МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ЦЕНТЪР
"КОНКУРЕНТНА СИСТЕМА ЗА ОБУЧЕНИЕ
И УПРАВЛЕНИЕ НА ВИСШЕТО ОБРАЗОВАНИЕ"

ПРОЕКТ:
"СЪЗДАВАНЕ НА УНИВЕРСИТЕТСКА СИСТЕМА
ЗА ОСИГУРЯВАНЕ КАЧЕСТВОТО НА ОБУЧЕНИЕ
В ТРЕТАТА СТЕПЕН НА ВИСШЕТО ОБРАЗОВАНИЕ -
ДОКТОРАНТУРАТА"

Георги Кръстев

Цветозар Георгиев

# СРЕДСТВА ЗА АВТОМАТИЗАЦИЯ НА НАУЧНИТЕ ИЗСЛЕДВАНИЯ

Русе
2002

**С П О Н С О Р**
**на изданието:**

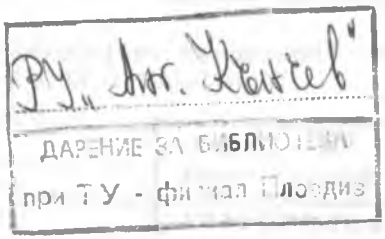
**"ЕЛСИ" ООД**
**Русе**
**тел./факс: 082-822 046**

РУСЕНСКИ УНИВЕРСИТЕТ
"Ангел Кънчев"



Георги Кръстев
Цветозар Георгиев

# СРЕДСТВА ЗА АВТОМАТИЗАЦИЯ НА НАУЧНИТЕ ИЗСЛЕДВАНИЯ




Русе
2002

Настоящото учебно пособие е от поредицата **"Библиотека за докторанта"**, издаването на която е част от програмата на Академичния съвет на Русенски университет "Ангел Кънчев" за работа с докторантите. Пособието е предназначено за бакалаври и магистри, които са в третата степен на висшето си образование, но може да се използва и от всички, които се занимават с научно-изследователска дейност.

**Първият раздел** съдържа информация за основните възможности на програмната среда MATLAB, предназначена за провеждане на инженерни и научни изчисления с висококачествена визуализация на получените резултати. Материалът е поднесен във вид, подходящ за възприемане от възможно най-широк кръг читатели, интересуващи се от практикуване на системния анализ с високо производителни средства, при което степента на личната компютърна подготовка не трябва да е висока.

**Вторият раздел** разглежда възможностите на приложението работещо в средата на MATLAB – SIMULINK. Това е преди всичко инструмент за симулиране на системи, които се конструират чрез блокове, различни генератори на въздействия и средства за наблюдаване и обработване на получените резултати.

**Третият раздел** включва информация за основните концепции и възможности на софтуерния пакет LabVIEW и неговия език за програмиране G за създаване на виртуални инструменти, измерване, обработване, анализ и съхраняване на данни.

В **приложението** е дадена сравнително пълна информация за основните и стандартни функции в MATLAB.

Поради ограниченост на обема, учебното пособие покрива начален курс за работа с тези продукти.

Участието на авторите е както следва:
MATLAB, SIMULINK и Приложение – инж. Георги Кръстев;
LabVIEW – инж. Цветозар Георгиев.

Авторите изказват благодарност на рецензента доц. д-р Ангел Смрикаров.



ISBN 954-712-178-2

## СЪДЪРЖАНИЕ

# РАЗДЕЛ I. MATLAB

## 1. ВЪВЕДЕНИЕ

Системата *MATLAB* (съкращение от *MATrix LABoratory* - МАТрична ЛАБоратория) е разработена от фирмата *The MathWorks*. Тя е интерактивна среда за изпълнение на инженерни и научни разчети и допуска възможност за обръщения към подпрограми, написани на езиците *Fortran*, *C* и *C++.*

Системата е реализирана като удобна програмна среда, позволяваща формулиране на проблемите и получаване на решение в математическа форма, без програмиране.

Най- известни области на приложение на *MATLAB* са:

- ♦ математически изчисления;
- ♦ разработка на алгоритми;
- ♦ изчислителен експеримент, имитационно моделиране;
- ♦ анализ на данни, изследване и визуализация на резултати;
- ♦ научна и инженерна графика;
- ♦ разработка на приложения, включително графичен потребителски интерфейс.

*MATLAB*- това е интерактивна среда с основен обект масив, на който не е необходимо да се указва явно размера. Това позволява да се решават бързо много изчислителни задачи, свързани с векторно- матрични формулировки.

Системата *MATLAB* включва програмна среда и език за програмиране. Една от най- силните страни на системата се състои в това, че на езика *MATLAB* могат да бъдат написани програми за многократно използване. Потребителя може сам да напише специализирани функции и програми, които да оформи във вид на *M*- файлове.

Колекцията от *M*- файлове за решаване на определени задачи или проблеми са обособени в тематични групи в специални папки, включени в папката *MATLAB Application Toolboxes*. Всяка папка е набор от полезни функции и често е резултат от работата на много изследователи по цял свят.

Системата *MATLAB 5.x* включва следните програмни продукти, съпровождани от справочна и методическа документация:

## Базови програмни средства

| № | Наименование на продукта | Версия | | Предназначение |
|---|---|---|---|---|
| | | **5.2** | **5.0** | |
| 1 | *MATLAB for Windows* | 5.2 | 5.0 | Система за инженерни и научни разчети |
| 2 | *MATLAB Compiler* | 1.2 | 1.1 | Компилатор на програми от *MATLAB* на език *C* |
| 3 | *MATLAB C Math Library* | 1.2 | 1.1 | Библиотека на математическите функции на системата *MATLAB* на език *C* |
| 4 | *MATLAB C++ Math Library* | 1.2 | 1.0 | Библиотека на математическите функции на системата *MATLAB* на езика *C++* |
| 5 | *SIMULINK for Windows* | 2.2 | 2.0 | Моделиране на динамични системи |
| 6 | *Real- Time Workshop (RTW)* | 2.2 | 1.1c | Моделиране на системи в реално време |
| 7 | *SIMULINK RTW Ada Extension* | 1.2 | 1.1c | Разширение на *RTW* на базата на езика *Ada* |
| 8 | *SIMULINK Accelerator* | 2.2 | 1.1a | Ускорител на процедурите за моделиране |
| 9 | *Applix Link* | 1.0.2 | - | Интерфейс със семейство продукти *Applix* ware на фирмата *Applix* за платформи *UNIX* и *WindowsNT* |
| 10 | *Excel Link* | 1.0 | 1.0 | Интерфейс със системата *Excel* (5.0 и по-висока версия) |
| 11 | *The Student Edition of MATLAB* | 5 | 5 | Версия на *MATLAB* за студенти |
| 12 | *The Student Edition of SIMULINK* | 2.0 | 2.0 | Версия на *SIMULINK* за студенти |

**Пакети приложни програми**

| № | Наименование на продукта | Версия | | Предназначение |
|---|---|---|---|---|
| | | **5.2** | **5.0** | |
| | **Математика** | | | |
| 1 | *NAG Foundation Toolbox* | 1.0.2 | 1.0 | Библиотека от математически функции *The Numerical Algorithms Group Ltd.* |
| 2 | *Spline Toolbox* | 2.0 | 1.1.3 | Сплайн апроксимация |
| 3 | *Statistics Toolbox* | 2.1.1 | 2.0.1 | Статистика |
| 4 | *Optimization Toolbox* | 1.5.2 | 1.5.0 | Оптимизация |
| 5 | *Fuzzy Logic Toolbox* | 2.0 | 1.0.2 | Размити множества |
| 6 | *Neural Network Toolbox* | 3.0 | 2.0.3 | Невронни мрежи |
| 7 | *Partial Differential Equations Toolbox* | 1.0.3 | 1.0.1 | Уравнения с частни производни |
| 8 | *Symbolic Math Toolbox* | 2.0.1 | 2.0 | Символна математика |
| 9 | *Extended Symbolic Math Toolbox* | 2.0.1 | 2.0 | Разширена символна математика |
| | **Анализ и синтез на системи за управление** | | | |
| 10 | *Control System Toolbox* | 4.1 | 4.0 | Системи за управление |
| 11 | *Nonlinear Control Design Toolbox* | 1.1.2 | 1.1.0 | Проектиране на нелинейни системи |
| 12 | *Robust Control Toolbox* | 2.0.5 | 2.0.3 | Робастно управление |
| 13 | *Model Predictive Control Toolbox* | 1.0.3 | 1.0.1 | Управление с еталонен модел |
| 14 | *$\mu$-Analysis and Synthesis Toolbox* | 3.0.3 | 3.0.1 | $\mu$-анализ и синтез |
| 15 | *Quantitative Feedback Theory Toolbox* | 1.0.3 | 1.0.1 | Проектиране на робастни системи с обратна връзка |
| 16 | *LMI (Linear Matrix Inequality Control Toolbox* | 1.0.4 | 1.0.1 | Синтез на системи за управление на основата на линейни матрични неравенства |
| | **Идентификация на системи за управление** | | | |
| 17 | *System Identification Toolbox* | 4.0.4 | 4.0.1 | Идентификация на обекти |
| 18 | *Frequency Domain System Identification Toolbox* | 2.0.3 | 2.0.1 | Идентификация в честотната област |

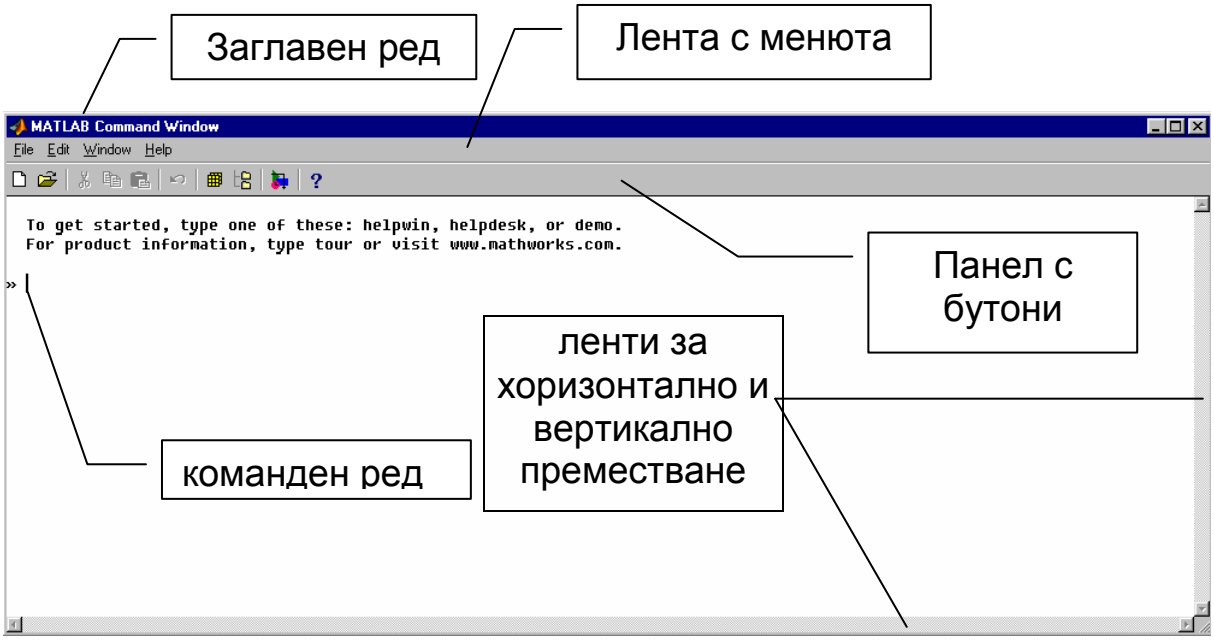
| | | | | |
|---|---|---|---|---|
| | **Моделиране с използване на SIMULINK** | | | |
| 19 | *Data Signal Processing (DSP) Blockset* | 2.2 | 1.0 | Обработка на цифрови сигнали |
| 20 | *Fixed Point Blockset* | 1.2 | 1.0 | Фиксирана разрядност |
| 21 | *Power System Blockset* | 1.0 | - | Моделиране на енергетични системи |
| 22 | *Stateflow Toolbox* | 1.0.6 | - | Моделиране на събития |
| | **Обработка на сигнали и изображения** | | | |
| 23 | *Signal Processing Toolbox* | 4.1 | 4.0 | Обработка на сигнали |
| 24 | *Higher- Order Spectral Analysis Toolbox* | 2.0.2 | 2.0.1 | Спектрален анализ |
| 25 | *Image Processing Toolbox* | 2.1 | 2.0 | Обработка на изображения |
| 26 | *Wavelet Toolbox* | 1.1 | 1.0.2 | Импулсна декомпозиция |
| | **Разни** | | | |
| 27 | *Communication Toolbox* | 1.3 | 1.0.1 | Комуникационни системи |
| 28 | *Financial Toolbox* | 1.1 | 1.0.2 | Финанси |
| 29 | *Database Toolbox* | 1.0 | - | Работа с бази данни |
| 30 | *Notebook Toolbox* | | | Написване на *M*- книги |

## 2. ПРОГРАМНА СРЕДА НА *MATLAB*

*MATLAB* е изключително голям продукт, състоящ се от интегрирана среда за разработка и език за програмиране. Средата съдържа редактор с вградена програма за настройка (*debugger* - дебъгер), средства за преглед на работните области и пътищата за достъп, поддържа динамично взаимодействие с външни системи от типа на *Microsoft Word*, *Excel* и др.

### 2.1. КОМАНДЕН ПРОЗОРЕЦ

Стандартният начин за зареждане на *MATLAB* е чрез щракване върху бутона *Start*, откъдето се избира *Programs* | *MATLAB*. Още по бързо зареждане на *MATLAB* е чрез двукратно щракване върху иконата за бързо избиране на *MATLAB* (ако е създадена). След успешното му стартиране се появява команден прозорец (Фиг. 1.1).

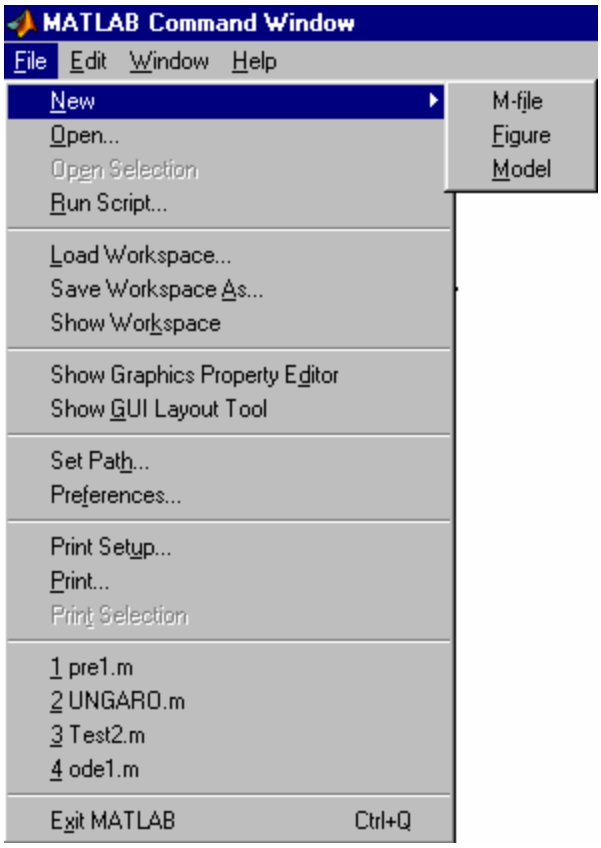



*Фиг. 1.1. Команден прозорец на MATLAB*

- *заглавен ред* – на този ред се изписва *MATLAB Command Window*;
- *лента с менюта* - тук са групирани всички функционални възможности на *MATLAB*;
- *панел с бутони* - по-важните команди се представят чрез бутони с икони. При избиране на даден бутон се изпълнява съответстващата му команда;
- *ленти за хоризонтално и вертикално преместване* - намират се в десния и долния край на прозореца. Служат за преместване на представения участък от изображението;
- *команден ред* - за въвеждане на желаните за изпълнение команди. При завършване на въвеждането за да се изпълни командата се натиска клавиша *Enter*.

## 2.2. ОПИСАНИЕ НА МЕНЮТАТА

Падащо меню ***File***.



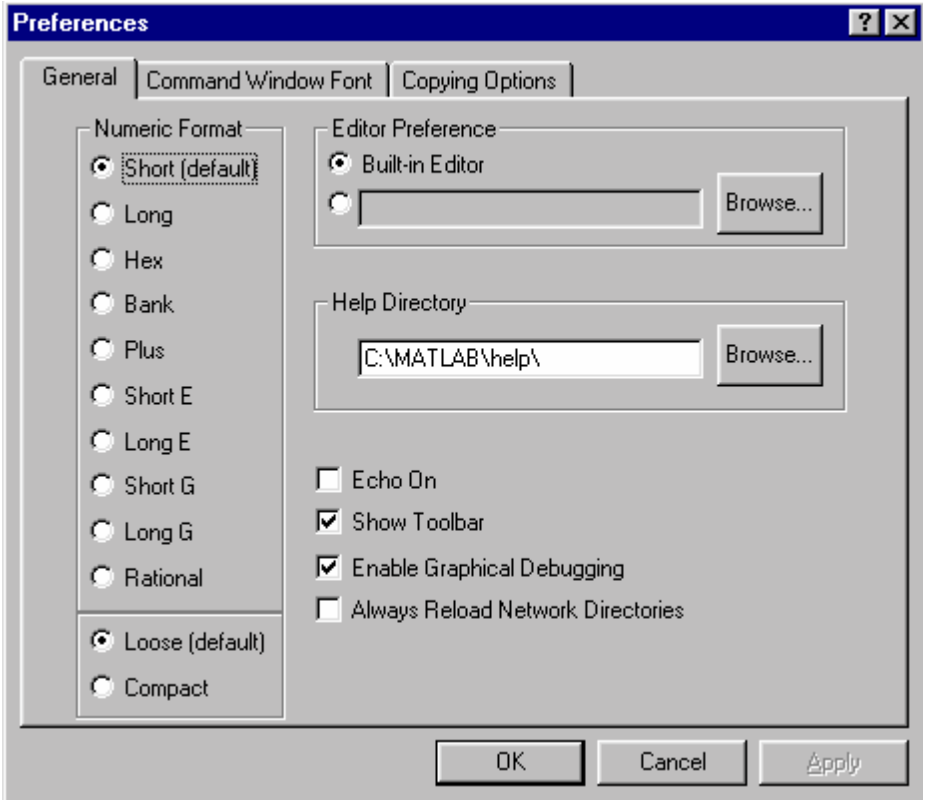

*Фиг.1.2. Падащо меню File*

Предлаганите функционални възможности са представени в следната таблица.

| **Опция** | **Под-опция** | **Предназначение** |
|---|---|---|
| *New* | *M-file*<br>*Figure*<br>*Model* | Отваря в редактора нов файл<br>Отваря графичен прозорец<br>Отваря в редактора нов файл - модел за динамична симулация |
| *Open* | | Отваря в редактора указания файл |
| *Open Selection* | | Отваря в редактора файл с име изписано на командния ред |
| *Run Script* | | Отваря прозорец за стартиране на *Script*-файл |
| *Load Workspace* | | Отваря прозорец за зареждане на *MAT*-файл |
| *Save Workspace As* | | Отваря прозорец за съхранение на *MAT*-файл |
| *Show Workspace* | | Отваря прозорец със средства за разглеждане на работната област Workspace Browser |

| | | |
|---|---|---|
| *Show Graphics Property Editor ** | | Отваря прозорец на редактора на свойствата на графичните обекти |
| *Show GUI Layout Tool ** | | Отваря прозорец със средства за разработка на графичен потребителски интерфейс |
| *Set Path* | | Отваря прозорец *Path Browser* - средство за разглеждане и установяване на пътя за достъп |
| *Preferences* | | Избор на характеристики |
| *Print Setup* | | Установяване опциите на принтера |
| *Print* | | Установяване опциите на изхода за печат |
| *Print Selection* | | Печатане на отделни фрагменти |
| *Exit MATLAB Ctrl+Q* | | Прекратяване на работата с програмата |

Опцията характеристики (***Preferences***) включва три страници. На фиг. 1.3 е показана страницата *General*.

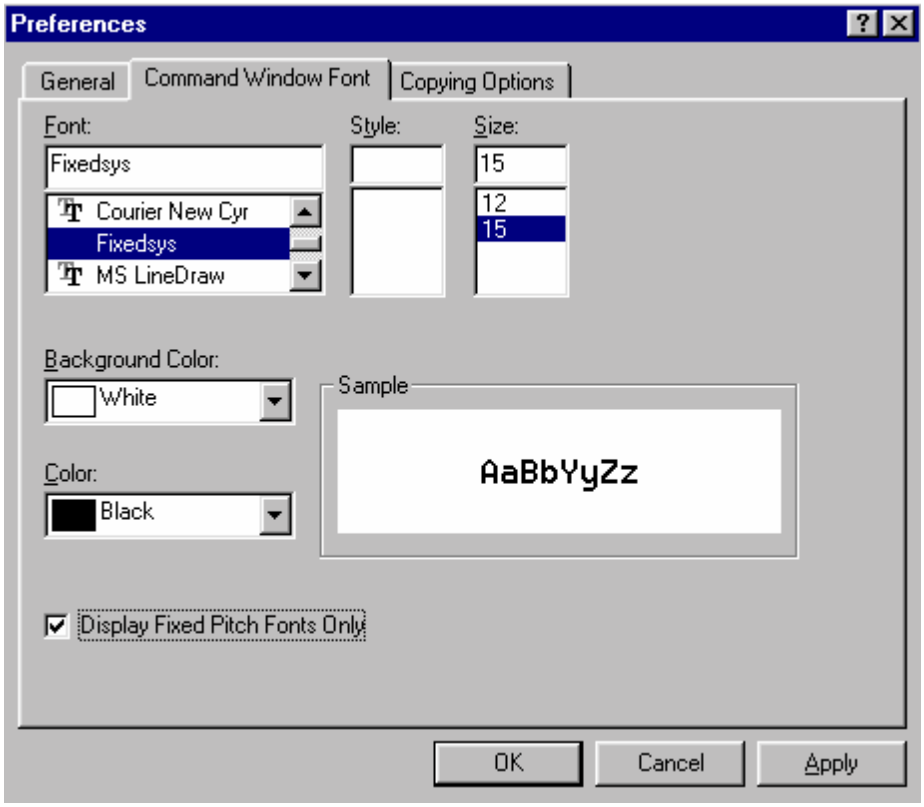



*Фиг. 1.3. Страница General на диалогов прозорец Preferences*

Отделните значения са:

| **Формат на данните** | **Предназначение** |
|---|---|
| *Numeric Format* | Избор на формата за представяне на числата. По подразбиране формата е *Short* (къс), а разстоянията са *Loose* (свободен). При избор на *Compact* резултатите са по-сбити. |
| *Editor Preference* | Избор на текстов редактор. По подразбиране е избран вградения редактор *Built in Editor* |
| *Help Directory* | Указване пътя до помощната информация Help |
| *Echo on* | Показва на екрана изпълняваните команди от *Script* файла |
| *Show Toolbar* | Показва на екрана инструменталния панел |
| *Enable Graphical Debugging* | Поддържа настройка на графиката |
| *Always Reload Network Directories* | Винаги презареждане на мрежовите папки |

Страницата шрифт на командния прозорец (***Command Window Font***) е показана на Фиг. 1.4.

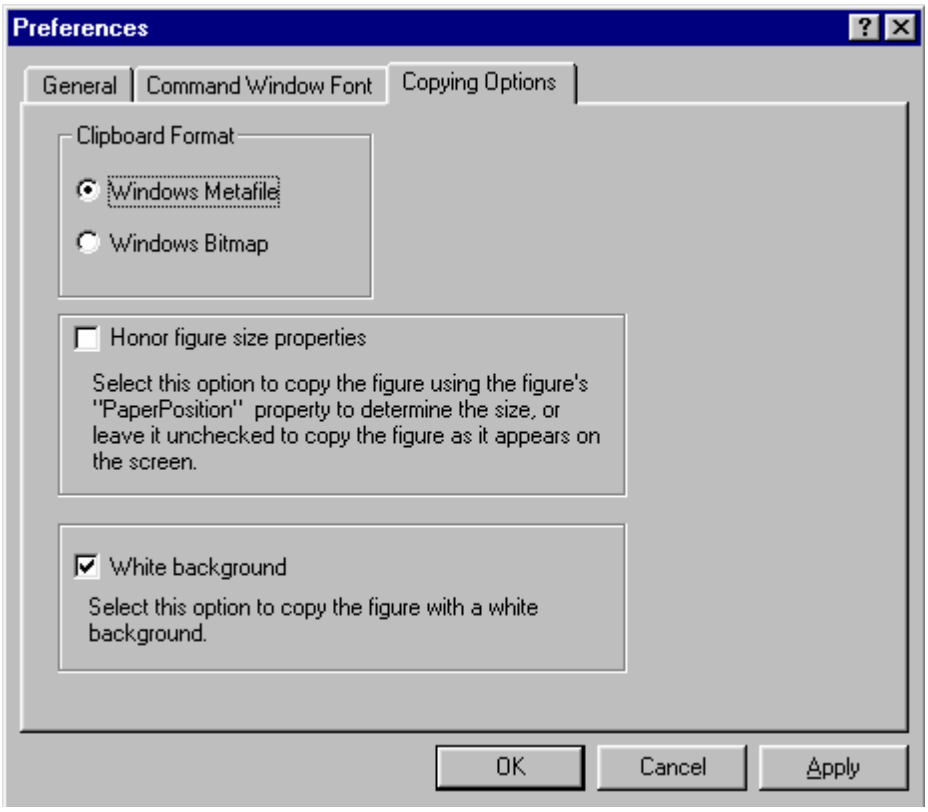



*Фиг. 1.4. Страница Command Window Font на диалогов прозорец Preferences*

Значенията са:

| Поле или маркер | Предназначение |
|---|---|
| *Font* | Шрифт на извеждания текст в командния прозорец |
| *Style* | Тип на шрифта: *Light*- светъл<br>*Regular*- нормален<br>*Bold*- удебелен |
| *Size* | Размер на шрифта: 10<br>12<br>15 |
| *Background Color* | Цвят на фона: *Silver*- сребрист<br>*Red*- червен<br>*Lime*- лимонен<br>*Yellow*- жълт<br>*Blue*- син<br>*Fuscia*- светло- виолетов<br>*Aqua*- сив<br>*White*- бял |
| *Color* | Цвят на символа |
| *Sample* | Образец на избрания фон и шрифт |
| *Display Fixed Pitch Fonts Only* | Показва само шрифтове с фиксирани стъпки<br>Показва всички шрифтове |

Страница опции при копиране (***Copying Options***) Фиг. 1.5.



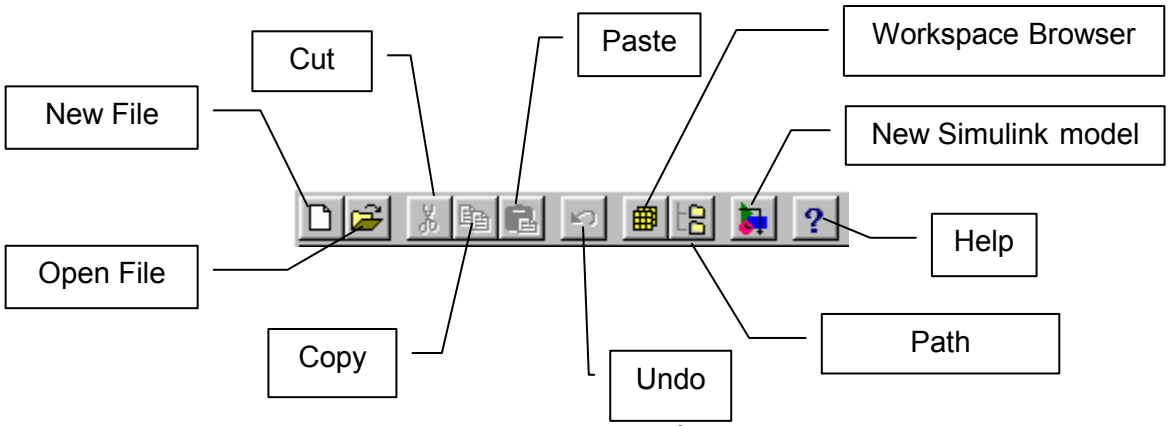

*Фиг. 1.5. Страница Copyng Options на диалогов прозорец Preferences*

Значенията са:

| Поле или маркер | Предназначение |
|---|---|
| *Clipboard Format* | Формат на копираната информация |
| *Honor figure size properties* | Маркер за размера на фигурата. Изборът на тази опция позволява копиране на фигурата, отчитайки свойствата на ”*Paper- Position*” (*за Windows Bitmap* не действуват) |
| *White Background* | Маркер за фона бял/ черен (за *Windows Bitmap* не действуват) |

**Панел с бутони**

По-важните команди се представят чрез бутон с икона. При посочване на бутон от панела, под него се появява жълт правоъгълник с кратка информация за съответната функция. При натискане на бутон се изпълнява съответстващата му команда. По този начин се осигурява бърз достъп до операциите с *M* - файлове (фиг. 1.6).



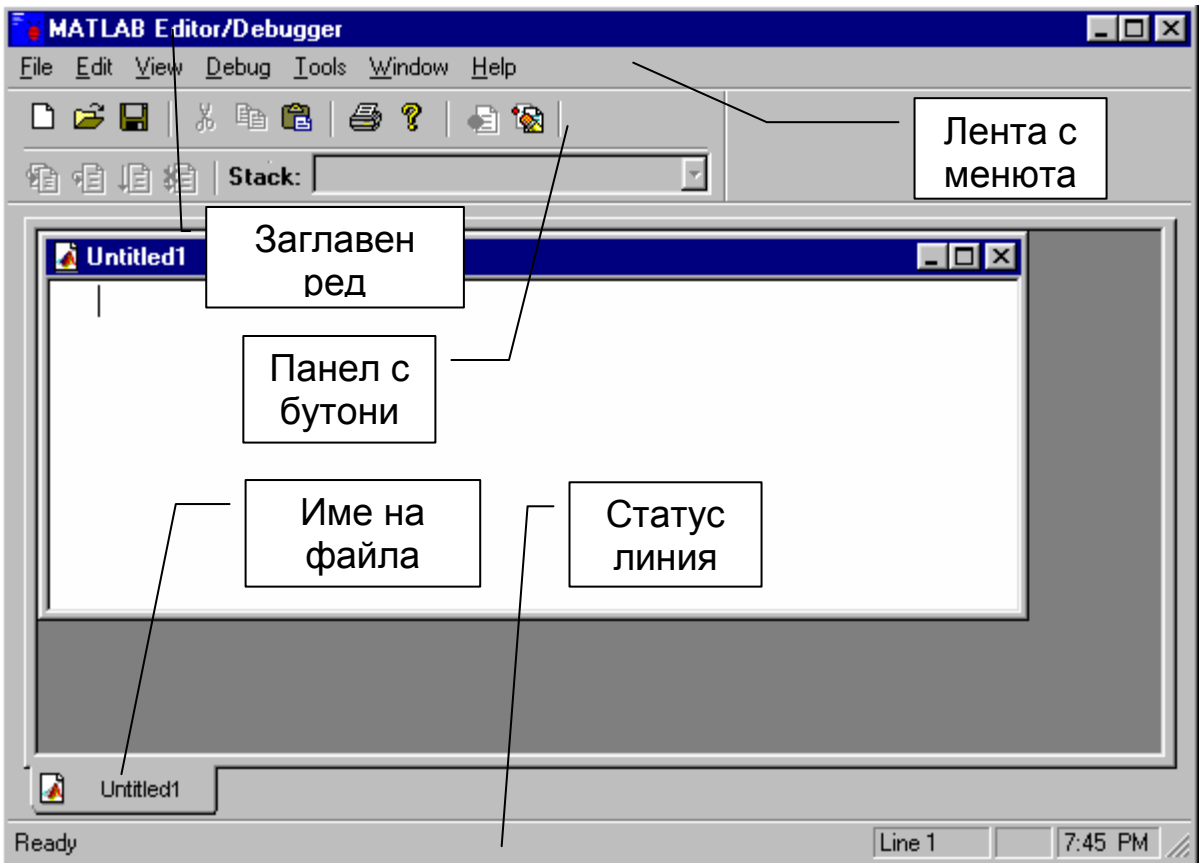

*Фиг. 1.6. Панел с бутони*

Тези операции включват:

- ***New File*** - създаване на нов *M*- файл.
- ***Open File*** - отваряне на съществуващ *M*- файл.
- ***Cut*** - отделяне на фрагмент.
- ***Copy*** - копиране на фрагмент.
- ***Paste*** - вмъкване на фрагмент.
- ***Undo*** - възстановяване само на изпълнените операции.
- ***Workspace Browser*** - разглеждане на работната област.
- ***Path Browser*** - разглеждане на пътя за достъп.
- ***Help -*** текуща помощ.

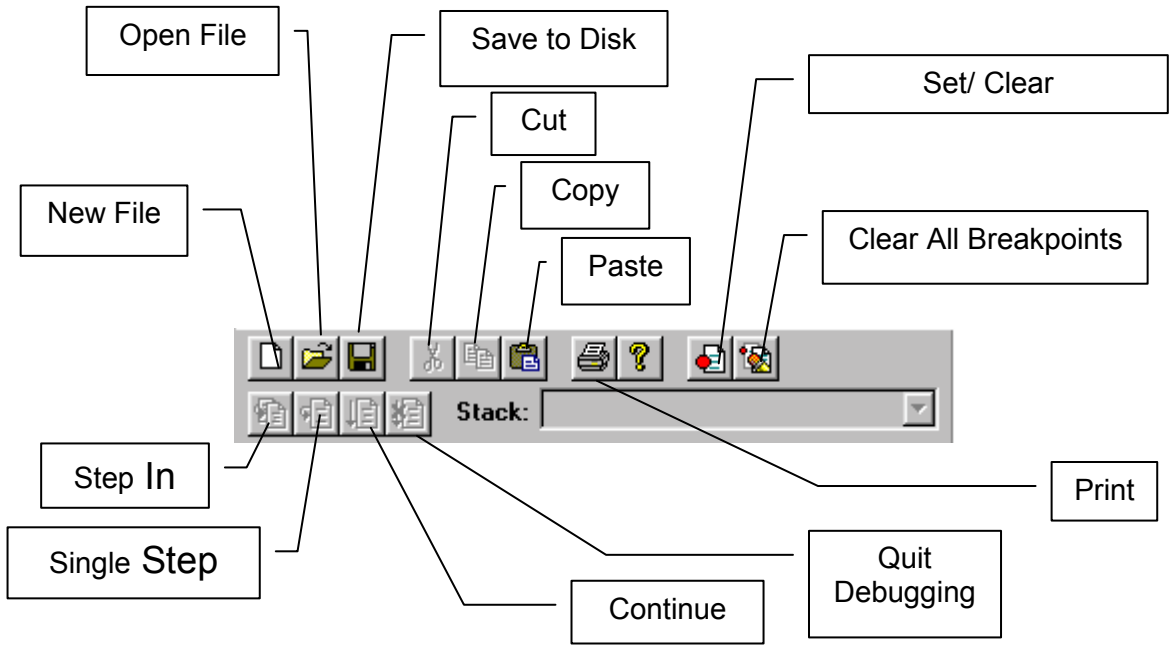
**Редактор на *M*- файлове**

В състава на системата *MATLAB 5.x* е включен редактора на *M*-файлове. Стартирането му може да се извърши по няколко начина: чрез командата ***edit*** или ***edit*** <име на *M*-файл> изпълнена от командния ред или чрез командата ***File/New/M-file*** от командния прозорец. На фиг. 1.7 е показан работният прозорец на редактора.




*Фиг. 1.7. Работен прозорец на редактора*

Панелът с бутони на редактора за версия *MATLAB 5.2* е показан на фиг. 1.8.




*Фиг.1.8. Панел с бутони на редактора на M- файлове*

Показаните бутони имат следното предназначение:

- ***New File*** - създаване на нов *M*-файл.
- ***Open File*** - отваряне на съществуващ *M*-файл.
- ***Save to Disk*** - съхранение на *M*-файл на диска.
- ***Cut*** - отделяне на фрагмент.
- ***Copy*** - копиране на фрагмент.
- ***Paste*** - вмъкване на фрагмент.
- ***Print*** - отпечатване на *M*-файл.
- ***Help*** - текуща помощ.
- **Set/ Clear Breakpoint** - установяване/ изчистване на контролна точка.
- ***Clear All Breakpoints*** - изчистване на всички контролни точки.
- ***Step In*** - влизане в *M*-модула.
- ***Single Step*** - изпълнение на една стъпка в режим на настройка.
- ***Continue*** - продължаване на изпълнението.
- ***Quit Debugging*** - край на режима на настройка.

**Текуща папка**

В операционната система *MATLAB* файловете се съхраняват в папки (*folders*) и всеки файл има собствен уникален път (*pathname*). Той представлява списък от папките, през които трябва да се премине от най-външното ниво от папки на устройството, за да се стигне до папката, съдържаща файла.

Системата *MATLAB* използва понятието текуща папка при работа с *M*- и *MAT*- файлове. Това е последната папка, използвана в програмата и е мястото където *MATLAB* записва файловете по подразбиране. За показване на текущата папка се изпълнява командата *cd*. С разширения запис на командата се извършва промяна на текущата папка *cd* <нов път за достъп>.

**Списък на пътищата за достъп**

За търсене на *M*- файлове системата *MATLAB* използва т. н. механизъм на път за достъп.

Например, при търсене на файла с име *foo* в *MATLAB* се изпълняват следните действия:

- проверява се, дали *foo* е име на променлива;
- проверява се, дали *foo* е вградена функция;
- има ли в текущата папка *M*-файл с име *foo.m*;
- има ли във всички папки от списъка на пътя за достъп *M*-файл с име *foo.m*.

**Работа със списъка на пътищата за достъп**

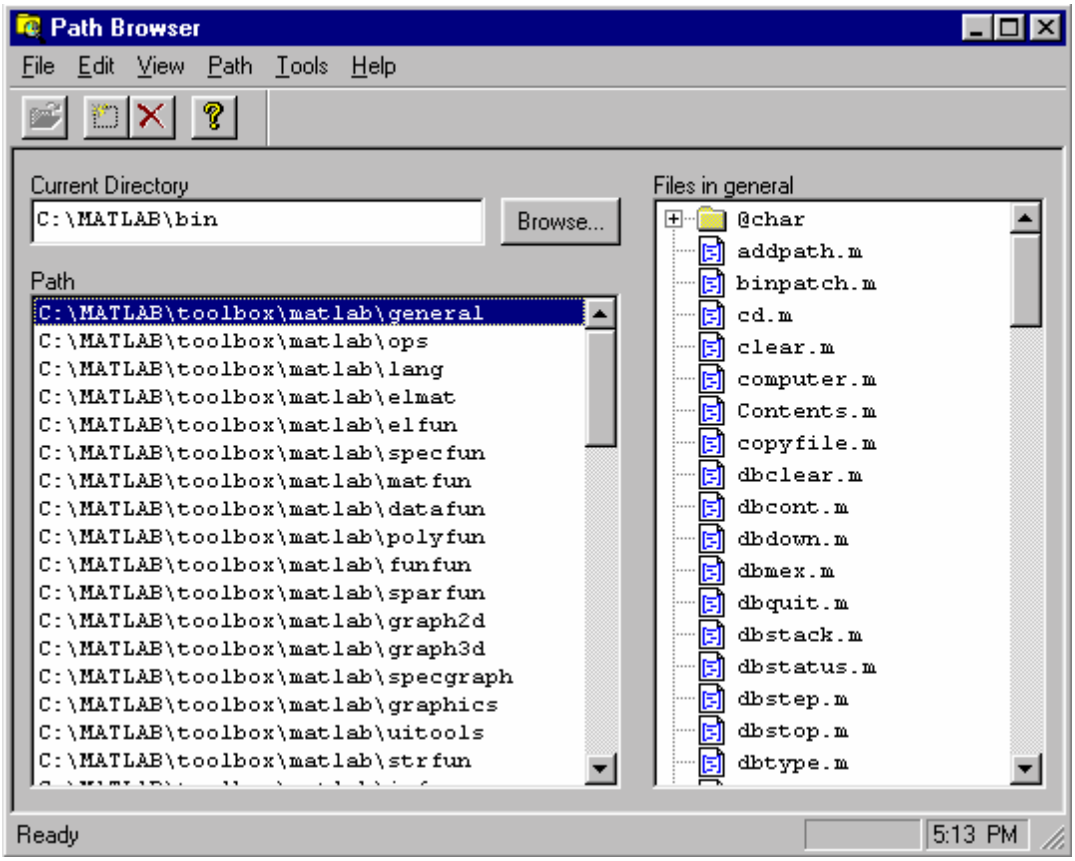
В процеса на работа може да се внесат изменения в списъка на пътя за достъп, използвайки следните функции:

- ***path*** извежда на екрана списък на пътя за достъп;
- ***path(s)*** заменя съществуващ списък със списък *s*;
- ***addpath /home/lib path(path, '/home/ lib')*** добавят нова папка в списъка на пътя за достъп;
- ***mpath/home/lib*** изтрива пътя */home/ lib* от списъка.

Списъкът на пътя за достъп, който се използва по подразбиране е зададен във файла *pathdef.m*, разположен в папката *local*. Този файл се изпълнява при всяко стартиране на системата *MATLAB*.

**Средство за разглеждане на пътя за достъп**

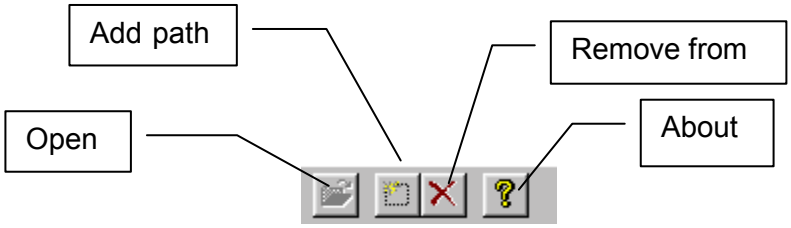
*Path Browser* (фиг.1.9). е средство за визуално разглеждане на пътя за достъп. Чрез него се извършва разглеждане и модифициране на пътя. Стартира се с опцията ***Seth Path*** от менюто ***File*** или чрез бутона от панела с бутоните.




*Фиг.1.9. Прозорец на Path Browser*

Прозорецът на *Path Browser* включва:

- поле *Current Directory* с бутон *Browse*, предназначено за промяна на текущата папка;
- поле *Path*- съдържа списък на пътищата за достъп;
- поле *Files in* < име на папка маркирана в полето *Path* >- съдържа списък на файлове и подпапките от типа *private,@*;
- панел с бутони (фиг. 1.10).




*Фиг.1.10. Панел с бутони*

Менюта на ***Path Browser***:

| **Име на меню** | **Име на подменю** | **Пояснение** |
|---|---|---|
| *File* | *Open* | Отваряне на файл |
| | *Save Path* | Запис на пътя за достъп |
| | *Exit Path Browser* | Изход от приложението |
| *Edit* | *Undo* | Отменя последната операция |
| | *Redo* | Възстановява отменената операция |
| *View* | *Toolbar* | Показва панела с бутони |
| | *Status Bar* | Показва лентата на състоянието |
| | *Editor Debugger* | Показва редактора |
| | *Path Browser* | Показва *Path Browser* |
| | *Workspace Browser* | Показва *Workspace Browser* |
| *Path* | *Add to Path* | Добавя папка към пътя за достъп |
| | *Remove from Path* | Изтрива папка от пътя за достъп |
| | *Refresh* | Обновява пътя за достъп |
| | *Restore Defaults* | Възстановява настройките, приети по подразбиране |
| *Tools* | *Run* | Изпълнява *M*- файл |
| | *Customize* | Избор на настройка |
| | *Options* | Задава опции |
| | *Font* | Задава шрифта |
| *Help* | *About Matlab Path Browser* | Информация за *Path Bro*wser |

За преместване на папка в друга позиция в областта *Path* е необходимо да се натисне левия бутон на мишката и да се премести на нужното място.

Ако изменението на списъка на пътя за достъп се изпълнява в командния прозорец, то отражението на тези изменения в средствата за разглеждане на *Path Browser* ще стане възможно чрез използване на командата *Refresh* от менюто *Path*.

Всички изменения, които се внасят в списъка на пътя за достъп, действат само по време на работа. За да е постоянно изменението трябва да се използва командата *Save Path* от менюто *File*.

**Интерактивен достъп до справочната информация**

Начините за получаване на помощна информация относно функциите на системата *MATLAB* са следните:

- команда ***help***;
- команда ***lookfor***;
- меню ***Help***;
- преглеждане и печат на страниците на документацията;
- обръщение към *Web*- сървъра на фирмата *The Math Works*.

Най- бързият начин за получаване на помощ е командата ***help*** <име на *M*- функция>. Съответстващата информация се появява непосредствено в командния прозорец.

Например, командата *help for* извежда в командния прозорец информация за оператора *for*.

Следва да се обърне внимание, че в текста на интерактивните справки се използват главни букви за изписване на имената на функции и променливи. Когато се въвеждат имена на функции от потребителя те се пишат с малки букви.

Всички функции в системата *MATLAB* (те са повече от 800) са организирани в логически групи и структурата на папките е основана на тази организация. Например, всички функции на линейната алгебра се намират в папката *matfun*. Всички функции в нея могат да се разпечатат с кратки пояснения, ако се използва командата *help matfun*.

Командата ***lookfor***

Тази команда позволява търсене на *M*-функции по ключови думи. Анализира се първият ред от коментара и ако в него е срещната ключовата дума, то той се извежда на екрана,

Добавянето на опция - *all* в командата ***lookfor*** <шаблон> - *all* позволява да се преглеждат всички редове на коментарите на *M*-функциите, а не само първия ред.

**Помощна информация (*Help Window*).**

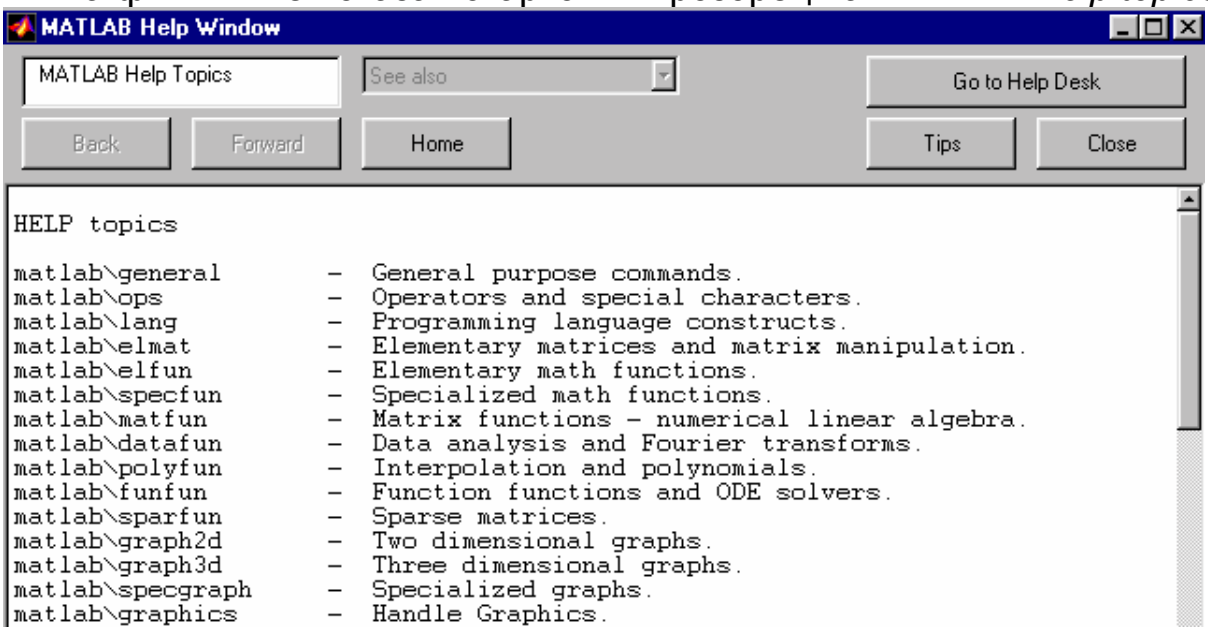
Достъпът до помощната информация става по няколко начина: като опция *Help Window* на менюто *Help*; чрез избор на бутон ***?*** от панела с бутони или с помощта на командата ***helpwin***.

Предлагани възможности на меню ***Help***, което не е показано на фигура.

| | |
|---|---|
| *Help Window* | Прозорец за справки |
| *Help Tips* | Помощен прозорец |
| *Help Desk (HTML)* | Достъп до справочната система на твърдия диск или на *CD-ROM* в *HTML* формат |

| | |
|---|---|
| *Examples and Demos* | Достъп до демонстрационните програми |
| *About MATLAB* | Информация за инсталираната версия |
| *Subscibe (HTML)* | Информация за връзка с фирмата производител |

На фиг. 1.11 е показан стартовия прозорец на *MATLAB Help topics*.




*Фиг. 1.11.*

Чрез маркиране на произволен ред от списъка и двукратно щракване на левия бутон на мишката може да се премине към списъка на съответния раздел. Това действие е аналогично на командата ***helpwin*** <име на раздел>.

Падащият списък *See also* в този случай е неактивен. Бутоните ***Back***, ***Forward***, ***Home*** позволяват да се преминава от един активен прозорец към друг или произволно връщане към началния прозорец.

В десния ъгъл на прозореца са разположени 3 бутона ***Go to Help Desk***, ***Tips***, ***Close***.

С бутона ***Tips*** падащото меню се активира и позволява да се изпълни редица допълнителни справочни команди: *More help info help (HTML)*, *lookfor*, *which*, *demo*, *general*.

**Опция *Help Desk***

Тази опция позволява достъп до голям обем справочна информация записана на твърдия диск или на *CD-ROM*.

Много от документите използват езика за хипертекстови връзки *HTML* (*Hyper Text Markup Language*) и са достъпни за преглеждане с помощта на средства от тип *Netscape Navigator* или *Microsoft Explorer*.

Всички оператори и функции на системата *MATLAB* са описани във формата *HTML* и съдържат много примери. Версията на *HTML* е

достъпна за различни типове документи и описания. Реализираната търсеща система позволява изпълнение на желаните запитвания.

**Опция *About MATLAB***

Извежда на екрана информация за версията на системата и лицензните права.

**Преглеждане и разпечатка на документацията.** Версията на справочната документация е достъпна за разглеждане и разпечатка във формат *PDF* (*Portable Document Format*). Този формат е достъпен със средствата *на Adobe's Acrobat*. Те позволяват преглед на текста във формата на печатна страница, с пълния набор шрифтове, графики и изображения.

***Web*- сървър на фирмата *Math Works*.** Ако вашия компютър е включен към *Internet*, то може да се направи обръщение към *www* - сървъра на фирмата за допълнителна информация.

### 2.3. Режими на работа

*MATLAB* позволява два режима на работа: директен и програмен. При първият режим на работа командите се въвеждат последователно и резултата от изпълнението следва непосредствено. Това превръща *MATLAB* в извънредно мощен калкулатор, който извършва не само обикновените изчисления но и операции с вектори, матрици и комплексни числа. Позволява визуализирането на дву- и тримерни графики.

