ENGR 1181  |  MATLAB 11: Program Design

Preparation Material

Learning Objectives
1. Explain the significance of the planning stage in computer problem solving
2. Solve complex MATLAB problems involving all previously used concepts
3. Construct flow charts to develop code

Topics
There is no reading required from the MATLAB book before coming to class. This preparation material is provided to fully prepare students for in-class instruction.

Students will gain a solid understanding of the problem solving method, why it is important, and how to work through the process. This material contains the following:

1. Problem Solving Overview
2. Problem Solving Using Computers
3. Tools for Program Development
4. Top-Down Design Technique
5. Flow Charts

1. Problem Solving Overview
Engineers must analyze and solve a wide range of technical problems. They can:

- be reasonably simple single-solution problems,
- be open-ended that require a team of engineers from several disciplines,
- have no clear solution.

Sometimes there is only one possible answer, which can be solved based on having exactly the necessary amount of information. An example is calculating how fast a car is traveling by knowing the distance and time, in the proper units.
In the real world, we rarely have all of the necessary information and also don't have to filter through extraneous information. Engineers are often tasked with collecting information and then determining if there is no solution, one solution, or multiple possible solutions. It is rare to arrive at exactly one solution. Most often, there are multiple solutions and a “best” solution must be chosen.

**Problem Solving Method**
Refer to EXCEL Problem Solving Method information for details on the DRPIE method. You can remember the 5 steps with the acronym DR. PIE.
The Engineering Method:

- Recognize and understand the problem
- Gather data (and verify its accuracy)
- Select Guiding theories and principles
- Make assumptions when necessary
- Solve the problem
- Verify the results
- Present the solution

2. Problem Solving Using Computers

Problem solving includes:

- Experience
- Knowledge
- Process
- Art

This class focuses on Computer Assisted Problem Solving and Programming. Computers are good tools for solving rule-based problems. For example, computers are good at chess but bad at bridge.

They are good at chess because:

- Much of basic chess is related to defined rules and keeping track of all possibilities so many moves ahead. These are the kinds of things that computers are very good at.
- Judgment in chess comes in at the higher levels, and is much of what distinguishes a grand master from a club player.
- Recent chess teaching programs, such as “Chess Titans” have the computer restricted at lower levels so that novices have a chance.

They are bad at bridge because:
ENGR 1181 MATLAB 10: Program Design
Preparation Material

- In bridge, judgment is required at the lowest levels.
- Computers have trouble playing even club level bridge.
- In the future, computers may develop judgment through neural networking or other advances, but at the present time this doesn’t effectively exist.

Therefore, look to computers when you have quantifiable rules to follow, large quantities of numbers to keep track of, and well defined procedures.

- Judgment is still the realm of Engineers.

Problem solving requires a combination of science and art:

**SCIENCE**
- Math
- Physics
- Chemistry
- Mechanics

**ART**
- Creativity
- Judgment
- Common-Sense

3. Tools for Program Development
A variety of tools and techniques can be used in the process of program development. They are useful for organizing the tasks in problem solving.

- Many of the tools are focused on the:
  - development or formulation of algorithms
  - representation of algorithms
  - refinement or structuring of algorithms
- Top-down design technique
  - Start with overall function and perform several step-wise refinements
- Outlining
  - Follow a top down approach to create an outline from the high level ideas down to the level of single logical actions.
- Pseudo code
  - Plain English descriptions of what a computer is to do
- Logic diagrams or flowcharts
  - Alternate representations of algorithms including graphic and state representations, logic flows
4. **Top-Down Design Technique**

- Begin with a single statement that conveys the overall function of the program. This is a complete (but simple) representation of the program.
- Divide this "top" statement into a series of smaller tasks which break the program down into the major components required to accomplish its function.
- Next, refine each of the smaller tasks into yet smaller steps, defining specific internal and external data as may be needed in this second refinement.
- Continue refinement until algorithm has been broken into easily developed tasks.
- Code for each of these tasks are then constructed using knowledge of available computer commands and syntax.
5. Flow Charts

- Useful visual tool for program development
- Describes the logic flow of the program
- Not a complete description of program
- Not only tool to use
- Made before writing the program
- No formal standards, but common guidelines
- Usually used to describe major program components, not individual lines of code

<table>
<thead>
<tr>
<th>Start or End a Program:</th>
<th>Decision or Selection Structure:</th>
<th>General Processing:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Quit?</td>
<td>process</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A&lt;B?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>