UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
DEEC

Cama de Testes Para Módulos e Componentes Electrónicos

(*Relatório de Estágio* **PRODEP**)

Manuel Fernando dos Santos Silva

Porto, 05 de Novembro de 1993

Original no Relatório de anexo 1.38

Porto, Dezembro de 1993

PRODEP

MEDIDA 4.3 - Estágios Profissionais para Bacharelatos, Licenciaturas e Pós-Graduação

PARECER

Assunto: Estágios de
(1.38) Alberto Xavier Pires Borges da Cunha
(1.42) José Carlos Lobinho Gomes
(1.44) Manuel Fernando dos Santos Silva
(1.48) Pedro Manuel Gonçalves Campelos de Sousa Lobo.

Na qualidade de supervisores dos estágios acima mencionados e em face do trabalho desenvolvido e relatado pelos estagiários pode concluir-se que:

1- Os estagiários manifestaram entusiasmo e dedicação na execução do trabalho intitulado:

" Projecto de uma cama de testes para componentes e módulos electrónicos incluindo a concepção da interface (entradas e saídas) e do sistema operativo".

O trabalho consistiu no projecto e teste em montagem experimental de vários módulos cujas responsabilidades foram distribuidas:

Software para interface com o utilizador em PC - Manuel Fernando dos Santos Silva; "device driver" para interface software/hardware - José Carlos Lobinho Gomes; hardware de teste com microcontrolador, conversão e software operativo - Manuel Fernando dos Santos Silva e Pedro Manuel Gonçalves Campelos de Sousa Lobo.

Nesta execução revelaram apreciáveis qualidades de trabalho que mais desenvolveram. Do trabalho resultou um protótipo do hardware em "breadboard" e protótipos dos módulos de software.

2- No cumprimento dos objectivos propostos, os estagiários revelaram bons conhecimentos científicos e técnicos, capacidade de aplicação desses conhecimentos e espírito de iniciativa. A metodologia seguida consistiu no estudo da proposta base, desenvolvimento por análise das especificações do sistema objecto, projecto de soluções tecnicamente adequadas eventualmente inovadoras, implementação de protótipos e validação do projecto.

pelo que somos do parecer que foram cumpridos os objectivos que o estágio se propunha atingir.

Prof. Doutor António Maria da Rocha Quintas
Centro de CIM do Porto

Prof. Doutor Diamantino Rui da Silva Freitas
Faculdade de Engenharia da Universidade do Porto

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores

CAMA DE TESTES PARA MÓDULOS E COMPONENTES ELECTRÓNICOS

Manuel Fernando dos Santos Silva

Porto, Outubro de 1993

INTRODUÇÃO

INTRODUÇÃO

O relatório agora apresentado, diz respeito ao estágio realizado entre Abril e Outubro de 1993, pelos alunos Manuel Fernando dos Santos Silva, José Carlos Lobinho Gomes, Alberto Xavier e Pedro Manuel Lobo, da Faculdade de Engenharia da Universidade do Porto, no âmbito do PRODEP e sob a supervisão do Prof. Dr. Diamantino Rui da Silva Freitas.

O estágio realizado tinha por tema "Cama de testes para módulos e componentes electrónicos", tendo sido realizado na Faculdade de Engenharia da Universidade do Porto e tendo a Universidade do Porto como parceiro, neste estágio, o Centro de C.I.M. do Porto.

Este relatório é respeitante ao trabalho desenvolvido pelo estagiário Manuel Fernando dos Santos Silva, sendo o trabalho desenvolvido pelos restantes estagiários apresentado em relatórios separados, entregues pelos próprios.

OBJECTIVOS

Pretendeu-se com a realização deste projecto conceber e desenvolver um sistema de teste, quer para componentes electrónicos discretos, quer mesmo para placas de circuito impresso ou para qualquer outro tipo de circuitos electrónicos.

Como facilmente compreenderemos o teste de qualquer circuito electrónico é fundamental, não só para verificar o seu correcto funcionamento, mas também para identificar eventuais defeitos de fabrico ou avarias e possibilitar diagnósticos nas situações de avaria.

Durante a fase de concepção deste sistema de teste procuramos desenvolver um sistema o mais genérico possível, que permitisse uma grande flexibilidade ao nível dos testes possíveis de efectuar a qualquer módulo ou componente electrónico. De forma a responder a estes objectivos optamos por desenvolver um sistema completamente novo de raiz.

IMPLEMENTAÇÃO

SOLUÇÃO ADOPTADA

O sistema, por nós concebido, compreende fundamentalmente três módulos:

1° - um módulo de interface com o utilizador, de forma a que este possa escolher os testes que pretende realizar e o local (do circuito ou componente electrónico) em que os pretende realizar;

2° - um módulo de interface entre o software e o hardware;

3° - por último, um módulo de hardware, ou seja, um sistema encarregado de executar as ordens fornecidas pelo utilizador.

O primeiro módulo consta de um pacote de software, constituído por um programa principal e várias rotinas, que fazem, como já foi dito, a interface entre o utilizador e o sistema de teste propriamente dito. Todo o software foi desenvolvido em ambiente MATLAB, versão 3.13, para PC. Se bem que o programa corra perfeitamente bem num PC com μP 386 é aconselhável que este disponha de coprocessador matemático, para aumentar a velocidade dos cálculos necessários, nomeadamente no que diz respeito a cálculos de ondas e cálculos de funções de correlação e convolução e de fft's.

O software desenvolvido permite ao utilizador gerar uma das seguintes formas de onda:

- onda sinusoidal;
- onda quadrada;
- onda triangular;
- onda dente de serra;
- seno cardinal;
- impulso rectangular;
- impulso gaussiano;
- componente continua;
- degrau de heaviside;
- onda exponencial,

e usando as operações matemáticas mais elementares tais como soma, subtracção e multiplicação, gerar qualquer tipo de onda a partir da execução destas operações sobre duas ou mais ondas elementares; permite ainda operar quer sobre as ondas elementares, quer sobre as mais complexas, não só através das operações matemáticas atrás referidas mas também através da adição de componente contínua às ondas e a possibilidade de aumentar a amplitude das ondas, amplitude essa que é por defeito definida como sendo unitária.

O segundo módulo consta de um "Device Driver" que faz a interface entre o software a que o utilizador tem acesso e o hardware. O "Device Driver" transforma o hardware, do ponto de vista do software, num ficheiro. Isto permite aceder ao hardware através das funções de trabalho com ficheiros. Desta forma o envio de dados para o exterior (para o hardware) é visto como a escrita num ficheiro com o nome *adc* e a recepção de dados é realizada efectuando a leitura do mesmo ficheiro.

O terceiro módulo consta de uma placa com hardware dedicado, desenvolvida especificamente para este projecto, com uma ligação RS-232 à porta série do PC, sendo a placa constituida por um μC com a função de supervisionar todas as operações envolvidas no hardware e efectuar a comunicação com o PC, conversores A/D e D/A e ainda multiplexer's com funções de linhas de I/O.

As linhas de I/O permitem identificar as agulhas da cama de testes onde se vão aplicar as ondas e de onde vão ser recolhidas as ondas de saída. Os conversores A/D e D/A permitem, respectivamente, que os sinais recolhidas nas agulhas sejam convertidas para forma digital e enviadas através da porta série para o PC para posterior visualização ou tratamento e que as ondas geradas pelo utilizador, que estão guardadas na forma digital sejam enviadas para o exterior, sendo convertidas para forma analógica após o que são aplicadas nas agulhas pretendidas.

É de referir que desta forma evitam-se problemas de temporização que surgiam no caso de trabalharmos com placas de I/O e placas de conversão A/D e D/A ligadas ao circuito exterior através do barramento interno do PC.

SOFTWARE

Vou aqui, neste ponto, procurar descrever de forma o mais completa possível o software desenvolvido ao longo deste estágio, para o módulo de interface com o utilizador, software esse, que como já foi referido atrás, foi desenvolvido em ambiente MATLAB, versão 3.13. Como forma de complementar esta descrição, as listagens de todas as rotinas são apresentadas em anexo.