Програмното решаване на задачи е чрез последователно интер-претиране на команди (оператори) от езика на *MATLAB*. Тези команди предварително се оформят като текстови файлове в *ASCII* формат, известни като *M*-файлове поради разширението си '*xxx.m*' и могат да се изпълняват многократно.

# 3. ОСНОВНИ ТИПОВЕ ДАННИ И ОПЕРАЦИИ ВЪРХУ ТЯХ

## 3.1. Специални символи

*MATLAB* работи със следните специални символи:

***[ ]*** – ограничители за вектори и матрици. Указване на последователността на изходните параметри при обръщения към функции, връщащи повече от един параметър;

***( )*** – скоби в аритметичните изрази. Указване на индексите на елементите при работа с вектори и матрици. Указване на последователността на входните параметри на функции;

**=** - оператор за присвояване;

***'*** - оператор за транспониране;

***.*** – десетична точка и указател за поелементно изпълняване на операция над масив от данни;

***…*** - признак за продължение на командата на следващия команден ред;

***;*** - забрана на визуализацията на отговора, служи за разделяне на редовете на матрица и за разделител между няколко оператора при записът им на един ред;

***:*** - за формиране на вектори, за отделяне на редове, стълбове, подблокове на матрица и елемент от **for** цикъл;

***%*** - начало на коментарен текст;

***!*** – префикс към команда на операционната система.

## 3.2. Константи

В *MATLAB* са дефинирани следните константи:

***pi*** – числото $\pi$=3.14159265…;

***realmin*** – най-малкото нормализирано положително число във формат на плаваща десетична точка;

***realmax*** – най-голямото число във формат на плаваща десетична точка;

***i*** или ***j*** – имагинерна единица ($i = j = \sqrt{-1}$).

## 3.3. Променливи

Променливите са данни, чиято стойност може да се променя по време на работа.

В *MATLAB* могат да се задават променливи по следния начин:

***име на променлива=Израз [;]***.

Името може да е с произволна дължина, но системата отчита само първите 31 символа. Препоръчително е да се задават съдържателни имена.

По подразбиране *MATLAB* различава малки и големи букви. Това може да бъде променено със следната команда: ***casesen on/ff***.

Променливите могат да са обикновени и индексни (елементи на вектори и матрици).

Символните променливи се задават като символния низ е в апострофи.

**Пример:**

»Tekst='Това е символна променлива'

*MATLAB* работи със следните системни променливи:

***inf*** – системна променлива със стойност машинна безкрайност;

***eps*** – относителна точност на операциите с плаваща десетична точка;

***NaN*** – (*Not a Number* - не е число). Означава неопределеност, например от вида 0/0 или *Inf/Inf*;

***ans***- променлива съхраняваща резултата от последната операция.

Променливите ***ans*** и **inf** могат да се използват в математическите изрази.

За изчисляване на математически изрази е достатъчно те да се изпишат по общоприетите правила и при завършване на въвеждането да се натисне клавиша *Enter*. Резултатът се присвоява на системната променлива **ans** (ANSwer) и се изобразява на екрана.

**Примери:**

```
»2+4-5
ans =
      1
»2*sin(0.3)
ans =
      0.5910
»9*(1-exp(-2))
ans =
      7.7820
```

*MATLAB* работи както с реални, така и с комплексни числа от вида: z=Re(z)+i*Im(z),

където *i* или *j* – имагинерна единица ($i = j = \sqrt{-1}$);

*Re(z)* – реална част на комплексното число ***z***;

*Im(z)* – имагинерната част.

**Пример:**

```
»i^2
ans =
  -1.0000 + 0.0000i
```

Основните функции за работа с комплексни числа:

***real(z)*** - връща като резултат реалната част на комплексното число *z*;

***imag(z)*** - връща като резултат имагинерна част на z;

***angle(z)*** - изчислява фазов ъгъл;

***abs(z)*** - изчислява абсолютна стойност или модул на комплексно число;

***sign(z)*** - връща число с реална и имагинерна части равни съответно на ***real(z)/abs(z)*** и ***imag(z)/abs(z)***;

***conj(z)*** - изчислява комплексно спрегнато число.

**Примери:**

```
»z=3+4i
»angle(z)
ans =
   0.9273
»abs(z)
ans =
  5
»sign(z)
ans =
   0.6000 + 0.8000i
»conj(z)
ans =
    3.0000 - 4.0000i
```

Данните могат да се съхраняват върху магнитни носители (твърд диск, дискети) и да се ползват повторно при следващи сеанси с *MATLAB*. Обикновено променливите в работното пространство се записват/ възстановяват чрез ***save/load*** във/от файл със ‘име на файл’ и подразбиращо се разширение 'xxx.*mat'*. Това е специфичен файл, който може да се използва в този вид само в *MATLAB*.

**Примери:**

***save/load*** 'име на файл' – действие с всички променливи от работното пространство;

***save/load*** 'име на файл' *Y, Z* – действие само с променливите *Y* и *Z*.

Променливите от работното пространство могат да се запишат/ възстановят и с командите ***Save Workspace As/Load Workspace*** от менюто ***File***.

### 3.4. Вектори

*MATLAB* е система специално предназначена за провеждане на сложни изчисления с вектори и матрици. По подразбиране тя приема, че всяка зададена променлива е вектор или матрица. Всичко се определя от конкретната стойност на променливата. Например, ако е зададено q=1, това означава, че *q* е вектор с един елемент със стойност 1.

#### 3.4.1. Генериране на вектори

Задаването на вектор от няколко елемента става по следния начин: ***име на вектора (променлива)=[стойност$_1$, стойност$_2$, ... стойност$_n$]***.

За разделител вместо запетая може да се използва и интервал.

**Пример:**

Задаване на вектор *V* притежаващ три елемента със стойности 1, 2 и 3.

```
»V=[1 2 3]
V =
    1     2     3
```

Освен чрез изброяване на елементите може да бъде използвана и конструкцията: ***име на вектора (променлива)=начална стойност:стъпка: крайна стойност***.

**Пример:**

```
»Y = 0:pi/4:pi
```

### 3.4.2. Функции за генериране на вектори:

***linspace(m,n,p)*** - генерира вектор-ред, съдържащ ***p*** на брой, линейно (равномерно) разпределени числа между ***m*** и ***n***.

**Пример:**

```
»k = linspace(-pi,pi,4)
k =
   -3.1416   -1.0472    1.0472    3.1416
```

***logspace(m,n,p)*** - генерира вектор-ред, съдържащ ***p*** на брой, логаритмично разпределени числа между $10^m$ и $10^n$.

**Пример:**

```
»l = logspace(-2,2,100)
```

Друг подход за въвеждане на данни в *MATLAB* е с вградения редактор:

Стъпка 1. Създава се файл A.m от вида: a=[1; 2; 3; 4; 5; 6; 7; 8; 9; 10;]

Стъпка 2. Файлът *A.m* се активира в средата на *MATLAB* с: *A*. В резултат се формира и извежда променливата

```
a=
   1
   2
   …
  10
```

### 3.4.3. Полиноми

Системата *MATLAB* предоставя големи възможности за работа с полиноми от вида: $P(x)=a_nx^n+\ldots+a_2x^2+a_1x+a_0$.

Полиномът се задава с вектор, съхраняващ коефициентите от $a_n$ до $a_0$ по намаляващата степен на аргумента: p=[$a_n$ … $a_2$ $a_1$ $a_0$].

За намирането на корените на полином служи функцията: ***roots(a)***, където в ***a*** са коефициентите (по намаляващата степен на аргумента).

**Пример:**

Да се намерят корените на полинома:

pol(x)=$x^5+8x^4+31x^3+80x^2+94x+20$.

```
»pol=[1 8 31 80 94 20]
»roots(pol)
ans =
   -1.0000 + 3.0000i
   -1.0000 - 3.0000i
   -3.7321
   -2.0000
   -0.2679
```

***poly(a)*** - резултатът от действието на оператора е вектор, чиито елементи са коефициентите на полином с корени елементите на вектора **a**. За вектори функциите ***roots*** и ***poly*** са обратни една спрямо друга.

**Пример:**

```
»poly(ans)
ans =
   1.0000    8.0000   31.0000   80.0000   94.0000   20.0000
```

Функцията Y=***polyval(p,x)***, изчислява стойностите на полинома ***p(x)*** в точките зададени в масива ***x***, където p=[$p_1$ $p_2$ … $p_n$ $p_{n+1}$] е вектор с коефициентите на полинома $p(x)=p_1x^n+p_2x^{n-1}+\ldots+p_nx+p_{n+1}$.

**Пример:**

Изчисляваме стойността на полинома $p(x)=3x^2+2x+1$ в точка x=5.

```
»p=[ 3 2 1]
»y=polyval(p,5)
y=86
```

***conv(a,b)*** - произведение на полиноми. Коефициентите са зададе-ни във векторите ***a*** и ***b*.**

***[q, r] = deconv(a,b)*** - в ***q*** са коефициентите на частното на **a** и **b,** а в **r** - коефициентите на остатъка.

**Примери:**

```
»a = [1   2   3   4]
»b = [10   20   30]
произведението е
»c = conv(a,b)
c =
   10    40    100    160    170    120
»[q,r] = deconv(c,a)
q =
   10    20    30
```

```
r =
   0     0     0     0     0     0
```

### 3.5. Матрици

### 3.5.1. Въвеждане на матрици

При въвеждане на елементите на матрица с [ ***;*** ] се разделят отделните редове.

**Пример:**

Задаване на квадратната матрица $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$.

```
»M=[1 2 3;4 5 6;7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
```

Матрицата може да се зададе и по редове, като за край на всеки ред се натисне бутон *ENTER*:

```
»M=[1 2 3
4 5 6
7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
```

Този начин може да е по-удобен при задаване на матрици с голяма размерност.

Елементите на матриците могат да бъдат всички допустими за *MATLAB* изрази.

**Пример:**

```
»x = [-1.3, sqrt(3); (1+2+3)*4/5, log10(33)]
x =
   -1.3000    1.7321
    4.8000    1.5185
```

Възможно е и задаването и матрици с комплексни числа.

**Пример:**

```
»ZM=[1 2;3 4]+i*[5 6;7 8]
ZM =
     1.0000 + 5.0000i   2.0000 + 6.0000i
     3.0000 + 7.0000i   4.0000 + 8.0000i
или
»ZM=[1+5*i 2+6*i; 3+7*i 4+8*i]
ZM =
  1.0000 + 5.0000i   2.0000 + 6.0000i
```

```
  3.0000 + 7.0000i   4.0000 + 8.0000i
```

Матриците могат да се разширяват.

**Пример:**

Разширяване на матрицата M с вектора V.

```
»M=[M;V]
M =
     1     2     3
     4     5     6
     7     8     9
     1     2     3
```

Команди ***size*** и ***length***

***size(M)*** - връща размерността на ***M***.

***length(M)*** **-** връща по-голямата размерност на ***M*** или дължина, ако ***M*** е вектор.

**Основни функции за генериране на матрици:**

***ones(M,N)*** - генерира матрица с единични елементи;

***zeros(M,N)*** - генерира матрица на която всички елементи са равни на нула;

***rand(M,N)*** - генерира матрица с елементи, които имат случайна стойност в диапазона от 0.0 до 1.0.

***eye(M,N)*** - генерира матрица с единични диагонални елементи. ***M*** задава броя на редовете на матрицата, а ***N*** – броя на стълбовете. За квадратна матрица е достатъчно задаването само на един аргумент.

**Примери:**

```
»ones(3,5)
ans =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
»zeros(2,3)
»A=rand(3)
»B=rand(3,5);
»C = rand(size(B))
»eye(4)
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

*MATLAB* предоставя две възможности за минимизиране на използваната оперативна памет за съхраняване на вектори и матрици:

- чрез преобразуване на съществуваща матрица към нулева размерност ***име на матрица=[ ]***. Матрицата се съхранява и в последствие може да се разширява и използва;
- изтриване на матрицата с ***clear име на матрица***. Винаги когато е необходимо матрица със същото име може отново да се дефинира.

### 3.5.2. Индексиране

За указване на отделен елемент на вектор или матрица се използват записи от вида: ***V(i)*** или ***M(i,j)***.

**Пример:**

Стойността на елемента от втори ред втора колона на матрицата M е 5.

```
»M(2,2)
ans =
      5
```

Знакът двоеточие връща последователни редове или стълбове. Когато се приложи върху цялата матрица резултатът е вектор-стълб от всички елементи на изходната матрица.

**Примери:**

```
»A=[1 2 3;4 5 6]
A =
   1   2   3
   4   5   6
»B=A(:)
B =
   1
   4
   2
   5
   3
   6
»A(:)=10:15
A =
  10  12  14
  11  13  15
```

В тези примери се създаде нов вектор *B* и се промени матрицата *A*.

Следващият пример показва създаването на подматрица.

**Пример:**

```
 »A=[1 2 3 4 5;6 7 8 9 10;11 12 13 14 15;16 17 18 19 20]
A =
```

```
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
    16    17    18    19    20
»A(2:4,2:5)
ans =
     7     8     9    10
    12    13    14    15
    17    18    19    20
```

### 3.5.3. Операции и манипулации с матрици

#### 3.5.3.1 Аритметични оператори

В изчислителните изрази могат да се използват следните оператори: **+** (събиране); **-** (изваждане); ***** (умножение); **/** (дясно деление); **** (ляво деление); **^** (степенуване); **'** (транспониране), както и достъпните системни функции (алгебрични, тригонометрични, хиперболични и др.).

**Пример:**

X = A/B е решение на X*B = A.

X = A\B е решение на A*X = B.

При транспонирането на вектор ред - става вектор стълб и обратно.

**Пример:**

```
»V=[1 2 3]
V =
     1     2     3
»V'
ans =
       1
       2
       3
```

Ако пред оператора е поставена точка, това означава, че операцията ще се извършва върху всеки елемент поотделно.

**Примери:**

```
» M=[1 2 3;4 5 6;7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
» N=M;
» M*N
ans =
    30    36    42
    66    81    96
```

```
   102   126   150

» M.*N
ans =
    1     4     9
   16    25    36
   49    64    81
```

**Аритметичните операции са със следния приоритет:**

**Първо ниво**: поелементно транспониране (**.'**), поелементно степенуване (**.^**) и матрично степенуване (**^**).

**Второ ниво**: унарно събиране (**+**) и унарно изваждане (**-**).

**Трето ниво**: умножение на масиви (**.***), дясно деление (*./*), ляво деление (**.**\), матрично умножение (***), решаване на система линейни уравнения – операция (*/*) и операция (****).

**Четвърто ниво**: събиране (**+**) и изваждане (**-**).

**Пето ниво**: оператор за формиране на масиви (*:*).

На всяко ниво операторите имат равен приоритет и се изчисляват в реда на следване от ляво на дясно. Зададеният по подразбиране ред на изчисляване може да бъде променен с помощта на кръгли скоби.

**Примери:**

Зададени са следните два вектора

```
»A=[3 9 5]
»B=[2 1 5]
```

Резултатите от изпълнението на операторите

```
»C=A./B.^2
C =
   0.7500    9.0000    0.2000
»C=(A./B).^2
C =
   2.2500   81.0000    1.0000
```

са съвършено различни

Ако един от операндите е скалар, то той се разширява до размера на втория операнд и дадената операция се прилага поелементно.

### 3.5.3.2 Оператори за отношение

В *MATLAB* са определени следните 6 оператори за отношение:

**<** по- малко;
**<=** по- малко или равно;
**>** по- голямо;
**>=** по- голямо или равно;
**==** равно;
**~=** различно.

Операторите за отношение се прилагат поелементно при сравняването на два масива с еднаква размерност. В позициите където съотношението е истина се получава стойност 1, а при неистина – 0.

Операторите за отношение се прилагат за промяна на последователността на изпълнение на операторите в програмата. Те винаги се изпълняват поелементно.

**Пример:**

Сравняване на два масива:

```
»A=[2 7 6; 9 0 -1; 3 0.5 6];
»B=[8 0.2 0; -3 2 5; 4 -1 7];
»A<B
ans =
    1    0    0
    0    1    1
    1    0    1
```

Получената матрица показва позициите, където елемента на *A* е по-малък от елемента на *B*.

При изчисляване на аритметични изрази операторите за отношение имат по-нисък приоритет, отколкото аритметичните, но по-висок от логическите оператори.

За проверка, дали даден масив е празен се използва функцията ***isempty(A)***.

### 3.5.3.3 Логически оператори

Логическите оператори в *MATLAB* са:

***&*** И;

**|** ИЛИ;

**~** НЕ.

В допълнение на тези оператори в папката ***bitfun*** се съдържат редица функции, които изпълняват поразрядни логически операции.

### 3.5.3.4 Матрични манипулации

***rot90(M)*** - завърта матрицата на 90 градуса.

***fliplr (M)*** - дава хоризонтален огледален образ на матрицата.

***flipud(M)*** - дава вертикален огледален образ на матрицата.

***triu(M)*** - формира горна триъгълна матрица от входната.

***tril(M)*** - формира долна триъгълна матрица от входната.

***reshape(A,m,n)*** - трансформира всяка матрица ***A*** в нова с размерност ***(m,n)*** като елементите се вземат по стълбове (Ако ***A*** е с размерност ***(v,w)***, очевидно трябва ***vw=mn***).

**Примери:**

```
»x=[1 2 3 4; 1 2 3 4; 1 2 3 4; 1 2 3 4];
»rot90(x)
ans =
    4    4    4    4
```

```
     3     3     3     3
     2     2     2     2
     1     1     1     1
»fliplr(x)
ans =
     4     3     2     1
     4     3     2     1
     4     3     2     1
     4     3     2     1
```

### 3.5.4. Матрични функции

Всяка тригонометрична или елементарна математична функция може да се приложи върху квадратна матрица (а не върху всеки елемент поотделно), по следния начин:

***funm(X,'function'),*** където *X* е матрицата, а във ***function*** се указва желаната функция *sin, log10, exp*, ....

**Други елементарни функции връщат съответно:**

***poly(M)*** - характеристичен полином (вектор - ред) на матрица, а ако аргументът е вектор, връща коефициентите на уравнение, чиито корени са входните елементи;

***det(M)*** - детерминантата на аргумента;

***trace(M)*** - сума на елементите от главния диагонал на аргумента (следа на матрица);

***rank(M)*** - ранг на аргумента;

***eig(M)*** - характеристични числа (собствени стойности) и собствени вектори .

**Примери:**

```
» M=[1 2;3 4];
» P=poly(M)
P =
    1.0000   -5.0000   -2.0000
» D=det(M)
D =
   -2
» T=trace(M)
T =
    5
» R=rank(M)
R =
    2
» E=eig(M)
```

```
E =
  -0.3723
   5.3723
```

### 3.6. Обработка на данни

Аргументът за следващите функции може да бъде вектор (ред или стълб) или матрица (масив с данни). При втория случай оценките се изчисляват по стълбове.

Всеки от операторите връща съответно:

***max*** - максимална стойност;
***min*** - минимална стойност;
***mean*** - средно-аритметична стойност;
***median*** - медиана;
***std*** - средно квадратично отклонение;
***sort*** - сортировка на елементите по големина;
***sum*** - сума на елементите;
***prod*** - произведение на елементите;
***cumsum*** - кумулативна сума на елементите;
***cumprod*** - кумулативно произведение на елементите;
***diff*** - разлики между всеки елемент и предходния му;
***corrcoeff*** - коефициенти на корелация между елементите;
***cov*** - ковариационна матрица;

**Примери:**

```
»xydata=[2 4 6 8 10;5.5 6.3 6.8 8 8.6]
xydata =
   2.0000    4.0000    6.0000    8.0000   10.0000
   5.5000    6.3000    6.8000    8.0000    8.6000
» max(xydata)
ans =
   5.5000    6.3000    6.8000    8.0000   10.0000
» min(xydata)
ans =
   2.0000    4.0000    6.0000    8.0000    8.6000
» mean(xydata)
ans =
   3.7500    5.1500    6.4000    8.0000    9.3000
» std(xydata)
ans =
   2.4749    1.6263    0.5657         0    0.9899
```

## 4. ПРОГРАМЕН РЕЖИМ НА РАБОТА

Програмното решаване на задачи е чрез последователно интерпретиране на команди (оператори) от езика на *MATLAB*. Тези команди могат предварително да се оформят като текстови файлове в *ASCII* формат, известни като *M*-файлове поради разширението си '*xxx.m*' и да се изпълняват многократно. Най-елементарен вид на *M*-файл представлява т. нар. описателен файл: текстов файл с последователно записани и изпълнявани команди, чиито променливи остават глобални за цялото работно пространство на *MATLAB*. Те са най-простите *M*-файлове, които поддържа системата *MATLAB* и нямат входни и изходни аргументи. Основното им предназначение е да автоматизират многократно изпълнявани изчисления. Те оперират с данни от работната област и могат да генерират нови данни за следваща обработка в този файл. Използваните данни в тези файлове се съхраняват в работната област и са достъпни и след приключване на работа с описателния файл за последваща работа.

Съдържанието на помощните пакети, '*toolbox*'-овете, се определя от т. нар. функционални файлове, които съхраняват програми за многократно ползване. Първият ред в тях започва с *function*, а чрез списък от аргументи (входни и изходни) се осъществява връзка между локалното пространство на файла и работното пространство на задачата, която активира файла.

По този начин всички процедури (функции) включват две операции:

- Създаване на *M*-файл с помощта на текстов редактор
  function c=myfile(a,b)
  c=sqrt((a.^2+(b.^2))
- извикване (стартиране) на M-файла от командния ред или от друг M- файл

```
»a=7.5
»b=3.342
»c=myfile(a,b)
c=8.2109
```

Основни разлики между двата типа *M*-файлове:

| **Описателен *M*-файл** | ***M*-функция** |
|---|---|
| Не използват входни и изходни аргументи | Използват входни и изходни параметри |
| Оперират с данни от работната област | По подразбиране вътрешните променливи се явяват локални по отношение на функцията |
| Основното му предназначение е за автоматизиране на последователността от стъпки, които се изпълняват многократно | Служи за разширяване на възможностите на езика *MATLAB* (изграждане на библиотечни функции, пакети приложни програми и др.) |

**Структура на *M*-функция:**

***function*** Име на променлива=Име на функцията(Списък на аргументи)

% Коментар, отпечатва се при използване на командата help

Оператори % Коментар

.....

Име на променливата=Израз

**Пример:**

| | |
|---|---|
| Дефиниране на функцията | function y=average (x) |
| Първи коментарен ред | % AVERAGE Изчислява средна стойност на елементите на вектор |
| Коментар | % Извикване average (x), където x е вектор. Изчислява средната стойност на елементите на вектор.<br>% Ако входният аргумент не е вектор се генерира грешка. |
| Тяло на функцията | [m,n]=size(x);<br>if (~((m==1)\|(n==1))\|(m==1&n==1))<br>error('Входният масив трябва да е вектор')<br>end<br>y=sum(x)/length(x); % Изчисляване на средната стойност |

- **Дефиниране на функцията**. В този ред се задава името, броят и последователността на следване на входните и изходни аргументи.

**Пример:**

function y=average (x),

където ***function*** – ключова дума, определяща *M*-функция;

*y* – изходен аргумент;

*average* – име на функцията;

*x* – входен аргумент.

Ако функцията притежава повече от един изходен аргумент - те се записват в квадратни скоби, а входните аргументи се записват в кръгли скоби. За разделители на аргументите във входните и изходните списъци се използва запетая.

**Пример:**

function [x, y, z]=sphere (theta, phi, rho)

- **Първи коментарен ред** определя предназначението на функцията. Той се извежда на екрана с командите ***lookfor***;
- **Коментариите** се извеждат на екрана съвместно с първия коментарен ред с командата ***help*** <име на функцията>;

- **Тяло на функцията** – съдържа програмният код, реализиращ изчисленията и присвояване на стойностите на изходните аргументи.

**Име на *M*-функцията**. В системата *MATLAB* за имената на *M*-функциите са в сила същите ограничения, както и за имената на променливите – дължината им не трябва да надвишава 31 символа. По-точно името може да е с произволна дължина, но системата отчита само първите 31 символа. Имената на М-функциите трябва да започват с буква, а останалите символи могат да са произволна комбинация от букви, цифри и знак за подчертаване.

Препоръчва се името на файла съдържащ М-функция да съвпада с името на функцията следвано от разширение xxx.m.

**Пример**:
average.m
Ако имената са различни се използва името на файла.

### 4.1. Типове променливи

**Локални и глобални променливи**. Използването на променливи в *M*-файла не се различава от използването на променливи от командния ред, а именно:

- Не е необходимо деклариране на използваните променливи;
- Имената на променливите започват с буква, следвана от произволен брой букви, цифри и знак за подчертаване. Вземат се в предвид само първите 31 символа.

Всяка *M*-функция притежава свои локални променливи. Ако е необходимо дадена променлива може да стане глобална с командата ***global***. По този начин тя е достъпна в произволен брой функции и работната област (естествено, че трябва да е обявена във всяка от тях като глобална).

За работа с глобални променливи е необходимо:

- Да се обяви променливата като глобална във всяка М-функция, където се използва. Обявяването на променлива за работната област става от командния ред.
- Във всяка функция командата global трябва да предшества първото използване на променливата.

Езикът на *MATLAB* съдържа голям брой основни команди, чрез които се програмират допълнителните функции (в настоящото пособие са разгледани няколко):

### 4.2. Управляващи оператори

Всички езици за програмиране поддържат определено множество от оператори, наречени управляващи оператори. Чрез тях се задава реда на изпълнение на операторите, формиращи програмите.

### 4.2.1. Въвеждане на информация от потребителя

За въвеждане на информация от потребителя се използва оператора ***input***.

Синтаксис:

m = ***input('текст')***,

m = ***input('текст','s')***.

При първия случай изпълнението се прекъсва, на екрана се появява съобщението 'текст', изчаква се въвеждане на стойност от клавиатурата, след натискане на *Enter* стойността се присвоява на променливата m и изпълнението на програмата продължава.

При втория запис указваме, че ***m*** е променлива тип низ.

***disp(z)*** -отпечатва на екрана променливата *z*, която може да бъде и тип низ (т.е. може да се използува за извеждане на съобщения), без да се изведе нейното име.

***error('текст')*** - прекъсва изпълнението на програмата и на екрана се отпечатва error +текста**.**

***pause*** - при срещане на тази команда в програмата, нейното изпълнение се прекъсва до натискане на произволен клавиш.

### 4.2.2. Оператор за цикъл *for*

Операторът ***for … end*** включва променлива брояч. Чрез този оператор може да се контролира точно колко пъти да се изпълнят командите в тялото.

Синтаксис:

***for*** <променлива-брояч> = <начална стойност> : <нарастване (стъпка)>: <крайна стойност>

инструкции

***end***

Изпълнява се инструкцията (инструкциите) определения брой пъти. По подразбиране стъпката е 1. Може да се задава произволна стъпка, в това число и отрицателна. За положителните индекси изпълнението завършва, когато стойността на променливата брояч превиши крайната стойност.

**Примери:**

Следващият цикъл ще се изпълни 5 пъти

```
for I=2:6
  x(i)=2*x(i-1);
end
```

Допустими са и вложени цикли

```
for I=1:m
  for j=1:n
    a(i,j)=1/(i+j-1)
  end
end
```

Следва да се обърне внимание, че трябва да се избягва употребата на цикли там където *MATLAB* допуска алтернативни операции във векторна или матрична форма.

**Пример:**

```
i=0
for t=0:0.001:1;
 i=i+1;
 y(i)=sin(t);
end
```

Същата задача се решава със следните два реда и се изпълнява многократно по-бързо

```
t=0:0.001:1;
y=sin(t);
```

### 4.2.3. Оператор за цикъл *while*

Позволява повторение на едно или няколко действия, докато определено условие е вярно.

Синтаксисът на цикъла **while** е следния:

***while*** <логически израз (логическо условие)>
  инструкции
***end***

Логическият израз е от вида: израз <оператор за отношение> израз. Оператор за отношение: ==, <=, >=, <, >, ~

Тъй като най-напред се проверява условието, то тялото на цикъла **while** може да не бъде изпълнено нито веднъж. Така се получава, ако още при първото изчисление на условието резултатът е **false**.

**Пример:**

Да се изчисли първото цяло число n, чийто факториел се състои от 100 значещи цифри.

```
n = 1;
while prod(1:n) < 1e100
    n = n+1;
end
```

### 4.2.4. Оператор за прекъсване на изпълнението на цикъл *break*

При срещане на ***break*** в цикъл ***for*** или ***while***, той се напуска без изпълнение на командите до края на тялото на цикъла.

### 4.2.5. Условен оператор *if … else…elseif…end*

Операторът ***if*** се използва за условно изпълнение на команди, т. е. изпълнението на тези команди зависи от оценката на израз.

Синтаксис:

***if*** логически израз
 инструкции
***end***

***if*** логически израз
инструкции
***else***
инструкции
***end***

***if*** логически израз
инструкции
***elseif***
логически израз
инструкции
***else***
инструкции
***end***

Ако логическия израз е истина се изпълняват всички инструкции между ***if*** и ***end***. Ако е неистина се пропускат всички инструкции между ***if*** и ***end***.

Операторите ***if…else…end*** и ***if…elseif…end*** създават допълнителни разклонения в тялото на оператора ***if***.

- Операторът ***else*** не съдържа логическо условие. Инструкциите свързани с него се изпълняват ако предшестващия оператор ***if*** (и възможно ***elseif***) е неистина.
- Операторът ***elseif*** съдържа логическо условие, което се проверява ако предшестващия оператор ***if*** (и възможно ***elseif***) е неистина. Инструкциите свързани с оператора ***elseif*** се изпълняват ако съответния логически израз е истина. Операторът ***elseif*** може многократно да се използва в условния оператор ***if***.

**Пример:**

Формиране на диагонална матрица

```
n=4
for i = 1:n
   for j = 1:n
      if i == j
         a(i,j) = 2;
      elseif abs([i j]) == 1
         a(i,j) = 1;
      else
         a(i,j) = 0;
      end
   end
end
```

### 4.2.6. Оператор switch

Чрез оператора **switch** се осъществява избор на един от няколко възможни варианта. Операторът **switch** има следния синтаксис:

**switch** *израз*
  **case** <константен израз-1> оператор-1,...,оператор-n
  **case** {константен израз-1*,*константен израз-2*,*константен израз-3*,...}* оператор-1,...,оператор-n
...
  **otherwise**
    оператор-1,...,оператор-n
**end**

Описанието на всеки вариант се състои от ключовата дума **case**. При изпълнение на оператора **switch** изразът, указан след ключовата дума **switch** (switch-израза) се изчислява и стойността му се сравнява последователно със стойностите на всеки от константните изрази на вариантите (case-изразите). При съвпадение се изпълняват операторите на съответния вариант.

След вариантите има един специален, който се означава с ключовата дума **otherwise** и няма константен израз. Този вариант се изпълнява при условие, че никой от останалите варианти не е бил изпълнен.

### 4.2.7. Създаване на меню

За създаване на меню се използва командата ***menu*.**

Синтаксис:

*k* = ***menu***('заглавие','избор 1','избор 2',…'избор n');

изчертава на екрана прозорец с бутони: ‘избор 1’, ‘избор 2’ и ‘избор n’

При натискане на първия бутон (в случая избор 1), на променливата ***к*** се присвоява стойност 1, при натискане на втория - стойност 2 и т.н. Има възможност да се създадат до 32 бутона. При черен фон на графичните прозорци, първият низ се изписва като заглавие на менюто над бутоните.

## 5. ГРАФИЧНИ ПРЕДСТАВЯНИЯ

*MATLAB* разполага с големи възможности за графично изобразяване на вектори и матрици, а също за създаване на коментари и отпечатване на графики.

В състава на системата *MATLAB* е включена мощна графична подсистема, която поддържа както средствата за визуализация на двумерна и тримерна графика, така и средствата на презентационната графика. Следва да се отбележат няколко нива за работа с графични обекти. Първо, това е интерфейс на високо ниво, включващ команди и функции, ориентирани към краен потребител и предназначени за построяване на: графики в правоъгълни и полярни координати, тримерни повърхнини и **"**линии на ниво**",** хистограми, стълбови диаграми и други специални графики. Графичните команди от високо ниво автоматично контролират мащаба, избора на цветове и други свойства на графичните обекти.

### 5.1. Разработване на двумерни графични изображения

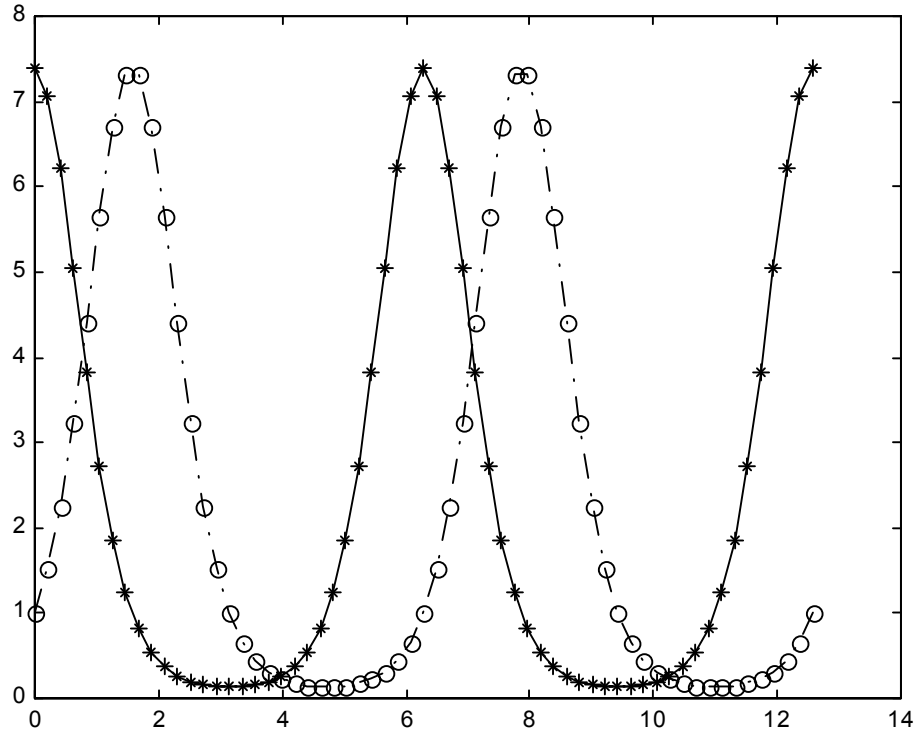
За изчертаване на графики в *MATLAB* се използуват следните команди: ***plot(x, y)***- изчертава в декартова координатна система всяка от зависимостите $y_i(x_i)$ между всеки два съответни елемента на матриците ***x*** и ***y***.

Има възможност за указване на цвета и типа на линията, с която да се изчертава графиката или типа и цвета на символите, с които да се изчертават само точките.

| **Типове линии** | **Типове символи (по-често срещани)** | **Типове цветове** |
|---|---|---|
| '**-**' – непрекъсната (по подразбиране); | '**+**' - знак плюс | '**y**' - жълт |
| '**—**' – прекъсвана; | '**o**' - кръг | '**g**' - зелен |
| '**:**' – пунктирана; | '*****' - звездичка | '**m**' - пурпурен |
| '**-.**' – осева | '**.**' - точка | '**b**' - син |
| | '**x**' - кръст | '**c**' - светло син |
| | '**d**' - квадрат | '**w**' - бял |
| | | '**r**' - червен |
| | | '**k**' – черен |

**Примери:**

```
»X = 0:pi/15:4*pi;
»Y = exp(2*cos(X));
»plot(X,Y,'b+')  % всяка точка ще бъде изчертана със син '+'.
»plot(X,Y,'r-',X,Y,'ko') % изчертава непрекъсната червена линия и с черен кръг ще се маркират точките
»X = 0:pi/15:4*pi;
»Y1 = exp(2*cos(X));
»Y2 = exp(2*sin(X));
»plot(X,Y1,'-k*',X,Y2,'-.ko')
```




***loglog(x,y)*** - аналогична на ***plot***, но мащаба по ***x*** и ***y*** е логаритмичен.

***semilogx(x,y)*** - изчертава ***y(x)*** като мащаба по абсцисата е логаритмичен.

***semilogy(x,y)*** - изчертава ***y(x)*** като мащаба по ординатата е логаритмичен.

***fill(x,y,c)*** – чертае запълнена с цвят двумерна фигура. В **c** се задава тип цвят.

**Пример:**

```
»t = (1/16:1/8:1)'*2*pi;
»x = sin(t);
»y = cos(t);
»fill(x,y,'r')
```

### 5.1.1. Специализирани двумерни графики

***polar(f,r)*** - чертае в полярни координати зависимостта между ъгъла в радиани - ***f*** и радиуса - ***r***.

***bar(y)*** - бар диаграма на стойностите на ***y***. ***bar(x,y)*** позиционира баровете ***y*** в точките определени в ***x***.

***stairs(y)*** - стъпаловидна диаграма. Аналогична на ***bar*** , но не се изчертават вертикалните линии.

***hist(y)*** - хистограма. ***hist(x)*** разделя диапазона между *min(x)* и *max(x)* на 10 класа. Чрез ***hist(x,n)*** броят на класовете се указва в *n*.

**compass(y) –** радиус-вектори, излизащи от нулата.

**Примери:**

```
»t = 0:.01:2*pi;
»polar(t,sin(2*t).*cos(2*t),'--r')
»x = -2.9:0.2:2.9;
»bar(x,exp(-x.*x))
» x = 0:.25:10;
»stairs(x,sin(x))
»x = -2.9:0.1:2.9;
»y = randn(10000,1);
»hist(y,x)
»Z = eig(randn(20,20));
»compass(Z)
```

### 5.2. Оформяне на графиките

Методите за оформяне на графики се използват не само за добавяне на информация като имена и надписи по графиката, но и за по-доброто изобразяване на данни, които трудно могат да се видят вследствие обикновеното автоматично мащабиране на графиката.

В прозореца на графиката, изчертана с някоя от горните команди, може да се добави текст, означение на осите и/или мрежа.

***title('Текст')*** - поставя текста центриран над графиката (като заглавие).

***xlabel('Текст')*** - поставя текста под оста ***Ox***.

***ylabel('Текст')*** - поставя текста вляво от оста ***Oy***.

***text(x1,y1,'Текст',x2,y2,'Текст', ...)*** - разполага всеки един от текстовете вдясно от точката със съответните координати $x_i$ и $y_i$.

***grid*** - изчертава мрежа.

Веднага трябва да се отбележи, че кирилизирането на надписите в *MATLAB* е възможно, но може да създаде множество различни трудности. Затрудненията се проявяват по различен начин в зависимост от вида на ОС (DOS, Windows 9x, 2000, NT или XP), както и от използваната версия на *MATLAB*. Поради това няма да бъде направен опит за разглеждането на този въпрос.

Поставянето на надписи в произволна точка от графичното поле се извършва с командите ***text*** или ***gtext***. Командата ***text (x,y,'надпис'***) изпълнява надписване, като с аргументи задаваме координатите **x, y** на надписа и текста, заграден в единични кавички. Командата ***gtext('надпис')*** служи за интерактивно надписване, като след въвеждане на командата MATLAB прехвърля управлението в графичния прозорец, където се появява кръстат курсор, подканващ избирането на точка за разполагане на текста вдясно от посочената точка.

***subplot(m,n,p)*** - разделя прозореца на ***m*** клетки по вертикала и ***n*** хоризонтални клетки и следващата графична команда чертае в клетка ***p*** (първи са клетките от първи ред и т.н.).

Комадите **subplot(1,2,1)** и **subplot(1,2,2)** разделят графичния прозорец на две еднакви половини, разположени една до друга. Следваща-

та веднага след **subplot(1,2,1)** команда **plot** чертае графиката в полето отляво.

След изпълнение на командата **subplot** връщането към единичен графичен прозорец се извършва с комадата **subplot(1,1,1)**.

В общия случай, за да се намали вероятността от пропускане интересни свойства на данните, добре е като правило, данните да се представят по няколко различни начина. Представянето на логаритъма е лесно и често се отразява добре на мащабирането. За съжаление на практика се появяват много случаи, при които трябва да се приложат други методи за мащабиране. Логаритмуването може да се приложи само върху положителни числа. В други случаи деформирането на мащаба чрез логаритмуване може да е нежелателно или да води до заблуждение. Поради това, че логаритмуването компресира големите числа, а малките числа обръща в отрицателни с големи абсолютни стойности, то може да превърне графиката, която в обикновен мащаб веднага ще разпознаем, в нещо съвършено непознато на вид.

**Пример:**

```
»Y = round(rand(5,3)*10);
»subplot(2,2,1)
»bar(Y,'group')
»title 'Group'
»subplot(2,2,2)
»bar(Y,'stack')
»title 'Stack'
»subplot(2,2,3)
»barh(Y,'stack')
»title 'Stack'
»subplot(2,2,4)
»bar(Y,1.5)
»title 'Width = 1.5'
```

### 5.3. Чертане на тримерни графики

Пространствената графична информация може да се представя по три начина: с мрежеста графика (меш), с контурна графика и с пространствени криви. Последните се чертаят с командата plot3, която е 3D аналог на 2D командата plot.

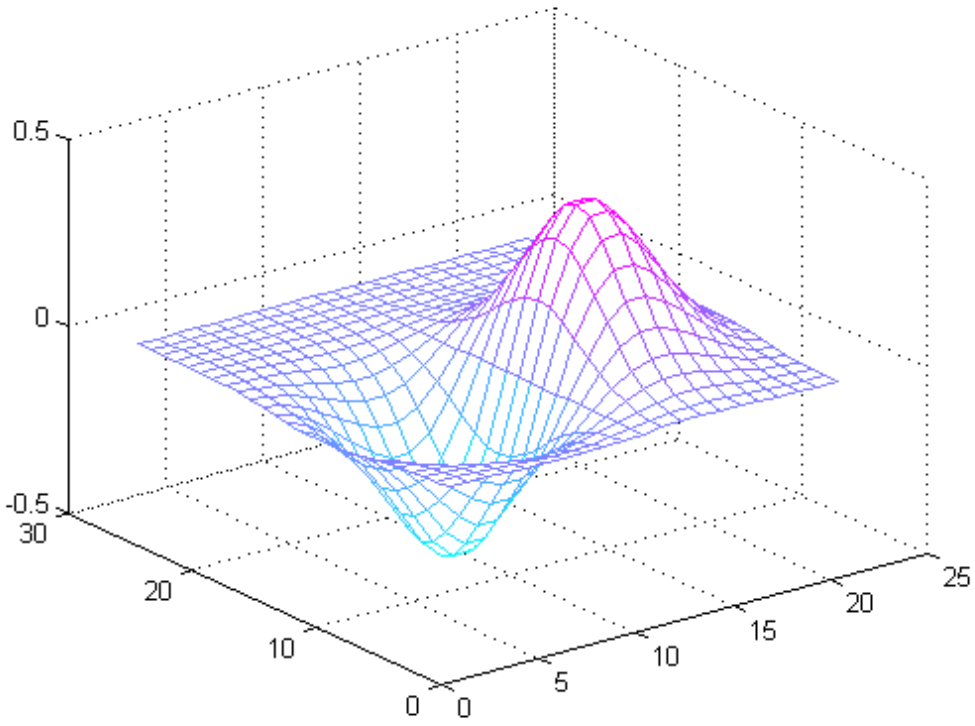
С командата ***mesh(z)*** се изчертава тримерната графика на зависимостта z(x,y).

Командата [X,Y] = ***meshgrid(x,y)*** - генерира матриците X и Y за използване от ***mesh***. Всеки ред в X е равен на x, а броят на редовете на X е равен на дължината на y. Всяка колона на Y се състои от транспонирания y', а броят на колоните на Y е равен на дължината на x.

**Примери:**

Изчертаване на графиката на функцията $z = x * e^{(-x^2-y^2)}$ в граници x=[-2; 2] и y=[-2; 3] се извършва със следните команди:

```
»[x, y] = meshgrid(-2:.2:2,-2:.2:3);
»z = x.*exp(-x.^2-y.^2);
»mesh(z)
```



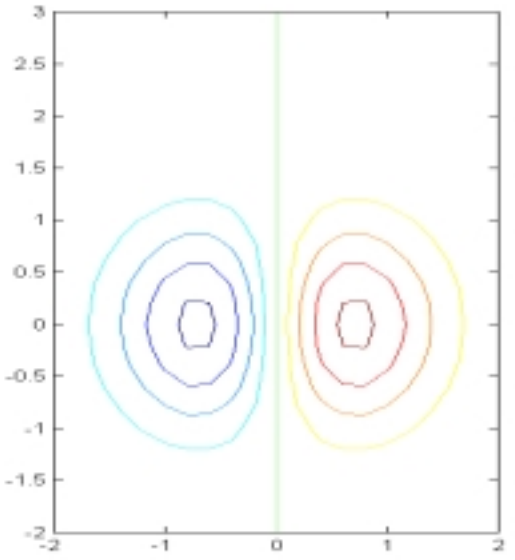

***contour(z)*** - алтернативно представяне на тримерни графики (при поглед перпендикулярен на равнината ***xOy***). Всяка крива от контурната графика представлява стойностите x и y, съответващи на постоянна стойност на z. Автоматично се избира броя на контурите. Той може да бъде указан и от потребителя - ***contour(x,y,z,n)*** - в скалара ***n***, а векторите ***x*** и ***y*** задават ограниченията по тези оси.

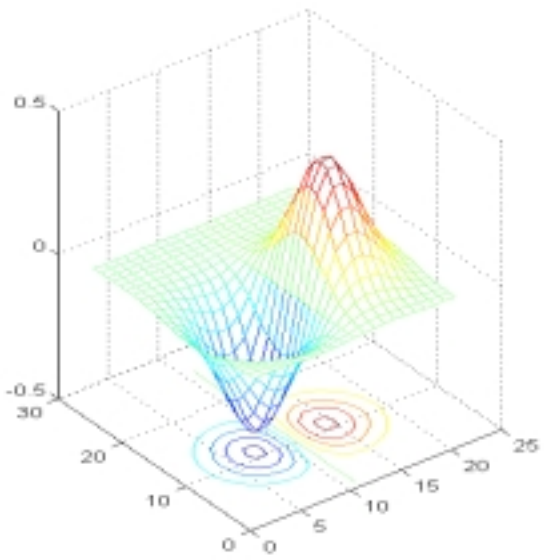
Командата ***meshc*** комбинира в едно графично поле контурна графика и меш. Контурите в контурната графика могат да бъдат надписвани с командата **clabel**.

**Пример:**

Графика на същата функция в същите интервали x=[-2; 2] и y=[-2; 3]:

```
»[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
»Z = X.*exp(-X.^2-Y.^2);
»subplot(1,2,1)
»[C,h] = contour(X,Y,Z);
»subplot(1,2,2)
» meshc(Z);
```





### 5.4. Други команди, свързани с графики

**figure** – отваря (създава) графичен прозорец.

***whitebg*** - указва бял фон на графичните прозорци и черни линии за всички следващи графики (по подразбиране е обратното).

***clf*** – изчиства съдържанието на текущия графичен прозорец.

***close*** - затваря текущия графичен прозорец.

***hold on/off*** - задава режима на изобразяване на графики в текущия графичен обект. Възможните режими са:

а) добавяне на графики със запазване на вече изчертаните - (***on***);

б) изтриване на съществуващите и изчертаване на новите графики - (***off***).

***ishold*** – проверява състоянието на hold и връща 1 при hold on и 0 за hold off.

***axis*** - определя режима на изобразяване и мащабиране на осите на графиките. Има два начина на задаване на командата: ***axis(v)*** и ***axis('s')***, където ***v*** е вектор, а ***s*** е стринг. Векторът съдържа елементи за мащабиране на осите както следва: [***Xmin Xmax Ymin Ymax Zmin Zmax***]. Стринговата променлива може да приема следните стойности: **'*auto', 'ij', 'xy', 'equal', 'square', 'image', 'normal', 'off'', 'on'*.**

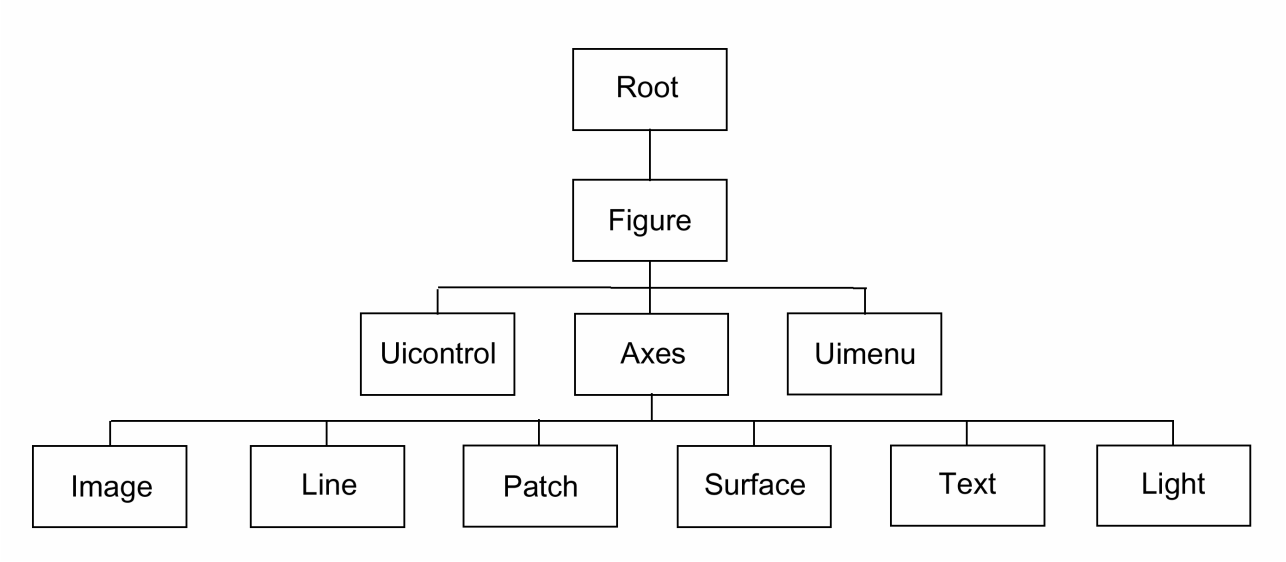
### 5.5. Въведение в операторната графика

Операторна графика (**Handle Graphics**) – това е обектно-ориентирана графична подсистема, която поддържа компоненти, необходими за създаване на компютърни графики.

**Графичните обекти** са базови графични елементи на системата. Те са организирани като дърво на структурна йерархия, което отразява връзката между графичните обекти. Например обекти ***Line*** (линия) се нуждаят от обекти ***Axes*** (оси) в координатна система. Обекти **Axes** съществуват само с обекти ***Figure***. На всеки графичен обект се присвоя-

ва уникален идентификатор, който се нарича дескриптор **(***handle*). Чрез дескрипторите се управляват свойствата на графичните обекти.

Йерархията на графичните обекти в *MATLAB* е показана на фиг. 1.12, а пример за тяхната визуализация на фиг. 1.13.

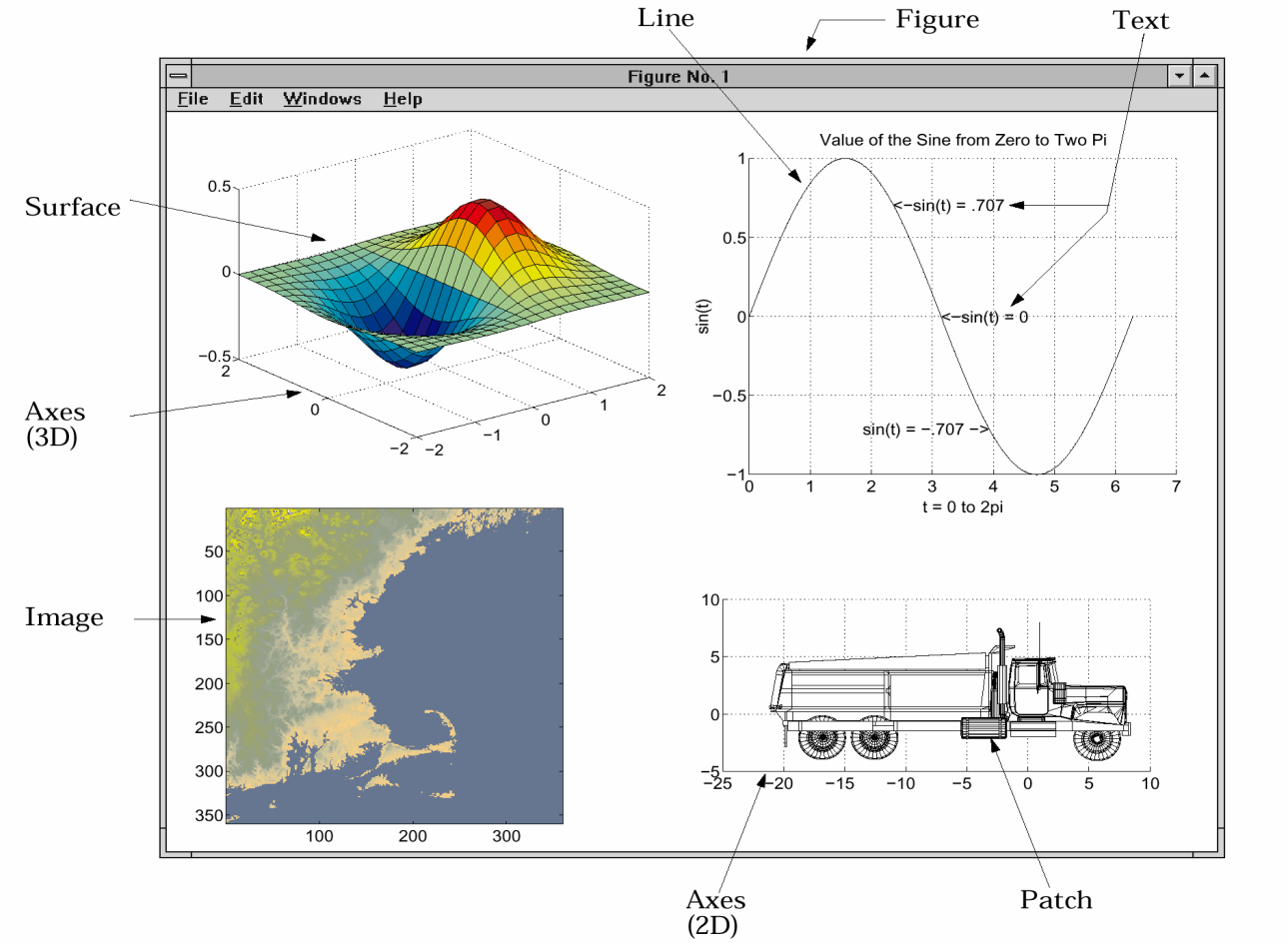



Фиг. 1.12.

Графичният обект ***Root*** се намира на върха на йерархията и съответства на екрана на компютъра. Съществува единствен обект ***Root***, който няма родители, а всички останали графични обекти се явяват неговите наследници. Пряк потомък на този обект се явява графичния обект ***Figure***. Дескрипторът на обекта ***Root*** е 0. Този обект се създава в момента на стартиране на системата *MATLAB* и не може да бъде изтрит. Потребителят може да управлява неговите свойства.

Графичните обекти ***Figure*** съответстват на отделни графични прозорци на екрана на компютъра, където се извеждат (визуализират) данните. В системата *MATLAB* няма ограничения за количество на създаваните прозорци, обаче те зависят от особеностите на използвания компютър. Всички команди от високо ниво и функции, които извеждат графики, например ***plot***, ***surf***, ***contou*r**, автоматично създават графичен обект ***Figure***, ако такъв не съществува. Ако са отворени много графични прозорци, то един от тях се маркира като текущ (активен) графичен обект. Пример за дескриптор на графичен обект ***Figure*** се явява **номера на графичния прозорец.**

Чрез обектите **Uicontrol** се осъществява потребителско управление на интерфейса. Когато потребителят активира обект, се извиква съответстваща функция. Те включват в себе си стандартните средства ***pushbutton***, ***radio button*** и ***slider***.




Фиг.1. 13.

Графичният обект ***Axes*** задава област за изчертаване на графика в прозореца на обекта ***Figure***. Обектите ***Axes*** се явяват наследници за обекта ***Figure*** и в същото време се явяват родители за графичните обекти ***Line***, ***Patch***, ***Surface*** и ***Text***, които се използват за визуализация на данни. Всички команди и функции, които извеждат графики, например ***plot***, ***surf***, ***contour***, ***mesh***, ***bar***, автоматично създават графичен обект ***Axes***, ако той не съществува. Ако в един графичен прозорец са създадени няколко обекта ***Axes***, то един от тях се отбелязва като текущ. Дескрипторът на графичния обект ***Axes*** може да бъде определен с помощта на команда ***gca***.

За практическо усвояване на свойствата на графичния обект ***Axes*** може да се използва демонстрационния файл *hndlaxis*. При стартиране се показва графичен прозорец *Handle Graphics and Axes Object*, който позволява да се управляват такива свойства на обекта ***Axes***, като избор на линейна, логаритмична или полулогаритмична скала по осите, използване на мрежа, посоката на изменение на променливите по осите, цвят на мрежата и размерни линии по осите.

Прозорецът *MiniCommand Window* позволява да се въвеждат собствени команди и да се управляват свойствата на графиката.

Чрез обектите ***Uimenu*** се създават потребителски (падащи) менюта в горната част на прозореца ***Figure***.

***Image*** са двумерни обекти, които използват елементи на правоъгълна матрица като индекси на палитра.

Графичните обекти ***Line*** се явяват графични примитиви, които се използват за създаване на двумерни и тримерни графики. Обектите ***Line*** се явяват наследници за обекта ***Axes*****.** Командите от високо ниво и функциите ***plot, plot3, contour*** създават графични обекти ***line***.

За практическо усвояване на свойствата на графичния обект ***Line*** служи демонстрационният файл *hndlgraf.* След стартиране се показва графичен прозорец *Handle Graphics and Line Objects*, който позволява да се управляват свойствата на обекта ***Line***, като типа, дебелината, размера на маркера и цвета на линията.

Графичният обект ***Patch*** е един или няколко оцветени 'многоъгълника' с отбелязани граници. Обектът ***Patch*** се явява наследник за обекта ***Axes***. Командите и функции от високо ниво ***fill, fill3, contour, contour3*** създават графични обекти ***Patch***.

Графичният обект ***Surface*** служи за тримерна визуализация на масив от данни, когато елементите на масива определят височините на точки над равнината **x-y**. По този начин се формира тримерна повърхнина, състояща от 4-ъгълници, върховете, на които се определят от изходните данни. Повърхността може да бъде със цвят или да представлява само мрежа от линии, свързващи върхове. Обектите **Surface** се явяват наследници на обекта ***Axes*****.** Командите и функции от високо ниво ***pcolor***, ***surf***, ***mesh*** създават графични обекти ***Surface***.

За практическо усвояване на свойствата на графичния обект ***Surface*** може да се експериментира с демонстрационния файл *graf3d*. След стартиране се показва графичният прозорец *3-D Plots in Handle Graphics*, който позволява да се управляват свойствата на обекта ***Surface***.

Графичният обект ***Text*** представлява редове от символи. Обектът ***Text*** се явява наследник на обекта ***Axes***. Командите и функции от високо ниво ***title***, ***xlabel***, ***ylabel*** и ***gtext*** създават графични обекти ***Text***.

Следва да се отбележи, че по подразбиране в графичния обект ***Text*** за писане на текстове със специални символи се използва синтаксисът на езика *Tex*. *Tex*[*] е написан в началото на 80-те от математика Доналд Кнут и много бързо става стандарт за предпечатна подготовка на математически текстове.

Например, формулата на Ойлер $e^{j\varphi} = \cos(\varphi) + j\sin(\varphi)$, може да се представи на езика *Tex* по следния начин: '\ite^{j\phi}=cos(\phi)+jsin(\phi)'.

В табл. 1.1 са показани командите с които се изобразяват специалните символи.

[*] Произнася се *тек* или *тек-х*, но не и *текс*.

За определяне на източника на светлина действаща на всички обекти в границите на ***Axes*** се използва обекта ***Light***.

**Пример:**

Да се визуализира и надпише функцията $f = Ae^{-\alpha t}\sin(\beta t)$

```
»t=0:1000;
»y=0.25*exp(-5e-3*t).*sin(0.2*t);
»plot(t,y,'LineWidth',1.5), grid
»ht=title('');
»set(ht,'String','\itAe^{\rm\alpha\itt} \rmsin(\rm\beta\itt\rm)','FontSize',12)
»hl=Xlabel('');
»set(hl,'String','\itt, \rm\mu\rm sec','FontSize',12)
```

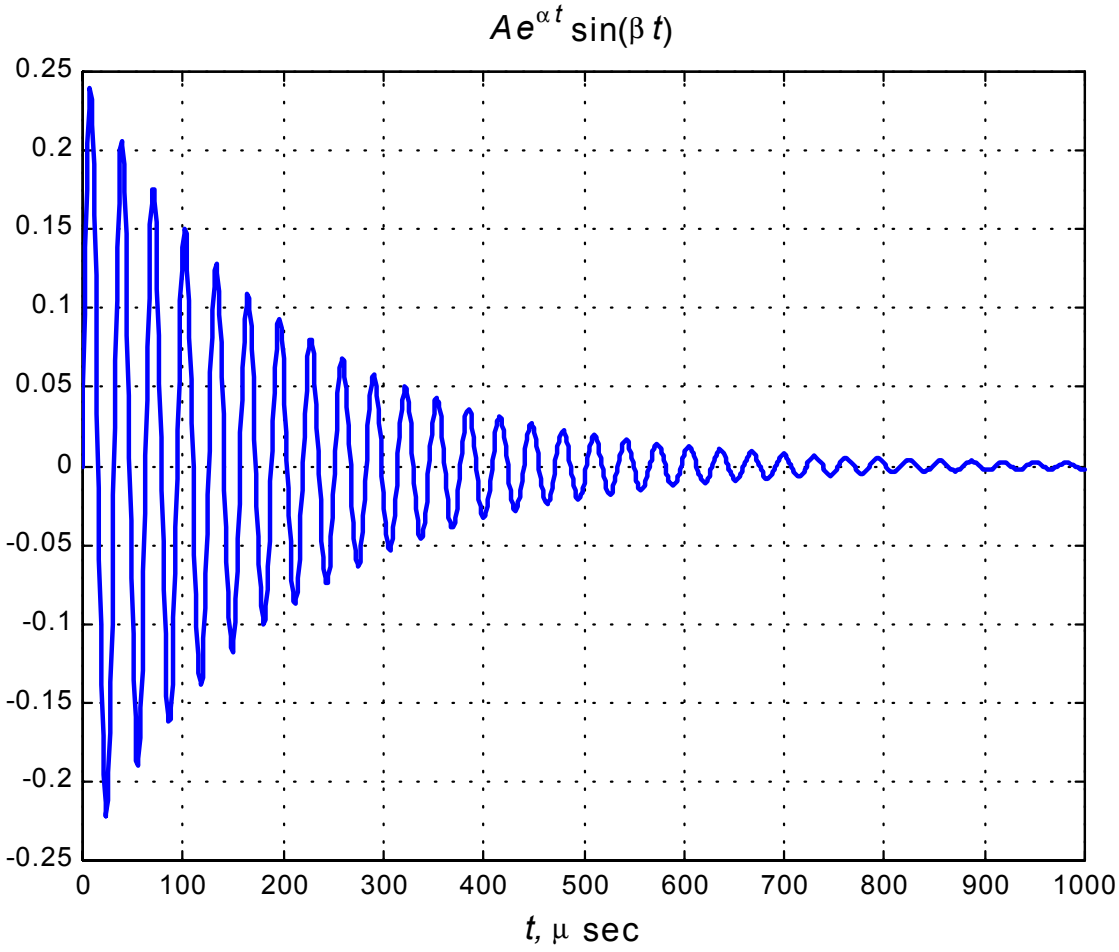



Таблица 1.1.

| Команда | Символ | Команда | Символ | Команда | Символ |
|---|---|---|---|---|---|
| \alpha | α | \upsilon | υ | \sim | ~ |
| \beta | β | \phi | φ | \leq | ≤ |
| \gamma | γ | \chi | χ | \infty | ∞ |
| \delta | δ | \psi | ψ | \clubsuit | ♣ |
| \epsilon | ε | \omega | ω | \diamondsuit | ♦ |
| \zeta | ζ | \Gamma | Γ | \heartsuit | ♥ |
| \eta | η | \Delta | Δ | \spadesuit | ♠ |
| \theta | θ | \Theta | Θ | \leftrightarrow | ↔ |

| | | | | | |
|---|---|---|---|---|---|
| \vartheta | ϑ | \Lambda | Λ | \leftarrow | ← |
| \iota | ι | \Xi | Ξ | \uparrow | ↑ |
| \kappa | κ | \Pi | Π | \rightarrow | → |
| \lambda | λ | \Sigma | Σ | \downarrow | ↓ |
| \mu | μ | \Upsilon | Υ | \circ | ° |
| \nu | ν | \Phi | Φ | \pm | ± |
| \xi | ξ | \Psi | Ψ | \geq | ≥ |
| \pi | π | \Omega | Ω | \propto | ∝ |
| \rho | ρ | \forall | ∀ | \partial | ∂ |
| \sigma | σ | \exists | ∃ | \bullet | • |
| \varsigma | ζ | \ni | ∍ | \div | ÷ |
| \tau | τ | \cong | ≅ | \neq | ≠ |
| \equiv | ≡ | \approx | ≈ | \aleph | ℵ |
| \Im | ℑ | \Re | ℜ | \wp | ℘ |
| \otimes | ⊗ | \oplus | ⊕ | \oslash | ∅ |
| \cap | ∩ | \cup | ∪ | \supseteq | ⊇ |
| \supset | ⊃ | \subseteg | ⊆ | \subset | ⊂ |
| \int | ∫ | \in | ∈ | \o | o |
| \rfloor | ⌋ | \lceil | ⌈ | \nabla | ∇ |
| \lfloor | ⌊ | \cdot | · | \ldots | … |
| \perp | ⊥ | \neg | ¬ | \prima | ′ |
| \wedge | ∧ | \times | × | \0 | ∅ |
| \rceil | ⌉ | \surd | √ | \mid | \| |
| \vee | ∨ | \varpi | ϖ | \copyright | © |
| \langle | ⟨ | \rangle | ⟩ | | |

Както стана ясно всички графични обекти имат свойства, които определят, как те да се извеждат на екрана. *MATLAB* предоставя 2-механизма за задаване на свойствата. Свойствата на обекта могат да бъдат зададени от функцията при създаване на обекта или могат да бъдат променени чрез функцията ***set*** (когато обектът вече съществува).

**Пример:**

Следващите редове създават три графични обекта (figure, axes и line)

```
»days = [ 'Su' ; 'Mo' ; 'Tu' ; 'We' ; 'Th' ; 'Fr' ; 'Sa' ]
»temp = [ 21.1 22.2 19.4 23.3 23.9 21.1 20.0 ];
»f = figure;
»a = axes( 'YLim' , [16 26] , 'Xtick' , 1:7 , 'XtickLabel' , days )
»h = line(1:7, temp)
```

където

**days** - е масив от символи, съдържащ съкращения на дните от седмицата;

**temp** - е числен масив съдържащ типични дневни температури.

Прозорецът на изображението се създава след извикване на ***figure*** без аргументи, т.е. със стойности по премълчаване. Трите идентификатора на графичните обекти (figure, axes и line) се съхраняват в променливите f, a, h за по-нататъшно използване.

За да се изведе списък на всички текущи установени свойства на зададения обект се извиква ***get*** с идентификатор на обекта

```
» get(h)
Color = [0 0 1]
EraseMode = normal
LineStyle = -
LineWidth = [0.5]
Marker = none
MarkerSize = [6]
MarkerEdgeColor = auto
MarkerFaceColor = none
XData = [ (1 by 7) double array]
YData = [ (1 by 7) double array]
ZData = []
…
```

За да се види стойността на отделно свойство се използва командата ***get*** с име на желаното свойство.

```
» get(h , 'Color')
ans =
        0    0.8000    0.8000
```

Функция ***set*** позволява да се зададат нови свойства на графичните обектите, чрез указване на идентификатора на графичния обект и съвкупности по двойки: наименование на свойство - нова стойност. В качеството на упражнение ще променим цвета и ширината на линнията от предходния пример.

```
» set( h , 'Color' , [0 .8 .8] , 'LineWidth' , 3)
```

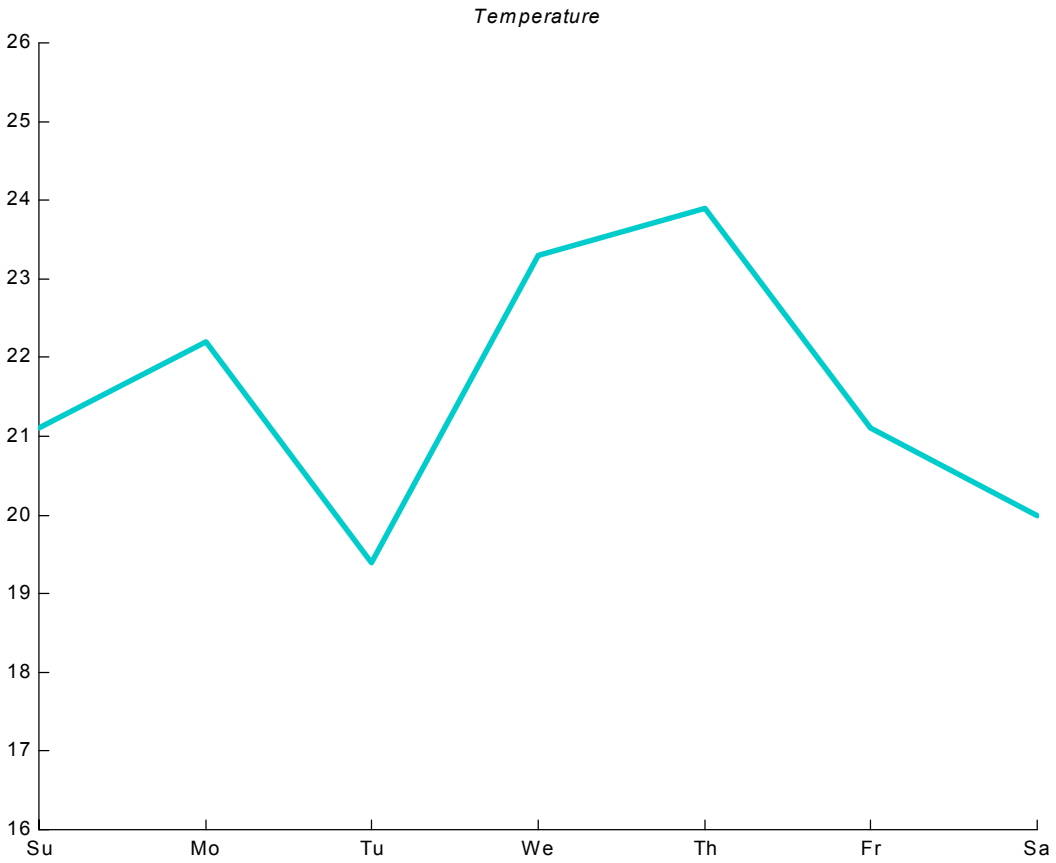
За да се види списъка на всички достъпни свойства на зададения графичен обект, се извиква ***set*** с идентификатор на графичния обект.

```
» set(h)
Color
EraseMode: [ {normal} | background | xor | none ]
LineStyle: [ {-} | -- | : | -. | none ]
LineWidth
   Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | < | pentagram |
                                hexagram | {none} ]
MarkerSize
```

MarkerEdgeColor: [ none | {auto} ] -or- a ColorSpec.
MarkerFaceColor: [ {none} | auto ] -or- a ColorSpec.
XData
YData
ZData
...

Обекти ***axes*** имат много детални свойства. Със следните редове озаглавяваме фигурата.

```
» t = get(a , 'title' );
» set(t , 'String' , 'Temperature' , 'FontAngle' , 'oblique')
```




### 5.5.1. Графичен потребителски интерфейс (Graphical User Interface)

Принципите за изграждане на графичен потребителски интерфейс (ГПИ) в по – голямата си част са прости и универсални. Те се прилагат в *MATLAB* също така добре, както и в много други програми.

За илюстриране на предоставяните възможности ще разгледаме един пример за създаване на ГПИ. Първо се създава ***pushbutton*** в центъра на прозореца (**figure**) и неговият идентификатор се запомня в променливата **b**. Натискайки така създадения бутон не се предизвиква никаква реакция.

```
» b = uicontrol('Style','pushbutton', ...
        'Units','normalized', ...
        'Position',[.5 .5 .2 .1], ...
        'String','click here');
```

В символната променлива **s** се записва команда, която да променя положението (координатите за визуализиране) на бутона в графичния прозорец.

```
» s = 'set(b,"Position",[.8*rand .9*rand .2 .1])';
```

За да се изпълни израза записан в символната променлива **s** се използва функцията ***eval***.

```
» eval(s)
```

Чрез командата ***set*** се променя свойството за обработка на събитие "натискане на бутона" на създадения бутон

```
» set(b,'Callback',s)
```

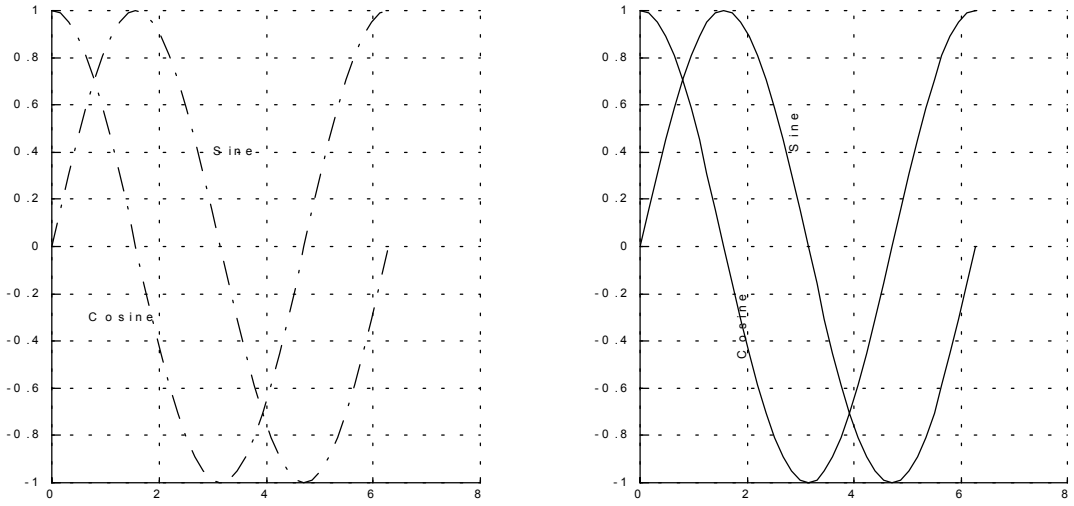
В резултат на което при всяко натискане на бутона той ще се премества на случайни места в графичния прозорец.

*MATLAB* предоставя инструментално средство за проектиране и бързо изграждане на ГПИ. Стартира се с командата ***guide*** и се състои от пет инструмента. Работата с инструменталното средство няма да бъде разгледана поради ограничения обем на пособието.

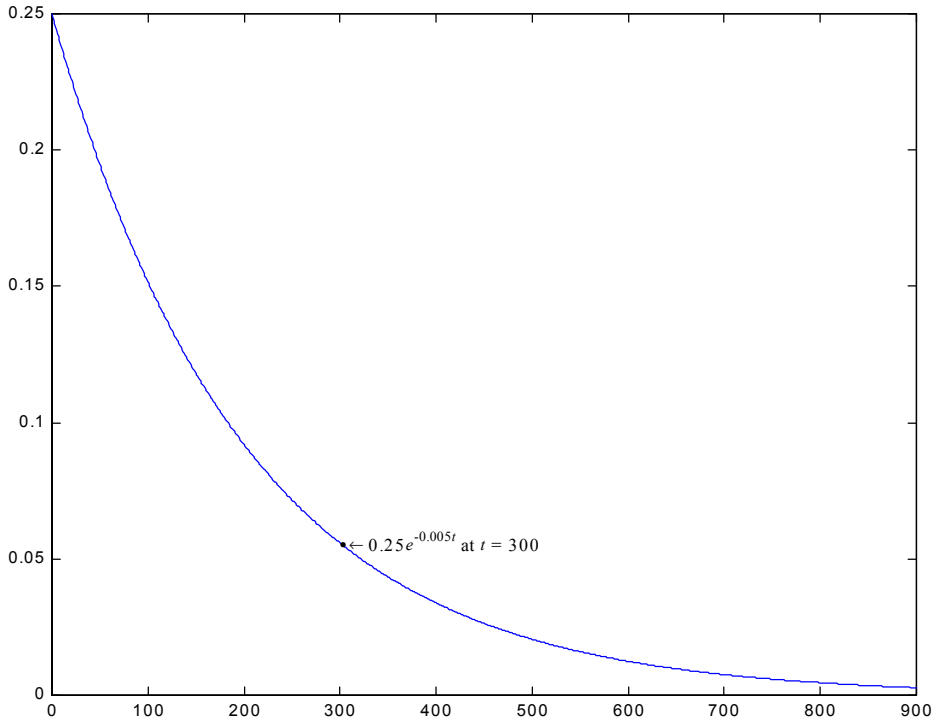
Допълнителни примери за самостоятелна работа.

**Примери:**

```
»t = 0:pi/20:2*pi;
»s = sin(t);
»c = cos(t);
»figh = figure('Position',[30 100 800 350],...
'DefaultAxesColor',[.8 .8 .8])
»axh1 = subplot(1,2,1); grid on
»set(axh1,'DefaultLineLineStyle','-.')
»line('XData',t,'YData',s)
»line('XData',t,'YData',c)
»text('Position',[3 .4],'String','Sine')
»text('Position',[2 -.3],'String','Cosine',...
'HorizontalAlignment','right')
»axh2 = subplot(1,2,2); grid on
»set(axh2,'DefaultTextRotation',90)
»line('XData',t,'YData',s)
»line('XData',t,'YData',c)
»text('Position',[3 .4],'String','Sine')
»text('Position',[2 -.3],'String','Cosine',...
'HorizontalAlignment','right')
```




```
»t = 0:900;
»plot(t,0.25*exp(-0.005*t))
»text(300,.25*exp(-0.005*300),...
'\bullet\leftarrow\fontname{times}0.25{\ite}^{-0.005{\itt}} at {\itt} = 300');
```



# 6. ПРИЛОЖЕНИЯ НА MATLAB В РАЗЛИЧНИ ПРЕДМЕТНИ ОБЛАСТИ

## 6.1. Решаване на системи линейни уравнения

Разглежда се системата от ***n*** линейни уравнения с ***n*** неизвестни:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2$$
$$.....$$
$$a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n$$

Решение на тази система е всяка наредена *n*-торка $(\alpha_1, \alpha_2, ..., \alpha_n)$ от стойности на неизвестните $x_1=\alpha_1, x_2=\alpha_2, ..., x_n=\alpha_n$, които заместени в уравненията на системата, удовлетворяват всяко едно от тях. Системата се нарича *съвместима*, ако притежава поне едно решение. В противен случай тя се нарича *несъвместима*.

Ако се означи

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ..... & & & \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}, \; b = \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}, \; X = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix},$$

то системата може да се запише в матричен вид като *AX=b.*

**Пример:**

Зададена е следната система от линейни уравнения

$x_1 + 2x_2 + 3x_3 = 4$

$2x_1 + 3x_2 + 4x_3 = 5$

$4x_1 + 2x_2 + 5x_3 = 1$

Тази система в матрична форма има вида:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 2 & 5 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}$$

Записът в MATLAB е:

```
»a=[1 2 3; 2 3 4; 4 2 5]
a =
     1     2     3
     2     3     4
     4     2     5
»b=[4; 5; 1]
b =
     4
     5
     1
```

Решението е:

```
»x=a\b
x =
   -1.4000
    1.8000
```

0.6000

### 6.1.1. Лошо обусловени линейни системи

Почти всяка аритметична операция при компютърни пресмятания завършва със закръгляне. При решаване на система линейни уравнения се извършват голям брой аритметични операции, резултатът на всяка от които на практика съдържа грешка от закръгляне. Тези грешки могат да се интерпретират като малки промени в коефициентите и свободните членове на системата. Поради това изключително важен въпрос е как малки промени в коефициентите или в свободните членове на системата се отразяват на решението. Въпреки че са разработени алгоритми, които потискат разпространението на грешките, съществуват системи, които са много чувствителни дори към малки промени в коефициентите. Тези системи се наричат *лошо обусловени.* Анализ на причините за лошата обусловеност и практически критерий за установяването й може да се намери в специализираната литература. За илюстрация на този проблем се разглежда система от две уравнения с две неизвестни.

**Пример:**

$$x_1 + 0.99x_2 = 1.99$$
$$0.99x_1 + 0.98x_2 = 1.97$$

Решението на тази система е:

```
» a=[1 0.99;0.99 0.98]
a =
    1.0000    0.9900
    0.9900    0.9800
» b=[1.99;1.97]
b =
   1.9900
   1.9700
» x=a\b
x =
   1
   1
```

Ако в свободните членове се появи малко смущение в третия знак след десетичната запетая, вместо изходната трябва да решим системата:

$$\tilde{x}_1 + 0.99\tilde{x}_2 = 1.991$$
$$0.99\tilde{x}_1 + 0.98\tilde{x}_2 = 1.969$$

```
» a=[1 0.99;0.99 0.98]
a =
    1.0000    0.9900
    0.9900    0.9800
```

```
» b=[1.991;1.969]
b =
        1.9910
        1.9690
» x=a\b
x =
   -18.7000
    20.9000
```

Точното решение на тази система е $x_1 = -18.7$ и $x_2 = 20.9$. Промени в дясната страна около 0.05% съответно водят до промени в решението $\delta x_1 = \left|\frac{(x_1 - \tilde{x}_1)}{x_1}\right| = 1970\%$ и $\delta x_2 = \left|\frac{(x_2 - \tilde{x}_2)}{x_2}\right| = 1990\%$.

За да се оцени доколко е обусловена дадена матрица е предвидена командата ***cond***. Тази команда има за входен аргумент проверяваната матрица. Например ***cond(a)*** връща числото на обусловеност на матрицата **a**. Числото на обусловеност е количествена мярка за това колко лошо обусловена е една матрица. В примера по-горе матрицата **a** притежава голямо число на обусловеност:

```
» cond(a)
ans =
        3.9206e+004
```

Десетичяият логаритъм от числото на обусловеност ни дава груба представа за това, колко десетични знака биха могли да бъдат загубени с компютърното изчисление вследствие на грешките от закръгляване. Следователно от горното се вижда, че решението с използване на ***a*** ще доведе до загубата на 4-5 десетични разряда. Тъй като лошо обусловените матрици се срещат относително често на практика, полезно е да имаме пример на лошо обусловена матрица за изпитване на програмите, които разработваме. Един последен пример на лошо обусловена матрица е следният:

```
» matrix=[1 10000;0 1]
matrix =
        1       10000
        0           1
» cond(matrix)
ans =
        100000002
» log10(ans)
ans =
                8.0000
```

### 6.2. Полиномно изглаждане и полиномна интерполация

Зададена е параболата $y = a_1x^2 + a_2x + a_3$, която преминава през следните 3 точки: (-1, 0), (0, 2) и (1, 6). Да се определят коефициентите на полинома *y*.

Функцията ***polyfit(x,y,n)*** връща като резултат решението на тази задача. Векторите *x* и *y* съдържат координатите на точките, а *n* е реда на полинома.

```
»x=[-1, 0,1]
»y=[0, 2, 6]
»p=polyfit(x,y,2)
```

Резултат:

```
p= 1 3 2
```

Това означава, че търсеният полином е $y = x^2 + 3x + 2$.

В общия случай функцията ***p = polyfit(x,y,n)*** намира коефициентите на полинома, който апроксимира функцията *y(x)* в смисъл на метода на най-малките квадрати. В ***p*** се изчисляват коефициентите на полинома, а във векторите *x* и *y* се задават входните данни, в *n* се указва желания ред на полинома.

За интерполация с помощта на кубичен сплайн се използва функцията ***yi = spline(x,y,xi)*.** Координатите на възловите точки се задават във векторите *x* и *y*. ***xi*** е вектор с желаните стойности, а ***yi*** - вектор с интерполираните стойности.

**Пример:**

В таблицата са зададени стойности за *x* и *y*

| X | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Y | 2.6913 | 3.2007 | 3.9049 | 4.7170 | 5.5902 |

Да се определи *yi* за *xi = 2.5*.

Решение:

```
»x=1:5
»y=[2.6913 3.2007 3.9049  4.7170 5.5902]
»yi=spline(x,y,2.5)
yi=3.5345
```

За едномерна таблична интерполация се прилага ***yi=interp1(x, y, xi, '<метод>')***, където незадължителния параметър *'метод'* може да има следните значения:

***nearest*** - стъпаловидна;

***linear*** - линейна ;

***spline*** - с кубичен сплайн;

***cubic*** - с кубична интерполация.

**Пример:**

Във вектора *t* са зададени години, а в *p* населението на USA за съответните години. Да се намери колко е било населението на USA за 1975 г.

```
»t = 1900:10:1990;
»p = [75.995  91.972  105.711  123.203  131.669...
    150.697  179.323  203.212  226.505  249.633];
»interp1(t, p,1975)
ans =
   214.8585
```

### 6.3. Намиране на екстремум на функция на една променлива

При много оптимизационни задачи е необходимо да се намери максимумът или минимумът на целева функция, която зависи само от един управляващ параметър.

Функцията ***fmin*** в MATLAB се прилага за минимизация на функция на една променлива. Използваните числени методи са - на *"златното сечение"* и на *параболичната интерполация*. Общият вид е: ***xmin=fmin('име на функцията', x1,x2,options).*** В ***xmin*** се запомня стойността на локалния минимум на функцията в интервала $x_1 \le x \le x_2$. Векторът ***options*** съдържа 18 управляващи параметъра. Чрез него се настройват алгоритмите за оптимизация, прилагани, както в MATLAB, така и в приложния пакет Optimization Toolbox. Функцията ***fmin*** използва само 3 от тези параметри: ***options(1)*** - за извеждане на междинни резултати (0 - не се извеждат, 1 - извеждат се), ***options(2)*** - итерационна грешка (разликата между две съседни итерации и подразбиране е 1e-4), ***options(14)*** - максималния брой итерации (500).

**Пример:**

Да се намери минимума на функцията $f(x) = x^3 - 2x - 5$ в интервала [0, 2].

Първо написваме M-функцията с име *f.m*.

```
function y = f(x)
y = x.^3-2*x-5;
```

От средата на MATLAB извикваме ***fmin*** с необходимите параметри

```
» x=fmin('f',0,2)
```

Полученият резултат е:

```
x =
   0.8165
```

Стойността на функцията $f(x) = x^3 - 2x - 5$ при *x = 0.8165* е:

```
» y=f(x)
y =
  -6.0887
```

### 6.4. Числено интегриране

Най-често приближената стойност на интеграла се пресмята с помощта на *формули за числено интегриране,* които обикновено се наричат *квадратурни формули.* Общата идея е приближената стойност на интеграла да се пресметне като сума $\tilde{I} = \sum_{k=0}^{n} A_k f(x_k)$ от стойности на подинтегралната функция, пресметнати в някакви точки $x_k$, наречени *възли,* умножени с коефициенти $A_k$, наречени *тегла* (или *тегловни коефициенти).* Възлите и тегловните коефициенти не зависят от функцията.

Обикновено възлите и тегловните коефициенти се избират така, че формулата да е точна за полиноми от възможно най-висока степен. Казва се, че квадратурната формула е *точна за полиноми от степен m*, ако за всеки полином *P(x)* от степен *m* - $\int_a^b P(x)dx = \sum_{k=0}^{n} A_k P(x_k)$.

Според начина на избор на възлите и тегловните коефициенти се получават различни типове квадратурни формули.

Ако възлите $x_k$ са предварително избрани и равноотдалечени, т.е.

$x_k = x_0 + kh, \quad k = 0,1,...n; \quad h = \frac{b-a}{n}; \quad x_0 = a; \quad x_n = b$ и тегловните коефициенти са подбрани така, че формулата да е точна за полиноми от най-висока степен в този случай квадратурните формули се наричат *формули на Newton-Cotes* или *интерполационни квадратурни формули.*

Функциите в MATLAB ***quad*** и ***quad8*** използват различни квадратурни формули. Първата ***(quad)*** използва *квадратурните формули на Newton-Cotes от 2-ри ред (адаптивно правило на Симсон),* а функцията ***quad8*** *– квадратурните формули на Newton-Cotes от 8-ми ред*.

Синтаксисът на извикване на функциите е следния:

***I=quad('<име на подинтегралната функция>', a, b, tol, trace, P1, P2, …)***

***I=quad8(…),***

където

***'<име на подинтегралната функция>'*** може да бъде вградена функция или M файл.

***a*** и ***b*** са границите на интегриране,

***tol*** – зададена относителна грешка. По подразбиране *tol=1e-3*.

***trace*** – когато този аргумент е различен от 0, се построява точкова графика на подинтегралната функция.

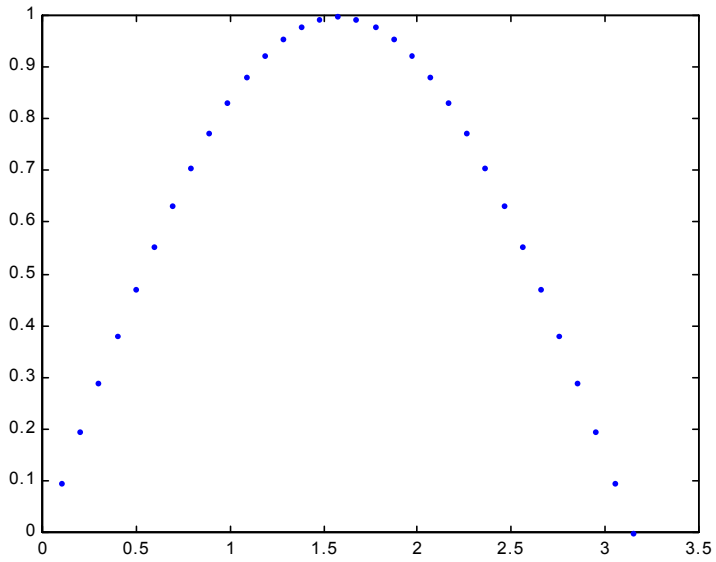
***P1, P2, …*** - предаване на тези параметри на подинтегралната функция.

**Пример:**

Изчисляване на интеграла

$I = \int_0^{\pi} \sin(x)dx$, с точна стойност 2.

» I=quad('sin',0,pi,1e-4,1)
I =2.0000




Ако подинтегралната функция е от особен тип, например:

$I = \int_0^1 \sqrt{x}dx$ се предпочита използването на ***quad8***.

Тези функции не позволяват да се интегрира особена функция от вида $I = \int_0^1 \frac{1}{\sqrt{x}}dx$.

В *MATLAB* може да се интегрира и използвайки формулата на трапеците.

Синтаксисът на функцията, която реализира тази формула е следния:

***I=trapz(x,y,dim)***,
където
***x*** - е аргумент на подинтегралната функция
***y*** - подинтегрална функция.
***dim*** - размерност на аргумента.

**Пример:**

Изчисляване на същият интеграл $I = \int_0^{\pi} \sin(x)dx$, с точна стойност 2 по формулата на трапеците.

Избира се равномерна мрежа
» x=0:pi/100:pi; y=sin(x);
Стойността на интеграла е:
» I=trapz(x,y)
I =1.9998.

Вижда се, че полученият резултат е по-неточен, отколкото при използване на квадратурните формули.

### 6.5. Решаване на обикновени диференциални уравнения

Много математически модели, които описват физически процеси, протичащи в непрекъснати среди, водят до решаване на диференциални уравнения. Поради това въпросите, свързани с намирането на решение на диференциални уравнения, са изключително важни за практиката.

При числено решаване на диференциални уравнения винаги се търси частно решение, което удовлетворява съответния брой гранични условия. Ако всички гранични условия са зададени в една точка, то задачата се нарича *задача на Коши.* Този тип задачи най-често са свързани с описание на процеси, които се развиват във времето и тогава граничните условия описват състоянието на системата в началния момент от време. Затова задачата на Коши понякога се нарича *начална задача.* Ако върху търсената функция са наложени гранични условия в две различни точки, то се получава така наречената *гранична задача.* Числените методи за решаване на начални и гранични задачи са съществено различни.

Обикновено диференциално уравнение от *m*-ти ред, което може да се запише така, че в лявата му страна да се намира са най-старшата производна, а в дясната му страна тази производна да не участва, може да се приведе до система от *m* уравнения първи ред чрез въвеждане на нови *m*-1 променливи. Например уравнението $y'' = f(x, y, y')$ може да се сведе до системата

$$\begin{vmatrix} y' = z \\ z' = f(x, y, z) \end{vmatrix}.$$

По същество методите за решаване на задачата на Коши за системи обикновени диференциални уравнения не се различават от тези за решаване на едно уравнение.

Системата *MATLAB* позволява решаване на диференциални уравнения по повече от 7 числени методи. В разгледания пример се използва функцията ***ode23***, реализираща явните формули на Рунге-Кута от 2-ри и 3-ти ред.

В качеството на пример се разглежда едноизходна механична система.

Прилага се сила за хоризонтално преместване на тяло, на която противодейства паралелно разположена пружина и хидравличен амортисьор, свързващи тялото с неподвижна среда.

Уравнението на динамиката на система е

$$f(t) = M\frac{d^2y(t)}{dt^2} + B\frac{dy(t)}{dt} + Ky(t)$$

където

*f(t)* – приложената върху системата сила е *f(t)=Asin(t)* (входна величина);

*y(t)* – хоризонтално преместване на тялото (изходна величина);

*M* – маса на тялото (параметър);

*K* – константа на пружината (параметър);

*B* – коефициент на вътрешно триене (параметър).

След преобразуване се получава

$$\frac{d^2y(t)}{dt^2}+\frac{B}{M}\frac{dy(t)}{dt}+\frac{K}{M}y(t)=\frac{A}{M}\sin(t)$$

или

$$y''+\frac{b}{m}y'+\frac{k}{m}y=\frac{a}{m}\sin(t)$$

След полагане $w_1 = y'$ и $w_2 = y$, това уравнение се представя във вид на система от две уравнения от от първи ред

$$w_1' = -\frac{b}{m}w_1 - \frac{k}{m}w_2 + \frac{a}{m}\sin(t)$$

$$w_2' = w_1$$

Тази система от две уравнения се записва във векторна форма във функцията *spring.m*.

```
function wd=spring(t,w)
% y''+b/m*y'+k/m*y=a/m*sin(t)
a=2.0; % N
b=1.4; % Ns/m
k=0.1; % N/m
m=2.0; % kg
[rows,cols]=size(w);
wd=zeros(rows,cols);
%
wd(1)=-b/m*w(1)-k/m*w(2)+a/m*sin(t);
wd(2)=w(1);
```

За да се реши даденото диференциално уравнение (да се види реакцията) трябва да се въведат интервала на интегриране и началните условия

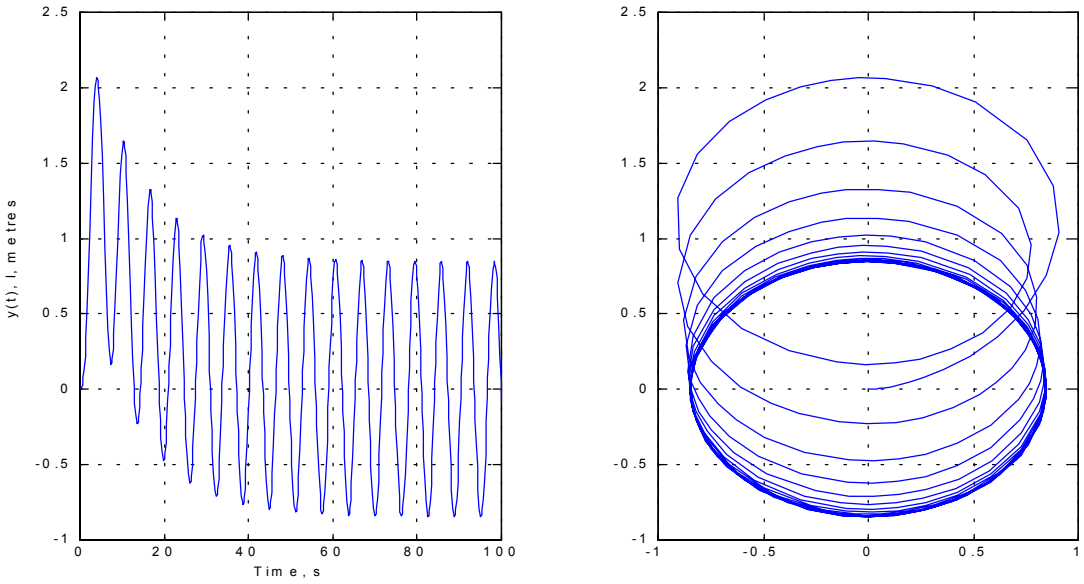
```
» t0=0; tf=100; % Интервала на интегриране
» w0=[0.0 0.0]; % Задаване на начални условия
» tspan=[t0 tf]
```

Решаване на диференциалното уравнение

```
» [t,w]=ode23('spring',tspan,w0)
» subplot(1,2,1)
» plot(t,w(:,2)); grid
» xlabel('Time, s')
» ylabel('y(t), l, metres')
```

Фазовият потрет може да се види с командата

```
» subplot(1,2,2)
» plot(w(:,1),w(:,2)); grid
```




## 6.6. Обработка на опитни данни

Получените данни от непосредствените измервания при провеждане на опитите са в “суров” вид, т.е. не е известна тяхната точност и достоверност и не могат да се използват за анализ и заключения. Затова е необходима допълнителна математическа обработка. Понеже тези данни в голямата си част са случайни числа и величини, затова се обработват по методите на математическата статистика.

Почти винаги при експерименти се търси връзка между две величини. В този случай като резултат се получава група от сдвоени данни, условно означени с $X_i$ $Y_i$, които определят някаква зависимост между величините. Ако тази зависимост е еднозначно определена тя се нарича функционална, а ако е межу случайни величини – стохастична (вероятностна). Във втория случай за обработка на данните и установяване на зависимостта между величините се използва регресионния анализ. С него се решават следните задачи:

- определяне на формата на зависимостта между променливите;
- определяне на функцията на регресия и оценка на параметрите й;
- проверка на статистически хипотези за надеждността на регресионния модел и параметрите му.

Аналитичният израз на зависимостта между величините *x* и *y* се установява чрез метода на най-малките квадрати.

Като примери са разгледани задачите за линеен и квадратичен (параболичен) регресионен анализ.

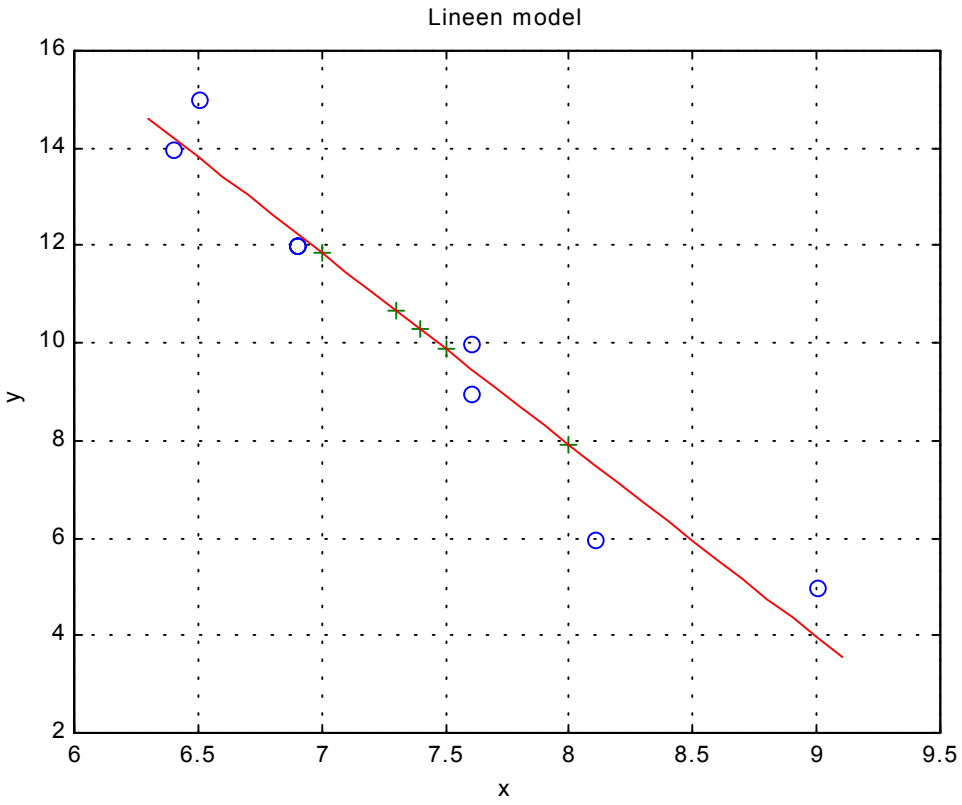
### 6.6.1. Линеен регресионен анализ и коефицент на корелация.

Моделът се търси от вида Y=ax+b, където **a** и **b** са неизвестни коефициенти. В таблицата са дадени използваните формули.

| **Основно уравнение:** | **Y=ax+b;** |
|---|---|
| Средни стойности: | $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n};$ <br> $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n};$ |
| Основни зависимости: | $S_{xx} = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2;$ <br> $S_{xy} = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y};$ <br> $S_{yy} = \sum_{i=1}^{n} y_i^2 - n\bar{y}^2;$ |
| Ковариация: | $q = \frac{S_{xy}}{n-1};$ |
| Коефицент на корелация: | $r = \frac{S_{xy}}{\sqrt{S_{xx} S_{yy}}};$ |
| Коефицент: | $a = \frac{S_{xy}}{S_{xx}};$ |
| Коефицент: | $b = \bar{y} - a\bar{x};$ |

```
% Експериментални данни
x=[6.9,7.6,7.6,9,8.1,6.5,6.4,6.9]
y=[12 10 9 5 6 15 14 12]

% Програма
n=length(x);
xmean=mean(x)
ymean=mean(y)
Sxx=sum(x.^2)-n*(xmean^2);
Sxy=sum(x.*y)-n*xmean*ymean;
Syy=sum(y.^2)-n*(ymean^2);
q=Sxy/(n-1)
r=Sxy/sqrt(Sxx*Syy)
% Коефициенти на уравнението y=ax+b
a=Sxy/Sxx
b=ymean-a*xmean
```

