UNIVERSIDADE ESTADUAL PAULISTA

Faculdade de Ciências - Bauru

Bacharelado em Ciência da Computação

Luiz Fernando Ferreira Gomes de Assis

Estudo e Simulação de Redes de Sensores em Superfícies Não-Planas

UNESP

2012

Luiz Fernando Ferreira Gomes de Assis

Estudo e Simulação de Redes de Sensores em Superfícies Não-Planas

Orientador: Prof. Dr. João E. M. Perea Martins

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

UNESP

2012

Luiz Fernando Ferreira Gomes de Assis

Estudo e Simulação de Redes de Sensores em Superfícies Não-Planas

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

BANCA EXAMINADORA

Prof. Dr. João E. M. Perea Martins
Professor Doutor
DCo – FC - UNESP – Bauru
Orientador

Profª. Drª. Simone das Graças Domingues Prado
Professora Doutora
DCo – FC - UNESP – Bauru

Prof. Dr. Wilson Massashiro Yonezawa
Professor Doutor
DCo – FC - UNESP – Bauru

Bauru, 12 de Novembro de 2012.

Dedico o trabalho a minha família, por me apoiar incondicionalmente em todos os momentos.

AGRADECIMENTOS

Agradeço inicialmente a Deus por saber que existe algo ou alguém que nos presenteia diariamente com as dádivas da vida. Também a Jesus Cristo, pois com muita fé e força realizei esse trabalho.

A minha família por ter me apoiado e ao meu orientador, o Prof. Dr. João E. M. Perea Martins, que me auxiliou durante a universidade quando precisei, não somente com conhecimentos técnicos, mas com conselhos pessoais.

A todos os meus professores que me ensinaram os conhecimentos necessários para a minha formação acadêmica e me mostraram o quão gratificante pode ser o exercício de minha profissão.

E por fim, a todos os meus amigos.

RESUMO

Essa monografia do Trabalho de Conclusão de Curso do Bacharelado em Ciência da Computação define o estudo e a simulação de redes de sensores em superfícies não-planas. A monografia é composta por uma introdução do tema de pesquisa, que engloba um estudo teórico, o estado da arte e a contextualização histórica das redes de sensores. A simulação é constituída pela geração do terreno, a distribuição aleatória dos nós e a geração e transmissão dos pacotes a partir dos nós. Com base nessas três importantes partes, a troca de informações/pacotes entre os nós acontece através do algoritmo de busca em largura. São definidos os nós ativos na rede através desse algoritmo, a quantidade de nós e o raio de alcance de cada nó. Depois de verificado a ativação de cada nó, os pacotes são gerados e transmitidos aos nós seguintes. Esse processo ocorre inúmeras vezes e ajuda na análise de parâmetros existentes nas redes de sensores em superfícies não planas.

PALAVRAS-CHAVE: REDES DE SENSORES, SIMULAÇÃO, SUPERFÍCIES NÃO PLANAS

ABSTRACT

This Bachelor's Thesis of Bachelor of Computer Science defines a research and a network sensor simulation on non-planar surfaces. The report is composed of an introduction of the research, a theoretic study, a state of the art and a historic context of sensor network. The simulation consists of the formation of terrain, node's random distribution and a production and a transmission of the node's packages. Based on these three important topics, the exchange of information/packages between multiple nodes is through breadth-first search algorithm. The active nodes, node quantity and operation range are also defined in the program. After the program analysis the node activation, the packages are created and transmitted to the next node. This process occurs many times and help on the analysis of the sensor network on non-planar surfaces parameters.

KEYWORDS: SENSOR NETWORKS, SIMULATION, NON-PLANAR SURFACES

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

RSSF	Redes de Sensores sem Fio
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
MP1	Macroprograma 1
OSI	Organização Internacional para a Normalização
MATLAB	Matrix Laboratory

SUMÁRIO

1 INTRODUÇÃO

A área de redes de sensores sem fio está em constante evolução na área da computação, e geralmente ela é utilizada para monitorar parâmetros físicos de um determinado sistema. O desenvolvimento dos dispositivos de hardware e a crescente utilização desse tipo de rede contribuem para o seu avanço tecnológico.

As redes de sensores sem fio (RSSFs) são classificadas de acordo com sua aplicação e sua finalidade. As RSSFs foram desenvolvidas para atuarem em lugares de difícil acesso, os quais não são viáveis a utilização de fios e precisam-se das informações em tempo real.

Geralmente as RSSFs se baseiam em aquisição e distribuição de dados, monitorados e controlados por uma base central. As figuras 1 e 2 mostram a complexidade associada as redes de sensores sem fio.

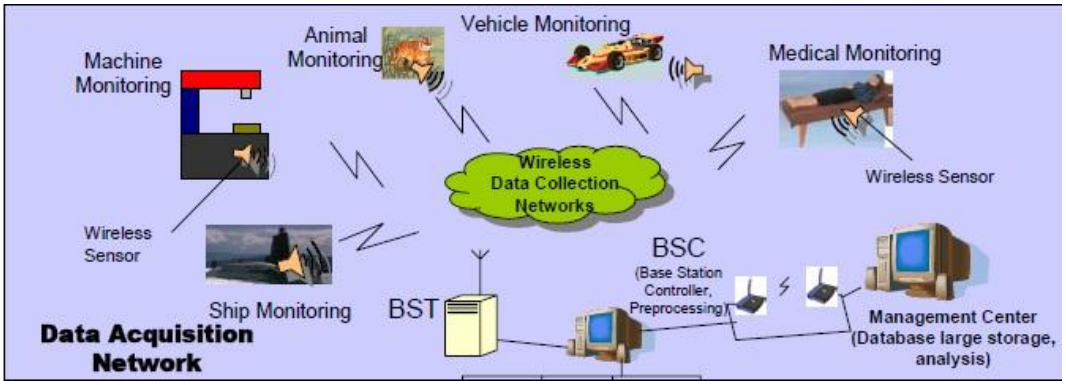


Figura 1 – Rede de Aquisição de Dados (LEWIS, 2004)

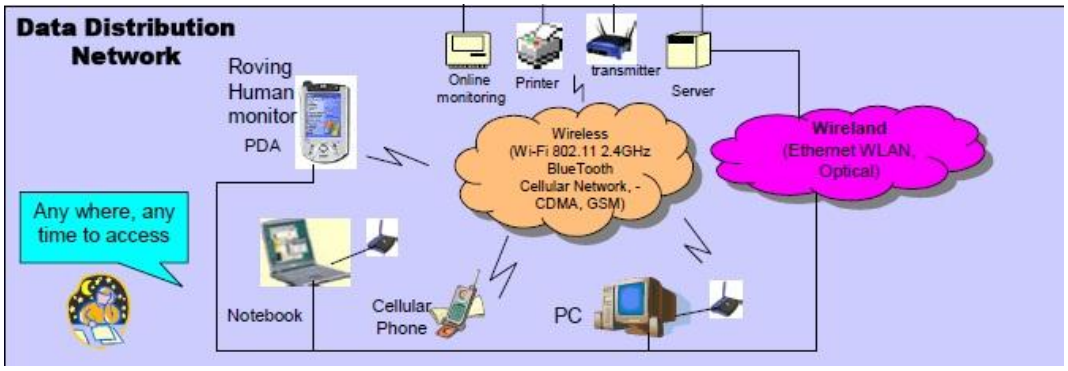


Figura 2 – Rede de Distribuição de Dados (LEWIS, 2004)

Os sensores são os elementos básicos, eles se comunicam e trocam informações entre si conforme a necessidade do sistema. O poder computacional e a capacidade de energia dos sensores são limitações que devem ser tratadas na rede.

Uma rede é composta por sensores distribuídos. Os sensores também conhecidos por nós são responsáveis por realizarem a coleta e o processamento dos dados, enviando assim a um ponto de acesso.

Em tese, a transmissão de dados sem fio existe a muitos anos, podem-se citar os telefones celulares, satélites e redes de sensores como exemplos de dispositivos que fazem uso desse tipo de transmissão.

Por volta de 1820, Hans Christian Oersted percebeu que as correntes elétricas produziam campos magnéticos, denominados depois de campos de ondas eletromagnéticas. O telégrafo foi o primeiro dispositivo a utilizar da troca de informações sem um meio físico através dessas ondas, o código funcional dessa troca era denominado Morse.

Atualmente a eletrônica viabiliza o desenvolvimento da comunicação sem fio, que por sua vez permite a interconexão de dispositivos. Existem diversos aparelhos que fazem uso dessa tecnologia, como por exemplo, os *notebooks*.

Os notebooks possuem receptores e transmissores para a formação de redes sem fio, sendo o IEEE (Institute of Electrical and Electronics Engineers) responsável pela padronização das características dessas redes.

Segundo Neto (2009), algumas pesquisas da EMBRAPA (Empresa Brasileira de Pesquisa Agropecuária) acontecem com base em redes de sensores. A EMBRAPA já divulgou um boletim de pesquisa e desenvolvimento de redes de sensores sem fio e computação ubíqua na agropecuária, o qual faz parte do projeto de pesquisa em rede MP1 “Agricultura de precisão para a sustentabilidade de sistemas produtivos do agronegócio brasileiro”.

Tal boletim de pesquisa mostra algumas atividades monitoradas e controladas de processos na agropecuária, os quais se podem citar a irrigação espacialmente diferenciada, a pulverização de precisão, o mapeamento da fertilidade do solo, a rastreabilidade animal e vegetal e por fim as mudanças climáticas e problemas fitossanitários.

As RSSFs podem ser empregadas para medir características que ambientes externos apresentam, como temperatura, pressão, umidade, etc. Também podem analisar eventos e informações de elementos nele presentes.

Em ambientes inteligentes, o sistema troca informação entre si e altera uma determinada situação. Por exemplo, em aplicações domésticas, um homem assim que abre a porta de sua casa, provoca o início da execução de uma determinada música que ele gosta. Isso pode ocorrer através da transmissão de dados de sensores posicionados na porta e no som.

No meio militar também existem diversas formas das redes serem utilizadas. No monitoramento de movimento de tropas e de terrenos, em armas, prevenção e/ou detecção de ataques, entre outros.

Na medicina podem ser utilizados para monitorar os órgãos e diagnosticar doenças como na figura 3.

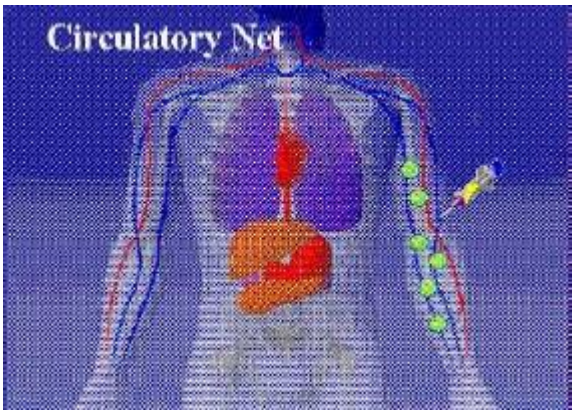


Figura 3 - Sensores dentro do corpo humano (LOUREIRO, 2003)

Também podem fornecer informações importantes em grandes centros urbanos, como por exemplo, através do tráfego de veículos, monitorando o fluxo do trânsito e a velocidade dos veículos.

Muitas indústrias estão utilizando das redes de sensores sem fio para coletar informações em pontos críticos e muitas vezes nem chegam a implementação desse sistema, simplesmente simulam essa rede por causa de fatores econômicos e temporais.

O processo de simulação das RSSFs é realizado através de modelos matemáticos com o intuito de reproduzir o funcionamento do sistema real, facilitando assim o entendimento de seu comportamento sem necessariamente ter que implementá-lo.

Qualquer aplicação que envolva a medição de um dado ou troca de informações entre dispositivos pode ser implementada através das redes de sensores sem fio.

A respeito das motivações de se desenvolver o trabalho, Fábio Gandour, cientista chefe da IBM Brasil, recentemente informou a tendência de se utilizar sensores em todos os objetos daqui a alguns anos.

Por exemplo, a quantidade de endereços obtidos através do protocolo IPv4, de 32 bits, é de aproximadamente 2^{32} (4.294.967.296) endereços, os quais estão se esgotando. Uma das maneiras de se contornar esse problema é realizando sua expansão através do protocolo IPv6.

O IPv6 é composto por $3{,}4 \times 10^{38}$ endereços. Para se ter uma ideia do quão grande é esse número, se dividirmos esse valor pela área em m^2 da superfície da Terra, obteremos um valor aproximado de 3×10^{24} endereços/m^2, o que possibilitaria endereçar muitos objetos por metro quadrado.

Nessa situação o sensor está presente na troca de informações e na verificação do estado do objeto, ou seja, objetos que podem ser representados através de estados possivelmente conterão sensores, como por exemplo, portas (aberta ou fechada), lâmpadas (acesa ou apagada), etc.

Segundo Colwell (2012), dados recentes mostram que o mercado de sensores sem fio nos últimos dois anos dobrou e que em 2016, 24 milhões de pontos de sensoriamento estarão habilitados. Consequentemente, 39% das novas aplicações, serão unicamente habilitadas pela tecnologia RSSF.

Tais dados podem ser vistos na figura 4.

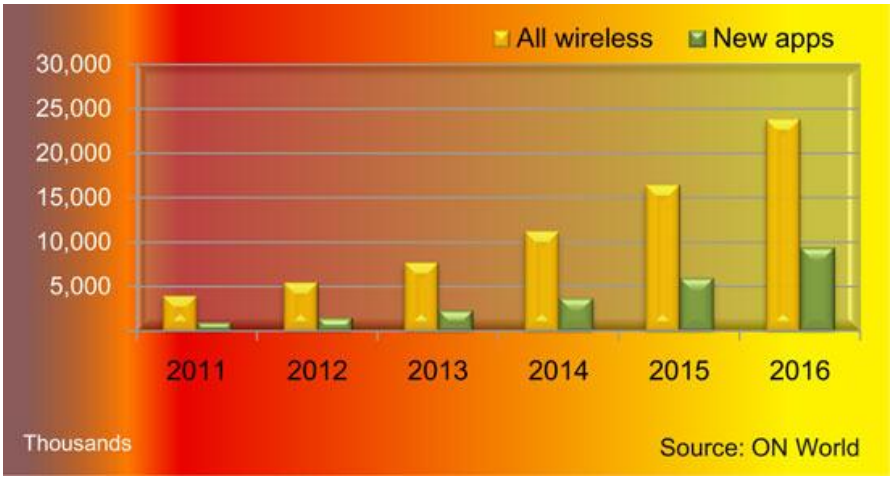


Figura 4 – Wireless Sensor Network Trends (COLWELL, 2012)

1.1 Proposta do Trabalho

Neste trabalho é proposto um estudo e uma simulação de redes de sensores em superfícies não planas. Inicialmente foram sugeridas pesquisas da área e um estudo das redes de sensores, baseados em aspectos históricos, estado da arte e contextualização.

O estudo das redes de sensores deu suporte a simulação, pois com base na fundamentação teórica das características das redes, como por exemplo, definições, áreas de aplicação, características de seus componentes, suas classificações quanto a configuração, ao sensoriamento, ao tipo de comunicação e ao tipo de processamento, entre outros, auxiliou a execução da simulação ser realizada de maneira mais eficiente.

Em relação aos aspectos históricos e o estado da arte, a evolução e o que pode se esperar da área influenciou na diretiva de pesquisa, pois tendências e contextualização histórica foram importantes para o desenvolvimento do projeto.

Na parte de simulação das redes de sensores, o desenvolvimento do algoritmo era constituído de três importantes partes, a simulação do terreno, a distribuição dos nós e a geração de pacotes.

O algoritmo de roteamento engloba as três partes, pois dependeria da disposição do terreno, como os nós estariam distribuídos nele e como o nó gerenciaria os pacotes gerados em relação ao seu sensoriamento e a sua

transmissão, balizado por um algoritmo que fizesse a informação ser percorrida de maneira otimizada.

Para a simulação do terreno, distribuição dos nós e a geração de pacotes, modelos de distribuições de números aleatórios foram propostos a serem analisados. Tais modelos correspondem respectivamente ao modelo de distribuição Normal e Uniforme. Ambos modelos de distribuição de variáveis aleatórias contínuas, e o modelo de distribuição de Poisson, modelo de distribuição de variável discreta.

O estudo desses modelos juntamente com a efetiva aplicação no processo de simulação através de funções da linguagem utilizada dariam o respaldo necessário para sua implementação.

A linguagem para a implementação definida no início do trabalho foi o MATLAB, por se tratar de uma linguagem de programação apta a realizar operações matemáticas avançadas.

Importante ressaltar que todas as atividades propostas em relação ao cronograma foram cumpridas.

Dentre as atividades estão os testes, fase em que o algoritmo utilizado para efetuar a simulação foi testado em várias situações, com o objetivo de encontrar possíveis erros, observar melhorias em modificações e verificar o melhor caminho para o seu desenvolvimento.

Lembrando que a fase de testes aconteceu paralelamente ao desenvolvimento e a otimização.

A fase de otimização foi realizada depois do período de desenvolvimento, essa foi a fase em que melhorias foram feitas no intuito de tornar o processo de simulação mais eficiente, de maneira que a quantidade de redundâncias de informações encontradas seja ínfima.

A fase de análise dos resultados foi a fase em que os resultados obtidos tanto no processo de desenvolvimento, quanto na fase de testes e na fase de otimização amparam futuros projetos, mostram quais foram as tendências da simulação, se existiram, e ajudaram na fase de conclusões do projeto.

As conclusões foram baseadas em todas as atividades propostas e realizadas, descrevendo todas as variações e constantes relacionadas ao projeto.

As conclusões basicamente definiram se o projeto foi executado de maneira satisfatória, se apresentou características similares em relação as redes de sensores reais, se os resultados obtidos foram condizentes com a maneira como o algoritmo foi implementado, se os testes apontaram alguma informação contraditória, quais podem ser possíveis melhorias para futuros projetos, entre outros.

1.2 Objetivos do Trabalho

O objetivo geral é o objetivo central do trabalho. O projeto será desenvolvido realizando atividades em torno de um propósito. Tal propósito é explicado e definido no próximo sub-tópico.

Enquanto que os objetivos específicos dizem respeito a atividades específicas efetuadas para que o objetivo geral seja atingido.

1.2.1 Objetivo Geral

Avaliar os parâmetros e desempenho da rede de sensores, de maneira que se consiga otimizar o processo observado, respeitando suas restrições e limitações.

1.2.2 Objetivos Específicos

Simular a distribuição dos nós em uma superfície não-plana, conseguindo assim uma máxima conectividade entre eles, independente do raio de alcance e da quantidade de nós naquela área.

Estudar o terreno explorado, para que não haja tanta interferência do terreno na comunicação entre os nós.

Elaborar um algoritmo de comunicação entre os nós, que minimize a redundância de informação trafegada na rede.

Analisar parâmetros da rede como tamanho de fila.

Monitorar os nós de maneira que a comunicação não se prejudique com a alteração da quantidade de nós, fazendo com que a rede seja auto-organizável, ou seja, controle tal situação.

Definir um ótimo raio de alcance para uma determinada quantidade de nós, e definir uma quantidade ótima de nós para um determinado raio de alcance.

1.3 Organização da Monografia

O Capítulo 1 contém uma introdução do tema da pesquisa, que engloba o estado da arte e uma contextualização histórica das redes de sensores. O Capítulo 1 contém também os objetivos da pesquisa, divididos em geral e específico e a estrutura da monografia.

No Capítulo 2 é apresentada a fundamentação teórica das redes de redes de sensores. Em relação a simulação das redes de sensores são apresentados todos os modelos de distribuições de números aleatórios envolvidos na geração do terreno, na distribuição dos nós, e na geração dos pacotes. No Capítulo 2 também é apresentado o algoritmo de roteamento escolhido para a simulação das redes.

No Capítulo 3 é apresentada a proposta do trabalho. No Capítulo 4 são apresentados os materiais utilizados durante o desenvolvimento do projeto e os métodos implementados para se conseguir todos os resultados.

No Capítulo 5 são mostrados alguns experimentos realizados, variando situações que a rede poderia ser encontrada. No Capítulo 6 são apresentados alguns resultados encontrados. Tais resultados são providos de três exemplos com diferentes topologias de rede.

No capítulo 7 são apresentadas as conclusões obtidas através dos experimentos realizados e resultados encontrados.

Por fim, são apresentadas as referências bibliográficas e dois apêndices que contém os programas implementados na simulação.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Rede de Sensores

Uma rede de sensor possui sensores que trocam informações entre si, cada nó não necessariamente possui apenas um sensor. Geralmente possuem um baixo aproveitamento computacional e pouca capacidade de energia.

A rede de sensores é utilizada para aplicações com lugares de difícil acesso. Os nós são lançados através de dispositivos mecânicos, formando uma rede ad hoc (rede descentralizada sem infraestrutura pré-definida).

Cada nó captura o dado referente a sua área de atuação e transmite aos outros de maneira multi-hop, ou seja, a informação salta de um nó a outro.

A rede pode ser dividida em três partes: os nós, o receptor da informação final e a informação analisada.

De acordo com Martins (2010) um nó é o elemento básico da rede e é constituído de quatro componentes, o que pode ser visto na Figura 5.

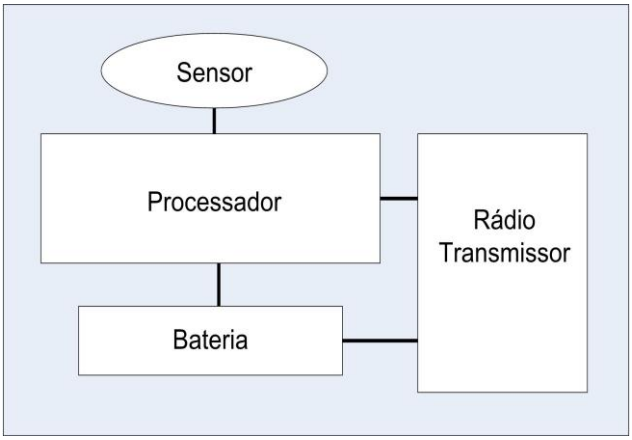


Figura 5 – Componentes de um nó (MARTINS, 2010)

Os quatro componentes são identificados como o sensor, responsável pela coleta dos dados, o processador que realiza o processamento da informação, a

bateria que é o componente que gerenciará a energia dos nós e o rádio transmissor que transmitirá a informação ao nó seguinte.