Não farei aqui neste relatório nenhuma descrição mais aprofundada do "device driver" e do hardware desenvolvido neste estágio, uma vez que a descrição desses módulos é apresentada nos relatórios dos colegas que comigo estagiaram e a quem coube a responsabilidade do desenvolvimento desses módulos.

O software desenvolvido consta de várias rotinas que se podem dividir em vários grupos; existe a rotina principal, que é a que dá o nome ao programa, chamada *CAMATEST.M*, existem duas rotinas que apresentam menus, uma chamada *MENUOPER.M* e que apresenta o menu de operações que o utilizador pode realizar neste programa e outra chamada *MENUONDA.M* e que, tal como o seu nome indica, apresenta ao utilizador um menu com as várias ondas que é possível gerar dentro deste programa, existem rotinas para envio, recepção e envio e recepção simultâneo de sinais, respectivamente designadas por *ENVIAOND.M*, *RECEBEON.M* e *ENVRECON.M*, existe uma rotina de ajuda, que funciona como um manual do software, designada *ROTAJUDA.M*, existe uma rotina de visualização destinada a permitir visualizar qualquer tipo de sinais com que este programa trabalha, bem como o resultado de qualquer das operações possíveis sobre esses sinais chamada *VISUAL.M*, existem rotinas inerentes a certas necessidades características do software tais como a rotina *LIMPEZA.M* que permite limpar, ou melhor dizendo, inicializar as variáveis *posvec* e *vector*, a rotina *DESISTIR.M* que permite implementar a opção de abandono do programa sem que o utilizador, para abandonar o programa, tenha que o abortar e existe ainda a rotina *ESCTEMPO.M* que permite ao utilizador definir um intervalo de tempo para gerar, tratar e analisar os sinais, diferente do intervalo de tempo definido por defeito e que varia entre -1 e 1 seg. Existem, além destas, rotinas que permitem gerar os vários sinais com que vamos trabalhar e realizar operações quer entre os sinais, quer sobre esses mesmos sinais. As rotinas para gerar sinais são s rotinas *GERASENO.M* (para gerar ondas sinusoidais), *GERAQUAD.M* (para gerar ondas quadradas), *GERATRIA.M*

(para gerar ondas triângulares), *GERADSER.M* (para gerar ondas em dente de serra), *GERASENC.M* (para gerar ondas em seno cardinal), *GERAIREC.M* (para gerar impulsos rectângulares), *GERAIGAU* (para gerar impulsos Gaussianos), *GERACONT* (para gerar ondas contínuas), *GERAHEAV* (para gerar o degrau de Heaviside) e *GERAEXPO* (para gerar ondas exponênciais) e as rotinas que permitem realizar operações são as rotinas *SOMAR.M*, *SUBTRAIR.M*, *MULTIPLI.M*, *COMPODC.M*, *MAISAMPL.M*, *CONVOLUC.M* e *CORRELAC.M* para realizar operações entre sinais e as rotina *TFOURIER.M* e *ESPECTRO.M*, que calculam, respectivamente, a transformada de Fourier de um sinal e o seu espectro de densidade de potência, que realizam operações sobre um único sinal.

Um possível ciclo de trabalho do utilizador com este pacote de software, será o utilizador entrar no software, escolher um determinado intervalo de tempo para trabalhar (o intervalo de tempo é definido através das variáveis *tempoini* e *tempofim* que guardam, respectivamente, os valores inicial e final do intervalo de tempo e estas variáveis são guardadas no ficheiro *inttempo.mat* para que se possa limpar a memória do MATLAB periodicamente sem haver perda de informação e para se poderem passar estes valores entre as várias rotinas). Seguidamente o utilizador pode gerar uma ou mais ondas, que vão ser guardadas na variável *vector*, variável essa que é uma matriz com seis linhas e mil colunas. Em cada linha da matriz podemos guardar um sinal e a linha da matriz onde vamos guardar um sinal ou onde o vamos buscar é apontada através da variável *posvec*. Neste ficheiro é ainda guardada a variável *tempo*, que é um vector com uma linha e mil colunas e que representa a evolução temporal entre o tempo inicial e o tempo final do intervalo de tempo que estamos a considerar. Depois de geradas várias ondas, o utilizador pode realizar operações entre elas para obter sinais mais complexos, ou pode optar por enviar estas ondas elementares directamente para o hardware através das rotinas *ENVIAOND.M* ou *ENVRECON.M*. Depois de ter enviado os sinais para os aplicar nas agulhas desejadas o utilizador pode receber os sinais de resposta às entradas aplicadas, através da chamada às rotinas *RECEBEON.M* ou *ENVRECON.M*. Depois de receber os sinais vindos do exterior (do módulo ou componente electrónico em teste), o utilizador pode novamente operar sobre estes sinais, por exemplo, calculando o seu espectro ou a sua transformada de Fourier, e após isto visualizar os resultados obtidos através da rotina *VISUAL.M*.

Depois de executado este ciclo, o utilizador pode fazer a limpeza das variáveis *vector* e *posvec* usando a rotina *LIMPEZA.M* e fazer um novo ciclo de teste ao circuito ou componente electrónico que está a ser

testado ou a outro componente ou circuito; o utilizador pode ainda optar por abandonar o programa, caso em que será chamada a rotina *DESISTIR.M*.

Quanto à troca de dados com o exterior, através do envio de sinais para o circuito de teste e da recepção dos sinais de resposta às entradas de teste, os dados são enviados num formato que torna possível a compreensão mútua entre hardware e software. Num envio de dados, o formato de dados é o que a seguir se apresenta:

N° de envios : N° agulha ; T ; 1000 amostras separadas por espaço ;

N° agulha ; T ; 1000 amostras separadas por espaço ; ..... ;

Numa recepção de dados o formato de dados é essencialmente o mesmo do envio, tal como a seguir se mostra, com pequenas diferenças inerentes às características que diferenciam o envio e a recepção de dados:

N° de recepções : N° agulha ; T ; 1000 amostras separadas por espaço ;

N° agulha ; T ; 1000 amostras separadas por espaço ; ..... ;

e o formato da mensagem para o pedido de envio de dados é a que se segue:

N° de recepções : N° agulha ; N° agulha ; ..... N° agulha ;

ROTINAS DO SOFTWARE

A rotina *CAMATEST.M*

Esta rotina é a rotina principal de todo o software desenvolvido. É esta a rotina que é invocada quando se pretende correr o software. Assim, ao correrrmos o software entramos logo nesta rotina e logo no início começa por ser feita uma apresentação e identificação do trabalho desenvolvido no estágio e dos elementos que trabalharam neste grupo de estágio.

Após esta breve apresentação é feita uma inicialização de algumas variáveis que vão ser necessárias no decurso das operações de computação, e algumas dessas variáveis vão ser gravadas num ficheiro que funciona como buffer, para que estas variáveis possam ser passadas de umas rotinas para as outras sem problemas de serem apagadas ou alteradas.

Depois desta inicialização das variáveis, o programa prossegue com a chamada à rotina *rotajuda.m*, que é uma rotina que contém algumas informações acerca do software desenvolvido e finalmente entra-se na rotina *menuoper.m*, que é a rotina que contém o menu principal do software e a partir do qual podemos decidir qual a operação a realizar, dentro das possibilidades do software.

Na saída desta rotina é feita a limpeza do ecrã e de todas as variáveis que foram abertas, para o caso de se pretender continuar a trabalhar em ambiente MATLAB, mas em outras aplicações.

A rotina *ROTAJUDA.M*

Esta rotina é essencialmente uma rotina de texto, e na qual é dada a possibilidade ao utilizador de ficar a saber algo mais sobre o funcionamento do pacote de software; no fundo, esta rotina pode ser vista como um pequeno manual do funcionamento de todo o software.

Ao utilizador é dada a hipótese de receber esta breve explicação do funcionamento do programa, e caso ele a aceite, pode rever a explicação tantas vezes quantas o desejar.

A rotina *MENUOPER.M*

A rotina *MENUOPER.M* é a rotina que é responsável pela apresentação do menu principal do software e a partir da qual se pode decidir qual a operação que se pretende realizar. O utilizador tem à sua disposição dezasseis alternativas diferentes.