```
% Пример
xx=[7 8 7.5 7.3 7.4]
yy=a*xx+b

xx_model=6.3:0.1:9.1;
yy_model=a*xx_model+b;

plot(x,y,'o',xx,yy,'+',xx_model,yy_model)
xlabel('x')
ylabel('y')
title('Lineen model')
grid
```




Описаният алгоритъм за регресионен анализ реализиран като M-функция *linear.m* е:

```
function [a,b,q,r,xmean,ymean]=linear(x,y)
% Функция за линеен регресионен анализ [a,b,q,r,xmean,ymean]=linear(x,y)
% входни аргументи са векторите x и y;
% изходни аргументи:
% a и b - коефициенти на линейното уравнение y=ax+b;
% q - коефициент на ковариация;
% r - коефициент на корелация;
% xmean и ymean - средно аритметични стойности на x и y
%
```

```
% Извикване на функцията linear(x,y) е
% [a,b,q,r,xmean,ymean]=linear(x,y), като се
% записват само тези изходни аргументи, които са необходими
% напр. [a,b]=linear(x,y)
nargchk(2,2,nargin) % проверка за броя на входните аргументи
n=length(x);
xmean=mean(x);
ymean=mean(y);
Sxx=sum(x.^2)-n*(xmean^2);
Sxy=sum(x.*y)-n*xmean*ymean;
Syy=sum(y.^2)-n*(ymean^2);
q=Sxy/(n-1); % изчисляване на коефициентът на ковариация
r=Sxy/sqrt(Sxx*Syy); % изчисляване на коефициентът на корелация
% Коефициенти на уравнението y=ax+b
a=Sxy/Sxx;
b=ymean-a*xmean;
```

Извикване на М-функцията от командния ред или от друг М-файл ***[a,b,q,r,xmean,ymean]=linear(x,y).***

Важно е да се обърне внимание, че се записват само желаните изходни аргументи. Например ако желаем да получим само коефициентите ***a*** и ***b*** на уравнението

```
» [a,b]= linear(x,y)
a =
  -3.9420
b =
  39.4476
```

Получаване на помощна информация за разработената функция

```
» help linear
```

*Функция за линеен регресионен анализ*
*[a,b,q,r,xmean,ymean]=linear(x,y)*
*входни аргументи са векторите x и y;*
*изходни аргументи:*
*a и b - коефициенти на линейното уравнение y=ax+b;*
*q - коефициент на ковариация;*
*r - коефициент на корелация;*
*xmean и ymean - средно аритметични стойности на x и y*

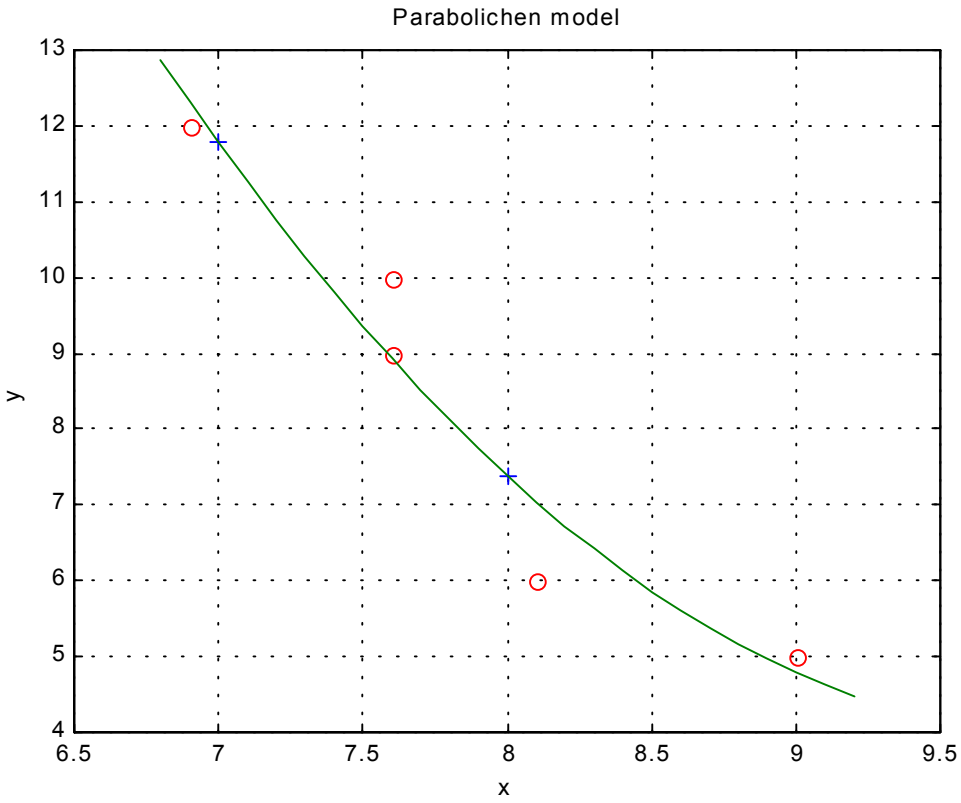
*Извикване на функцията linear(x,y) е*
*[a,b,q,r,xmean,ymean]=linear(x,y), като се*
*записват само тези изходни аргументи, които са необходими*
*напр. [a,b]=linear(x,y)*

### 6.6.2. Квадратичен (параболичен) регресионен анализ.

Използваните формули са дадени в таблицата.

| **Основно уравнение:** | $\mathbf{y=ax^2+bx+c;}$ |
|---|---|
| Средни стойности: | $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n};$ <br> $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n};$ |
| Основни зависимости: | $S_{xx} = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2;$ <br> $S_{xy} = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y};$ <br> $S_{yy} = \sum_{i=1}^{n} y_i^2 - n\bar{y}^2;$ |
| Ковариация: | $q = \frac{S_{xy}}{n-1};$ |
| Коефицент на корелация: | $r = \frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}};$ |
| Коефицент: | $a = \frac{S_{xx}\left(\sum_{i=1}^{n} x_i^2 y_i - \bar{y}\sum_{i=1}^{n} x_i^2\right) - S_{xy}\left(\sum_{i=1}^{n} x_i^3 - \bar{x}\sum_{i=1}^{n} x_i^2\right)}{S_{xx}\left(\sum_{i=1}^{n} x_i^4 - \frac{\left(\sum_{i=1}^{n} x_i^2\right)^2}{n}\right) - \left(\sum_{i=1}^{n} x_i^3 - \bar{x}\sum_{i=1}^{n} x_i^2\right)^2};$ |
| Коефицент: | $b = \frac{S_{xy}\left(\sum_{i=1}^{n} x_i^4 - \frac{\left(\sum_{i=1}^{n} x_i^2\right)^2}{n}\right) - \left(\sum_{i=1}^{n} x_i^2 y_i - \bar{y}\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} x_i^3 - \bar{x}\sum_{i=1}^{n} x_i^2\right)}{S_{xx}\left(\sum_{i=1}^{n} x_i^4 - \frac{\left(\sum_{i=1}^{n} x_i^2\right)^2}{n}\right) - \left(\sum_{i=1}^{n} x_i^3 - \bar{x}\sum_{i=1}^{n} x_i^2\right)^2};$ |
| Коефицент: | $c = \bar{y} - b\bar{x} - \frac{a\sum_{i=1}^{n} x_i^2}{n};$ |

```
% Експериментални данни
x=[6.9,7.6,7.6,9,8.1]
y=[12 10 9 5 6]
% Програма
n=length(x);
xmean=mean(x)
ymean=mean(y)
Sxx=sum(x.^2)-n*(xmean^2);
Sxy=sum(x.*y)-n*xmean*ymean;
Syy=sum(y.^2)-n*(ymean^2);
q=Sxy/(n-1)
r=Sxy/sqrt(Sxx*Syy)
KV=(sum((x.^2).*y));
OW=ymean*sum(x.^2);
KZ=Sxx*(sum(x.^4)-(sum(x.^2))^2/n);
% Коефициенти на уравнението y=ax^2+bx+c
a=(Sxx*(KV-OW)-Sxy*(sum(x.^3)-xmean*sum(x.^2)))/(KZ-(sum(x.^3)-
xmean*(sum(x.^2)))^2)
b=(Sxy*(sum(x.^4)-(sum(x.^2)).^2/n)-(sum(x.^2.*y)-
ymean*sum(x.^2))*(sum(x.^3)-xmean*sum(x.^2)))/(KZ-(sum(x.^3)-
xmean*(sum(x.^2))).^2)
c=ymean-b*xmean-(a*sum(x.^2))/n

% Пример
xx=[7 8]
yy=a*xx.^2+b*xx+c

xx_model=6.8:0.1:9.2;
yy_model=a*xx_model.^2+b*xx_model+c;

plot(x,y,'or',xx,yy,'+',xx_model,yy_model)
xlabel('x')
ylabel('y')
title('Parabolichen model')
grid

xx_model=6.8:0.1:9.2;
yy_model=a*xx_model.^2+b*xx_model+c;

plot(x,y,'or',xx,yy,'+',xx_model,yy_model)
xlabel('x')
ylabel('y')
title('Parabolichen model')
grid
```


Parabolichen model
13
12
11
10
9
8
7
6
5
4
y
6.5
7
7.5
8
8.5
9
9.5
x

# РАЗДЕЛ II. SIMULINK

## 1. ФУНКЦИОНАЛНИ ВЪЗМОЖНОСТИ НА *SIMULINK*

*SIMULINK* е графична среда за симулиране на динамични системи, които се описват като набор от линейни/нелинейни диференциални или диференчни уравнения. Системите се изграждат под формата на блок-диаграми с елементи, които се копират от съпътстващи *SIMULINK* библиотеки със стандартни блокове.

### Подходи за провеждане на симулирането

Съществуват три различни начина за използване на *SIMULINK*:

- *изграждане на собствена блок-диаграма с елементи от библиотеките на SIMULINK и стартиране на опцията*
- ***Simulation***. Това е най-удобният за работа в диалогов режим подход, използващ стандартни менюта и бърза визуализация на резултатите с подходящ блок за това. Препоръчва се при първоначално изследване на системата.
- *описание на системата в SIMULINK и използване на съответните функции за симулиране в командния ред на MATLAB*. Този подход не е диалогов, но осигурява по-голяма гъвкавост от първия, защото използва резултатите от симулирането директно в работното пространствона *MATLAB* с възможности за последващ анализ на системата и по-качествено представяне на нейните характеристики.
- *аналитично описание на системата чрез S-функция и активиране на съответващия й M-файл в средата на MATLAB.* Това е най-сложния и най-гъвкав подход за използване на *SIMULINK*, защото потребителят получава достъп до аналитичните функции, които описват динамичните свойства на системата.

### Провеждане на симулирането чрез опция *Simulation*

Симулирането чрез меню дава възможност да се осъществят в диалогов режим редица операции по време на самия процес:

- промяна на параметрите на блок, ако това не предизвиква изменение на броя на състоянията, входовете или изходите на блока;
- промяна на параметрите на симулиране, с изключение на началния момент на интегриране и променливите, които се връщат в работното пространство с ново съдържание;
- промяна на алгоритъма на интегриране;
- промяна на такта за дискретизация (за дискретни системи);
- симулиране на друга система в същото време;

- активиране на линия от блок-диаграмата за визуализиране на съответния сигнал върху плаващ (несвързан) блок за извеждане на данни.

Извършването на промени в схемата (добиване или изтриване на линия или блок) предизвиква спиране на симулирането, което се подновява чрез ***Restart*** от ***Simulation***.

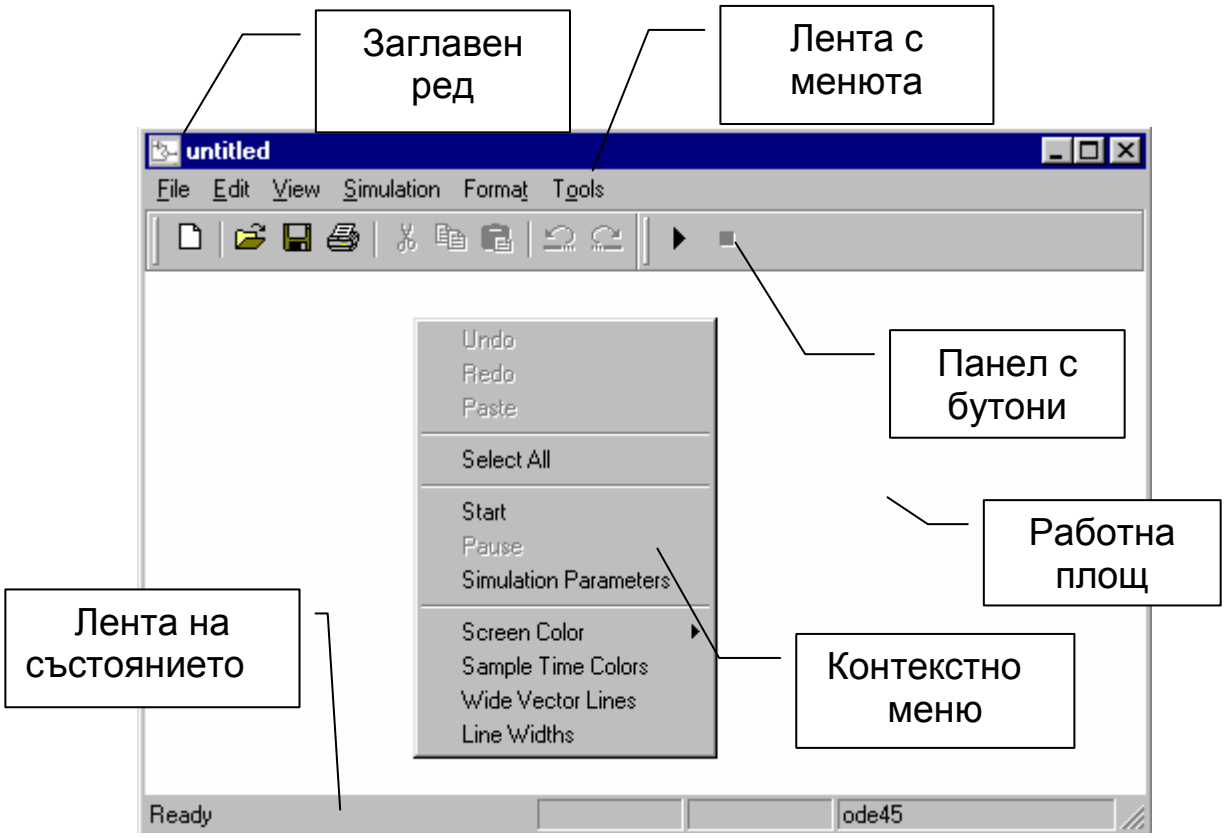
Поради ограничения обем на пособието ще бъде разгледан само този подход.

## 2. СТАРТИРАНЕ НА *SIMULINK*

*SIMULINK* се активира от *MATLAB* по три начина:

- Натиска се бутона с иконата на *SIMULINK* от панела с бутони на *MATLAB*.
- Избира се опцията *Model* на *New* от менюто *File*.
- С командата ***simulink*** от командния ред на *MATLAB*.

След стартиране на екрана на компютъра се появяват: нов празен прозорец за моделиране (с име *Untitled*), фиг. 2.1 и прозорец-меню на основната библиотеката на програмния продукт (фиг. 2.2), чийто елементи представляват условни блокове с разделите на *SIMULINK*.

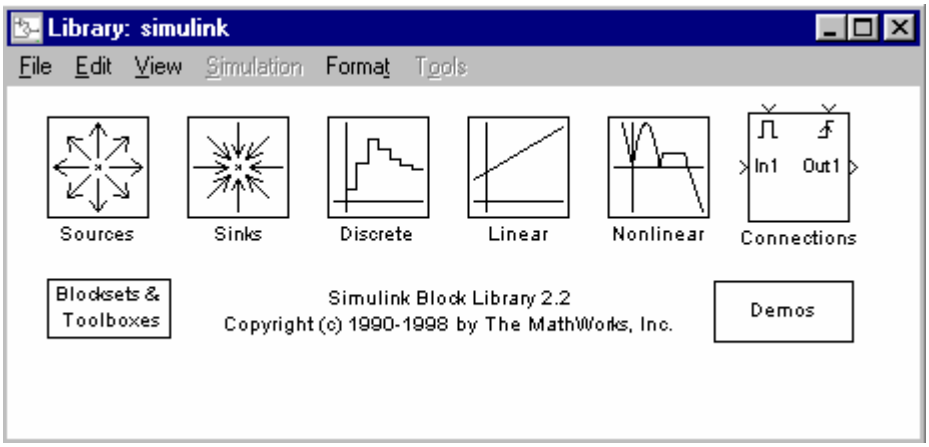



*Фиг. 2.1*

- *заглавен ред* – лента в горната част на прозореца, показваща името му. При издърпването й, прозорецът се премества. Когато заглавният ред е осветен, прозорецът е активен;
- *лента с менюта* - тук са групирани всички функционални възможности на *SIMULINK*. Командите се изпълняват само за елементите на този прозорец;
- *панел с бутони* - по-важните команди се представят чрез бутони с икони, например: отваряне, стартиране и затваряне на модели. Тези команди се изпълняват при натискане на съответния бутон. Например, за отваряне на модел на *SIMULINK* се натиска бутона с икона, представляваща отворена папка. При пре-

местване на курсора върху някой бутон се появява подсказващ текст (*Screen Tip*), описващ функцията на бутона. Панелът с бутоните може да се скрие като се щракне върху опцията *Toolbar* от менюто *View* на *SIMULINK*;

- *контекстно меню* – появява се при натискане на десния бутон на мишката. Съдържанието на менюто зависи от това дали даден блок е избран. Ако не е избран блок, менюто показва командите, прилагани към модела или библиотеката като цяло.
- *лента за състоянието* - намира се в долния край на прозореца. При провеждаща се симулация тя показва параметри на симулацията: изтеклото време на симулация и името на числения метод на интегриране. Лентата може да се скрива с опцията *StatusBar* от менюто *View*.
- *работна площ* – заема най-голямата площ. Цялото проектиране се извършва върху нея.



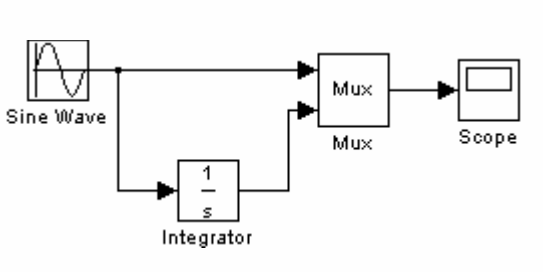

*Фиг. 2.2*

Достъпът до конкретен изпълним в процеса на симулиране блок е организиран в разклонена дървовидна структура на програмния продукт. За по-голямо удобство на ползвателя съвкупности от близки по предназначение блокове са групирани в раздели-подбиблиотеки (показани са в приложение), които са оформени като надстроечни условни блокове. Крайното групиране на условни блокове представлява основната библиотека в *SIMULINK* (фиг. 2.2) и показва икони, които представляват блоковите подбиблиотеки. Изграждането на модел става с копиране на блокове от тези библиотеки в моделния прозорец. *SIMULINK* използва отделни прозорци за показване на модел, библиотека от блокове или графичен симулиран изходен сигнал. Тези прозорци не са фигурните прозорци на *MATLAB* и не могат да бъдат използвани чрез командите за ръчна графика.

При стартиране на симулацията и анализа на резултатите от нея, команди на *MATLAB* може да се въвеждат в командния прозорец на *MATLAB*.

## 3. СЪЗДАВАНЕ НА НОВ МОДЕЛ

Моделът интегрира синусоидна вълна и изобразява двете криви. Създава се в прозореца за моделиране (фиг. 2.1). Блок – диаграмата на модела е показана на фиг. 2.3.



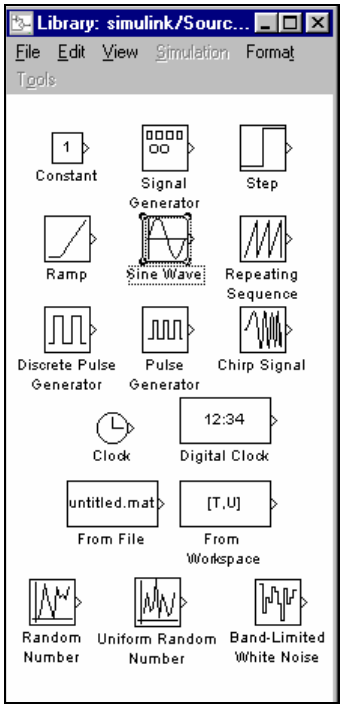

*Фиг.2.3.*

В този модел се вземат блокове от следните библиотеки:

- Източници (генератори) на сигнали (*Sources*).
- Крайни ползватели на сигнали (*Sinks*).
- Линейни непрекъснати системи (*Linear*).
- Свързващи елементи (*Connections*).

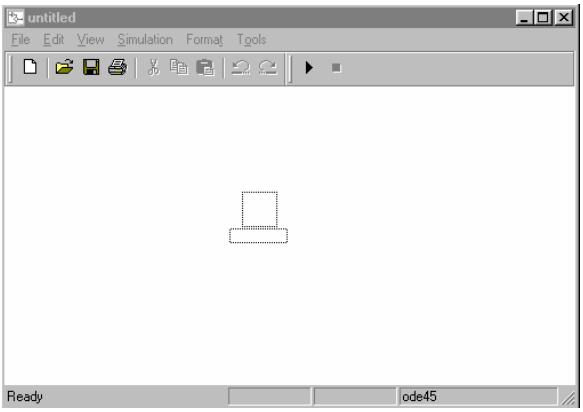
За да се достигне блока *Sine Wave* (синусоидна вълна) първо трябва да се отвори библиотеката на източници (*Source library*). Това става като се щракне два пъти върху иконата на библиотеката. Отваря се прозорец, в който са показани всички блокове в библиотеката фиг. 2.4.

Добавянето на блок върху модела става чрез копирането му от библиотеката или от друг модел. За разглеждания модел е необходимо да се копира *Sine Wave* блока. За да се направи това курсора се позиционира върху *Sine Wave* блока, след което се натиска и задържа бутона на мишката. *Simulink* огражда с линия блока и наименованието му фиг. 2.4.



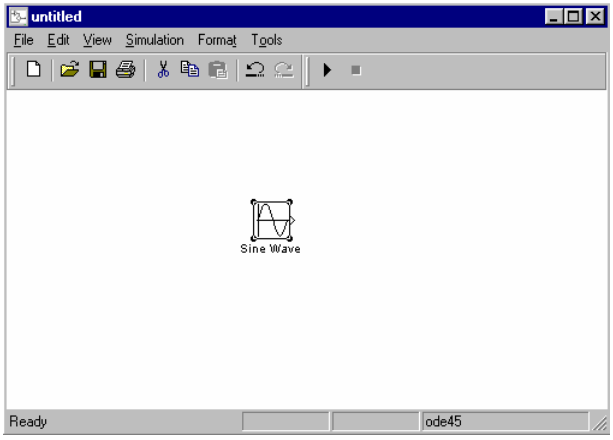

*Фиг. 2.4.*

Блока се премества в моделния прозорец. При преместването се виждат ограждащи линии, които се движат заедно с курсора фиг. 2.5.



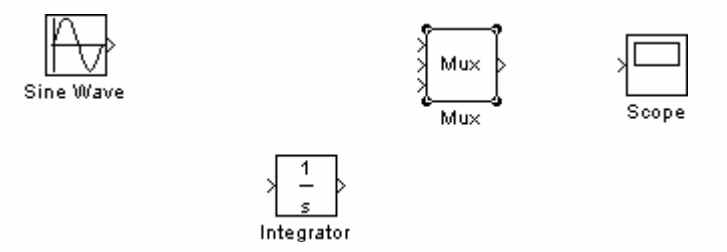

*Фиг. 2.5.*

Когато се достигне желаното място за позициониране на блока, бутона на мишката се отпуска и *Sine Wave* блока е копиран в моделния прозорец фиг. 2.6.



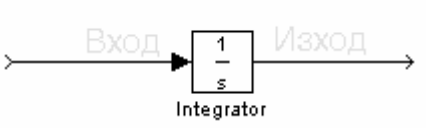

*Фиг. 2.6.*

По същият начин става копирането и на другите блокове фиг. 2.7. За малки премествания се използват стрелките на клавиатурата, след като блока се избере.




*Фиг. 2.7.*

На изображението на блока *Sine Wave* се забелязва ъглова скоба отдясно. Символът > сочещ извън блока е знак за изход на блока. Ако символа сочи към блока – той индицира вход. Сигналът преминава от един изход към входа на друг блок през съединителни линии. Когато блоковете са свързани с такива линии, символите за вход и изход изчезват фиг. 2.8.

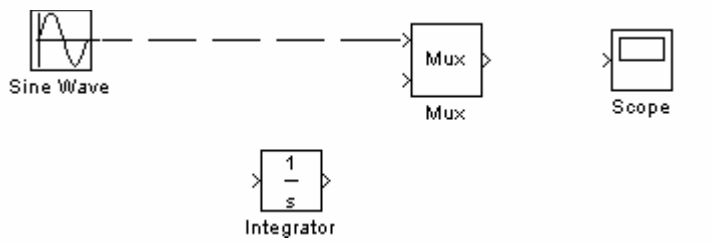



*Фиг. 2.8.*

Забелязва се, че *Mux* блока има 3 входа, но в разглеждания пример са необходими само 2. Броят на входовете може да се промени от диалоговия прозорец на блока, който се отваря при двойно щракване върху него с мишката (фиг. 2.9). Променят се параметрите на опцията “Брой входове“ (*Number Inputs*) на 2 и се натиска бутона *Close*. *SIMULINK* коригира броя на входовете.

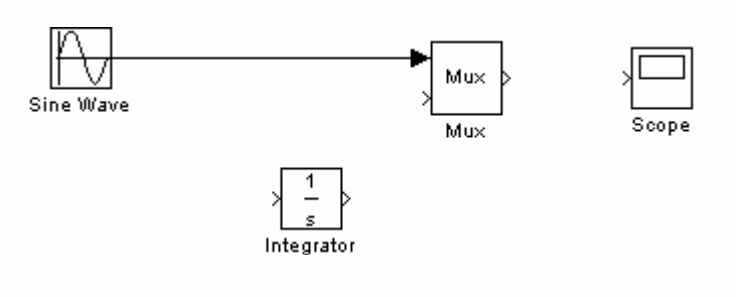

*Фиг. 2.9.*

Свързването на блокове става по следния начин: за да се свърже блока *Sine Wave* с горния вход на блока *Mux*, курсора се позиционира върху изхода, отдясно на блока *Sine Wave*. При позиционирането формата на курсора се променя във вид на кръст. С натиснат бутон на мишката курсора се премества към горния вход на блока *Mux*. Свързващата линия е във вид на прекъсната линия, а курсора дублира кръста при достигане на входа на блока *Mux* (фиг. 2.10).




*Фиг. 2.10.*

Бутонът на мишката се отпуска и блоковете са свързани (фиг. 2.11). Свързването на линии към блок е възможно и когато курсора е в иконата на блока. В такъв случай, линията се свързва с най–близкия до позицията на курсора вход.

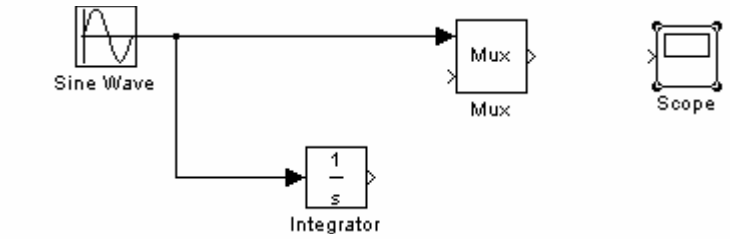



*Фиг. 2.11.*

При разглеждане на показания модел се забелязва, че линиите свързват изход на блок с вход на друг блок. Има линия, която свързва линия с вход на блок. Тази линия се нарича разклоняваща (*Branched*) и свързва изхода на блока *Sine Wave* с входа на блока *Integrator*. По тази линия се предава същия сигнал като този, който постъпва в блока *Mux*.

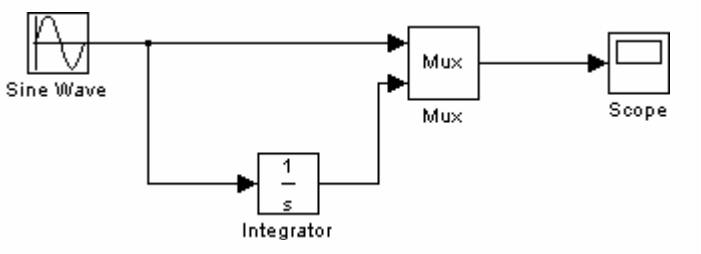
Изчертаването на разклоняваща линия:

- Курсорът се позиционира върху линията, свързваща блока *Sine Wave* с блока *Mux* .
- Натиска се и се задържа клавиша *Ctrl*. След това се натиска бутона на мишката и се премества курсора върху входа на блока *Integrator* или върху самия блок.
- Бутонът на мишката се освобождава и *SIMULINK* изчертава линия от стартовата точка до входа на блока (фиг. 2.12).



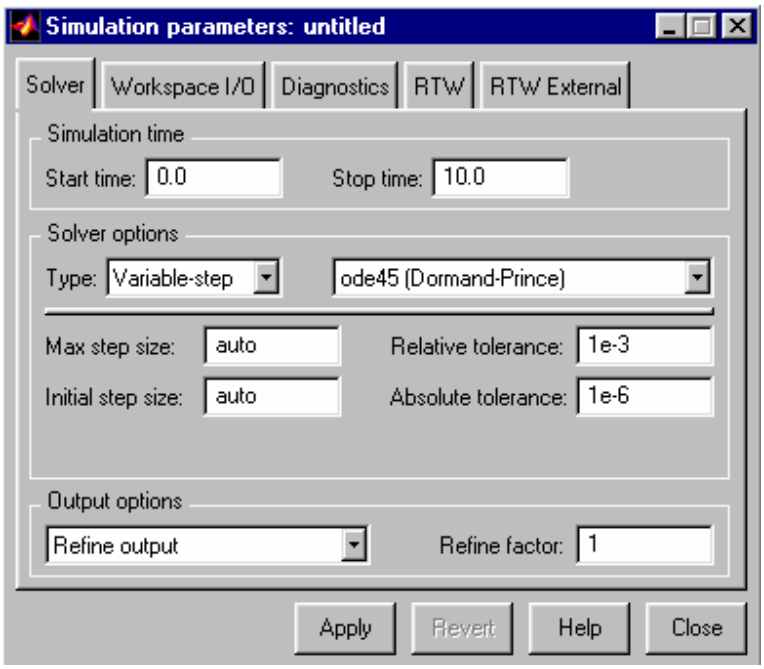

*Фиг. 2.12.*

След поставянето на всички връзки модела би трябвало да изглежда така:




*Фиг. 2.13.*

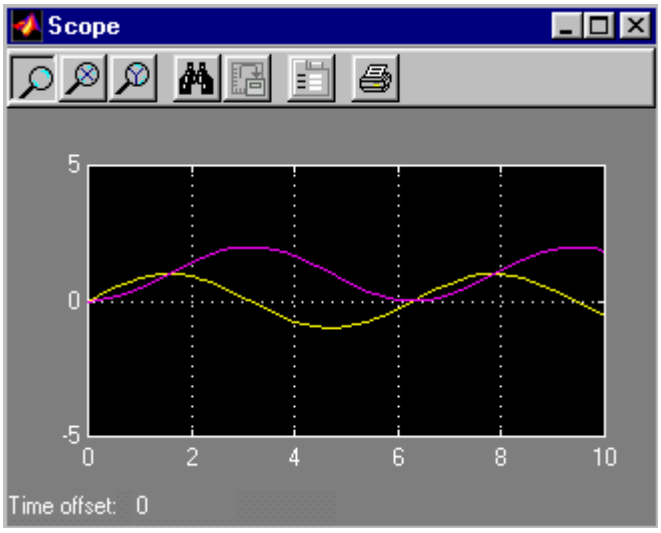
Scope блока се отваря, за наблюдаване на изходния сигнал. При отворен прозорец *Scope*, симулацията се стартира например за 10 сек. Първо трябва да се определят параметрите на симулацията. Това се прави чрез избор на опцията *Parameters* от менюто *Simulation*. В отворилият се диалогов прозорец опцията *Stoptime* е на 10 сек. (това е подразбираща се стойност).




*Фиг. 2.14.*

Затварянето на диалоговия прозорец става чрез натискане на бутона ***Close*** и *SIMULINK* прилага настройките.

Избира се опцията ***Start*** от менюто ***Simulation*** и се наблюдават кривите в графичния прозорец на блока *Scope*.




*Фиг. 2.15*

Симулацията спира, при изтичане на времето определено в диалоговия прозорец *Simulation Parameters*, или при натискане на ***Stop*** от менюто ***Simulation***. Моделът се запазва чрез ***Save*** от менюто ***File*** и се задава име на файла. Файлът ще съдържа описание на този модел.

За прекратяване на работата със *SIMULINK* и *MATLAB* се избира *Exit MATLAB* от менюто ***File***. Може да се излезе и чрез набирането на командата ***quit*** в командния прозорец на *MATLAB*. Може да се излезе от *SIMULINK* без да се прекъсва *MATLAB* чрез затваряне на прозореца *SIMULINK*.

## 4. РЕДАКТИРАНЕ НА СЪЩЕСТВУВАЩ МОДЕЛ

- Избира се командата **Open** от менюто **File**, след което се избира или написва името на модела;
- Написва се името на модела (без разширение .mdl) в командния прозорец на MATLAB. Моделът трябва да е в текущата папка, или да е указан път.

**Отменяне на команди**

Отменянето на ефектите от 101 последователни команди става чрез командата ***Undo*** от менюто ***Edit***. Могат да се отменят следните операции:

- Прибавяне или изтриване на блок
- Прибавяне или изтриване на линия
- Прибавяне или изтриване на обяснителни бележки за модела
- Редактиране на блоково име

Може да се възвърнат ефектите от командата ***Undo***, като се избере ***Redo*** от менюто ***Edit***.

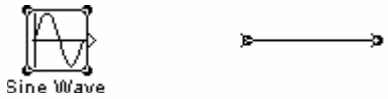
### 4.1. Редактиране на блокове

**Избиране (Селектиране) на обекти**

Много действия при построяването на модели, като копиране или изтриване на линия изискват първо да се изберат един или повече блокове и линии (обекти).

**Избор на един обект.**

За да се избере обект се щраква върху него. Малки черни квадратчета се появяват близо до ъглите на избрания блок или близо до краищата на избраната линия. Фиг. 2.16 показва селектиран блок и селектирана линия:




*Фиг. 2.16.*

При избиране на обект по този начин, всички други избрани преди това престават да бъдат такива.

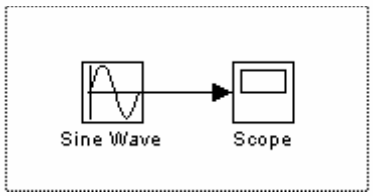
**Избор на повече от един обект един по един.**

Изборът на обекти един по един става чрез натискане и задържане на клавиша Shift и щракване бутона на мишката върху обектите последователно. За отказ от направения избор се щраква повторно върху дадения обект като продължава да се държи натиснат клавиша Shift.

**Избор на повече от един обект едновременно**

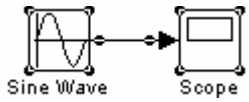
Лесен начин за едновременно избиране на обекти, разположени близо един до друг е чрез изчертаване на ограждащ правоъгълник (Bounding box):

- Определя се началната точка на правоъгълника чрез натискане и задържане на бутона на мишката в избрания ъгъл. Формата на курсора се променя.
- Курсорът се премества до срещуположния ъгъл на правоъгълника. Точковия правоъгълник затваря в себе си избраните блокове и линии (Фиг. 2.17).




*Фиг. 2.17.*

- Бутона на мишката се отпуска. Всички блокове и линии в правоъгълника са избрани (Фиг. 2.18).




*Фиг. 2.18.*

**Копиране и преместване на блокове от един прозорец в друг**

При построяване на модел често се налага копирането на блокове от библиотеките на *SIMULINK*, от други библиотеки или модели. За да се направи това е нужно:

- Отваря се прозорецът на необходимата библиотеката или модел.
- Нужният блок се премества в желания прозорец: блока се маркира; с натиснат бутон на мишката курсора се премества в другия прозорец и бутона на мишката се освобождава.

Копиране на блокове може да се извърши и чрез командите ***Copy*** и ***Paste*** от менюто ***Edit*** по следния начин:

- Селектира се блока, който ще се копира.
- Избира се командата *Copy* от менюто *Edit*.
- Активира се прозорецът, в който ще се копира блока.
- Избира се командата *Paste* от менюто *Edit*.

*SIMULINK* дава име на всеки копиран блок. Ако копираният блок е първи от даден тип за модела, неговото име е същото като в прозорецът източник. Например при копиране на блок *Gain* (Усилване) от библиотеката *Linear* (Линейни непрекъснати системи), в желания модел името на блока е *Gain*. Ако в модела съществува вече такъв блок, *SIMULINK* поставя последователен номер след името му (*Gain1*, *Gain2* и т.н.). ). При копиране на блок, новият блок получава стойностите на параметрите на оригинала. Имената на блоковете могат да се променят.

*SIMULINK* използва невидима 5–пикселова решетка, която улеснява подравняването на моделите. Всички блокове в модела са "залепени" за линии от решетката. Изместването на блок в прозореца става с мишката или чрез маркирането му и използвайки клавишите със стрелки на клавиатурата.

Може да се копират или преместват блокове в други приложения (напр. *Microsoft Word*), чрез командите ***Copy***, ***Cut*** и ***Paste*** от менюто ***Edit***. Тези команди копират само графичното изображение на блока, без неговите параметри. Преместването на блокове от един прозорец в друг става по същия начин като копирането им, само че клавиша *Shift* се държи натиснат при селектирането им.

Командата ***Undo*** може да се използва за изваждане на прибавен блок.

**Преместване на блокове вътре в модела**

За преместване на един блок от едно място на друго в модела, блока се изтегля до новото местоположение. *SIMULINK* автоматично препозиционира линиите, свързани с изместения блок.

Изместването на блокове, заедно със свързващите ги линии:

- Маркират се блоковете и линиите, по един от начините описани в "Избиране на повече от един обект ".
- Изтеглят се обектите до желаното им ново местоположение и се освобождава бутона на мишката.

**Размножаване на блокове в модел**

С натиснат клавиш *Ctrl* се маркират блоковете с левия бутон на мишката, след което се изтеглят до новото местоположение. Другият начин е чрез изтеглянето им при натиснат десен бутон на мишката.

Размножените блокове имат същите параметри като оригиналните. Към имената им са прибавени последователни номера.

**Определяне на параметрите на блоковете**

Потребителският интерфейс на *SIMULINK* позволява задаването на стойности на параметрите на блока. Начинът на задаване зависи от вида на параметъра.

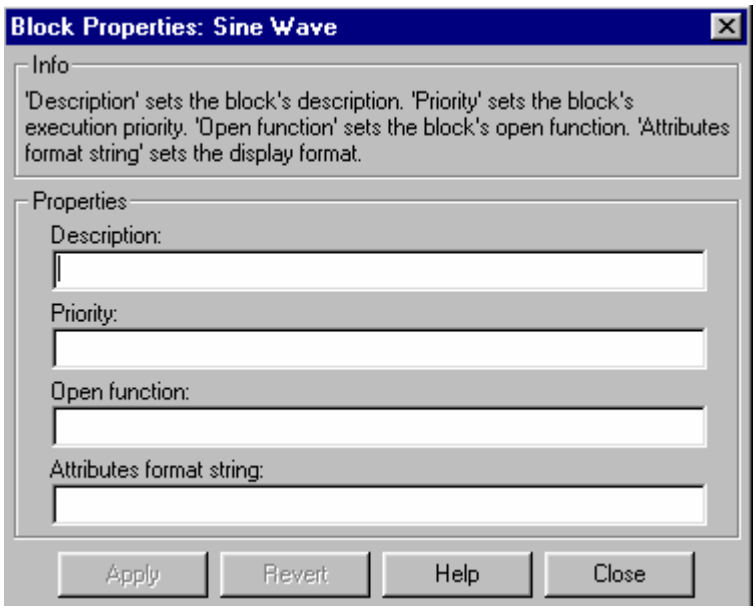
**Задаване на диалоговите параметри**

Някои аспекти от функциите на блока се определят от параметрите в диалоговия прозорец на блока. Възможно е въвеждането на стойности за всеки параметър в диалоговия прозорец на блока.

Двойното щракване върху блока отваря диалоговият му прозорец. Дадените стойности на параметрите могат да се приемат или променят.

**Определяне ( задаване ) на основните параметри на блока.**

*SIMULINK* позволява да се определят (задават) някои основни параметри на блока в диалоговия прозорец *Block Properties*. За да се направи това, първо се маркира блока, чиито параметри ще се променят. След това се избира опцията *Block Properties* от менюто *Edit*. Появява се диалоговият прозорец *Block Properties* (фиг. 2.19).




*Фиг. 2.19.*

Той съдържа следните полета:

- *Description* (Описание). Текуща стойност на блоковия параметър *Description*. Този параметър ясно описва целта на блока.
- *Priority* (Приоритет). Текуща стойност на блоковия параметър *Priority*. Този параметър определя изпълнимия приоритет на блока.
- *Open Function* (Отворена функция). Текуща стойност на блоковия параметър *Open Function*. Отворената функция е *m* – функцията, която *SIMULINK* извиква при отваряне на блок.
- *Attribute Format String*. Текуща стойност на блоковия параметър *Attribute Format String*. Този параметър определя какво да се изписват под иконата на блока.

За смяна на даден параметър се редактира съответното поле и се натиска бутона ***Apply***. За възстановяване на предишните стойности се натиска бутона ***Revert***.

**Изтриване на блокове**

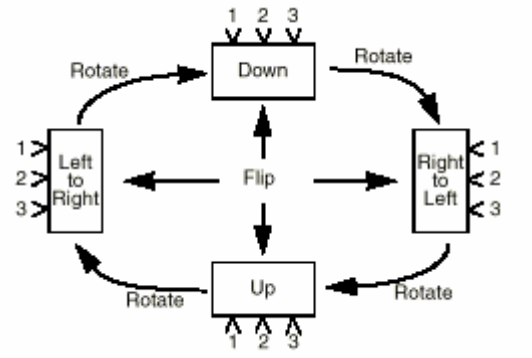
За да се изтрият един или повече блокове, първо се маркират, след което се натиска клавиша ***Delete*** или ***Backspace*** на клавиатурата. Може също да се изтриват чрез командите ***Clear*** или ***Cut*** от менюто ***Edit***. Командата ***Cut*** записва блоковете в клип борда (*Clipboard*), което позволява блока да бъде копиран по–късно в друг модел. Използвайки клавишите ***Delete*** или ***Backspace*** копирането на блоковете по–късно не е възможно.

Може да се възстанови изтритият блок с командата ***Undo*** от менюто ***Edit***.

**Променяне ориентацията на блок**

По подразбиране, сигналният поток през един блок тече от ляво на дясно. Входовете на блока са от ляво, а изходите – отдясно. Ориентацията на блока може да се смени като се избере една от следните команди от менюто *Format*:

- Командата ***Flip Block*** завърта блока на 180°
- Командата ***Rotate Block*** завърта блока на 90° по часовниковата стрелка
- Фиг. 2.20 показва как *SIMULINK* подрежда изходите и входовете след смяната на ориентацията, използвайки командите ***Flip Block*** и ***Rotate Block*** Текстът в блоковете показва тяхната ориентация.

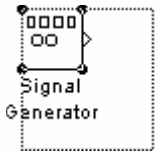



*Фиг. 2.20.*

**Промяна на размера на блоковете**

Блокът се избира. С натиснат бутон на мишката се изтегля едно от неговите квадратчета, появили се в ъглите на избрания блок. Правоъгълникът с прекъснати линии показва новия размер на блока. След освобождаването на бутона на мишката блока е с новият си размер.

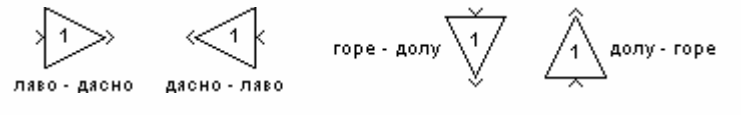
Фиг. 2.21 показва блок *Signal Generator* (Сигнал генератор) при промяна на размера му. Долното дясно квадратче е изтеглено до позицията на курсора. При освобождаване бутона на мишката блока добива новия размер.




*Фиг. 2.21.*

**Промяна на имената на блоковете**

Всички имена на блокове в модел трябва да имат различни имена, съдържащи поне 1 символ или буква. Имената се изписват под блоковете, когато входовете и изходите на блока са отляво и отдясно. Когато са отгоре и отдолу, името се изписва отляво.




*Фиг. 2.22.*

**Смяна на името на блока**

Смяната става по един от следните начини:

- маркира се цялото име, след което се въвежда новото;
- щраква се 2 пъти върху текста, след което се коригира.

При щракване на друго място в модела или при друго действие, новото име се приема или отхвърля. При смяна на име със съществуващо вече такова *Simulink* извежда съобщение за грешка.

Може да се смени стила (*Font*) на изписване на името като се избере опцията ***Font*** от менюто ***Format***. Стилът се избира от диалоговия прозорец *SetFont*. Тази процедура сменя и стила на текста върху иконата на блока.

Тази команда се отменя с командата ***Undo*** от менюто ***Edit***.

**Забележка:** При смяна на име на блок в библиотека, всички връзки с този блок стават несигурни.

**Промяна на позицията на името на блока**

Промяната може да се извърши по 2 начина:

- Маркира се името и се изтегля до срещуположната страна на блока.
- Избира се командата ***FlipName*** от менюто ***Format***. Тази команда измества името на срещуположната страна на блока.

**Скриване на името**

За да се скрие името на блока се избира опцията ***HideName*** от менюто ***Format***. При избирането тя автоматично се променя на ***Show-Name***.

Опцията ***ShowName*** показва отново името на блока.

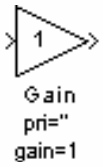
**Показване на параметрите под иконата на блока**

*SIMULINK* може да изписва един, или повече от параметрите на блока, под неговата икона в блок – диаграма. Кои параметри да бъдат изписани се определя чрез въвеждане на форматен текст в полето *Attribute Format String* на диалоговия прозорец *Block Properties*.

Форматният текст може да бъде всеки текст включващ имената на вградените параметри на блока. Тези имена на параметри се предхождат от %< и следвани от >, например, %<priority>. *SIMULINK* показва форматния текст под иконата на блока, заменяйки името на параметъра с отговарящите стойности на дадения параметър. Показването на всеки параметър на отделен ред става чрез използването на \n.

**Пример:**

Определянето на форматния текст
pri = %<priority>\ngain = %<Gain>
за блока Gain показва




*Фиг. 2.23.*

Ако стойността на параметъра не е текст или цяло число (*integer*), *SIMULINK* изписва N/S (не поддържан) за стойността на параметъра. Ако името на параметъра е невалидно *SIMULINK* показва “???”.

**Отделяне на блокове**

За отделяне на блокове от свързващите ги линии блока се изтегля до ново местоположение, като се държи натиснат клавиша *Shift*.

**Векторен вход и изход**

Понеже блоковете на *SIMULINK* приемат скаларни или векторни входни сигнали, генерират скаларни или векторни изходни сигнали, то е допустимо да се определят векторни или скаларни параметри. Тези блокове са описани в пособието като “векторизирани“.

Може да се определи кои линии в модела пренасят векторни сигнали, чрез избиране на опцията ***Wide Vector Lines*** от менюто ***Format***. При избиране на тази опция линиите, пренасящи векторни сигнали, се изчертават по–тънки от тези, пренасящи скаларни сигнали.

Ако след избирането на опцията ***Wide Vector Lines*** модела се променя, то е наложително да се избере командата ***Update Diagram*** от менюто ***Edit***. Стартиране на симулацията също отбелязва промените на екрана.

**Определяне приоритета на блока**

На невиртуалните (*non - virtual*) блокове в един модел могат да се определят приоритетите. Блоковете с по–голям приоритет се оценяват (*evaluate* – определяне, изчисление) преди блоковете с по–нисък, но не задължително преди тези с неопределен приоритет.

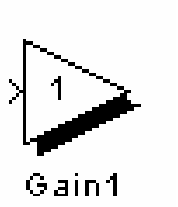
Определянето на приоритетите може да стане интерактивно или програмно. За задаване на приоритетите по програмен път се използва командата:

set_param (b,'priority','n'),

където *b* е блока, *n* е всяко валидно цяло число ( 0 и отрицателните числа също са валидни стойности за приоритета ). Колкото е по–малко числото, толкова е по–голям приоритета (например 2 е по–голям приоритет от 3 ). За интерактивно задаване на приоритета на блока се задава желаната стойност в полето *Priority* на диалоговия прозорец *Block Properties*.

**Използване на *Drop Shadows* (добавяне на сянка)**

Блокът първо се избира, след което се избира опцията ***Show Drop Shadow*** от менюто ***Format***. При избиране на блок с *Drop Shadow*, опцията на менюто се променя на ***Hide Drop Shadow***.




*Фиг. 2.24.*

**4.2. Редактиране на линии**

Линиите пренасят сигнали. Всяка линия пренася векторен или скаларен сигнал. Линията може да свързва изхода на един блок с входа на друг или да свързва изхода на блок с входовете на много блокове.

**Изчертаване на линия между блокове**

Свързване на изход на блок с вход на друг блок:

- Курсорът се позиционира върху знака за изход на първия блок. Не е задължително прецизно да се позиционира курсора върху изхода. Формата на курсора се променя на кръст.
- Натиска се и се задържа бутона на мишката.
- Курсорът се премества до входа на другия блок. Може да се позиционира близо до входа или в блока. При позициониране върху блока линията се свързва с най – близкия вход. Формата на курсора се променя на двоен кръст.

Отпуска се бутона на мишката. *SIMULINK* заменя знаците за вход и изход на блоковете със свързваща линия със стрелка в единия край, която показва посоката на движение на сигнала. Може да се създават линии или от изход към вход, или от вход към изход.

*SIMULINK* изчертава линии, използвайки вертикални и хоризонтални сегменти. За изчертаване на диагонална линия, клавиша *Shift* се държи натиснат при изчертаването на линията.

**Изчертаване на разклоняваща линия**

Разклоняваща е линията, която започва от съществуваща вече линия и пренасяща нейния сигнал до вход на блок. И двете линии пренасят един и същ сигнал. Използването на разклоняваща линия позволява сигнала да бъде пренасян до повече от един вход.

За изчертаване на такава връзка се прави следното:

- Курсорът се позиционира върху линията, от която трябва да започне разклоняващата линия.
- Натиска се и се задържа клавиша *Ctrl*. След това се натиска и се задържа бутона на мишката.
- Курсорът се премества върху входа на желания блок и бутона на мишката се освобождава, както и клавиша *Ctrl*.

Може да се използва и десния бутон на мишката вместо клавиша *Ctrl*.

**Изчертаване на сегмент (част) на линия**

При необходимост, може да се изчертае линия със сегменти, които да са разположени на необходимите места, независимо къде ги изчертава *SIMULINK*. Може да се изчертае и линия преди копирането на блока, към който ще бъде свързана. И двете операции се извършват чрез чертане на линия от сегменти.

Изчертаването на такава линия става по следния начин: изчертава се линия, края на която не е свързан с елемент от диаграмата. Появява се стрелка на несвързания край на линията. Добавянето на нов сегмент става като се позиционира курсора върху края на първия сегмент и се изчертава нов сегмент. *SIMULINK* чертае сегментите хоризонтални или вертикални. За изчертаване на диагонален сегмент се държи натиснат клавиша *Shift* по време на чертането.

**Преместване на сегмент**

За преместването на сегмент е нужно:

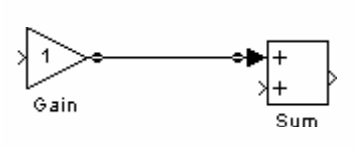
1. Курсорът се позиционира върху сегмента, който ще се премества.
2. Натиска се и се задържа бутона на мишката.
3. Курсорът се премества до желаното местоположение.
4. Бутонът на мишката се отпуска.

Не могат да бъдат местени сегменти, свързани директно към вход или изход на блок.

**Делене на линия на сегменти**

Една линия може да се раздели на сегменти, като краищата й остават на същото място. *SIMULINK* създава нов възел с два сегмента. За да се раздели линия на сегменти трябва:

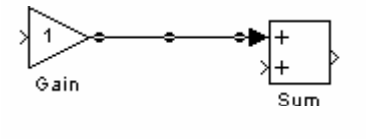
1. Избира се линията.




*Фиг. 2.25.*

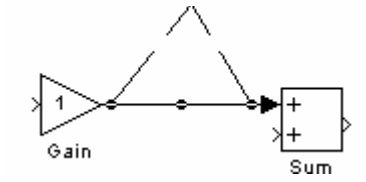
2. Курсорът се позиционира върху линията, на мястото където ще бъде възела.

3. При натиснат клавиш *Shift* се натиска и задържа бутона на мишката. Формата на курсора се сменя на кръг, чийто център е новия възел.



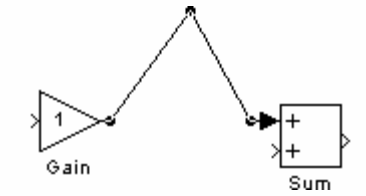

*Фиг. 2.26.*

4. Курсорът се премества до желаното местоположение на върха.




*Фиг. 2.27.*

5. Освобождават се клавиша *Shift* и бутона на мишката.




*Фиг. 2.28.*

**Преместване на възел**

За да се премести възел на линия се прави следното:

1. Позиционира се курсора върху възела, след което се натиска и задържа бутона на мишката. Формата на курсора се сменя на кръг.

2. Курсорът се премества до желаното местоположение на възела.

3. Бутонът на мишката се освобождава.

**Изобразяване на ширината (дебелината) на линиите.**

Може да се покаже ширината (дебелината) на всяка една от линиите в модела като се избере опцията ***Line Widths*** от менюто ***Format***.

*SIMULINK* отбелязва големината на всеки сигнал в блока, който го излъчва и в блока, който го приема.

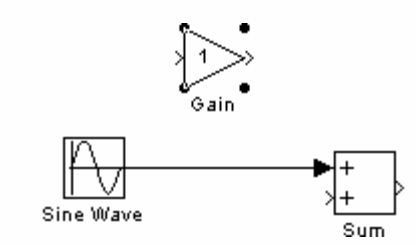
При обновяване на диаграмата или при стартиране на симулация ако *SIMULINK* открие несъответствие на входовете и изходите, показва съобщение за грешка и посочва дебелините на линиите в модела.

**Вмъкване на блок в линия**

Вмъкването на блок в линия става, като се постави блока на линията. *SIMULINK* вкарва блока в точката, на която е поставен на линията. Вмъкнатият блок трябва да има само един вход и един изход.

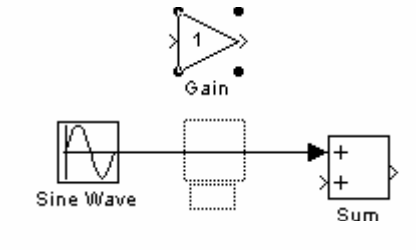
За да се вкара блок в линия трябва:

1. Желаният блок се избира.



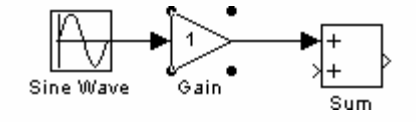

*Фиг. 2.29.*

2. Премества се върху линията, в която трябва да се вмъкне.




*Фиг. 2.30.*

3. Освобождава се бутона на мишката. *SIMULINK* вмъква блока на мястото където е поставен.




*Фиг. 2.31.*

**Поставяне на етикети (надписване) на сигнали**

Може да се поставят етикети на сигналите за да има обяснителни бележки модела. Етикетите се изписват над хоризонталните линии / или сегменти на линии /, или от ляво или дясно на вертикалните линии или сегменти. Етикетите могат да се поставят в един от двата края на линията, в центъра на линията, или в която и да е комбинация от тези положения.

**Използване на етикети на сигнали**

За да се постави етикет на сигнал се щраква 2 пъти върху дадената линия или сегмент и се набира желания текст. При щракване на друго място от модела, етикета се позиционира на мястото си.

**Забележка:** Трябва да се щраква точно върху линията. При щракване близо до линията се създава бележка за самия модел.

Етикетът се мести като се избере, след което се премества до желаното местоположение. При освобождаване на бутона на мишката, етикета фиксира положението си близо до линията.

Етикетът се копира, когато е натиснат клавиша *Ctrl*, докато се премества етикета до ново местоположение. При освобождаване на бутона на мишката, етикетът се появява и на оригиналното и на новото място.

За редактиране на етикет на сигнал е нужно:

- За смяна на етикет се маркира етикета, щраква се два пъти върху него, след което се набира новият текст.
- За въвеждане на отделни писмени знаци се щраква 2 пъти между двете букви или символа, където трябва да се направи промяната, след което се въвежда новия текст.
- За смяна на писмени знаци се маркира нужната част от текста и се въвежда новия текст.

# РАЗДЕЛ III. LABVIEW

## 1. ВЪВЕДЕНИЕ В *LABVIEW*

Разгледани са терминологията и елементите, които се използват за създаване и изпълнение на *LabVIEW* програми.

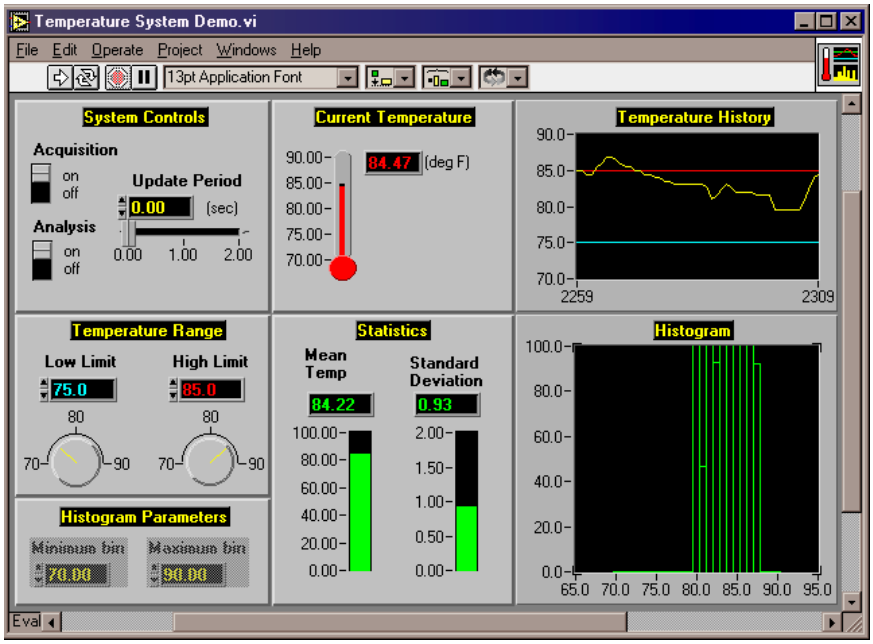
### 1.1. Виртуални инструменти

*LabVIEW (Laboratory Virtual Instrument Engineering Workbench)* представлява среда за разработване на приложения подобно на модерните *C* и *BASIC* среди. Но *LabVIEW* се различава от тях по един основен показател – докато другите системи използват текстови езици за програмиране, то *LabVIEW* използва графичен програмен език *G*.

*LabVIEW* , подобно на *C* или *BASIC* е програмна система с общо предназначение, притежаваща мощни библиотеки от функции. *LabVIEW* включва библиотеки за приемане, анализ, представяне и съхраняване на данни. Средата извършва обмен на информация с хардуер, поддържащ стандартите GPIB, VXI, PXI, RS-232, RS-485.

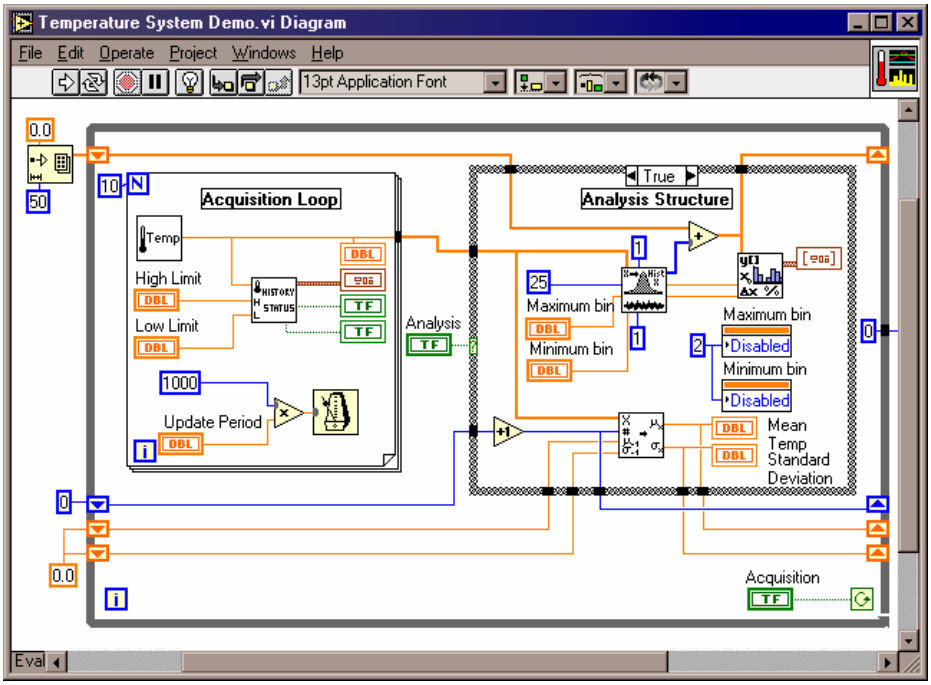
Програмите, които се разработват с *LabVIEW* се наричат виртуални инструменти (*virtual instruments*), тъй като техният вид и действие имитират реални устройства. Виртуалните инструменти (*VI*) се състоят от интерактивен потребителски интерфейс, диаграма на потока от данни (код на програмата) и икона с връзки, която дава възможност на *VI* да се извиква като подпрограма от други *VI*:

Интерактивният потребителски интерфейс на *VI* се нарича ***преден панел***, тъй като наподобява панела на физическо устройство. Предният панел (фиг.3.1) може да включва бутони, плъзгачи, графичен дисплей и други входни полета и индикатори. Чрез клавиатурата или мишката може да се въвежда информация и след това да се видят резултатите на екрана.




*Фиг. 3.1. Преден панел на виртуален инструмент*

***Предният панел*** предлага интерактивен интерфейс за връзка на входове и изходи с разработвана инструментална система. Когато виртуалният инструмент бъде завършен, неговият предния панел може да се използва за управление на системата чрез превключване на "захранването", преместване на прозореца, превключване на бутони или въвеждане на стойност от клавиатурата. Панелът отговаря моментално, като предоставя обратна връзка от системата в реално време.




*Фиг. 3.2. Блокова диаграма на виртуален инструмент*

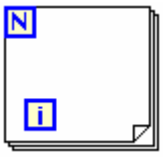
Виртуалният инструмент изпълнява инструкции, създадени в диаграмата на потока от данни (***блокова диаграма***) с помощта на езика за програмиране ***G***. Блоковата диаграма представлява програмния код на *VI*. При нейното конструиране не се налага съобразяване със синтактичните подробности на традиционните езици за програмиране. За създаването ѝ се използват функционални блокове от палетите на меню *Functions*. След това тези функционални блокове се свързват с линии, наречени жици (*wires*), указващи начина на преминаването на данните от един блок към следващия. Тези блокове могат да бъдат аритметични и логически функции, функции за управление, съвременни *VI* за събиране на данни, *VI* за анализи или входно-изходни функции за работа с файлове. Последните могат да запазват или получават данни в *ASCII*, двоичен или дори произволни формати. Всеки блок се представя във вид на икона. Потоците на данните се движат по жиците, които свързват крайните стойности и иконите. Типът на данните, определя и начина, по който се изчертават жиците. Например: тънки линии означават цифрови константи, дебели линии означават масив от цифрови данни, а много дебели линии означават магистрални данни.

### 1.2. Модулност и йерархия

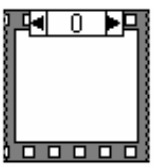
*LabVIEW* е проектиран на модулен принцип. По този начин *VI* могат да бъдат използвани като вътрешни *VI* в блоковите диаграми на други *VI*. Всяка софтуерна система може да бъде разбита на вътрешни *VI*, които първо да бъдат тествани поотделно, а след това да бъдат използвани като възли в основната диаграма. По този начин като се използва модулната йерархия може да се проектират, променят и комбинират *VI,* за да бъдат задоволени променящите се потребителски нужди. Йерархията разширява възможностите на *VI*. Чрез създаване на икона на потребителски *VI* и използването му в диаграмата на друг *VI*, могат да не се показват подробности от схемата на диаграмите от по-ниско ниво. Панелите на тези вътрешни *VI* могат да бъдат конфигурирани да се отварят автоматично, като се създадат анимирани, контекстно-чувствителни потребителски интерфейси. Тези възможности позволяват да се извършва т.нар. модулно програмиране – една задача се разбива на множество подзадачи, които се реализират като отделни *VI*. След това тези *VI* се обединяват в един общ *VI* представящ задачата.

### 1.3. Програмиране с използване на езика *G*

***G*** представлява графичен език за програмиране на потока от данни. Потокът данни е една от най-новите програмни концепции при извършване на едновременни операции. Често е необходимо да се осигури и определен ред на изпълнение. *LabVIEW* представлява завършена система за програмиране. Тя предлага програмни структури като итеративни цикли, условни цикли и условни оператори за последователно, повтарящо се изпълнение или за разклоняване. Тези структури се появяват като графични рамки, които ограждат иконите, които те контролират. Някой основни функции на *G* са за работа с:




***Цикли*** (*Loops*) – G притежава две структури за циклично изпълнение на инструкции – цикъл *While* и цикъл *For*. И двете структури представляват обекти с променлива големина, в рамката на които потребителя може да помества поддиаграми. Операторът *For Loop* изпълнява операциите, намиращи се в неговата рамка определен брой пъти, като обикновено тази стойност се получава чрез свързване със стойност за край *N* (*count terminal*). Броячът на итерациите *i* показва текущата итерация. Той приема стойности от 0 до *N-1*.




***Условна*** (*Case*) и ***последователна*** (*Sequence*) управляващи структури. *Case* представлява условна структура, която изпълнява дадена поддиаграма в зависимост от входната стойност. Структурата за последователност (*Sequence structure*) изпълнява многобройни поддиаграми в своята рамка, в последователност определена от тяхната числена стойност. В традиционните езици за програмиране, последователността

на изпълнение се определя от последователността на текста. В *LabVIEW* обектите могат да се изпълняват паралелно, ако няма даннова връзка помежду им. Структурата за последователност се използва, за да определи изпълнението на възли, които трябва да бъдат изпълнени в определен ред, но не обменят данни един с друг.

***Масиви***, ***записи*** и ***графики***. Масивите представляват набор от данни от еднакъв тип. Записите представляват статичен набор от данни, които могат да бъдат еднотипни или от различен тип. Графиките се използват за извеждане на данни в графичен вид.

### 1.4. Програмиране на потока данни

Функционалните икони и структури, като *For Loop*, се наричат възли. Всеки възел започва работа само, когато има данни на всички негови входове. Когато възелът завърши изпълнението, той извежда данни на всички свои изходи. Този метод за управление изпълнението на данните се нарича поток на данните.

Програмирането на потока данни освобождава потребителя от линейната архитектура на текстово базираните езици за програмиране. Тъй като последователността на изпълнение в *LabVIEW* се определя от потока на данните между възлите, а не от последователни редове текст, то могат да бъдат създавани диаграми, които имат много пътища на данни и едновременно извършване на операции. *LabVIEW* изпълнява независимите пътища на виртуалните инструменти едновременно.

# 2. СЪЗДАВАНЕ НА ВИРТУАЛНИ ИНСТРУМЕНТИ

В тази глава се разглежда въпросът по създаване на собствени *VI*. Тя съдържа подробна информация за обектите на *LabVIEW*, как да бъдат конфигурирани и как да бъдат използвани заедно, за да бъдат създадени виртуални инструменти.

## 2.1. СЪЗДАВАНЕ НА ЦИФРОВ ТЕРМОМЕТЪР

Разглеждат се основните начини за създаване на *VI*, като се създаде *VI*, който моделира цифров термометър.

### 2.1.1. Симулиране и експериментално първоначално установяване

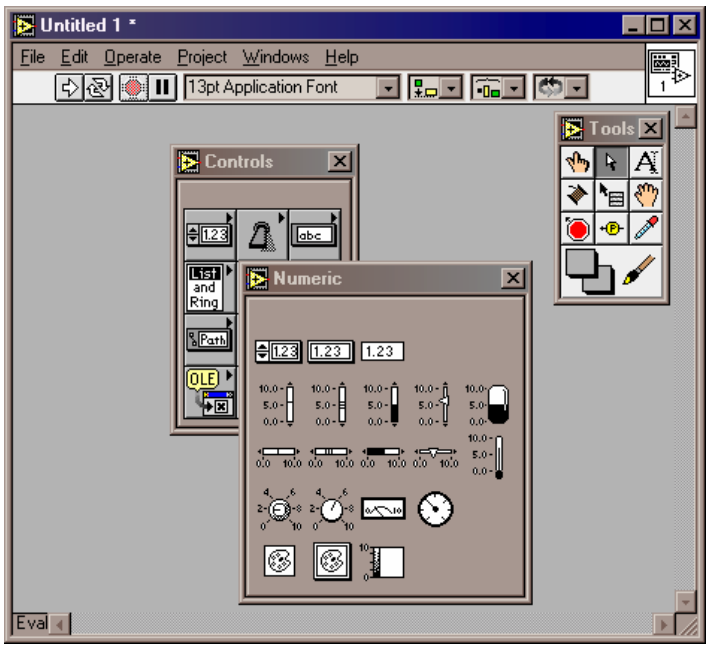
Да предположим, че трябва да се измери температура в камера. Нека имаме преобразувател или сензор, който превръща температурата в напрежение. Сензорът да е свързан към платка която превръща напрежението в цифрови данни.

- Ако *LabVIEW* не е стартиран, стартирайте го. Върху прозореца на диаграмата се появява ненаименован прозорец на панела.

### 2.1.2. Създаване на панел

Първо трябва да бъде създаден преден панел с термометър, който да визуализира температурата.

- Позиционирайте курсора близо до центъра на прозореца на панела. Щракнете и задръжте натиснат десният клавиш на мишката, за да задействате падащо управляващо (*Controls*) меню. Използването на падащи менюта в *LabVIEW* е предпочитан метод за създаване и променяне на обекти.
- Продължете да натискате десния клавиш на мишката докато придвижите стрелката върху ред *Numeric* и след това върху, таблото, което се появява (фиг.3.3). Когато стрелката преминава през всяка икона в горния край на таблото се появява нейното име. Отпуснете бутона на мишката, когато стрелката е позиционирана върху иконата на термометъра, както е показано на следващата фигура.




*Фиг. 3.3. Табло с полета и обекти в LabVIEW*

- Над цифровия дисплей на термометъра се появява етикет (малък правоъгълник). Напишете *Temperature*. Ако етикетът изчезне, извикайте менюто на термометъра като щракнете върху него с десния бутон на мишката и изберете *Show label* (Покажи етикета), след което напишете *Temperature*. Натиснете клавиш <*Enter*> от клавиатурата. Скалата по подразбиране за индикатора на температурата е в интервала 0.00 до 10.00. Тя може да бъде променена като се изберат крайните стойности и на тяхно място се въведат нови.
- Щракнете върху иконата на инструмента за писане A (*Labeling tool*), намиращ се в панел *Tools* (ако панел *Tools* не е визуализиран, първо от меню *Windows* изберете командния ред *Show Tools Palette*). Изберете стойността 0.00 като еднократно щракнете върху нея, напишете 70.0 и натиснете <*Enter*> от клавиатурата (или щракнете с левия бутон върху свободно място на панела). По същия начин променете 10.0 на 90.0. *LabVIEW* автоматично преизчислява междинните стойности.

За да бъде визуализирана стойността на температурата, трябва да се получи напрежение, след което да се преобразува в градуси, като се използва подходящ за типа на сензора метод. *VI* трябва да бъде програмиран да изпълни тези операции като се създаде негова диаграма. За този пример да допуснем, че сензорът е линеен и обикновен фактор на мащабиране е достатъчен.

- Изберете *Show Diagram* от *Windows* менюто. Активен прозорец става ненаименованата диаграма. Тя вече съдържа означението на температурния индикатор и неговия етикет, които бяха създадени преди това. Ако етикетът е невидим,

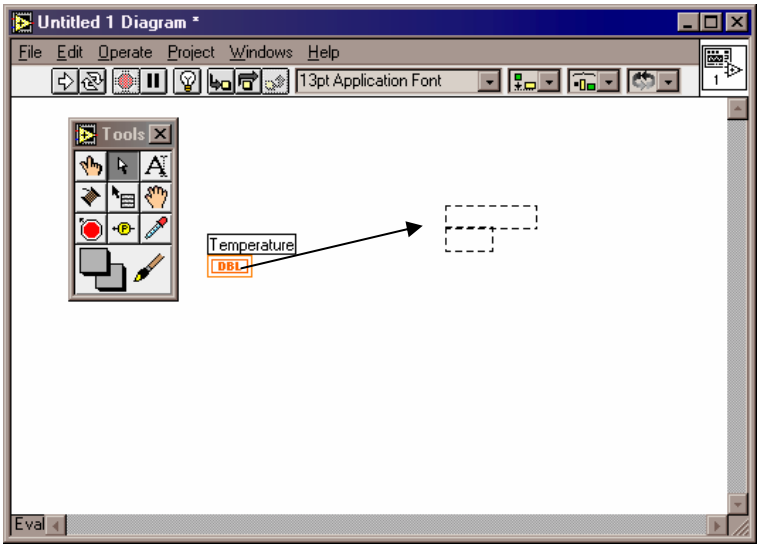
извикайте менюто за температурния индикатор и изберете *Show Label.*

### 2.1.3. Работа с обекти

- Изберете инструмента за позициониране (*Positioning tool*).
- Всеки един обект може да бъде избран след щракване върху него с левия бутон на мишката. При избиране означението се изобразява с трептяща, прекъсната рамка, наречена маркерна. Това индицира, че обектът е бил избран. След като бъде избран обект, той може да бъде изтрит, отрязан, копиран или преместен. Щракнете върху свободно място, за да откажете избора на обекта.
- Големината на обектите също може да бъде променяна. Обектите, които могат да променят големината си имат места за промяна, обикновено в четирите си ъгъла. Когато позициониращата стрелка премине над някое от полетата за промяна на големината, тя се превръща в средство за промяна на големината (*Resizing Tool*). След това, за да бъде променена големината на избрания обект, трябва да се натисне левият бутон на мишката и без да се отпуска да се изтегли в желаната посока.

Стрелката за позициониране може да се използва, за да бъде преместено означението на температурния индикатор в лявата част на блоковата диаграма.

- Това става по следния начин: щраква се върху означението с левия бутон на мишката и се издърпва, както е показано на фиг.3.4. Влаченето на означението също премества и етикета.



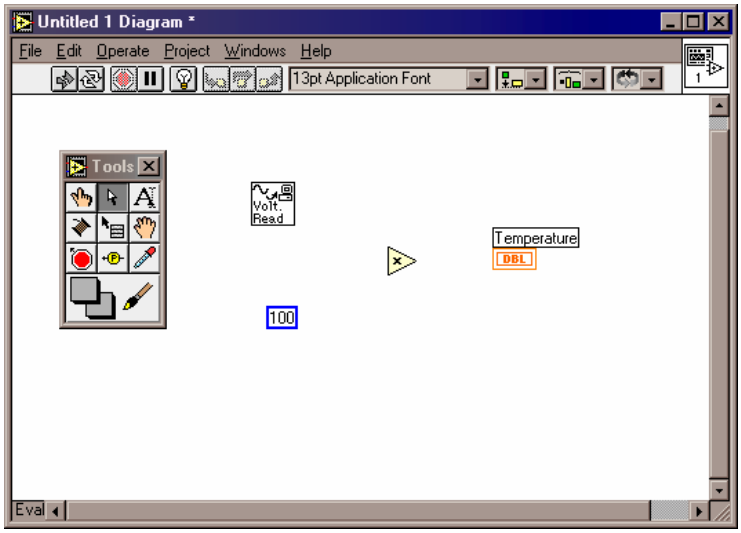

*Фиг. 3.4. Позициониране на обекти чрез изтегляне и пускане*

***Влаченето на етикета премества само етикета!***

### 2.1.4. Получаване на температурни данни

Необходимо е създаване на възел, който да програмира платката, така че тя да връща напрежение. След това тази стойност трябва да се мащабира в съответна температура.

- От меню *Windows* изберете командния ред *Show Functions* Palette. На екрана ще се появи панел *Functions*.
- От панел *Functions* изберете иконата на поле *Tutorial*, а след това щракнете върху *Demo Voltage Read VI*. Позиционирайте го в лявата половина на прозореца за диаграма.
- От поле *Arithmetic* изберете функцията за умножение (*Multiply function*) и я позиционирайте между *Demo Voltage Read VI* и означението за контрол на температурата.
- От табло *Structs&Constants* на меню *Functions* изберете цифрова константа.
- Позиционирайте я под възела *Demo Voltage Read*. Константата се появява и веднага може да бъде въведена желана стойност.
- Въведете 100 и натиснете клавиш <*Enter*>, за да дефинирате стойност на константата.
- Ако прозорецът на константата остане не избран или е необходима промяна стойността на константата, изберете инструмента A. След това щракнете върху прозореца на константата, за да изберете стойността и въведете нова стойност.
- С помощта на позициониращата стрелка, подредете обектите на блоковата диаграма, както е показано на фиг.3.5.




*Фиг. 3.5. Блокова диаграма на виртуалния инструмент (без връзки)*

### 2.1.5. Свързване на обектите

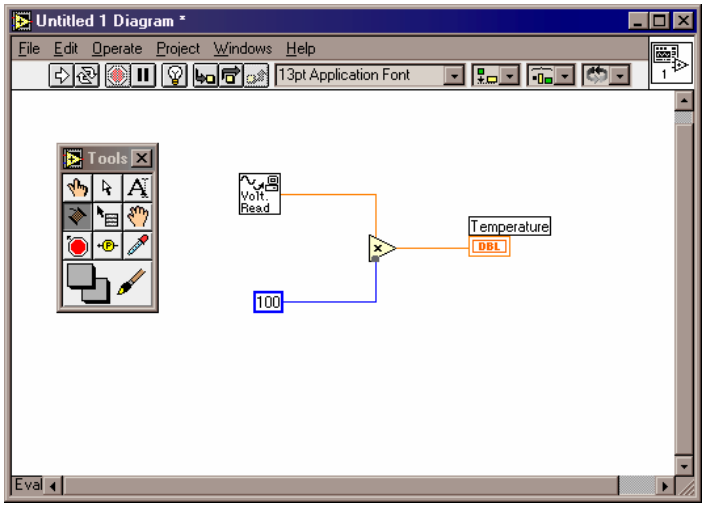
- Изберете инструмента за свързване (*Wiring tool*). Когато работите в прозореца на блоковата диаграма между

инструментите за свързване и позициониране може да се превключва чрез натискане на клавиша за интервал.

- Придвижете инструмента за свързване над възела *Demo Voltage Read.* Възелът започва да премигва, указвайки, че можете да се свържете към него. Щракнете и отпуснете левия бутон на мишката, след което придвижете мишката на дясно.
- Променете посоката на жицата надолу.
- За да направите втора промяна в посоката, натиснете левия бутон на мишката. Това застопорява жицата, така че може да промените посоката на движение. Придвижете инструмента за свързване над възела за умножение, както е показано на фиг.3.6. Горният ляв вход на възела за умножение започва да премигва, показвайки че може да се направи връзка.
- Натиснете левия бутон на мишката за да завършите операцията.

Плътната линия означава, че има добра връзка между възел *Demo Voltage Read* и възела за умножение. Прекъсната линия, означава неправилно свързване или лоша връзка (*bad wire*). Ако случайно направите лоша връзка, от меню *Edit* изберете *Remove Bad Wires* (премахване на лошите връзки). След това свържете отново възел *Demo Voltage Read* и възела за умножение.

- Завършете свързването на блоковата диаграма, както е показано на фиг.3.6.
- Ако бутон *Run* (Старт) е изобразен като счупен  , това означава, че има грешка в диаграмата и *VI* не може да бъде компилиран. Ако обектите са свързани, както е показано на фиг.3.6 и бутонът *Run* е изобразен като счупен, то най-вероятно възлите закриват малки лоши връзки. В този случай от меню *Edit* изберете *Remove Bad Wires*. Когато лошите връзки бъдат премахнати, бутонът *Run* изглежда отново цял  .




*Фиг. 3.6. Блокова диаграма на виртуалния инструмент (с връзки)*

### 2.1.6. Изпълнение на *VI*

- *VI* може да бъде стартиран както от прозореца на панела, така и от прозореца на диаграмата. За този пример от меню *Windows* изберете *Show Panel*.
- Щракнете върху бутон *Run*. По този начин преминавате в стартов режим, индициран от бутон Режим (*Mode button*) и стартирате *VI*, който извежда на екрана стойност на термометъра. След изпълнението на *VI*, *LabVIEW* се връща в режим редактиране (*Edit mode*).

Ако желаете да запазите собствени *VI* се постъпва по следния начин:

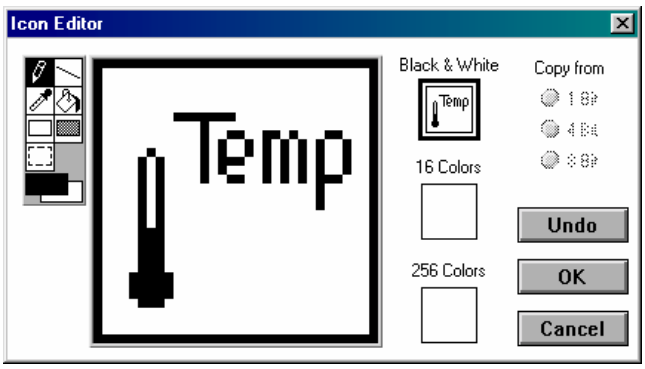
- От меню *File* се избира *Save*.
- В диалоговия прозорец, който се появява на екрана напишете например *Digital Thermometer.VI* и щракнете върху *OK*. Прозорецът с името приема името, което е зададено на *VI*.

### 2.1.7. Създаване на *Икона* и *Съединител*

За да може *Digital Thermometer VI* да бъде използван в други блокови диаграми на него трябва задължително да му бъде създадена ***Икона*** и ***Съединител***. *И*коната представлява графичното означение на *VI*, а съединителят присъединява контроли и индикатори към входовете и изходите на *VI*.

#### 2.1.7.1. Създаване на *Икона*

- Направете активен прозорец предния панел и преминете в режим редактиране.
- Активирайте падащото меню на празната икона, намираща се в горния десен ъгъл на прозореца на панела като щракнете с десния бутон на мишката. *И*зберете *Edit Icon* (Редактиране на икона).
- На екрана се появява прозореца на редактора на икони (фиг.3.7). *И*ползвайте инструментите, намиращи се в лявата част на иконата, за да създадете иконата в полето за работа. Нормалният размер на иконата се появява в една от кутиите в дясната част на полето за редактиране.



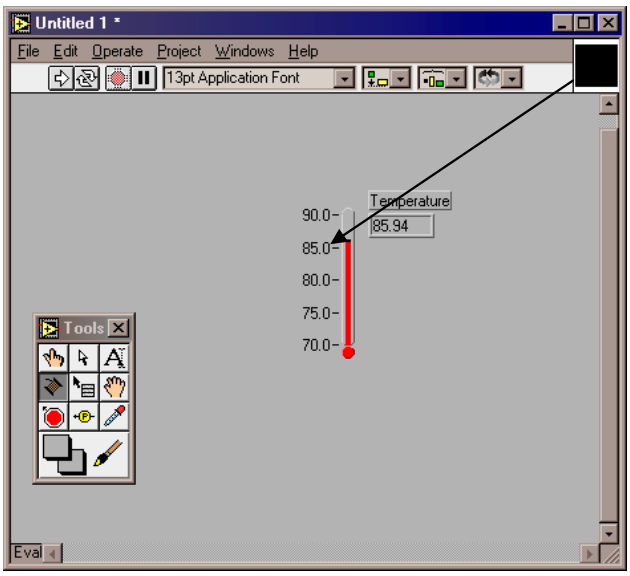

*Фиг. 3.7. Прозорец на редактора на икони*

- Използвайте инструментите, за да създадете черно-бяла икона, подобна на тази от фиг.3.7.
- Когато завършите създаването на иконата, щракнете върху бутон *OK* на редактора. Новата икона ще замени иконата изобразяваща празно поле.

### 2.1.7.2. Създаване на съединител

Изпращането и приемането на данни в *LabVIEW* става от вътрешни *VI* чрез помощта на входове/изходи (В/И) от съединителния панел. Връзките се дефинират по следния начин: избира се съединителен панел с необходимите В/И; на всеки от тези В/И се присъединяват контроли и индикатори от предния панел. Само тези от контролите и индикаторите, които ще бъдат програмирани трябва да имат В/И на съединителния панел.

- Щракнете върху иконата с десния бутон на мишката и изберете *Show Connector* (Покажи съединител). *LabVIEW* извежда съединителя по подразбиране, който съответства на броя на входовете и изходите на *VI*. Ето защо съединителят, който се появява в този пример има един изход.
- Извикайте падащото меню на празния съединител като щракнете с десния бутон на мишката и изберете *Patterns* (Шаблони), за да видите другите възможни съединители. Може да бъде избран всеки от изобразените съединителни шаблони като се щракне върху него. За този пример шаблонът по подразбиране е правилният избор.
- Изберете инструмента за свързване и щракнете върху В/И на съединителя. В/И придобива чер цвят.
- След това щракнете върху температурния индикатор. С това инициализирането на В/И на съединителя завършва (фиг.3.8).




*Фиг. 3.8. Създаване на съединител в LabVIEW*

*<u>Забележка:</u> Въпреки, че се извършва свързване между съединителя и обекти от предния панел, никакви жици не се чертаят.*

- Извикайте падащото меню на съединителя като щракнете с десния бутон на мишката върху него и изберете *Show Icon* (Покажи икона).
- Затворете *VI* като използвате командата *Close* (Затвори) от меню *File* (Файл). Когато диалоговия прозорец запита, дали да запази промените, щракнете върху бутон *Yes* (Да).

## 2.2. ИЗВИКВАНЕ НА *VI* КАТО ПОДПРОГРАМА

*LabVIEW* има модулна, йерархична архитектура, в която *VI* могат да извикват други *VI*. Всяко приложение може да бъде разделено на функционални части, за които да бъдат създадени *VI*. Чрез извикване на тези *VI* от по-високи нива може да се създадат сложни системи за измерване. По този начин се създават модули за многократно използване, които се добавят към възможностите на *LabVIEW*. В този модул, ще бъде използван виртуалния инструмент, който бе създаден в т.2.1. Този *VI* ще бъде използван във *VI*, който измерва температурата, като чете данните от вътрешния *VI*.

### 2.2.1. Симулиране и експериментално първоначално установяване

Да предположим, че е необходимо превключване на температурната скала между $^{o}$C и $^{o}$F. Трябва да се създаде *VI*, който да дава възможност на потребителя да избира с коя температурна скала да работи. Формулата, по която става преобразуването на $^{o}$C в $^{o}$F е следната:

$$T_F = \frac{9}{5} T_C + 32,$$

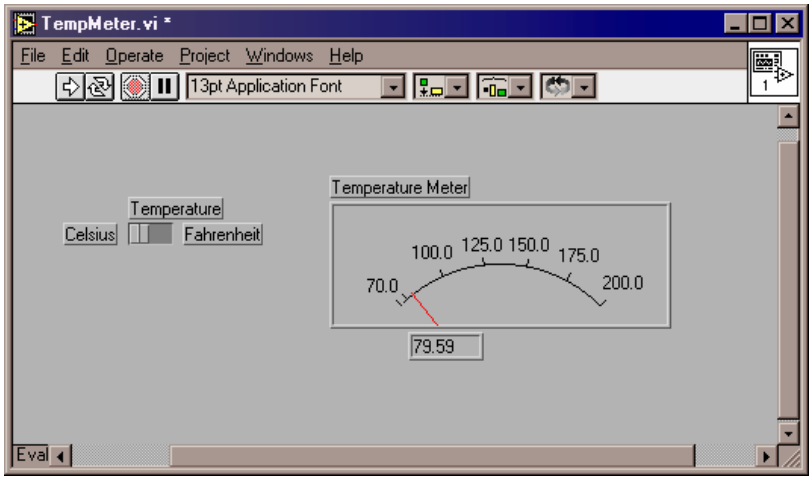
където $T_F$ е температурата в $^{o}$F, а $T_C$ е температурата в $^{o}$C.

- Затворете всички прозорци на *LabVIEW* и отворете нов *VI* като щракнете върху бутона *New VI* в диалоговия прозорец, който се появява.

### 2.2.2. Създаване на панел

Новият виртуален инструмент трябва да има хоризонтален превключвател, разположен на предния панел, с който да се извършва превключването между двете скали и индикатор, който да показва температурата.

- Изберете хоризонтален превключвател (*Horozontal Swich*) от поле *Boolean* на панел *Controls*. Поставете етикет *Temperature* на превключвателя.
- Поставете надписи *Celsius* и *Fahrenheit* от двете страни на превключвателя.




*Фиг. 3.9. Преден панел на VI за индикация на температура в различни скали*

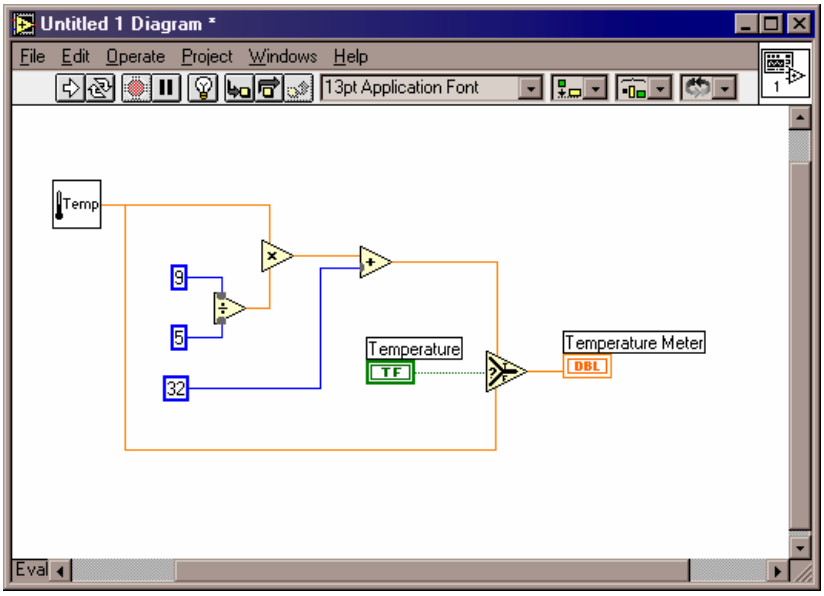
- От поле *Numeric* на панел *Controls* изберете индикатор и го позиционирайте върху предния панел. Задайте му етикет *Temperature Meter* (фиг.3.9).
- Променете долния диапазон на индикатора от 0.0 на 70.0. Променете и горния диапазон от 10.0 на 200.0.

### 2.2.3. Създаване на диаграма

- От меню *Windows* изберете *Show Diagram* (Покажи диаграма).
- От меню *File* изберете командата *Open*. В диалоговия прозорец намерете името на файла, който запазихте в предишното упражнение и го отворете. В блоковата диаграма ще се появи иконата на виртуалния инструмент за измерване на температура.
- От меню *Edit* изберете командния ред *Functions* и на екрана ще се появи неговия панел.
- Добавете останалите елементи на блоковата диаграма

(фиг.3.10) като използвате панел *Functions*:

- От поле *Numeric* изберете *Numeric Constant* и я добавете към диаграмата. Задайте стойност 9 като използвате инструмента за писане. По същия начин добавете още две цифрови константи и им задайте съответно стойности 5 и 32.
- От същото поле *Numeric* изберете и функциите за умножение, деление и събиране и ги добавете към блоковата диаграма.
- От поле *Comparison* на панел *Functions* изберете функция *Select*.
- Свържете елементите както е показано на фиг.3.10.




*Фиг. 3.10. Блокова диаграма на VI за индикация на температура в различни скали*

### 2.2.4. Изпълнение на *VI*

- Преминете към предния панел, като изберете *Show Panel* от меню *Windows* или като използвате бързата комбинация <*Ctrl+E*> от клавиатурата. Бързи комбинации от клавиши са включени в много от падащите менюта.
- Щракнете върху превключвателя *Temperature*, за да му зададете стойност *Celsius*. Индикаторът ще покаже температурата в $^{o}$C.
- Щракнете върху бутон *Run* от таблицата за изпълнение.
- Щракнете върху превключвателя *Temperature*, за да му зададете стойност *Fahrenheit*. Индикаторът ще покаже температурата в $^{o}$F.

Ако искате да запазите *VI*, използвайте командата *Save* от меню *File*.

## 2.3. ОТКРИВАНЕ НА ГРЕШКИ

Виртуалните инструменти не могат да се компилират или стартират, ако има грешки. Обикновено всеки виртуален инструмент показва грешки по време на създаване и редактиране, докато не бъдат свързани всички елементи в блоковата диаграма. Ако след създаването на виртуалния инструмент все още има грешки може да се използва командния ред *Remove Bad Wires* (Премахване на лоши връзки) от меню *Edit*.

Когато даден виртуален инструмент не може да бъде изпълнен, вместо бутон  *Run* се появява "счупен" бутон  *Run*. За да бъдат изведени грешките трябва да се щракне върху бутона  . Щракнете върху някоя от изброените грешки и след това върху *Find*, за да откриете обекта, предизвикващ грешката.

Изпълнението на блоковата диаграма може да бъде анимирано чрез използване на бутон  *Highlight Execution*. Този начин най-често се прилага при постъпково изпълнение, с цел проследяване потока на данни в блоковата диаграма.

С цел откриване на грешки потребителят може да използва постъпково изпълнение на диаграмата. За разрешаване на този режим трябва да се щракне върху бутон  *Step Into* или  *Step Over*. След изпълнението на тази операция първият елемент от блоковата диаграма започва да мига, което показва, че е готов да изпълни операцията. След това може да се щракне отново върху  *Step Into* или  *Step Over*, за да се изпълни дадената операция и да се премине към следващия елемент. Ако следващият елемент е структура или виртуален инструмент може да се щракне върху бутон  *Step Over*. По този начин елементът ще изпълни операцията, без да се проследява постъпково неговото изпълнение. Ако се щракне върху бутон  *Step Into*, това ще предизваика постъпково изпълнение на елемента, представляващ структура или виртуален инструмент.

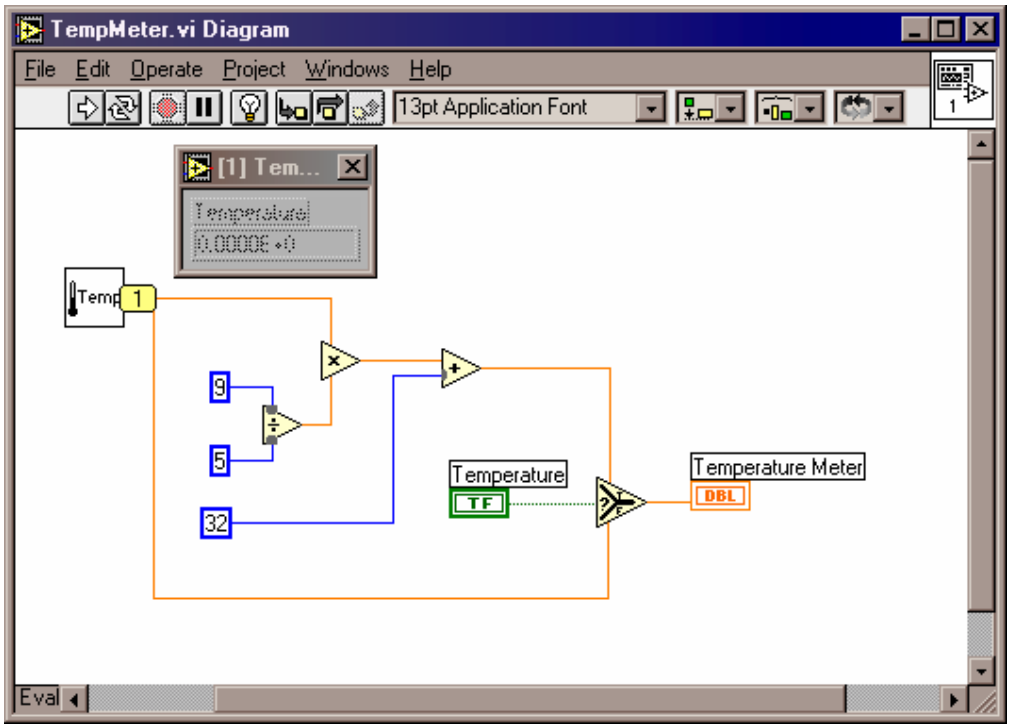
За прекратяване изпълнението на блоковата диаграма или за завършване на постъпковото изпълнение трябва да се щракне върху бутон  *Step Out*.

### 2.3.1. Проследяване изпълнението на програма в *LabVIEW*

Целта на упражнението е да се използва пробник и да се проследи потока на данните в блокова диаграма.

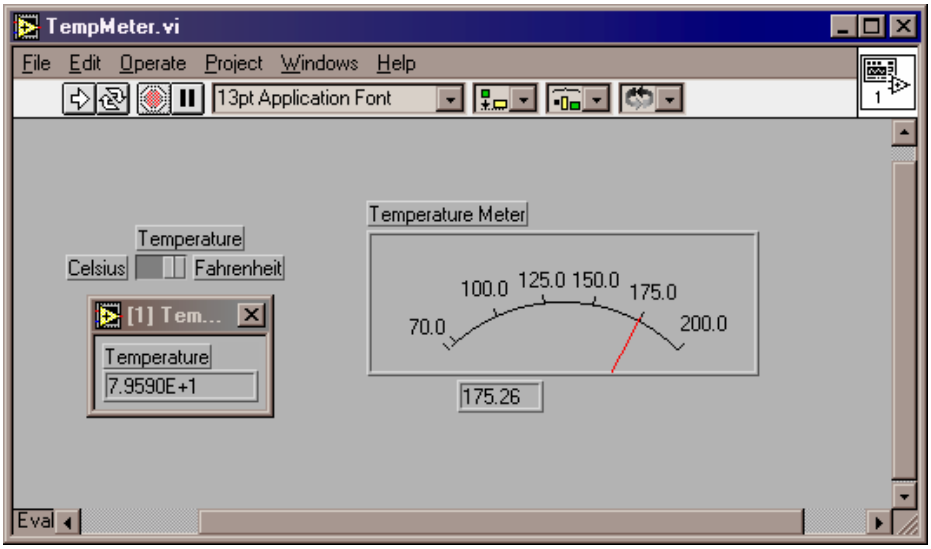
- Отворете виртуалния инструмент, създаден в предходното упражнение.
- Преминете към блоковата диаграма като от меню *Windows* изберете *Show Diagram*.
- Ако панелът с инструменти не е отворен изберете командния ред *Show Tools Palette* от меню Windows.

- От панел *Tools* изберете инструмента  *Probe*. Щракнете с него върху жицата, излизаща от елемента "Temp".
- На екрана ще се появи прозорец с име *[1] Temp...*, а върху диаграмата ще се появи жълта елипса с номера на пробата (фиг.3.11). Прозореца с пробата остава отворен дори, ако се премине към предния панел.




*Фиг. 3.11. Блокова диаграма с пробник и прозорец [1] Temp...*

- Върнете се към предния панел и преместете прозореца *[1] Temp...* по такъв начин, че да се виждат пробата и стойностите (фиг.3.12).
- Стартирайте виртуалния инструмент.
- Стойността на температурата в $^{o}$C ще се появи в прозореца на пробата, докато индикаторът ще отчете температурата в $^{o}$F.

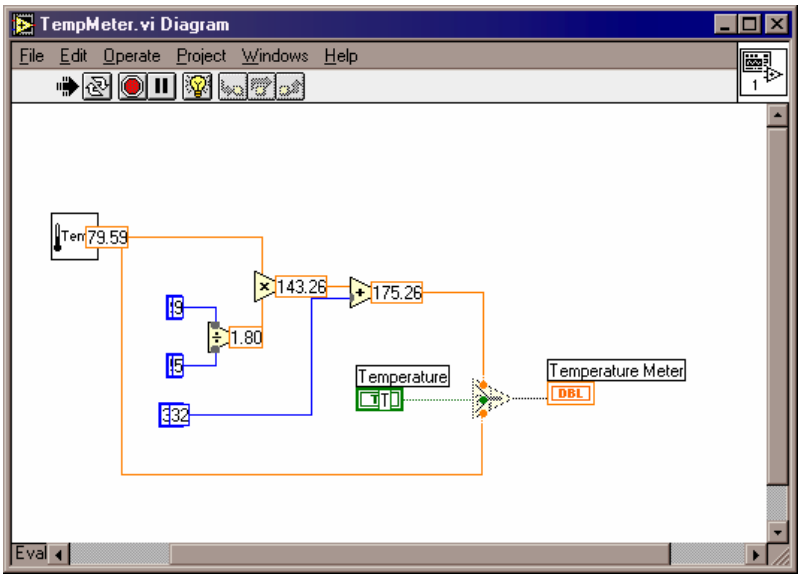



*Фиг. 3.12. Преден панел с прозорец за проби [1] Temp...*

- Затворете прозореца на пробата като щракнете върху символа за затваряне в горния десен ъгъл.

Друга полезна технология при тестване изпълнението на програма е проследяване потока на данните.

- Преминете към блоковата диаграма.
- Стартирайте проследяването на изпълнението като щракнете върху бутон  *Highlight Execution*. Иконата на бутона ще се промени и ще добие вида  .
- Щракнете върху бутон *Run*, за да стартирате програмата. Ще се стартира анимация, като движещите се точки показват потока на данните. В момента на изпълнение на програмата се извеждат и стойностите на данните по жиците (фиг.3.13).




*Фиг. 3.13. Блокова диаграма с проследяване на изпълнението*

- Спрете проследяването на изпълнението като щракнете върху бутон  . Иконата на бутона ще се промени и ще добие вида  .

В *LabVIEW* може да бъде извършено и проследяване изпълнението на програмата стъпка по стъпка.

- За целта трябва да се щракне върху бутон  *Step Over*.
- Използването на бутоните  *Step Over* и  *Step Into* води до постъпково изпълнение, като първия бутон прескача структурите и виртуалните инструменти, а вторият позволява постъпково изпълнение в сложни структури.
- За спиране на постъпковото изпълнение трябва да се щракне върху бутон  *Step Out*.

## 2.4. ИЗПОЛЗВАНЕ НА ЦИКЪЛ *WHILE* И ГРАФИКА

### 2.4.1. Симулиране и експериментално първоначално установяване

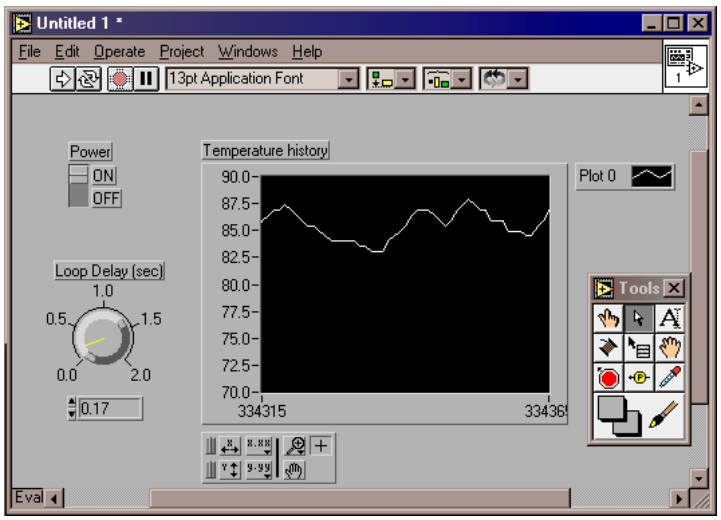
Да предположим, че е необходимо непрекъснато измерване на температурата в камерата, описана в т.2.1. Трябва да се създаде *VI*, който извършва измерване на температурата през интервал, задаван от потребителя и извежда резултатите в графичен вид.

- Затворете всички прозорци на *LabVIEW* и отворете нов *VI* като щракнете върху бутона *New VI* в диалоговия прозорец, който се появява.

### 2.4.2. Създаване на панел

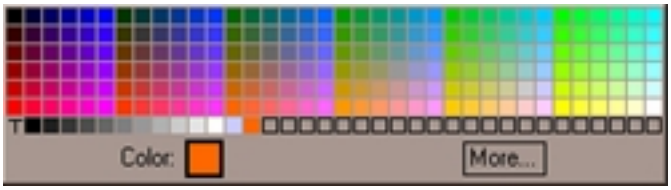
Новият виртуален инструмент трябва да има бутон за включване/изключване на захранването, разположен на предния панел. С този бутон ще се стартира и спира събирането на данни. Също така трябва да притежава графичен дисплей, който да извежда температурата.

- Изберете вертикален превключвател от *Boolean* поле на панел *Controls*. Поставете етикет *Power* на превключвателя.
- От поле *Numeric* на панел *Controls* изберете *Knob*. Неговото предназначение ще бъде да задава измервателния интервал. Задайте му етикет *Loop delay (sec)*.
- Като използвате инструмента за задаване на надписи  променете горния обхват от 10.0 на 2.0.
- За извеждане на графичната информация изберете *Waveform Chart* от поле *Graph* на панел *Controls*. Поставете етикет *Temperature History*. Позиционирайте управляващите елементи, както е показано на фиг.3.14.




*Фиг. 3.14. Преден панел на VI за измерване на температура*

- Променете обхвата, за да получите температурни стойности. Чрез инструмента за позициониране се щраква двукратно върху долната граница и се написва 70. По същия начин се променя на 90 и горната граница. Междинните стойности автоматично се преизчисляват, когато се натисне <*Enter*> от клавиатурата. Същото може да се извърши и като се щракне върху панела или върху бутон *Enter* от лентата с инструменти.
- Ако искате да промените цвета на модула за графична информация, изберете инструмента за оцветяване и щракнете върху модула за графична информация (фиг.3.15) с левия бутон на мишката. Щракнете върху цвят, който желаете. Щраквайте последователно с инструмента за оцветяване върху елементите, на които искате да промените цвета.




*Фиг. 3.15. Панел за избор на цветове*

### 2.4.3. Създаване на диаграма

- От меню *Windows* изберете *Show Diagram* (Покажи диаграма). Изберете инструмента за позициониране и преместете елементите, ако е необходимо.
- От меню *File* изберете командата *Open*. В диалоговия прозорец намерете името на файла, който запазихте в първото упражнение и го отворете. В блоковата диаграма ще се появи иконата на виртуалния инструмент за измерване на температура.

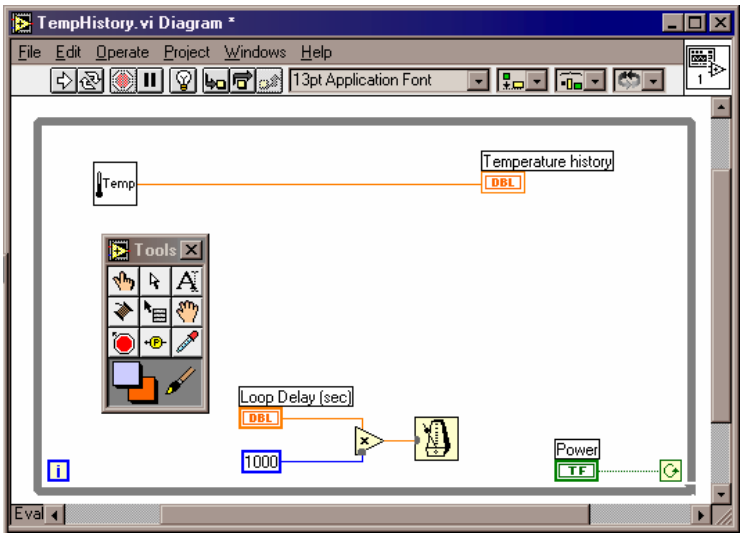
### 2.4.4. Използване на *While Loop* (Цикъл *while*)

Ако *VI* бъде стартиран, той ще измери температурата еднократно и ще я визуализира в графичен вид. За да се получат многократни измервания трябва да се използва цикъл *While*.

Цикълът *While* представлява условна структура за изпълнение. Поддиаграмите, намиращи се в неговата рамка се изпълняват докато условието е истина.

- Изберете *While Loop* от поле *Structures* на панел *Functions*. Когато *While Loop* се появи, той може да закрие някой от обектите от диаграмата. Ако това се получи преместете *While Loop* като използвате инструмента за позициониране.
- Уголемете *While Loop*, така че да може да включи в себе си всички други елементи. За целта позиционирайте курсора върху някой от ъглите на цикъла докато се появи маркер за промяна на големината. Щракнете и изтеглете, за да уголемите цикъла.
- Изберете всички елементи като щракнете с маркера за позициониране отгоре вляво на *VI* и след това издърпате надолу вдясно. Правоъгълникът, избира всички обекти, които се намират вътре в него, когато се отпусне бутона на мишката. Преместете избраните обекти в *While Loop*.
- Свържете превключвателя *Power* с входа за условие. С помощта на права, изобразена с точки се представят булеви данни.
- Свържете иконата на термометъра с полето за графики.

### 2.4.5. Контролиране скоростта на изпълнение на *While Loop*




*Фиг. 3.16. Блокова диаграма на VI за измерване на температура*

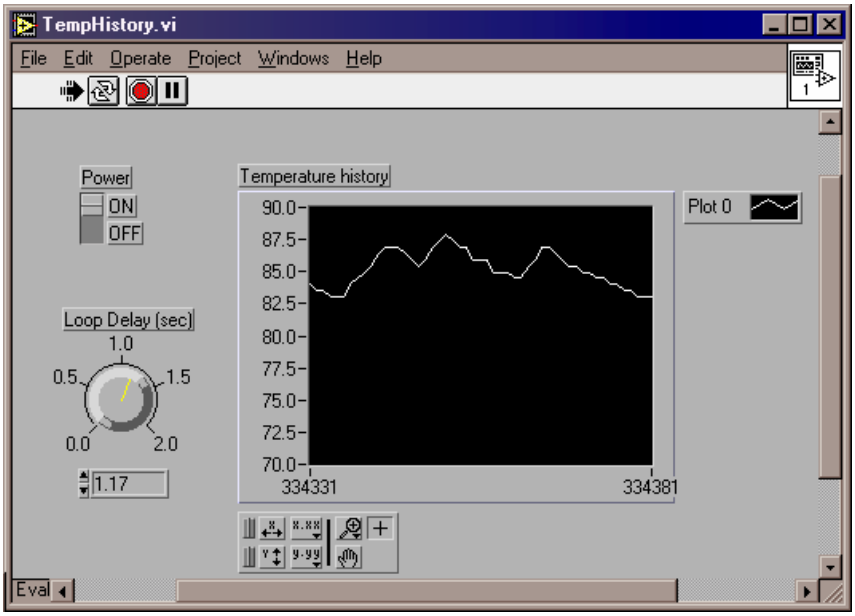
- От поле *Time&Dialog* на панел *Functions* изберете функция *Wait Until Next ms*. Тази функция осигурява изпълнение на цикъла всяка *ms*.

- От поле *Numeric* на панел *Functions* изберете цифрова константа и променете нейната стойност на 1000. Функцията *Wait Until Next ms* с константа 1000 осигурява изпълнението на този цикъл всяка секунда.
- От същото поле *Numeric* изберете и функцията за умножение и я добавете към блоковата диаграма.
- Свържете елементите на блоковата диаграма, както е показано на фиг.3.16.

### 2.4.6. Изпълнение на *VI*

- Преминете към предния панел, като изберете *Show Panel* от меню *Windows* или като използвате бързата комбинация <*Ctrl+F*> от клавиатурата. Бързи комбинации от клавиши са включени в много от падащите менюта.
- Щракнете върху превключвателя *Power*, за да му зададете стойност *ON*.
- Задайте стойност на интервала за измерване като използвате *Loop Delay (sec)*.
- Щракнете върху бутон *Run* от таблицата за изпълнение.

Полето за графика ще изчертае температурните стойности (фиг.3.17). *VI* получава и извежда на екрана нова стойност за всяка итерация на *While Loop*. Старите стойности на графиката се изместват в ляво, а новите стойности се изчертават отдясно. Ако по време на изпълнение промените стойността на интервала за измерване ще забележите промяна и в скоростта на извеждане на информацията в полето за графика.




*Фиг. 3.17. Работещ преден панел на VI за измерване на температура*

- За да бъде спряно изпълнението е необходимо да изключите *Power* (долно положение). Това предизвиква излизане от цикъла

*While* и спиране на програмата.

- Ако искате да запазите *VI*, използвайте командата *Save* от меню *File*.

## 2.5. ИЗПОЛЗВАНЕ НА ИЗМЕСТВАЩИ РЕГИСТРИ

В този пример създадения виртуален инструмент ще бъде променен, така че да извършва елементарен анализ на получените данни като за целта се използват изместващи регистри.

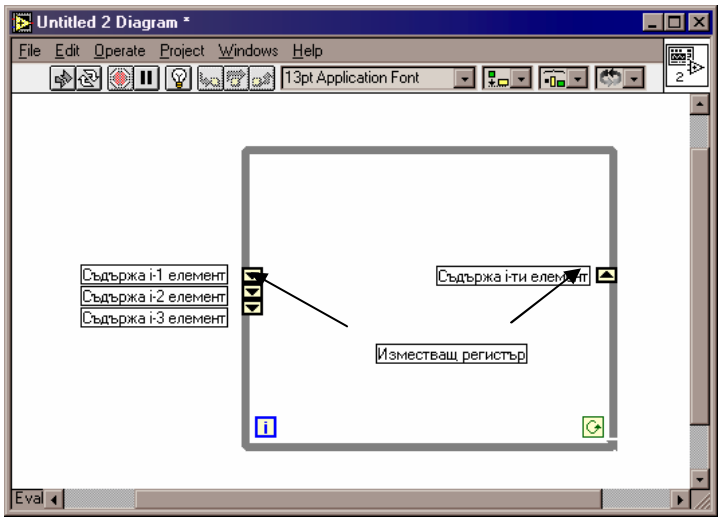
### 2.5.1. Симулиране и експериментално първоначално установяване

Да предположим, че температурната камера в симулирания експеримент внася шум и системата за измерване на температурата трябва да го отчита. За да се направи най-прост анализ, трябва да се изчисли средната стойност на последните три измервания, като по този начин се намали ефекта от шума. За целта:

- Отворете *VI*, който създадохте в предишното упражнение.
- Изберете *Save As*... и запазете *VI* под ново име.

### 2.5.2. Изместващи регистри

Изместващите регистри са част от функциите за цикли в езика G. Те служат за предаване на стойности от една итерация на цикъла към следващата. Изместващ регистър може да се добави към цикъл чрез щракване с десния клавиш на мишката върху лявата или дясната рамка и избиране на командния ред *Add Shift Register* от менюто, което се появява.




*Фиг. 3.18. Цикъл While с изместващ регистър*

Изместващият регистър се състои от двойка терминали, разположени един срещу друг върху вертикалните страни на рамката на цикъла. Десният терминал съхранява стойност до изпълнението на

итерацията. След края на итерацията тази стойност се измества и се появява в левия терминал в началото на следващата итерация. Изместващия регистър автоматично се адаптира към типа на данните и може да съхранява числови, логически, стрингове, масиви и др. данни.

Изместващият регистър може да бъде конфигуриран по такъв начин, че да помни няколко предишни стойности (фиг.3.18). Тази функция е полезна, когато трябва да се извършва усредняване. За целта трябва да се добавят допълнителни терминали като се щракне с десния клавиш на мишката върху лявата част на регистъра и от появилото се меню се избере *Add Element* (добави елемент).

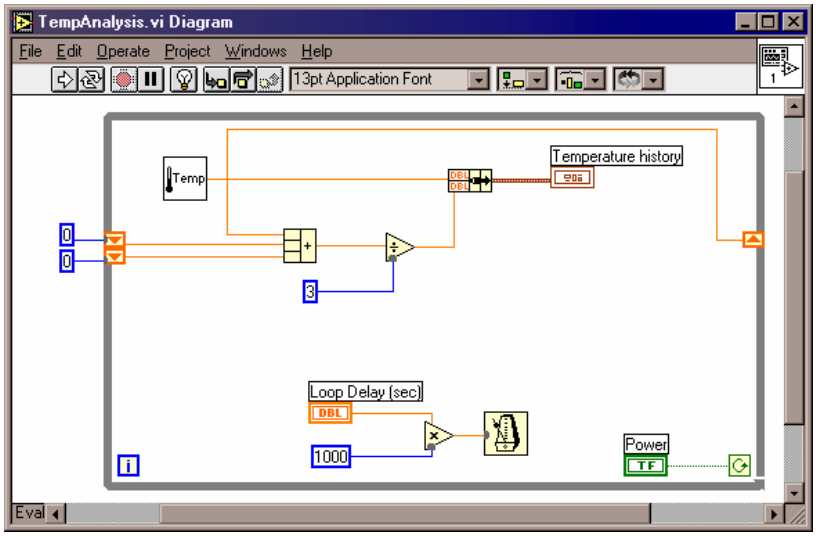
Изместващите регистри могат да бъда инициализирани с начални стойности като терминалите от лявата част на цикъла се свържат с константи извън цикъла. Ако не се зададат начални стойности, при първото стартиране на VI изместващия регистър се инициализира със случайни стойности. При следващо стартиране в регистъра се съхраняват последните стойности от предходното стартиране. На фиг.3.18 изместващият регистър не е инициализиран с начални стойности, докато на фиг.3.19 е инициализиран със стойност 0-ла.

### 2.5.3. Промяна на блоковата диаграма

- Щракнете с десния клавиш на мишката върху лявата или дясната линия на цикъла *While*. От появилото се меню изберете *Add Shift Register* (добавяне на изместващ регистър).
- Добавете нов елемент към изместващия регистър като щракнете с десния клавиш на мишката върху лявата част на регистъра и от появилото се меню изберете *Add Element* (добави елемент).
- От поле *Numeric* на панел *Functions* изберете *Compound Arithmetic* и го добавете към блоковата диаграма. За да добавите трети елемент към тази функция щракнете с десния клавиш на мишката върху нея и от менюто изберете *Add Input* (добавяне на вход).
- От поле *Numeric* на панел *Functions* изберете функцията за деление (Divide) и я добавете към блоковата диаграма.
- От същото поле на *Functions* изберете цифрова константа (*Numeric Constant*) и я добавете към блоковата диаграма. Задайте й стойност 3.0 (за усредняване по три точки). Добавете още две цифрови константи, които ще бъдат използвани за инициализация на изместващия регистър.
- От поле *Clusters* на панел *Functions* изберете функция *Bundle*  (Връзка). Тази функция дава възможност за съвместяване на няколко диаграми в един графичен прозорец.
- Съединете всички елементи, както е показано на фиг.3.19. *VI* ще извежда в графичен вид едновременно текущата и усредената

по три стойности температура.

- Когато завърши свързването на новата *Bundle* функция, се изчертава друг тип жица от дясната страна на функцията. Този тип представлява тип данни, наречен клъстер (*cluster*). Клъстерите са типове данни за групиране на информацията, подобни на записите в езика *Pascal* или структурите в *C*.




*Фиг. 3.19. Блокова диаграма на VI за анализ на температурата*

*<u>Забележка:</u> Когато премествате обекти, свързаните с тях жици също се преместват, за да се запазят създадените връзки. Чрез щракване се избират обекти, а с помощта на комбинацията <Shift>+щракване се добавят или отстраняват обекти от групата на избраните.*

### 2.5.4. Промяна на предния панел

Единствената промяна, която се налага да бъде направена на предния панел е да се промени модула за изчертаване на графика, така че да се извеждат двете функции.

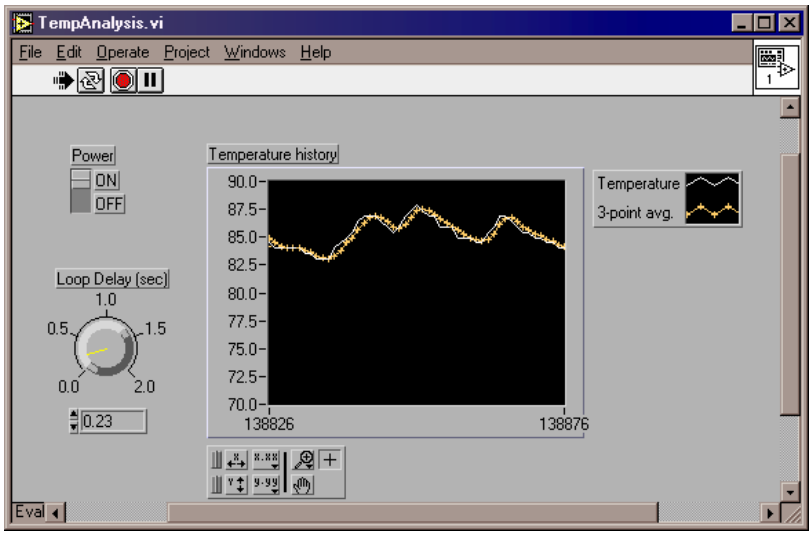
- От меню *Windows* изберете *Show Panel*, за да бъде изведен предния панел.
- Извикайте падащото меню на модула за изчертаване на графики като щракнете с десния бутон на мишката върху него и изберете от подменю *Accessories* командата *Show Legend* (Покажи легенда). Ако е необходимо променете големината на предния панел.

В следващите няколко стъпки трябва да се използват няколко инструмента. За да се улесни избора на инструмент, може да се натиска клавиш <*Tab*> и да се преминава от инструмент към инструмент.

- Преместете маркера за позициониране над долния десен ъгъл на легендата, докато се появи маркера за промяна на големината. Щракнете и изтеглете надолу, докато се появи

легендата на втората графика.

- Двукратно щракнете с маркера за поставяне на имена върху всяко от двете имена на графики (0. и 1.). Преименувайте ги, както е показано на фиг.3.20. Променете големината на легендата от долния ляв ъгъл, така че имената да се побират. (Ако напишете *Temperature* преди да уголемите легендата, текстът може да изчезне, тъй като не се побира в отреденото му място. Той се появява, когато освободите място за него като увеличите големината на легендата с помощта на долния ляв ъгъл).
- Атрибутите на двете графики трябва да бъдат различни. За целта извикайте падащото меню на графичния прозорец на легендата като щракнете с десния бутон на мишката. Променете начина на изчертаване на точките и типа на линиите за двете графики. Може да използвате командата за промяна на цветовете от същото падащо меню, за да промените цвета на графиките.




*Фиг. 3.20. Преден панел на работещ VI за измерване и анализ на температура*

### 2.5.5. Изпълнение на *VI*

- Изберете инструмента.
- Включете "захранването" и щракнете върху бутон *Run*, за да стартирате виртуалния инструмент.
- Изключете "захранването", за да спрете изпълнението на *VI*.
- Запазете *VI* като използвате командата *Save* от меню *File*.

## 2.6. ОПЕРАЦИИ ЗА СРАВНЕНИЕ И КОНТРОЛ

В този модул, виртуалния инструмент за анализ на температурата трябва да бъде променен, така че дава възможност за контрол.

### 2.6.1. Симулиране и експериментално първоначално

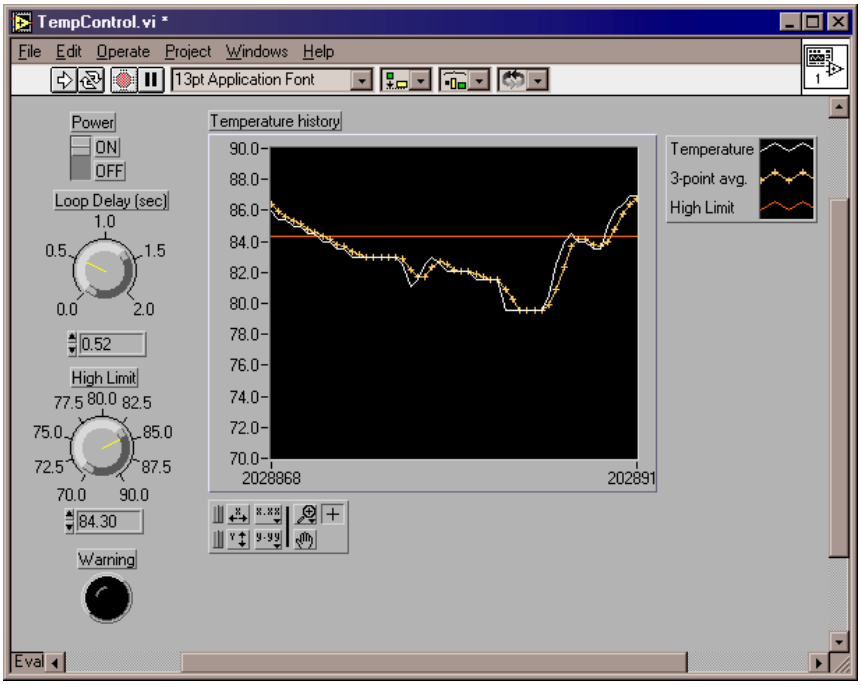
**установяване**

Нека предположим, че температурата от експеримента трябва да бъде поддържана в определени граници. Необходимо е да се определят границите, да се сравняват с текущата температура и да предупреди оператора, ако температурата излезе извън тези граници. В този пример ще бъде добавен цифров контрол към предния панел, с помощта на който да се задава горна граница на температурата. Също трябва да бъде променен и модулът за графика, така че да извежда както данните, така и горната граница на температурата. По този начин операторът ще бъде информиран визуално, когато температурата надхвърли горната граница. Като допълнение ще бъде създадена и сигнална индикация.

- Отворете *VI*, който създадохте в предишното упражнение.
- *Изберете Save As*... и запазете *VI* под ново име.

### 2.6.2. Промяна на панела

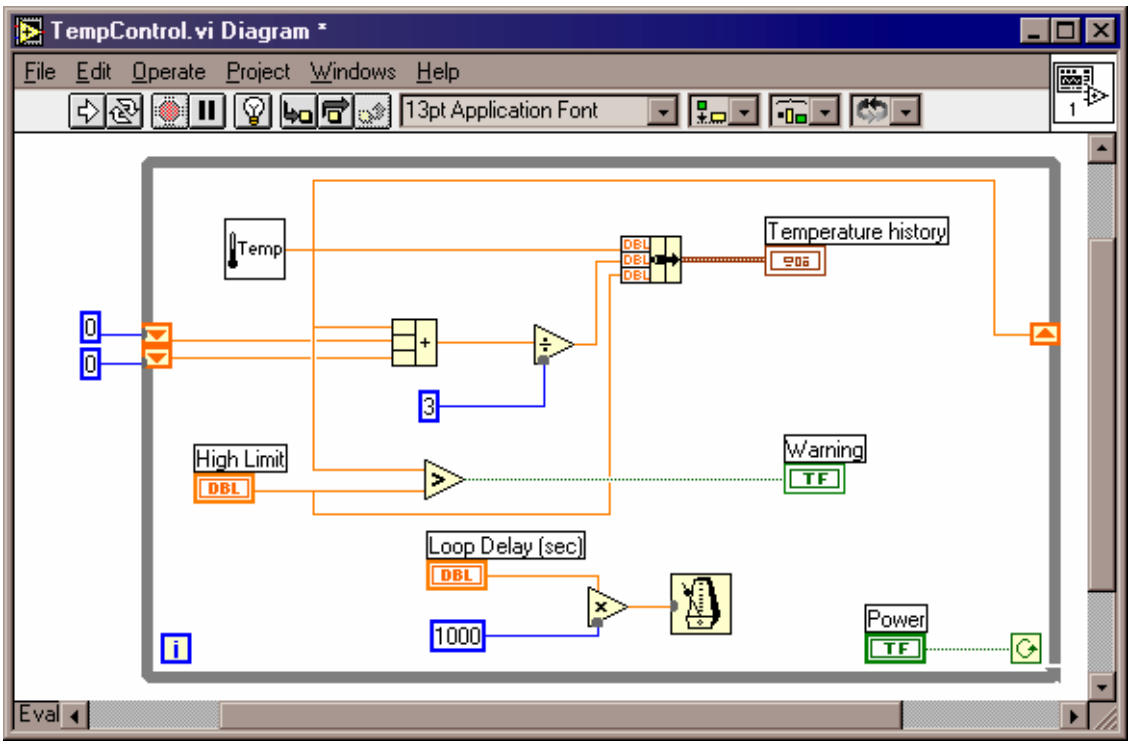
- От поле *Numeric* на панел *Controls* изберете *Knob* (Бутон). Поставете му име *High Limit* и променете скалата му, използвайки средството за писане като се ръководите от фиг.3.21.
- Придвижете бутона до позицията, посочена на фиг.3.21. Може да се наложи промяна на големината на панела и преместване на модула за графика, за да се освободи място.
- Изберете кръгъл *LED* индикатор от поле *Boolean* на панел *Controls*. Поставете му име *Warning* и го позиционирайте, както е показано на фиг.3.21.




*Фиг. 3.21. Промяна на предния панел*

### 2.6.3. Промяна на блокова диаграма

- Активирайте прозореца на блоковата диаграма.




*Фиг. 3.22. Промяна на блоковата диаграма*

- За да добавите нов вход във възела *Bundle*, променете неговата големина като издърпате долния десен ъгъл. По този начин може да добавите данни от контрола на горната граница. Може да се наложи уголемяване на *While Loop* и преместване на някои възли, така че да се поберат вътре в цикъла.
- Изберете функцията *Greater* (По-голям) от поле *Comparison* на меню *Functions*. Свържете блоковата диаграма, както е показано на фиг.3.22.

*<u>Забележка:</u> Тъй като сложността на диаграмата нараства, понякога може да изберете грешни възли от меню Functions или да свържете възли неправилно. За да откриете повече за тези грешки, щракнете върху "счупения" бутон Run.*

За да избегнете усложняване на грешките, те трябва да се отстраняват възможно най-бързо. За да изтриете нежелан обект, щракнете върху него с маркера за позициониране. След това използвайте един от трите начина за изтриване: натиснете клавиш <*Backspace*>; натиснете клавиш <*Delete*> или изберете *Clear* (Изчистване) от меню *Edit*. За да премахнете лоши връзки, използвайте *Remove Bad Wires* (Премахване на лоши връзки) от меню *Edit*.

### 2.6.4. Изпълнение на *VI*

- Активирайте предния панел.

- Уголемете легендата, за да визуализирате новата графика. Задайте й име *High Limit*.
- Задайте горната граница като щракнете двукратно на прозорчето на *High Limit* и въведете 80.00. Границата може да зададете и като щракнете върху указателя на бутона и издърпате, докато се появи 80.00.
- Изберете *Save* от меню *File*.
- Включете "захранването" и щракнете върху бутон *Run*. Можете да променяте стойността на горната граница докато *VI* се изпълнява, като използвате един от двата метода, описани в стъпка 3. Ако промените горната граница на 85 чрез завъртане на бутона, на екрана се появяват стойности между 80 и 85.
- Спрете изпълнението като изключите "захранването" (*off* позиция).
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

***Задача:*** *Добавете контрол за долната граница на температурата.*

## 2.7. НАМИРАНЕ НА МИНИМАЛНА, МАКСИМАЛНА И СРЕДНА СТОЙНОСТ

В този модул, виртуалния инструмент за анализ на температурата трябва да бъде променен, така че дава възможност извеждане на текущата минимална и максимална стойност и средната стойност след края на всички измервания.

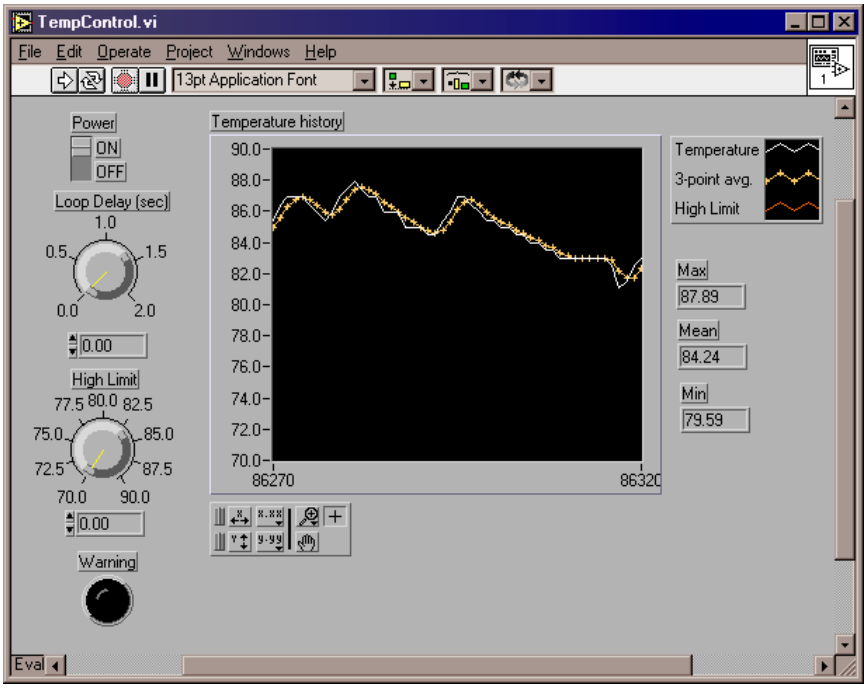
### 2.7.1. Симулиране и експериментално първоначално установяване

Нека предположим, че трябва да се получи информация за минималната, максималната и средната температурата от експеримента. В този пример ще бъдат добавени индикации към предния панел, които да извеждат тези стойности.

- Отворете *VI*, който създадохте в предишното упражнение.
- Изберете *Save As*... и запазете *VI* под ново име.

### 2.7.2. Промяна на панела

- От поле *Numeric* на панел *Controls* изберете *Digital Indicator* (Цифров индикатор). Поставете му име *Max.* Аналогично добавете още два цифрови индикатора и им задайте съответно имена *Min* и *Mean* (фиг.3.23).

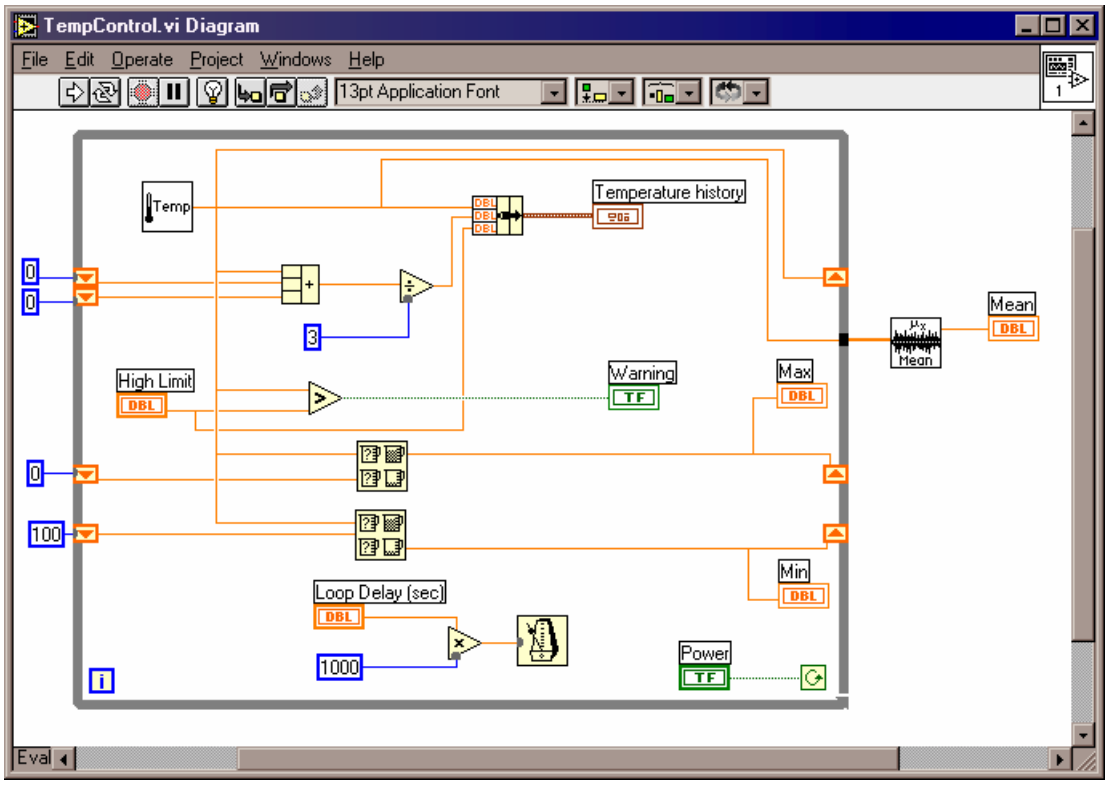



*Фиг. 3.23. Промяна на предния панел*

### 2.7.3. Промяна на блоковата диаграма

За да се извърши намиране на минималната и максималната стойност е необходима промяна на блоковата диаграма.

