**A Loopy Lab**

**I. Elastic Energy Example: Using a For Loop to Accumulate an Answer**

This section examines the use of a summation to approximate an integral for the elastic energy. Many elastic devices (springs, rubber …) follow Hook’s Law: the force is proportional to the distance the spring (or other elastic device) is displaced from rest. Equation 1 is the equation for this case.

F = kx (1)

where: x = the displacement from rest

k = the hooks law constant

F = the force at x displacement

Work (W) is defined to be a force acting through a distance and if the process has no losses the resulting stored energy will be equal to the work. This results in Equation 2.

E = W = FΔx or if F varies with x, E (2)

**Integral approach:**  Substituting Hooks law (Equation 1) for the force in the integral form of Equation 2 and integrating from zero to any position x results in Equation 3. Which is exactly what was used for spring energy in ENGR 127! (See? We weren’t just making stuff up to torture you)

(3)

**Numerical approach:** If there is a very small change in x the force will be approximately constant. Based on this concept Equation 4 estimates the change in energy without an integral for a small change in spring length.

ΔE ≈ k x Δx (4)

This equation can be written as a *recursion formula* (Equation 5). A recursion formula is an equation that calculates the value at one point by using the value at a previous point (e.g., the factorial formula: yk =k \* yk-1). The subscripts represent the sequential number of the steps. Equation 5 calculates the additional energy for a small change in the spring length. The total energy can be found by calculating by adding up small steps from the spring rest to its final position. This recursion *formula* can be used to set up code where the total energy is calculated based on a series of small steps using a loop.

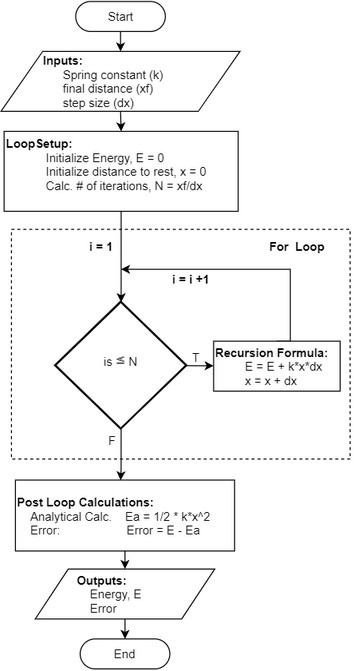
Ei - Ei-1 = k \*xi \*Δxor Ei = Ei-1 + k \*xi \*Δx (5)

To use this formula the current total displacement must be calculated for each iteration step. The total displacement is the number iterations so far times the change in distance for each iteration step; this is the first option shown in equation 6. Alternatively it may be accumulated using a recursion formula: this is the second option shown in equation 6.

xi  = i \*Δxor xi = xi-1 + Δx (6)

**Examine this Code:** Review the flow chart and code implementation if equations 5 and 6 shown in Figure 1 (see how the equations translate to the flowchart and then to the code). Understanding this code in detail will be essential to programming the problem in the next section. Examine how the *for loop* is used to apply the recursion formula, Equations 5 & 6. Notice that the total number of iterations is rounded to an integer. It does not make sense to do a fractional iteration. Also notice how the energy variable, E, is used throughout the program.

**Question:** Why is E equal to zero before the loop? What would happen to the program if E was not set to a value before the loop (comment it out and see what happens)?

****

**Code:**

function [E, error] = elastic(k, xf, dx)

%

% 🡺 Lots ‘o’ intro. Comments   
% (purpose author, date …)

%

% [E, error] = elastic(k, x, dx)

% Inputs:

% k = spring constant (N/m)

% xf = desired distance from rest (m)

% dx = the step size in numerical calc.(m)

%

% Outputs:

% E = Energy from the numerical calc.(J)

% error = numerical - analytical calc.(J)

% Internal:

% E1 = analytical solution

% N = total number of iterations

% i = current iteration number

% x = current length (m)

% for loop setup, initial energy, initial

% position, and number of steps

E = 0;

x = 0;

N = round(xf/dx);

% Loop to accumulate the total energy by

% making small changes in spring

for i = 1:N

x = x + dx;

E = E + k\*(x)\*dx;

end

% Energy by analytical equation

Ea = 1/2 \* k \*xf^2;

% calculation of error

error = E - Ea;

**Execution**Test case: k= 3000 N/m, xf = 0.1 m, Δx = 10-4 m

>> [E, error] = elastic(3000, 0.1, 0.0001)

E =

15.0150

error =

0.0150

**Figure 1:** Flowchart and code (with abbreviated comments) for the elastic energy function, elastic.m. This is an implementation of the recursion formula (Equation 5 & 6). Particularly notice the set up required before the loop and the use of E in the loop. The current distance is determined by adding the distance step to the previously accumulated distance.