Quando esta rotina é corrida, são apresentadas no ecrã as dezasseis opções que o utilizador pode tomar e é pedido ao utilizador que entre o número da opção que pretende. As dezasseis opções disponíveis são as seguintes:

0 - Gerar sinal	*menuonda.m*
1 - Enviar sinal	*enviaond.m*
2 - Receber sinal	*recebeon.m*
3 - Enviar e receber um sinal	*envrecon.m*
4 - Visualizar um sinal	*visual.m*
5 - Somar sinais	*somar.m*
6 - Subtrair sinais	*subtrair.m*
7 - Multiplicar sinais	*multipli.m*
8 - Introduzir componente contínua	*compodc.m*
9 - Aumentar a amplitude do sinal	*maisampl.m*
10 - Convolução de sinais	*convoluc.m*
11 - Correlação de sinais	*correlac.m*
12 - FFT de sinais	*tfourier.m*
13 - Espectro de densidade de potência	*espectro.m*
14 - Alterar intervalo de visualização	*esctempo.m*
15 - Fazer o reset dos sinais	*limpeza.m*
16 - Abandonar o programa	*desistir.m*

em frente das opções possíveis, apresenta-se o nome da rotina que é invocada, quando essa é a opção escolhida. Depois de o operador ter escolhido uma determinada opção, salta para a rotina correspondente e quando essa rotina termina, o programa volta para esta rotina e pede ao utilizador que entre o caracter s ou n, conforme pretenda realizar outra operação das indicadas neste menu ou não. Caso o utilizador pretenda realizar outra operação, entra o caracter s e o menu com todas as operações possíveis volta a ser apresentado; caso o utilizador pretenda terminar a execução do programa, entra o caracter n e é chamada a rotina *DESISTIR.M*, com vista a realizar a saída do programa.

A rotina *MENUONDA.M*

Quando, no menu principal, o utilizador escolhe a opção 0, correspondente a gerar um sinal, o programa salta para esta rotina. O seu funcionamento e estrutura são bastante semelhantes ao da rotina *MENUOPER.M*, pois quando esta rotina é chamada também é apresentado ao utilizador um menu, mas agora este menu contém todos os tipos de ondas que se podem gerar neste programa.

As ondas que se podem gerar neste programa são as mais utilizadas e mais genéricas, pois consideramos que com estas ondas e com as operações que se podem realizar sobre elas, podemos gerar a quase totalidade das ondas de interesse para fazer o teste de circuitos ou componentes electrónicos.

As opções que o utilizador pode escolher, relativamente às ondas que se podem gerar, são as indicadas a seguir e, tal como anteriormente, apresentam-se à frente das opções as rotinas para onde o programa salta quando se opta por gerar esse tipo de onda:

1 - Onda sinusoidal	*geraseno.m*
2 - Onda quadrada	*geraquad.m*
3 - Onda triângular	*geratria.m*
4 - Onda dente de serra	*geradser.m*
5 - Seno cardinal	*gerasenc.m*
6 - Impulso rectângular	*gerairec.m*
7 - Impulso Gaussiano	*geraigau.m*
8 - Componente contínua	*geracont.m*
9 - Degrau de Heaviside	*geraheav.m*
10 - Onda exponêncial	*geraexpo.m*

Antes de o sinal ser gerado é pedido ao utilizador que indique em que vector da matriz que contém os sinais, quer guardar a onda que vai gerar e, após o sinal ter sido gerado, é pedido ao utilizador que opte por realizar a geração de outra onda ou por regressar ao menu principal, onde são indicadas as operações que se podem realizar sobre os sinais aqui gerados. Para gerar outra onda, como é óbvio, o utilizador deve responder com o caracter s à pergunta se "Pretende gerar outra onda?" e com o caracter n se pretende regressar ao menu principal.

As rotinas *GERA????.M*

Todas as rotinas designadas por *GERA*, seguidas de quatro caracteres referentes ao nome de um tipo de sinal, destinam-se a gerar esse tipo de onda.

A estrutura de todas estas rotinas é semelhante; no início da rotina é feita uma limpeza do ecrã e depois são carregados os ficheiros *buffer.mat* e *inttempo.mat*. O primeiro contém uma matriz de seis linhas por mil colunas na qual são guardados, em cada linha, um sinal especificado pelo utilizador e o segundo contém, por sua vez, duas variáveis, *tempoini* e *tempofim*, onde são guardados, respectivamente o início e o fim do intervalo de tempo para o qual vamos gerar o sinal pretendido.

Depois de terem sido carregados estes ficheiros, as rotinas pedem ao utilizador que entre as características respeitantes a cada tipo de onda, tais como a sua frequência, fase inicial, "duty cycle" (no caso da onda quadrada), constante da exponencial (no caso da onda exponencial), etc.

Chegados aqui, é gerada a onda. Enquanto são realizados os cálculos, é apresentada no ecrã uma mensagem de espera e quando os cálculos terminam pede-se ao utilizador que carregue numa tecla para prosseguir a execução do programa.

O cálculo do sinal é realizado, dividindo o intervalo de tempo para o qual se vai gerar a onda em mil instantes de tempo, igualmente espaçados, e calculando, para cada um desses instantes, o valor que o sinal assume. Esses mil valores são depois guardados nas mil posições do vector em que vamos guardar o sinal.

Depois de terminado o cálculo do sinal e antes de se retornar ao menu de geração das ondas, é gravada a variável *vector* que contém os sinais (e também o sinal agora gerado) no ficheiro *buffer.mat*.

A rotina *ENVIAOND.M*

Esta rotina é a responsável pelo envio dos sinais gerados para o exterior, ou seja para o hardware, através do "device driver".

Na realidade, o que esta rotina faz, é simular a escrita dos dados para um ficheiro com o nome *adc*, que na realidade é o "device driver", que depois se vai encarregar de encaminhar os dados para o hardware.

Esta rotina começa com a limpeza do ecrã e logo de seguida carregam-se os ficheiros *buffer.mat* e *inttempo.mat* que contêm, respectivamente, as variáveis *vector* e *tempoini* e *tempofim*. Depois de carregadas estas variáveis, calcula-se o período de cálculo das ondas, ou

seja o período com que calculamos o valor das ondas. Depois disto é perguntado ao utilizador quantas ondas pretende enviar e esse valor é enviado para o "device driver". Feito isto é perguntado ao utilizador em que vectores estão as ondas que ele pretende enviar e para que agulhas as quer enviar. Depois de o utilizador ter introduzido todos os dados necessários, começa o envio, propriamente dito, da informação e durante o decurso do envio da informação é apresentada no ecrã uma mensagem de espera.

O formato de envio de dados foi apresentado acima.

A rotina *RECEBEON.M*

A rotina *RECEBEON.M* tem uma estrutura complementar da rotina *ENVIAOND.M*.

Nesta rotina começa-se por fazer a limpeza do ecrã e por se carregar os ficheiros *buffer.mat* e *inttempo.mat*, após o que é pedido ao utilizador que entre o número de aquisições que pretende efectuar, qual o número das agulhas de que pretende fazer a aquisição e em que vectores quer guardar os dados recebidos. Feito isto, é enviado para o "device driver" o número de aquisições que se querem fazer e o número das agulhas de onde se pretende fazer a aquisição, após o que a rotina entra num ciclo de espera, findo o qual vai ver se os dados já foram adquiridos. Quando os dados acabam de ser adquiridos, a variável binária *finalaq* vem a 1 e a rotina vai então ler o ficheiro *adc* onde o "device driver" guardou os resultados e guardar os resultados dos sinais adquiridos nos vectores correspondentes.

Os formatos de dados de entrada e do pedido de aquisição de dados já foram apresentados atrás.

A rotina *ENVRECON.M*

Esta rotina vai ser referida apenas de forma breve, uma vez que ela pode ser considerada como a junção das duas anteriores, apresentando exactamente a mesma estrutura no que diz respeito ao envio e à recepção dos dados. A única diferença que esta rotina apresenta em relação às anteriores é que permite o envio e a recepção simultânea das ondas, pelo que desta forma se pode ver como o circuito a ser testado responde aos sinais que lhe foram aplicados.

A rotina *ESCTEMPO.M*

A função desta rotina é permitir ao utilizador alterar o intervalo de tempo de geração, visualização e tratamento dos sinais, que é definido, por defeito, como sendo o intervalo entre -1 e 1 seg (esta definição é feita na rotina *CAMATEST.M*, portanto logo no início da chamada do programa).

Ao utilizador é pedido, nesta rotina, que defina qual o instante inicial do intervalo de tempo de tratamento dos sinais e qual o instante final desse mesmo intervalo. Após o utilizador ter introduzido os valores pedidos, estes são-lhe apresentados no ecrã e é-lhe pedido que valide os dados introduzidos; caso ele não os valide, tem que reintroduzir os valores inicial e final do intervalo de tempo de tratamento dos sinais, e no caso de o utilizador validar os dados, estes são gravados no ficheiro *inttempo.mat* após o que é corrida a rotina de limpeza. Esta rotina de limpeza é chamada, como veremos adiante, para limpar a variável *vector*, para que todos os sinais fiquem definidos com o mesmo intervalo de amostragem de forma a não haver problemas quando se efectuarem operações entre eles.

A rotina *LIMPEZA.M*

A única acção realizada por esta rotina é efectuar a limpeza das variáveis *posvec* e *vector* que estão guardadas no ficheiro *buffer.mat*. Para esse efeito é carregado o ficheiro *buffer.mat* e as variáveis *posvec* e todas as posições da variável *vector* são colocadas a zero, ou seja, como se não se tivesse ainda começado a trabalhar. Depois de feita a limpeza destas variáveis, voltam a ser gravadas no ficheiro *buffer.mat* para serem depois utilizadas nas outras rotinas do programa. Durante o decurso de todas as operações de limpeza das variáveis é apresentada no ecrã uma mensagem de espera e quando a limpeza é concluída é pedido ao utilizador que carregue numa tecla para prosseguir a execução do programa.

A rotina *DESISTIR.M*

Esta rotina é invocada sempre que o utilizador pretende abandonar o programa. A sua estrutura é bastante simples, o que é derivado do facto de que a única coisa que esta rotina faz é pedir ao utilizador que confirme se pretende abandonar o programa. Se for esse o caso, o programa é abandonado e o utilizador retorna ao sistema operativo. Caso o utilizador

não pretenda abandonar o programa retorna ao menu principal após se ter feito uma limpeza do ecrã.

A rotina *VISUAL.M*

A rotina *VISUAL.M* destina-se a permitir a visualização de um qualquer sinal ou resultado da operação de um sinal, de forma gráfica. As diferentes hipóteses de visualização que são dadas, são as seguintes:

1- Visualizar uma onda
2 - Visualizar uma onda e a sua fft
3 - Visualizar duas ondas
4 - Visualizar duas ondas e as suas fft's
5 - Visualizar uma fft
6 - Visualizar o resultado de uma convolução ou correlação

O utilizador tem que escolher qual destas hipóteses de visualização pretende efectuar, após o que a rotina salta para o código especifico desse tipo de visualização. As várias hipóteses de visualização de que o utilizador dispõe, são apresentadas depois de se fazer a limpeza do ecrã e depois de se terem carregado os ficheiros *buffer.mat* e *inttempo.mat*.

No caso de o utilizador escolher visualizar uma onda, é feito, no ecrã, um gráfico dessa onda em função do tempo. O utilizador tem que indicar qual o vector que pretende visualizar e o gráfico é-lhe apresentado com o tamanho do ecrã. Caso o utilizador opte por visualizar uma onda e a sua fft, tem que indicar qual o vector que quer visualizar e a fft é calculada dentro desta própria rotina. O ecrã é dividido em dois e na parte esquerda é apresentada a onda que se pretendia visualizar e no lado direito é apresentada a sua respectiva transformada de Fourier em função da frequência.

Se o utilizador escolher a opção 3, visualizar duas ondas, o processo é o mesmo da opção em que pretende visualizar uma onda, mas o ecrã é dividido em dois e é apresentada uma onda do lado esquerdo do ecrã e outra do lado direito. O mesmo se passa se o utilizador optar por visualizar duas ondas e as respectivas fft's, em que o processo é semelhante ao caso de pretender visualizar uma onda e a respectiva fft, mas em que o ecrã é agora dividido em quatro partes, sendo apresentado na parte superior esquerda do ecrã uma onda e na parte superior direita a sua respectiva transformada de Fourier e na parte inferior esquerda apresenta-se a segunda onda e à sua direita a sua transformada de Fourier.

Se o utilizador optar por visualizar a fft de um sinal, tem que indicar qual o vector que pretende visualizar, ou seja, qual o vector em que se

encontra a fft, calculada na rotina *TFOURIER.M*, a visualizar e neste local só se vai proceder ao cálculo da escala frequêncial para traçar o eixo das abcissas do gráfico. A escolha desta opção pressupõe que a fft foi calculada na rotina *TFOURIER.M* e gravada numa posição da variável vector. Caso isto não tenha acontecido o resultado é descabido de sentido.

No caso de o utilizador pretender visualizar o resultado de uma correlação ou convolução, escolhe a opção 6 e vai ter que entrar os números dos dois vectores onde se encontram os resultados da correlação ou da convolução de dois sinais, uma vez que, como veremos, o resultado de uma destas operações ocupa dois vectores. Depois de o utilizador ter entrado os vectores onde se encontram os resultados da convolução ou da correlação, é apresentada no ecrã uma mensagem de espera e vai-se proceder ao cálculo da escala temporal. A escala temporal é guardada na variável *tau* e o resultado da convolução ou correlação é passado para um só vector com uma linha e duas mil colunas e denominado *CO*. Isto tem em vista facilitar a representação gráfica dos dados.

Depois de realizada a visualização escolhida pelo utilizador, este deve carregar numa tecla para limpar o ecrã e é-lhe perguntado se pretende visualizar outra onda. Caso ele pretenda efectuar outra visualização, a rotina é repetida, com a apresentação novamente do menu com as várias hipóteses de visualização que estão ao dispôr do utilizador. Caso contrário, o programa retorna ao menu principal.

A rotina *SOMAR.M*

Esta rotina é chamada quando se pretende adicionar duas onda. No início desta rotina é feita a limpeza do ecrã e carrega-se o ficheiro *buffer.mat* com a variável *vector*, após o que é pedido ao utilizador que entre os dois vectores a serem somados e o número do vector onde se pretende guardar o resultado. Feito isto, é apresentada no ecrã uma mensagem de espera e é efectuada a soma das duas ondas. A soma das ondas é realizada somando cada posição dos dois vectores que contêm as ondas a somar. Depois da soma realizada, é gravada a variável *vector* no ficheiro *buffer.mat* e é perguntado ao utilizador se pretende efectuar outra adição de ondas. Em função da resposta do utilizador o programa retorna ao menu principal ou volta a correr esta rotina.

A rotina *SUBTRAIR.M*

A rotina *SUBTRAIR.M* apresenta exactamente a mesma estrutura que a rotina *SOMAR.M*, à excepção de que agora os vectores que contêm os dois sinais vão ser subtraidos, ao contrário do que acontecia na rotina anterior em que eram somados.

A rotina *MULTIPLI.M*

Esta rotina apresenta ainda a mesma estrutura das duas rotinas anteriores, sendo a única diferença, o facto de que a operação que agora é realizada é a multiplicação dos dois sinais, ao contrário do que acontecia nas duas rotinas anteriores. A multiplicação dos dois sinais é feita ponto a ponto.

A rotina *COMPODC.M*

Esta rotina destina-se a adicionar uma componente contínua a ondas. No início desta rotina é feita a limpeza do ecrã e é carregado o ficheiro *buffer.mat*. Feito isto, é pedido ao utilizador que entre o vector ao qual se quer adicionar a componente contínua, qual o valor da componente contínua a adicionar a este vector e qual o vector em que se pretende guardar o resultado. Depois de o utilizador entrar estes dados, é apresentada no ecrã uma mensagem de espera e é realizada a operação de adicionar uma componente contínua ao vector pretendido. A adição da componente contínua é realizada, adicionando o valor da componente contínua que se quer adicionar ao sinal, ao valor que o sinal tem. Esta adição é feita em cada coluna da variável *vector*. O resultado é depois guardado e a variável vector é gravada no ficheiro *buffer.mat*.