O receptor da informação final pode ser um usuário ou um nó sink (nó servidouro). Segundo Luo (2009, p.79) um nó sink é o nó responsável por capturar e gerenciar os processos nas redes de sensores. A informação capturada dependerá da aplicação e do objetivo das redes de sensores.

A distribuição dos nós acontece aleatoriamente em relação ao plano, porém em relação a altimetria o posicionamento dos nós dependerá da topologia do terreno.

Podemos visualizar uma distribuição aleatória na Figura 6.

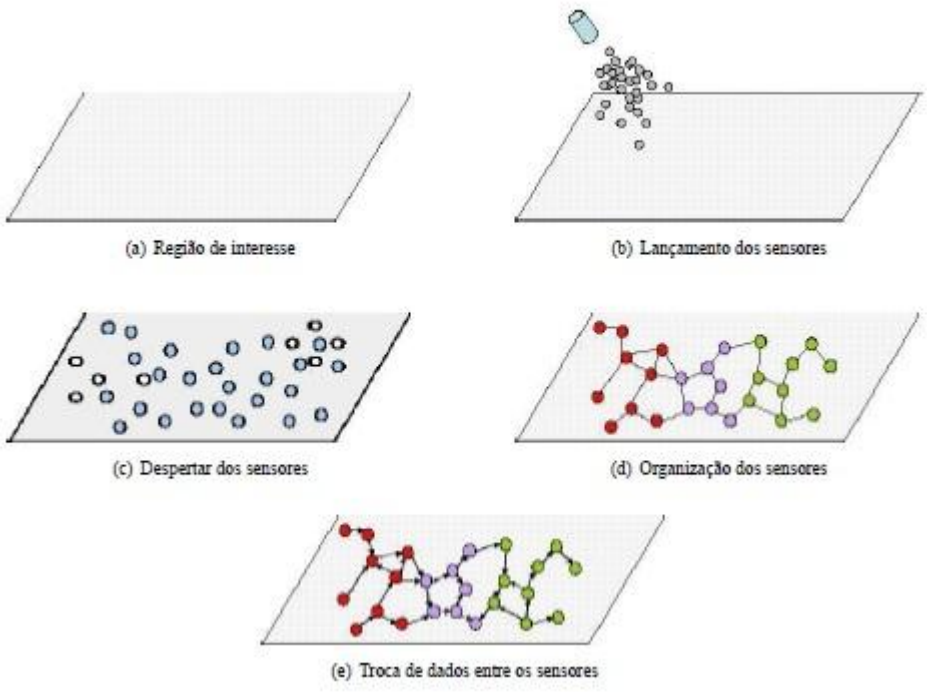


Figura 6 – Distribuição dos nós (LOUREIRO, 2003)

De acordo com Araújo (2012) a classificação das redes de sensores pode se dar de quatro formas, através da configuração, do sensoriamento, do tipo de comunicação e do tipo de processamento.

Dentre elas, existem diversas maneiras de se classificar cada uma das formas.

A classificação por configuração acontece de várias maneiras. Entre elas estão a classificação feita por composição homogênea e heterogênea, sendo classificados

através da capacidade de hardware que eles apresentam. A classificação feita também pela organização, sendo hierárquica ou plana, nós que possuem ou não importância maior em relação a outros.

Por mobilidade estacionária ou móvel, classificando assim a mobilidade dos nós. Por densidade, podendo ser balanceada, densa ou esparsa, avaliando a concentração dos nós posicionados.

Pode ser classificada por distribuição irregular ou regular, sendo a distribuição uniforme ou não.

A classificação através do sensoriamento também é feita de diversas maneiras. A coleta pode ser periódica, contínua, reativa ou em tempo real. Os sensores podem captar os dados em intervalos regulares, continuamente, quando algum evento acontece ou a todo o momento.

A classificação pelo tipo de comunicação pode ser dividida por disseminação dos dados, sendo programada, contínua ou sob demanda. Por tipo de conexão, simétrica ou assimétrica, os nós podem se diferenciar por alcance das conexões.

Por transmissão, simplex, half-duplex e full-duplex. Essas transmissões se diferem, no caso do simplex os sensores apenas recebem a informação; no half-duplex recebem e enviam, mas não ao mesmo tempo; e no full-duplex, recebem e enviam ao mesmo tempo.

Por alocação de canal, sendo estática ou dinâmica, ou seja, o nó possui ou não uma atribuição da largura de banda, havendo ou não preferência na transmissão, minimizando assim possíveis interferências. E também pode ser classificado pelo fluxo de informação.

E por fim, a classificação através do processamento pode ser feita pelo critério de cooperação, sendo opções a infraestrutura, localizada e a correlação.

Araújo (2012) ressalta também os protocolos de rede. Na sua camada física, uma rede de sensor pode enviar dados através da comunicação ótica, infravermelho ou rádio de frequência. Na camada de enlace o gerenciamento da energia e a auto-organização fazem parte do controle de acesso ao meio.

A camada de rede é responsável pelas rotas dos envios de dados através do algoritmo de roteamento, com algumas limitações relacionadas as RSSFs. Muitas vezes as camadas de transporte não são necessárias.

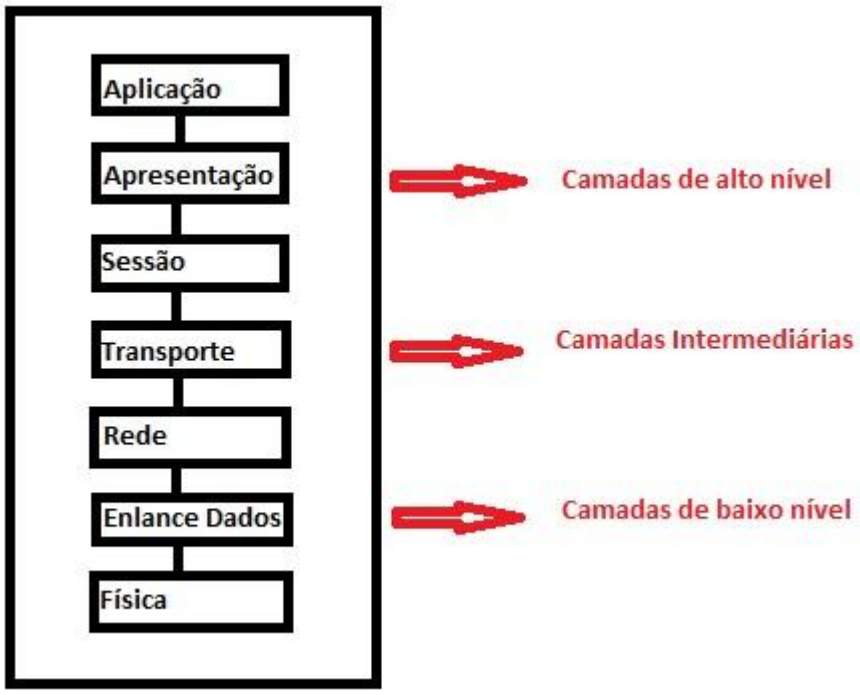


Figura 7 – Modelo OSI (PAPA, 2010).

A figura 7 representa as sete camadas de uma rede de computadores de acordo com o modelo OSI (Open Systems Interconnection).

Continuando a análise de Araújo (2004), a importância da segurança nas redes é fundamental para aplicações onde a confiabilidade das informações é necessária.

Atualmente as redes de sensores sem fio para monitoramento de habitats é uma aplicação muito pesquisada e utilizada para identificar importantes áreas de trabalho em amostras de dados, comunicações, refinamento da rede e monitoramento do estado da rede.

O baixo nível de energia dos nós ainda é uma restrição a ser tratada, combinada com os requisitos da transferência de dados. Os limites escassos de energia e a necessidade de operações previsíveis guiam o desenvolvimento da arquitetura da aplicação e dos serviços.

A aplicação define os serviços dos sensores da rede através da coleta e das amostras dos dados, das comunicações, do refinamento da rede e do monitoramento do estado da rede.

De acordo com Hill (2002) da universidade de Berkeley na Califórnia, MICA é uma plataforma sem fio profundamente incorporada as redes, que possui baixo poder de integração e de sensoriamento. Um exemplo de MICA pode ser visto na Figura 8.

Figura 8 – Plataforma MICA (MAINWARING, 2002).

2.2 Simulação de Rede de Sensores

Para realizar a simulação das redes de sensores, distribuições aleatórias serão abordadas para facilitar o entendimento da geração do terreno onde os nós serão posicionados; da distribuição uniforme dos nós, maneira como a qual eles serão distribuídos; e da geração dos pacotes realizadas por eles, como ocorrerá o surgimento do pacote em um determinado instante.

Na geração do terreno a distribuição estudada será a distribuição normal, pois a compreensão de tal distribuição ajuda na formação das curvas. Por exemplo, para a formação de montanhas e vales, a altura, a profundidade e a inclinação dessas irregularidades no terreno irão influenciar diretamente a comunicação dos nós. A distribuição normal oferece parâmetros para tratar tais características da curva.

O posicionamento dos nós acontecerá de maneira uniforme, sendo assim essencial o estudo dessa distribuição.

Enquanto a geração de pacotes ocorrerá de acordo com a distribuição de Poisson. Tal distribuição é utilizada, pois ela fornece valores compatíveis com a realidade.

Nas redes de sensores os pacotes são gerados através de rajadas, ou seja, em um determinado momento pacotes não são gerados e em outros são gerados três ou quatro pacotes.

2.2.1. Modelos de Distribuições de Números Aleatórios

Na estatística, as variáveis aleatórias são utilizadas para representar os números associados aos eventos estudados.

Segundo Ribeiro (2008) as variáveis aleatórias podem ser divididas em dois grupos, as discretas e as contínuas. As variáveis aleatórias discretas são obtidas a partir de uma contagem e são representadas por números inteiros positivos, partindo do número zero. Enquanto as contínuas são utilizadas para medições.

2.2.1.1. Distribuições de Variáveis Aleatórias Discretas

Existem diversas distribuições de variáveis aleatórias discretas, podem-se citar como exemplos: as distribuições de Bernoulli, Binomial e de Poisson.

O modelo de distribuição de Poisson é utilizado para a geração de pacotes na simulação de redes de sensores desse trabalho.

Esse modelo é utilizado, pois a geração de pacotes não é feita de forma uniforme, e sim através de rajadas. Ou seja, em um determinado momento pode não haver geração de pacotes, mas em outros podem ser que quatro, cinco ou seis pacotes sejam gerados.

Na Seção 2.2.1.1.1, tal distribuição é explicada.

2.2.1.1.1. Distribuição de Poisson

A distribuição de Poisson possui uma função de distribuição de probabilidade, a qual associa cada valor que a variável aleatória assume a uma probabilidade. Esse valor corresponde à quantidade de ocorrências de um determinado evento em um determinado tempo.

Segundo Ribeiro (2008) a distribuição possui algumas características como:

- Os eventos são interdependentes entre si.

- O número médio de ocorrências por unidade de tempo é constante.
- O número de ocorrências está associado ao tamanho do tempo.

Essa função pode ser visualizada em (1).

$$P(X) = \frac{\lambda^{X} * e^{-\lambda}}{X!} \quad (1)$$

Em (1), *λ é a t*axa de ocorrência, X é o valor da variável aleatória e, *e* é a constante natural de valor 2.71828....

Segundo Ribeiro (2008) a função da distribuição de Poisson possui a propriedade de recorrência, pode-se encontrar seus valores através do uso de seus antecessores.

A fórmula de recorrência pode ser vista em (2).

$$P(X) = P(X-1) * \frac{\lambda}{X} \quad (2)$$

De acordo com Wilks (2011) a esperança matemática (valor esperado da função) é encontrada através de (3).

$$E(X) = \lambda \quad (3)$$

Enquanto a variância de uma distribuição de Poisson é encontrada através de (4).

$$\sigma_x^2 = \lambda \quad (4)$$

2.2.1.2. Distribuição de Variáveis Aleatórias Contínuas

As variáveis aleatórias contínuas são utilizadas em medições, suas probabilidades são calculadas baseadas em intervalos de valores, diferente das variáveis aleatórias discretas que se baseiam em valores pontuais.

Segundo Ribeiro (2008) a função densidade probabilidade mostra que a probabilidade de certo evento acontecer é a área definida pela função entre os dois valores do intervalo. Tal definição pode ser vista abaixo em (5).

$$P(a <= x <= b) = \int_{a}^{b} f(x)dx \qquad (5)$$

Na figura 9 pode-se visualizar um exemplo de um gráfico de uma função com o intervalo definido entre *a* e *b*. A parte destacada é a área definida pela função densidade probabilidade.

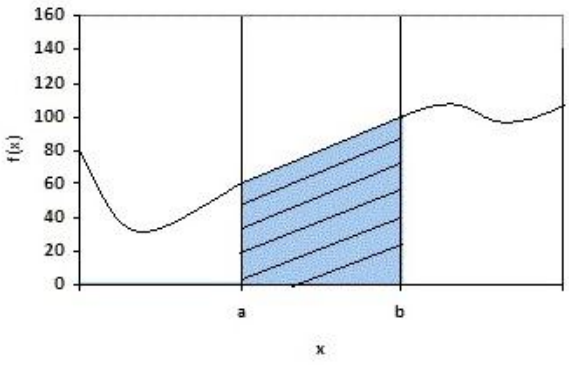


Figura 9 – Intervalo definido entre a e b

2.2.1.2.1. Distribuição Uniforme

A distribuição uniforme é contínua e possui valor idêntico para todos os valores analisados. Ela depende somente dos valores contidos na extremidade do intervalo.

Sua função de densidade probabilidade pode ser vista em (6), para valores de x maiores que *a* (limite inferior do intervalo) e menores que *b* (limite superior do intervalo).

$$f(x) = \frac{1}{b-a} \tag{6}$$

Agora para valores de x fora do intervalo, a função é nula e pode ser vista em (7).

$$f(x) = 0 \tag{7}$$

A figura 10 mostra um exemplo de uma função com distribuição uniforme dos dados.

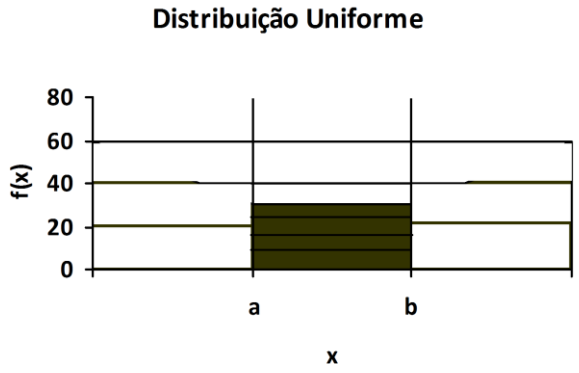


Figura 10 – Distribuição Uniforme

Os valores da média e da variância para distribuições uniformes podem ser observados em (8) e (9) respectivamente.

$$\mu = \frac{a+b}{2} \tag{8}$$

$$\sigma^2 = \frac{(b-a)^2}{12} \tag{9}$$

2.2.1.2.2. Distribuição Normal

A distribuição normal contém um conjunto de valores, que através de dois parâmetros, a média e o desvio padrão se encontra a probabilidade associada a eles.

A curva normal do gráfico da distribuição na figura 11 tem como domínio todos os números reais, sendo que sua imagem não atinge valores menores ou iguais a zero. A área que nela compreende, possui valor igual a um, ou seja, cem por cento da probabilidade estudada.

O gráfico da função probabilidade da distribuição normal possui como ponto máximo o valor de x correspondente a média da distribuição.

A curva normal tem o formato de um sino, com concavidade para baixo. Para descobrir a probabilidade de um valor x, deve-se analisar a área da curva a partir do - ∞ até o x escolhido, através da função mostrada em (10).

$$f(x) = \frac{1 * e^{\frac{-(x-\mu)^2}{2*\sigma^2}}}{\sigma * \sqrt{2 * \pi}} \qquad (10)$$

A figura 11 mostra um exemplo de uma função com distribuição normal dos dados, englobando média e desvio padrão.

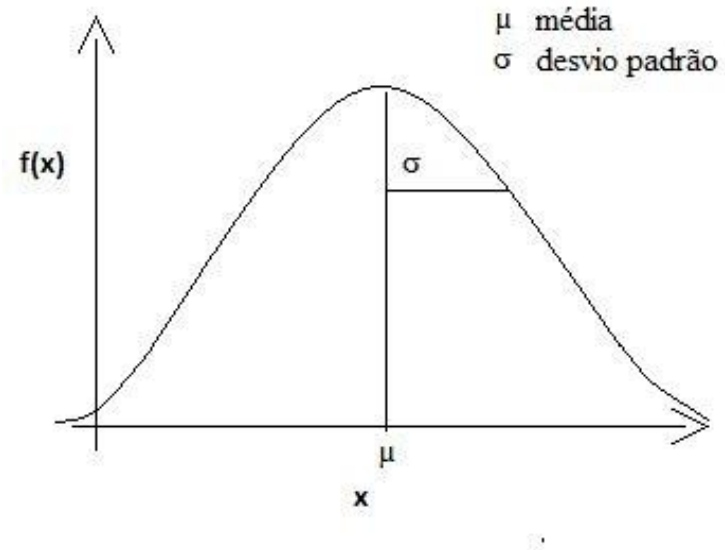


Figura 11 – Curva Normal

Para calcular a probabilidade de um determinado evento ocorrer, deve-se analisar a área da função representada por ele, limitado inferior e superiormente por valores, *a* e *b* respectivamente. Logo a probabilidade é vista em (11).

$$P(a <= x <= b) = \int_{a}^{b} f(x)dx \qquad (11)$$

Lembrando que a média e a variância são encontradas respectivamente através da μ e σ².

2.3. Funções utilizadas no MATLAB

A linguagem de desenvolvimento escolhida para simular o posicionamento das redes de sensores e a troca de informações entre seus nós foi o MATLAB.

O MATLAB foi desenvolvido pela Mathworks, Inc. e sua sigla significa MATrix LABoratory, ressaltando assim, a finalidade de resolver problemas matemáticos relacionados a matrizes e números complexos. Além de ajudar na resolução de problemas com controle de fluxo, medidas estatísticas, gráficos, soluções de sistemas de equações lineares, interpolação e ajuste de curvas, análise polinomial,

integração numérica e diferenciação, equações diferenciais ordinárias, processamento de sinais, entre outros.

O MATLAB é uma linguagem de alto desempenho e interpretada que integra computação, visualização e programação, através de um modo fácil de utilização relacionada com o uso do ambiente de desenvolvimento.

> "O programa interpretador age como uma simulação de software de uma máquina cujo ciclo buscar-executar lida com instruções de programa em linguagem de alto nível em vez de instruções de máquina. Essa simulação de software, evidentemente, fornece uma máquina virtual para a linguagem." (Sebesta, 2002)

Nas seções a seguir serão explicadas algumas funções associadas ao projeto proposto. A explicação está baseada de acordo com os arquivos de ajuda do programa MATLAB, tais arquivos foram desenvolvidos pela Mathworks.

2.3.1. Funções utilizadas para a geração dos pacotes de dados

Abaixo serão apresentadas duas funções relacionadas a distribuição de Poisson. Foram feitos alguns estudos e testes para um melhor entendimento da distribuição de Poisson no MATLAB, já que tal distribuição será utilizada na geração de pacotes.

A primeira função é poisspdf (Função Densidade Probabilidade Poisson). A poisspdf pode ser entendida como uma sigla para função densidade probabilidade de Poisson, logo abaixo pode-se visualizar algumas das características dos parâmetros de saída e de entrada.

- poisspdf(Conjunto de Valores, Lambda)
  - Gera números aleatórios da distribuição de Poisson com parâmetro lambda;
  - Lambda pode ser vetor, matriz ou arranjo multidimensional;

- O tamanho do conjunto de valores e o tamanho de lambda devem ser iguais;
- Os parâmetros em lambda devem ser todos positivos;
- Para cada elemento do conjunto de valores, e lambda correspondente, são substituídos na fórmula apresentada em (12). A função retorna a função densidade probabilidade de Poisson para cada valor analisado.
- O conjunto de valores pode ser qualquer inteiro não negativo;

Um exemplo de sua utilização pode ser visto no quadro 1, para *X*, um vetor de 10 posições e lambda igual 1.

```
>> X = [1 2 3 4 5 6 7 8 9 10];
>> lambda = 1;
>> Y = poisspdf(X, lambda);
>> Y
Y =
    0.3679 0.1839 0.0613 0.0153 0.0031 0.0005 0.0001 0.0000 0.0000 0.0000
```

Quadro 1 – Para um vetor X e um escalar lambda.

$$P(1) = \frac{1^{1} * e^{-1}}{1!} = \frac{1}{e} = 0{,}3679 \qquad (12)$$

Em (12) é mostrada a fórmula da probabilidade para distribuição de Poisson para o valor da primeira posição do vetor X e lambda igual a 1.

No quadro 2 pode-se visualizar um segundo exemplo para os mesmos valores de X mostrados anteriormente, mas para valores diferentes de lambda. No caso, lambda é um vetor de dez posições, os quais para cada posição de lambda é usado a posição correspondente de X.

```
>> X = [1 2 3 4 5 6 7 8 9 10];
>> lambda = [1 2 3 4 5 6 7 8 9 10];
>> Y = poisspdf(X, lambda);
>> Y
Y =
    0.3679 0.2707 0.2240 0.1954 0.1755 0.1606 0.1490 0.1396 0.1318 0.1251
```

Quadro 2 – X e lambda são vetores

Obs: Quando *lambda* e *X* não são escalares, eles devem possuir a mesmo tamanho.