- Първо трябва да се добавят още два изместващи регистъра. За целта щракнете с десния клавиш на мишката върху лявата или дясната линия на цикъла *While*. От появилото се меню изберете *Add Shift Register* (добавяне на изместващ регистър).




*Фиг. 3.24. Промяна на блоковата диаграма*

- От поле *Numeric* на *Functions* изберете цифрова константа (*Numeric Constant*) и я добавете към блоковата диаграма. Добавете още една цифрова константа и й задайте стойност 100. Тези константи ще бъдат използвани за инициализация на двата нови изместващи регистъри.
- От поле *Comparison* на панел *Functions* изберете функция *Max&Min* и го добавете към блоковата диаграма. Добавете още една функция *Max&Min*. Първата функция служи за намиране на максималната, а втората за намиране на минималната температура.
- От поле *Analysis / Probability and Statistics* на панел *Functions* изберете функция *Mean* и я добавете към блоковата диаграма. Тази функция извършва пресмятане на средноаритметична стойност.
- Свържете новите елементи по начина, показан на фиг 2.24.

*Забележка: Ако при свързване на данните към елемент Mean се получи лоша връзка, щракнете с десния клавиш на мишката върху тунела в дясната рамка на цикъла While и изберете командния ред Enable Indexing (Разрешено индексиране).*

### 2.7.4. Изпълнение на *VI*

- Активирайте предния панел.
- Включете "захранването" и щракнете върху бутон *Run*. В полета *Max* и *Min* започват да се извеждат съответно текущата най-голяма и най-малка стойност на температурата.
- Спрете изпълнението като изключите "захранването" (*off* позиция).
- В поле *Mean* ще се изведе средноаритметичната стойност на всички измервания.
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

## 2.8. ИЗПОЛЗВАНЕ НА МАСИВИ. ЗАПАЗВАНЕ НА ДАННИ ВЪВ ФАЙЛ

В този модул, виртуалния инструмент за анализ на температурата трябва да бъде променен, така че да извежда резултатите върху предния панел в цифров вид като използва масив. Ще трябва да се добави и възможност за запазване на получените резултати във файл.

### 2.8.1. Симулиране и експериментално първоначално установяване

Нека предположим, че върху предния панел трябва да се изведат в

цифров вид всички получени стойности, а след края на опита данните да бъдат записани във файл.

- Отворете *VI*, който създадохте в предишното упражнение.
- Изберете *Save As*... и запазете *VI* под ново име.

### 2.8.2. Масиви

Масивът представлява съвкупност от елементи, които са от един и същ тип. Масивите в *LabView* могат да имат една или повече димeнсии и до $2^{31}$-1 елемента във всяка димeнсия. Достъпът до всеки елемент от даден масив се осъществява чрез индекс. Индексът може да приема стойности от 0 до n-1, където n е броят на елементите в масива. Първият елемент на масивите в *LabVIEW* има индекс 0! Елементите на масивите могат да бъдат числови, логически, стрингове или клъстери. Масиви и графики не могат да бъдат елементи на масив.

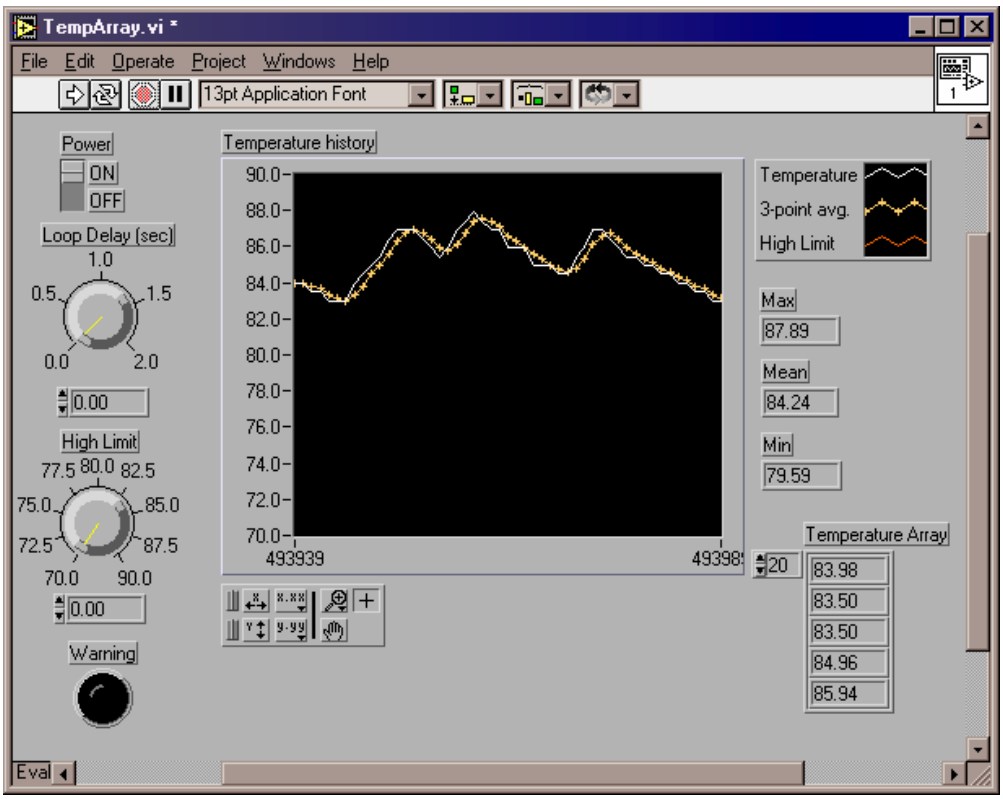
Циклите *For* и *While* могат да индексират и да натрупват елементи на масиви автоматично. Тези възможности се наричат автоматично индексиране (*auto-indexing*). Автоматичното индексиране е разрешено по подразбиране за цикъл *For*, докато за цикъл *While* по подразбиране е забранено. За да се разреши/забрани автоматичното индексиране трябва да се щракне с десния клавиш върху тунела (появява се при свързване на елемент от вътрешната страна на рамката с елемент от външната) на цикъла (в блоковата диаграма) и да се избере съответно *Enable Indexing / Disable Indexing*.

Когато е разрешено автоматичното индексиране, при свързване на масив към лявата част на рамката на цикъл, този масив става входен и елементите му започват да постъпват един по един във всяка итерация. Аналогично, ако масив се свърже към дясната част, той става изходен и в него елементите се записват един по един във всяка итерация.

*LabView* притежава функции за създаване на масиви, получаване броя на елементите в масив, заместване на елементи, търсене на елементи, сортиране и др. Тези функции са достъпни от поле *Array* на панел *Functions* от блоковата диаграма.

### 2.8.3. Промяна на панела

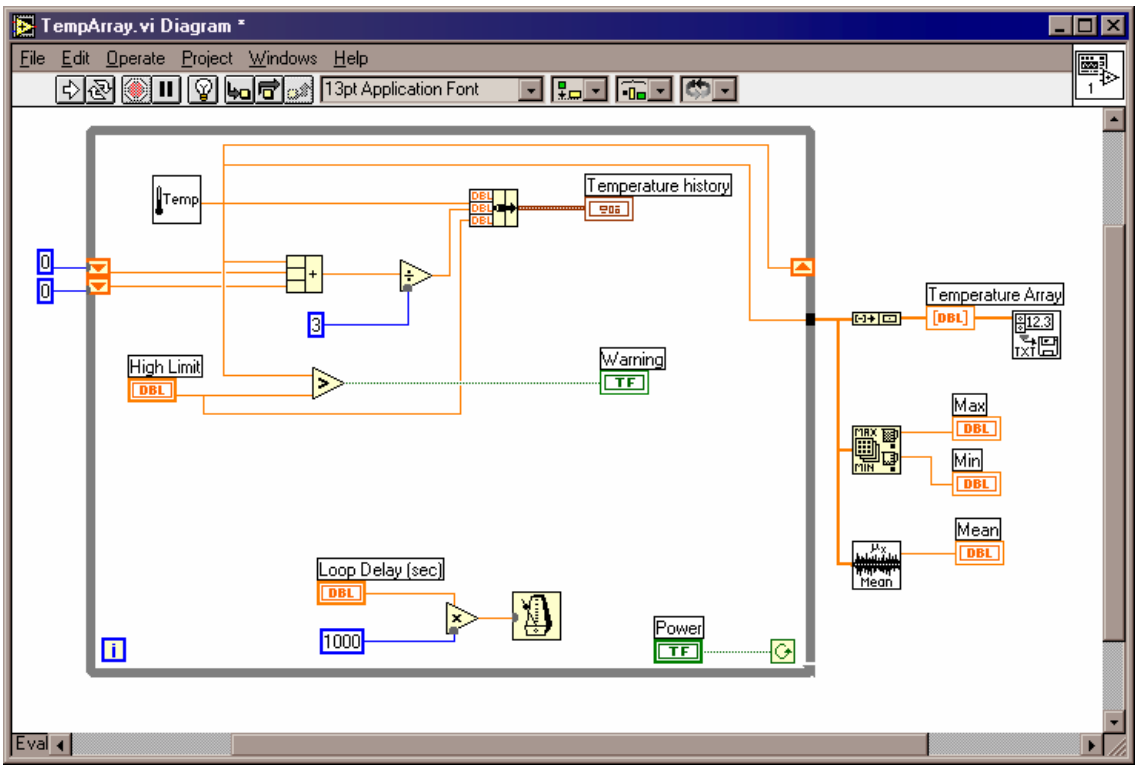
- Първо от поле *Array & Cluster* на панел *Controls* изберете *Array* (Масив). Поставете му име *Temperature Array.*
- След това трябва да укажете типа на масива. Тъй като температурата се получава във вид на числа, от поле *Numeric* на панел *Controls* трябва да изберете *Digital Indicator*. Чрез влачене и пускане го вмъкнете в полето на масива (фиг.3.25). По този начин се указва, че елементите на масива ще бъдат числови стойности.




*Фиг. 3.25. Промяна на предния панел*

### 2.8.4. Промяна на блоковата диаграма

- Първо ще приложим друг метод за определяне на минималната и максималната стойност. За целта от поле *Array* на панел *Functions* изберете функция *Array Max&Min*. Тази функция намира минималната и максималната стойност в масив. Позиционирайте елемента извън цикъла *While.* Премахнете вече ненужните връзки и изместващи регистри (фиг.3.26). Ако е необходимо използвайте командата *Remove Bad Wires* от меню *Edit*.
- Добавете функция, която да създаде масив от направените измервания. От поле *Array* на панел *Functions* изберете *Build Array* (създаване на масив) и го вмъкнете в блоковата диаграма.
- По подразбиране функцията *Build Array* създава само един елемент. За да създадете масив щракнете върху елемента с десния клавиш на мишката и изберете *Change to Array* (Превръщане в масив). Позиционирайте елемента извън цикъла и го свържете, както е показано на фиг.3.26.
- За да добавите възможност за запазване на данните във файл от поле *File I/O* изберете функцията *Write to Spreadsheet File* (Запис във файл).




*Фиг. 3.26. Промяна на блоковата диаграма*

### 2.8.5. Изпълнение на *VI*

- Активирайте предния панел.
- Включете "захранването" и щракнете върху бутон *Run*. В полето за графика, започва да се извеждат данните за температурата в графичен вид.
- Спрете изпълнението като изключите "захранването" (*off* позиция).
- На екрана ще се появи диалогов прозорец. В него трябва да зададете името на файла, в който ще се съхранят получените резултати.
- В полета *Max*, *Min* и *Mean* ще се изведе средноаритметичната стойност на всички измервания.
- В поле *Temperature Array* ще се изведат данните за температурата в цифров вид. За да получите информация за други елементи изберете номер от полето за индекс на масива.
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

## 2.9. ЧЕТЕНЕ НА ДАННИ ОТ ФАЙЛ

В този модул трябва да се създаде виртуален инструмент, който да чете данните от файла, създаден в предходното упражнение.

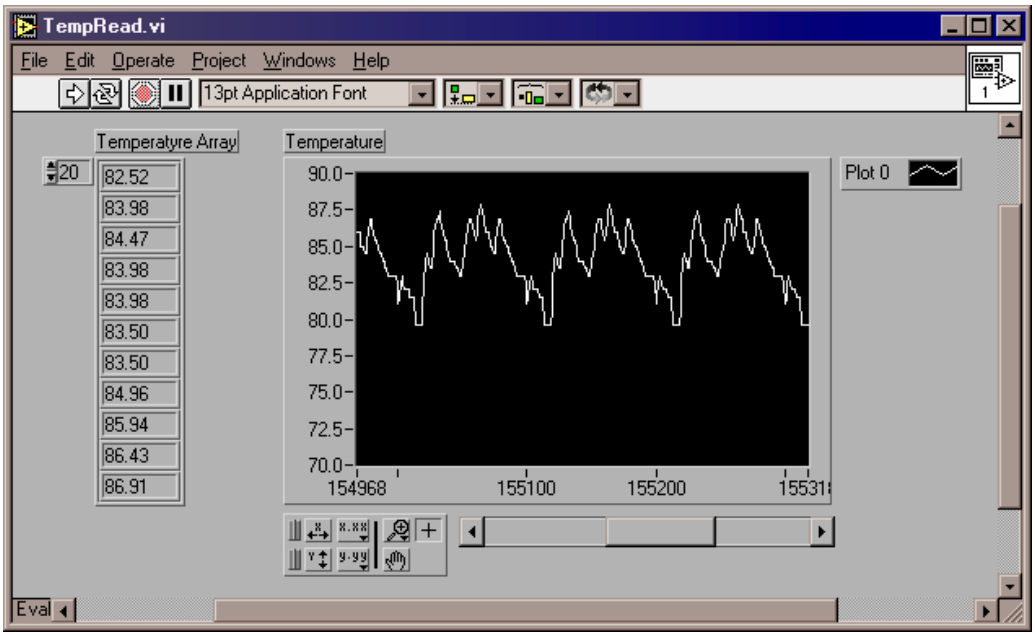
### 2.9.1. Симулиране и експериментално първоначално установяване

Да предположим, че сме запазили данни във файл и искаме да ги визуализираме на екрана в табличен и графичен вид.

- Отворете нов *VI*, като от меню *File* изберете командата *New*.

### 2.9.2. Създаване на преден панел

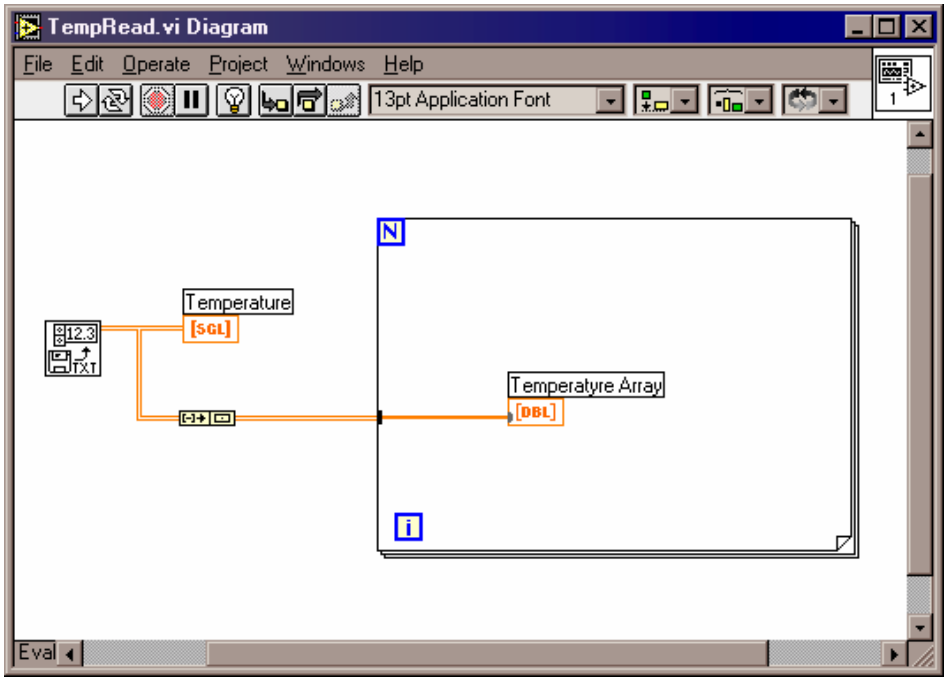
- Първо от поле *Array & Cluster* на панел *Controls* изберете *Array* (Масив). Поставете му име *Temperature Array.*
- След това трябва да укажете типа на масива. Тъй като температурата се получава във вид на числа, от поле *Numeric* на панел *Controls* трябва да изберете *Digital Indicator*. Чрез влачене и пускане го вмъкнете в полето на масива (фиг.3.27). По този начин се указва, че елементите на масива ще бъдат числови стойности.
- За извеждане на графичната информация изберете *Waveform Chart* от поле *Graph* на панел *Controls*. Поставете етикет *Temperature*. Позиционирайте управляващите елементи, както е показано на фиг.3.27.
- Щракнете с десния клавиш на мишката върху графиката и от появилото се меню включете опцията *Scrollbar* на командата *Show*. Под полето за графика ще се появи хоризонтален плъзгач, с чиято помощ потребителя може да преглежда данните.




*Фиг. 3.27. Създаване на предния панел*

### 2.9.3. Промяна на блоковата диаграма

- За да добавите възможност за четене на данни от файл от поле *File I/O* трябва да се избере функцията *Read from Spreadsheet File* (Четене от файл).
- Добавете функция, която да създаде масив от прочетените данни. От поле *Array* на панел *Functions* изберете *Build Array* (създаване на масив) и го вмъкнете в блоковата диаграма.
- По подразбиране функцията *Build Array* създава само един елемент. За да създадете масив щракнете върху елемента с десния клавиш на мишката и изберете *Change to Array* (Превръщане в масив).
- За да се прехвърлят данните от масива в таблицата на предния панел е необходимо използване на цикъл. Този път може да се използва цикъл *For*. За целта изберете *For Loop* от поле *Structures* на панел *Functions*. Когато *For Loop* се появи, той може да закрие някой от обектите от диаграмата. Ако това се получи преместете *For Loop* като използвате инструмента за позициониране.
- Уголемете *For Loop*, така че да може да включи в себе си всички други елементи. За целта позиционирайте курсора върху някой от ъглите на цикъла докато се появи маркер за промяна на големината. Щракнете и изтеглете, за да уголемите цикъла.
- Преместете полето *Temperature Array* вътре в цикъла.
- Свържете елементите на блоковата диаграма така, както е показано на фиг.3.28. По подразбиране за цикъл For е включено автоматичното индексиране на масиви.




*Фиг. 3.28. Създаване на блоковата диаграма*

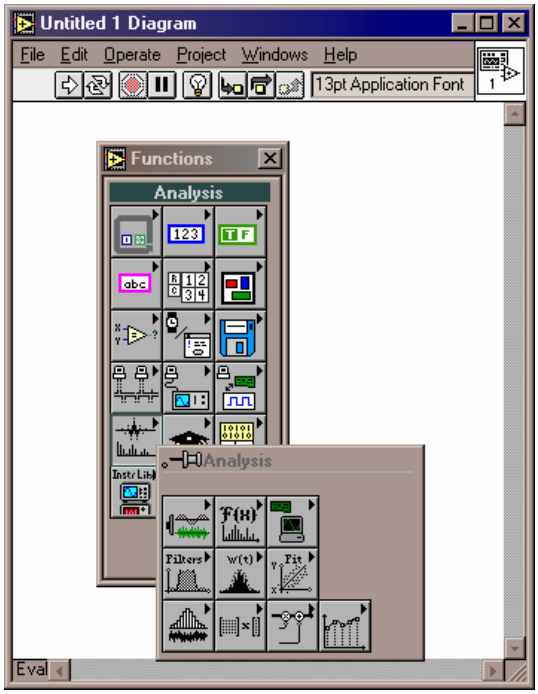
### 2.9.4. Изпълнение на *VI*

- Активирайте предния панел.
- Щракнете върху бутон *Run*. На екрана ще се появи диалогов прозорец. В него трябва да зададете името на файла, от който ще се четат данните.
- В полето за графика, се извеждат данните за температурата в графичен вид. За да разгледате други измервания използвайте хоризонталния плъзгач.
- В поле *Temperature Array* ще се изведат данните за температурата в цифров вид. За да получите информация за други елементи изберете номер от полето за индекс на масива.
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

## 3. АНАЛИЗ НА ДАННИТЕ

Модерните, бързи цифрови процесори, работещи с плаваща запетая заемат все по-голямо място в системите за анализ в реално време. Някои от многото възможни приложения са анализ на медицински резултати, синтез и разпознаване на говор, обработка на звук и изображения.

Много важно е средите за обработване на данни да притежават пълни библиотеки с функции за анализ, тъй като по този начин може да се получи достоверна информация. Чрез анализиране и обработване на получени данни в цифров вид, потребителя може например чрез елиминиране на шума да извлече полезната информация и да я представи в много по-разбираем вид.



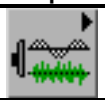
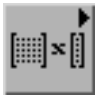

*Фиг. 3.29. Поле Analysis на панел Functions*

*LabVIEW* притежава голям набор от виртуални инструменти, с чиято помощ се улеснява разработването на приложения за анализ на данни.

Основната библиотека за анализ включва *VI* за статистически анализ, линейна алгебра и числен анализ. Разширената библиотека включва допълнителни *VI* за генериране на сигнали, времеви и честотни алгоритми, изглаждащи прозорци, цифрови филтри, регресионен анализ.

За работа с виртуалните инструменти на *LabVIEW* за анализ трябва да се избере поле *Analysis* от панел *Functions* на блоковата диаграма (фиг.3.29). Съществуват 10 библиотеки за анализ на данни:

Таблица 3.1

| Диаграма | Описание |
|---|---|
|  | Генератор на сигнали (*Signal Generation*)<br>Тези *VI* служат за генериране на цифрови сигнали. |
|  | *Цифрова обработка на сигнали (Digital Signal Processing)*<br>Тези *VI* извършват честотни преобразования и анализ, анализ във времето, както и преобразования на Хилберт и Хартли. |
|  | *Функции за измерване (Measurement Functions)*<br>Тези *VI* изпълняват измервателни функции като изчисляване на спектър, мащабиращи прозорци и др. |
|  | *Цифрови филтри (Digital Filters)*<br>Тези *VI* извършват цифрова филтрация на сигнали. |
|  | *Прозоречни функции (Windowing Functions)*<br>Тези *VI* предоставят възможност за прилагане на изглаждащи прозорци. |
|  | *Вероятностен и статистически анализ (Probability and Statistics Functions)*<br>Тези *VI* предоставят функции за вероятностен и статистически анализ. |
|  | *Апроксимиране на криви (Curve Fitting Functions)*<br>Тези *VI* извършват апроксимиране и интерполиране. |
|  | *Линейна алгебра (Linear Algebra Functions)*<br>Тези *VI* изпълняват алгебрични функции върху реални и комплексни вектори и матрици. |
|  | *Операции с масиви (Array Operations)*<br>Тези *VI* изпълняват операции с едномерни и двумерни масиви. |
|  | *Допълнителни числени методи (Additional Numerical Methods)*<br>Тези *VI* използват числени методи за изчисляване корени на уравнения, числено интегриране и намиране на екстремни точки. |

### 3.1. ГЕНЕРИРАНЕ НА СИНУСОИДАЛЕН СИГНАЛ

*LabVIEW* притежава специална библиотека с *VI*, с чиято помощ могат да се генерират сигнали. Някой от приложенията, където може да се използва генерирането на сигнали са:

- Тестване на алгоритми, когато не е възможно получаването на реални сигнали.
- Генериране на сигнали, които да се използват от цифрово-аналогов преобразувател.

### 3.1.1. Нормализирана честота

Честотата на аналоговите сигнали се измерва в Hz или брой цикли в секунда. Цифровите системи често използват цифрова честота, която представлява съотношение между аналоговата честота и честотата на дискретизация:

••••••• ••••••• = ••••••••• ••••••• / ••••••• •• ••••••••••••

Тази цифрова честота се нарича нормализирана и се измерва в цикли/дискрета.

Някой от виртуалните инструменти използват нормализран честотен вход. Тази входна честота варира от 0.0 до 1.0, което отговаря на реална честота от 0 до честотата на дискретизация $f_s$. Например сигнал, който е дискретизиран с честотата на Найквист ($f_s$/2) ще има две дискрети за един цикъл. Реципрочната стойност на нормализираната честота 1/*f* дава броя на дискретите в един цикъл.

Когато се използват *VI*, изискващи входна нормализирана честота, предварително честотата трябва да бъде конвертирана към брой цикли / дискрета. Нормализирана честота може да бъде използвана с *VI* за генериране на:

- Синусидален сигнал
- Правоъгълен сигнал
- Трионообразен сигнал
- Триъгълен сигнал
- Случаен сигнал

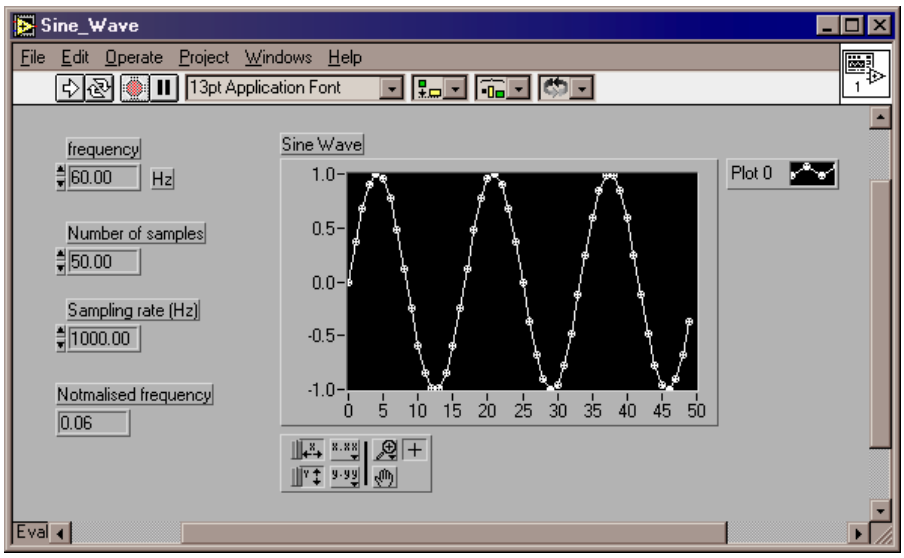
### 3.1.2. Симулиране и експериментално първоначално установяване

Следващата задача, която трябва да изпълним е да създадем *VI* за генериране и визуализиране на синусоидален сигнал със зададена от потребителя честота в Hz. Виртуалният инструмент трябва да изчислява и визуализира нормализираната честота.

- Отворете нов *VI*, като от меню *File* изберете командата *New*.

### 3.1.3. Създаване на преден панел

- Първо от поле *Numeric* на панел *Controls* изберете *Digital Control*. Поставете му име *Frequency* (Аналогова честота).
- По същия начин добавете още два цифрови входни елемента, като им зададете съответно имена *Number of samples* (Брой дискрети) и *Sampling rate* (Hz) (Честота на дискретизация).
- От поле *Numeric* на панел *Controls* добавете *Digital Indicator*. Задайте му име *Normalised frequency* (Нормализирана честота). Този индикатор ще служи за извеждане на нормализираната честота.

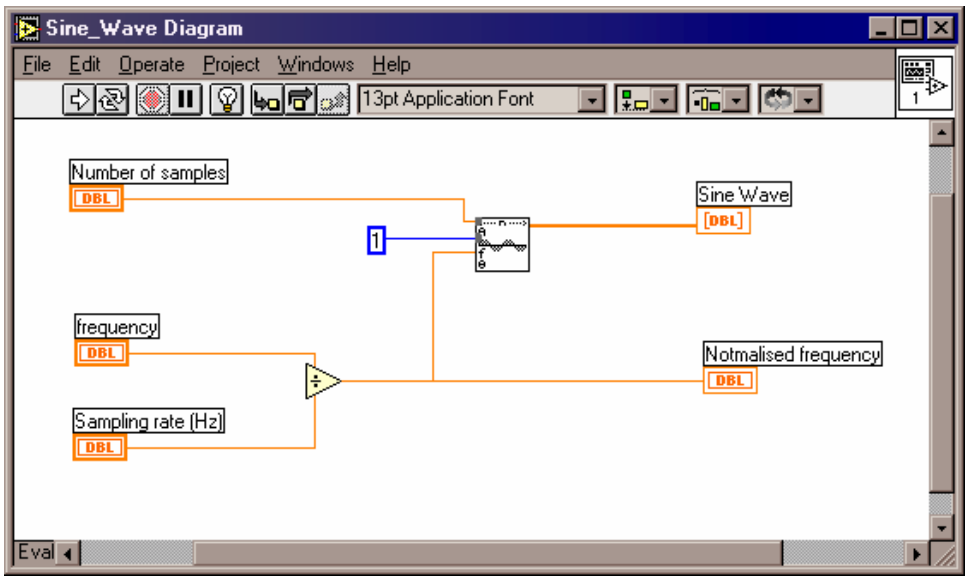



*Фиг. 3.30. Създаване на предния панел*

- За извеждане на графичната информация изберете *Waveform Graph* от поле *Graph* на панел *Controls*. Поставете етикет *Sine Wave*. Позиционирайте управляващите елементи, както е показано на фиг.3.30.

### 3.1.4. Промяна на блоковата диаграма

- От меню *Windows* изберете *Show Diagram* (Покажи диаграма).
- От меню *Edit* изберете командния ред *Functions* и на екрана ще се появи неговия панел.
- От поле *Numeric* изберете *Numeric Constant* и я добавете към диаграмата. Задайте стойност 1 като използвате инструмента за писане. Тази константа задава амплитудата на синусоидалния сигнал.
- От същото поле *Numeric* изберете и функцията за деление и я добавете към блоковата диаграма.
- От библиотека *Signal Generation* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *Sine Wave* и го добавете към блоковата диаграма.
- Свържете елементите както е показано на фиг.3.31.




*Фиг. 3.31. Създаване на блоковата диаграма*

### 3.1.5. Изпълнение на *VI*

- Активирайте предния панел. Задайте в поле *Frequency* стойност 60, за брой дискрети (в поле *Number of samples*) задайте стойност 50, а за честота на дискретизация задайте 1000.
- Щракнете върху бутон *Run*.
- В полето за графика ще се изведат данните за температурата в графичен вид (фиг.3.30).
- В полето за нормализирана честота ще се изведе стойност 0.06, което представлява (брой цикли / дискрета). Реципрочната стойност на нормализираната честота е ≈17 (1/0.06), което представлява брой дискрети за генериране на един цикъл от синусоидалния сигнал.
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

## 3.2. СЪЗДАВАНЕ НА ФУНКЦИОНАЛЕН ГЕНЕРАТОР. CASE СТРУКТУРА

В този модул трябва да бъде разширен виртуалният инструмент за генериране на синусоидален сигнал, като му се добавят възможности за генериране и на правоъгълен, триъгълен и трионовиден сигнал.

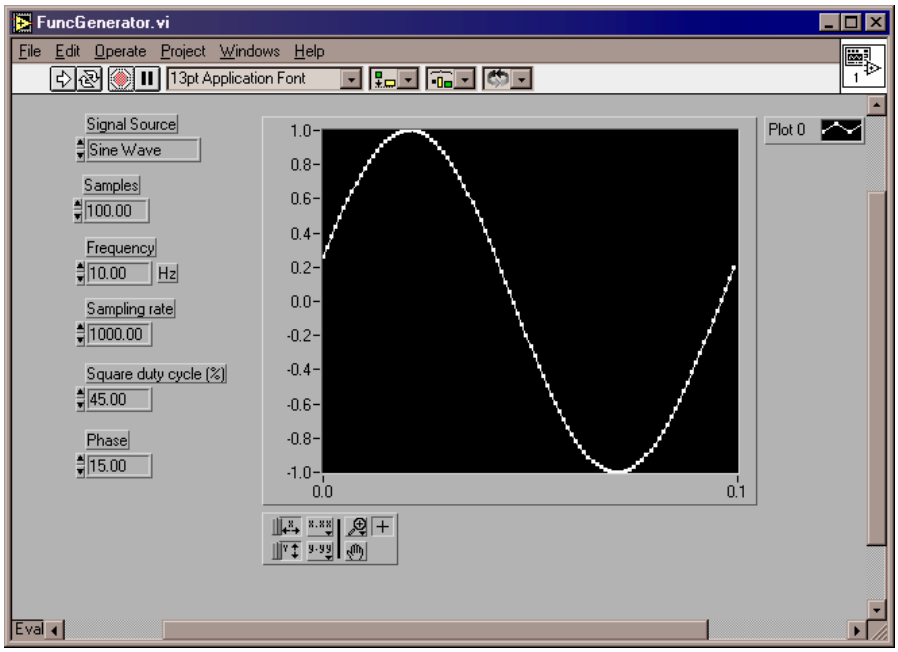
### 3.2.1. Симулиране и експериментално първоначално установяване

Нека предположим, че трябва да се генерират различни типове сигнали в зависимост от желанието на потребителя.

- Отворете *VI*, който създадохте в предишното упражнение.
- Изберете *Save As*... и запазете *VI* под ново име.

### 3.2.2. Промяна на панела

- Добавете още два цифрови входа като от поле *Numeric* на панел *Controls* изберете *Digital Control.* Задайте им съответно имена *Phase* (Фаза) и *Square duty cycle* (%) (Запълване на правоъгълния сигнал).
- Добавете текстов вход като от поле *List & Ring* на панел *Controls* изберете *Test Ring.* Задайте му име *Signal Source* (фиг.3.32).




*Фиг. 3.32. Промяна на предния панел*

- Щракнете с десния клавиш върху текстовия вход. От падащото меню изберете *Add Item*. В полето напишете Sine Wave. По същия начин добавете още три полета на текстовия вход и им задайте съответно стойности: *Square Wave*, *Triangle Wave* и *Sawtooth Wave*.

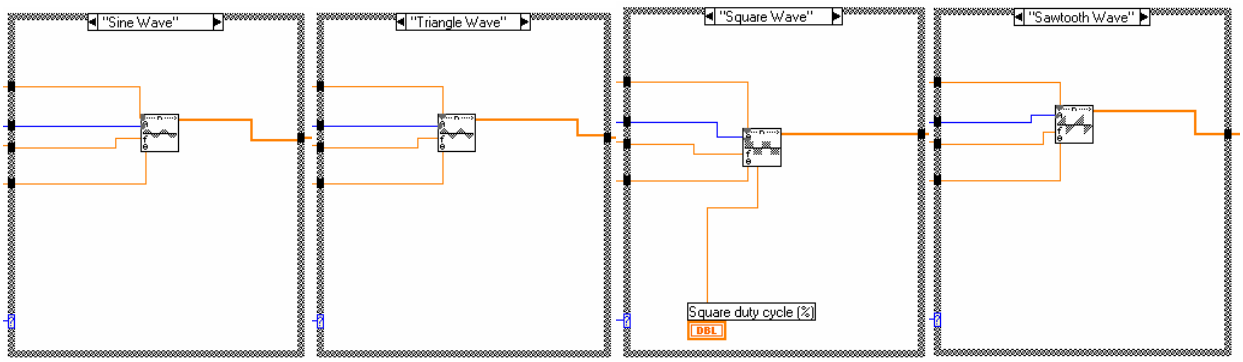
### 3.2.3. Промяна на блоковата диаграма

За да се извърши намиране на минималната и максималната стойност е необходима промяна на блоковата диаграма.

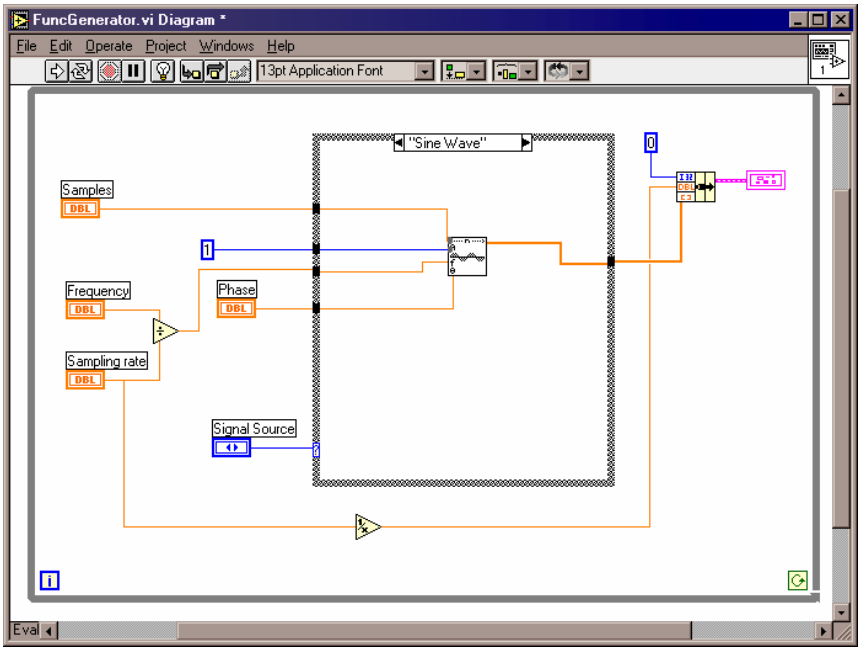
- Изберете *While Loop* от поле *Structures* на панел *Functions*. Когато *While Loop* се появи, той може да закрие някой от обектите от диаграмата. Ако това се получи преместете *While Loop* като използвате инструмента за позициониране.
- Уголемете *While Loop*, така че да може да включи в себе си всички други елементи. За целта позиционирайте курсора върху някой от ъглите на цикъла докато се появи маркер за промяна на големината. Изтеглете, за да уголемите цикъла.
- Изберете всички елементи като щракнете с маркера за

позициониране отгоре вляво на *VI* и след това издърпате надолу вдясно. Правоъгълникът, избира всички обекти, които се намират вътре в него, когато се отпусне бутона на мишката. Преместете избраните обекти в *While Loop*.

- От поле *Cluster* на панел *Functions* добавете *Bundle*. Уголемете го, така че да има три входа.
- От поле *Numeric* изберете *Numeric Constant* и я добавете към диаграмата. Задайте стойност 0 като използвате инструмента за писане.
- Изберете структура *Case* от поле *Structures* на панел *Functions* и я вмъкнете в диаграмата.
- Щракнете с десния клавиш върху правоъгълника в горната част на структурата *Case*. От падащото меню изберете *Add Case* (добавяне на условие).
- На новото условие задайте име *Triangle Wave*.
- Повторете последните две стъпки още два пъти, за да създадете още две условия. Задайте им съответно имена *Square Wave* и *Sawtooth Wave.* Между условията може да се превключва чрез последователно щракване с левия клавиш на мишката върху символите◄ и ► в лявата и дясната част на полето с надписа на условието.
- В структурата *Case* последователно вмъкнете генераторите на синусоидален, триъгълен, паравоъгълен и трионовиден сигнал (типа на генератора трябва да съвпада с името на условието). За целта от библиотека *Signal Generation* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *Sine Wave* и го добавете към блоковата диаграма. Аналогично в *Case* структурата добавете *Square Wave*, *Triangle Wave* и *Sawtooth Wave* (фиг.3.33).
- Свържете елементите както е показано на фиг.3.34 и фиг.3.35 като не забравяте, че трябва да се свържат всички генератори на сигнали от *Case* структурата.



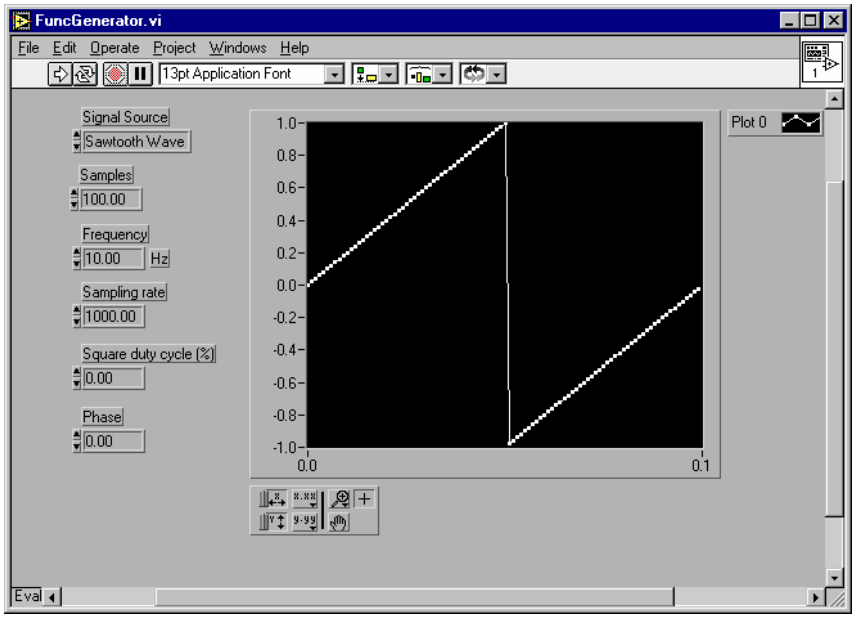

*Фиг. 3.33. Case структура с различни генератори на сигнали*




*Фиг. 3.34. Промяна на блоковата диаграма*

### 3.2.4. Изпълнение на *VI*

- Активирайте предния панел. Задайте в поле *Frequency* стойност 10, за брой дискрети (в поле *Samples*) задайте стойност 100, а за честота на дискретизация задайте 1000.
- Щракнете върху бутон *Run*.
- В полето за графика ще се изведат данните в графичен вид (фиг.3.35).




*Фиг. 3.35. Генериране на трионовиден сигнал*

- От поле *Signal Source* изберете друг генератор на сигнал. В полето за графика ще се изведе новия тип сигнал.
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

# 4. ЦИФРОВА ОБРАБОТКА НА СИГНАЛИ

## 4.1. ИЗВЛИЧАНЕ НА СИНУСОИДАЛЕН СИГНАЛ ЧРЕЗ ИЗПОЛЗВАНЕ НА ЦИФРОВ ФИЛТЪР

Проектирането на аналогови филтри е една от най-важните области на електрониката. Често проектирането им се възлага на специалисти, тъй като проектирането на аналогови филтри изисква отлична математическа подготовка и анализ на процесите в системата, където ще се прилагат.

Съвременното развитие на софтуера прави възможно заместването на аналогови филтри с цифрови в приложения, изискващи адаптивност и препрограмируемост. Такива приложения включват аудио, телекомуникации, медицинска диагностика и др.

Цифровите филтри имат следните предимства пред аналоговите:

- те са софтуерно препрограмируеми;
- стабилни са и са предсказуеми;
- не се влияят от температура и влажност и не изискват прецизни елементи;
- имат много ниска цена.

*LabVIEW* притежава богата библиотека от цифрови филтри (*Butterworth*, *Chebishev*, *Eliptic* и др.), чието използване е изключително лесно. Потребителят не трябва да бъде експерт по цифрови филтри, за да обработи своите данни.

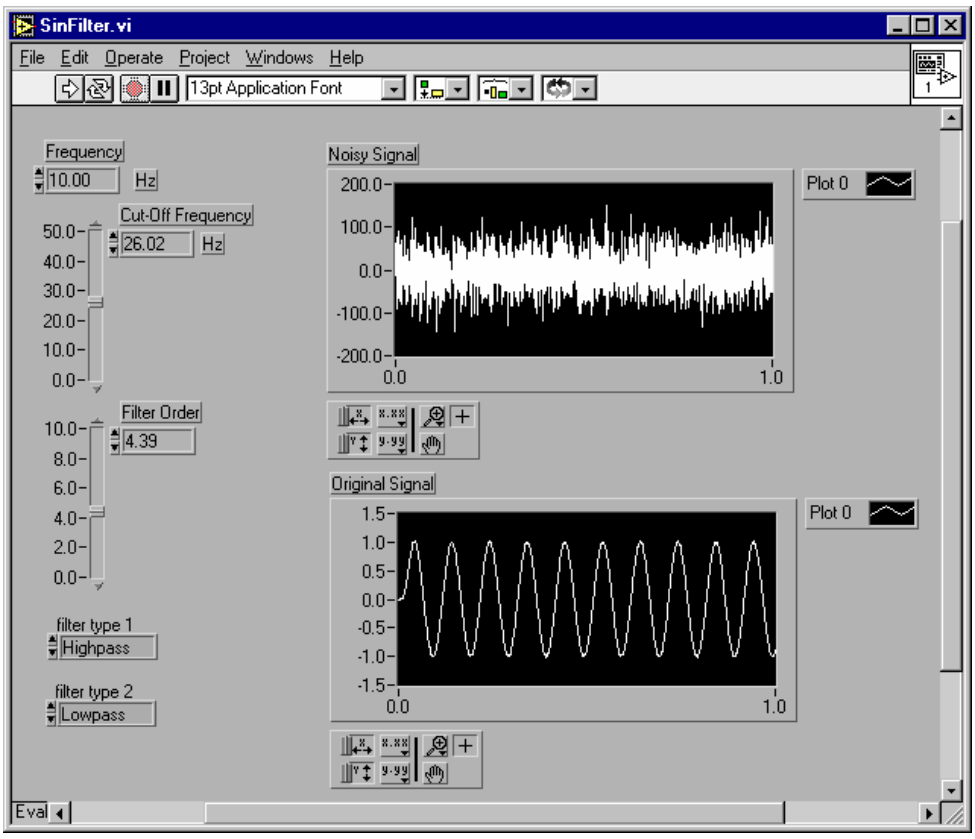
В този модул трябва да бъде създаден виртуален инструмент, които да филтрира данни, състоящи се от смесен високочестотен шум и от синусоиделен сигнал. За целта синусоидален сигнал, генериран от виртуалния инструмент *Sine Pattern VI* се комбинира с високочестотен шум. След това този сигнал се филитрира с нискочестотен *Butterworth* филтър и се извлича синусоидалния сигнал.

### 4.1.1. Симулиране и експериментално първоначално установяване

- Отворете нов *VI* , като от меню *File* изберете командата *New*.

### 4.1.2. Създаване на предния панел

- Добавете цифров вход като от поле *Numeric* на панел *Controls* изберете *Digital Control.* Задайте му име *Frequency* (*Честота*).

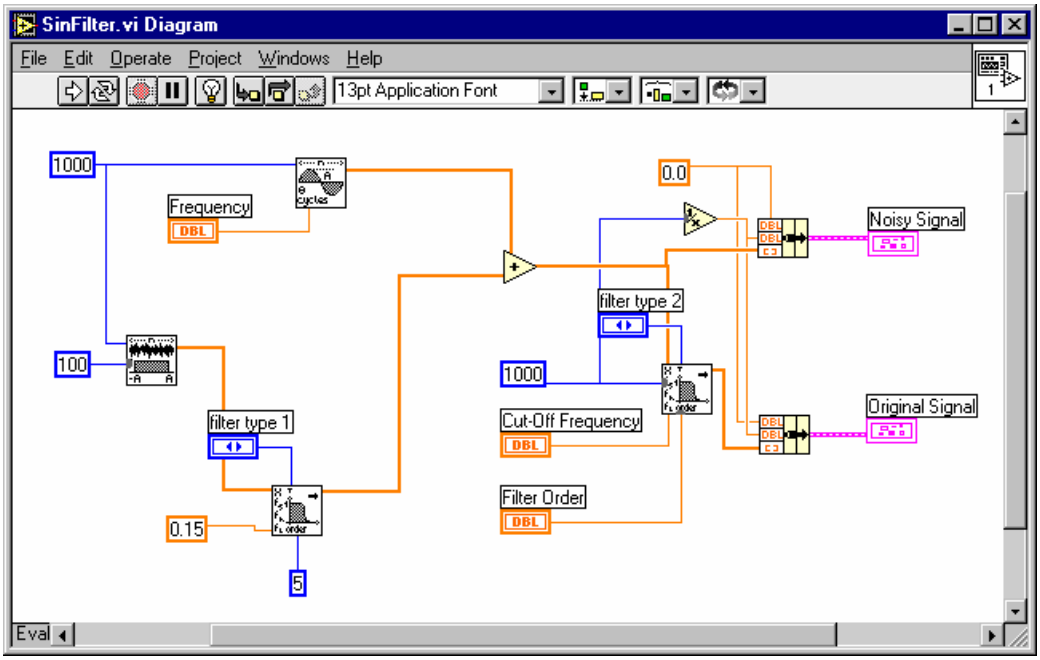



*Фиг. 3.36. Преден панел*

- Добавете вертикален плъзгач като от поле *Numeric* на панел *Controls* изберете *Vertical Slide.* Задайте му име *Cut-Off Frequency (Честота на среза)*.
- Добавете още един вертикален плъзгач като от поле *Numeric* на панел *Controls* изберете *Vertical Slide.* Задайте му име *Filter Order (Порядък за филтриране)*.
- Добавете контрол за задаване типа на първия филтър. За целта от поле *List & Ring* на панел *Controls* изберете *Text Ring*. Задайте му име *Filter type 1*. Като използвате инструмента за писане напишете в полето *Highpass* (Високочестотен).
- Добавете контрол за задаване типа на втория филтър. За целта от поле *List & Ring* на панел *Controls* изберете *Text Ring*. Задайте му име *Filter type 2*. Като използвате инструмента за писане напишете в полето *Lowpass* (Нискочестотен).
- За извеждане на графична информация изберете *Waveform Graph* от поле *Graph* на панел *Controls*. Поставете етикет *Noisy Signal (Разшумен сигнал)*.
- Добавете още едно поле за извеждане на графична като информация изберете *Waveform Graph* от поле *Graph* на панел *Controls*. Поставете етикет *Original Signal (Оригинален сигнал)*. Позиционирайте управляващите елементи, както е показано на фиг.3.36.

### 4.1.3. Промяна на блоковата диаграма

- Изберете генератор на синусоидален сигнал като за целта от библиотека *Signal Generation* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *Sine Wave VI.*
- Към блоковата диаграма добавете генератор на бял шум като за целта от библиотека *Signal Generation* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *White Noise VI*.
- Добавете високочестотен филтър на белия шум като от библиотека *Filters* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *Butterworth Filter VI*.

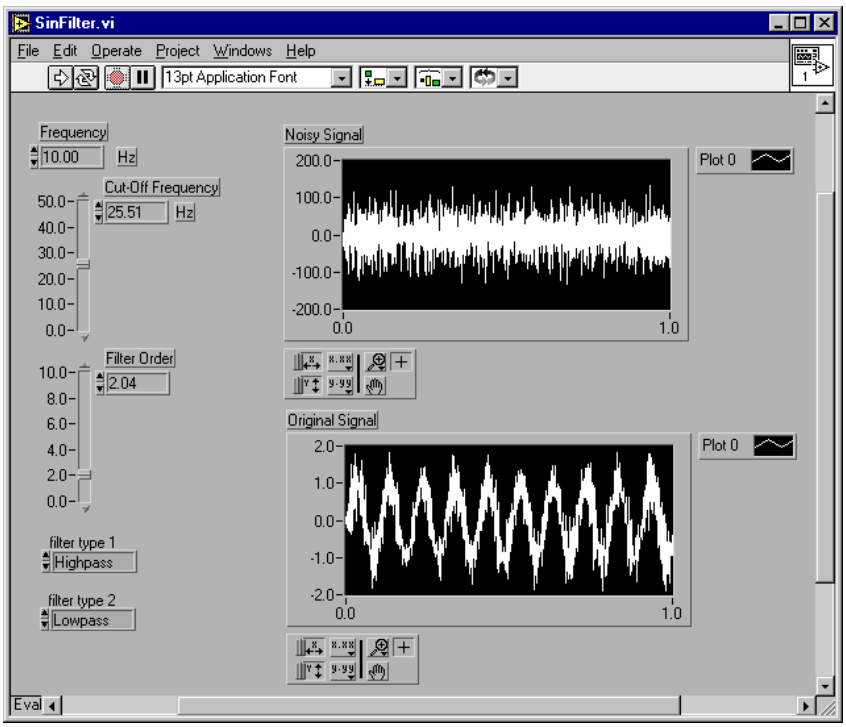



*Фиг. 3.37. Блокова диаграма*

- Добавете нискочестотен филтър на смесения сигнал като от библиотека *Filters* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *Butterworth Filter VI*.
- Добавете необходимите допълнителни елементи и ги свържете както е показано на фиг.3.37.

### 4.1.4. Изпълнение на *VI*

- Активирайте предния панел. Задайте в поле *Frequency* стойност 10 Hz, за честота на среза (в поле *Cut-Off Frequency*) задайте стойност 25 Hz, а за порядък на филтъра (в поле *Filter Order*) задайте стойност 5.
- Щракнете върху бутон *Run*.
- В полетата за графика ще се изведат данните в графичен вид.
- Променяйте порядъка на филтриране и обърнете внимание на разликата във филтрирания сигнал (фиг.3.38) спрямо фиг.3.36.




*Фиг. 3.38. Разшумен и филтриран сигнал*

- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

## 4.2. ПОЛУЧАВАНЕ ЧЕСТОТНИЯ СПЕКТЪР НА СИГНАЛ

В този модул трябва да бъде създаден виртуален инструмент, които да изчисли и визуализира честотния спектър на получения в предишната задача сигнал.

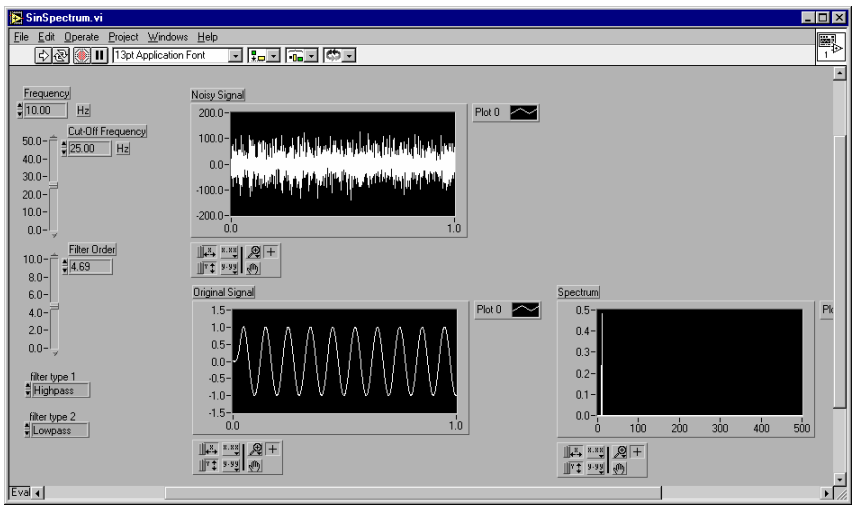
### 4.2.1. Симулиране и експериментално първоначално установяване

Нека предположим, че трябва да се получи честотния спектър на сигнал.

- Отворете *VI*, който създадохте в предишното упражнение.
- Изберете *Save As*... и запазете *VI* под ново име.

### 4.2.2. Промяна на панела

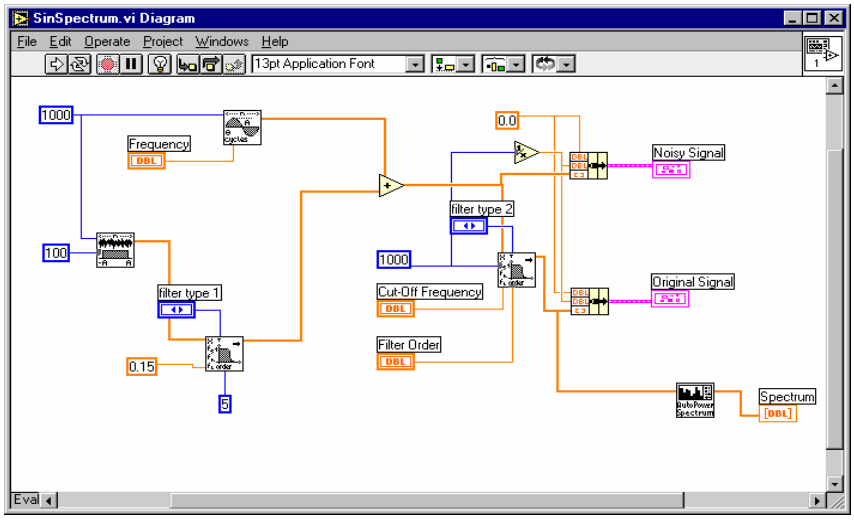
- Добавете още едно поле за извеждане на графична информация като изберете *Waveform Graph* от поле *Graph* на панел *Controls*. Поставете етикет *Spectrum (Спектър)*. Позиционирайте управляващите елементи, както е показано на фиг.3.39.




*Фиг. 3.39. Промяна на предния панел*

### 4.2.3. Промяна на блоковата диаграма

- Добавете виртуален инструмент за изчисляване спектъра на сигнала. За целта от библиотека *Measurement* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *Auto Power Spectrum VI.*




*Фиг. 3.40. Промяна на блоковата диаграма*

- Към блоковата диаграма добавете генератор на бял шум като за целта от библиотека *Signal Generation* на поле *Analysis* от панел *Functions* изберете виртуалния инструмент *White Noise VI*.
- Свържете новите елементи, както е показано на фиг.3.40.

### 4.2.4. Изпълнение на *VI*

- Активирайте предния панел. Задайте в поле *Frequency* стойност 10 Hz, за честота на среза (в поле *Cut-Off Frequency*) задайте стойност 25 Hz, а за порядък на филтъра (в поле *Filter Order*) задайте стойност 5.
- Щракнете върху бутон *Run*.
- В полето *Spectrum* ще се изведе спектъра на сигнала.

## 5. ИЗГРАЖДАНЕ НА ВИРТУАЛНИ СИСТЕМИ ЗА ИЗМЕРВАНЕ НА СИГНАЛИ И АВТОМАТИЗАЦИЯ

Измерванията на сигнали се извършват с помощта на специализирани инструменти. Първите аналогови инструменти за измерване са се появили през 19 век, когато производството на часовници е било водеща технология.

Виртуалните инструменти представляват комбинация от хардуер и софтуер с индустриални компютърни технологии. Това позволява създаването на потребителски дефинирани решения. Като хардуерна част от виртуалните инструменти могат да бъдат използвани *DAQ* (*Data Acquisition*) устройства, *GPIB* инструменти, *VXI* инструменти и инструменти за сериен интерфейс. За софтуерна част могат да бъдат използвани програмни пакети, като *LabVIEW*, *LabWindows*/*CVI*, *Component Works* и др. Комуникацията между хардуера и софтуерните платформи се осъществява от специализирани драйвери.

С използването на виртуални инструменти е възможно бързо създаване на системи за измерване, тестване и автоматизация чрез комбинация от хардуер и софтуер по желание на клиента. При промяна на изследваната система виртуалният инструмент може да бъде допълнен с хардуер и софтуер.

### 5.1. СРАВНЕНИЕ НА *DAQ* УСТРОЙСТВА И СПЕЦИАЛИЗИРАНИ ИНСТРУМЕНТИ ЗА ИЗМЕРВАНЕ

Предназначението на измервателните устройства като например *DAQ* с общо предназначение и специализираните инструменти е получаване на данни, анализ и представяне на измерванията.

Получаването на данни (*Data Acquisition*) е процес, при който физическите сигнали, като например напрежение, ток, налягане или температура се преобразуват в цифров вид и се предават на компютър. Широко използвани методи за получаване на данни са *DAQ* устройства, *GPIB*, *VXI* инструменти и инструменти за сериен интерфейс.

Анализът на данните преобразува първоначалните данни в полезна информация. Анализът може да включва: апроксимиране, статистически анализ, честотен анализ и др.

Представянето на данните от измерванията трябва да осигурява интуитивна и цялостна комуникация със системата.

Създаването на компютърна измервателна система може да бъде трудна задача. В настоящия момент има голям набор от хардуерни компоненти, които могат да бъдат използвани за следене и управление на процеси или за изследване на обекти.

Основната задача на всички измервателни системи е измерването и/или генерирането на реални физически сигнали. Първата разлика между различните хардуерни възможности е в метода на комуникация

между измервателния хардуер и компютъра.

*DAQ* устройствата с общо предназначение се свързват с компютър, давайки възможност на потребителя да получи дигитализирани данни. Този тип устройства основно са вътрешни и се свързват с компютъра директно чрез plug-in слот. Някои *DAQ* устройства са външни и се свързват с компютър посредством сериен, *GPIB* или *Ethernet* порт. *DAQ* устройствата единствено преобразуват постъпващите сигнали в цифров вид, който се изпраща към компютъра. Този тип устройства не извършва изчисления на крайните резултати. Тази задача се извършва от софтуера, намиращ се на компютъра. Едно и също *DAQ* устройство може да извършва множество измервания, при което единствено се налага промяна на програмното осигуряване за обработка на данните. Предимство на този метод за измерване е, че се постига голяма гъвкавост - може да се използва едно хардуерно устройство и различно програмно осигуряване. Недостатък е, че за всеки нов тип измерване потребителя трябва да разработва нов софтуер.

Специализираните инструменти, подобно на *DAQ* устройствата с общо предназначение, също извършват измервания и дигитализират данните. Но те притежават и специални измервателни възможности. Софтуерът, необходим за обработване на данните и изчисляване на резултата е обикновено вграден в инструментите и не може да бъде променян. По-голямата част от специализираните инструменти са външни за компютъра и могат да работят самостоятелно или да бъдат управлявани посредством връзка с компютър. Инструментите притежават специфичен протокол, които трябва да бъде поддържан от компютъра, за да бъде осъществена комуникацията между тях. Връзката с компютъра може да бъде *Ethernet*, *RS-232*, *GPIB* или *VXI*. Някои от специализираните инструменти могат да бъдат инсталирани в компютъра, подобно на *DAQ* с общо предназначение. Те се наричат компютърно-базирани инструменти.

#### 5.1.1.1. Връзка компютър-*DAQ* устройство

Преди компютърно-базирана система да измери даден физически сигнал, е необходимо първичен преобразувател да преобразува физическия сигнал в електрически, например ток или напрежение.

Възможни са 3 конфигурации за реализиране на *DAQ* система:

- чрез използване на *plug-in DAQ* устройство. То се инсталира в компютъра. За разлика от повечето специализирани инструменти, полученият от първичния преобразувател сигнал не може да се подаде директно на *plug-in DAQ* устройство. Необходимо е използването на допълнителни устройства за преобразуване на сигнала, преди такъв тип *DAQ* устройство да го преобразува в цифров вид. Софтуерът управлява *DAQ* устройството като получава първичните данни, обработва ги, извършва анализ и ги визуализира.

- чрез използване на външно *DAQ* устройство. По този начин може да бъде изградена *DAQ* система, използваща компютър без *plug-in* слотове, каквито са преносимите компютри. Компютърът и *DAQ* модула могат да си обменят информация през паралелния порт, серийния порт или *Ethernet*. Такива системи са подходящи за отдалечено измерване и управление.
- чрез *DAQ* устройство, използващо *PCMCIA* шина. Такова решение е подходящо за портативни, компактни *DAQ* системи. Както и при първата конфигурация е необходимо допълнително преобразуване на сигнала, получен от първичните преобразуватели.

#### 5.1.1.2. Задачи на програмното осигуряване

Компютърът получава необработени данни. Една от задачите на софтуера е да обработи данните и да ги представи във вид удобен за възприемане. Друга задача е да управлява *DAQ* системата, като указва кога и от кои канали да се извършва измерване.

Обикновено *DAQ* софтуерът включва драйвери и приложна програма. Драйверите са уникални за дадено устройство (или фамилия устройства) и представляват набор от команди, които устройството възприема. Приложният софтуер, като *LabVIEW*, изпраща команди към драйвера, анализира и визуализира получените данни.

*LabVIEW* притежава набор от виртуални инструменти, с чиято помощ могат да бъдат конфигурирани *DAQ* устройства, както и да се получават и изпращат данни към тях. *LabVIEW DAQ VI* от своя страна правят обръщение към *NI-DAQ Application Program Interface* (*API*).

#### 5.1.2.1. Връзка компютър-специализиран инструмент

За разлика от *DAQ* устройствата, сигналът, получаван от специализираните инструменти не изисква допълнително преобразуване. Как компютърът ще управлява инструмента и ще получава данните зависи от начина, по който инструмента е изграден: *GPIB*, серийни, *VXI*, *PXI* или компютърно-базирани.

Всички външни инструменти комуникират с компютъра посредством определен интерфейс и протокол. Всеки инструмент притежава набор от команди, които разбира. Задачата на потребителя е да създаде софтуер, който изпраща такива команди и получава данни от инструмента. За целта трябва да се изясни начина, по който приложението и специализирания инструмент ще си комуникират.

#### 5.1.2.2. Задачи на програмното осигуряване

Драйверите за специализираните инструменти са изключително важен фактор за разработването на компютърни измервателни системи. Драйверът представлява набор от функции, изпълняващи командите

необходими за работа с инструмента. Драйверите, включени в *LabVIEW* опростяват програмирането на инструментите до команди от високо ниво. За управлението на инструментите не е необходимо изучаването на специфичен синтаксис от ниско ниво. По този начин значително се намалява времето за разработване на дадено програмно осигуряване.

Инструменталните драйвери създават командите за инструментите и комуникират с тях по сериен, *GPIB* или *VXI* интерфейс. Освен това, инструменталните драйвери получават, анализират и скалират стринговата информация от инструментите по такъв начин, че да може да се използва от програмното осигуряване.

Инструменталните драйвери правят поддръжката на програмите по-лесна, тъй като се съхраняват отделно от програмното осигуряване и могат лесно да бъдат обновявани при необходимост.

В настоящия момент *LabVIEW* може да работи с повече от 700 инструментални драйвери на над 50 производители.

## 5.2. ВИДОВЕ ИНСТРУМЕНТИ ЗА ИЗМЕРВАНЕ И АВТОМАТИЗАЦИЯ

Когато за създаване на измервателни системи и системи за автоматизация се използва компютър, няма ограничение в типа на инструмента, който може да се управлява. Могат да се сместват и допълват инструменти от различни категории, като серийни, *GPIB*, *VXI*, *PXI* и компютърно-базирани.

При управление на инструменти от персонален компютър трябва да се вземат под внимание следните особености:

- тип на конектора на инструмента;
- какъв кабел е необходим (брой пера, мъжки/женски);
- какви са електрическите изисквания (нива на сигнала, заземяване, ограничение в дължината на кабела);
- какви комуникационни протоколи се използват (*ASCII* команди, двоични команди, формат на данните);
- какви са наличните драйвери.

### 5.2.1. Комуникация чрез използване на серийния порт на компютъра

Серийната комуникация е популярен начин на изпращане на данни между компютър и периферно устройство, като например програмируем инструмент или друг компютър. Серийната комуникация използва предавател за изпращане на данни към приемник бит след бит по единствена комуникационна линия. Този начин на комуникация се използва, когато скоростта на обмен е ниска или трябва да се предават данни на дълги разстояния. Серийната комуникация е популярна, защото повечето компютри имат един или повече серийни порта, не е необходим допълнителен хардуер освен кабел за връзка между компютъра и

инструмента (или друг компютър).

При серийна комуникация трябва да бъдат зададени 4 параметъра:

- скорост на обмен на информацията;
- брой на данновите битове;
- използване/неизползване на бит за четност;
- брой стоп битове.

В настоящия момент съществуват няколко различни стандарта за серийна комуникация. Най-известни са следните:

***RS-232*** **(*ANSI/EIA-232*)** се използва за много цели, като връзка с мишка, принтер, модем или инструменти за измерване и автоматизация. *RS-232* е ограничен до връзка между серийните портове на персонален компютър и устройства.

***RS-422*** **(*AIA RS-422A Standard*)** използва различен тип електрически сигнал в сравнение с *RS-232*. При предаването на данни се използват две линии за предаване и получаване на сигнали. Това води до по-добра шумоустойчивост и по-големи разстояния на предаване спрямо *RS-232*, което е голямо предимство при индустриални приложения.

***RS-485*** **(*EIA-485 Standard*)** е подобрение на стандарта *RS-422* като позволява включване на няколко устройства (до 32) към един сериен порт. Този стандарт дефинира електрически характеристики, необходими да осигурят адекватен сигнал в смесена среда. С помощта на тези нови възможности може да бъде изградена мрежа от устройства, включена към един *RS-485* сериен порт. Този стандарт е широко използван в индустриални приложения, изискващи включването в мрежа на разпределени устройства за измерване.

### 5.2.2. Използване на *GPIB* (*General Purpose Industrial Bus*) инструменти

Много производители предлагат широк избор от *GPIB* инструменти (фиг.3.41). Тези инструменти могат са бъдат с общо предназначение или специализирани, като традиционно са самостоятелни устройства.