Depois de estas operações terem sido realizadas, é dada a hipótese ao utilizador de realizar a adição de outra componente contínua ao sinal.

A rotina *MAISAMPL.M*

Esta rotina apresenta a mesma estrutura da rotina vista anteriormente, a rotina *COMPODC.M*, sendo a única diferença a operação

realizada, que nesta rotina é o aumento de amplitude do sinal. Este aumento de amplitude é conseguido multiplicando todas as posições do vector ao qual o utilizador quer aumentar a amplitude, por um factor de escala, que o utilizador define e que dá a relação de amplitudes, antes e depois do aumento de amplitude do sinal.

A rotina *TFOURIER.M*

Esta rotina implementa o cálculo da transformada de Fourier de um sinal. A rotina começa com a limpeza do ecrã e a seguir é carregado o ficheiro *buffer.mat*, que contém a variável *vector*. Seguidamente é perguntado ao utilizador qual o vector do qual ele pretende realizar o cálculo da Transformada de Fourier e qual o vector em que pretende guardar o resultado. Após o utilizador ter entrado estes valores é apresentada no ecrã uma mensagem de espera e começa o cálculo da transformada de Fourier do sinal desejado.

Para se calcular a transformada de Fourier do sinal, calcula-se a fft do sinal desejado e guardamos o resultado na variável *FFT*. Como o resultado da fft é simétrico, vamos truncar a variável *FFT*, limitando-a aos seus primeiros 512 elementos e vamos calcular o valor absoluto da fft, desprezando desta forma o efeito da sua fase, que não nos interessa considerar. O resultado assim obtido é guardado no vector indicado pelo utilizador, sendo as restantes posições deste vector (as posições 513 a 1000) ocupadas com o valor zero. Feito isto, vamos gravar de novo a variável *vector* no ficheiro *buffer.mat* e inquirir o utilizador se pretende efectuar a transformada de Fourier de outro sinal. Em função da sua resposta, esta rotina é abandonada ou executada de novo.

A rotina *ESPECTRO.M*

Esta rotina apresenta sensivelmente a mesma estrutura da rotina *TFOURIER.M*, à excepção de que agora o cálculo que é realizado é o do espectro de densidade de potência do sinal desejado pelo utilizador. Para este efeito, uma vez obtida a fft do sinal desejado, vamos multiplicá-la pelo seu complexo conjugado, o que nos dá o espectro de densidade de potência do sinal. Seguidamente o resultado é truncado, da mesma forma que na rotina anterior, e os restantes elementos do vector onde vamos guardar os resultados são também postos a zero.

A rotina *CONVOLUC.M*

Nesta rotina é implementado o cálculo da correlação de dois sinais. Para esse efeito a rotina começa por fazer a limpeza do ecrã e logo de seguida é carregado o ficheiro *buffer.mat*. Seguidamente o utilizador tem que entrar quais são os dois vectores onde se encontram os sinais a convoluir e quais são os dois vectores onde pretende guardar o resultado da convolução. O facto de serem necessários dois vectores para guardar o resultado da convolução prende-se com a própria definição de convolução, que diz que o resultado da convolução de dois sinais discretos com N amostras cada, tem um comprimento de 2N-1 amostras. Assim é calculada a convolução dos dois sinais e as primeiras N amostras do resultado são guardadas no primeiro vector onde se vai guardar o resultado da convolução e as restantes amostras do resultado são guardadas no segundo vector que contém o resultado. Neste segundo vector com o resultado, a sua última coluna é colocada a zero, uma vez que não tem nenhum valor do resultado da convolução para lá ser colocado.

Concluído o cálculo da convolução, a variável *vector* é gravada no ficheiro *buffer.mat*, e como normalmente o utilizador é inquirido sobre se pretende realizar mais outra convolução.

A rotina *CORRELAC.M*

Esta rotina tem uma estrutura semelhante à rotina CONVOLUC.M, excepto no facto de que nesta rotina a operação de convolução entre dois sinais (que era realizada na rotina CONVOLUC.M) é aqui substituída pela operação de correlação entre dois sinais.

CONCLUSÃO

POSSIBILIDADES DE EVOLUÇÃO

O trabalho desenvolvido neste estágio funcionou de forma correcta, se bem que não tenha sido possível testar o seu funcionamento em conjunto com o hardware por manifesta falta de disponibilidade temporal; no entanto, o software funcionou de forma correcta e a interligação experimental com o "device driver" correu da forma esperada.

Gostaria, mesmo assim, de deixar aqui presente, certas modificações que, penso, poderiam ser feitas a este conjunto de software de forma a melhorá-lo. Essas modificações passariam por desenvolver rotinas que permitissem fazer a análise cepstral de sinais e fazer a análise fft a dois canais (possibilitando a computação do espectro cruzado, função de coerência, funções resposta em frequência e funções de correlação).

Seria também de interesse aplicar uma função de janela, por exemplo a janela de Hanning, aos sinais recebidos do exterior para evitar o efeito de aliasing no cálculo da fft destes sinais. Outro melhoramento que penso que seria de bastante interesse, seria o de tornar possível o programa trabalhar com mais de seis vectores simultaneamente, o que neste momento limita bastante o poder de trabalho deste software.

ÍNDICE

BIBLIOGRAFIA

- Manual do MATLAB

- Apontamentos da cadeira de Sistemas de Instrumentação II
  Prof. Dr. Diamantino da Silva Freitas

ANEXOS

CAMATEST.M

```
%
% SOFTWARE DE GESTÃO DA CAMA DE TESTE PARA
MÓDULOS ELECTRÓNICOS
%

% Rotina de inicialização

 clear
 clc
 clg

% Rotina de apresentação

 disp('  CAMA DE TESTES PARA MÓDULOS E COMPONENTES
ELECTRÓNICOS  ')
 fprintf('\n\n\n')
 disp('  Trabalho realizado no âmbito do PRODEP por:          ')
 fprintf('\n\n')
 disp('     Alberto Xavier                              ')
 disp('     José Carlos Lobinho                           ')
 disp('     Manuel Fernando Silva                          ')
 disp('     Pedro Lobo                                ')
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 clear
 buffer=0;
 tempoini=-1;
 tempofim=1;
 save inttempo tempoini tempofim
 disp(' carregue numa tecla para prosseguir a execução do programa... ')
 pause
 clc

% Rotina de ajuda

 rotajuda

% Rotina de menu de operações possíveis
```

```
menuoper

% Rotina de saida

clear
clc
clg
```

ROTAJUDA.M

```
% Rotina de ajuda

 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 explicacao=input(' Pretende uma breve explicação do funcionamento do
programa?(s/n) ','s');
 clc
 while explicacao=='s'
     fprintf('\n\n\n\n\n\n\n\n\n')
     disp('     Este programa permite gerar várias formas de onda
simples ')
     disp(' e por meio de operações matemáticas obter formas de onda
mais ')
     disp(' complexas.                                          ')
     disp('     Uma vez gerada a forma de onda desejada este programa
permite ')
     disp(' que essa onda seja enviada para o circuito que se pretende
')
     disp(' analisar e permite também a recolha dos sinais de resposta
do ')
     disp(' circuito em análise.                                ')
     fprintf('\n\n\n\n\n\n\n\n')
     disp(' carregue numa tecla para prosseguir... ')
     pause
     clc
     fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
     explicacao=input('       Quer rever a explicacão?(s/n) ','s');
     clc
 end
```

