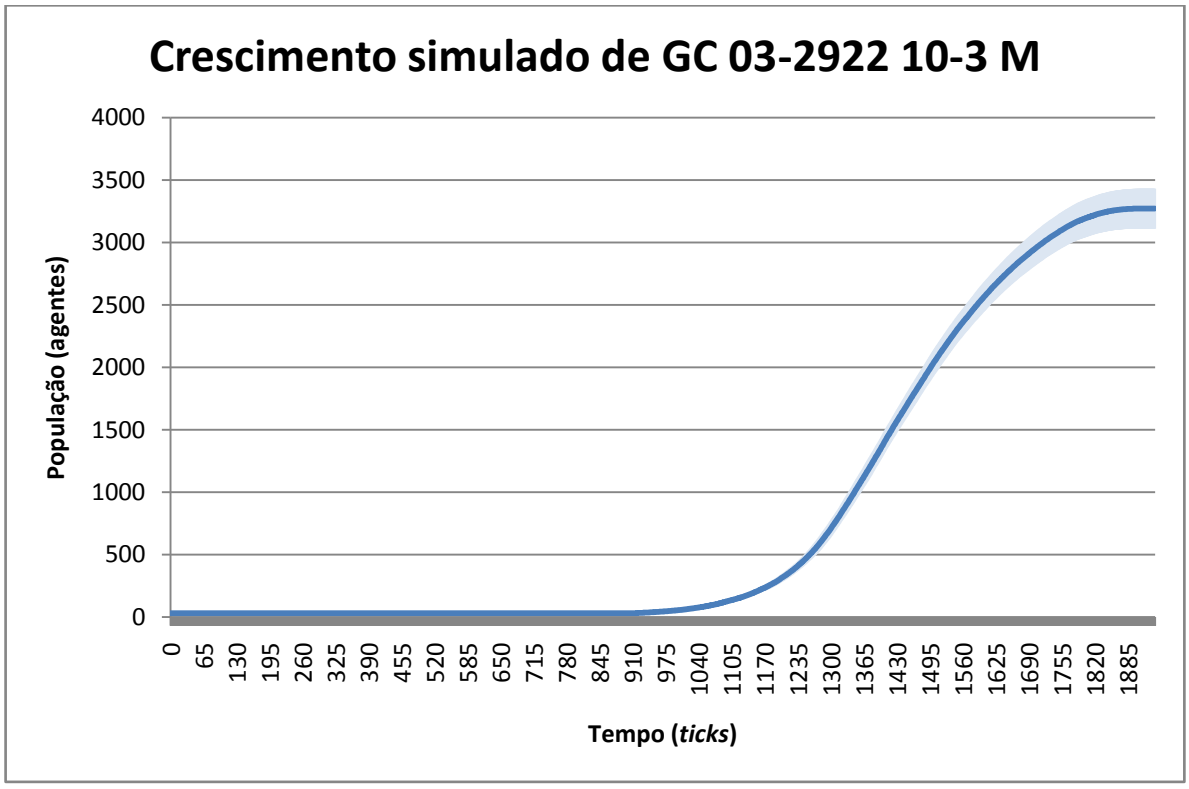
(b)


Crescimento simulado de GC 01-2522 10-3 M
População (agentes)
4000
3500
3000
2500
2000
1500
1000
500
0
0
63
126
189
252
315
378
441
504
567
630
693
756
819
882
945
1008
1071
1134
1197
1260
1323
1386
1449
1512
1575
1638
1701
1764
1827
Tempo (ticks)

(c)


Crescimento simulado de GC 02-2761 10-3 M
População (agentes)
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
61
122
183
244
305
366
427
488
549
610
671
732
793
854
915
976
1037
1098
1159
1220
1281
1342
1403
1464
1525
1586
1647
1708
1769
Tempo (ticks)

(d)


Crescimento simulado de GC 03-2922 10-3 M
População (agentes)
4000
3500
3000
2500
2000
1500
1000
500
0
0
65
130
195
260
325
390
455
520
585
650
715
780
845
910
975
1040
1105
1170
1235
1300
1365
1430
1495
1560
1625
1690
1755
1820
1885
Tempo (ticks)

(e)