**II. Capacitor Charging Exercise: Using a for loop to accumulate an answer**

Figure 2 shows a simple RC charging circuit with a switch. The capacitor starts with no charge. When the switch is closed current flows to the capacitor and the charge builds up, raising the capacitor’s voltage. As the capacitor voltage increases the voltage drop across the resistor is less and therefore the current flowing is less.



**Figure 2:** Schematic diagram of charging circuit. The capacitor is initially fully discharged (Q0 = 0). At the start time (t = 0) the switch is closed and charging begins. As the capacitor increases in charge the voltage drop across the resistor will decrease.

Equation 7 is the differential equation that results from this case for the voltage change across the capacitor. Note: you should be able to derive this formula (soon anyway).

(7)

where: *V* = Voltage across the capacitor (Volts)

*Vs* = Source voltage (Volts)

*R* = Resistance of Resistor (Ohms)

*C* = Capacitance of Capacitor (Farads)

*t* = time (s)

A numerical estimate of Equation 7 can be found (similar to the spring example) by replacing the infinitesimal dt with a finite Δt and the dV with ΔV (= Vn –Vn-1). This results in equation 8, a *recursion formula* for the change in voltage over a very short period of time where the change in current is negligible.

(8)

where: *Vi* = Voltage across the capacitor in iteration i (Volts)

*i, i+1*= subscript representing the iteration (step) number

*Δt* = time change between iterations (seconds)

This recursion formula can be used to add up the change in voltage across capacitor over many short periods. This equation can be programed similar to the recursion formula for the spring problem (Equation 6).

**Calculations to Compare Program Results:** The results of a program based on the recursion equation 8 can be compared to the results from equation 9. This equation is the solution to the differential equation 7.

(9)

where: *Va* = the capacitor voltage from this analytic solution at time t

To calculate an exponential in most computer languages, including MATLAB and Excel, use an *exp()* function. For example if you wish to know ***e2***, the MATLAB command would be ***exp(2)***.

**Function Problem:** Develop a flowchart using the flowchart template available online, and then develop a function to calculate the voltage across the capacitor, given the supply voltage (Vs), the resistance(R), the capacitance (C), and the charging time (t) as inputs. Base the program on the setup shown in Figure 3. The initial charge and therefore voltage on the capacitor at time zero is zero. Be sure and have a single line for hard coding the step size input.

***Note:*** This function exercise parallels the elastic example (elastic.m) – review that code for ideas.

Set Up/ Planning Type of Program:  Script 🗹 Function

1. Problem Statement (in your own words):

The goal of this program is to calculate the charge of a capacitor in a simple DC charging circuit after a given length of time. The function is to calculate the time via a charging equation and a numerical estimate. The output will be a value of V calculated from Eq. 8 and an error term.

1. Inputs: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Units or Values | Input Source\* |
| Vs | The source voltage (a constant DC) | Volts | Command Line |
| R | Resistance of the resistor | Ohms | Command Line |
| C | Capacitance of the capacitor | Farads | Command Line |
| tf | Charging time | seconds | Command Line |
| dt | Delta time used in the numerical estimate | seconds | Hard Code |

\* Possible sources: command line, file, interactive input (input or menu functions)

1. Output: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Units or Values | Output type\* |
| Vn | Voltage from numerical equation | Volts | Command Line |
| Error | Vn-Va | Volts | Command Line |

\* Possible types: command line, file, display, figure window

1. Solution Steps (order of these two parts may be varied):
2. Perform calculation on test case(s) (2) Identify the steps/equations to be used in code

Include flowchart when appropriate

Use equation 8 in an accumulation loop like the one used in the program elastic.m

Test Cases (you choose a small dt)

Steps: Hand draw a flowchart to calculate capacitor voltage based on the recursive voltage formula (eq. 8) and compare it to the analytical formula (eq 9)

1. Vs = 9 V; R = 300 Ω; C = 0.003 F; tf = 1
2. Same as above with t = 3
3. Test a case of your own

**Figure 3:** The set up portion of the Program Development Worksheet for the first Capacitor Charging Exercise. Hand draw a flow chart for this program and then develop the code. Be sure and pay attention to units.