MENUOPER.M

```
% Rotina de menu de operações possíveis

opcao2=-1;
opcao21='s';
while opcao21=='s'
  while (opcao2<0 | opcao2>16)
    fprintf('\n')
    disp('  Qual a operação que pretende executar?          ')
    fprintf('\n\n')
    disp('   0- Gerar sinal                                 ')
    disp('   1- Enviar sinal                                ')
    disp('   2- Receber sinal                               ')
    disp('   3- Enviar e receber simultâneamente um sinal   ')
    disp('   4- Visualisar um sinal                         ')
    disp('   5- Somar sinais                                ')
    disp('   6- Subtrair sinais                             ')
    disp('   7- Multiplicar sinais                          ')
    disp('   8- Introduzir componente contínua no sinal     ')
    disp('   9- Aumentar a amplitude do sinal               ')
    disp('  10- Convolução de sinais                        ')
    disp('  11- Correlação de sinais                        ')
    disp('  12- FFT de sinais                               ')
    disp('  13- Espectro de densidade de potência           ')
    disp('  14- Alterar intervalo de visualização           ')
    disp('  15- Fazer o reset dos sinais                    ')
    disp('  16- Abandonar o programa                        ')
    fprintf('\n\n')
    opcao2=input('  Qual a sua opção? ');
    clc
  end

  if    opcao2==0,
        menuonda;
  elseif opcao2==1,
        enviaond;
  elseif opcao2==2,
        recebeon;
```

```
elseif opcao2==3,
    envrecon;
elseif opcao2==4,
    visual;
elseif opcao2==5,
    somar;
elseif opcao2==6,
    subtrair;
elseif opcao2==7,
    multipli;
elseif opcao2==8,
    compodc;
elseif opcao2==9,
    maisampl;
elseif opcao2==10,
    convoluc;
elseif opcao2==11,
    correlac;
elseif opcao2==12,
    tfourier;
elseif opcao2==13,
    espectro;
elseif opcao2==14,
    esctempo;
elseif opcao2==15,
    limpeza;
else
    desistir;
end

fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n')
opcao21=input('      Pretende executar outra operação?(s/n) ','s');
opcao2=-1;
if opcao21=='n'
   clc
   desistir;
end
clc
end
```

MENUONDA.M

```
% Rotina de menu de ondas possiveis

opcao1=0;
opcao11='s';
while opcao11=='s'
  while (opcao1<1 | opcao1>10)
    fprintf('\n\n')
    disp('  Que tipo de onda pretende gerar?                ')
    fprintf('\n\n')
    disp('     1- Onda sinusoidal                          ')
    disp('     2- Onda quadrada                            ')
    disp('     3- Onda triângular                          ')
    disp('     4- Onda dente de serra                      ')
    disp('     5- Seno cardinal                          ')
    disp('     6- Impulso rectângular                      ')
    disp('     7- Impulso gaussiano                        ')
    disp('     8- Componente contínua                        ')
    disp('     9- Degrau de Heaviside                      ')
    disp('    10- Onda exponêncial                         ')
    fprintf('\n\n')
    opcao1=input('  Qual a sua opção? ');
    clc
  end

  posvec=0;
  while (posvec<1 |posvec>6)
      fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
      posvec=input('       Em que vector quer guardar esta onda?(1-6) ');
      clc
  end

  save buffer posvec

  if    opcao1==1,
        geraseno;
  elseif opcao1==2,
        geraquad;
```

```
  elseif opcao1==3,
        geratria;
  elseif opcao1==4,
        geradser;
  elseif opcao1==5,
        gerasenc;
  elseif opcao1==6,
        gerairec;
  elseif opcao1==7,
        geraigau;
  elseif opcao1==8,
        geracont;
  elseif opcao1==9,
        geraheav;
  else
        geraexpo;
  end

  fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
  opcao11=input('       Pretende gerar outra onda?(s/n) ','s');
  opcao1=0;
  clc
end
```

GERASENO.M

```
% Função para gerar uma onda sinusoidal

clc
clg
load buffer
load inttempo
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
freq=input('      Qual a frequência da sinusoide? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
fase=input('      Qual a fase inicial da sinusoide em graus? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
disp('                    Aguarde um momento. ')
disp('                    Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
w=2*pi*freq;
dt=(tempofim-tempoini)/1000;
t=tempoini;
for i=1:1000
   tempo(i)=t;
   sinal(i)=sin(w*tempo(i)+fase);
   vector(posvec,i)=sinal(i);
   t=t+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clc
clg
```

GERADSER.M

```
% Função para gerar uma onda em dente de serra

clc
clg
load buffer
load inttempo
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
freq=input('      Qual a frequência da onda em dente de serra? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
fase=input('      Qual a fase da onda em dente de serra em graus? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
disp('                  Aguarde um momento. ')
disp('                  Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
dt=(tempofim-tempoini)/1000;
T=1/freq;
t=fase/(freq*360)+tempoini;
t1=tempoini;
while  t>=T
     t=t-T;
end
while  t<=0
     t=t+T;
end
for i=1:1000,
   sinal(i)=freq*t;
   vector(posvec,i)=sinal(i);
   tempo(i)=t1;
   t=t+dt;
   t1=t1+dt;
   if t>=T,
      t=0;
   end
end
save buffer vector tempo
```

```
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

```
    end
    t=t+dt;
    tempo(i)=t1;
    t1=t1+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

GERAQUAD.M

```
% Função para gerar uma onda quadrada

clc
clg
load buffer
load inttempo
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
freq=input('       Qual a frequência da onda quadrada? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
dc=input('       Qual o duty cycle da onda quadrada em percentagem? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
fase=input('       Qual a fase da onda quadrada? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
disp('                    Aguarde um momento. ')
disp('                    Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
T=1/freq;
DC=dc/100;
t=fase/(360*freq)+tempoini;
t1=tempoini;
dt=(tempofim-tempoini)/1000;
while t<=0
    t=t+T;
end
while t>=T
    t=t-T;
end
for i=1:1000
   if   t<=T*DC
        sinal(i)=1;
   else
        sinal(i)=-1;
   end
   vector(posvec,i)=sinal(i);
```

```
    if  t>T
        t=0;
    end
    t=t+dt;
    tempo(i)=t1;
    t1=t1+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

GERACONT.M

```
% Função para gerar um sinal contínuo

 clc
 clg
 load buffer
 load inttempo
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 ampl=input('       Qual a amplitude que pretende para a componente
contínua? ');
 clc
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 disp('                    Aguarde um momento. ')
 disp('                    Cálculos em curso.  ')
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 dt=(tempofim-tempoini)/1000;
 t=tempoini;
 for i=1:1000
    sinal(i)=ampl;
    vector(posvec,i)=sinal(i);
    tempo(i)=t;
    t=t+dt;
 end
 save buffer vector tempo
 clc
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
 disp(' carregue numa tecla para prosseguir a execução do programa... ')
 pause
 clg
 clc
```

GERAIREC.M

```
% Função para gerar um impulso rectângular

clc
clg
load buffer
load inttempo
tempoin=0;
tempofi=1000;
while (tempoin<=tempoini | tempofi >=tempofim)
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   tempoin=input('      Qual o instante em que começa o impulso em
seg.? ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   tempofi=input('      Qual o instante em que acaba o impulso em seg.?
');
   clc
end
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
disp('                  Aguarde um momento. ')
disp('                  Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
dt=(tempofim-tempoini)/1000;
t=tempoini;
for i=1:1000
   if  (t<=tempofi & t>=tempoin)
      sinal(i)=1;
   else
      sinal(i)=0;
   end
   vector(posvec,i)=sinal(i);
   tempo(i)=t;
   t=t+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
```

```
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

GERAHEAV.M

```
% Função para gerar um degrau de heaviside

clc
clg
load buffer
load inttempo
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
disp('                    Aguarde um momento. ')
disp('                    Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
t=tempoini;
dt=(tempofim-tempoini)/1000;
for i=1:1000
   if  t>0
       sinal(i)=1;
   else
       sinal(i)=0;
   end
   vector(posvec,i)=sinal(i);
   tempo(i)=t;
   t=t+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

GERASENC.M

```
% Função para gerar um seno cardinal

 clc
 clg
 load buffer
 load inttempo
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n')
 freq=input('        Qual a frequência do seno cardinal? ');
 clc
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n')
 fase=input('        Qual a fase do seno cardinal em graus? ');
 clc
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 disp('                    Aguarde um momento. ')
 disp('                    Cálculos em curso. ')
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
 fase=fase*pi/180;
 w=2*pi*freq;
 dt=(tempofim-tempoini)/1000;
 t=tempoini;
 for i=1:1000
    if  (w*t+fase==0)
        sinal(i)=1;
    else sinal(i)=sin(w*t+fase)/(w*t+fase);
    end
    vector(posvec,i)=sinal(i);
    tempo(i)=t;
    t=t+dt;
 end
 save buffer vector tempo
 clc
 fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
 disp(' carregue numa tecla para prosseguir a execução do programa... ')
 pause
 clg
 clc
```

GERAIGAU.M