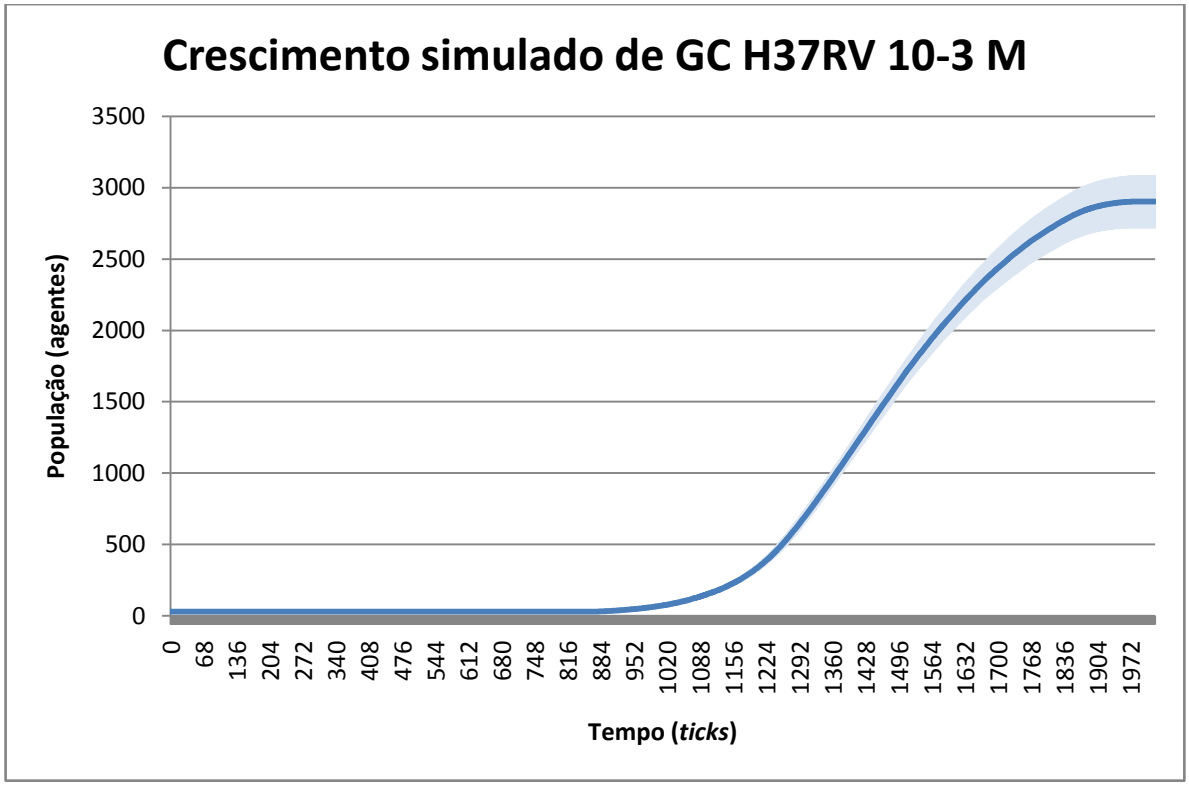


Figura 4.5 - Curvas de crescimento geradas a partir dos valores parâmetros encontrados para *enconsumo*, *decaiconsumo* e *loglim*.
As populações representam as cepas GC 03-850 10-3 M (a), GC 01-2522 10-3 M (b), GC 02-2761 10-3 M (c), GC 03-2922 10-3 M (d) e GC H37RV 10-3 M (e). A linha azul forte representa a média das populações das 50 simulações realizadas enquanto a área azul clara representa o desvio padrão.

As curvas apresentadas na Figura 4.5 são o resultado de uma busca com um grau de similaridade de 97% em comparação às curvas reais. A Tabela 4.5 apresenta um comparativo entre a curva média das 50 simulações e as curvas reais, enquanto a Figura 4.6 mostra graficamente a similaridade entre as curvas para cada cepa.

Tabela 4.5 - Comparativo entre as características encontradas nas curvas de crescimento reais e nas curvas médias simuladas.

Estão representados os comparativos de GC 03-850 10-3 M em (a), GC 01-2522 10-3 M em (b), GC 02-2761 10-3 M em (c), GC 03-2922 10-3 M em (d) e GC H37RV 10-3 M em (e).

(a)

GC 03-850 10-3 M			
	Real	**Simulado**	**Erro**
Lag	833	835	0,24%
Crescimento	959	1010	5,32%
Decaimento	479	516	7,72%
Pop. Máx.	4056	4262	5,08%

(b)

GC 01-2522 10-3 M			
	Real	**Simulado**	**Erro**
Lag	875	877	0,23%
Crescimento	979	1011	3,27%
Decaimento	562	652	16,01%
Pop. Máx.	3236	3325	2,75%

(c)

GC 02-2761 10-3 M			
	Real	**Simulado**	**Erro**
Lag	817	819	0,24%
Crescimento	937	985	5,12%
Decaimento	454	510	12,33%
Pop. Máx.	4087	4052	0,86%

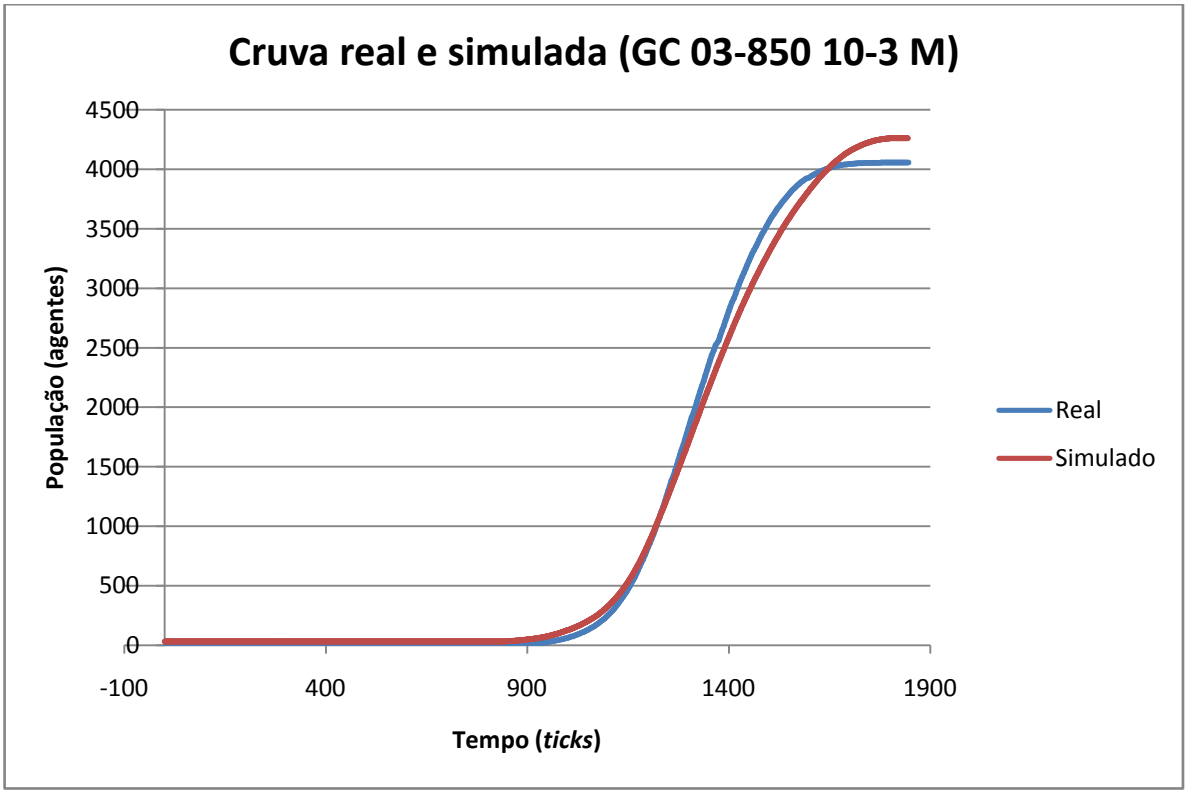
(d)

GC 03-2922 10-3 M			
	Real	**Simulado**	**Erro**
Lag	908	910	0,22%
Crescimento	1000	1028	2,80%
Decaimento	491	530	7,94%
Pop. Máx.	3206	3270	2,00%