**III. Impact of Step Size**

**MATLAB Function:** Use the code developed in the last section to look at the impact of step size (dt).

**Exercise:** Run *your code* for, Vs = 9 V; R = 1000 Ω; C = 0.001 F; t = 1, use several the Δt values shown in Table 1 (i.e., Δt = 0.1, 0.01, & 0.001, and 0.0001 s) for the program’s step size. Pay attention to the units used by the program. Summarize the results using Table 1.

**Table 1:** Numerical estimates of voltage (pay attention to program units)

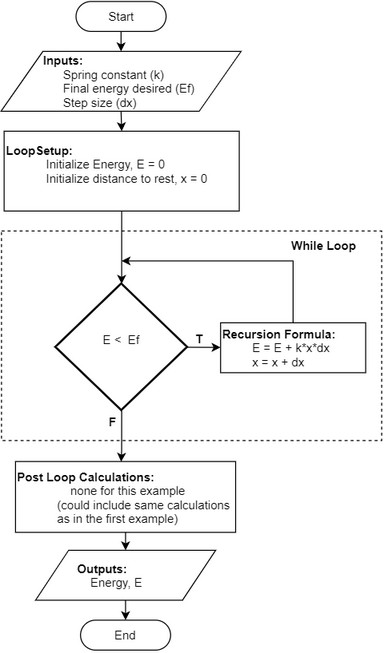
|  |  |  |
| --- | --- | --- |
| **Δ t (sec)** | **Numerical estimate of V (Volts)** | **Error in Estimate of V (Volts)** |
| 0.1 |  |  |
| 0.01 |  |  |
| 0.001 |  |  |
| 0.0001 |  |  |

**Include:** The filled inTable 1 with your final problem submittal.

**IV. Accumulation with a while loop**

***Example Problem:*** In the previous spring energy problem the energy is calculated based on the compression of a spring. What if the opposite is desired? In other words if a function is needed to return the compression required to obtain a specified energy storage. In this case it is no longer possible to calculate the number of loops required and therefore a *for loop* will not work. For this case, a *while loop* is needed. A while loop will continue to repeat until a desired condition is met. Appendix B provides background on while loops.

* 1. Review Appendix for background on while loops (Ignore these step at your own peril).
  2. Create and try the simple code example in Appendix B. Observe the resulting loop behavior. This particular example is a loop counter. This simple example is a critical one to remember. Loop counters are often needed and provide a good basis for figuring out the code for other accumulation type problems. This simple code example is one you are expected to be able to code with little effort. **There may be a quiz next week on creating a loop counter.**
  3. **Example Flow Chart and Code:** The program **elastic2.m** is shown in Figure 4, along with its flowchart and an execution example (full validation would require more tests). This program provides an example of using a while loop to solve the problem of how much spring compression is needed to reach a specific energy. Study this code and make sure you understand how it is working.
  4. **An important note:** With while loops it is very easy to get an infinite loop, one that never stops running but keeps looping forever. If this happens press the control (CTRL) key and the “c” key simultaneously. This will stop the program and return the command prompt.
  5. **Flowchart:** For the charging capacitor, develop the flow chart for a program that will determine the time required to reach a specific voltage based on the recursive formula. Use a while loop. Use the online template to develop this flow chart. To figure out how to do this review the flow chart in Figure 4.
  6. **Function:** Based on the flowchart you have developed, create a function that will accomplish this goal. Pay attention to the example code in Figure 4 for the spring problem.

****

**Figure 4:** Flowchart, code and execution for the second elastic energy function, elastic2.m. This program calculates the compression needed to reach a desired energy. Here the recursion formula in Equations 5 & 6 are implemented with a while loop. Particularly notice the set up required before the loop and the use of E in the loop. Notice the execution example is simply the reverse of the problem done in part I.

**Code**

function x = elastic2(k, Ef, dx)

% Calculates the stretch required to

% reach a specified elastic energy

% based on a numerical approximation.