```
% Função para gerar um impulso gaussiano

clc
clg
load buffer
load inttempo
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
K=input('       Qual a constante do impulso gaussiano? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
disp('                  Aguarde um momento. ')
disp('                  Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
dt=(tempofim-tempoini)/1000;
t=tempoini;
for i=1:1000
        sinal(i)=exp(-K*(t*t));
        vector(posvec,i)=sinal(i);
        tempo(i)=t;
        t=t+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

GERAEXPO.M

```
% Função para gerar uma onda exponêncial

clc
clg
load buffer
load inttempo
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
K=input('       Qual a constante da exponêncial? ');
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
disp('                  Aguarde um momento. ')
disp('                  Cálculos em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
t=tempoini;
dt=(tempofim-tempoini)/1000;
for i=1:1000
   if  K==0
       sinal(i)=0;
   else sinal(i)=exp(-t/K);
   vector(posvec,i)=sinal(i);
   end
   tempo(i)=t;
   t=t+dt;
end
save buffer vector tempo
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clg
clc
```

ENVIAOND.M

```
% Rotina para proceder ao envio de ondas para o exterior

clc
clg

load buffer
load inttempo

T=(tempofim-tempoini)/1000;

numenv=0;
while (numenv<1 | numenv>6)
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   numenv=input('  Quantas ondas pretende enviar?(1-6)  ');
   clc
   if  (numenv<1 | numenv>6)
       fprintf('\n\n\n\n\n\n\n\n\n\n')
       disp('          Só pode enviar simultâneamente de 1 a 6 ondas! ')
       fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
       disp('  carregue numa tecla para prosseguir a execução do
programa... ')
       pause
       clc
   else
       fprintf('device','%g',numenv);
       fprintf('device',':');
   end
end
clc
for i=1:numenv,
   vecpos=0;
   while (vecpos<1 | vecpos>6)
      fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
      fprintf('       Qual o vector em que está a %gª onda a enviar? ',i)
      vecpos=input('       Introduza um número entre 1 e 6: ');
      clc
      if  (vecpos<1 | vecpos>6)
```

```
        fprintf('\n\n\n\n\n\n\n\n\n\n\n')
        disp('                    Esse vector não existe!.    ')
        disp('                  Introduza um número entre 1 e 6. ')
        fprintf('\n\n\n\n\n\n\n\n\n\n\n')
        disp(' carregue numa tecla para prosseguir a execução do
programa... ')
        pause
        clc
      end
    end
    agulha=0;
    while (agulha<1 | agulha>64)
      fprintf('\n\n\n\n\n\n\n\n\n\n\n')
      fprintf('      Em que agulha quer aplicar a %gª onda? ',i)
      agulha=input('      Introduza um número entre 1 e 64: ');
      clc
      if  (agulha<1 | agulha>64)
        fprintf('\n\n\n\n\n\n\n\n\n\n\n')
        disp('                    Essa agulha não existe!     ')
        disp('                  Introduza um número entre 1 e 64. ')
        fprintf('\n\n\n\n\n\n\n\n\n\n\n')
        disp(' carregue numa tecla para prosseguir a execução do
programa... ')
        pause
        clc
      else
        fprintf('\n\n\n\n\n\n\n\n\n\n\n')
        disp('                    Aguarde um momento.  ')
        disp('                     Envio em curso.    ')
        fprintf('\n\n\n\n\n\n\n\n\n\n\n')
        fprintf('device','%g',agulha);
        fprintf('device',';');
        fprintf('device','%g',T);
        fprintf('device',';');
        for i=1:1000
          fprintf('device','%g ',vector(vecpos,i))
        end
        fprintf('device',':');
        clc
      end
    end
  end
```

ESCTEMPO.M

```
% Rotina para alteração da escala temporal de visualização dos sinais

 tempomal='s';
 while tempomal=='s'
    clc
    fprintf('\n\n\n\n\n\n\n\n\n\n\n')
    tempoini=input('    A partir de que instante quer começar a visualizar os
sinais? ');
    clc
    fprintf('\n\n\n\n\n\n\n\n\n\n\n')
    tempofim=input('    E até que instante os quer ver? ');
    clc
    save inttempo tempoini tempofim
    fprintf('\n\n\n\n\n\n\n\n\n\n\n')
    fprintf('    O sinal vai começar a ser visualizado a partir do instante
%g\n',tempoini)
    fprintf('    e vai ser visualizado até ao instante %g\n\n',tempofim)
    tempomal=input('    Quer corrigir algum destes valores?(s/n) ','s');
    clc
    if tempomal=='n'
       limpeza;
    end
end
```

LIMPEZA.M

```
% Rotina de limpeza do acumulador das ondas

fprintf('\n\n\n\n\n\n\n\n\n\n\n')
disp('                    Aguarde um momento. ')
disp('                     Limpeza em curso.  ')
fprintf('\n\n\n\n\n\n\n\n\n\n\n')
load buffer
posvec=0;
for linha=1:6
   for coluna=1:1000
      vector(linha,coluna)=0;
   end
end
save buffer posvec vector
clc
fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
disp(' carregue numa tecla para prosseguir a execução do programa... ')
pause
clc
```

DESISTIR.M

```
% Rotina de terminação do programa

fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
desistir=input('      Tem a certeza que quer desistir?(s/n) ','s');
if   desistir=='s'
     clc
     fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n')
     disp(' carregue numa tecla para abandonar o programa... ')
     pause
     exit
else
    clc
end
```

VISUAL.M

```
% Rotina para visualização de ondas e suas fft's

clc
clg

load buffer
load inttempo

opver='s';
while opver=='s'
   fprintf('\n\n\n\n\n\n\n')
   disp('     Que tipo de visualização pretende efectuar?              ')
   disp('      1- Visualizar uma onda                                  ')
   disp('      2- Visualizar uma onda e a sua fft                      ')
   disp('      3- Visualizar duas ondas                                ')
   disp('      4- Visualizar duas ondas e as respectivas ffts          ')
   disp('      5- Visualizar uma fft                                   ')
   disp('      6- Visualizar o resultado de uma convolução ou correlação ')
   fprintf('\n\n\n')
   opvisual=input('        Qual a sua opção? ');
   clc

   if opvisual==1

      fprintf('\n\n\n\n\n\n\n\n\n\n\n')
      posvec=input('     Qual o vector que pretende visualizar?(1-6) ');
      clc

      plot(tempo,vector(posvec,1:1000))

      pause
      clc
      clg

   end

   if opvisual==2
```

```
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    posvec=input('     Qual o vector que pretende visualizar?(1-6) ');
    clc

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    disp('                    Aguarde um momento.    ')
    disp('                  Cálculo da fft em curso. ')
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
       FFT=fft(vector(posvec,1:1000));
       FT=abs(FFT(1:512));
       T=(tempofim-tempoini)/1000;
       f=(1/T)*(0:511)/1024;
    clc

    subplot(121),plot(tempo,vector(posvec,1:1000))
    subplot(122),plot(f,FT)

    pause
    clc
    clg

end

if opvisual==3

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    posvec1=input('     Qual o 1º vector a visualizar?(1-6) ');
    clc

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    posvec2=input('     Qual o 2º vector a visualizar?(1-6) ');
    clc

    subplot(121),plot(tempo,vector(posvec1,1:1000),'g')
    subplot(122),plot(tempo,vector(posvec2,1:1000),'b')

    pause
    clc
    clg

end
```

```
if opvisual==4

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    posvec1=input('     Qual o 1º vector a visualizar?(1-6) ');
    clc

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    posvec2=input('     Qual o 2º vector a visualizar?(1-6) ');
    clc

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    disp('                    Aguarde um momento.      ')
    disp('                  Cálculo da fft em curso.   ')
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
       FFT1=fft(vector(posvec1,1:1000));
       FT1=abs(FFT1(1:512));
       FFT2=fft(vector(posvec2,1:1000));
       FT2=abs(FFT2(1:512));
       T=(tempofim-tempoini)/1000;
       f=(1/T)*(0:511)/1024;
    clc

    subplot(221),plot(tempo,vector(posvec1,1:1000),'g')
    subplot(222),plot(f,FT1,'g')
    subplot(223),plot(tempo,vector(posvec2,1:1000),'b')
    subplot(224),plot(f,FT2,'b')

    pause
    clc
    clg

end

if opvisual==5

    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    posvec=input('     Qual o vector em que se encontra a fft?(1-6) ');
    clc

    T=(tempofim-tempoini)/1000;
    f=(1/T)*(0:999)/1024;
```

```
        plot(f,vector(posvec,1:1000))

        pause
        clc
        clg

    end

    if opvisual==6

        fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
        posvec1=input('     Qual o 1° vector onde se encontra o
resultado?(1-6) ');
        clc

        fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
        posvec2=input('     Qual o 2° vector onde se encontra o
resultado?(1-6) ');
        clc

        fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
        disp('                    Aguarde um momento.   ')
        disp('                    Operações em curso.   ')
        fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')

        deltat=tempofim-tempoini;
        T=(tempofim-tempoini+2*deltat)/2000;
        tau(1)=tempoini-deltat;
        for i=2:2000
           tau(i)=tau(i-1)+T;
        end

        for i=1:1000
           CO(i)=vector(posvec1,i);
        end

        for i=1001:2000
           CO(i)=vector(posvec2,i-1000);
        end
        clc
```

```
        plot(tau,CO)

        pause
        clc
        clg

    end

  fprintf('\n\n\n\n\n\n\n\n\n\n\n\n\n')
  opver=input('       Pretende visualizar outra onda?(s/n) ','s');
  clc
end
```