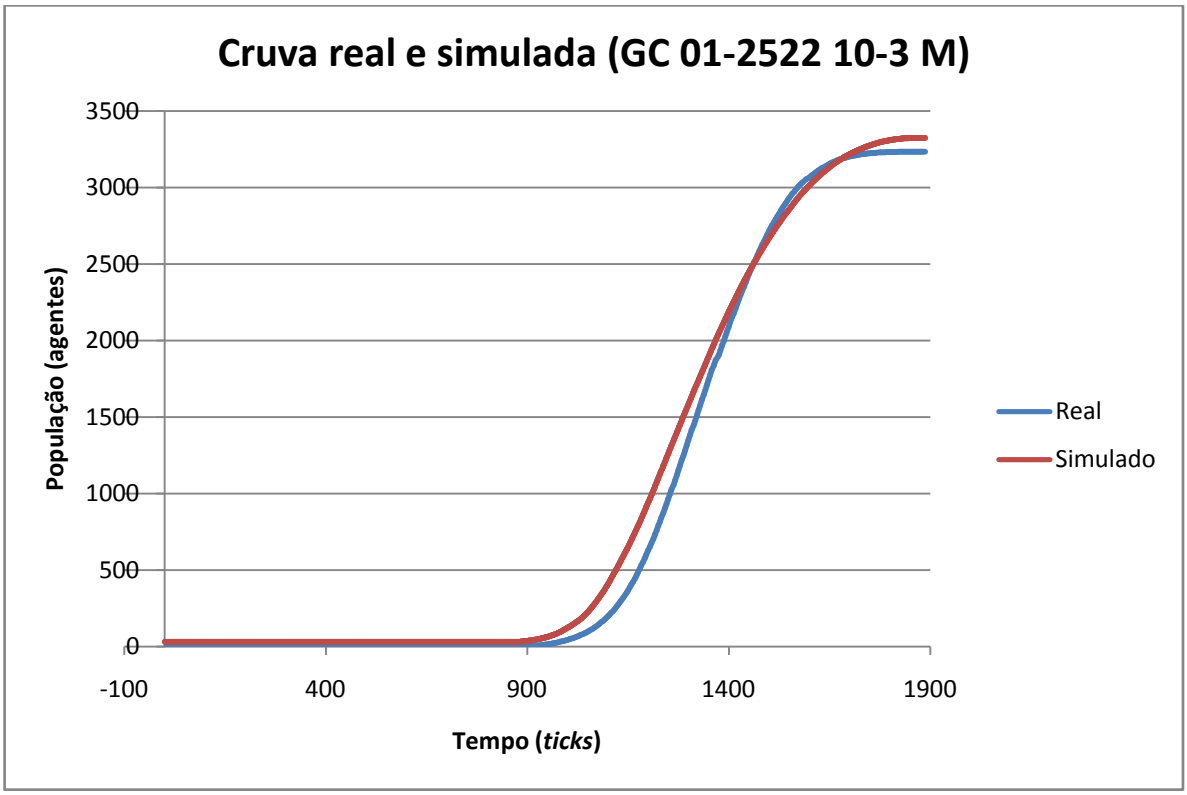
(e)

GC H37RV 10-3 M			
	Real	**Simulado**	**Erro**
Lag	871	874	0,34%
Crescimento	1083	1148	6,00%
Decaimento	591	587	0,68%
Pop. Máx.	2872	2901	1,01%

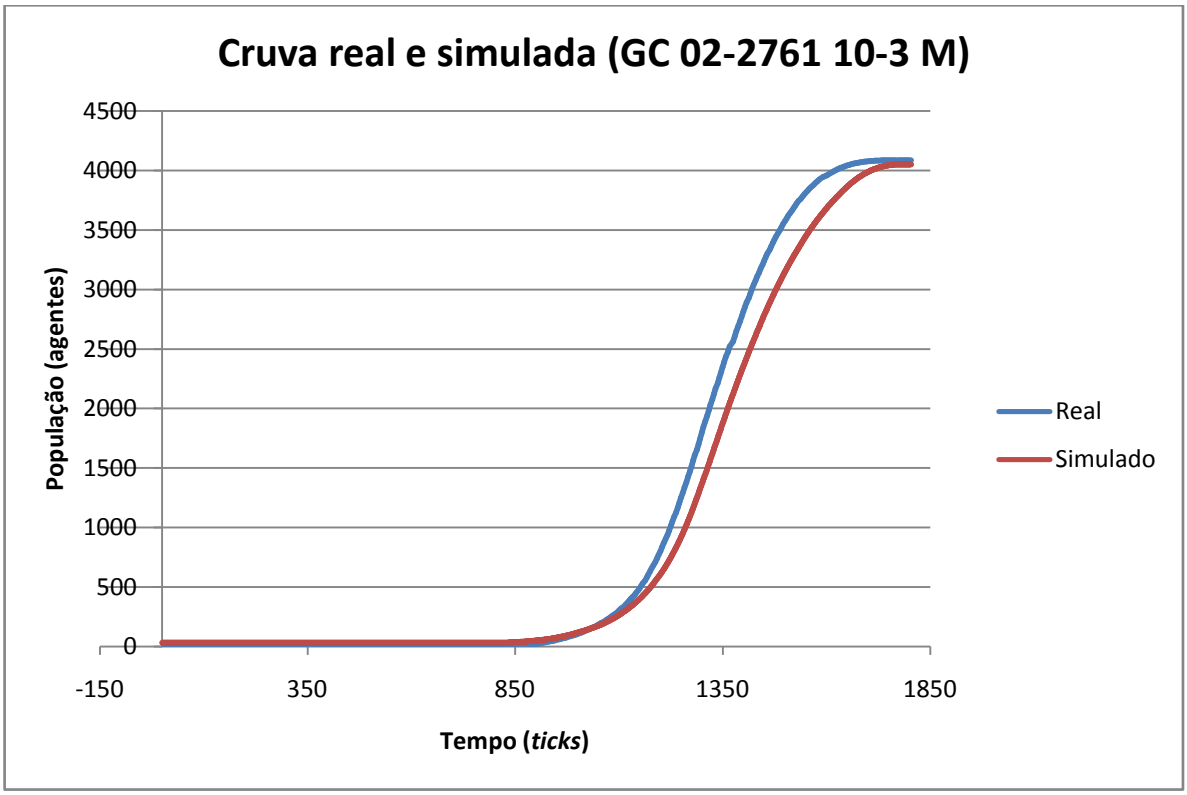
(a)



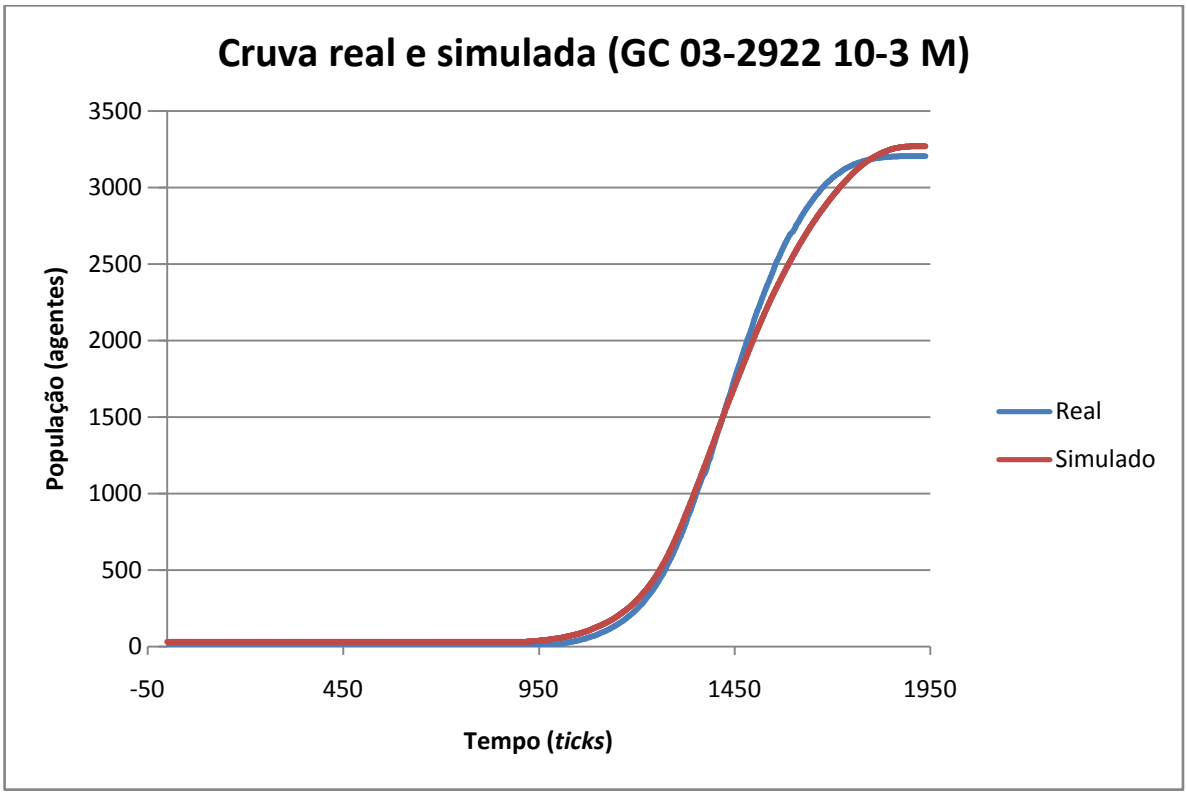
(b)


Cruva real e simulada (GC 01-2522 10-3 M)
3500
3000
2500
2000
1500
1000
500
0
População (agentes)
-100
400
900
1400
1900
Tempo (ticks)
Real
Simulado

(c)


Cruva real e simulada (GC 02-2761 10-3 M)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
População (agentes)
-150
350
850
1350
1850
Tempo (ticks)
Real
Simulado

(d)



(e)

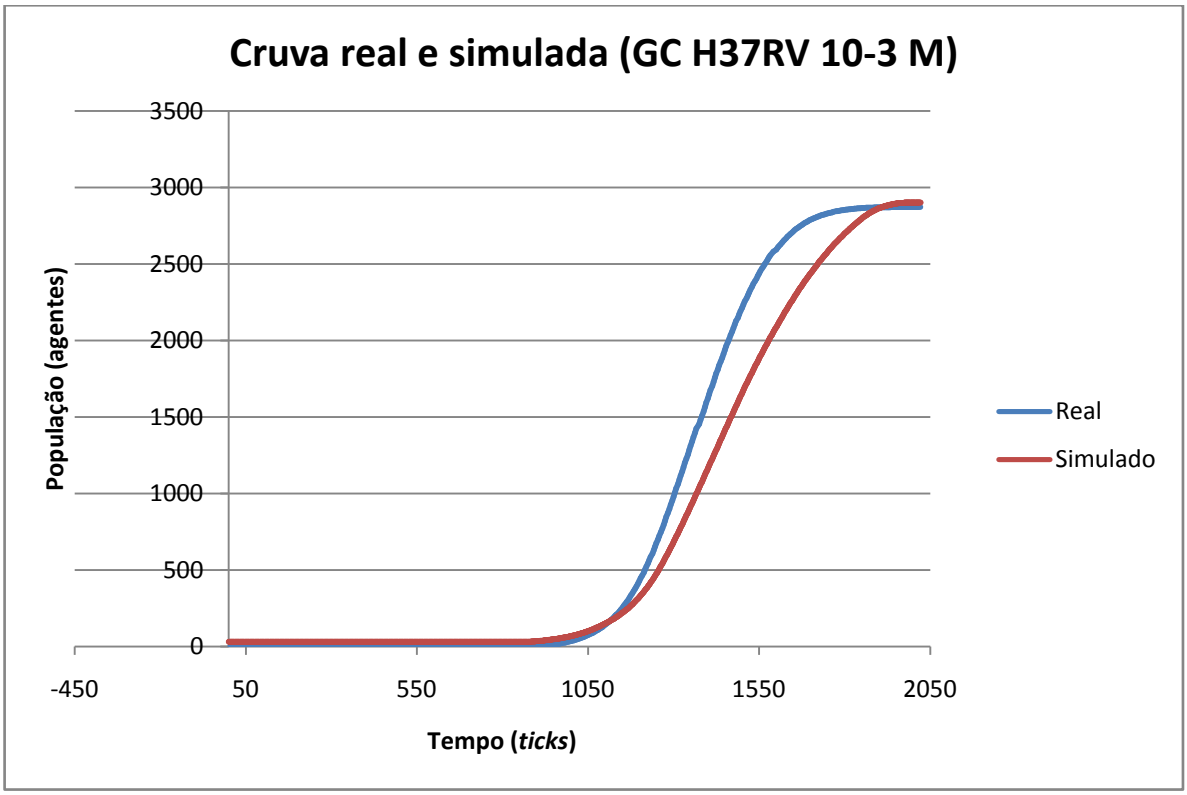


Figura 4.6 - Comparação entre as curvas reais e simuladas.

A linha azul representa a curva de crescimento obtida experimentalmente e a linha vermelha a curva gerada através média das simulações. As cepas GC 03-850 10-3 M, GC 01-2522 10-3 M, GC 02-2761 10-3 M, GC 03-2922 10-3 M e GC H37RV 10-3 M estão representadas em (a), (b), (c), (d) e (e) respectivamente.

Embora o mecanismo de estimativa de parâmetros tenha feito uma busca para um alto grau de similaridade, os resultados das médias das simulações variam bastante devido a natureza complexa que um ambiente multi-agente proporciona. Os valores apresentados dos parâmetros foram encontrados utilizando uma média de 10 iterações e erro de 3%, e por sua vez os resultados das 50 simulações apresentaram margens de erro variadas, indo de 0,22% até 16,01%.