% S. Scott Moor

% Revised November 2016

%

% function: x = elastic2(k, Ef, dx)

%

% Inputs:

% k = spring constant (N/m)

% Ef = desired Energy (J)

% dx = the step size in numerical calc.(m)

%

% Outputs:

% x = final distance from rest (m)

% + for stretch - for compression

% Internal:

% x = current length (m)

% while loop setup:   
% initial energy, and initial position

E = 0;

x = 0;

% Loop to accumulate the total energy

while E < Ef

x = x + dx;

E = E + k\*(x)\*dx;

end

**Execution**   
for: k = 3000 N/m, E = 15 N, dx = 0.0001 m

>> x = elastic2(3000,15, 0.0001)

x =

0.1000

**V. Lab Assignment Summary** (due next laboratory session)**:**

1. Voltage across a capacitor in an RC circuit (Section II) : Develop a function that matches the setup in Figure 3 for calculating the voltage across a capacitor as a function of time. Include a clear flowchart, the hand solution to the test cases using equation 9, well commented code and the validation of this function. For test cases use the Figure 3 test case value plus one other test case of your choosing.

Include a filled in copy of Table 1 (Section III), your testing results for different time step sizes.

1. Develop flowchart & code that will return the required charging time given a desired capacitor voltage (i.e. the above problem). Document your setup, code and validation runs with the program development worksheet.

**Problem Statement:** Must be a paraphrase in your own words.

**Input/output tables:** Use the variable names listed in these tables in your function

**Steps & Calculation**: A Visio-drawn flowchart as the planned steps for # 4 on the worksheet, plus clearly presented hand calculations of the tests cases for comparison to validation   
(e.g., solve equation 9 for time and apply).

**Validation:** use a rearrangement of Equation 9 to validate the program’s result.   
**Required Test Case:** R = 200 Ohms, C = 0.0004 Farads (400 μF), Vs = 9 Volts, Vc = 8 Volts.

The Program Development Worksheet and flow chart template are available online.

1. **Capacitor Charging Function 1: Voltage after a given time**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Area** | **Expectation** | **✓ = 1 pt** |
| 1 | Lay out | Problem clearly laid out in a logical order. Including:   1. Flow Chart, 2. Hand test calculations 3. Code 4. Validation |  |
| 2 | Flow Chart | Flow Chart is used to show Program Steps |  |
| 3 | Flow Chart is complete & accurate.  Properly and clearly formatted, easy to read |  |
| 4 | Program Code | Code for a function provided with comments including useful help response, comments listing variables & units, and program logic |  |
| 5 | .m file included can run |  |
| 6 | Code includes some correct elements |  |
| 7 | Code logic is largely correct calculations |  |
| 8 | Code is completely correct |  |
| 9  10 | Validation | Program execution provided showing  match to known correct results  Includes required test case  Includes complete and accurate Table 1. |  |
|  |

1. **Capacitor Charging Function 2: Time to charge a given voltage**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Area** | **Expectation** | **✓ = 1 pt** |
| 1 | Worksheet – Set up and Flowchart | Problem clearly laid out using Program Development Worksheet  Goal of program presented, Inputs & Outputs for program listed (1-3) |  |
| 2 | Flow Chart is used to show Program Steps (4) |  |
| 3 | Flow Chart is complete & accurate.  Properly and clearly formatted (4) |  |
| 4 | Code for a Function Provided with comments including useful help response, comments listing variables & units, and program logic |  |
| 5 | Program Code | .m file included can run |  |
| 6 | Code includes some correct elements |  |
| 7 | Code logic is essentially correct |  |
| 8 | Code is completely correct |  |
| 9  10 | Validation | Program execution provided showing  match to known correct results  Includes required test case |  |
|  |

**Appendix:** While Loop Background

1. Initial example of simple indexing (type in and try this script)

% program whileeg

% this script demonstrates a simple while loop

x=6

k=0; % this initializes the loop variable

while k <= x % logical condition controls loop

k % echo print loop variable to see what is happening

k=k+1; % change the loop variable

end

Inspect this code and the result – what do you notice?

Try some variations:

* vary the loop index increment calculation (k = k + 1) above to have different step sizes (e.g., try k = k + 3)
* use a different operation (e.g., try k = k\*2)
* vary the starting point by changing the initialization (i.e., change the k = 0 value).

1. English: while \_\_\_\_ is true do this
2. General form:   
    *initial definition of loop variable(s)*

while *logical expression*

*statements (including change of loop variable(s))*

end

1. Flowchart: