DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Relatório do Projeto de LAPR1

24 de janeiro de 2021

1DGH, Grupo 06, Os Granjeiros_

1201564, Jorge Ferreira

1201566, Rafael Leite

1201568, Rui Pina

1201424, João Torres

Orientadores_

Ana Barata (ABT)

Ana Moura (AIM)

Rosa Barroso (RGB)

Carlos Ferreira (CGF)

isep Instituto Superior de Engenharia do Porto

ÍNDICE

Conteúdo

1. Introdução

No âmbito da disciplina de LAPR1, no ano letivo 2020/2021, foi-nos proposto a elaboração de uma aplicação em linguagem *Java*. Nessa aplicação é pretendido um processo onde, através da introdução de dados, se previsse a evolução de espécies utilizando um modelo que prevê a evolução de uma dada espécie. Neste caso, foi nos aventado o Modelo introduzido por Lotka e formalizado por Leslie.

Neste documento iremos, primeiramente, expor os métodos de trabalho da equipa que , ao longo destas semanas desenvolveu a aplicação, referindo como utilizamos a estrutura do *scrum* no desenvolvimento do projeto, como realizamos o planeamento e definimos a divisão das tarefas e ainda terá uma sucinta reflexão crítica sobre a dinâmica do grupo.

Após isso iremos expor o modelo de evolução de espécies que teve por base este projeto (Modelo de Lotka-Leslie). Onde, também iremos manifestar a sua importância para os estudos populacionais atuais.

Em seguida iremos exibir o modo como desenvolvemos e implementamos a aplicação. Para isso irá ser mostrado e explicado todos os métodos realizados e iremos apresentar um diagrama que explica a sua integração.

Por fim iremos estudar, analisar e discutir 2 casos de estudo. Onde iremos simular 100 gerações de duas espécies e mostrar todas as conclusões que conseguirmos chegar após o emprego da aplicação, desenvolvida pela nossa equipa.

2. Metodologia de Trabalho

Neste projeto é utilizado uma *framework* intitulada de "Scrum". Esta metodologia foi exposta e analisada aos membros da equipa no módulo anterior de LAPR1 (Competências Pessoais e Métodos de trabalho), tendo sido usada para gerir as tarefas em boa parte do anterior módulo. É um método que todos os membros do grupo dominam e estão bastante habituados a usar. Com isto pode-se afirmar que a nossa forma de trabalhar foi bastante influenciada por este método, que iremos explicar no que consiste no próximo ponto.

2.1 Scrum no desenvolvimento do Projeto

O Scrum é caracterizado por ser uma framework ágil de desenvolvimento de produtos. Esta abordagem define uma estratégia flexível e globalizante, onde todos os membros fazem parte do desenvolvimento dos produtos. Ao invés de uma abordagem sequencial, os produtos são progressivamente desenvolvidos e melhorados de forma iterativa e incremental, em que as equipas se auto-organizam.

Todos os participantes que se referem a este processo precisam de usar uma linguagem comum, e todos os envolvidos tanto no desenvolvimento como na aceitação do trabalho, precisam de ter uma definição padrão de "Done".

Esta "ferramenta" permite controlar de uma forma eficaz e eficiente o trabalho, potencializando as equipas que trabalham em prol de um objetivo em comum.

É uma metodologia essencial para muitas empresas atualmente, porque não apenas facilita a definição de objetivos, como também ajuda a cumprir os prazos estabelecidos.

Neste método há 2 noções que foram bastante vincadas no nosso trabalho: --

Scrum Master - A sua principal responsabilidade é ajudar a equipa a seguir o processo Scrum. Ao contrário de um gestor de projetos numa abordagem, o Scrum Master não é responsável pelo planeamento, isso é feito pelo cliente e pela equipa. Também não gere os elementos da equipa, nem é responsável pela qualidade do produto, isso é feito pela equipa. Dito isto, qual é então o seu papel? Ele é responsável por assegurar que a equipa segue o processo, bem como ser um facilitador que remove obstáculos da equipa. Estas responsabilidades traduzem-se em:

- Facilitar a reunião diária (daily Scrum).
- Organizar as sessões de planeamento, revisões de sprint e retrospetivas.
- Proteger a equipa de interrupções durante o sprint.
- Escudo contra interferências externas.
- Ser um líder sem liderar.
- Remover obstáculos que possam prejudicar a equipa.
- Encorajar a colaboração entre a equipa e o cliente.
- Auxiliar o *Product Owner*.

Sprint - Os sprints são um elemento fundamental do Scrum, onde as ideias ganham valor. Cada Sprint tem uma duração fixa, de no máximo um mês, em que todo o trabalho necessário para chegar ao produto final está inserido no sprint. No caso de haver mais um sprint, como é o caso, o próximo sprint começa logo a seguir ao anterior.

Durante os sprints:

- Não são feitas mudanças que possam mudar de forma radical o resultado.
- A Qualidade não se reduz.
- A direção pode ser clarificada e renegociada de acordo com novas especificações do cliente.

Com a grande volatilidade dos clientes, que foi possível ver em certos momentos deste projeto, pode-se afirmar que os requisitos não podem ser abordados de forma planeada. Daí a existência do Scrum, que não assume o problema como podendo ser caracterizado de forma verdadeira, mas sim como uma ideia motora. Isso faz com que seja uma metodologia mais dinâmica e não seja tão agarrado à conceção inicial, como outros métodos tradicionais.

Tendo isto em conta, é possível afirmar que a melhor abordagem da gestão de tarefas para este tipo de projetos é o Scrum, logo foi este método que usamos. O nosso trabalho foi divido em 2 Sprints, como foi sugerido pelo cliente.

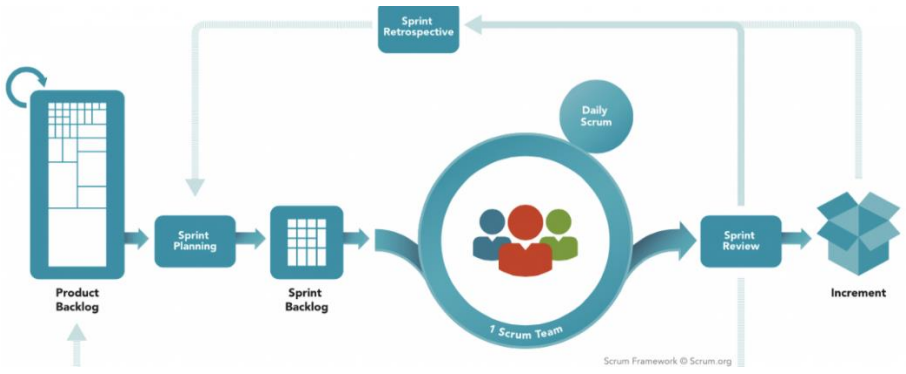


Figura 1- Modelo do Scrum

O primeiro Sprint foi maioritariamente utilizado para fazer uma análise escrutinada do projeto e para estudar os assuntos utilizados neste projeto (Modelo de Lotka-Leslie (1945), la4j, Gnuplot, etc.). Além disso foi estabelecido a criação de alguns métodos do projeto. Esses foram

maioritariamente de caráter matemático, como por exemplo encontrar o maior valor próprio de uma população para estudar e quantificar o seu crescimento(aumento/declínio/estagnação).

Por outro lado, no segundo Sprint foi onde fizemos a aplicação propriamente dita, com a união de métodos feitos separadamente, alterações no input para ir buscar os parâmetros à linha de comandos, formatação do output, implementação de gráficos, entre outras tarefas relacionadas à programação. A realização do relatório também está inserida nesta parte. Esta foi uma tarefa em que todos os elementos contribuíram e que faz um sumário de todo o projeto.

2.2 Planeamento e distribuição de tarefas

Durante estas semanas realizamos as atividades propostas dando grande ênfase ao trabalho de equipa, em que cada um tinha oportunidade para pôr duvidas e sugerir ideias de como resolver certo problema.

Para isso, os sprints foram separados em tarefas menores, repartidas de igual forma pelos membros da nossa equipa, em que cada um tinha tempo de pensar em cada problema que o competia resolver, tirando partido de artigos/vídeos tutoriais e da ajuda dos colegas para o realizar.

Por fim, reuníamo-nos no final do horário das aulas, no *Discord*, para apresentar e discutir aquilo que fizemos durante a semana, pondo de forma oportuna dúvidas e dando feedback ao trabalho realizado pelo elemento do grupo.

Para fazer tudo isto referenciado anteriormente, recorremos a diversos programas e plataformas, com vista a tornar o nosso trabalho organizado e eficiente. Estes programas podem ser divididos em 2 grupos: os de Trabalho/Gestão do projeto e os de comunicação.

No início do projeto usamos o *Mendeley*. Este é um programa que já tínhamos usado no módulo passado de LAPR1, por sugestão da professora, e podemos referir que talvez foi um dos programas mais úteis na gestão de fontes que e facilitou a apresentação das referências bibliográficas (segundo as normas da APA).

O *Bitbucket* e o *Sourcetree* (que são 2 programas interligados) serviram para a gestão do código no desenvolvimento do programa. Usamos para juntar tudo o que os membros do grupo tinham feito para depois chegar ao resultado comum.

Utilizamos o *Trello*, como plataforma de gestão do *Scrum*. Onde podemos pôr as finalidades do projeto e onde conseguimos repartir as tarefas por partes, sendo que cada membro ficava com algumas tarefas (que já tinham sido repartidas de forma igualitária).

Na parte da comunicação usamos o Teams para contacto com os professores auxiliares e, também com o cliente. Além disso, foi utilizado para ver novos requisitos do cliente. Usamos maioritariamente o *Discord* para contacto entre os membros do grupo e para discutirmos o trabalho que tínhamos realizado. Ambas as plataformas são parecidas, mas preferimos usar mais o *Discord* por ser uma plataforma que estamos mais habituados, e que já faz parte do nosso quotidiano há algum tempo.

Por fim, podemos afirmar que o nosso trabalho foi realizado da mesma forma como tínhamos referido no início de janeiro. Todos foram bastante participativos e ajudaram a realizar o trabalho.

Não podemos de deixar de notar a comunicação que tínhamos sem ser nos tempos estabelecidos para o projeto. Fora do horário não havia grande interatividade entre os membros do grupo, mas na parte final conseguimos passar este "obstáculo".

2.3 Reflexão crítica sobre a dinâmica do grupo

Na nossa conceção, o ponto de partida de um trabalho de grupo é o entendimento de que existe um objetivo maior, neste caso o aplicativo finalizado, e que cada um vai colaborar de alguma maneira para alcançá-lo. É aqui que a dinâmica de grupo entra, pois ajuda na integração e possibilita chegar ao desejado.

A nossa dinâmica correu muito bem, tal como esperado. Um fator que nos favoreceu foi o facto de já nos conhecermos e nutrirmos uma amizade. Fomos sempre frontais uns com os outros, se estava mal ou não gostávamos de algo sempre exponhamos isso nas nossas reuniões.

Acreditamos que a distribuição do trabalho em partes, dando equitativamente uma tarefa a cada elemento, fez com que a nossa concentração e atenção ao detalhe fosse maior. Assim, as chances de o produto final ter um melhor resultado aumentam.

Este projeto fez-nos evoluir em termos de trabalho em conjunto, e pensamos que esta dinâmica serve de preparação para o mercado de trabalho, pois é uma espécie de "simulação" possível de uma eventual realização de aplicação.

De forma a salientar as nossas opiniões, decidimos inserir um parágrafo individual com uma pequena reflexão sobre a dinâmica de grupo.

João Torres - "Desde que fui adicionado à equipa antes do trabalho começar, fui logo inserido num grupo no Discord para podermos falar sobre o trabalho e tirar dúvidas. Quando o trabalho começou, acho que as tarefas foram bem distribuídas, e os meus colegas disponibilizaram-se para me tirar dúvidas. Agora que o projeto já está quase finalizado, posso afirmar que o trabalho de equipa foi muito bom."

Jorge Ferreira - "A nossa dinâmica foi muito boa. Todos os elementos mostraram-se disponíveis para a realização de qualquer tarefa e quando necessitávamos de ajuda havia sempre alguém disponível para ajudar. Estamos todos de parabéns."

Rafael Leite - "Na minha opinião as tarefas foram bem distribuídas ao longo do trabalho, com cada membro a focar-se num conhecimento específico, fazendo com que o trabalho ficasse melhor no fim. Além disso acho que o grupo comunicou suficientemente bem para chegarmos à aplicação que desenvolvemos."

Rui Pina - "Tendo priorizado a comunicação, acho que tivemos bem em termos completado as nossas tarefas em tempo e de termos partilhado conhecimentos depois de cada um ter estudado uma parte específica do seu trabalho. Não só isso, mas também o facto de os scrum masters terem gerido bem os prazos e dividido as partes foram todos fatores que levaram ao sucesso deste projeto."

3. Evolução das espécies e o modelo de Lotka-Leslie

Esta secção tem de incluir o estudo sobre o modelo de Lotka-Leslie e sua análise, incluindo exemplos ilustrativos.

Esta secção pode ser organizada nas subsecções que considerarem adequadas para melhor descreverem o desenvolvimento do trabalho.

3.1 Contextualização do modelo de Leslie

Sendo um dos modelos mais respeitados de crescimento populacional, o modelo de Lotka-Leslie, vem aparecer no seguimento de uma necessidade de estudar a evolução de espécies, quer ela seja humana ou animal. Este modelo é baseado em matrizes, tendo sido apresentado primeiramente por Alfred J. Lokta na década de 20 e formalizado por Patrick Holt Leslie, em 1945, no artigo intitulado "On the use of matrices in certain population mathematics". O modelo de Leslie caracteriza o crescimento da parcela feminina de uma população assumindo uma expectativa máxima de vida, subdividido em intervalos de tempos iguais, onde se considera taxas de fertilidade e mortalidade em cada faixa etária.

3.2 Modelo de Leslie

O modelo de Lotka-Leslie baseia-se no estudo de matrizes, com recurso a variáveis determinadas pelo utilizador. Entre essas variáveis estão: a população inicial da espécie para cada classe, a taxa de fecundidade (F_i) e a taxa de sobrevivência para cada classe (S_i).