SOMAR.M

```
% Rotina para somar ondas

clc
clg
opsoma='s';
while opsoma=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   s1=input('   Qual o primeiro vector a ser somado? ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   s2=input('   Qual o segundo vector a ser somado? ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   st=input('   Qual o vector em que pretende guardar o resultado? ');
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento.  ')
   disp('                    Operações em curso.  ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n')

   vector(st,1:1000)=vector(s1,1:1000)+vector(s2,1:1000);

   save buffer vector

   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   opsoma=input('   Pretende efectuar outra adição?(s/n) ','s');
   clc

end
```

SUBTRAIR.M

```
% Rotina para subtrair ondas

clc
clg
opsub='s';
while opsub=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   s1=input('   Qual o primeiro vector a ser subtraido? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   s2=input('   Qual o segundo vector a ser subtraido? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   subt=input('   Qual o vector em que pretende guardar o resultado? ')
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n')

   vector(subt,1:1000)=vector(s1,1:1000)-vector(s2,1:1000);

   save buffer vector

   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   opsub=input('   Pretende efectuar outra subtração?(s/n) ','s')
   clc

end
```

MULTIPLI.M

```
% Rotina para multiplicar ondas

clc
clg
opmul='s';
while opmul=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   m1=input('   Qual o primeiro vector a ser multiplicado? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   m2=input('   Qual o segundo vector a ser multiplicado? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   mt=input('   Qual o vector em que pretende guardar o resultado? ')
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n')

   vector(mt,1:1000)=vector(m1,1:1000).*vector(m2,1:1000);

   save buffer vector

   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   opmul=input('   Pretende efectuar outra multiplicação?(s/n) ','s')
   clc

end
```

COMPODC.M

```
% Rotina para adicionar componente continua a ondas

clc
clg
opdc='s';
while opdc=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   dc1=input('   Qual o vector ao qual vai adicionar componente dc? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   dc=input('   Qual o valor da componente dc a adicionar? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   dct=input('   Qual o vector em que pretende guardar o resultado? ')
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n')

   vector(dct,1:1000)=vector(dc1,1:1000)+dc;

   save buffer vector

   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   opdc=input('   Pretende adicionar outra componente continua?(s/n) ','s')
   clc

end
```

MAISAMPL.M

```
% Rotina para aumentar a amplitude de ondas

clc
clg
opamp='s';
while opamp=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   amp1=input('   Qual o vector ao qual vai aumentar a amplitude? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   amp=input('   Qual o valor da relação de amplitudes? ')
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   ampt=input('   Qual o vector em que pretende guardar o resultado? ')
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n')

   vector(ampt,1:1000)=vector(amp1,1:1000)*amp;

   save buffer vector

   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   opamp=input('   Pretende alterar de novo a amplitude do sinal?(s/n)
','s')
   clc

end
```

TFOURIER.M

```
% Rotina para calcular a fft de ondas

clc
clg
opfft='s';
while opfft=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   tf1=input('   Qual o vector ao qual vai ser calculada a fft?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   tft=input('   Qual o vector em que pretende guardar a fft?(1-6) ');
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')

   FFT=fft(vector(tf1,1:1000));
   for i=1:512
      vector(tft,i)=abs(FFT(i));
   end
   for i=513:1000
      vector(tft,i)=0;
   end

   save buffer vector

   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   opfft=input('   Pretende efectuar outra fft?(s/n) ','s');
   clc

end
```

ESPECTRO.M

```
% Rotina para calcular o espectro em frequência de ondas

 clc
 clg
 opesp='s';
 while opesp=='s'

    clc
    load buffer
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    esp1=input('    Qual o vector ao qual vai ser calculado o espectro?(1-6)
');
    clc
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    espt=input('    Qual o vector em que pretende guardar o espectro?(1-6)
');
    clc

    fprintf('\n\n\n\n\n\n\n\n\n\n\n')
    disp('                    Aguarde um momento.  ')
    disp('                    Operações em curso.  ')
    fprintf('\n\n\n\n\n\n\n\n\n\n\n')

    FT=fft(vector(tf1,1:1000));
    vector(tft,1:512)=FT(1:512).*conj(FT(1:512);
    vector(tft,513:1000)=0;

    save buffer vector

    clc
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    opesp=input('    Pretende calcular outro espectro?(s/n) ','s');
    clc

 end
```

CONVOLUC.M

```
% Rotina para realizar a convolução de duas ondas

clc
clg
opconv='s';
while opconv=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   conv1=input('   Qual o primeiro vector a ser convoluido?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   conv2=input('   Qual o segundo vector a ser convoluido?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   convt1=input('   Qual o 1° vector em que pretende guardar o
resultado?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   convt2=input('   Qual o 2° vector em que pretende guardar o
resultado?(1-6) ');
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')

   CON(1:1999)=conv(vector(conv1,1:1000),vector(conv2,1:1000));

   vector(convt1,1:1000)=CON(1:1000);
   vector(convt2,1:999)=CON(1001:1999);
   vector(convt2,1000)=0;

   save buffer vector
```

```
    clc
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    opconv=input('    Pretende efectuar outra convolução?(s/n) ','s');
    clc

end
```

CORRELAC.M

```
% Rotina para realizar a correlação de duas ondas

clc
clg
opcorr='s';
while opcorr=='s'

   clc
   load buffer
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   corr1=input('   Qual o primeiro vector a ser correlacionado?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   corr2=input('   Qual o segundo vector a ser correlacionado?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   corrt1=input('   Qual o 1° vector em que pretende guardar o
resultado?(1-6) ');
   clc
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   corrt2=input('   Qual o 2° vector em que pretende guardar o
resultado?(1-6) ');
   clc

   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
   disp('                    Aguarde um momento. ')
   disp('                    Operações em curso. ')
   fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')

   COR(1:1999)=xcorr(vector(corr1,1:1000),vector(corr2,1:1000));

   vector(corrt1,1:1000)=COR(1:1000);
   vector(corrt2,1:999)=COR(1001:1999);
   vector(corrt2,1000)=0;

   save buffer vector
```

```
    clc
    fprintf('\n\n\n\n\n\n\n\n\n\n\n\n')
    opcorr=input('    Pretende efectuar outra correlação?(s/n) ','s');
    clc

end
```