A média dos erros encontrados nas simulações foi de 4,01% com desvio padrão de 0,0434. O nível de erro obtido poderia ser reduzido utilizando na estimativa de parâmetros um número maior de iterações (*iteracoes*) ou então um erro menor (*errogeral*), porém isto demandaria um poder computacional maior.

Os valores de parâmetros utilizados nas simulações resultaram em curvas que, quando comparadas diretamente às curvas reais apresentam forma e valores bastante semelhantes.

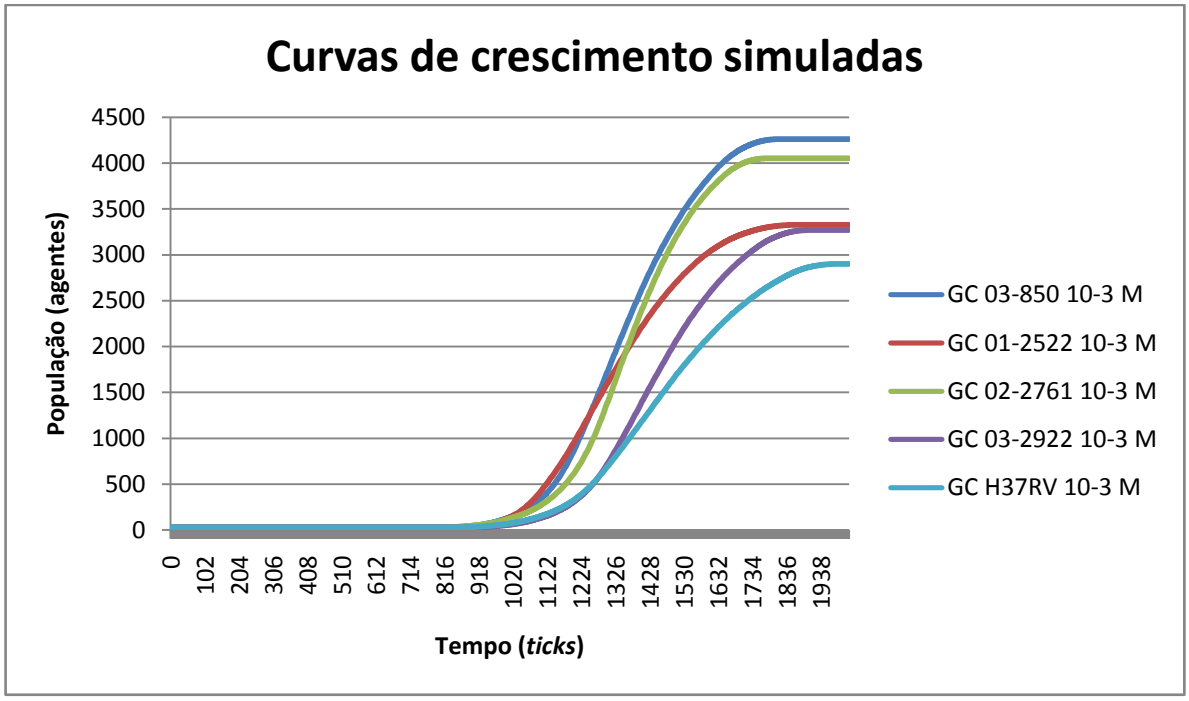


Figura 4.7 - Curvas de crescimento sobrepostas geradas pela média de simulações utilizando valores de parâmetros encontrados através do algoritmo de estimativa de parâmetros.

As curvas de crescimento sobrepostas da Figura 4.7 podem ser comparadas às da Figura 4.4 para fins de comparação dos resultados gerais do modelo. A Figura 4.8 apresenta as curvas destas duas figuras sobrepostas. As populações estão agrupadas por cor, onde a cor azul representa o resultado das cinco populações simuladas, enquanto a cor vermelha representa as curvas reais obtidas.

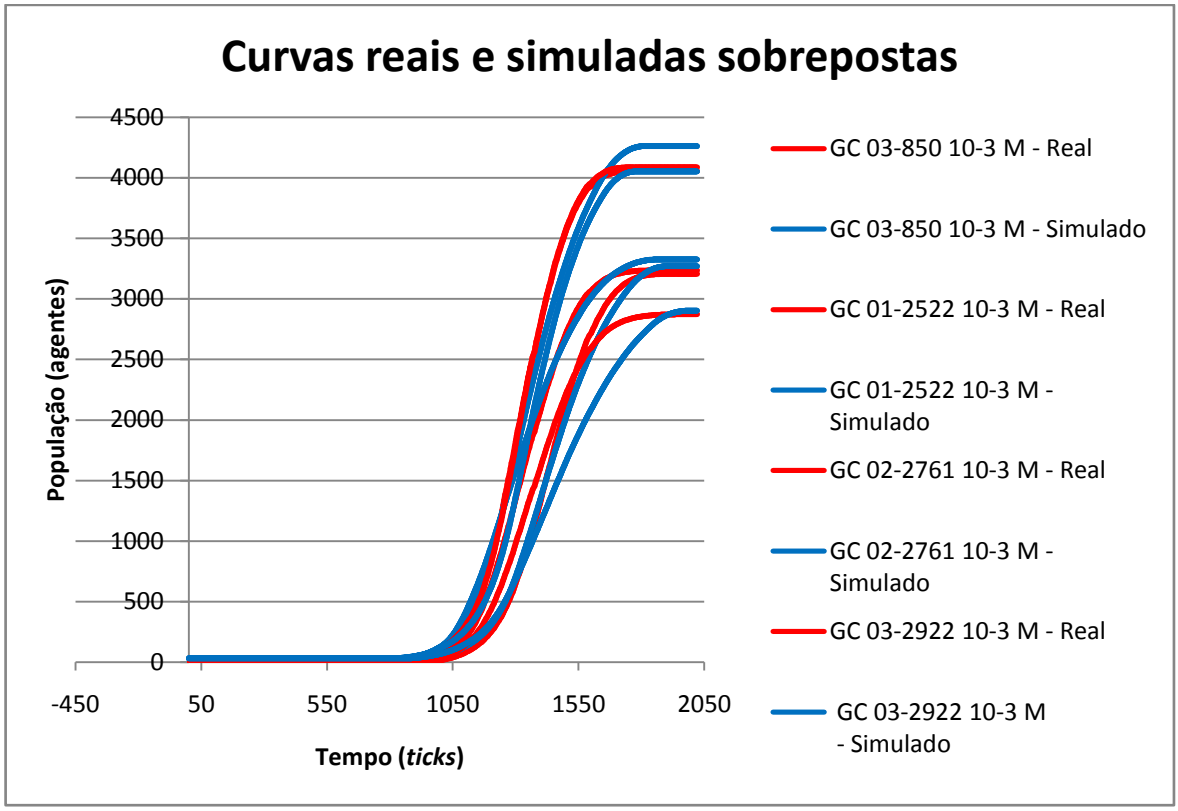


Figura 4.8 - Curvas de crescimento reais e simuladas sobrepostas das cinco populações.

5. CONCLUSÕES E TRABALHOS FUTUROS

O trabalho apresentou um modelo que possui o objetivo de simular curvas de crescimento do *Mycobacterium tuberculosis*, bem como apresentar uma solução que encontre os valores dos parâmetros necessários para reproduzir curvas de crescimento dadas como entrada.

Pela natureza distribuída do problema, um ambiente multi-agente foi escolhido para o desenvolvimento do modelo, onde os agentes representam o bacilo e o ambiente representa o meio de cultura.

As regras comportamentais dos agentes foram estipuladas de acordo com o conhecimento adquirido de especialistas e de referências bibliográficas da área, onde a partir destas regras pode-se observar que certos parâmetros do modelo influenciavam de maneira mais direta na curva de crescimento, e conforme seus valores eram alterados, a forma da curva se adaptava.

Curvas de crescimento provenientes de experimentos reais foram obtidas, e a fim de verificar a possibilidade do modelo de refletir tais curvas, foram feitos testes iniciais tentando reproduzir as curvas através do modelo. Os resultados obtidos foram bem sucedidos e geraram um artigo submetido e aprovado (WERLANG, FAGUNDES, *et al.*, 2013).

Como os parâmetros do modelo foram calibrados manualmente nos testes preliminares, a necessidade de criar um mecanismo que encontrasse automaticamente os valores destes parâmetros que refletissem curvas de crescimento reais se tornou mais evidente, e a partir desta necessidade o mecanismo de estimativa de parâmetros foi desenvolvido.

A criação do mecanismo de estimativa de parâmetros foi validada com as próprias curvas de crescimento utilizadas nos testes anteriores, e com os valores dos parâmetros do modelo tendo sido encontrados, diversas simulações foram executadas a fim de verificar os resultados obtidos.

Os resultados apresentados mostraram-se satisfatórios e o modelo então provou ser eficiente na tarefa de simular curvas de crescimento dadas como entrada. Embora sempre haja um certo nível de erro nas curvas geradas como resultado, o valor deste erro é controlado pelo modelo, e na medida que sejam necessárias modelagens mais precisas é possível ajustar o erro máximo aceitável conforme a necessidade, tendo sempre em mente que o tempo computacional para a obtenção do resultado irá crescer também em ordem não linear.

Outro ponto a ser levantado é que o algoritmo de estimativa de parâmetros desenvolvido neste trabalho, embora aplicado ao modelo de crescimento bacteriano proposto, sua aplicação não se restringe ao referido modelo. O modelo de crescimento proposto, conforme discutido anteriormente, foi criado com base nos dados coletados de especialistas, e faz uso do conceito de *quorum sensing*. Outro modelo poderia ser implementado, e sendo assim os parâmetros existentes no mesmo seriam diferentes dos parâmetros apresentados neste trabalho. Ainda assim a inferência dos valores poderia ser realizada através do algoritmo de estimativa de parâmetros, pois este independe do modelo.

Tendo isto em vista, pode-se afirmar que o uso do algoritmo de estimativa de parâmetros pode ser usado para inferir parâmetros de qualquer modelo proposto para o problema, e inclusive com a ajuda de especialistas da área, ajudar na seleção de modelos mais fiéis aos fenômenos observados *in vitro*, e que sejam biologicamente viáveis, resultando na obtenção dos modelos que gerem as curvas desejadas nos estudos dos profissionais da área, onde os parâmetros para a obtenção destas curvas são conhecidos.

Destacam-se como trabalhos futuros esforços para melhorar a compreensão dos mecanismos biológicos envolvidos no ciclo de vida do *Mycobacterium tuberculosis* e a tradução destes processos em regras comportamentais que podem ser modeladas nos agentes presentes no modelo. Outro fator que pode ser considerado é a obtenção de conhecimento mais detalhado sobre o efeito do mecanismo de *quorum sensing* na curva de crescimento do bacilo, e a melhoria da implementação destes conceitos no modelo.

Na parte computacional, pode-se destacar melhorias no algoritmo de estimativa de parâmetros, buscando maneiras de alcançar o resultado esperado mais rapidamente. Tal feito pode ser obtido estudando e aperfeiçoando o método numérico aplicado.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ADAMATTI, D. F. **Inserção de Jogadores Virtuais em Jogos de Papéis para Uso em Sistemas de Apois à Decisão em Grupo: Um experimento no Domínio da Gestão de Recursos Naturais**. Universidade de São Paulo. São Paulo, p. 196. 2007.

BUCHANAN, R. L.; WHITING, R. C.; DAMERT, W. C. When is simple good enough: a comparison of the Gompertz, Baranyi, and three-phase linear models for fitting bacterial growth curves. **Food Microbiology**, v. 14, n. 4, p. 313-326, 1996.

CLAUS, C.; BOUTILIER, C. The Dynamics of Reinforcement Learning in Cooperative Multiagentes Systems. **AAAI-98: Proceedings of the Fifteenth National Conference on Artificial Intelligence**, Madison, 27 Setembro 1998. 746-752.

DROGOUL, A.; FERBER, J. Multi-agent simulation as a tool for modeling societies: Application to social diferentiation in ant colonies. **Proceedings of Workshop on Modelling Autonomous Agents in a Multi-Agent World**, 1992.

FREITAS, S. R. D. **Métodos Numéricos**. [S.l.]: [s.n.], 2000.

GOLDMAN, L.; AUSIELLO, D. **Cecil - Tratado de Medicina Interna**. 22ª Edição. ed. [S.l.]: Elsevier, v. II, 2004.

KRAWCZYK, K.; DZWINEL, W.; YUEN, D. A. Nonlinear Development of Bacterial Colony Modeled with Cellular Automata and Agent Objects. **International Journal of Modern Physics C**, v. 14, n. 10, p. 1385–1404, 2003.

KREFT, J. U.; BOOTH, G.; WIMPENNY, J. W. T. BacSim, a simulator for individual-based modelling of bacterial colony growth. **Microbiology**, v. 144, n. 12, p. 3275-3287, 1998.

LANGTON, C. G. Self-reproduction in cellular automata. **Physica 10D**, Amsterdam, 1984. 135-144.

LANGTON, C. G. **Artificial Life:** An Overview. [S.l.]: MIT Press, 1997.

MGIT Procedure Manual, 2006. Disponivel em: <http://www.finddiagnostics.org/export/sites/default/resource-centre/find_documentation/pdfs/mgit_manual_nov_2007.pdf>. Acesso em: 30 Agosto 2013.

NETLOGO 5.0.4 User Manual. **The Center for Connected Learning (CCL) and Computer-Based Modeling**, 2013. Disponivel em: <http://ccl.northwestern.edu/netlogo/docs/>. Acesso em: 20 Março 2013.

REZENDE, S. O. **Sistemas Inteligentes:** Fundamentos e aplicações. 1. ed. Barueri: Manole, 2005.

RUGGIERO, M. A. G.; LOPES, V. L. D. R. **Cálculo Numérico:** Aspectos teóricos e computacionais. 2ª Edição. ed. São Paulo: Pearson, 1997.

SIFRI, C. D. Quorum Sensing: Bacteria talk sense. **Clinical Infectious Diseases**, v. 47, n. 8, p. 1070-1076, 2008.

SYCARA, K. P. Multiagent Systems. **AI Magazine**, v. 19, n. 2, p. 79-92, 1998.

TERANO, T. Exploring the Vast Parameter Space of Multi-Agent Based Simulation. **Multi-Agent-Based Simulation VII**, Hakodate, 8 Maio 2006. 1-14.

VON GROLL, A. **Fitness of Mycobacterium tuberculosis associated to genotypes and drug resistance: new approaches for understanding the transmission dynamics of tuberculosis**. Ghent University. Ghent, p. 137. 2010.

WERLANG, P. S. et al. **Multi-Agent-Based Simulation of Mycobacterium tuberculosis growth**. MABS 2013 - 14th International Workshop on Multi-Agent-Based Simulation. Saint Paul, Minnesota, USA: [s.n.]. 2013.

WOLFRAM Alpha. **Wolfram Alpha**, 2013. Disponivel em: <http://www.wolframalpha.com/>. Acesso em: 08 Agosto 2013.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. Liverpool: John Wiley & Sons, 2009.

ZWIETERING, M. H. et al. Modeling of the Bacterial Growth Curve. **Applied and Environmental Microbiology**, v. 56, n. 6, p. 1875-1881, 1990.