Com o recurso a estas variáveis é possível fazer previsões à população da espécie. Nomeadamente à evolução da distribuição da população, isto é, a evolução de cada classe ao longo do tempo. À taxa de variação ao longo dos anos e ao comportamento assimtótico dessa espécie através do estudo maior valor próprio e do seu respetivo vetor.

Com isto em mente, pode-se proceder representar a definição geral da Matriz de Leslie, como sendo:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_{i-1} \\ x_i \end{bmatrix}_{t+1} = \begin{bmatrix} f_1 & f_1 & f_2 & \dots & f_{i-1} & f_i \\ s_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & s_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & s_3 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & s_{i-1} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_{i-1} \\ x_i \end{bmatrix}_t$$

Traduzindo esta operação entre matrizes em linguagem natural, pode-se afirmar que a distribuição da população (por classes) da geração seguinte será igual à multiplicação entre a matriz contendo as taxas de fecundidade e de sobrevivência e o vetor contendo a distribuição da população (por classes) da geração anterior.

Aqui estão algumas informações a salientar:

- A matriz de Leslie é sempre uma matriz quadrada.
- A primeira linha da matriz é ocupada pelos membros denotados em F_i.Estes são os números médios de nascimentos por cada membro feminino de cada classe.
- A partir da segunda linha da matriz pode-se observar que a diagonal, começando pelo primeiro membro da segunda linha, está ocupada pelos termos denotados com S_i. Cada um destes termos representam a fração da i-ésima classe que sobrevivem e passam para a próxima classe. Por exemplo, no fim do ciclo da primeira geração o número de sobreviventes da classe 1 é s_1x_1, e estes sobreviventes são agora parte da classe 2.
- Cada elemento do vetor X_t é igual ao número de indivíduos da classe i no início da geração atual.
- Cada elemento do vetor X_{t+1} é igual ao número de indivíduos da classe i no fim da geração atual.

Considerando que x_i^t é o número de indivíduos reprodutores (normalmente, do sexo feminino) cujas idades no tempo t pertencem a classes muito bem definidas i. Seja x_i^0 o número de indivíduos reprodutores do intervalo de idade i no momento inicial (t = 0), que é conhecido, e utilizando as variáveis F_i e S_i pode-se estabelecer as seguintes equações:

- $x_{i+1}^{t+1} = x_i^t s_i$
- i=0, ..., n-1
- $x_0^{t+1} = \sum_{i=0}^{n} x_i^t F_i$

Pode-se ainda estudar a dinâmica da população recorrendo ao estudo do maior valor próprio, λ, enquanto a distribuição dos sujeitos nas diferentes classes (normalizada pela população total), obtém-se como o limite de X^t para t → + ∞ e verifica Ax = λX, A sendo a matriz correspondente aos valores de Fecundidade (f_i) e de Sobrevivência s_i devidamente implementados.

Com o estudo do maior valor próprio pode-se concluir o estado da evolução da espécie em questão, podendo ter os seguintes casos:

- λ<1 ->A população diminui.
- λ>1-> A população "explode" (cresce).
- λ=1->A população estagna.

Além disso o seu respetivo vetor próprio irá nos possibilitar a visualização da distribuição por classe, isto é, consoante um espaço temporal infinito, em que determinamos o maior vetor próprio, o seu respetivo vetor irá representar a distribuição por classe (normalizada).

3.2 Exemplo demonstrativo do Modelo de Leslie

Para motivos puramente demonstrativos, iremos supor que queremos estudar uma geração da evolução da população de linces ibéricos em Portugal, utilizando valores fictícios.

Admitindo a existência de 4 classes, expressas em anos: Juvenis [0,3[, Jovens [3,6[, Adultos [6,9[e Maduros [9,11] iremos preencher o vetor da distribuição inicial (X_0). com a população inicial de cada classe

$$X_0=\begin{bmatrix}200\\300\\330\\100\end{bmatrix}$$

Além disso, é imperativo a introdução a taxa de fecundidade e a taxa de sobrevivência que iremos supor que é a seguinte matriz:

$$M=\begin{bmatrix}0 & 1,7 & 3,2 & 0,4\\0,9 & 0 & 0 & 0\\0 & 0,7 & 0 & 0\\0 & 0 & 0,4 & 0\end{bmatrix}$$

Com o intuito de estudar a evolução da população de uma espécie, pode-se fazer uma simulação, possibilitando a demonstração do número de indivíduos, a taxa de variação da população entre gerações e a distribuição da população por classes (também normalizada).

Facilmente se consegue chegar à distribuição da população por classes fazendo a Multiplicação entre a matriz com os valores (S_i) e (F_i) e o vetor com a população inicial (X_0).

$$\begin{bmatrix}0 & 1,7 & 3,2 & 0,4\\0,9 & 0 & 0 & 0\\0 & 0,7 & 0 & 0\\0 & 0 & 0,4 & 0\end{bmatrix} * \begin{bmatrix}200\\300\\330\\100\end{bmatrix} = \begin{bmatrix}1606\\180\\210\\132\end{bmatrix}$$

- O número de indivíduos da primeira classe da geração 1 será igual à taxa de fecundidade de cada classe, multiplicada pela população dessa mesma classe. Daí pode-se referir que:
  X_1^1=0*200+1,7*300+3,2*330+0,4*100 $\Leftrightarrow X_1^1$=1606
- A partir da segunda classe, o número de sujeitos será equivalente ao produto do número da população da classe anterior com a sua respetiva taxa de sobrevivência, com exceção da última classe que irá morrer toda e será substituída pela taxa de sobrevivência e pelo número de indivíduos da classe anterior. Com isto pode-se completar a matriz com os seguintes cálculos:

X_2^1=0,9*200$\Leftrightarrow X_2^1$=180
X_3^1=0,7*330$\Leftrightarrow X_3^1$=210
X_4^1=0,4*100$\Leftrightarrow X_4^1$=132

Com estes dados, pode-se agora calcular o **número total de indivíduos**:

$N_{total}= X_1^1 + X_2^1+X_3^1+X_4^1 \Leftrightarrow N_{total}$=2128.

A **taxa de variação**:

$\Delta t = N_{total}^t / N_{total}^{t-1} \Leftrightarrow \Delta t \simeq 2.29$

Distribuição normalizada por classe da população:

$D_{normal1}=(X_1^1/N_{total})*100 \Leftrightarrow D_{normal1} \simeq 75,47\%$

$D_{normal2}=(X_2^1/N_{total})*100 \Leftrightarrow D_{normal2} \simeq 8,46\%$

$D_{normal3}=(X_3^1/N_{total})*100 \Leftrightarrow D_{normal3} \simeq 9,87\%$

$D_{normal4}=(X_4^1/N_{total})*100 \Leftrightarrow D_{normal4} \simeq 6,20\%$

Comportamento Assintótico (utilizando a nossa aplicação):

Maior λ =1,6662

Vetor próprio associado:

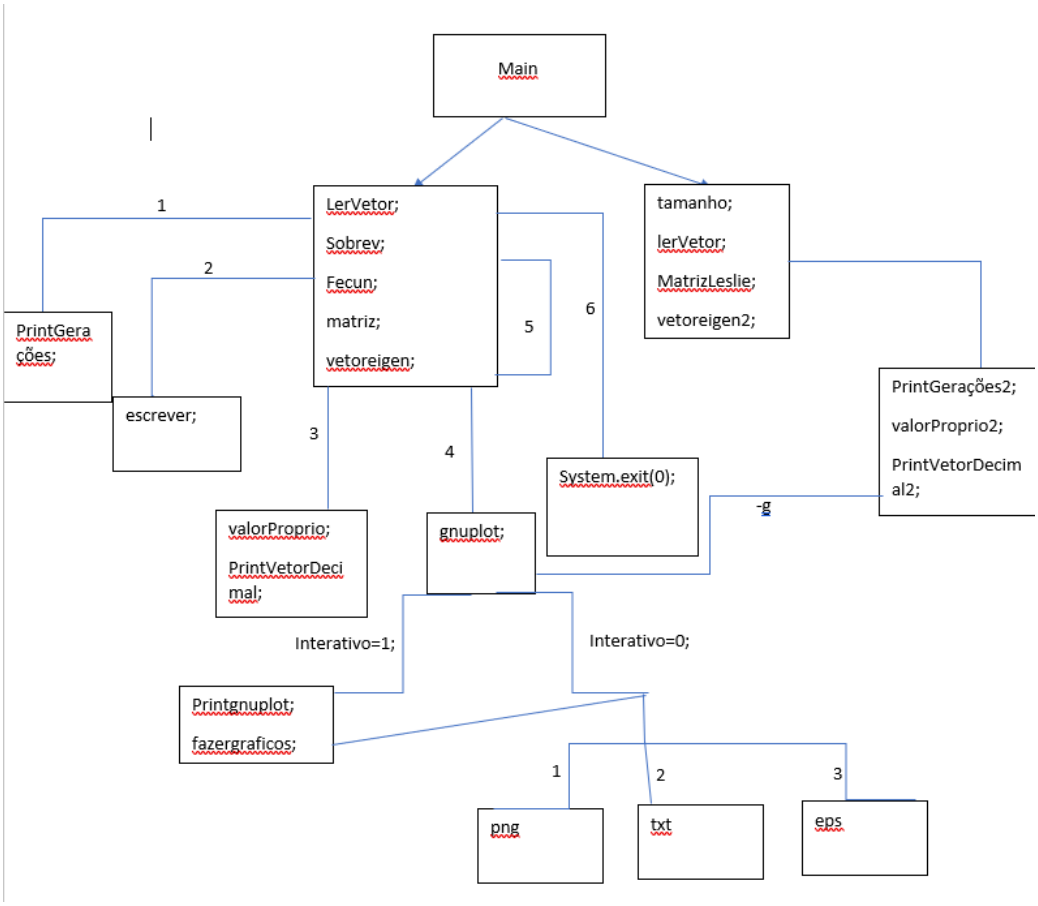
$$V = \begin{bmatrix} 0{,}549 \\ 0{,}297 \\ 0{,}125 \\ 0{,}030 \end{bmatrix}$$

Com estes cálculos é possível demonstrar a distribuição de uma espécie por classes (também normalizada), o seu número total de indivíduos e o seu comportamento assintótico.

Quanto ao seu comportamento assintótico é possível concluir que a população de linces, neste exemplo, irá aumentar já que o seu maior valor próprio (λ) > 1 e que a sua distribuição por classes quando estudado para espaço temporal infinito, irá ficar com a distribuição demonstrada em V.

4. Desenvolvimento e implementação da aplicação

O nosso programa possui o seguinte mapa:



Funcionalidades

PrintDimensãoPopulação

Este método recebe como parâmetro um double[][](matriz), um double[](vetor) e um int (t) e dá print a uma string com o seguinte formato: ("Dimensão da População = %.0f%n", DimensaoPopulacao(matriz,vetor,t). DimensaoPopulacao é um método que dá a dimensão da população a partir de uma matriz de Leslie e uma população inicial no instante t.

PrintTaxaVariação

Este método recebe como parâmetro um double(taxavariacao) e dá print a uma string com o seguinte formato: ("Taxa de Variação = %.2f%n", taxavariacao). Caso a variavél taxavariacao não seja um número(NaN), o método dá print à seguinte String:("Taxa de Variação = 1.00%n")

PrintVetorDecimal

Este método recebe como parâmetro um double[](vetor) e dá print a uma string *i* vezes, sendo *i* o tamanho do vetor(vetor.length o seguinte formato: ("%.3f%%\n", vetor[i]).

populacaonormalizada

Este método recebe por parâmetro um vetor(double[]) da população calcula a sua soma (população total), após isso utiliza os valores do vetor original e divide-os pela soma. Retorna um vetor(double[]) com os valores já normalizados.

PrintVetorPop

Dá print ao vetor com as populações, com as populações normalizadas separadas por um espaço.

Vetoreigen (criado com base no código fornecido no moodle)

O método vetoreigen recebe por parâmetro um array bidimensional double(double[][]). O método começa por tornar o array bidimensional numa "Matrix" (fornecida pela biblioteca la4j). Após isso cria um Eigendecompositor que vai decompor a matriz que foi feita no passado. O resultado de índice 0 é utilizado para criar um novo array de doubles bidimensional que vai ser uma matriz com vários vetores, sendo a primeira coluna os valores do vetor próprio com o maior valor próprio. Essa coluna vai ser toda somada numa variável double soma. Depois pega nesse vetor da primeira coluna e multiplica cada um dos seus valores por (1/soma). Após isso multiplica esses valores por 100. O vetor que este metodo retorna (double []) já é o valor multiplicado por 100.

ValorProprio (criado com base no código fornecido no moodle)

O método valorProprio recebe por parâmetro um array bidimensional double(double[][]). O método começa por tornar o array bidimensional numa "Matrix" (fornecida pela biblioteca la4j). Após isso cria um Eigendecompositor que vai decompor a matriz que foi feita no passado. O resultado de índice 1 vai conter um valor no índice de um array double [0] [0] que corresponde ao maior valor próprio da matriz que o método vai retornar.

MatrizLeslie

O método MatrizLeslie recebe por parâmetro dois arrays double (Fecundidade e sobrevivencia) que correspondem aos vetores com os valores da fecundidade e sobrevivência. Começa por criar uma matriz quadrada com o tamanho (Fecundidade.length*Fecundidade.length). Após isso atribui os valores do vetor fecundidade à linha 1 da matriz. Depois de aplicar esses valores vai colocar os valores de sobrevivência, começando no índice [1] [0], e colocando o resto nos índices [n+1] [n+1].

Lervetor (ficheiro texto)

O método lerVetor vai receber uma String que equivale a uma linha com os valores pedidos. O algoritmo começa por utilizar o split na String(s) recebida, criando um array de Strings(nums), em que os índices eram divididos por virgulas. Depois cria um array double(ds) com o tamanho do array de Strings (nums.length). Após isso o algoritmo vai percorrer o array nums, e em cada elemento vai procurar o caráter "=", quando encontrar esse caráter vai pegar nos caráteres seguintes (que supostamente são números) e vai torná-los num double, que vai ser incorporado no vetor final, retornado pelo método. O método retorna um array unidimensional double com o vetor lido.

Tamanho

O método tamanho é só utilizado quando é necessário ler um ficheiro de texto. Este método recebe por parâmetro o ficheiro que vai ser lido e vai ler o ficheiro até encontrar uma linha que começa por "x","f" ou "s", caso encontre com "x"ou "f", vai ver o tamanho (length) do vetor que forma através do split da String(tam[]) que equivale a linha; caso encontre o "s" vai fazer o mesmo, so que o tamanho vai ser equivalente ao tamanho do vetor +1(tam.length+1).

Escrever

O método escrever recebe como parâmetro um array double[][] (matriz) e vai percorrer todos os elementos da matriz, dando print a cada um com o formato" %.3f ", e cada vez que acaba de percorrer uma linha utiliza um System.out.println com uma String vazia para mudar de linha.

MultiMatrizPorVetor

Este método recebe com parâmetro um array double [][](matriz) e um array double [](vetor). O método começa por criar um array double [](vf) com o tamanho vetor.length(tamanho do double[] vetor). Após isso vai percorrer a matriz linha a linha. Cada vez que percorre uma linha vai pegar nos elementos dessa linha e vai multiplicar pelo valor correspondente no vetor (ou seja, matriz[nlinha][i]*vetor[i]), somando os resultados dessas multiplicações. Esse resultado vai ser atribuído ao índice equivalente à linha que tinha sido operada anteriormente, depois a variável soma é reduzida a zero, e assim, o ciclo vai-se repetir enquanto houver linhas.

TaxaVariacao

O método TaxaVariacao recebe por parâmetro um array double[][](matriz), um array double[](vetor) e um int(t). O método vai utilizar o método DimensaoPopulacao para calcular a dimensão da população no instante e divide pela dimensão do instante anterior(t-1). Após isso retorna o valor dessa divisão.

DimensaoPopulacao

O método DimensaoPopulacao recebe por parâmetro um array double[][](matriz), um array double[](vetor) e um int(t). O método começa por realizar os cálculos das multiplicações de matrizes e vetores com o método MultiMatrizporVetor até chegar ao vetor pedido da geração t. Após isso somam-se os valores todos do vetor e esse valor é retornado.

Multiplicar Uma Matriz Por Um vetor

Este método recebe por parâmetro um array double[][](matriz)e um array double[](vetor matriz coluna(array double []), e vai multiplicar os elementos de cada linha pelos respetivos elementos da matriz coluna, retornando um array double [](vf).

Reconhecer se é interativo ou não

O modo interativo tem duas formas de ser utilizado a primeira é só executando o programa pelo cmd sem argumentos e a segunda é executar o programa pelo cmd mas com argumentos ("– n " e um ficheiro de texto).

O modo não interativo só tem uma forma de ser utilizado, que é executando o programa pelo cmd e tem que ter obrigatoriamente como argumentos " -t" e de seguida um número, " - g " e um número de 1 a 3 ,como opção o usuário pode escrever os argumentos " - e ", " - v ", " - r ", que se escritos dão informações adicionais, no final têm que obrigatoriamente estar escritos primeiro o ficheiro de entrada e depois o ficheiro de saída onde estarão as informações que foram pedidas.

Para saber qual dos modos o utilizador escolheu o código analisa a variável args que é preenchida como um array quando o programa é executado pelo cmd. Para saber se é interativo o código verifica o tamanho da variável sendo que se tiver tamanho igual a zero significa que é para executar pela primeira forma do modo interativo, se for maior que zero pode ou ter argumentos inválidos, ou ser a segunda forma do modo interativo, ou ser o modo não interativo, para saber qual dos modos é o programa verifica o primeiro elemento do array .Se for "– n " o código verifica se o array tem um tamanho igual a 2 e se o segundo elemento termina em ".txt " se tiver significa que o programa é para ser executado pelo segundo modo interativo. Se o tamanho do array for maior que 6 então o código vai verificar os argumentos. A forma mais fácil que encontrei de verificar se tinha os argumentos obrigatórios sendo que eles podiam ter posições aleatórias foi procurar pelo "-t" e verificar que tem um número na posição seguinte do array depois disso o código procura o "-g" e verifica se tem um número na posição seguinte que está entre 1 e 3 de seguida o código verifica se tem o nome do ficheiro de entrada e de saída e por último é verificado se tem algum dos argumentos opcionais. Se o que foi escrito não se enquadrar em nenhum destes parâmetros o programa enviar uma mensagem a dizer qual é o problema.

printgnuplot

Preenche um ficheiro auxiliar .txt com os dados formatados em coluna, em que a primeira coluna é sempre igual às gerações e as próximas colunas possuem dados que variam consoante o gráfico pretende. Isto é importante já que assim que o programa *gnuplot* irá reconhecer os dados para fazer os gráficos.

Gnuplot

Com o intuito da apresentação de gráficos com os dados relativos a cálculos anteriores realizados pelo programa, utilizou-se uma estratégia específica para a elaboração desses mesmos gráficos.

Através do resultado de métodos de cálculo realizados anteriormente foi criado um método onde, consoante o gráfico que se quer criar foi usado o método "printgnuplot" para criar um ficheiro auxiliar txt. Após ser criado o ficheiro txt com estas especificações, este ficheiro é usado no método "fazergráficos" que recebe por parâmetro o tipo do formato, qual o gráfico a estudar, o nome da população, que foi pedido logo no início, e um parâmetro que diz caso o ficheiro já foi visto.

No modo interativo:

Com o parâmetro do tipo do formato é mudado as variáveis correspondentes ao terminal do gnuplot e à extensão do ficheiro a guardar consoante o que o utilizador pretendeu.

Com o parâmetro do tipo de gráfico os eixos do gráfico, nome a mostrar no gráfico, o título e a legenda são mudados de acordo com o tipo de gráfico que foi pedido.

Após isso, com essas informações, este método cria um ficheiro auxiliar (auxfile.gp) com o código a utilizar no gnuplot. Este código é variável, caso seja pedido para fazer um gráfico de distribuição da população por classes, quer ela seja normalizada ou não, o código do gnuplot é diferente caso apenas seja pedido a taxa de variação e a dimensão total.

Com isto, o Programa chama o método gnuplot que contem um menu onde o utilizador irá escolher o gráfico que pretende visualizar e consoante o que escolher, os parâmetros que irão para o gnuplot são mudados a fim de modelar o gráfico. Após isso é pedido o número de gerações a estudar e é gerado o gráfico em png para poder mostrar ao utilizador, chamando o método fazergráficos. Com isto é aberto o png gerado e é perguntado ao utilizador se pretende guardar o gráfico.

Caso queira, é perguntado ao utilizador que tipo de formato deseja guardar (png, eps, txt) e é passado um parâmetro com esta intenção. Tirando proveito disto o programa chama o "fazergráficos" novamente, mas desta vez sem um pedaço de código que possibilita a visualização. O ficheiro é gravado com a seguinte disposição: nome da população + gráfico desejado + data + .extensão .

Caso não queira, o ficheiro png que foi criado será apagado e irá voltar ao menu onde se escolhe o gráfico com a informação a simular.

Para sair desta funcionalidade dos gráficos basta escrever ("fim"), que é sempre dito quando se volta para o menu principal dos gráficos. Quando é escrito fim, o programa elimina os ficheiros auxiliares (auxfile.gp e dados.txt) e volta ao menu inicial do programa.

5. Casos de Estudo

Parágrafo introdutório, referindo o conteúdo desta secção.

Nesta secção irá ser realizado o estudo de dois casos independentes utilizando a nossa aplicação. Para isso, será apresentado os cálculos necessários para o estudo completo da população de uma espécie, utilizando o modelo de Lotka-Leslie e gráficos contendo informação essencial para chegarmos a conclusões de cada caso.

5.1. Caso de Estudo 1: Pombos em MarSol

Pombos

No primeiro caso de estudo vamos observar a evolução de uma população de pombos na cidade MarSol. A população contém 1000 juvenis, 300 jovens, 330 adultos e 100 maduros com as taxas de fecundidade e taxas de sobrevivência seguintes:

Os 4 valores que estão na primeira linha da matriz correspondem às taxas de fecundidade das faixas etárias respetivas. Os 3 valores que estão numa diagonal que começa na segunda linha correspondem às taxas de sobrevivência da mudança de uma faixa etária para a outra.

O seguinte estudo vai ser realizado com um intervalo de tempo de 100 gerações (é possível visualizar um estudo com 30 gerações aqui):

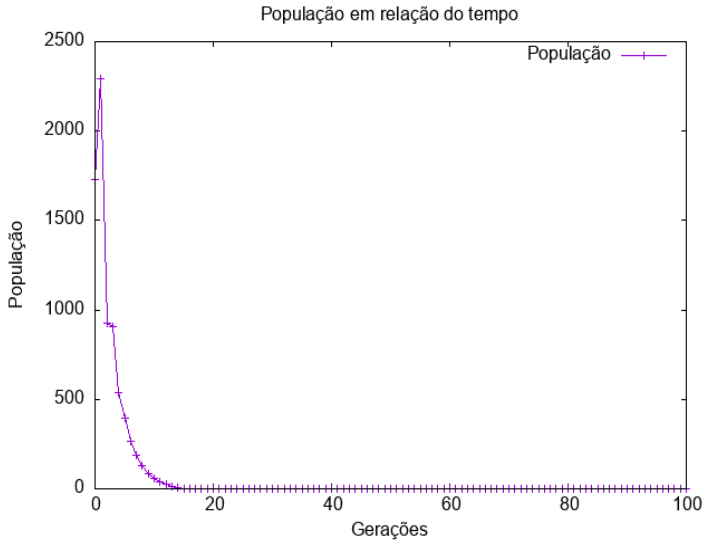


Figura 2 Dimensão da população total de pombos em MarSol

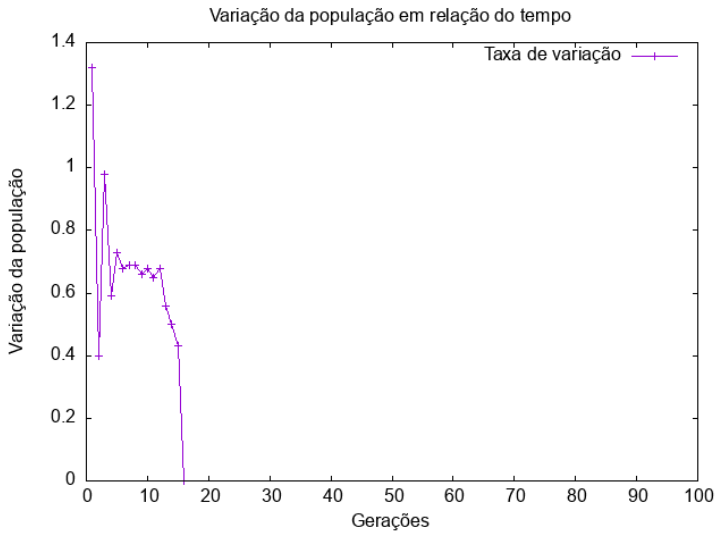


Figura 3 Taxa de Variação de pombos em MarSol

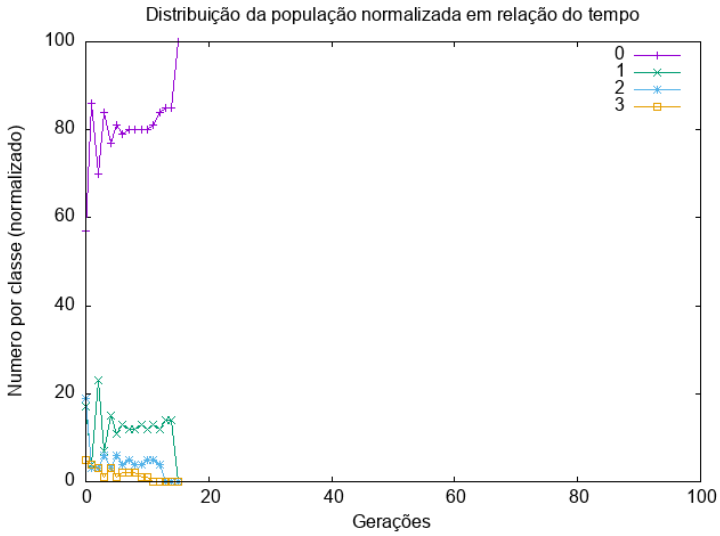


Figura 4 População normalizada de pombos em MarSol(em percentagem)

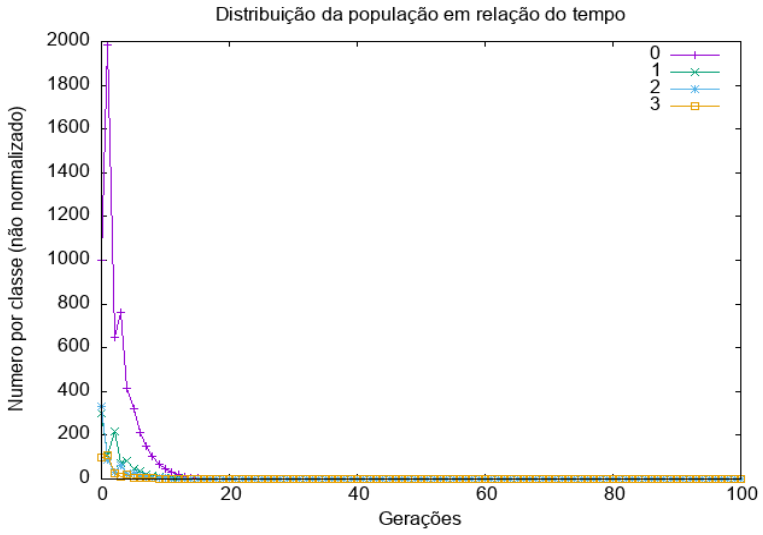


Figura 5 Distribuição da população de pombos em MarSol

5.2 Caso de Estudo 2: Gaivotas em MarSol

Gaivotas

No primeiro caso de estudo vamos observar a evolução de uma população de gaivotas na cidade MarSol. A população contém 600 juvenis, 200 jovens, 130 adultos e 40 maduros com as taxas de fecundidade e taxas de sobrevivência seguintes:

Os 4 valores que estão na primeira linha da matriz correspondem às taxas de fecundidade das faixas etárias respetivas. Os 3 valores que estão numa diagonal que começa na segunda linha correspondem às taxas de sobrevivência da mudança de uma faixa etária

O seguinte estudo vai ser realizado com um intervalo de tempo de 100 gerações(é possível visualizar um estudo com 30 gerações aqui):

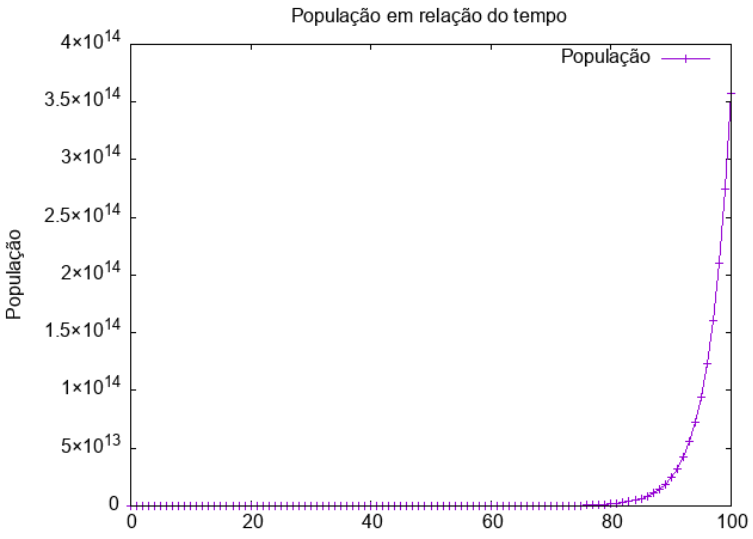


Figura 7 Dimensão da população total de gaivotas em MarSol

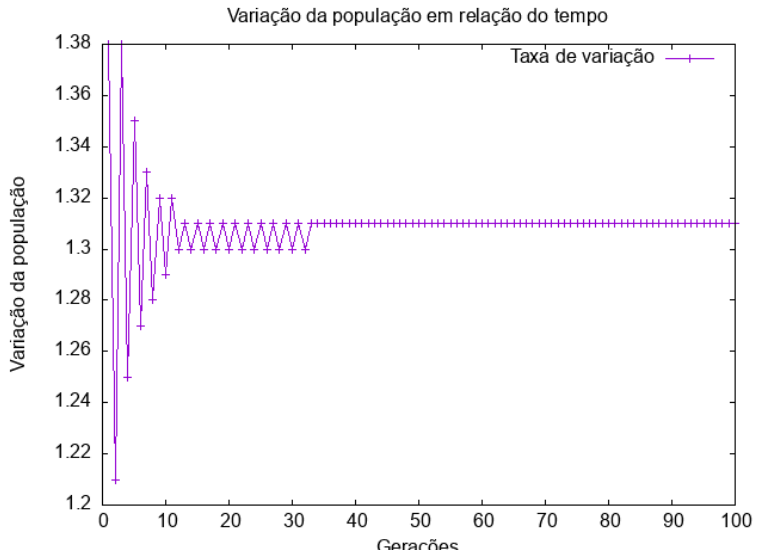


Figura 6 Taxa de Variação de gaivotas em MarSol

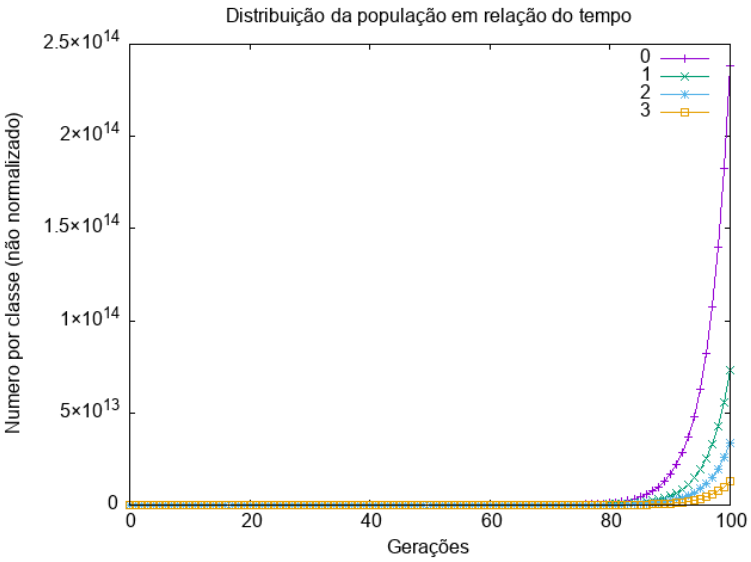


Figura 9 População normalizada de gaivotas em MarSol(em percentagem)

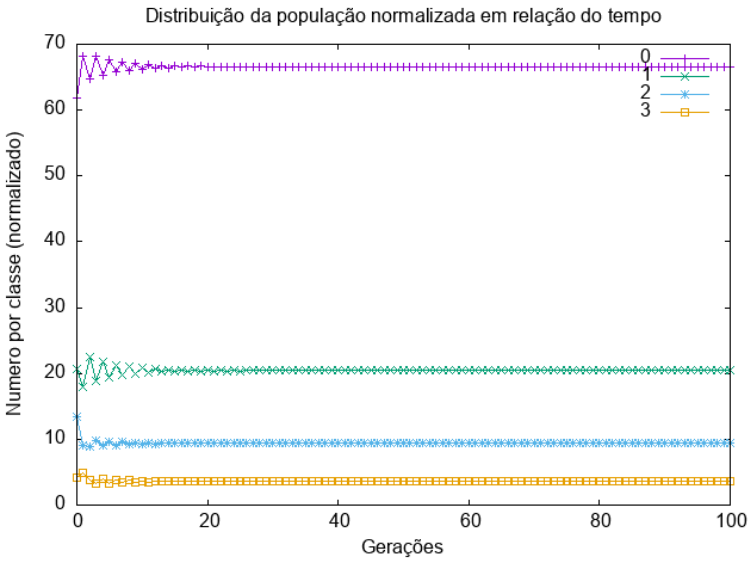


Figura 8 Distribuição da população de gaivotas em MarSol

5.3. Análise e discussão dos resultados

Discussão dos casos de estudo.

Dimensão da população total

No caso de estudo número 1 através dos dados podemos observar que a população total aumenta uma vez e depois vai descer até chegar a 0 na 16ª geração, o que significa que se extinguiu.

No caso de estudo número 2 através dos dados é possível ver que a população total vai subir até um número muito elevado, caso não haja alteração nas taxas de sobrevivência e de fecundidade das gaivotas.

Taxa de variação

No caso de estudo número 1 a taxa de variação vai-se alternado, estabiliza-se um pouco em torno das gerações 10-15, tornando-se 0 na 16ª, pois a população extinguiu-se.

No caso de estudo número 2 a taxa de variação vai-se alternando até se estabilizar permanentemente à volta da 35ª geração com um valor acima de 1, o que significa que o a população vai crescer sempre.

Distribuição da população

No caso de estudo número 1, o número de indivíduos juvenis é significantemente maior do que o resto e vai continuar assim até a sua extinção.

No caso de estudo número 2, o número de indivíduos juvenis é maior do que o reste e continuará assim até à geração 100.