*Фиг.3.41. GPIB инструменти*

### 5.2.2.1. Контролери, предаватели и приемници (*Controllers, Talkers, Listeners*)

За определяне кое устройство има активен контрол върху интерфейса, устройствата се делят на: контролери, предаватели и приемници, като всяко устройство има уникален *GPIB* адрес със стойности между 0 и 30. Контролерът дефинира комуникационните връзки, отговаря на устройствата, изискващи обслужване, изпраща *GPIB* команди и предава/поема контрола върху интерфейса. Предавателите получават инструкции от контролера да предават данни по *GPIB*. Само едно устройство може да бъде предавател в даден момент. Приемниците получават инструкции от Контролера да четат данните от *GPIB*. Няколко устройства могат да бъдат адресирани като приемници.

### 5.2.2.2. Хардуерни спецификации

*GPIB* е цифров 24-битов паралелен интерфейс. Той се състои от 8-битова даннова магистрала, 5 управляващи линии, 8 линии земя и 3 линии за осъществяване на връзка между устройства. Тъй като при *GPIB* се изпращат и получават 8 бита (1 байт), предаваните съобщения най-често се кодират като *ASCII* символни стрингове.

Допълнителни електрически спецификации позволяват данните да бъдат предавани по *GPIB* интерфейса с максимална скорост 1 MB/sec. Тези спецификации са:

- максимално разстояние от 4 м между 2 устройства и средно разстояние от 2 м относно целия интерфейс;
- максимална дължина на кабела до 20 м;
- към една линия могат да бъдат включени максимално 15 устройства, като най-малко 2/3 от тях да са работещи.

### 5.2.3. Използване на *VXI* (*VME eXtension for Instrumentation*) модулни инструменти

*VXI* системите се изграждат на модулен принцип. Една *VXI* система се състои от следните хардуерни компоненти: основен модул, контролер, инструменти и кабели. Основният модул представлява кутия, в която се намират захранващ блок, охладителна система, вътрешни връзки. В основния модул се монтират *VXI* модули. Основният модул може да бъде в четири различни размера, отговарящи на най-големия модул, който може да бъде монтиран.

#### 5.2.3.1. *VXI* конфигурации

*VXI* може да бъде използвана по различни начини. Може да бъде интегрирана в система, заедно с други *GPIB* инструменти или може да бъде изградена система изцяло от *VXI* инструменти (фиг.3.42).

*Фиг.3.42. VXI инструменти*

Съществуват няколко *VXI* системни конфигурации, като всяка една от тях има свои предимства:

**Първа конфигурация** - с използване на вградени контролери. При тази конфигурация потребителски *VXI* компютър се вгражда директно в основния модул. По този начин може да се използва високата производителност на *VXI* системите, тъй като потребителския компютър комуникира директно с вътрешните връзки на *VXI*.

**Втора конфигурация** - с използване на *MXI* (*Multisystem eXtension Interface*). Тази конфигурация комбинира производителността на потребителски вграден компютър с гъвкавостта на настолните компютри. При тази конфигурация може да бъде използвана високоскоростна *MXIbus* връзка (23 MB/s) за свързване на външен компютър директно с вътрешните връзки на *VXI*.

**Третата конфигурация** - с използване на *IEEE 1394* (*FireWare*) или сериен интерфейс за управление на *VXI* система. Предимство на тази конфигурация е ниската цена на интерфейса. Чрез използване на *IEEE*

*1394* се постига по-висока производителност от *GPIB-VXI*, но по-ниска от *MXI*.

**Четвърта конфигурация** - състои се от един или няколко *VXI* основни модула, свързани с външен компютър посредством *GPIB*. Тази конфигурация може да се използва за постепенно интегриране на *VXI* към съществуващи *GPIB* системи. *VXI* инструментите могат да бъдат програмирани чрез съществуващия *GPIB* софтуер.

### 5.2.4. Използване на PXI (PCI eXtensions for Instrumentation) модулни инструменти

Това е нов модел модулна инструментална система, която използва интерфейса *PCI eXtensions for Instrumentation* (*PXI*). Тя се изгражда на основата на *PC* интерфейса *PCI* и осигурява висока производителност (фиг.3.43).

*Фиг.3.43. PXI инструменти*

*PXI* е напълно съвместим със стандарта *CompactPCI* и обединява предимствата на модерните тактуващи и превключващи технологии, използвани в *VXI*. *PXI* запълва пазарната ниша между евтините PC-базирани решения и мощните *VXI* и *GPIB* решения като комбинира стандартите на *Windows*, *PCI*, *CompactPCI* и *VXI*.

Потребителят има възможност да проектира *PXI* система по собствено желание като избира всичко необходимо за реализация на дадена задача: контролер, кутия, цифрово-аналогови модули, аналогово-цифрови модули, цифрови вход-изходи и др.

### 5.2.5. Използване на компютърно базирани инструменти

Компютърно базираните инструменти са разработени за няколко различни платформи (фиг.3.44), включително *PCMCIA* (преносими компютри), *PCI* (настолни компютри) и *PXI*.

Компютърно базираните инструменти са пример за виртуални инструменти, които се състоят от *PC*-базиран инструментален модул,

компютър и софтуер.


*Фиг.3.44. Компютърно базирани инструменти*

Предимство на компютърно базираните инструменти е, че използват процесорната мощ, паметта и монитора на компютъра, както и възможността за връзка към *Internet*.

## 5.3. ПРИМЕРИ ЗА ИЗМЕРВАНЕ НА СИГНАЛИ С DAQ УСТРОЙСТВА

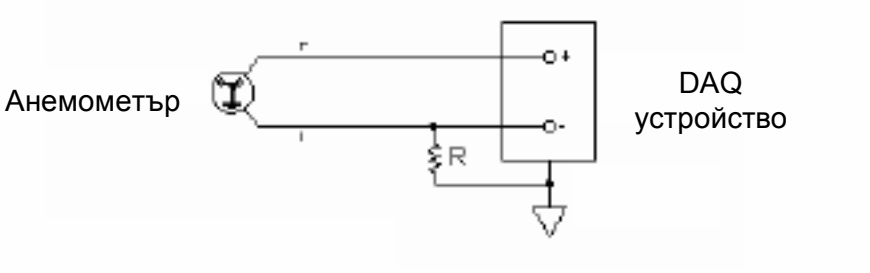
В този модул ще бъдат дадени примери как да бъдат извършени някои измервания с помощта на *LabVIEW*. Примерите показват *DAQ* система, сигналите, които се измерват, схеми на физическата връзка с първичните преобразуватели и диаграми на *LabVIEW*, показващи начина за измерване.

### 5.3.1. Измерване скоростта на вятър в една точка

На фиг.3.45 е показана опростена *DAQ* система, която използва анемометър за измерване скоростта на вятъра.


*Фиг. 3.45. DAQ система за измерване скоростта на вятъра*

На Фиг.3.46 е показана типична диаграма за свързване на анемометър с изходен обхват от 0 до 10 V, което отговаря на скорост на вятъра от 0 до 200 км/ч. Това означава, че в програмата за LabVIEW е необходимо мащабиране на данните.



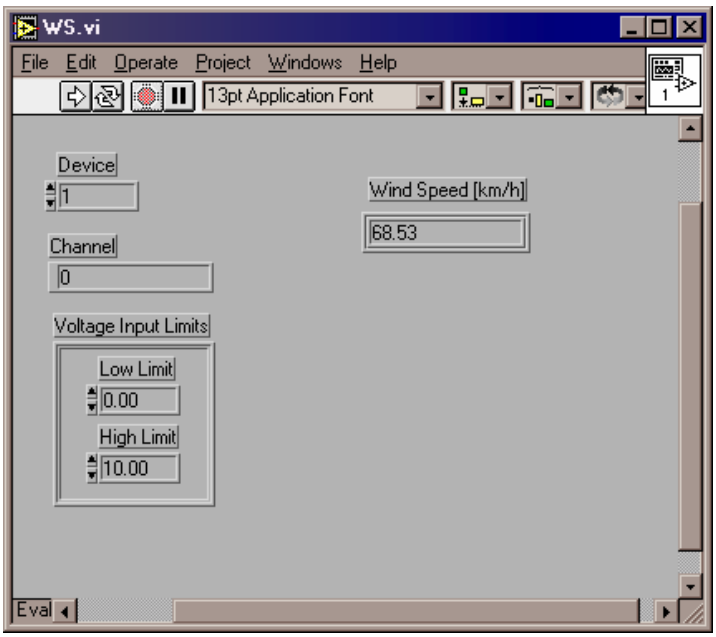

*Фиг. 3.46. Схема на свързване на анемометър към DAQ устройство*

#### 5.3.1.1. Симулиране и експериментално първоначално установяване

- Инсталирайте *DAQ* устройство и неговия драйвер.
- Свържете първичния преобразувател за измерване на скоростта на вятъра към *DAQ* устройството.
- Стартирайте *LabVIEW*. Отворете нов *VI* , като от меню *File* изберете командата *New*.

#### 5.3.1.2. Създаване на предния панел

- Добавете цифров вход като от поле *Numeric* на панел *Controls* изберете *Digital Control.* Задайте му име *Device* (*Устройство*). От този вход потребителя ще може да задава номера, присвоен на *DAQ* устройството по време на конфигурирането.
- Щракнете с десния бутон върху цифровия вход *Device* и от падащото меню изберете поле *Representation* и след това *I16*.
- Добавете стрингов вход като от поле *String & Table* на панел *Controls* изберете *String Control.* Задайте му име *Channel* (*Канал*). От този вход потребителя може да задава входния аналогов канал на устройството, към който е свързан анемометъра.
- Добавете клъстер като от поле *Array & Cluster* на панел *Controls* изберете *Cluster.* Задайте му име *Voltage Input Limits* (*Обхват на входното напрежение*).
- Към клъстера добавете цифров вход като от поле *Numeric* на панел *Controls* изберете *Digital Control* и го позиционирате в полето *Voltage Input Limits.* Задайте му име *High Limit* (*Горна граница*).

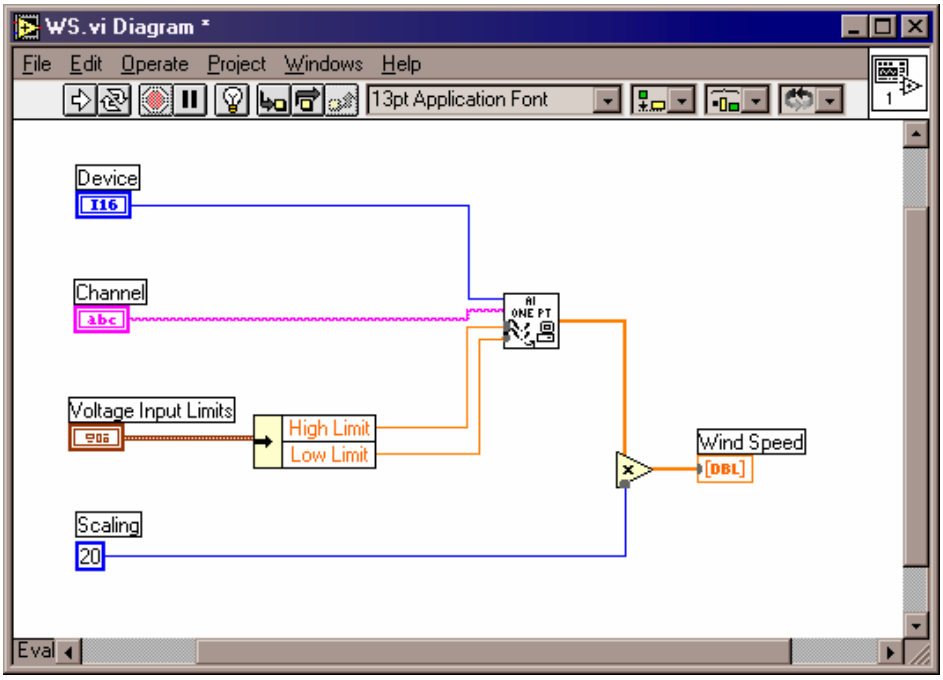



*Фиг. 3.47. Преден панел на инструмента за измерване скорост на вятъра*

- Към клъстера добавете още един цифров вход като от поле *Numeric* на панел *Controls* изберете *Digital Control* и го позиционирате в полето *Voltage Input Limits.* Задайте му име *Low Limit* (*Долна граница*).
- Добавете цифров масив като от поле *Array & Cluster* на панел *Controls* изберете *Array.* Задайте му име *Wind Speed* (*Скорост на вятъра*). Към масива добавете цифров индикатор като от поле *Numeric* на панел *Controls* изберете *Digital Indicator* и го позиционирате в полето *Wind Speed.* Позиционирайте елементите, както е показано на фиг.3.47.

### 5.3.1.3. Промяна на блоковата диаграма

- От поле *Cluster* на панел *Functions* изберете *Unbundle By Name.* Добавете го към диаграмата и го разширете за да се покажат двата елемента *Low Limit* и *High Limit*.
- Към блоковата диаграма добавете цифрова константа като за целта от поле *Numeric* на панел *Functions* изберете *Numeric Constant*. Въведете стойност 20.
- Към блоковата диаграма добавете модул за умножение като за целта от поле *Numeric* на панел *Functions* изберете *Multiply*.
- Добавете модул за измерване на сигнала от *DAQ* устройство. За целта от поле *Data Acquisition* на панел *Functions* изберете *Analog Input* и след това виртуалния инструмент *AI Sample Channels*.




*Фиг. 3.48. Блокова диаграма на инструмента за измерване скоростта на вятъра*

- Свържете елементите от диаграмата, както е показано на фиг.3.48.

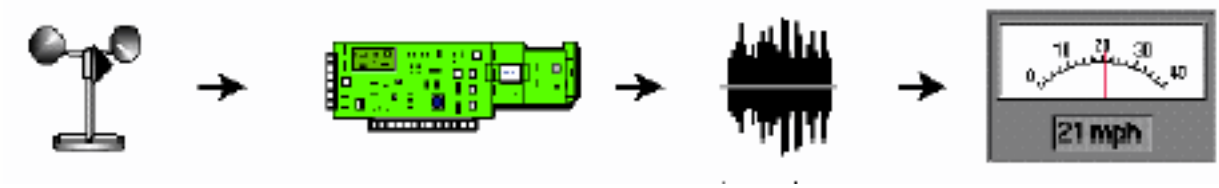
#### 5.3.1.4. Изпълнение на *VI*

- Активирайте предния панел. Задайте в поле *Device* номера на устройството (1). В поле *Channel* задайте номера на канала (0), в поле *Low Limit* задайте 0, а в поле *High Limit* задайте 10.
- Щракнете върху бутон *Run*.
- В поле *Wind Speed* ще бъде изведена измерената стойност на вятъра (фиг.3.47).
- Затворете *VI* като изберете *Close* от меню *File*. Когато диалоговият прозорец запита, дали да запази промените щракнете върху бутон *Yes*.

### 5.3.2. Измерване скоростта на вятъра в няколко точки и усредняване на резултатите

На фиг.3.49 е показана опростена *DAQ* система, която използва анемометър за измерване скоростта на вятъра, DAQ устройство за аналогово-цифрово преобразуване и програма на LabVIEW за обработване и визуализиране на резултатите.

Анемометър АЦП (DAQ устройство) Обработване на резултатите (LabVIEW)



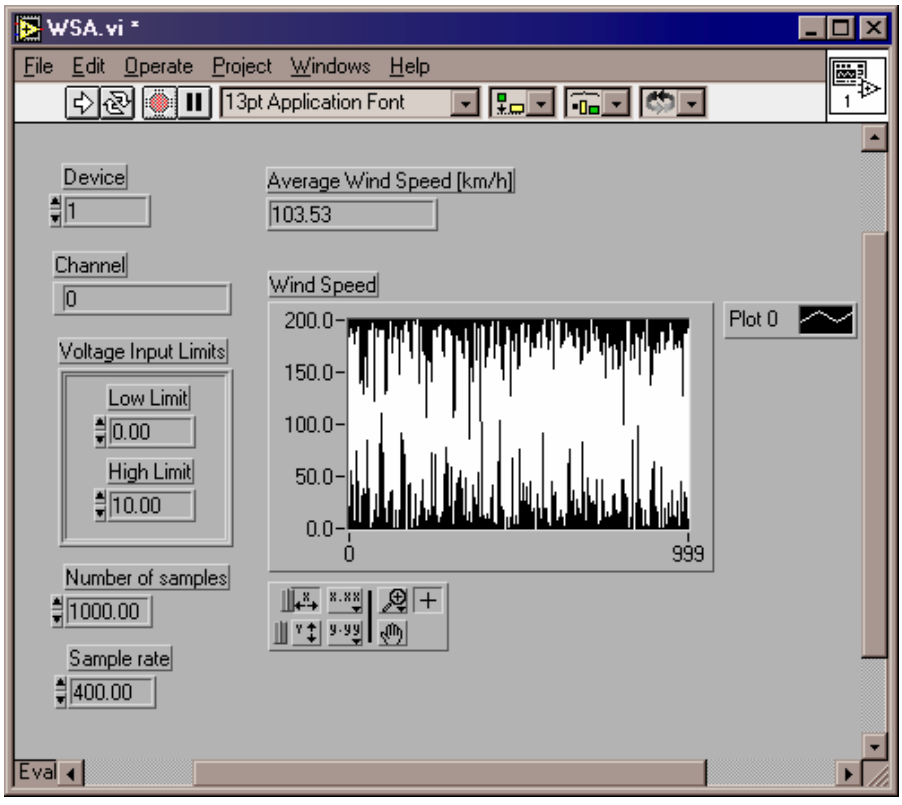
*Фиг. 3.49. DAQ система за измерване ср. скорост на вятъра*

### 5.3.2.1. Симулиране и експериментално първоначално установяване

- Стартирайте *LabVIEW*. Отворете виртуалния инструмент, който беше създаден в предишното упражнение, като от меню *File* изберете командата *Open*.

### 5.3.2.2. Промяна на предния панел

- Премахнете цифровия масив *Wind Speed* като щракнете с левия клавиш на мишката върху него и след това натиснете клавиш *Delete* от клавиатурата.
- Добавете цифров вход като от поле *Numeric* на панел *Controls* изберете *Digital Control.* Задайте му име *Number of samples* (*Брой измервания*).
- Добавете цифров вход като от поле *Numeric* на панел *Controls* изберете *Digital Control.* Задайте му име *Sample rate* (*Честота на дискретизация*).
- Добавете цифров изход като от поле *Numeric* на панел *Controls* изберете *Digital Indicator.* Задайте му име *Average Wind Speed* (*Средна скорост на вятъра*).
- За извеждане на графичната информация изберете *Waveform Chart* от поле *Graph* на панел *Controls*. Поставете етикет *Wind Speed*. Позиционирайте управляващите елементи, както е показано на фиг.3.50.

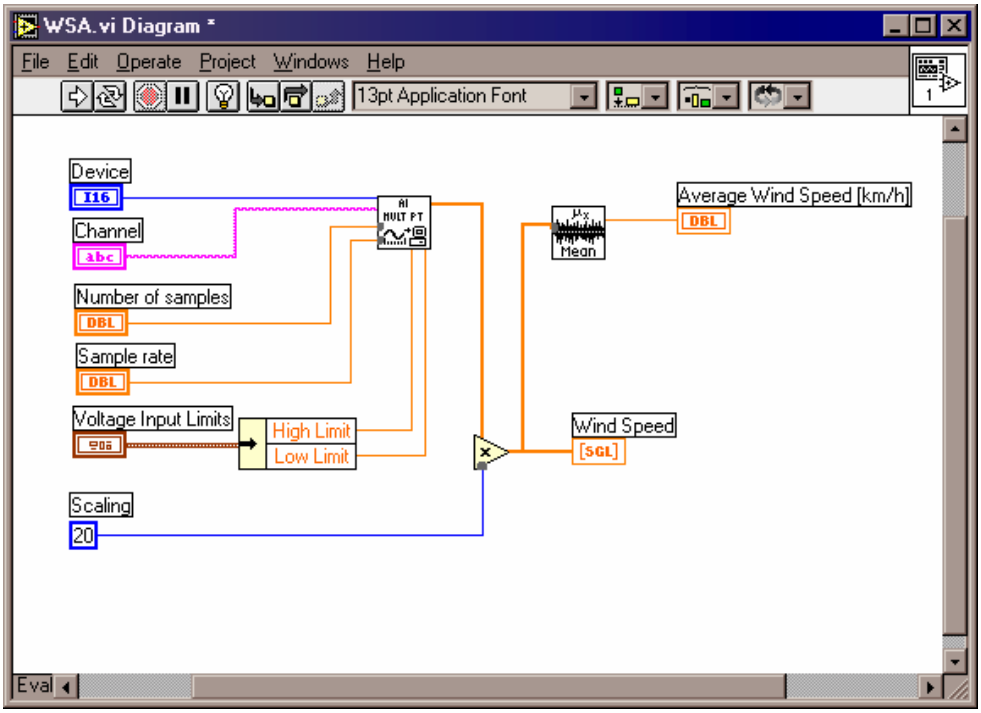



*Фиг. 3.50. Преден панел на инструмента*

*за измерване средната скорост на вятъра*

### 5.3.2.3. Промяна на блоковата диаграма

- Към блоковата диаграма добавете модул за усредняване на измерванията като за целта от поле *Analysis* на панел *Functions* изберете *Probability and Statistics* и след това виртуалния инструмент *Mean*.
- Към блоковата диаграма добавете модул за умножение като за целта от поле *Numeric* на панел *Functions* изберете *Multiply*.
- Заместете модула за измерване на единична стойност на сигнала от *DAQ* устройство, използван в предишното упражнение с модул за измерване на поредица от стойности. От поле *Data Acquisition* на панел *Functions* изберете *Analog Input* и след това виртуалния инструмент *AI Acquire Waveform*.




*Фиг. 3.51. Блокова диаграма на инструмента за измерване средната скорост на вятъра*

- Свържете елементите, както е показано на фиг.3.51.

### 5.3.2.4. Изпълнение на *VI*

- Активирайте предния панел. Задайте в поле *Device* номера на устройството (1). В поле *Channel* задайте номера на канала (0), в поле *Low Limit* задайте 0, а в поле *High Limit* задайте 10, в поле *Number of samples* задайте 1000, а в поле *Sample rate* - 400.
- Щракнете върху бутон *Run*.
- В поле *Average Wind Speed* ще бъде изведена средната стойност от измерванията, а в поле *Wind Speed* ще бъдат показани всички измерени стойности в графичен вид (фиг.3.50).

# ПРИЛОЖЕНИЕ С ОПИСАНИЕ НА ОСНОВНИТЕ И СТАНДАРТНИ ФУНКЦИИ В *MATLAB*

## 1. КОМАНДИ С ОБЩО ПРЕДНАЗНАЧЕНИЕ

### 1.1. Управление на команди и функции.

***cedit*** – Променя параметрите за редактиране в командния ред
***demo*** - Стартира демонстрационни програми.
***help*** - Извежда помощна информация.
***info*** - Извежда информация за *MATLAB.*
***lookfor*** - Търси помощна информация по ключова дума.
***path*** - Управлява пътеките за търсене в *MATLAB.*
***type*** - Извежда текста на *M*-файл.
***version*** – Извежда версията на MATLAB
***what*** - Извежда списък на *M*-, *MAT*- и *MEX*- файлове.
***which*** - Извежда разположение на специфициран файл.
***whatsnew*** – Показва *README* файловете на всеки "*toolbox*"
***why*** – Извежда остроумни отговори

### 1.2. Управление на променливи и на работното пространство.

***clear*** - Извежда помощна информация.
***disp*** - Стартира демонстрационни програми.
***length*** - Определя дължината на вектор.
***load*** - Възстановява променливи от диска.
***pack*** - Уплътнява паметта на работното пространство.
***save*** - Съхранява на диск променливи от работното пространство.
***size*** - Определя размерност на матрица.
***who*** - Извежда списък на променливите в паметта.
***whos*** - Както *who* плюс размера на извежданите променливи.

### 1.3. Работа с файлове и с операционната система.

***cd*** - Смяна на директория.
***delete*** - Изтрива файл.
***diary*** - Създава дневник на работа на *MATLAB* (без графиките).
***dir*** - Извежда файлове от директория на диска.
***getenv*** - Осигурява достъп до променливите на работната среда.
***pwd*** – Показва текущата работна директория
***unix*** – Изпълнява команда на *unix*
***!*** - Изпълнява команда на операционната система.

### 1.4. Управление на командния прозорец.

***clc*** - Изтрива съдържанието на командния прозорец.
***echo*** - Осъществява ехо от командите в описателните файлове.
***format*** - Установява изходен формат на типа на данните.
***home*** - Позиционира курсора в начална позиция.
***more*** - Извежда данните по страници.

### 1.5. Активиране и дезактивиране на *MATLAB.*

***matlabrc*** - Активира главния стартиращ *M*- файл на *MATLAB.*
***exit*** - Осъществява изход от *MATLAB* в ОС
***quit*** - Както *exit*.
***startup*** - *M*-файл който се изпълнява при стартиране на *MATLAB*.

## 2. ОПЕРАТОРИ И СПЕЦИАЛНИ СИМВОЛИ.

### 2.1. Оператори и специални символи

***kron*** – Кронекерово произведение

**Действия с матрици и вектори**

**+** - Събиране.
**-** - Изваждане.
***** - Умножение.
**/** - Дясно деление.
**** - Ляво деление.
**^** - Степенуване.
**'** - Транспортиране.

**Специални символи.**

**=** - Оператор за присвояване.
**[** - Форматиране на матрици и вектори.
**]** - Виж [.
**(** - Форматиране на аритметични изрази.
**)** - Виж (.
**.** - Десетична точка.
**..** – По-горната директория в дървото на директориите
**...** - Продължение на оператор на следващ ред.
**,** - Разделител на индекси и на функционални елементи.
**;** - Ограничител на ред на матрица или символ, който забранява извеждането на резултата на екрана.
**%** - Означение за коментар, който не се обработва от интерпретатора.
**:** - Означение за индексите при формиране на подматрица. Преобразува матрица във вектор - стълб
**!** – Префикс към команда на операционната система

**Логически сравнения.**

**<** - По-малко.
**<=** - По-малко или равно.
**>** - По-голямо.
**>=** - По-голямо или равно.
**==** - Равно.
**~=** - Неравно.
**&** - Логическо *И*.
**|** - Логическо *ИЛИ*.
**~** - Логическо *НЕ*.
***xor*** - Логическо *ИЗКЛЮЧВАЩО ИЛИ*.

### 2.2. Логически функции.

***all*** - Истина, ако всички елементи на вектор са истина.
***any*** - Истина, ако някой елемент на вектор е истина.
***exist*** - Проверка за съществуване на променливи или функции.
***find*** - Намира индексите на ненулевите елементи.
***finite*** - Истина за елементи с крайни стойности.
***isempty*** - Истина за празна матрица.
***isieee*** - Истина за *IEEE* аритметика с плаваща точка.

***isinf*** - Истина за елементи с безкрайни стойности.
***isnan*** - Истина за нечислови елементи.
***issparse*** - Истина за разредена матрица.
***isstr*** - Истина за символен текст.
***strcmp*** – Сравнява текстови низове

## 3. ЕЗИКОВИ КОНСТРУКЦИИ В *MATLAB*.

### 3.1. *MATLAB* като програмен език.

***eval*** - Изпълнява стринг с *MATLAB* израз.
***feval*** - Изпълнява функция, зададена чрез стринг.
***function*** - Добавя нова функция.
***global*** - Дефинира глобални променливи.
***narghk*** - Проверява броя на входните аргументи.

### 3.2. Управление на програмите.

***break*** - Прекъсва цикличното изпълнение на оператори.
***else*** - Използва се с *if*.
***elseif*** - Използва се с *if*.
***end*** - Край на действието на *if*, *for* и *while*
***error*** - Извежда съобщение и прекратява действието на функция.
***for*** - Циклично изпълнява оператори.
***if*** - Условно изпълнява оператори.
***return*** - Връща управлението към предходната функция.
***while*** - Повтаря изпълнението на оператори при дадени условия.

### 3.3. Диалогов вход.

***input*** - Позиционира курсора за вход от клавиатурата.
***keyboard*** - Интерпретира данните от клавиатурата като *M*-файл.
***menu*** - Формира меню за избор на възможните входове.
***pause*** - Реализира пауза, която се прекъсва от клавиатурата.

### 3.4. Настройка на програми (дебъгване)

***dbstop*** – Поставя точка на прекъсване
***dbclear*** – Премахва точка на прекъсване
***dbcont*** – Премахва изпълнението на оператор
***dbdown*** – Променя контекста на локалната работна памет
***dbstack*** – Показва коя функция какво извиква
***dbstatus*** – Показва всички точки на прекъсване
***dbstep*** – Изпълнява един или няколко оператора
***dbtype*** – Извежда m-файла с номера на редовете
***dbup*** – Променя контекста на локалната работна памет
***dbdown*** – Обратно на *dbup*
***dbquit*** – Преустановява режима на настройване

## 4. ЕЛЕМЕНТАРНИ МАТРИЦИ И МАТРИЧНИ ДЕЙСТВИЯ.

### 4.1. Елементарни матрици.

***eye*** - Единична матрица.
***meshgrid*** - *X*-*Y* масиви за *3-D* графика.
***ones*** - Матрица от единици.
***rand*** - Матрица от равномерно разпределени случайни числа.
***randn*** - Матрица от нормално разпределени случайни числа.

***zeros*** - Матрица от нули.

**4.2. Специални променливи и константи.**

***ans*** - Съдържа резултата, когато той не е присвоен на променлива.
***computer*** - Тип на компютъра.
***eps*** - Точност при действия с плаваща точка.
***flops*** - Брояч на операциите с плаваща точка.
***i, j*** - Имагинерни единици.
***inf*** - Стойност *" безкрайност"*.
***NaN*** - Означение, че променливата не е число.
***nargin*** - Брояч на входните аргументи на функцията.
***nargout*** - Брояч на изходните аргументи на функцията.
***pi*** - Числото *" пи "=3.1415926535897*.
***realmax*** - Най-голямото число с плаваща точка.
***realmin*** - Най-малкото число с плаваща точка.

**4.3. Време.**

***clock*** - Изобразява часовник.
***cputime*** - Определя процесорното време.
***date*** -Изобразява календарна дата.
***etime*** - Отчита интервал от време.
***tic, toc*** - Маркери за отчитане на времето.

**4.4. Действия върху матрици.**

***diad*** - Създава или извежда диагонал на матрица.
***fliplr*** - Извършва въртене на матрицата в ляво/дясно.
***flipud*** - Извършва въртене на матрицата нагоре/надолу.
***reshape*** - Променя размера на матрицата.
***rot90*** - Извършва въртене на матрицата на 90 градуса.
***tril*** - Извежда долна триъгълна част от матрица.
***triu*** - Извежда горна триъгълна част от матрица.

**5.СПЕЦИАЛНИ МАТРИЦИ.**

***compan*** – Матрица в съпровождаща форма.
***diag*** – Диагонална матрица.
***gallery*** – Две малки тестови матрици.
***hadamard*** – Матрица на Адамар.
***hankel*** – Матрица на Ханкел.
***hilb*** – Матрица на Хилберт.
***nvhilb*** – Обратна матрица на Хилберт.
***linspace*** – Равномерно разделено пространство.
***logspace*** – Логаритмично разделено пространство.
***magic*** – Магически квадрат.
***pascal*** – Матрица на Паскал.
***rosser*** – Симетрична матрица за тест на собствени стойности.
***toeplitz*** – Матрица на Тьоплиц.
***wilkinson*** – Тестова матрица на Уилкинсън за собствени стойности.

**6. ЕЛЕМЕНТАРНИ ФУНКЦИИ.**

**6.1. Елементарни математически функции.**

***abs*** - Абсолютна стойност.
***angle*** - Фазов ъгъл.

***ceil*** - Закръгляне до най-близкото по-голямо цяло число.
***conj*** - Комплексна спрегната стойност.
***exp*** - Експоненциална стойност.
***fix*** - Закръгляне до цялата част на числото.
***floor*** - Закръгляне до най-близкото по-малко цяло число.
***ged*** – Най-голям общ делител
***imag*** - Имагинерна част на комплексно число.
***lcm*** – Най-малко общократно
***log*** - Натурален логаритъм.
***log10*** - Десетичен логаритъм.
***real*** - Реална част на комплексно число.
***rem*** - Остатък след деление.
***round*** - Закръгляване до най-близкото цяло число.
***sign*** - Сигнум-функция.
***sqrt*** - Квадратен корен.

### 6.2. Стандартни тригонометрични функции.

***csc, acsc, csch, acsch–*** Косекант, аркускосекант, хиперболичен косекант, хиперболичен аркускосекант.

***cos, acos, cosh, acosh –*** Косинус, аркускосинус, хиперболичен косинус, хиперболичен аркускосинус.

***cot, acot, coth, acoth –*** Котангенс, аркускотангенс, хиперболичен котангенс, хиперболичен аркускотангенс.

***sec, asec, sech, asech –*** Секант, аркуссекант, хиперболичен секант, хиперболичен аркуссекант.

***sin, asin, sinh, asinh –*** Синус, аркуссинус, хиперболичен синус, хиперболичен аркуссинус.

***tan, atan, tanh, atanh –*** Тангенс, аркустангенс, хиперболичен тангенс, хиперболичен аркустангенс.

## 7. СПЕЦИАЛИЗИРАНИ МАТЕМАТИЧЕСКИ ФУНКЦИИ.

***bessel*** – Беселова функция.
***beta*** – Бета функция.
***gamma*** – Гама функция.
***erf*** – Функция на грешката.
***erfinv*** – Обратна функция на грешката.
***ellipke*** – Пълен елиптичен интеграл.
***ellipj*** – Елиптична функция на Якоби.
***expint*** – Експоненциален интеграл.
***log2 –*** Логаритъм при основа 2.
***pow2 –*** Повдигане 2 на степен.
***rat, rats –*** Рационална апроксимация.

## 8. МАТРИЧНИ ФУНКЦИИ ОТ ЛИНЕЙНАТА АЛГЕБРА.

### 8.1. Елементарни матрични функции.

***cond*** - Число на обусловеност на матрица.
***det*** - Детерминанта на матрица.
***eig*** - Собствени стойности и вектори на матрица.
**expm** - Матрична експонента.
***expm 1*** – Реализация на expm чрез *m*-файл.
***expm 2*** – Матрична експонента чрез ред на Тейлор.

***expm 3*** – Матрична експонента чрез собствени стойности и вектори.
***funm*** – Изчисляване на обобщени матрични функции.
***inv*** - Обратна матрица.
***logm*** - Матричен логаритъм.
***norm*** - Норма на матрица.
***poly*** - Характеристичен полином на матрица.
***rank*** - Ранг на матрица.
***rcond*** - Обратно ( реципрочно ) число на обусловеност.
***sqrtm*** - Матричен квадратен корен.
***trace*** - Следа на матрица.

**8.2.Специализирани матрични функции.**

***balance*** – Балансиране на матрица.
***cdf2rdf*** – Преобразуване от комплексна диогонална форма към реална блочно-диагонална форма.
***chol*** – Факторизация по Холецки.
***condest*** – Оценяване на число на обусловеност по Hager/Higham.
***hess*** – Форма на Хесенберг.
***lscov*** – Най – малки квадрати при известни ковариации.
***lu*** – Фактори на Гаусовата елминация (*LU* разлагане).
***nnls*** – Неотрицателни най-малки квадрати.
***normest*** – z – норма на матрица.
***null*** – Празно пространство.
***orth*** – Ортогонализация.
***pinv*** – Псевдообратна матрица.
***polyeig*** – Определяне собствени стойности чрез полином.
***qr*** – QR декомпозиция.
***qz*** – Обобщени собствени стойности.
***rsf2csf*** – Обратно на *cdf2rdf*.
***schur*** – Декомпозиция по Шур.
***svd*** – Декомпозиция по сингулярни числа.

**9. АНАЛИЗ НА ДАННИ.**

**9.1. Базови операции.**

***cumprod*** - Натрупващо произведение от елементи.
***cumsum*** - Натрупваща сума от елементи.
***hist*** – Хистограма.
***max*** – Най–голямата компонента.
***mean*** – Средна стойност.
***median*** – Медианна стойност.
***min*** – Най – малката компонента.
***prod*** – Произведение от елементи.
***quad*** – Адаптивен метод на Симпсон.
***quad8*** – Адаптивен метод на Нютон-Коутс.
***sort*** – Сортиране на елементи.
***std*** – Стандартно отклонение.
***sum*** – Сума от елементи.
***trapz*** – Числено интегриране чрез метод на трапеците.

**9.2. Корелация.**

***corrcoef*** - Коефициенти на корелация.

***cov*** - Ковариационна корелация.

**9.3. Филтрация и конволюция.**

***conv*** - Конволюция и умножение на полиноми.
***conv2*** - Двумерна конволюция.
***deconv*** - Деконволюция и деление на полиноми.
***filter*** - Едномерен цифров филтър.
***filter2*** - Двумерен цифров филтър.

**9.4. Преобразуване на Фурие (ПФ).**

***abs*** - Амплитуда.
***angle*** - Фазов ъгъл.
***cplxpair*** - Сортира членовете по комплексно-спрегнати двойки.
***fft*** - Дискретно ПФ.
***fft2*** - Двумерно дискретно ПФ.
***ffrshift*** - Премества нулевото отместване в центъра на спектъра.
***ifft*** - Обратно дискретно ПФ.
***ifft2*** - Двумерно дискретно ПФ.
***nextpow2*** - Най-близката степен на 2.
***unwrap*** - Отстранява граничните скокове във фазовия ъгъл.

**10. ПОЛИНОМИАЛНИ И ИНТЕРПОЛАЦИОННИ ФУНКЦИИ.**

**10.1. Полиноми.**

***conv*** - Умножава полиноми.
***deconv*** - Дели полиноми.
***poly*** - Конструира полином с определени корени.
***polyder*** - Диференцира полином.
***polyfit*** - Съпоставя полином на данни.
***polyval*** - Изчислява полином за конкретен аргумент.
***polyvalm*** - Изчислява полином за конкретен матричен аргумент.
***residue*** -Остатъци от деление на два полинома.
***roots*** - Отделяне корените на полином.
***roots1*** – Корени на полином по метода на Лагер.

**10.2. Интерполация на данни.**

***del2*** – Петточкова дискретизация *2D*-Лапласиана.
***diff*** – Разлики и приблизителни производни.
***gradient*** – Приблизителен градиент.
***griddata*** – Интерполация на данни за тримерна графика.
***interp1*** - Едномерна интерполация на данни.
***interp2*** - Двумерна интерполация на данни.
***interpft*** - Едномерна интерполация на данни чрез бързо ПФ.
***spline*** – Интерполация с кубични сплайни.
***subspace*** – Ъгъл между две подпространства.

**11. НЕЛИНЕЙНИ ЧИСЛЕНИ МЕТОДИ.**

**11.1. Диференциалнни уравнения.**

***ode23*** – Метод на Рунге–Кута от 2-3 ред.
***ode23p*** – Както *ode23*, но с изчертаване на резултата.
***ode45*** – Метод на Рунге–Кута от 4-5 ред.

**11.2.Нелинейни уравнения и оптимизация.**

***fmin*** – Минимум на функция на една променлива.
***fplot*** – Чертаене графика на функция.
***fmins*** - Минимум на функция на повече променливи.
***fsolve*** – Решение на системата от нелинейни уравнения.
***fzero*** – Намиране на нула на функция на една променлива.

**12. РАЗРЕДЕНИ МАТРИЦИ.**

***colmmd –*** Преподреждане по ”алгоритъма с минимална степен” за намаляване броя на новите ненулеви елементи.
***colperm*** – Преподрежда стълбовете според броя на ненулевите елементи.
***condest*** – Оценява числото на обусловеност.
***dmperm*** – Преподреждане по алгоритъм на Dulinaje-Mendelsohn.
***find*** – Определя индексите на ненулевите елементи.
***full*** – Преобразува матрица от разреден в пълен модел.
***gplot*** – Изобразява графа свързан с матрицата.
***issparse*** – “Истина”, ако матрицата е разредена.
***nnz*** – Брой на ненулевите елементи.
***nonzeros*** – Ненулеви елементи.
***normest*** - Оценява z-норма.
***nzmax*** – Обем памет за разполагане на ненулевите елементи.
***randperm*** – Случаен пермутационен вектор.
***spalloc*** – Заделя памет за разполагането на ненулевите елементи.
***sparse*** – Преобразува матрица от пълен в разреден модел.
***sparsefun*** – Директория на функциите за разредени матрици.
***spaugment*** – Формира присъединена матрица за задчата на най-малките.
***spconvert*** - Преобразува разредена матрица, зададена във външен формат.
***spdiags*** – Формира разредена матрица от диогонали.
***speye*** – Разредена единична матрица.
***spfun*** –Прилага функция към ненулевите елементи.
***spones*** – Заменя ненулевите елементи с единици.
***sprandn*** – Разредена матрица с елементи случайни числа.
***sprandsym*** – Разредена симетрична матрица с елементи случайни числа.
***sprank*** - Определя структурен ранг на разредена матрица квадрати.
***spparms*** – Установява параметри за програмите за разредени матрици.
***spy*** – Показва разредената структура.
***symbfact*** – Символен анализ на факторизация по Холески и LU факторизация.
***symmmd*** – Преподреждане по ”алгоритъма с минимална степен” за симетрични матрици , за намаляване броя на новите ненулеви елементи.
***symrcm*** – Обратно подреждане на Cuthill-McKee.

**13. 2-D ГРАФИКИ.**

**13.1. Елементарни X-Y графики.**

***fill*** - Чертае запълнена с цвят двумерна фигура.
***loglog*** - Графика с логаритмични координатни оси.
***plot*** - Линейна графика.
***polar*** – Графика в полярни координати.
***semilogx*** - Графика с логаритмични ос *x* и линейна ос *y*.
***semilogy*** - Графика с логаритмични ос *y* и линейна ос *x*.

**13.2. Специализирани X-Y графики.**

***bar*** - Бар-графика.
***compass*** - Компас-графика.
***errorbar*** - Бар-графика на грешката.
***feather*** - Перо-графика.
***fplot*** - Графика на функция.
***hist*** - Хистограма.
***rose*** - Графика в полярни координати.
***stairs*** - Стъпална графика.

### 13.3. Означения на графиките.

***giput*** – Връща координати на графичен курсор от прозореца.
***grid*** - Координатна мрежа.
***gtext*** - Позиционира мишката в текст.
***text*** - Текст.
***title*** - Заглавие на графика.
***xlable*** - Етикет по оста *x*.
***ylable*** - Етикет по оста *y*.
***zlabel*** – Надпис на z по оста.

## 14. 3-D ГРАФИКИ.

### 14.1. Извеждане на линии и запълване на области.

***image*** – Показва образ, създаден като обект.
***pcolor***– Изобразява цветна таблица по зададена матрица с цветове.
***quiver***– Изобразява векторно потенциално поле.
***slice*** – Изобразява специфични сечения на тримерни фигури.
***waterfall*** – Подобна на *mesh*.

### 14.2. Контури и други 2-D изображения в 3-D графиката.

***clabel*** – Поставя надписи към контурните графики.
***contour*** – Изобразява контурите на линиите с еднакво ниво.
***contourc*** – Изчислява матрица за функцията contour.
***contour3***– Изобразява контурите на линиите с еднакво ниво в тримерно пространство.
***fill3*** - Изобразява запълнен тримерен многостен.
***plot3*** – Изобразява тримерна крива в пространството.

### 14.3. Изображение на обвивка (surface) и мрежа (mesh).

***mesh*** – Изобразява повърхнина във вид на мрежа.
***meshc*** – Изобразява повърхнина в комбиниран вид мрежа – контурни линии.
***meshz*** – Изобразява повърхнина във вид на мрежа с равнина на нулево ниво.
***surf*** - Изобразява повърхнина във вид на плътна повърхност.
***surfc*** - Изобразява повърхнина в комбиниран вид плътна повърхност – контурни линии.
***surf1*** - Изобразява повърхнина със сенки и осветление.

### 14.4. Представяне на графиките.

***axis*** – Дефинира граници на скалите
***brighten*** – Засилва или отслабва цветната гама
***caxis*** – Дефинира скала на псевдоцветове на графични обекти
***colormap*** – Матрица, съдържаща цветна гама
***diffuse*** – Връща “отражението” от изобразявана повърхнина

***hidden*** – Включва/изключва режима за изобразяване на невидимите части
***hsv2rgb*** – Преобразува “*hsb*“ в “*rgb*” гама
***rgb2hsv*** – Преобразува “*rgb*” в “*hsb* “ гама
***rgbplot*** – Изобразява цветната гама
***shading*** – Задава режима на оцветяване на повърхнини
***specular*** – Връща “отражението” от изобразявана повърхнина
***spinmap*** – Динамично изменение на цветната гама
***surfnorm*** – Изчислява нормалите на повърхнина
***view*** – Задава гледна точка за наблюдаване на тримерен обект
***viewmtx*** - Пресмята трансформационни матрици в зависимост от гледната точка.

**14.5. Означения в графиките.**

***cylinder*** – Генерира цилиндър.
***peaks*** – Генерира демонстрационна повърхнина.
***sphere*** – Генерира сфера.

## 15. ГРАФИЧНИ ФУНКЦИИ С ОБЩО ПРЕДНАЗНАЧЕНИЕ.

**15.1. Създаване и управление на графичен прозорец.**

***clf*** - Изчиства съдържанието на текуща графика.
***close*** - Затваря (премахва) графика.
***figure*** - Отваря (създава) графика (графичен прозорец).
***gcf*** - Задава управление към текуща графика.
***whitebg*** – Инвертира фона на граничните прозорци.

**15.2. Създаване и управление на координатни оси.**

***axes*** - Създава координатна система на произволна позиция.
***axis*** - Управлява мащаба на координатните оси.
***caxis*** - Управлява псевдо цветовете на координатните оси.
***cla*** - Изтрива текущите координатни оси.
***gca*** - Задава управление към текущите оси.
***hold*** - Задържа на екран текущата графика.
***subplot*** - Създава координатни системи на един екран.
***whitebg*** – Инвертира фона на граничните прозорци.

**15.3. Управление на графични елементи.**

***line*** – Чертае линия.
***text*** – Изобразява текст.
***patch*** – Създава графичен елемент.
***surface*** – Изобразява повърхнина.
***uicontrol*** – Създава средства за управление на потребителския интерфейс.
***uimenu*** – Създава меню в потребителския интерфейс.

**15.4. Управление на графични операции.**

***delete*** – Изтрива обект.
***drawnow*** – Завършва дадено графично действие.
***get*** – Извлича свойствата на обектите.
***reset*** – Установява свойствата на обектите по подразбиране.
***set*** – Установява свойствата на обектите.

**15.5. Копиране и съхраняване.**

***orient*** – Установява ориентацията на хартията в печатащото устройство.
***print*** – Отпечатва или записва във файл графичен прозорец.
***printopt*** – m-файл, съдържащ конфигурацията на изходното печатащо устройство.

### 15.6. Анимация.

***getframe*** – Извлича кадър.
***movie*** – Изобразява записани кадри.
***moviein*** – Инициализира паметта за съхраняване на отделни кадри.

## 16. ФУНКЦИИ ЗА УПРАВЛЕНИЕ НА ЦВЕТОВЕТЕ.

***bone*** – Сива гама с оттенъци на синьо.
***cool*** – Преливащи оттенъци на циан и магента.
***copper*** – Линейна “медна “гама.
***flag*** – Редуващи се червено-бяло-синьо-черно.
***gray*** – Линейна сива гама.
***hot*** – Черно – червено – жълто – бяло.
***hsv*** – Преливаща периодична гама.
***jet*** – Вариант на *hsv*.
***pink*** – Пастелни оттенъци на розово.

## 17. ФУНКЦИИ ЗА ОБРАБОТВАНЕ НА ЗВУК.

***auread*** – Четене на *Sun*-аудиофайл.
***auwrite*** – Запис на *Sun*-аудиофайл.
***lin2mu*** – Конвертиране от линеен в *mu-law* аудиофайл.
***mu2lin*** – Конвертиране от *mu-law* в линеен аудиофайл.
***saxis*** – Промяна на звуковите скали.
***sound*** – Конвертиране на вектор в звук.

## 18. ФУНКЦИИ ЗА СИМВОЛНИ ДАННИ.

### 18.1. Общи функции.

***abs*** - Преобразува стринг в числени стойности.
***blanks*** – Създава текстов низ, съставен от празни интервали.
***dec2hex*** – Преобразува десетична цяла величина в шеснадесетично число.
***eval*** - Изпълнява стринг с *MATLAB* израз.
***hex2dec*** – Преобразува шеснадесетична величина в десетично цяло число.
***hex2num*** – Преобразува шеснадесетична величина в число с плаваща запетая.
***isstr*** - Истина за стринг.
***setstr*** - Преобразува числени стойности за стринг.
***str2mat*** - Формира текстова матрица от отделни стрингове.
***string*** - Управление на символна информация в *MATLAB.*

### 18.2. Сравнение на символни данни.

***blanks*** – Създава текстов низ, съставен от празни интервали.
***dec2hex*** – Преобразува десетична цяла величина в шеснадесетично число.
***hex2dec*** – Преобразува шеснадесетична величина в десетично цяло число.
***hex2num*** – Преобразува шеснадесетична величина в число с плаваща запетая.
***lower*** - Преобразува символни данни в долен регистър.
***strcmp*** - Сравнява символни данни.

***upper*** - Преобразува символни данни в горен регистър.

**18.3. Преобразуване на символни данни в числови.**

***blanks*** – Създава текстов низ, съставен от празни интервали.

***dec2hex*** – Преобразува десетична цяла величина в шеснадесетично число.

***hex2dec*** – Преобразува шеснадесетична величина в десетично цяло число.

***hex2num*** – Преобразува шеснадесетична величина в число с плаваща запетая.

***int2str*** - Преобразува целочислена променлива в стринг.

***num2str*** - Преобразува произволно число в стринг.

***sprintf*** - Преобразува с определен формат число в стринг.

***sscanf*** - Преобразува сринг в число с определен формат.

***str2num*** - Преобразува стринг в число.

**18.4. Преобразуване на данни от шестнайсетичен в десетичен формат.**

## 19. ВХОДНО-ИЗХОДНИ ДАННИ ЗА РАБОТА С ФАЙЛОВЕ.

**19.1. Отваряне и затваряне на файлове.**

***fclose*** - Затваря файл.

***fopen*** - Отваря файл.

**19.2. Безформатен вход-изход.**

***fread*** - Чете двоични данни от файл.

***fwrite*** - Записва двоични данни във файл.

**19.3. Форматиран вход-изход.**

***fgetl*** - Чете ред от файл, отхвърляйки символите за нов ред.

***fgets*** - Чете ред от файл, отчитайки символите за нов ред.

***fprintf*** - Записва форматирани данни във файл.

***fscanf*** - Чете форматирани данни от файл.

**19.4. Позициониране във файл.**

***ferror*** - Изяснява входно-изходните грешки във файл.

***frewind*** - Пренавиване на файл.

***fseek*** - Поставя позиционен индикатор.

***ftell*** - Отчита позиционния индикатор.

**19.5. Преобразуване на символна информация.**

***sprintf*** - Записва форматирани данни в стринг.

***sscanf*** - Чете форматирани символни данни.