7. APÊNDICE

7.1. CÓDIGO DA SIMULAÇÃO

```
;Equivalencias:
;5 lum = 1 agente
;1d = 100 ticks

breed [bac]
breed [sinal]
bac-own [adaptacao energia sense consumo]
patches-own [nutrientes sujeira]
globals [
  ;globais
  proporcao popantiga niter stopflag cont
  ;procurar loglim
  somapop mediapop llinf llsup popant
  ;procurar enconsumo
  ticksantigo no somacre mediacre mediacreantigo ecinf ecsup creant
  ;procurar decaiconsumo
  decinf decsup somadec mediadec difpop maiordifpop maiortick decant
]

to prepara
  ca
  reset-ticks
  set stopflag false
  set cont 0

  if procurapop = true or procuratodos = true[
    set somapop 0
    set llinf 0.001
    set llsup loglim
    set loglim (llinf + llsup) / 2
    set niter iteracoes
    set procurapop true
    set popant 0
  ]

  if procuracre = true or procuratodos = true[
    set no 0
    set somacre 0
    set ecinf 0.1
    set ecsup enconsumo
    set enconsumo (ecinf + ecsup) / 2
    set niter iteracoes
    set procuracre true
    set creant 0
  ]

  if procuradec = true or procuratodos = true[
    set niter iteracoes
```

```
    set decinf 0.001
    set somadec 0
    set decsup decaiconsumo
    set decaiconsumo (decinf + decsup) / 2
    set procuradec true
    set decant 0
  ]

  reinicia
end

to reinicia
  clear-ticks
  clear-turtles
  clear-patches
  clear-drawing
  clear-all-plots

  reset-ticks

  create-bac bacini [
    set color green
    set shape "dot"
    setxy random-pxcor random-pycor
    set adaptacao 0
    set energia random energiamax
    set sense false
    set consumo 1
  ]

  ask patches [
    set nutrientes nutriini
    set sujeira 0
  ]
end


to reproduz
  if any? neighbors with [not any? bac-here] and energia >=
energiamax[
    set energia random energiamax
    hatch 1 [
      move-to one-of neighbors with [not any? bac-here]
      set energia energiamax - energia
      set sense false
    ]
  ]
end

to move
  if any? neighbors with [not any? bac-here and nutrientes > 0] [
    face one-of neighbors with [not any? bac-here and nutrientes >
0]
    fd consumo
  ]
end
```

```
to adapta
  set adaptacao adaptacao + 1
end

to sinaliza
  if random 10000 < taxasinal * 100 * consumo [
    ask patch-here [
      if any? neighbors with [not any? sinal-here] [
        sprout-sinal 1 [
          set color white
          set shape "dot"
          move-to one-of neighbors with [not any? sinal-here]
        ]
      ]
    ]
  ]
end

to metaboliza
  if sense = true [
    set consumo consumo * (1 - decaiconsumo)
  ]

  sinaliza

  set sujeira sujeira + consumo

  if nutrientes > 0 [
    set nutrientes nutrientes - consumo
    if energia < energiamax [
      set energia energia + (nutrientes / (nutrientes + sujeira)) *
consumo
    ]
  ]


  ifelse consumo < consumomin [
    set energia energia - enconsumo * consumomin
  ]
  [
    set energia energia - enconsumo * consumo
  ]

  if energia <= 0 [die]
end

to achaloglim
  if cont >= 10 and count bac with [sense = true] = count bac [
    set niter niter - 1
    set somapop somapop + count bac
    if niter <= 0 [
      let errocalc sqrt((somapop / iteracoes - mediapop) ^ 2)
      set mediapop somapop / iteracoes
      print (word "População: " mediapop " Erro: " errocalc)
      ifelse errocalc < erropop [
```

```
        set procurapop false
        set cont 0
      ]
      [
        if somapop / iteracoes > popgoal [
          set llsup loglim
        ]
        if somapop / iteracoes < popgoal [
          set llinf loglim
        ]
        set loglim (llsup + llinf) / 2
      ]
      set niter iteracoes
      set somapop 0
    ]
    set procurapop false
    set procuracre true
    reinicia
  ]

end

to achaenconsumo
  if cont >= 10 and count bac with [sense = true] = count bac[
    set niter niter - 1
    set somacre somacre + ticks
    if niter <= 0 [
      let errocalc sqrt((somacre / iteracoes - mediacre) ^ 2)
      set mediacre somacre / iteracoes
      print (word "Crescimento: " mediacre " Erro: " errocalc)
      ifelse errocalc < errocre [
        set procuracre false
        set cont 0
      ]
      [
        if somacre / iteracoes > cregoal [
          set ecsup enconsumo
        ]
        if somacre / iteracoes < cregoal [
          set ecinf enconsumo
        ]
        set enconsumo (ecinf + ecsup) / 2
      ]
      set somacre 0
      set niter iteracoes
    ]
    set procuracre false
    set procuradec true
    reinicia
  ]
end

to achadecaiconsumo
  set difpop count bac - popantiga
  if difpop >= maiordifpop and  difpop != bacini [
    set maiordifpop difpop
```

```
    set maiortick ticks
  ]
  if cont >= 10 and count bac with [sense = true] = count bac[
    set niter niter - 1
    set somadec somadec + (ticks - maiortick)
    if niter <= 0 [
      let errocalc sqrt((somadec / iteracoes - mediadec) ^ 2)
      set mediadec somadec / iteracoes
      print (word "Decaimento: " mediadec " Erro: " errocalc)
      ifelse errocalc < errodec [
        set procuradec false
        set cont 0
      ]
      [
        if somadec / iteracoes > decgoal [
          set decinf decaiconsumo
        ]
        if somadec / iteracoes < decgoal [
          set decsup decaiconsumo
        ]
        set decaiconsumo (decinf + decsup) / 2
      ]
      set somadec 0
      set maiordifpop 0
      set difpop 0
      set niter iteracoes
    ]
    set procuradec false
    set procurapop true
    reinicia
  ]
end

to achatodos
  set difpop count bac - popantiga
  if difpop >= maiordifpop and  difpop != bacini [
    set maiordifpop difpop
    set maiortick ticks
  ]
  if cont >= 10 and count bac with [sense = true] = count bac [
    set niter niter - 1

    set somapop somapop + count bac
    set somacre somacre + ticks - laglim
    set somadec somadec + (ticks - maiortick)

    if niter <= 0 [
      print (word "----------------------------------------")

      let errocalccre sqrt((somacre / iteracoes - mediacre) ^ 2)
      set mediacre somacre / iteracoes
      print (word "Crescimento: " mediacre " Erro: " errocalccre)

      let errocalcdec sqrt((somadec / iteracoes - mediadec) ^ 2)
      set mediadec somadec / iteracoes
      print (word "Decaimento: " mediadec " Erro: " errocalcdec)
```

```
      let errocalcpop sqrt((somapop / iteracoes - mediapop) ^ 2)
      set mediapop somapop / iteracoes
      print (word "População: " mediapop " Erro: " errocalcpop)

      ifelse errocalcpop < erropop [
        set procurapop false
        set cont 0
      ]
      [
        if somapop / iteracoes > popgoal [
          set llsup loglim
        ]
        if somapop / iteracoes < popgoal [
          set llinf loglim
        ]
        set loglim (llsup + llinf) / 2
      ]

      ifelse errocalccre < errocre [
        set procuracre false
        set cont 0
      ]
      [
        if somacre / iteracoes > cregoal [
          set ecsup enconsumo
        ]
        if somacre / iteracoes < cregoal [
          set ecinf enconsumo
        ]
        set enconsumo (ecinf + ecsup) / 2
      ]

      ifelse errocalcdec < errodec [
        set procuradec false
        set cont 0
      ]
      [
        if somadec / iteracoes > decgoal [
          set decinf decaiconsumo
        ]
        if somadec / iteracoes < decgoal [
          set decsup decaiconsumo
        ]
        set decaiconsumo (decinf + decsup) / 2
      ]

      set niter iteracoes

      set somapop 0

      set somacre 0

      set somadec 0
      set maiordifpop 0
      set difpop 0
```

```
    ]
    reinicia
  ]
end

to executa
  set proporcao count patches with [any? sinal-here] / count patches
  ask bac [
    ifelse adaptacao > laglim [
      move
      metaboliza
      reproduz
    ]
    [
      adapta
    ]
    if proporcao > loglim [set sense true]
  ]
  ask patches [
    set pcolor 10 + 5 * (sujeira / (nutrientes + sujeira))
  ]

  ifelse count bac <= popantiga [
    set cont cont + 1
  ]
  [
    set cont 0
  ]

  ifelse procuratodos = true [
    achatodos
  ]
  [
    if procuracre = true [
      achaenconsumo
    ]
    if procuradec = true [
      achadecaiconsumo
    ]
    if procurapop = true [
      achaloglim
    ]
  ]

  set popantiga count bac

  if procurapop = false and procuradec = false and procuracre =
false and procuratodos = true [
    let difll sqrt((mediapop - popgoal) ^ 2)
    let difcre sqrt((mediacre - cregoal) ^ 2)
    let difdec sqrt((mediadec - decgoal) ^ 2)
    let crep 100 * difcre / cregoal
    let decp 100 * difdec / decgoal
    let popp 100 * difll / popgoal
    print (word "----------------------------------------")
    print (word "Valores: " enconsumo " " decaiconsumo " " loglim)
```

```
    print (word "Variações: " crep "% " decp "% " popp "%")
    ifelse popp <= errogeral and decp <= errogeral and crep <=
errogeral[; and popant != 0 and creant != 0 and decant != 0[
      stop
    ]
    [
      set llsup loglim + (0.05 - 0.001) * (popp / 100)
      set llinf loglim - (0.05 - 0.001) * (popp / 100)
      set mediapop 0
      if llsup > 0.05 [
        set llsup 0.05
      ]
      if llinf < 0.001 [
        set llinf 0.001
      ]

      set decsup decaiconsumo + (0.02 - 0.001) * (decp / 100)
      set decinf decaiconsumo - (0.02 - 0.001) * (decp / 100)
      set mediadec 0
      if decsup > 0.02 [
        set decsup 0.02
      ]
      if decinf < 0.001 [
        set decinf 0.001
      ]

      set ecsup enconsumo + (0.9 - 0.1) * (crep / 100)
      set ecinf enconsumo - (0.9 - 0.1) * (crep / 100)
      set mediacre 0
      if ecsup > 0.9 [
        set ecsup 0.9
      ]
      if ecinf < 0.1 [
        set ecinf 0.1
      ]

      set procurapop true
      set procuradec true
      set procuracre true
    ]
  ]

  tick
end
```