Em (13) e (14) são mostradas a fórmula da probabilidade para distribuição de Poisson para os valores da primeira e segunda posição do vetor X e para a primeira e segunda posição do vetor lambda respectivamente.

$$P(1) = \frac{1^1 * e^{-1}}{1!} = \frac{1}{e} = 0{,}3679 \qquad (13)$$

$$P(2) = \frac{2^2 * e^{-2}}{2!} = \frac{1}{e} = 0{,}2707 \qquad (14)$$

A segunda função é a *poissrnd* e pode ser entendida como uma sigla para valores randômicos de Poisson, logo abaixo pode-se visualizar algumas das características dos parâmetros de saída e de entrada.

- poissrnd(lambda)
  - Gera números aleatórios da distribuição de Poisson com parâmetro lambda.
  - Lambda pode ser vetor, matriz ou arranjo multidimensional.
  - O tamanho de conjunto de valores retornado e o tamanho de lambda são idênticos.
  - Os parâmetros em lambda devem ser todos positivos.

- ❖ poissrnd(lambda, vetor)
  - O *vetor* corresponde às dimensões do conjunto de valores de resposta.
  - Os elementos do *vetor* devem ser não negativos e inteiros.
  - O *vetor* deve ser linha.
- ❖ poissrnd(lambda, linhas, colunas)
  - Onde linhas e colunas correspondem às linhas e às colunas do conjunto resposta.

A probabilidade de ocorrência dos números de zero a dez, com lambda igual a um, utilizando a função de distribuição de Poisson, pode ser vista abaixo na Tabela 1.

Número	**Probabilidade**
0	0.3679
1	0.3679
2	0.1839
3	0.0613
4	0.0153
5	0.0031
6	0.0005
7	0.0001
8	0.0000
9	0.0000
10	0.0000

Tabela 1 – Distribuição de Poisson

Obs.: Foram utilizados valores com apenas 4 casas decimais.

De acordo com a Tabela 1, o número retornado é um valor inteiro não negativo, tendo grande probabilidade de ser zero e um. Uma probabilidade grande, porém menor do número gerado ser dois, e assim por diante.

O que a função poissrnd faz é retornar um número, ou conjunto de números, positivo e inteiro de acordo com a probabilidade encontrada, através da relação existente entre o número, lambda e a constante natural *e*, na fórmula de Poisson.

Um exemplo de sua utilização pode ser visto no quadro 3, para lambda igual a 1.

```
>> lambda = 1;
>> Y = poissrnd(lambda);
>> Y
Y =
     1
```

Quadro 3 – Lambda com valor 1

$$P(1) = \frac{1^{1} * e^{-1}}{1!} = \frac{1}{e} = 0{,}3679 \qquad (15)$$

A probabilidade de o número um aparecer quando lambda for 1 é de pouco mais de 36 % e pode ser vista em (15).

Na figura 12 são mostrados os valores de f(x), com x variando de 0 a 10, para diferentes valores de lambda, variando de 1 a 10. Para encontrar tais valores foi utilizada a função *poissrnd*.

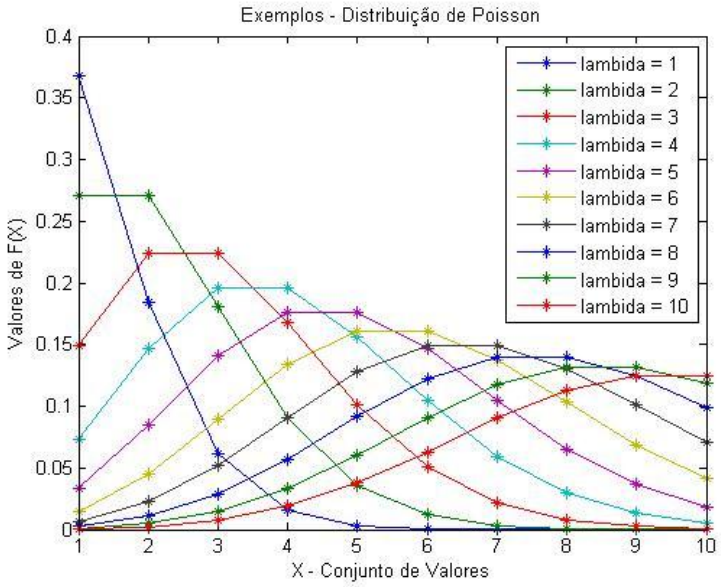


Figura 12 – Exemplos de utilização da função *poisspdf* no MATLAB.

2.3.2. Funções utilizadas para a geração do terreno

No quadro 4 é definido os valores de *x*, *y* e *z* do terreno:

```
>> [x,y] = meshgrid(-1:0.1:1,-1:0.1:1);
>> z = exp(-(x.^2+y.^2));
```

Quadro 4 – Exemplo de uso do meshgrid

A função *meshgrid* transforma o domínio especificado pelos vetores de valores em conjuntos de valores resposta. As linhas e colunas dos conjuntos de saída são cópias dos vetores de entrada.

No MATLAB, quando é definido algo do tipo *par1*:*par2*:*par3* significa que um conjunto de valores será criado a partir de par1 até par3, com intervalo de par2 entre cada valor.

Por exemplo, no quadro 5 é mostrado um exemplo de definição de um intervalo de valores.

```
>> -1:1:1;
```

Quadro 5 – Intervalo de valores

O conjunto definido será [-1 0 1]. Inicialmente *par1*, definido como -1 é adicionado ao conjunto. Em seguida a todos os valores adicionados ao conjunto é incrementado *par2*, definido como 1 até atingir um valor maior ou igual a *par3*, definido como 1. Ocasionando assim um conjunto com três elementos.

Logo no *meshgrid* do bloco de comando acima, haverá um conjunto no primeiro e no segundo parâmetro com vinte um elementos ([-1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]);

O conjunto de elementos resposta é definido na quantidade de elementos dos parâmetros, no caso acima ambos terão linhas e colunas iguais e equivalentes a vinte um.

O primeiro parâmetro tem como linha todos os valores do conjunto do primeiro parâmetro, no caso ele terá vinte uma colunas, e possuirá vinte uma linhas (correspondente a quantidade de elementos do segundo parâmetro). O mesmo acontece para o segundo parâmetro do vetor resposta.

Dado um plano com x e y, sabe-se que a malha formada entre eles é associada a figura gerada por z. No exemplo dado o plano possui cento e quarenta e um pontos (matriz 21x21), e cada ponto tem coordenadas (x(i,j),y(i,j)), onde *i* representa a linha e *j* a coluna das matrizes *x* e *y*.

Para gerar uma superfície 3D pode-se utilizar também a função *peaks*. Função que retorna diversos picos, ou pontos, para z, traçando planos entre eles, tais pontos são obtidos transladando e através de diferentes escalas de distribuições de Gauss.

No quadro 6, a função *peaks* associa pontos a *z* em função de *x* e *y* no MATLAB.

```
>> z = peaks(x,y);
```

Quadro 6 – Função peaks

Com base nas linhas de comando do quadro 4 são mostradas algumas funções em MATLAB abaixo que representam características ou figuras geométricas no

plano tridimensional. Elas estão divididas em duas categorias, curvas de nível e exibição de superfícies tridimensionais.

Nas curvas de nível, existem três funções a serem estudadas, foram analisadas cada uma delas a seguir:

- **Contour** – Curvas de Nível;
- **Contourf** – Curvas de Nível com Preenchimento;
- **Contour3** – Curvas de Nível em Superfícies 3D;

A figura 13 mostra as curvas de nível baseadas em x, y e z:

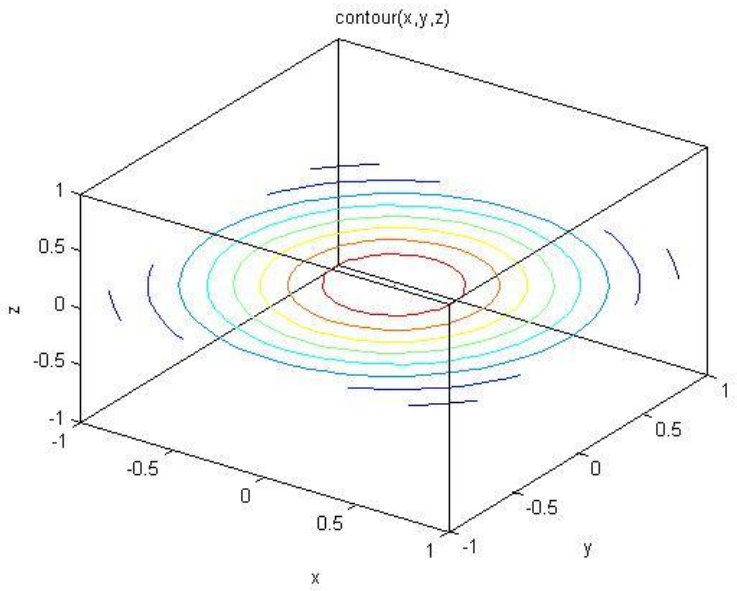


Figura 13 – contour(x,y,z)

A figura 14 mostra as curvas de nível com diferentes cores entre as regiões delimitadas pelas linhas baseadas em x, y e z:

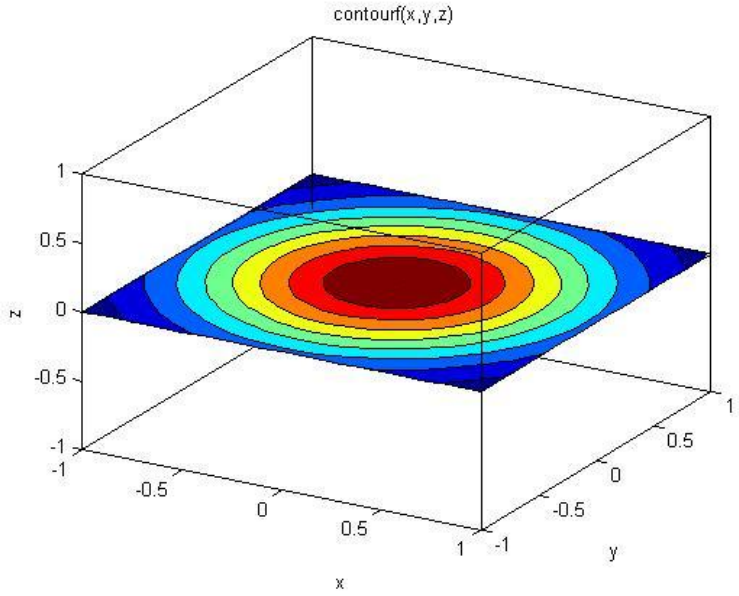


Figura 14 – contourf(x,y,z)

A figura 15 mostra as curvas de nível da superfície, em um plano tridimensional, baseadas em *x*, *y* e *z*.

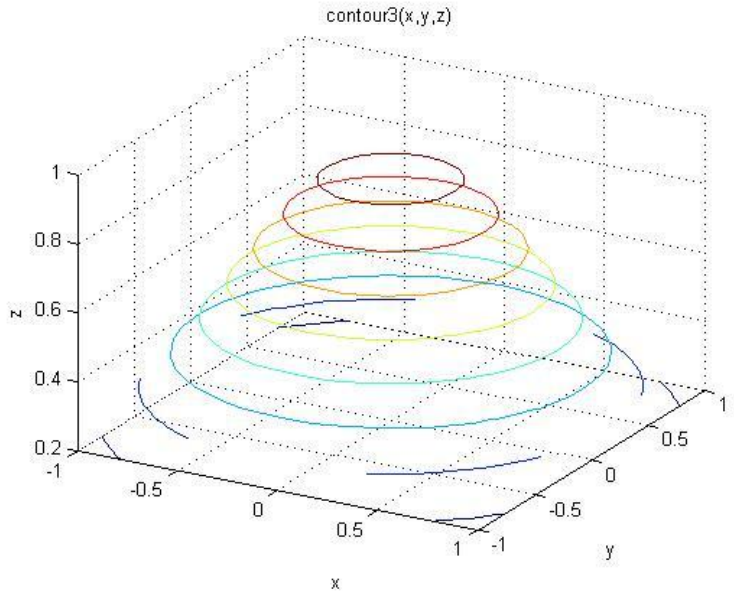


Figura 15 – contour3(x,y,z)

A outra categoria das funções mostra a superfície tridimensional formada por planos traçados entre os pontos determinados. Abaixo são mostradas as funções estudadas para o desenvolvimento do plano tridimensional:

- **3DSurf (Surface)** – Exibe Superfícies;
- **Surfc (Surface with Contour)**– Exibe Superfícies com Curvas de Nível;
- **Surfl (Lit Surface)** – Exibe Superfícies de modo Iluminado;
- **Mesh** – Exibe uma superfície com estrutura de arame;
- **Meshc** - Exibe uma superfície com estrutura de arame;
- **Meshz** – Exibe uma superfície com estrutura de arame com um plano de referência em volta da malha;
- **Waterfall** – Exibe uma superfície com estrutura de arame com um plano de referência com um plano de referência desenhado a partir das linhas da matriz;
- **Ribbon** – Exibe uma superfície desenhando as colunas da matriz como tiras segmentadas;

A figura 16 mostra os pontos como uma superfície, baseados em x, *y* e *z*.

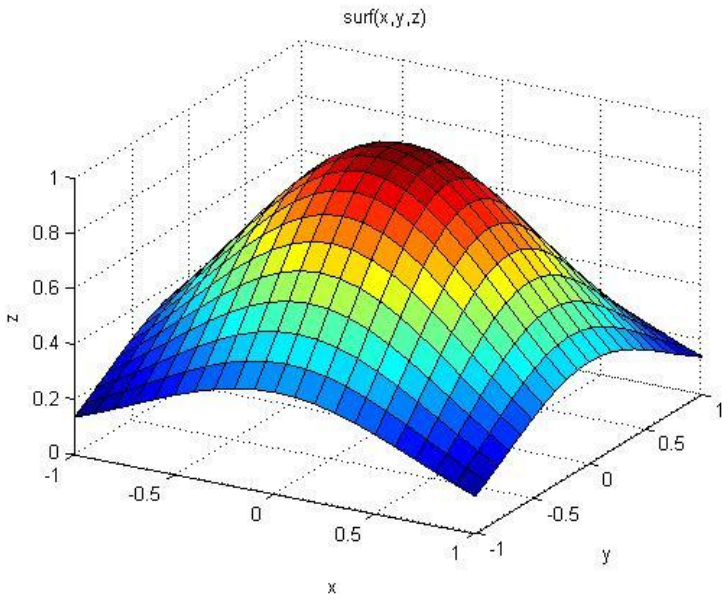


Figura 16 – surf(x,y,z)

A Figura 17 mostra os pontos como uma superfície com curvas de nível baseados em *x*, *y* e *z*.

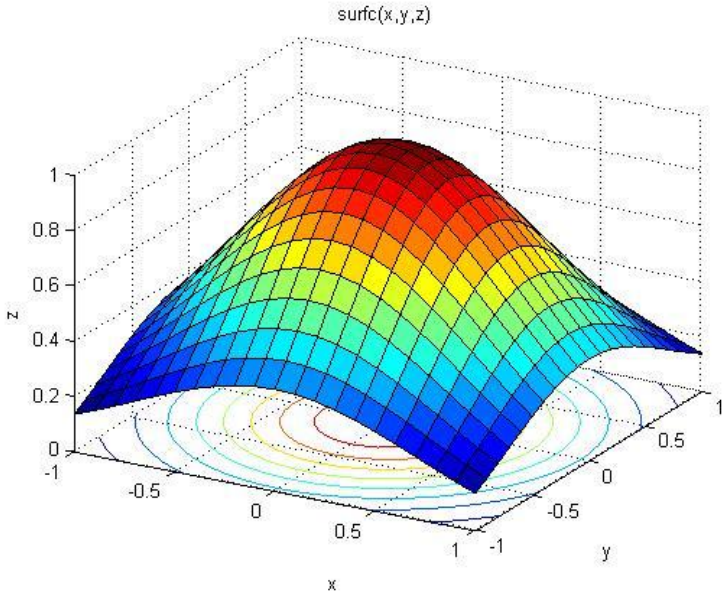


Figura 17 – surfc(x,y,z)

A Figura 18 traça os pontos com luzes baseado em uma mapa de cor, baseados em *x*, *y* e *z*.

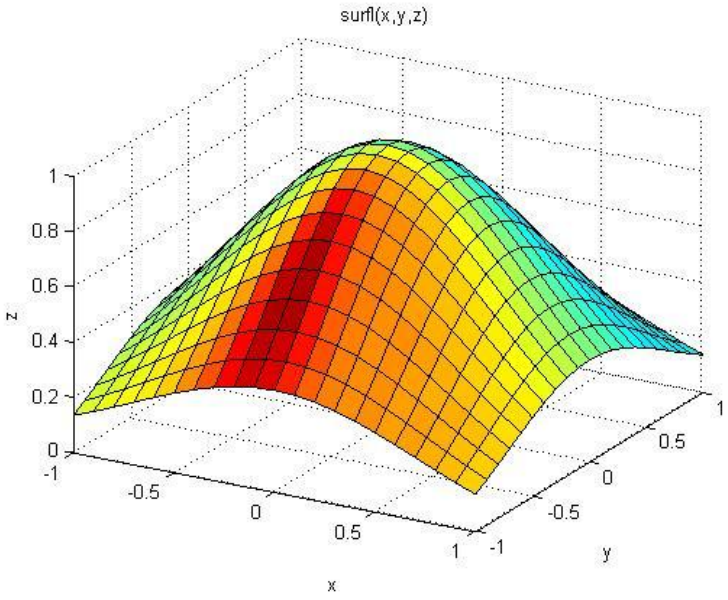


Figura 18 – surfl(x,y,z)

A Figura 19 mostra os pontos como uma superfície com painel de arame, baseados em *x*, *y* e *z*.

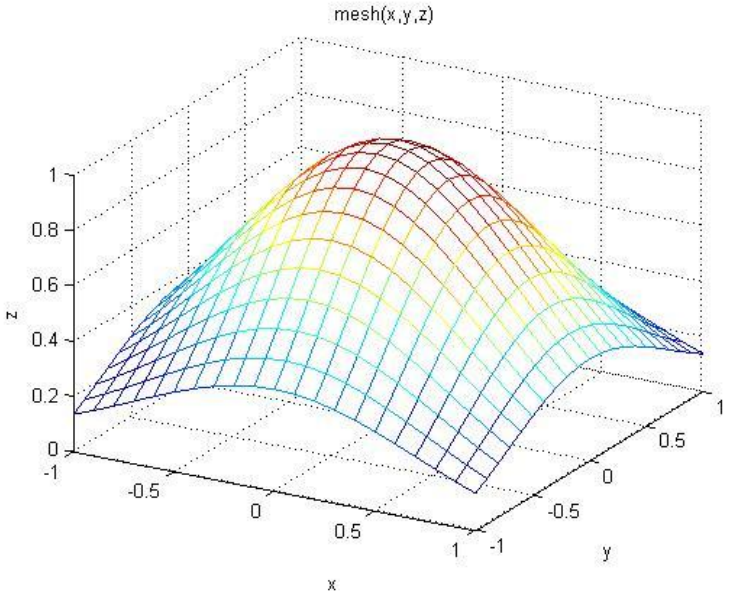


Figura 19 – mesh(x,y,z)

A Figura 20 mostra os pontos como uma superfície com painel de arame, adicionado de informações da curva de nível, baseados em *x*, *y* e *z*.

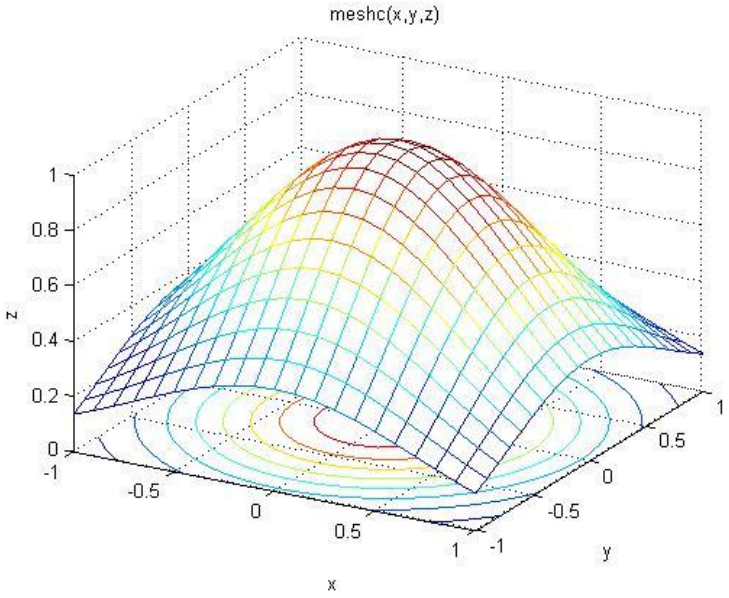


Figura 20 – meshc(x,y,z)

A Figura 21 mostra os pontos como uma superfície com painel de arame, adicionado de um plano em volta da malha, baseados em *x*, *y* e *z*.

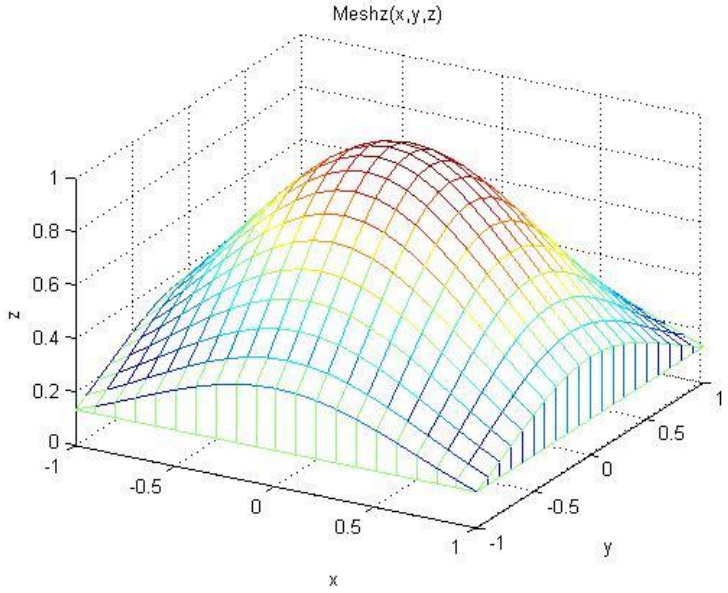


Figura 21 – meshz(x,y,z)

A Figura 22 mostra os pontos como uma superfície com painel de arame, adicionado de um plano desenhado a partir das linhas formadas pelos pontos, baseados em *x*, *y* e *z*.