Distribuição da população normalizada

No caso de estudo número 1, o número de indivíduos juvenis é de cerca de 80% da população, e vai continuar à volta desse valor até à extinção da espécie.

No caso de estudo número 2, o número de juvenis volta a ser maior do que os outros ao longo da história da população.

Conclusão

A diferença entre estes dois grupos é devida às diferentes taxas de sobrevivência, em que a gaivotas têm uma taxa de sobrevivência maior o que faz com que elas não se extingam. As taxas de sobrevivência são tão importantes que fazem com que as gaivotas cresçam apesar de terem uma fecundidade menor.

6. Conclusão

Nestas três semanas de projeto, podemos afirmar que o nosso conhecimento sobre todas as unidades curriculares aumentou consideravelmente, assim como o nosso entendimento da interligação entre todas as matérias lecionadas no primeiro semestre.

Além disso, adquirimos conhecimentos na área da Biologia, com o estudo e trabalho baseado no Modelo de evolução das espécies e na criação dos respetivos gráficos com o gnuplot.

Este trabalho permitiu aumentar o nosso dinamismo como equipa e permitiu a execução das diversas partes de um projeto como, análise do enunciado, planeamento, desenvolvimento do aplicativo, implementação de métodos e testes, acompanhada da elaboração de um relatório.

Esta atividade permitiu-nos adquirir experiência prática na nossa área, e acreditamos que esta nos vai ser útil quando formos inseridos no mercado de trabalho. Foi um projeto que visou simular esse mesmo contexto, e a inserção de um cliente com reuniões semanais foi, na nossa opinião, muito bem conseguida. A orientação dos professores foi muito boa, pois ajudou-nos a tirar algumas dúvidas e ao mesmo tempo, deixou-nos trabalhar por nossa própria responsabilidade.

Acreditamos que estamos numa área cada vez mais competitiva e que existem fatores que podem nos diferenciar dos demais, graças a este tipo de projetos. Alguns desses fatores são a obtenção de know-how específicos (bastante requisitado no mercado das TI), o trabalho em equipa, a divisão em sprints (também descobrimos que existe a profissão específica de Scrum Master) e a rentabilização de esforço e tempo.

Salientamos o bom clima de trabalho que existiu entre nós, todos se propuseram a realizar qualquer tarefa, e houve muita interajuda e comunicação. Adquirimos não só mais-valias profissionais, mas também competências pessoais e comunicativas.

Resumindo, este projeto fez-nos consolidar os conhecimentos adquiridos no primeiro semestre e podemos afirmar que o nosso domínio do Java aumentou. Tivemos também ganhos de experiência, que nos podem ser úteis no futuro, graças à elevada componente prática deste trabalho. Adquirimos ainda competências em relação a dinamismo e comunicação em equipa. Acreditamos que como a programação teve um fim específico, conseguimos visualizar uma forma de realizar e executar uma aplicação. Estes tipos de projetos ajudam muito os alunos pois dão-nos a oportunidade de crescer no quesito técnico, e podemos afirmar que foi um prazer fazer esta aplicação.

Referências

Math, C. (2017). Section 3.5 Leslie Matrices https://youtu.be/XrWEgvGYIFI

Scrum - Scrum Master Portugal. (n.d.). Retrieved January 16, 2021, from http://www.scrumportugal.pt/scrum/

ScrumGuides.org (n.d.). Retrieved January 16, 2021, from https://www.scrumguides.org/index.html

ScrumGuides.org (n.d.). Retrieved January 16, 2021, from https://www.scrumguides.org/index.html

Scrum.org – The home of Scrum (n.d.). Retrieved January 16, 2021, from https://www.scrum.org/resources/what-is-scrum

Scrum.org – The home of Scrum (n.d). The Scrum framework Poster [Poster]. Lisboa: INE

Codeco, P. (2018) Modelo matricial de Leslie: estudo do crescimento populacional do Brasil e estados do Espírito Santo e Acre. Universidade Estadual de Campinas.

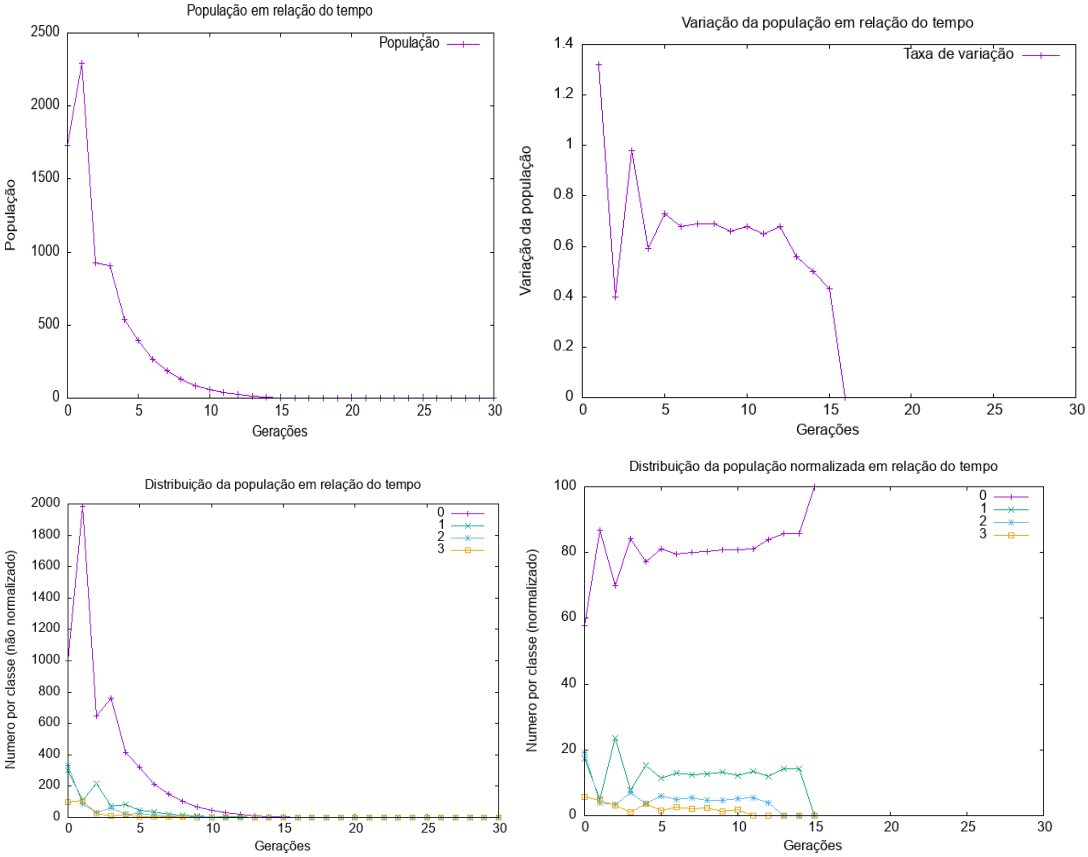
Howard, R. (n.d) A Note on Eigenvalues and Eigenvectors of Leslie matrices. Department of Mathematics, University of South Carolina.

ANEXOS

ANEXO A

(Incluir os anexos que considerarem relevantes.)

30 gerações Pombos:



30 Gerações Gaivotas