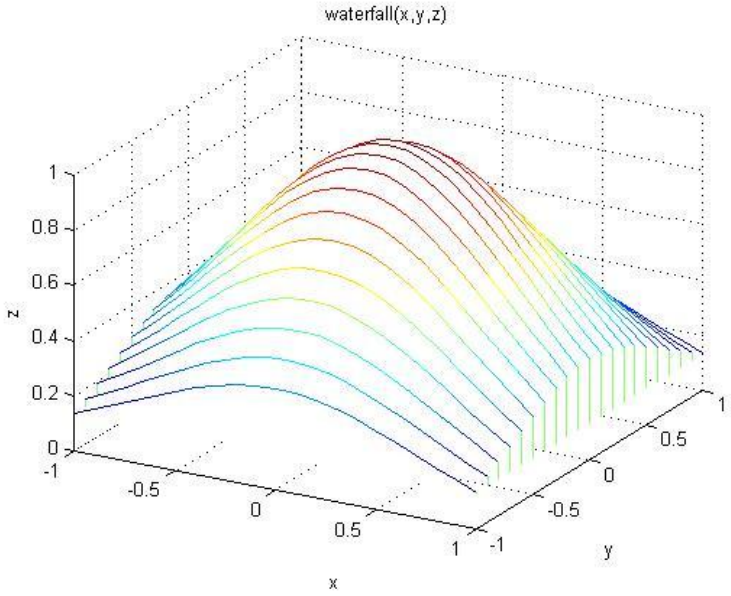


Figura 22 – waterfall(x,y,z)

A Figura 23 mostra os pontos da superfície desenhando as colunas como tiras segmentadas, baseados em *z*.

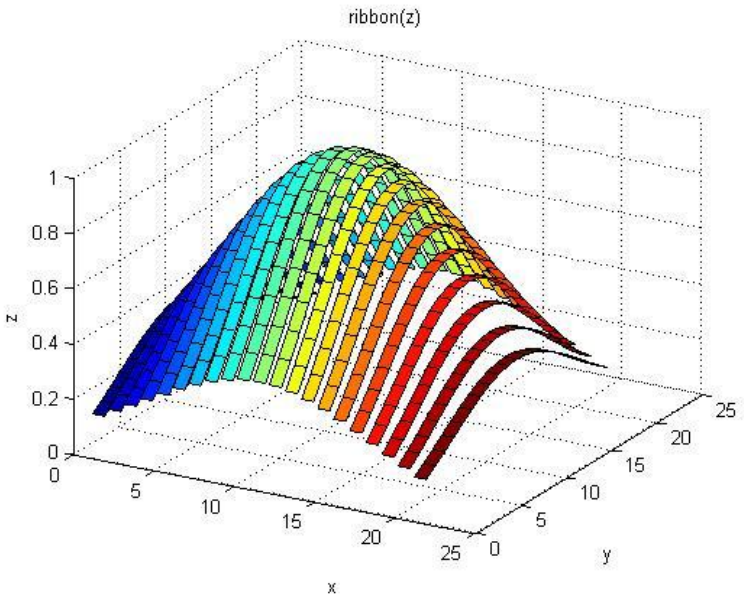


Figura 23 – ribbon(z)

Depois dos blocos de comando do quadro 4 serem interpretados, percebe-se que através de algumas variações da fórmula apresentada na atribuição aos valores

de z, consegue-se manipular a curva e desenhá-la de acordo com a necessidade do problema proposto.

$$z = valor\_transladado\_z + altura\_pico\_elevação * e^{-\left(\frac{(x - valor\_transladado\_x)^2}{inclinação\_x} + \frac{(y - valor\_transladado\_y)^2}{inclinação\_y}\right)} \quad (16)$$

Como observado em (16), pode-se definir as variáveis como:

- **valor_transladado_z:** altura incrementada da curva, onde ela iniciará no eixo z.
- **altura_do_pico_da_elevação:** valor correspondente a elevação encontrada entre a base da curva e seu pico.
- **valor_transladado_x:** deslocamento da curva em relação ao eixo x.
- **valor_transladado_y:** deslocamento da curva em relação ao eixo y.
- **inclinacao_x:** valor responsável pela inclinação da curva, quanto maior mais inclinado é a curva em relação ao eixo x.
- **inclinacao_y:** valor responsável pela inclinação da curva, quanto maior mais inclinado é a curva em relação ao eixo y.

Na figura 24 podem ser visualizados alguns exemplos de diferentes valores para a fórmula descrita anteriormente.

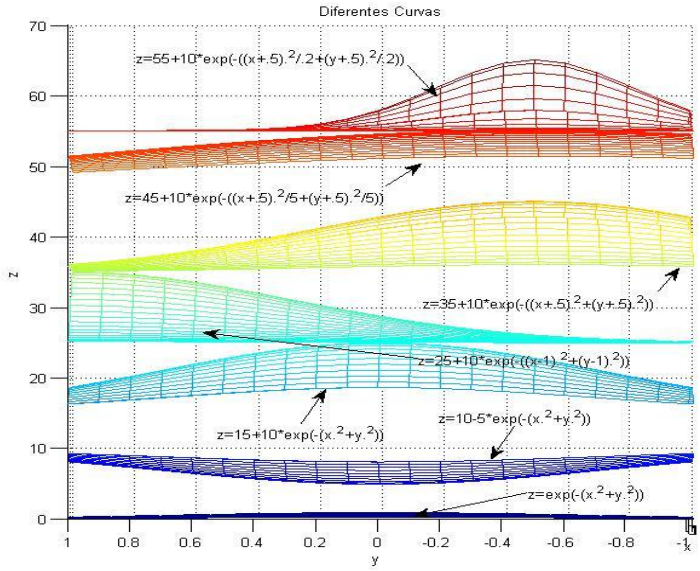


Figura 24 – Diferentes curvas

2.3.3. Funções utilizadas para o posicionamento dos nós

Para o posicionamento dos nós no terreno foi utilizada uma função que gera números aleatórios com base na distribuição uniforme, mostrada na Seção 2.2.1.2.1.

Abaixo será apresentada uma função do MATLAB relacionada a geração de números aleatórios com distribuição uniforme. A função é *rand* (Número Randômicos) e logo abaixo pode-se visualizar algumas das características dos parâmetros de saída e de entrada.

- rand(Conjunto de Valores)
  - Gera números pseudoaleatórios da distribuição Uniforme;
  - O tamanho do conjunto de valores é designado por números formando matrizes multidimensionais;
  - O conjunto de valores pode ser omitido, quando omitido a saída é formada por apenas um número;
  - O conjunto de valores pode ser qualquer inteiro não negativo;
  - Todo número pseudoaleatório é gerado através de um método definido pelo MATLAB e um valor inicial, denominado semente. Tal método pode ser um parâmetro de entrada.

Fazendo com que as coordenadas *x* e *y* dos pontos recebam valores pseudoaleatórios, tem-se como posicioná-los no terreno, aplicando tais coordenadas para encontrar o *z* equivalente.

Para visualizar os nós no gráfico é utilizada a função *scatter3.* Função que mostra em um plano tridimensional, marcadores com coordenadas x, y e z como parâmetros de entrada. Abaixo o comando designa os passos para a criação do gráfico com três variáveis.

No quadro 7 pode-se visualizar a inicialização das variáveis para utilização na funções do MATLAB. Mostra-se também a utilização da função *scatter3*.

```
>> x = 1:1:10;
>> y = 10:-1:1;
>> z = [1 2 3 4 5 5 4 3 2 1];
>> scatter3(x,y,z);
```

Quadro 7 – Função scatter

Na Figura 25 pode-se visualizar um exemplo de uso dessa função.

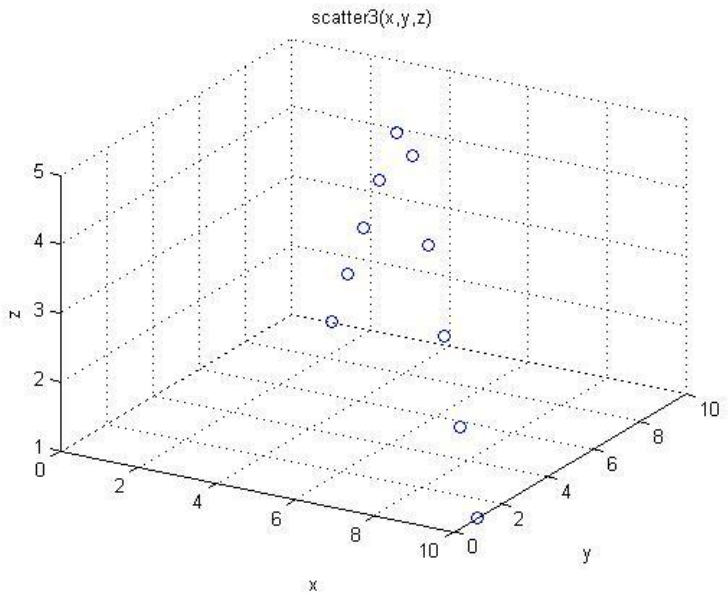


Figura 25 – Exemplo de utilização da função scatter3

2.4. Algoritmo de roteamento da rede de sensores

O algoritmo de roteamento, que determina o caminho da troca de informações entre os nós, das redes de sensores é o de *Busca em Largura*. O algoritmo implementado mostrará ao usuário o vetor de roteamento de uma rede qualquer, vetor no qual mostra os antecessores de cada nó.

> "O Algoritmo de Busca em Largura expande a fronteira entre vértices descobertos e não descobertos uniformemente através da largura da fronteira. O algoritmo descobre todos os vértices a uma distância k do vértice origem antes de descobrir qualquer vértice a uma distância k + 1." (ZIVIANI, 2010)

Dado qualquer nó da rede, será percorrido o menor caminho possível entre ele e o nó raiz (nó responsável pelo recebimento das informações da rede).

Ao final do algoritmo o vetor de roteamento mostrará quais nós possuem ligação direta ou indireta com o nó raiz. A ligação direta do nó *i* com o *j* é representada quando o nó *i* possui ligação com o nó *j*. A ligação indireta do nó *i* com o nó *j* ocorre através de um ou mais nós intermediários.

Todo nó perdido na rede, ou seja, nós que não possuem ligação direta ou indireta com o nó raiz, o algoritmo armazena que seu antecessor será zero. Lembrando que o nó raiz não possui antecessor, logo o seu valor armazenado no vetor de roteamento também será zero.

Como toda rede pode ser vista analogamente como um grafo, a estrutura das redes pode ser vista como caminhos com ciclos, caminho sem ciclos (árvore) ou florestas. A seguir pode-se visualizar a definição dada aos três tipos de redes:

- Caminho com ciclos: um caminho que quando percorrido passa por um nó duas vezes, ou seja, possui um ciclo.
- Caminho sem ciclos: um caminho que quando percorrido não possui nós repetidos, ou seja, dado um caminho qualquer da rede um nó não aparece mais de uma vez.
- Florestas: como há o risco de acontecer de um ou mais nós ficarem

perdidos, ou seja, não possuírem ligação direta ou indireta com o nó raiz, será apresentado diversos conjuntos de nós interligados.

O algoritmo utilizado pode ser visto em metalinguagem logo abaixo através da descrição de seis passos.

1º Passo: Declaração de todas as variáveis que serão utilizadas durante o algoritmo. Dois vetores e duas variáveis, denominados *VS*, *VR*, *proximo* e *ler*. O *VS* conterá os elementos que possuem ligação direta ou indireta com o primeiro nó, ou seja, há um caminho que pode ser percorrido entre o primeiro nó e o nó especificado. O *VR* será responsável pelos armazenamentos dos antecessores de cada nó, ou seja, quando percorrido o menor caminho do primeiro nó ao nó especificado, será armazenado na posição correspondente ao nó o número de seu antecessor. A variável *proximo* será responsável pela especificação da posição de inserção dos elementos no *VS* e a variável *ler* será responsável pela leitura do nó na posição especificada por ela no *VS*.

2º Passo: Inicialmente as duas variáveis e os dois vetores serão inicializados com zero, ou seja, os vetores não possuirão nenhum elemento inserido. Em algumas linguagens, quando declarados, os vetores são inicializados com todas as posições iguais a zero.

3º Passo: Inicia a busca a partir do primeiro nó, ou seja, a partir do nó associado com o menor índice.

4º Passo: Serão inseridos no vetor os elementos que possuírem ligação com o nó, em cada inserção a variável *proximo* será incrementada e a posição do elemento inserido no *VS*, no *VR* receberá o valor de onde sai essa ligação. Quando as inserções terminarem, ou seja, todas as ligações que o nó possui forem inseridas no *VS*, a variável *ler* é incrementada.

Lembrando que é armazenado no *VS* o número do nó correspondente se ele não se encontrar no *VS* ainda.

5º Passo: A variável *ler* é incrementada, e a posição correspondente a variável *ler* no *VS* passa pelo 4º Passo. Esse procedimento é feito até a *ler* assumir valor maior do que a quantidade de nós.

6º Passo: No final o *VR* conterá os antecessores de cada nó.

Nas seguintes subseções são apresentados três exemplos, o primeiro exemplo possui seis nós, o segundo nove nós e o terceiro oito nós.

Cada exemplo possui uma topologia diferente, sendo comum entre eles que todos os nós são numerados, e as informações são geradas a partir do primeiro nó.

O algoritmo inicia-se a partir do primeiro nó, sendo o *VS* iniciado com a primeira posição igual a um, a variável *ler* igual a um, sendo o *VR* e a variável *proximo* ainda não iniciados. A inicialização das variáveis pode ser vista no quadro 8.

```
VS = [1]
VR = []
ler = 1
proximo  = 0
```

Quadro 8 – Inicialização das variáveis

2.4.1. Topologia 1

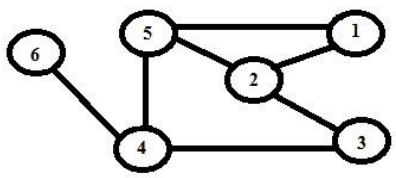


Figura 26 - Topologia 1

A figura 26 define a topologia definida para o primeiro exemplo de utilização do algoritmo.

O quadro 9 mostra a primeira iteração.

1ª Iteração

nó observado: 1

VS = [1 2 5]
VR = [0 1 0 0 1]
ler = 2
proximo = 4

Quadro 9 – Primeira Iteração(Topologia 1)

O quadro 10 mostra a segunda iteração.

2ª Iteração

nó observado: 2

VS = [1 2 5 3] – apenas é adicionado o nó 3 pois o nó 5 já está no VS
VR = [0 1 2 0 1]
ler = 3
proximo = 5

Quadro 10 – Segunda Iteração(Topologia 1)

O quadro 11 mostra a terceira iteração.

3ª Iteração

nó observado: 5

VS = [1 2 5 3 4]
VR = [0 1 2 5 1]
ler = 4
proximo = 6

Quadro 11 – Terceira Iteração(Topologia 1)

O quadro 12 mostra a quarta iteração.

4ª Iteração

nó observado: 3

VS = [1 2 5 3 4] – não se altera pois 4 já está no VS

VR = [0 1 2 5 1] – não se altera pois o antecessor do 4 continua sendo o 5, e não o 3

ler = 5

proximo = 6 – a variável proximo não se altera, pois não foi adicionado nenhum elemento no VS

Quadro 12 – Quarta Iteração (Topologia 1)

Lembrando que o antecessor de um nó é o nó anterior a ele percorrendo o menor caminho entre o nó especificado e o primeiro nó.

O quadro 13 mostra a quinta iteração.

5ª Iteração

nó observado: 4

VS = [1 2 5 3 4 6]

VR = [0 1 2 5 1 4]

ler = 6

proximo = 6

Quadro 13 – Quinta Iteração (Topologia 1)

O quadro 14 mostra a sexta iteração.

6ª Iteração

nó observado: 6

VS = [1 2 5 3 4 6]
VR = [0 1 2 5 1 4]
ler = 7
proximo = 6

Quadro 14 – Sexta Iteração (Topologia 1)

A variável *ler* é maior do que a quantidade de nós, logo o algoritmo se encerra.

2.4.2. Topologia 2

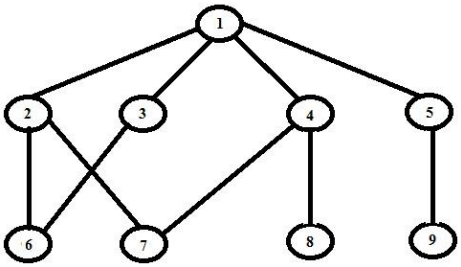


Figura 27 – Topologia 2

A figura 27 define a topologia definida para o segundo exemplo de utilização do algoritmo.

O quadro 15 mostra a primeira iteração.

1ª Iteração

nó observado: 1

VS = [1 2 3 4 5]
VR = [0 1 1 1 1]
ler = 2
proximo = 6

Quadro 15 – Primeira Iteração (Topologia 2)

O quadro 16 mostra a segunda iteração.

2ª Iteração

nó observado: 2

VS = [1 2 3 4 5 6 7]
VR = [0 1 1 1 1 2 2]
ler = 3
proximo = 8

Quadro 16 – Segunda Iteração (Topologia 2)

O quadro 17 mostra a terceira iteração.

3ª Iteração

nó observado: 3

VS = [1 2 3 4 5 6 7]
VR = [0 1 1 1 1 2 2]
ler = 4
proximo = 8

Quadro 17 – Terceira Iteração (Topologia 2)

O quadro 18 mostra a quarta iteração.

4ª Iteração

nó observado: 4

VS = [1 2 3 4 5 6 7 8]
VR = [0 1 1 1 1 2 2 4]
ler = 5
proximo = 8

Quadro 18 – Quarta Iteração (Topologia 2)

O quadro 19 mostra a quinta iteração.

5ª Iteração

nó observado: 5

VS = [1 2 3 4 5 6 7 8 9]
VR = [0 1 1 1 1 2 2 4 5]
ler = 6
proximo = 8

Quadro 19 – Quinta Iteração (Topologia 2)

O quadro 20 mostra a sexta iteração.

6ª Iteração

nó observado: 6

VS = [1 2 3 4 5 6 7 8 9]
VR = [0 1 1 1 1 2 2 4 5]
ler = 7
proximo = 8

Quadro 20 – Sexta Iteração (Topologia 2)

O quadro 21 mostra a sétima iteração.

7ª Iteração

nó observado: 7

VS = [1 2 3 4 5 6 7 8 9]
VR = [0 1 1 1 1 2 2 4 5]
ler = 8
proximo = 8

Quadro 21 – Sétima Iteração (Topologia 2)

O quadro 22 mostra a oitava iteração.

8ª Iteração

nó observado: 8

VS = [1 2 3 4 5 6 7 8 9]
VR = [0 1 1 1 1 2 2 4 5]
ler = 9
proximo = 8

Quadro 22 – Oitava Iteração (Topologia 2)

O quadro 23 mostra a nona iteração.

9ª Iteração

nó observado: 9

VS = [1 2 3 4 5 6 7 8 9]
VR = [0 1 1 1 1 2 2 4 5]
ler = 10
proximo = 8

Quadro 23 – Nona Iteração (Topologia 2)

A variável *ler* é maior do que a quantidade de nós.

2.4.3. Topologia 3

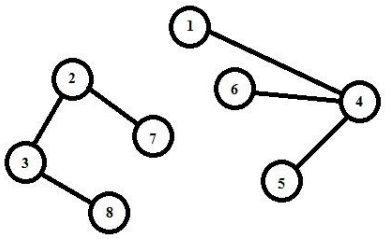


Figura 28 – Topologia 3

A figura 28 define a topologia definida para o terceiro exemplo de utilização do algoritmo.

O quadro 24 mostra a primeira iteração.

1ª Iteração
nó observado: 1
VS = [1 4]
VR = [0 0 0 1]
ler = 1
proximo = 2

Quadro 24 – Primeira Iteração (Topologia 3)

O quadro 25 mostra a segunda iteração.

2ª Iteração
nó observado: 4
VS = [1 4 5 6]
VR = [0 0 0 1 4 4]
ler = 2
proximo = 4

Quadro 25 – Segunda Iteração (Topologia 3)

O quadro 26 mostra a terceira iteração.

3ª Iteração

nó observado: 5

VS = [1 4 5 6]
VR = [0 0 0 1 4 4]
ler = 3
proximo = 4

Quadro 26 – Terceira Iteração (Topologia 3)

O quadro 27 mostra a quarta iteração.

4ª Iteração

nó observado: 6

VS = [1 4 5 6]
VR = [0 0 0 1 4 4]
ler = 4
proximo = 4

Quadro 27 – Quarta Iteração (Topologia 3)

No começo da 5ª iteração não existe mais nós interligados ao nó raiz, tanto direta quanto indiretamente.

2.5. Simulação de geração de pacotes

A rede de sensores é constituída de elementos denominados nós. Cada nó possui um ciclo relacionado à informação recebida. O nó gera, armazena, processa e transmite uma informação através de seus componentes.

Para a simulação da geração dos pacotes devem ser analisadas especificamente informações sobre o sensor e o transmissor, componentes do nó

responsáveis respectivamente pelo recebimento e envio dos pacotes de dados. Tais informações estão relacionadas com as taxas de geração e transmissão de dados.

Atualmente pode-se pressupor que a taxa de transmissão aceitável de um transmissor contido no nó pode ser definida com um valor de 9.600 bits por segundo (bps) e que paralelamente o tamanho do pacote de dados transmitido pode ser de 32 bits.

Uma observação importante é que para cada byte transmitido, ou seja, a cada 8 bits transmitidos são acrescentados 2 bits que definem o começo e o fim da transmissão.

Logo a quantidade de bits transmitidos não é igual a 32 bits (tamanho do pacote), e sim igual a 40 bits, somando assim para cada 8 bits, 2 bits adicionais. Tal observação pode ser vista no quadro 28.

32 bits → 4 bytes 1 byte → 8 bits 8 bits + bit de início + bit de término → 10 bits 4 bytes transmitidos, portanto 40 bits serão transmitidos

Quadro 28 – Bits transmitidos por pacote

Em (17) pode-se visualizar o tempo necessário para realizar o envio do pacote, ou seja, o tempo do ciclo de simulação (*Cs*), pois se sabe através do quadro 28 que 40 bits são transmitidos por pacote, e como visto acima, a taxa de transmissão é de 9600 bps.

Logo o tempo de transmissão é a quantidade de bits transmitidos pela taxa de transmissão.

$$tempo = 40bits / 9600bps$$
$$tempo = 4{,}17ms \qquad (17)$$

A representação de um nó mostrada na figura 29 evidencia seus componentes e o trajeto que o dado percorre. O dado é captado pelo sensor (*S*), que o armazena em um buffer (*b*) de memória (*M*) e o processa em (P). Por fim o transmissor (*Tx*) envia o pacote adiante.

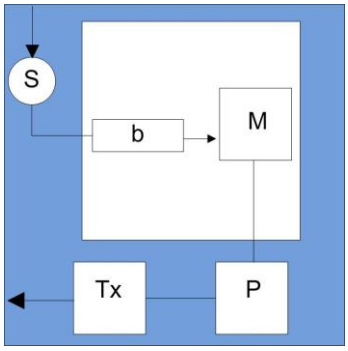


Figura 29 – Representação de um nó

Outro dado importante é a taxa de amostragem do sensor. A taxa está relacionada ao intervalo de tempo entre duas captações de informação que o sensor executa. O intervalo de tempo depende da velocidade de recepção das informações.

Geralmente a taxa de envio do transmissor é maior do que a taxa de recepção do sensor, evidenciando assim a dificuldade de encontrar um nó em uma situação de sobrecarga de dados.

Cada pacote enviado determina um ciclo, através do valor encontrado em (17) e acrescentando uma taxa de variação, sabe-se que cada ciclo possuirá 5 ms. Para encontrar o tempo entre amostragens basta definir quantos ciclos de simulação (*Cs*) será necessário para uma amostragem, o que pode ser visto em (18) para uma quantidade igual a 20.

$$\begin{aligned} \text{Tempo entre Amostragens} &= Cs * \text{Quantidade de ciclos entre amostragens} \\ \text{Tempo entre Amostragens} &= 0{,}005 * 20 = 0{,}1s \end{aligned} \quad (18)$$

Os possíveis valores utilizados de quantidade de ciclos entre amostragens podem ser vistos na tabela 2.

Quantidade de ciclos entre as amostragens	Tempo entre amostragens
1	0,005 s
2	0,01 s
20	0,1 s

Tabela 2 – Tempo entre amostragens

O resumo das variáveis que serão utilizadas no sistema descrito, com seus respectivos valores, é mostrado no quadro 29.

Tempo de Transmissão (*Tp)* → 4,7ms

Tempo de cada ciclo de simulação (*Cs)* → 5ms (*Tp*+variação)

Tempo entre amostragens ou Tempo de amostragem (*Tm*):

- Pacote a cada 20 (*Cs*) → 1 amostragem a cada 0,1 segundos → 1 geração de pacotes
- Pacote a cada 2 (*Cs*) → 1 amostragem a cada 0,01 segundos → 1 geração de pacotes
- Pacote a cada 1 (*Cs*) → 1 amostragem a cada 0,005 segundos → 1 geração de pacotes

Quadro 29 – Valores associados as variáveis

Para cada ciclo a geração de pacotes é feita utilizando a distribuição de Poisson com *lambda* igual a 1, já explicada na Seção 2.2.1.1.1.

3. MATERIAL E MÉTODOS

Neste Capítulo são apresentados os materiais e métodos. Na primeira seção são apresentados todos os materiais que foram utilizados durante o desenvolvimento do projeto.

Na segunda seção do capítulo é apresentada a metodologia, ou seja, quais foram os procedimentos realizados, e como eles foram executados para desenvolver tanto os estudos quanto a parte de simulações das redes de sensores.

3.1. Material

O projeto teve como objetivo principal desenvolver um estudo das redes de sensores e realizar uma simulação de tais redes em superfícies não-planas. Para alcançar os objetivos, os materiais utilizados, por se tratar de um estudo e uma simulação, envolveram livros, artigos e documentos, responsáveis por dar o suporte teórico a realização do experimento.

Todos os livros, artigos e documentos utilizados podem ser vistos nas Referências Bibliográficas descritas no final do documento. Foram escolhidos materiais de diversas áreas desse ramo de pesquisa para dar uma base teórica fundamentada nas características das redes de sensores, seu aspecto histórico e seu estado da arte.

Na parte relacionada a simulação, além dos documentos que continham estudos que auxiliariam na geração do terreno, na distribuição dos nós e na geração de pacotes, ferramentas e um PC (Computador Pessoal) foi utilizado.

O PC é um modelo Inspiron 1545, que tem a empresa Dell como seu fabricante, tem como processador um Pentium(R) Dual-Core CPU T4300 de 2.10 GHz, uma memória instalada (RAM) de 4,00 GB com um sistema operacional de 64 bits, denominado Microsoft Windows 7.

Entre as principais ferramentas utilizadas durante o desenvolvimento está o MATLAB, linguagem de programação para operações matemáticas avançadas, descrita na parte introdutória da seção 2.3 da monografia. O MATLAB foi utilizado para realizar todas as simulações contidas no projeto.

A ferramenta denominada Microsoft Word 2007 também foi utilizada, com o objetivo de apresentar os dados e informações, obtidos no desenvolvimento do projeto, de maneira textual, descrevendo e explicando pontos importantes através de textos, gráficos, tabelas, figuras e quadros.

3.2. Metodologia

Inicialmente, além do desenvolvimento da simulação de redes de sensores em superfícies não planas, foi proposto o estudo das redes de sensores. Tal estudo tinha como intuito dar uma base teórica para a execução da simulação das redes.

O estudo das redes de sensores abrangia exatamente uma pesquisa detalhada do tema proposto, com base em definições, aspectos teóricos, áreas de aplicação, exemplos de aplicação, características de funcionamento, de como ocorre a configuração, de que maneira ocorre o sensoriamento, o processamento e a comunicação, informações referentes a topologias, entre outros.

Depois de finalizado o estudo, começaram as fases de desenvolvimento relacionadas a simulação. O algoritmo da simulação envolvia a geração do terreno, a distribuição de nós e a geração de pacotes, além de um algoritmo envolvido na troca de informações.

Na parte de simulação inúmeros modelos matemáticos foram estudados para se chegar a escolha dos modelos utilizados, tais escolhas são mostradas abaixo. Como a linguagem de programação ficou definida desde o início que seria MATLAB, funções implementadas em MATLAB foram pesquisadas e testadas para adequar aos problemas relacionados as simulações do projeto.

Na geração do terreno, os estudos envolveram distribuições de variáveis aleatórias contínuas, como a distribuição normal. Tal distribuição auxiliaria no entendimento da formação das curvas do terreno, com relação a suas montanhas e seus vales.

As funções de distribuição normal foram implementadas e testadas no ambiente de desenvolvimento MATLAB para que o terreno fosse gerado.

A distribuição uniforme também foi pesquisada e utilizada no projeto, a distribuição de nós aconteceria através de distribuições uniformes, onde cada nó teria a mesma probabilidade de ser alocado em posições diferentes.

Depois da geração do terreno os nós foram distribuídos de maneira uniforme, onde foram definidas as conexões entre eles, através do raio de alcance e as características presentes no relevo.

As funções, de distribuição uniforme, responsáveis pelo posicionamento dos nós e pela verificação das conexões entre eles foram implementadas e testadas no MATLAB para simular essa fase do desenvolvimento.

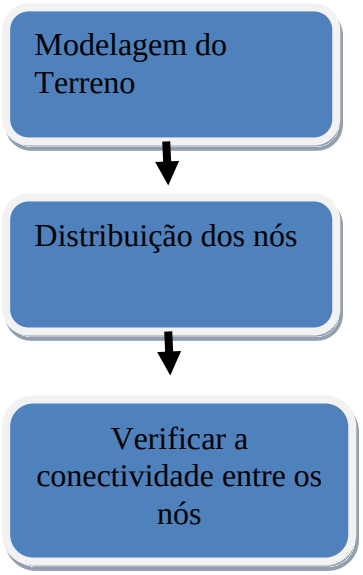
Com a definição do terreno e as informações da distribuição dos nós, foi tratado a partir daí a simulação da geração e transmissão dos pacotes. A simulação da geração e transmissão dos pacotes foi feita através de uma distribuição de variável aleatória discreta, a distribuição de Poisson.

Além da distribuição de Poisson, a taxa de transmissão dos nós, o tamanho do pacote, o tempo do ciclo de simulação, a taxa de amostragem do sensor e a quantidade de ciclos entre as amostragens foram parâmetros analisados e definidos quando foram iniciadas as simulações das gerações e as transmissões dos pacotes.

As simulações dos eventos de geração e transmissão de pacotes foram feitas com base em inúmeras simulações, com diferentes topologias da rede, distintas quantidades de nós e diversos raios de alcance.

Em cada simulação o nó gera uma quantidade variada de pacotes em função da distribuição de Poisson, armazena-o em uma fila e envia ao seu antecessor. Tal pacote é processado, armazenado e enviado até chegar ao nó sink (nó servidor), quando isso acontece ele armazena-o na fila e processa o primeiro elemento da mesma.

No diagrama 1 pode-se visualizar uma representação gráfica da metodologia apresentada.



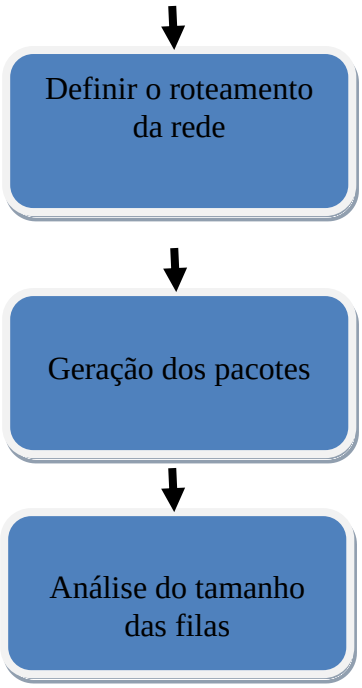


Diagrama 1 – Representação gráfica das atividades

4. EXPERIMENTOS

Neste capítulo são descritos os experimentos realizados para avaliar o trabalho realizado.

Os experimentos se baseiam na geração de um terreno, com um vale e uma montanha, e na distribuição dos nós ao longo do relevo desse terreno. A distribuição utilizada para esparramar os nós foi a uniforme.

Foram utilizados cinco valores diferentes para a quantidade de nós distribuídos, foram *20*, *50*, *100*, *200*, *500*. Em relação ao raio de alcance foram utilizados onze valores distintos, entre eles foram *0.0004, 0.0020, 0.004, 0.020, 0.04, 0.2, 0.4, 0.8, 1.2, 1.6* e *2.0*.

Para avaliar os parâmetros da rede é feito um arquivo com extensão .txt de saída que armazena todos os valores.

Com base em um malha de pontos nos eixos x e y, começa-se a delinear o terreno estudado. O ponto z responsável por definir a altimetria do relevo é definido através de funções do MATLAB.

Os nós são distribuídos uniformemente e tem como nó *sink*, o nó de índice 1. Ele está posicionado em 0 no eixo x e 0,5 no eixo y, ou seja, tem como coordenada o ponto (0, 0.5,z), onde z depende do terreno.

As mesmas funções utilizadas para definir a altimetria do terreno são utilizadas para definir a altimetria dos nós, para que na representação visual dê a sensação de que os nós estão sobre o terreno.

Depois disso sabe-se o posicionamento dos nós e a distância entre eles. A distância entre eles é comparada com o raio de alcance, o resultado dessa comparação define a conectividade entre eles.

Para a distribuição dos nós foram testadas inúmeras topologias de rede que mostrariam diversas situações.

Por se tratar de uma superfície não plana, a distância entre eles dependerá de diversos fatores. Por exemplo, se há uma montanha ou um vale entre eles, pois dependendo de qual seja altimetria encontrada, há interferência nessa comunicação.

Isso faria com que o sinal de transmissão se propague de maneira diferente da retilínea.

Para dar o efeito dessa influência do terreno, a distância entre os nós foi dividida em dez partes, que seguiam a superfície do terreno, fazendo com que curvas de distâncias fossem formadas.

Para descobrir a conectividade da rede o algoritmo de busca em largura foi utilizado. Cada nó que estiver na rede estará no vetor VS, e o vetor VR será o vetor de roteamento, onde cada nó conectado terá um nó antecessor.

Para a geração de pacotes alguns blocos de simulação foram executados, em cada bloco tinha um número grande de ciclos de simulação. Cada nó gera um pacote, onde o armazena em uma fila, antes de ser enviado ao seu antecessor.

Para a conclusão, são verificadas algumas características das filas, como a média, o desvio padrão, o valor máximo dos tamanhos de filas encontrados.

5. RESULTADOS

Os resultados encontrados se baseiam nos arquivos com extensão .txt executados. Os arquivos mostram os resultados obtidos a partir de diferentes valores de quantidade de nós e raios de alcance.

Abaixo são mostrados três exemplos completos, da execução do programa contido no APÊNDICE A, para uma determinada quantidade de nós e um raio de alcance.

Para facilitar a representação do terreno abaixo, a quantidade de nós será definida como *N* e o raio de alcance como *R*.

Além dos resultados apresentados, na seção 6.4 pode-se visualizar uma síntese dos resultados apresentados.

5.1. Exemplo 1

O exemplo 1 tem como características principais 20 nós distribuídos e raio de alcance 1,2.

O posicionamento dos nós pode ser visto no quadro 30, com os respectivos índices dos nós.

1) x = 0.0000 || y = 0.5000 || z = 0.0000
2) x = -1.8686 || y = 0.6712 || z = -0.0363
3) x = -1.2826 || y = 1.4412 || z = -0.0310
4) x = 0.4030 || y = -1.2083 || z = 0.0796
5) x = 0.8768 || y = 0.6899 || z = 0.2525
6) x = -1.2463 || y = -1.0220 || z = -0.0928
7) x = -1.3263 || y = 0.3969 || z = -0.1951
8) x = -0.8760 || y = 0.2274 || z = -0.3862
9) x = 0.8771 || y = 1.1173 || z = 0.1166
10) x = 1.8450 || y = -1.3780 || z = 0.0092
11) x = 0.0810 || y = -1.9551 || z = 0.0018
12) x = 0.1258 || y = -0.8754 || z = 0.0576
13) x = -0.4171 || y = -1.2534 || z = -0.0728
14) x = -0.0226 || y = 1.2425 || z = -0.0048

15) x = 1.1327 || y = -1.8334 || z = 0.0109

16) x = -1.1471 || y = 1.2444 || z = -0.0654

17) x = 0.4772 || y = 0.7457 || z = 0.2179

18) x = 1.7151 || y = -0.2402 || z = 0.0855

19) x = -1.4826 || y = 0.7769 || z = -0.0900

20) x = 0.1224 || y = 0.2496 || z = 0.1133

Quadro 30 – Posicionamento dos nós (Exemplo 1)

A matriz com as distâncias entre todos os nós pode ser vista no quadro 31. Para cada nó são mostradas as distâncias relacionadas com os outros nós na ordem por índice.

1) 0.0000 1.7344 1.4551 1.6014 0.8804 1.8002 1.2436 0.9114 1.0081 2.5298 2.2113 1.2460 1.6271 0.6686 2.4203 1.2544 0.5328 1.8440 1.3953 0.2703

2) 1.7344 0.0000 0.8709 2.6598 2.4865 1.6251 0.5648 1.0318 2.5108 3.9143 2.9477 2.2764 2.1722 1.7406 3.5272 0.8298 2.1259 3.4090 0.3633 1.8387

3) 1.4551 0.8709 0.0000 2.8317 2.0728 2.2194 0.9527 1.1995 1.9708 3.8653 3.2989 2.4442 2.5504 1.1486 3.6804 0.2172 1.7170 3.1795 0.6267 1.6640

4) 1.6014 2.6598 2.8317 0.0000 1.8268 1.5028 2.1556 1.8037 2.2235 1.3102 0.7367 0.3910 0.7523 2.2669 0.8672 2.6344 1.8058 1.4861 2.4745 1.3493

5) 0.8804 2.4865 2.0728 1.8268 0.0000 2.5563 2.0912 1.7907 0.4040 2.0898 2.5678 1.6397 2.2097 0.9812 2.3431 1.9224 0.3658 1.1446 2.1700 0.8228

6) 1.8002 1.6251 2.2194 1.5028 2.5563 0.0000 1.2845 1.2029 2.7373 2.8139 1.4629 1.2507 0.7753 2.3288 2.2676 2.0471 2.2481 2.8230 1.6356 1.6977

7) 1.2436 0.5648 0.9527 2.1556 2.0912 1.2845 0.0000 0.4661 2.1293 3.4091 2.4861 1.7656 1.7081 1.4111 3.0391 0.7877 1.6997 2.9317 0.3822 1.3475

8) 0.9114 1.0318 1.1995 1.8037 1.7907 1.2029 0.4661 0.0000 1.8532 3.0445 2.1786 1.3963 1.4228 1.2373 2.6732 0.9924 1.4089 2.6024 0.7850 0.9946

9) 1.0081 2.5108 1.9708 2.2235 0.4040 2.7373 2.1293 1.8532 0.0000 2.4435 2.9567 1.9987 2.5259 0.8265 2.7425 1.8392 0.5013 1.4483 2.1721 1.0604

10) 2.5298 3.9143 3.8653 1.3102 2.0898 2.8139 3.4091 3.0445 2.4435 0.0000 1.6707 1.6231 2.0500 3.0158 0.7609 3.7819 2.3334 1.0327 3.7303 2.2128

11) 2.2113 2.9477 3.2989 0.7367 2.5678 1.4629 2.4861 2.1786 2.9567 1.6707 0.0000 0.9737 0.7776 2.8800 0.9530 3.0858 2.4887 2.1417 2.8350 1.9878

12) 1.2460 2.2764 2.4442 0.3910 1.6397 1.2507 1.7656 1.3963 1.9987 1.6231 0.9737 0.0000 0.6108 1.9140 1.2572 2.2346 1.5260 1.6027 2.0921 1.0153

13) 1.6271 2.1722 2.5504 0.7523 2.2097 0.7753 1.7081 1.4228 2.5259 2.0500 0.7776 0.6108 0.0000 2.2792 1.4968 2.3508 2.0005 2.1874 2.0712 1.4473

14) 0.6686 1.7406 1.1486 2.2669 0.9812 2.3288 1.4111 1.2373 0.8265 3.0158 2.8800 1.9140 2.2792 0.0000 3.0582 1.0159 0.6626 2.1275 1.3850 0.9087

15) 2.4203 3.5272 3.6804 0.8672 2.3431 2.2676 3.0391 2.6732 2.7425 0.7609 0.9530 1.2572 1.4968 3.0582 0.0000 3.5443 2.4628 1.5291 3.3735 2.1281

16) 1.2544 0.8298 0.2172 2.6344 1.9224 2.0471 0.7877 0.9924 1.8392 3.7819 3.0858 2.2346 2.3508 1.0159 3.5443 0.0000 1.5497 3.0412 0.5185 1.4626

17) 0.5328 2.1259 1.7170 1.8058 0.3658 2.2481 1.6997 1.4089 0.5013 2.3334 2.4887 1.5260 2.0005 0.6626 2.4628 1.5497 0.0000 1.4697 1.7953 0.5613

18) 1.8440 3.4090 3.1795 1.4861 1.1446 2.8230 2.9317 2.6024 1.4483 1.0327 2.1417 1.6027 2.1874 2.1275 1.5291 3.0412 1.4697 0.0000 3.2661 1.5998

19) 1.3953 0.3633 0.6267 2.4745 2.1700 1.6356 0.3822 0.7850 2.1721 3.7303 2.8350 2.0921 2.0712 1.3850 3.3735 0.5185 1.7953 3.2661 0.0000 1.5365

20) 0.2703 1.8387 1.6640 1.3493 0.8228 1.6977 1.3475 0.9946 1.0604 2.2128 1.9878 1.0153 1.4473 0.9087 2.1281 1.4626 0.5613 1.5998 1.5365 0.0000

Quadro 31 – Matriz de distâncias (Exemplo 1)

A matriz de conectividade entre os sensores pode ser vista no quadro 32, para uma determinada conexão entre um nó *i* e um nó *j*, a posição da matriz correspondente a (*i*,*j*) e (*j*,*i*) terá valor, caso contrário valor 0.

1) 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1

2) 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0

3) 0 1 1 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0

4) 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0

5) 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 1

6) 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0

7) 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0

8) 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1

9) 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1

10) 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0

11) 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0

12) 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1

13) 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0

14) 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0 0 1

15) 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0

16) 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0

17) 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1

18) 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0

19) 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0

20) 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 1 0 0 1

Quadro 32 – Matriz de conectividade (Exemplo 1)

O vetor de sensores conectados é mostrado no quadro 33.

```
VS = 1 5 8 9 14 17 20 18 2 3 7 16 19 12 10 4 11 13 15 6
```

Quadro 33 – Vetor de sensores conectados (Exemplo 1)

O vetor de roteamento entre os sensores é mostrado no quadro 34.

```
VR = 0 8 8 12 1 13 8 1 1 18 12 20 12 1 10 8 1 5 8 1
```

Quadro 34 – Vetor de roteamento (Exemplo 1)

No quadro 35 são mostradas as características do vetor que contém as quantidades máximas encontradas na fila durante cada bloco de simulação.

```
Nó :  1
 Media LqA :  2578.07
 Desvio-Padrao LqA :  23.81
 Valor Absoluto LqA :  2614.00

Nó :  5
 Media LqA :  521.30
 Desvio-Padrao LqA :  22.07
 Valor Absoluto LqA :  555.00

Nó :  8
 Media LqA :  2541.40
 Desvio-Padrao LqA :  35.05
 Valor Absoluto LqA :  2617.00

Nó :  9
 Media LqA :  26.33
 Desvio-Padrao LqA :  12.06
 Valor Absoluto LqA :  55.00

Nó :  14
 Media LqA :  28.10
 Desvio-Padrao LqA :  12.70
 Valor Absoluto LqA :  62.00
```

Nó : 17
Media LqA : 30.57
Desvio-Padrao LqA : 13.36
Valor Absoluto LqA : 56.00

Nó : 20
Media LqA : 523.07
Desvio-Padrao LqA : 22.79
Valor Absoluto LqA : 565.00

Nó : 18
Media LqA : 528.10
Desvio-Padrao LqA : 24.05
Valor Absoluto LqA : 569.00

Nó : 2
Media LqA : 25.83
Desvio-Padrao LqA : 12.76
Valor Absoluto LqA : 68.00

Nó : 3
Media LqA : 24.57
Desvio-Padrao LqA : 9.13
Valor Absoluto LqA : 41.00

Nó : 7
Media LqA : 28.70
Desvio-Padrao LqA : 13.05
Valor Absoluto LqA : 58.00

Nó : 16
Media LqA : 29.63
Desvio-Padrao LqA : 13.03
Valor Absoluto LqA : 66.00

Nó : 19
Media LqA : 26.20
Desvio-Padrao LqA : 8.63
Valor Absoluto LqA : 43.00

Nó : 12

Media LqA : 1543.60
Desvio-Padrao LqA : 31.19
Valor Absoluto LqA : 1608.00

Nó : 10
Media LqA : 507.43
Desvio-Padrao LqA : 27.82
Valor Absoluto LqA : 554.00

Nó : 4
Media LqA : 26.70
Desvio-Padrao LqA : 11.60
Valor Absoluto LqA : 53.00

Nó : 11
Media LqA : 26.73
Desvio-Padrao LqA : 9.71
Valor Absoluto LqA : 49.00

Nó : 13
Media LqA : 504.40
Desvio-Padrao LqA : 25.80
Valor Absoluto LqA : 551.00

Nó : 15
Media LqA : 24.47
Desvio-Padrao LqA : 10.68
Valor Absoluto LqA : 47.00

Nó : 6
Media LqA : 26.20
Desvio-Padrao LqA : 9.49
Valor Absoluto LqA : 49.00

Quadro 35 – Informações sobre os valores máximos das filas em cada bloco de simulação (Exemplo 1)

Nos gráficos abaixo podem ser visualizados informações presentes no quadro 35.

No gráfico 1 pode-se visualizar graficamente as informações da média de LqA.

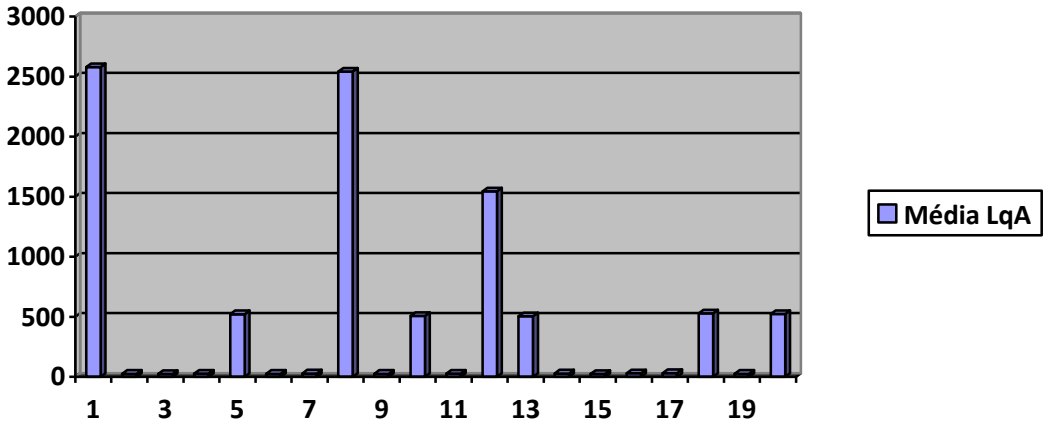


Gráfico 1 – Média de LqA (Exemplo 1)

No gráfico 2 pode-se visualizar graficamente as informações do desvio-padrão de LqA.

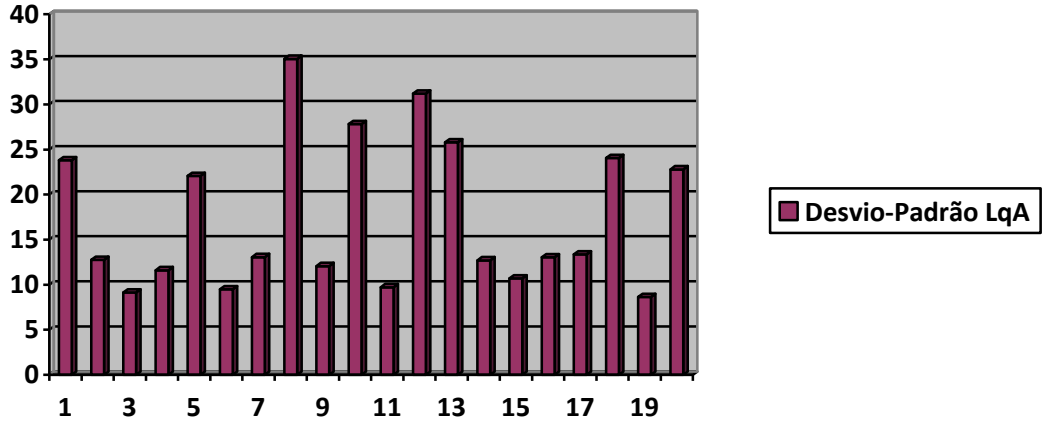


Gráfico 2 – Desvio-Padrão de LqA (Exemplo 1)

No gráfico 3 pode-se visualizar graficamente as informações do valor absoluto de LqA.

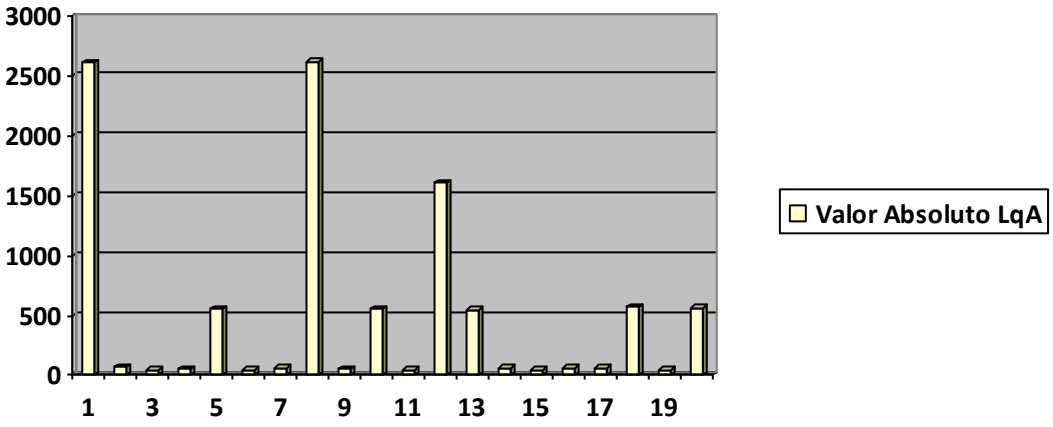


Gráfico 3 – Valor Absoluto LqA (Exemplo 1)

A representação do terreno, a distribuição dos nós e a conectividade entre eles, podem ser vistas na figura 30.

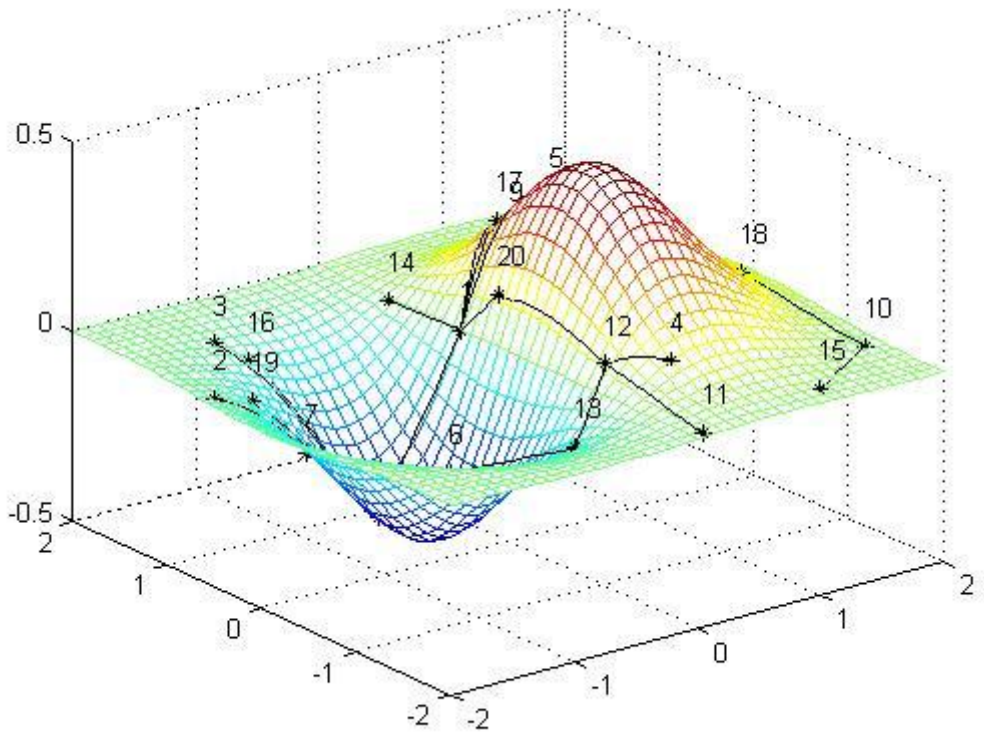


Figura 30 – Simulação Completa com R =1,2 e N = 20

5.2. Exemplo 2

O exemplo 2 tem como características principais 20 nós distribuídos e raio de alcance 0,8.

A matriz de posicionamento dos nós é apresentada no quadro 36.

1) x = 0.0000 || y = 0.5000 || z = 0.0000
2) x = -1.8725 || y = -1.4628 || z = -0.0066
3) x = 0.7121 || y = -1.0177 || z = 0.1522
4) x = 1.7821 || y = 0.0342 || z = 0.0743
5) x = 1.8285 || y = -0.4075 || z = 0.0547
6) x = 0.2377 || y = 0.5397 || z = 0.1679
7) x = -0.1975 || y = -0.3920 || z = -0.1629
8) x = -1.1133 || y = 1.1026 || z = -0.0956
9) x = 0.2926 || y = -0.2416 || z = 0.2533
10) x = -1.8944 || y = -0.2220 || z = -0.0498
11) x = -0.8077 || y = -0.8113 || z = -0.2178
12) x = 0.0523 || y = -0.8817 || z = 0.0240
13) x = -1.5504 || y = 0.9805 || z = -0.0536
14) x = 0.9920 || y = -1.2819 || z = 0.0717
15) x = -1.7898 || y = -1.3059 || z = -0.0132
16) x = -0.1227 || y = -1.6485 || z = -0.0080
17) x = -0.1794 || y = -1.5261 || z = -0.0169
18) x = 0.2261 || y = 0.2517 || z = 0.2016
19) x = 0.5401 || y = 1.7704 || z = 0.0176
20) x = 0.6057 || y = 0.2100 || z = 0.4016

Quadro 36 – Posicionamento dos nós (Exemplo 2)

A Matriz de distâncias entre os sensores é apresentada no quadro 37.

1) 0.0000 2.4876 1.5722 1.8103 2.0080 0.2658 0.8353 1.1655 0.7562 1.8758 1.4135 1.2450 1.4964 1.9293 2.3326 1.9371 1.8314 0.3515 1.2516 0.7181

2) 2.4876 0.0000 2.3644 3.6077 3.4889 2.6227 1.7954 2.4092 2.2482 1.1176 1.1415 1.8098 2.2184 2.5846 0.1597 1.5836 1.5249 2.4456 3.6319 2.7135

3) 1.5722 2.3644 0.0000 1.3654 1.1534 1.5107 1.0582 2.5805 0.8093 2.4655 1.4342 0.6202 2.7556 0.3540 2.2724 0.9535 0.9362 1.2553 2.6080 1.1499

4) 1.8103 3.6077 1.3654 0.0000 0.4001 1.5252 2.0041 2.9326 1.4322 3.5352 2.6540 1.8364 3.3086 1.3868 3.5715 2.3259 2.3042 1.4900 1.9312 1.1266

5) 2.0080 3.4889 1.1534 0.4001 0.0000 1.7423 1.9742 3.1821 1.4516 3.5248 2.5497 1.7021 3.5252 1.0908 3.4270 2.0976 2.0912 1.6406 2.2975 1.2885
6) 0.2658 2.6227 1.5107 1.5252 1.7423 0.0000 0.9760 1.3651 0.7127 2.0902 1.5810 1.3012 1.7071 1.8512 2.5047 2.0067 1.9060 0.2614 1.1496 0.4974
7) 0.8353 1.7954 1.0582 2.0041 1.9742 0.9760 0.0000 1.5994 0.5932 1.5735 0.6769 0.5266 1.7693 1.4244 1.6762 1.1441 1.0307 0.7661 2.0932 1.0665
8) 1.1655 2.4092 2.5805 2.9326 3.1821 1.3651 1.5994 0.0000 1.7823 1.3854 1.7539 2.0834 0.4101 2.9018 2.2564 2.6454 2.5236 1.4546 1.6086 1.7983
9) 0.7562 2.2482 0.8093 1.4322 1.4516 0.7127 0.5932 1.7823 0.0000 2.0521 1.2147 0.6500 2.0520 1.1562 2.1775 1.3469 1.2623 0.4508 1.8538 0.5209
10) 1.8758 1.1176 2.4655 3.5352 3.5248 2.0902 1.5735 1.3854 2.0521 0.0000 1.1230 1.8543 1.1259 2.7808 0.9806 2.0492 1.9411 1.9719 2.8375 2.3216
11) 1.4135 1.1415 1.4342 2.6540 2.5497 1.5810 0.6769 1.7539 1.2147 1.1230 0.0000 0.8032 1.7586 1.7093 1.0113 0.9956 0.8780 1.3872 2.6463 1.6658
12) 1.2450 1.8098 0.6202 1.8364 1.7021 1.3012 0.5266 2.0834 0.6500 1.8543 0.8032 0.0000 2.2281 0.9343 1.7174 0.7090 0.6183 1.0454 2.4622 1.1680
13) 1.4964 2.2184 2.7556 3.3086 3.5252 1.7071 1.7693 0.4101 2.0520 1.1259 1.7586 2.2281 0.0000 3.0808 2.0703 2.6992 2.5776 1.7453 2.0130 2.1010
14) 1.9293 2.5846 0.3540 1.3868 1.0908 1.8512 1.4244 2.9018 1.1562 2.7808 1.7093 0.9343 3.0808 0.0000 2.5078 1.0592 1.0810 1.5922 2.8789 1.4400
15) 2.3326 0.1597 2.2724 3.5715 3.4270 2.5047 1.6762 2.2564 2.1775 0.9806 1.0113 1.7174 2.0703 2.5078 0.0000 1.5318 1.4630 2.3008 3.4753 2.5765
16) 1.9371 1.5836 0.9535 2.3259 2.0976 2.0067 1.1441 2.6454 1.3469 2.0492 0.9956 0.7090 2.6992 1.0592 1.5318 0.0000 0.1216 1.7478 3.1674 1.8362
17) 1.8314 1.5249 0.9362 2.3042 2.0912 1.9060 1.0307 2.5236 1.2623 1.9411 0.8780 0.6183 2.5776 1.0810 1.4630 0.1216 0.0000 1.6516 3.0638 1.7574
18) 0.3515 2.4456 1.2553 1.4900 1.6406 0.2614 0.7661 1.4546 0.4508 1.9719 1.3872 1.0454 1.7453 1.5922 2.3008 1.7478 1.6516 0.0000 1.4084 0.3963
19) 1.2516 3.6319 2.6080 1.9312 2.2975 1.1496 2.0932 1.6086 1.8538 2.8375 2.6463 2.4622 2.0130 2.8789 3.4753 3.1674 3.0638 1.4084 0.0000 1.4433
20) 0.7181 2.7135 1.1499 1.1266 1.2885 0.4974 1.0665 1.7983 0.5209 2.3216 1.6658 1.1680 2.1010 1.4400 2.5765 1.8362 1.7574 0.3963 1.4433 0.0000

Quadro 37 – Matriz de distâncias (Exemplo 2)

A matriz de conectividade entre os sensores é apresentada no quadro 38.

1) 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1
2) 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
3) 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
4) 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5) 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6) 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1
7) 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0
8) 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0

9) 1 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 0 1
10) 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
11) 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
12) 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 1 0 0 0
13) 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0
14) 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
15) 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
16) 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0
17) 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0
18) 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 1
19) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
20) 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1

Quadro 38 – Matriz de conectividade (Exemplo 2)

O vetor de sensores conectados é apresentado no quadro 39.

VS = 1 6 9 18 20 7 12 11 3 16 17 14

Quadro 39 – Vetor de sensores conectados (Exemplo 2)

O vetor de roteamento entre os sensores é apresentado no quadro 40.

VR = 0 0 12 0 0 1 9 0 1 0 7 9 0 3 0 12 12 1 0 1

Quadro 40 – Vetor de roteamento (Exemplo 2)

No quadro 41 são apresentadas características dos máximos valores de fila encontrados em cada bloco de simulação.

Nó : 1
Media LqA : 2657.57
Desvio-Padrao LqA : 31.86
Valor Absoluto LqA : 2712.00

Nó : 6
Media LqA : 34.47

Desvio-Padrao LqA : 14.20
Valor Absoluto LqA : 72.00

Nó : 9
Media LqA : 1819.83
Desvio-Padrao LqA : 24.11
Valor Absoluto LqA : 1878.00

Nó : 18
Media LqA : 31.87
Desvio-Padrao LqA : 8.90
Valor Absoluto LqA : 51.00

Nó : 20
Media LqA : 39.93
Desvio-Padrao LqA : 16.43
Valor Absoluto LqA : 71.00

Nó : 7
Media LqA : 903.17
Desvio-Padrao LqA : 31.63
Valor Absoluto LqA : 957.00

Nó : 12
Media LqA : 2687.13
Desvio-Padrao LqA : 31.94
Valor Absoluto LqA : 2744.00

Nó : 11
Media LqA : 37.80
Desvio-Padrao LqA : 14.74
Valor Absoluto LqA : 67.00

Nó : 3
Media LqA : 901.10
Desvio-Padrao LqA : 27.09
Valor Absoluto LqA : 962.00

Nó : 16
Media LqA : 37.93
Desvio-Padrao LqA : 16.09

Valor Absoluto LqA : 71.00

Nó : 17

Media LqA : 36.93

Desvio-Padrao LqA : 14.62

Valor Absoluto LqA : 62.00

Nó : 14

Media LqA : 38.53

Desvio-Padrao LqA : 16.21

Valor Absoluto LqA : 77.00

Quadro 41 – Informações sobre os valores máximos das filas em cada bloco de simulação (Exemplo 2)

Nos gráficos abaixo podem ser visualizados informações presentes no quadro 41.

No gráfico 4 pode-se visualizar graficamente as informações da média de LqA.

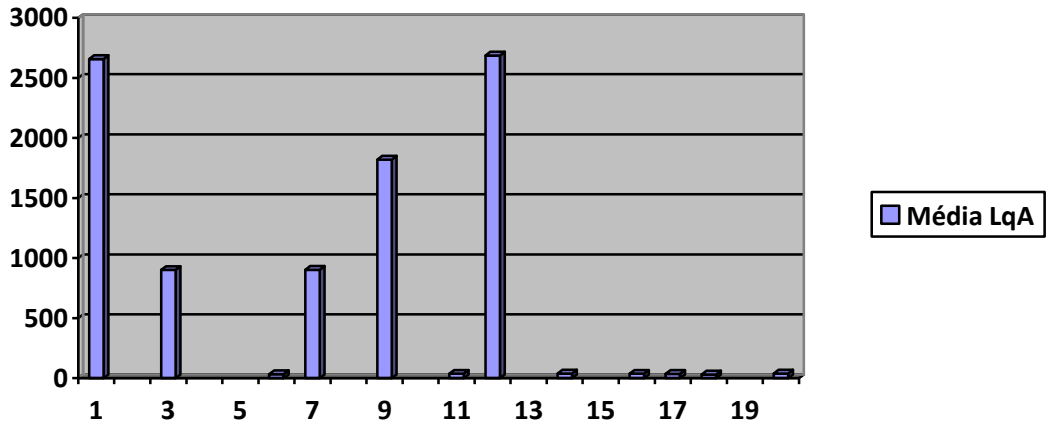


Gráfico 4 – Média de LqA (Exemplo 2)

No gráfico 5 pode-se visualizar graficamente as informações do desvio-padrão de LqA.

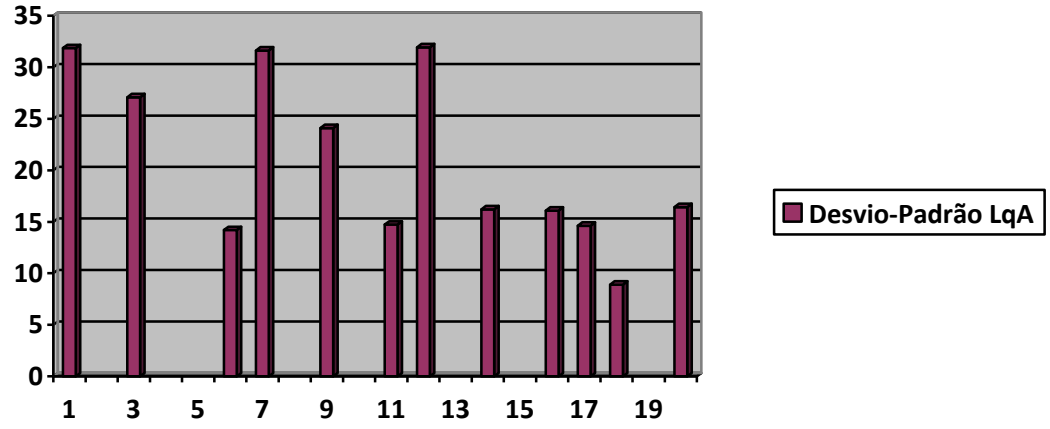


Gráfico 5 – Desvio-Padrão de LqA (Exemplo 2)

No gráfico 6 pode-se visualizar graficamente as informações do valor absoluto de LqA.

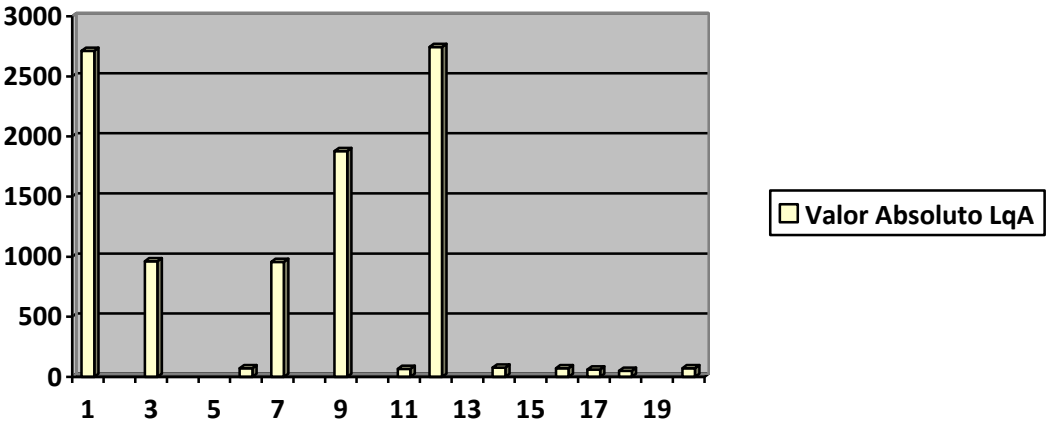


Gráfico 6 – Valor Absoluto LqA (Exemplo 2)

A representação do terreno, a distribuição dos nós e a conectividade entre eles, do exemplo 2, podem ser vistas na figura 31.

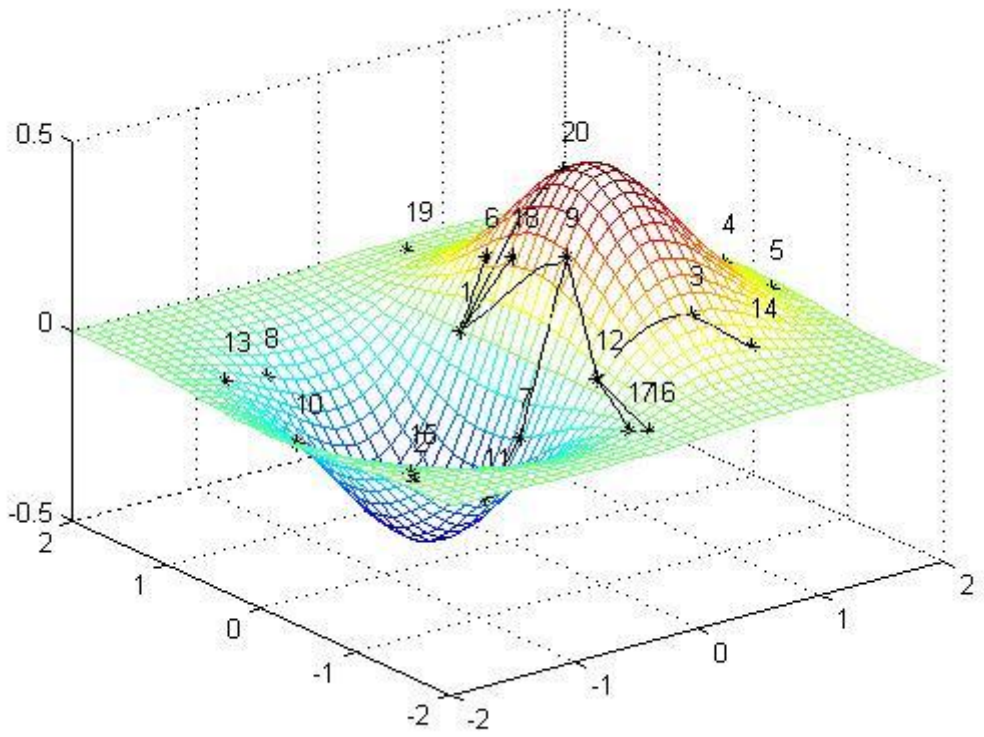


Figura 31 – Simulação Completa com R =0,8 e N = 20

5.3. Exemplo 3

A quantidade de sensores testados nesse exemplo foi de 20 e o alcance do rádio de transmissão era de 0.2.

A matriz de posicionamento dos sensores é apresentada no quadro 42.

1) x = 0.0000 || y = 0.5000 || z = 0.0000
2) x = -0.3190 || y = -1.3782 || z = -0.0431
3) x = 1.7323 || y = -0.1468 || z = 0.0843
4) x = 1.1167 || y = 1.4674 || z = 0.0372
5) x = -0.2009 || y = -0.5497 || z = -0.1426
6) x = 1.1266 || y = -1.5554 || z = 0.0282
7) x = 1.1765 || y = 0.4646 || z = 0.2375
8) x = 0.0117 || y = 0.3813 || z = 0.0101
9) x = -0.6370 || y = -1.8957 || z = -0.0117
10) x = -1.7573 || y = 0.7721 || z = -0.0441
11) x = 1.7808 || y = 0.1015 || z = 0.0739

12) x = 0.5298 || y = 0.9089 || z = 0.1752

13) x = 1.8749 || y = -1.4006 || z = 0.0078

14) x = -1.1726 || y = 1.5134 || z = -0.0300

15) x = -0.5155 || y = 0.9876 || z = -0.1490

16) x = -0.0635 || y = -0.0621 || z = -0.0630

17) x = -0.2283 || y = 0.8349 || z = -0.1079

18) x = -1.2999 || y = 1.8698 || z = -0.0073

19) x = 0.7099 || y = -0.7972 || z = 0.2271

20) x = 0.2293 || y = 0.3938 || z = 0.1863

Quadro 42 – Posicionamento dos nós (Exemplo 3)

A matriz de distâncias entre os sensores é apresentada no quadro 43.

1) 0.0000 1.7171 1.8239 1.3701 0.9708 2.2077 1.1598 0.1077 2.2381 1.6427 1.7913 0.6383 2.5636 1.4148 0.6606 0.5123 0.3796 1.7179 1.4004 0.2823

2) 1.7171 0.0000 2.2046 2.9733 0.7585 1.3160 2.2234 1.6129 0.5475 2.3337 2.3813 2.2235 1.9822 2.7197 2.1416 1.2074 1.9963 3.0613 1.0880 1.6813

3) 1.8239 2.2046 0.0000 1.5602 1.9290 1.3827 0.7579 1.7434 2.6861 3.4355 0.2279 1.4689 1.1379 3.1275 2.4224 1.7704 2.1125 3.3739 1.1081 1.5126

4) 1.3701 2.9733 1.5602 0.0000 2.2761 2.7780 0.9207 1.4217 3.5236 2.6689 1.3696 0.7394 2.6866 2.0636 1.5542 1.8022 1.3603 2.2060 2.1339 1.2746

5) 0.9708 0.7585 1.9290 2.2761 0.0000 1.5605 1.7120 0.8692 1.2784 1.8729 2.0809 1.5326 2.1149 2.0755 1.4188 0.4609 1.2485 2.4178 0.9458 0.9784

6) 2.2077 1.3160 1.3827 2.7780 1.5605 0.0000 1.8465 2.0817 1.6173 3.3659 1.6072 2.3596 0.6879 3.5264 2.8396 1.7693 2.5808 3.8737 0.7972 1.9896

7) 1.1598 2.2234 0.7579 0.9207 1.7120 1.8465 0.0000 1.1325 2.7836 2.7540 0.6515 0.7137 1.8088 2.3544 1.6817 1.3419 1.3922 2.5836 1.2406 0.8921

8) 0.1077 1.6129 1.7434 1.4217 0.8692 2.0817 1.1325 0.0000 2.1389 1.6819 1.7743 0.7003 2.4779 1.4994 0.7468 0.4101 0.4765 1.8089 1.3012 0.2533

9) 2.2381 0.5475 2.6861 3.5236 1.2784 1.6173 2.7836 2.1389 0.0000 2.6049 2.8675 2.7453 2.3052 3.1069 2.6006 1.7299 2.4869 3.4423 1.5773 2.2099

10) 1.6427 2.3337 3.4355 2.6689 1.8729 3.3659 2.7540 1.6819 2.6049 0.0000 3.2997 2.0732 3.8979 0.8498 1.1388 1.7026 1.3799 1.0708 2.6480 1.8337

11) 1.7913 2.3813 0.2279 1.3696 2.0809 1.6072 0.6515 1.7743 2.8675 3.2997 0.0000 1.3617 1.3561 3.0283 2.3445 1.8206 2.0423 3.2606 1.2797 1.4892

12) 0.6383 2.2235 1.4689 0.7394 1.5326 2.3596 0.7137 0.7003 2.7453 2.0732 1.3617 0.0000 2.4770 1.6424 0.9918 1.0676 0.7331 1.8758 1.5989 0.5407

13) 2.5636 1.9822 1.1379 2.6866 2.1149 0.6879 1.8088 2.4779 2.3052 3.8979 1.3561 2.4770 0.0000 3.9842 3.2274 2.2021 2.9239 4.2621 1.1955 2.2660

14) 1.4148 2.7197 3.1275 2.0636 2.0755 3.5264 2.3544 1.4994 3.1069 0.8498 3.0283 1.6424 3.9842 0.0000 0.7654 1.7359 1.0492 0.3413 2.6938 1.6262

15) 0.6606 2.1416 2.4224 1.5542 1.4188 2.8396 1.6817 0.7468 2.6006 1.1388 2.3445 0.9918 3.2274 0.7654 0.0000 1.0337 0.2948 1.0723 2.0063 0.9033

16) 0.5123 1.2074 1.7704 1.8022 0.4609 1.7693 1.3419 0.4101 1.7299 1.7026 1.8206 1.0676 2.2021 1.7359 1.0337 0.0000 0.8232 2.0919 1.0485 0.5413

17) 0.3796 1.9963 2.1125 1.3603 1.2485 2.5808 1.3922 0.4765 2.4869 1.3799 2.0423 0.7331 2.9239 1.0492 0.2948 0.8232 0.0000 1.3444 1.7727 0.6253

18) 1.7179 3.0613 3.3739 2.2060 2.4178 3.8737 2.5836 1.8089 3.4423 1.0708 3.2606 1.8758 4.2621 0.3413 1.0723 2.0919 1.3444 0.0000 3.0136 1.9200

19) 1.4004 1.0880 1.1081 2.1339 0.9458 0.7972 1.2406 1.3012 1.5773 2.6480 1.2797 1.5989 1.1955 2.6938 2.0063 1.0485 1.7727 3.0136 0.0000 1.1907

20) 0.2823 1.6813 1.5126 1.2746 0.9784 1.9896 0.8921 0.2533 2.2099 1.8337 1.4892 0.5407 2.2660 1.6262 0.9033 0.5413 0.6253 1.9200 1.1907 0.0000

Quadro 43 – Matriz de distâncias (Exemplo 3)

A matriz de conectividade entre os sensores é apresentada no quadro 44.

1) 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

2) 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3) 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4) 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

5) 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

6) 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

7) 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

8) 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

9) 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

10) 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

11) 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

12) 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

13) 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

14) 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

15) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

16) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

17) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

18) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

19) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

20) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Quadro 44 – Matriz de conectividade (Exemplo 3)

O vetor de sensores conectados é apresentado no quadro 45.

```
VS = 1 8
```

Quadro 45 - Vetor de sensores conectados (Exemplo 3)

O vetor de roteamento entre os sensores é apresentado no quadro 46.

```
VR = 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
```

Quadro 46 – Vetor de roteamento (Exemplo 3)

No quadro 47 são apresentadas características para os vetores que contém as máximas quantidades de fila nos blocos de simulação.

```
Nó :  1
 Media LqA :   0.00
 Desvio-Padrao LqA :   0.00
 Valor Absoluto LqA :   0.00

Nó :  8
 Media LqA :  117.57
 Desvio-Padrao LqA :  43.30
 Valor Absoluto LqA :  212.00
```

Quadro 47 - Informações sobre os valores máximos das filas em cada bloco de simulação (Exemplo 3)

Nos gráficos abaixo podem ser visualizados informações presentes no quadro 47.

No gráfico 4 pode-se visualizar graficamente as informações da média de LqA.

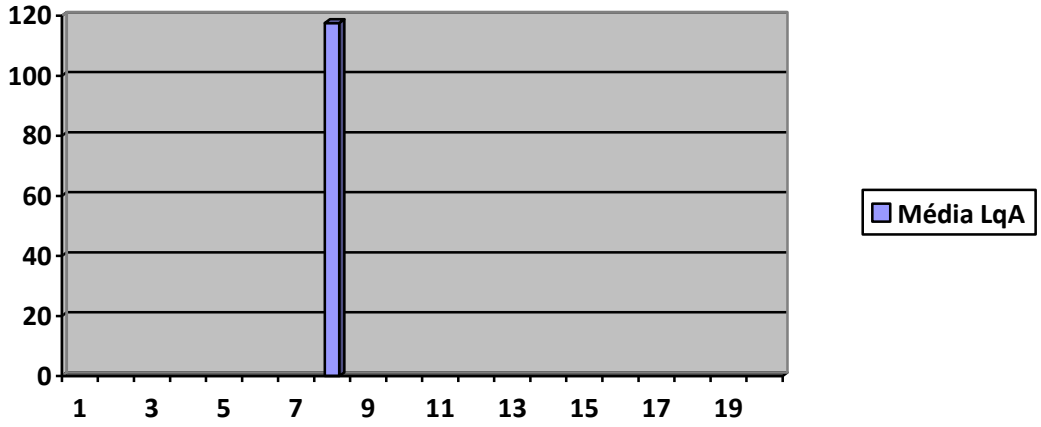


Gráfico 7 – Média de LqA (Exemplo 3)

No gráfico 8 pode-se visualizar graficamente as informações do desvio-padrão de LqA.

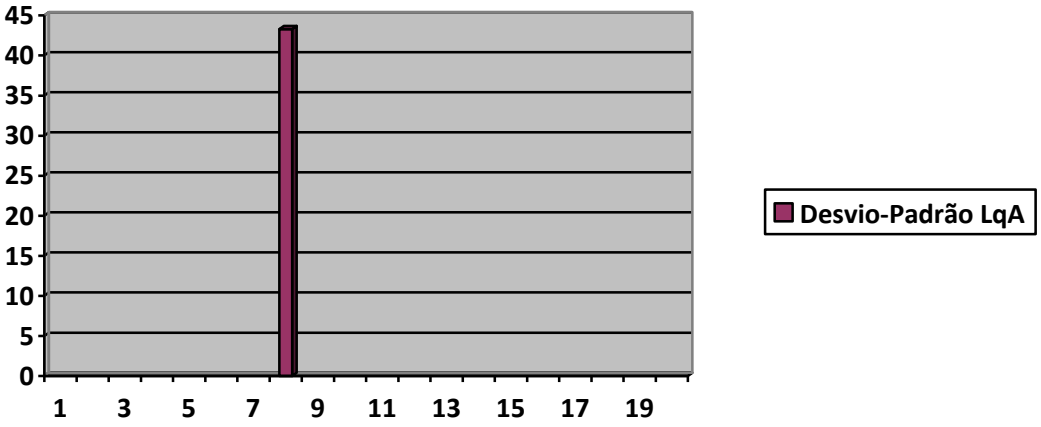


Gráfico 8 – Desvio-Padrão de LqA (Exemplo 3)

No gráfico 9 pode-se visualizar graficamente as informações do valor absoluto de LqA.

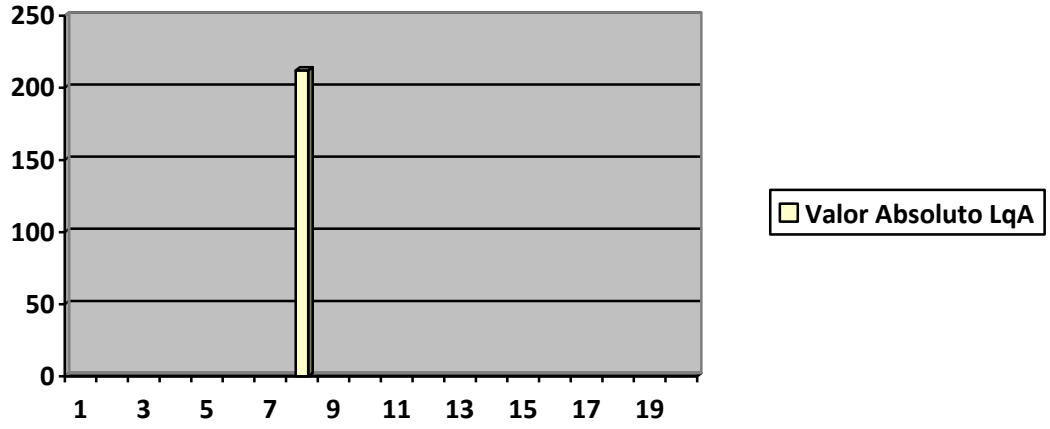


Gráfico 9 – Valor Absoluto LqA (Exemplo 3)

A representação do terreno, a distribuição dos nós e a conectividade entre eles, do exemplo 3, podem ser vistas na figura 32.

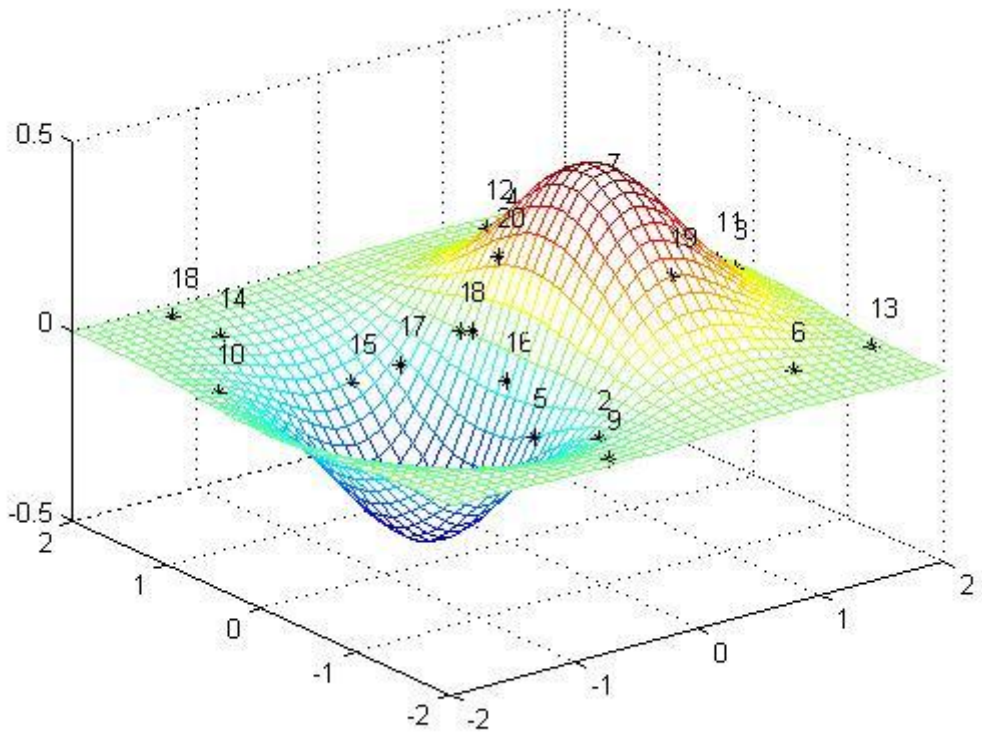


Figura 32 – Simulação Completa com R =0,2 e N = 20

5.4. Síntese dos Resultados

Os resultados apresentados nas seções anteriores consistem basicamente de uma simulação com uma quantidade de nós e raios de alcance específicos. O terreno, onde os nós são distribuídos, é formado por uma equação associada aos valores de x e y de cada nó.

Inicialmente em cada exemplo é apresentado a matriz de posicionamento dos nós, depois a matriz de distâncias e de conectividade da rede. Utilizando as matrizes é executado o algoritmo de roteamento, o qual auxilia no caminho traçado por cada pacote gerado.

Por fim, na geração dos pacotes são observados os tamanhos médios alcançados pelas filas, juntamente com o desvio padrão e o valor absoluto desses valores.

6. CONCLUSÃO

O objetivo deste trabalho foi realizar um estudo de rede de sensores que auxiliasse na simulação das redes de sensores em superfícies não planas. No processo estavam embutidos estudos teóricos, a contextualização histórica e o estado da arte das redes de sensores.

Dentre os objetivos específicos estavam a geração do terreno através de distribuição normal, o espalhamento dos nós através da distribuição uniforme e a geração e transmissão pacotes através da distribuição de Poisson.

Estava também entre os objetivos específicos a análise da quantidade de nós distribuídos, o raio de alcance de cada nó, a definição do terreno, a quantidade de pacotes gerados por cada nó. Para análise dos tamanhos das filas, foram feitos diversos blocos de simulação com milhares de ciclos de simulação com o intuito de gerar e transmitir os pacotes. Essa geração acontece com base na distribuição de Poisson com valor de lambda igual a 1.

Com estes resultados pode-se concluir que o trabalho foi feito corretamente, em relação a quantidade de nós, quanto mais nós estiverem na região estudada menor é o raio de alcance necessário para conectá-los.

O principal problema de se aumentar o raio de alcance é que a bateria dos sensores passa a gastar uma energia maior o que acarretaria uma quantidade maior de nós perdidos com o passar do tempo. Outro ponto importante é que para uma determinada quantidade de nós existe um valor para o raio de alcance que mantém a conectividade da rede, mesmo se aumentado.

O que tal ponto referencia é que se o raio de alcance for aumentado a conectividade praticamente não se altera, ou seja, o aumento é desnecessário.

Já em relação a geração e transmissão de pacotes, devido a características do algoritmo a tendência é que os pacotes fiquem concentrados nos nós mais próximos ao nó sink, nó servidor. Isso acontece por que o algoritmo faz com que o antecessor de cada nó seja o primeiro selecionado ao seu redor.

Tal tendência é evidenciada pelos resultados alcançados.

Em relação a trabalhos futuros, podem-se realizar trabalhos com outros parâmetros. Definir uma influência maior da inclinação do terreno, fazendo com ela dificulte o posicionamento dos nós em uma determinada posição na montanha.

A maneira como a distância entre dois nós e como o raio de alcance é definida pode ser mudada. Podem ser inseridos parâmetros como tempo de fila dos pacotes, o monitoramento da energia dos nós, entre outros.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, A. M. Redes de Sensores Sem Fio. Disponível em: < http://www.gta.ufrj.br/grad/04_2/sensores/>. Acesso em: 02 abr. 2012.

COLWELL, S. Industrial Wireless Sensor Networks: Trends and Development. Disponível em: <http://automation.isa.org/2012/10/industrial-wireless-sensor-networks-trends-and-development/>. Acesso em: 23 out. 2012

HILL, J. L.; CULLER, D. E. MICA: A wireless platform for deeply embedded networks. California: University of California, Berkeley. 2002

LEWIS, F. L. Wireless Sensor Networks. Texas: The University of Texas at Arlington. 2004.

LUO, Y. Cooperative Design, Visualization, and Engineering. Springer 1ª Edição. p. 79 Outubro 7,2009.

LOUREIRO, A. A. F. et al. Redes de Sensores Sem Fio. Minas Gerais: Universidade Federal de Minas Gerais, 2003.

MAINWARING, A. et. al. Wireless Sensor Networks for Habitat Monitoring.

MARTINS, J. E. M. P. **PROJETOS DE SIMULAÇÃO**. Universidade Estadual Paulista. Bauru. 2010.

MATHWORKS. MATLAB Documentation. Disponível em: http://www.mathworks.com/help/releases/R2008a/helpdesk.html. Acesso em: 10 maio 2012.

NETO, A. T. **REDES DE SENSORES SEM FIO E COMPUTAÇÃO UBÍQUA NA AGROPECUÁRIA**. São Carlos. 2009.

PAPA, J. P. **REDES DE COMPUTADORES**. Universidade Estadual Paulista. Bauru. 2010.

RIBEIRO, J.C. **Apostila de Estatística.** Mod. 03, p. 41. Mod. 04, p. 50. Mod. 05, p. 58. Mod. 06, p. 65. 2008.

SEBESTA, R. W. Conceitos de Linguagens de Programação – 5ª Edição. Bookman, 2002.

WILKS, D. S.. Statistical Methods in the Atmospheric Sciences - Third Edition. Academic Press – 2ª edição, 2011

ZIVIANI, N. Projeto de Algoritmos Third Edition. Minas Gerais: Cengage Learning 2010

ÂPENDICE A

```
function Simulacao
%% Cabeçalho
% Nome: Luiz Fernando Ferreira Gomes de Assis RA: 823261
% Curso: Bacharelado em Ciência da Computação
% Prof. Dr. João E. M. Perea Martins

% Não possui variáveis de entrada e de saída

   %% Inicialização das variáveis
   % mudando para o formato com apenas quatro casas decimais
   format short;

   % esta função altera a semente (valor inicial da geração dos
   % números randômicos) dos valores aleatórios gerados pela função
   % rand, usando o vetor clock
   rand('twister',1000*sum(clock));

   % rede_simulacao é o vetor que contém a quantidade de nós analisados
   % durante um determinado experimento
   rede_simulacao = [20 50 100 200 500];

   % quant_rede_simulacao é a quantidade de experimentos realizados para
   % valores distintos de rede_simulação
   quant_rede_simulacao = length(rede_simulacao);

   % R é o alcance do rádio de transmissão dos sensores analisados durante um
   % determinado experimento
   R = [0.0004 0.002 0.004 0.02 0.04 0.2 0.4 0.8 1.2 1.6 2.0];

   % quant_R é a quantidade de experimentos realizados para valores
   % diferentes de R
   quant_R = length(R);

   % quantidade máxima de simulações
   quant_simulacao = 10;

   % fid é uma variável inteira que associa o arquivo criado a seu endereço lógico
   % a função fopen possui dois parâmetros, o primeiro é o arquivo
   % criado, escrito ou lido. O segundo parâmetro declara o que vamos
   % fazer no arquivo, ler ou escrever, neste caso iremos escrever
   % (w - write(escrever em inglês). Lembrando que a extensão do
   % arquivo criado é txt.
   fid = fopen('Simulacao.txt','w');

   %% Início das simulações
   % Simulações feitas para diversas quantidade de nós
   for iter_rede_simulacao = 1:quant_rede_simulacao

      % Escreve no arquivo a quantidade de nos testados
      fprintf(fid,'%s %d','A quantidade de sensores testados  é :
',rede_simulacao(iter_rede_simulacao));
      fprintf(fid,'\r\n\r\n');
```

```
% Simulações realizadas para diversos valores de rádio de
% transmissão
for iter_R = 1:quant_R

    % Escreve no arquivo o alcance do rádio de transmissão
    fprintf(fid,'%s %5.4f','O alcance do rádio de transmissão e : ',R(iter_R));
    fprintf(fid,'\r\n\r\n');

    % Controle das iterações nas simulações realizadas
    for iteracao = 1:quant_simulacao

        % ponto_x é o vetor de coordenadas x dos sensores no plano, tal valor é
        % conseguido através da geração aleatória de valores
        ponto_x = - 2 + 4*rand([1 rede_simulacao(iter_rede_simulacao)]);

        % ponto_y é o vetor de coordenadas y dos sensores no plano, tal valor é
        % conseguido através da geração aleatória de valores
        ponto_y = - 2 + 4*rand([1 rede_simulacao(iter_rede_simulacao)]);

        % o nó com indíce 1 é considerado o sink (nó servidor que
        % recebe as informações), ele terá posicionamento fixo e
        % estará posicionado na posição 0 em x e 0.5 em y
        ponto_x(1) = 0;
        ponto_y(1) = 0.5;

        % ponto_z é o vetor de coordenadas z dos sensores no plano, a equação
        % das coordenadas é a mesma
        % ********** A fórmula deve ser alterada
        ponto_z = ponto_x.* exp(-ponto_x.^2 - ponto_y.^2);

        % inicializa a matriz de posições pos
        pos = zeros(rede_simulacao(iter_rede_simulacao),3);

        % O for é para estabelecer uma matriz de 2 colunas e quant linhas, ou seja,
        % uma matriz que possui a posição dos quant nós, a primeira coluna possui
        % a coordenada x(horizontal) de cada nó e a segunda coluna possui a coordenada
        % y(vertical).
        for sensor=1:rede_simulacao(iter_rede_simulacao)
            pos(sensor,1) = ponto_x(sensor);
            pos(sensor,2) = ponto_y(sensor);
            pos(sensor,3) = ponto_z(sensor);
        end

        % A função fprintf escreve no arquivo o que queremos, o primeiro
        % parâmetro possui a variável associada ao arquivo que iremos
        % utilizar, o segundo parâmetro é o tipo associado ao que
        % iremos escrever no arquivo. E o terceiro é o que iremos
        % escrever.
        fprintf(fid, '%s', 'Matriz de posicionamento dos sensores:');
        fprintf(fid,'\r\n');

        % Escreve no arquivo a matriz de posicionamento dos nós
        for sensor = 1:rede_simulacao(iter_rede_simulacao)
            fprintf(fid,'%d) x = %5.4f || y = %5.4f || z = %5.4f\r\n',sensor,
pos(sensor,1),pos(sensor,2),pos(sensor,3));
        end

        %% Atribuição dos valores das distâncias entre os nós
```

```
% Curvas é o conjunto de vetores que incluem o intervalo de
% valores traçados entre dois nós, o tamanho da variável
% Curvas
Curvas = cell(rede_simulacao(iter_rede_simulacao), rede_simulacao(iter_rede_simulacao));

% Inicialização da matriz de distâncias
dis = zeros(rede_simulacao(iter_rede_simulacao));

% analisa os sensores, traçando curvas que acompanham as curvas do
% relevo, somente se as mesmas tiverem comprimento inferior ao valor de
% R
for analisado=1:1:rede_simulacao(iter_rede_simulacao)

   for comparado=analisado+1:1:rede_simulacao(iter_rede_simulacao)

      % Chama a função distancia, com as coordenadas dos
      % pontos analisados como parâmetro de entrada. A função
      % retorna os vetores x, y e z, que designam o
      % intervalo de pontos traçados entre os dois
      % sensores e formarão a curva traçada entre eles.
      [Curvas{analisado, comparado, 1}, Curvas{analisado, comparado, 2},
Curvas{analisado, comparado, 3},dis(analisado, comparado)] =
distancia(ponto_x(analisado),ponto_y(analisado), ponto_z(analisado),
ponto_x(comparado),ponto_y(comparado), ponto_z(comparado));

      % a distância entre o sensor analisado e o
      % comparado é a mesma do sensor comparado para o
      % analisado, já que se trata dos mesmos nós de
      % análise
      dis(comparado, analisado) = dis(analisado, comparado);
      Curvas{comparado, analisado, 1} = Curvas{analisado, comparado, 1};
      Curvas{comparado, analisado, 2} = Curvas{analisado, comparado, 2};
      Curvas{comparado, analisado, 3} = Curvas{analisado, comparado, 3};

   end

end

% Escreve no arquivo a matriz de distâncias entre os
% sensores
fprintf(fid,'\r\n');
fprintf(fid,'%s','Matriz de distâncias entre os sensores :');
fprintf(fid,'\r\n');

for sensor_1 = 1:rede_simulacao(iter_rede_simulacao)
   fprintf(fid,'%d) ',sensor_1);
   for sensor_2 = 1:rede_simulacao(iter_rede_simulacao)
      fprintf(fid,' %5.4f ',dis(sensor_1, sensor_2));
   end
   fprintf(fid,'\r\n');
end

%% Atribuição dos valores de conectividade entre os nós

% Inicialização da matriz MC
MC = zeros(rede_simulacao(iter_rede_simulacao), rede_simulacao(iter_rede_simulacao));

% O primeiro for percorre as linhas e o segundo for percorre a
```

```
% as colunas da matriz conectividade. Lembrando que a conectividade
% é estabelecida através de uma constante, esta constante assume valores
% entre zero e um. Se a distância entre dois nós for menor que a constante
% então os nós possuem uma conectividade, caso contrário, não possuem.
for sensor_1 = 1:rede_simulacao(iter_rede_simulacao)
   for sensor_2 = sensor_1:rede_simulacao(iter_rede_simulacao)
      if(dis(sensor_1, sensor_2) < R(iter_R))
         MC(sensor_1, sensor_2) = 1;
         MC(sensor_2, sensor_1) = MC(sensor_1, sensor_2);
      end
   end
end

% Escreve no arquivo a matriz de conectividade entre os nós
fprintf(fid,'\r\n');
fprintf(fid,'%s', 'Matriz de conectividade entre os sensores :');
fprintf(fid,'\r\n');

for sensor_1 = 1:rede_simulacao(iter_rede_simulacao)
   fprintf(fid,'%d) ',sensor_1);
   for sensor_2 = 1:rede_simulacao(iter_rede_simulacao)
      fprintf(fid,'%d ',MC(sensor_1, sensor_2));
   end
   fprintf(fid,'\r\n');
end

%% Busca em Largura
% Inicialização das variáveis
proximo = 1;
ler = 1;
VR = zeros(1,rede_simulacao(iter_rede_simulacao));
VS = zeros(1,rede_simulacao(iter_rede_simulacao));
VS(ler) = ler;

% quantidade de nós conectados
conectado = 1;

% Se a variável ler for maior que a quantidade de nós então ocorreu a
% leitura de todos os nós, ou se VS(ler) for igual zero, quer dizer que
% não foi adicionado nenhum novo elemento no vetor, logo todos os
% elementos que possuem ligação com o nó raiz já foram inseridos
while(ler <= rede_simulacao(iter_rede_simulacao) && VS(ler) ~= 0)

   % Entra se no VS conter algum nó
   if (VS(ler) ~= 0)

      % Este for é para percorrer toda as colunas (nós)
      % da matriz de conectividade, para saber com qual
      % nó cada nó possui ligação.
      for j = 1:rede_simulacao(iter_rede_simulacao)

         % Se MC(VS(ler),j) for igual a um sigfica que o nó que
         % está sendo lido possui ligação com o nó j, logo ele deve
         % ser armazenado no VS, caso ele ainda não foi.

         if (MC(VS(ler),j) == 1)

            % pressupõe-se que o elemento não está contido no vetor
            % através da variável ok recebendo um
            ok = 1;
```

```
            for i = 1:proximo
                % Caso ele esteja contido no vetor é armazenado
                % valor zero na variável ok
                if (VS(i) == j)
                    ok = 0;
                    break;
                end
            end

            % se ok for igual um significa que o elemento não está
            % contido no vetor, ele é então adicionado
            if (ok == 1)
                proximo = proximo + 1;
                VS(proximo) = j;

                % incrementa-se a variável conectado
                conectado = conectado + 1;

                % Quando for adicionado guarda-se no vetor de
                % roteamento o seu primeiro antecessor.
                if (VR(j) == 0)
                    VR(j) = VS(ler);
                end
            end
        end
    end
end

  ler = ler + 1;

end

% Escreve no arquivo os vetores conectados
fprintf(fid,'\r\n');
fprintf(fid,'%s', 'Vetor de sensores conectados :');
fprintf(fid,'\r\n');

for sensor = 1:conectado
   fprintf(fid,'%d ',VS(sensor));
end

fprintf(fid,'\r\n');

% Escreve no arquivo o vetor de roteamento
fprintf(fid,'\r\n');
fprintf(fid,'%s', 'Vetor de roteamento entre os sensores :');
fprintf(fid,'\r\n');

for sensor = 1:rede_simulacao(iter_rede_simulacao)
   fprintf(fid,'%d ',VR(sensor));
end

fprintf(fid,'\r\n');

% função que permite a disposição de mais de um gráfico na tela de
% figuras do matlab
%hold off;

%% Geração de Pacotes
```

```
% caso tenha apenas um nó conectado na rede, seria
% apenas o nó sink
if (conectado ~= 1)

    % Quantidade de blocos de simulação
    quant_bloco_simulacao = 30;

    % quantidade de ciclos de simulação
    quant_ciclo_simulacao = 10000;

    % Inicialização da variável LqA
    % LqA é o vetor que contém a quantidade máxima de elementos
    % encontrados na Fila durante cada bloco de simulação
    LqA = cell(1,conectado);

    for bloco_simulacao = 1:quant_bloco_simulacao

        % Inicializa-se a variável de pacotes gerados com o valor 0
        % para cada produtor
        gerado(1:rede_simulacao(iter_rede_simulacao)) = 0;

        % Inicialização do vetor de comprimento da fila em uma determinada
        % iteração
        Lq = cell(1, rede_simulacao(iter_rede_simulacao));

        % Inicialização da fila de pacotes
        Fila = cell(1, rede_simulacao(iter_rede_simulacao));

        % valor de lambida para uso na distribuição de
        % Poisson
        lambida = 1;

        % pacote é definido com 1 como valor simbólico
        pacote = 1;

        % produtor é o nó que gera pacotes em um determinado
        % momento, ele poderá ser qualquer nó conectado direta
        % ou indiretamente ao nó sink
        produtor = conectado;

        for ciclo_simulacao = 1:quant_ciclo_simulacao

            % Insere na Fila  poissrnd(lambida) pacotes
            % poissrnd(lambida) é uma função que dado uma
            % distribuição de Poisson com lambida como
            % parâmetro retorna um valor a partir de zero com
            % base nas probabilidades da distribuição

            quant_pacote = poissrnd(lambida);

            for i = 1:quant_pacote

                % Insere tal pacote no fim da Fila
                Fila{1, VS(produtor)}(length(Fila{1, VS(produtor)})+1) = pacote;

            end
```

```
                % Caso o tamanho da Fila do produtor analisado seja maior do que zero, existe
                % pacotes na Fila, logo o pacote na primeira posição é
                % processado
                if (length(Fila{1,VS(produtor)}) > 0)

                    % o nó antecessor recebe processando um pacote da Fila do produtor
                    % analisado
                    Fila{1, VR(VS(produtor))}(length(Fila{1, VR(VS(produtor))})+1) = Fila{1,
VS(produtor)}(1);

                    % Retira o primeiro pacote da Fila do produtor
                    % analisado
                    Fila{1,VS(produtor)} = Fila{1,VS(produtor)}(2:length(Fila{1,VS(produtor)}));

                end

                % analisa o tamanho da fila do produtor ao final
                % do ciclo de simulação
                Lq{1, VS(produtor)}(ciclo_simulacao) = length(Fila{1,VS(produtor)});

                %% Tratamento para o nó sink

                if produtor == 2

                    produtor = 1;

                    % se a fila do nó sink for maior do que zero ele
                    % deve processar algum pacote
                    if (length(Fila{1,VS(produtor)}) > 0)

                        % Retira o primeiro pacote da Fila do produtor
                        % analisado
                        Fila{1,VS(produtor)} = Fila{1,VS(produtor)}(2:length(Fila{1,VS(produtor)}));

                        % analisa o tamanho da fila do produtor
                        % ao final
                        % dos ciclos de simulação
                        Lq{1, VS(produtor)}(ciclo_simulacao) = length(Fila{1,VS(produtor)});

                    end

                    produtor = conectado;

                else
                    produtor = produtor - 1;
                end


            end

            % A posição bloco_simulação do vetor LqA e NC de todos os produtores corresponde
ao
            % bloco_simulação analisado
            % A posição em LqA analisado recebe a quantidade máxima de pacotes encontrados
            % no vetor Lq
            % A posição em NC analisado recebe a quantidade passos de
            % simulação do bloco de simulação analisado
            for i = 1:conectado
                LqA{1,VS(i)}(bloco_simulacao) = max(Lq{1,VS(i)});
```

```
                end

            end

            for j = 1:conectado

                % Media dos valores encontrados em LqA
                media = mean(LqA{1,VS(j)});

                % Desvio-Padrão dos Valores encontrados em LqA
                desvio_padrao = std(LqA{1,VS(j)});

                % Valor Absoluto dos valores encontrados em LqA
                valor_absoluto = max(LqA{1,VS(j)});

                % Escreve no arquivo o produtor analisado
                fprintf(fid,'\r\n\r\n');
                fprintf(fid,'%s %d','Nó : ',VS(j));

                % Escreve no arquivo a media de LqA
                fprintf(fid,'\r\n');
                fprintf(fid,'%s %5.2f',' Media LqA : ',media);

                % Escreve no arquivo o desvio padrão
                fprintf(fid,'\r\n');
                fprintf(fid,'%s %5.2f',' Desvio-Padrao LqA : ',desvio_padrao);

                % Escreve no arquivo o valor máximo de pacotes na fila em uma
                % simulacao ao fim das iteracoes
                fprintf(fid,'\r\n');
                fprintf(fid,'%s %5.2f',' Valor Absoluto LqA : ',valor_absoluto);

            end

          end

        end

        % pula duas linhas no arquivo
        fprintf(fid,'\r\n\r\n');

      end

    end

    % fecha o arquivo que utiliza o id fid
    fclose(fid);

end
```

APÊNDICE B

```
function [x,y,z,dis] = distancia(x1,y1,z1,x2,y2,z2)
% Nome: Luiz Fernando Ferreira Gomes de Assis RA: 823261
% dis é a variável que retorna o valor da distância entre os dois pontos
% x1, y1 e z1 são as coordenadas do ponto 1
% x2, y2 e z2 são as coordenadas do ponto 2

   % divide a distância entre os dois pontos em numero_partes vezes
   numero_partes = 10;

   % intervalo em x responsável pela determinação dos pontos em x
   intervalox = abs(x2 - x1)/numero_partes;

   % determinação dos pontos em x, iniciando em x1, terminando em x2,
   % incrementado por intervalox entre dois pontos
   if x1 < x2
      x = x1:intervalox:x2;
   else
      x = x1:-intervalox:x2;
   end

   % intervalo em y responsável pela determinação dos pontos em y
   intervaloy = abs(y2 - y1)/numero_partes;

   % efetua a alocação dos valores dependendo dos valores das coordenadas
   % y
   % determinação dos pontos em y, iniciando em y1, terminando em y2,
   % incrementado por intervalox entre dois pontos
   if y1 < y2
      y = y1:intervaloy:y2;
   else
      y = y1:-intervaloy:y2;
   end

   % intervalo em z responsável pela determinação dos pontos em z
   intervaloz = abs(z2 - z1)/numero_partes;

   % efetua a alocação dos valores dependendo dos valores das coordenadas
   % z
   % determinação dos pontos em , iniciando em x1, terminando em x2,
   % incrementado por intervalox entre dois pontos
   if z1 < z2
      z = z1:intervaloz:z2;
   else
      z = z1:-intervaloz:z2;
   end

   % função responsável
   plano_z = x.* exp(-x.^2 - y.^2);

   % atribui 0 a variável dis, valor inicial da distância entre os nós
   dis = 0;

   % efetua o cálculo da distância de acordo com a quantidade de partes
   % definida
   for iter = 1:1:numero_partes - 1

      % distancia entre dois pontos no plano tridimensional
```

```
        if plano_z(iter) >= z(iter)
            dis = dis + sqrt( (x(iter)-x(iter+1)).^2 + (y(iter)-y(iter+1)).^2 + (plano_z(iter)-plano_z(iter+1)).^2 );
            z(iter) = plano_z(iter);
        else
            dis = dis + sqrt( (x(iter)-x(iter+1)).^2 + (y(iter)-y(iter+1)).^2 + (z(iter)-z(iter+1)).^2 );
        end

    end

end
```