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.

$$F = kx \quad (1)$$

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.

$$E = W = Fx \quad \text{or if F varies with x, } E = \int_0^x F dx \quad (2)$$

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)

$$E = \int_0^x F dx = \int_0^x kx dx = \frac{1}{2}kx^2 \quad (3)$$

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.

$$\Delta E \approx k \times \Delta x \quad (4)$$

This equation is written as a *recursion formula* in equation 5. A recursion formula is an equation that calculates the value at one point by using the value at a previous point (ex: the factorial formula we reviewed earlier in lab). The subscripts represent the sequential number of the steps. This formula calculates the additional energy for a small change in the spring length. Calculating in small steps from the spring rest to its final position the total energy. This recursion *formula* can be used to set up code where the total energy is calculated based on a series of small steps.

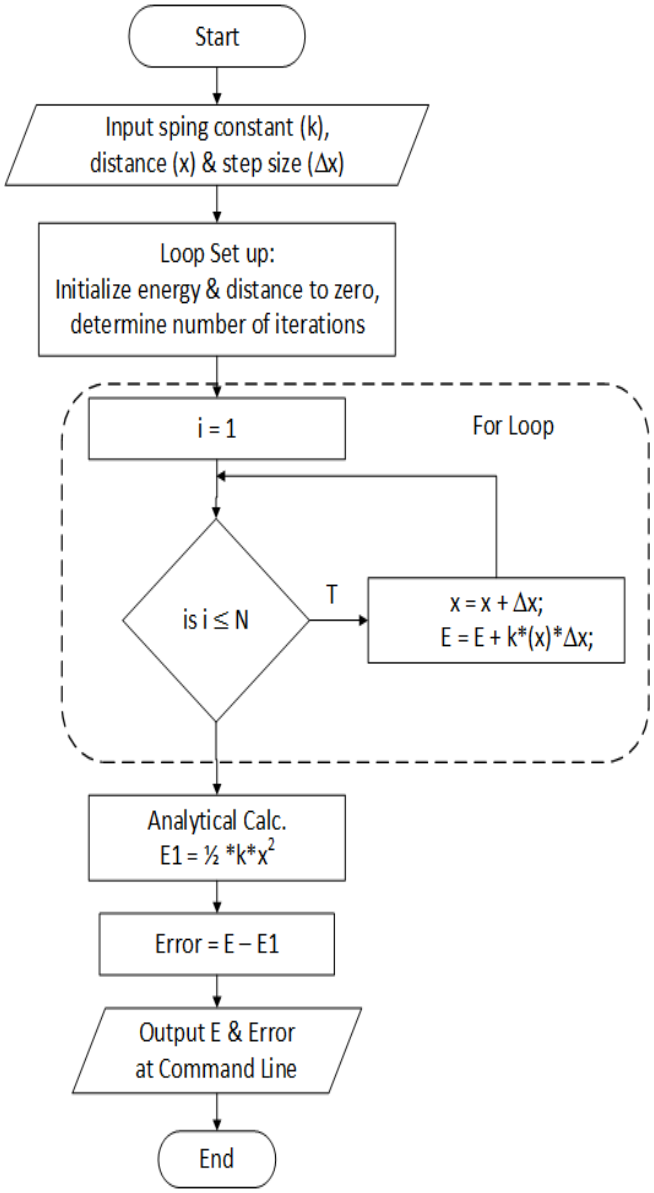
$$E_i - E_{i-1} = k * x_i * \Delta x \quad (5)$$

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 the recursion formula: this is the second option shown in equation 6.

$$x_i = i * \Delta x \quad \text{or} \quad x_i = x_{i-1} + \Delta x \quad (6)$$

Examine this Code: review the flow chart and implementation in 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. A flowchart and the abbreviated code for this problem are shown in Figure 1. Notice that the number of iterations is rounded to an integer, so that the loop index statement uses integers. 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)?



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
   E1 = 1/2 * k *x^2;

% calculation of error
   error = E - E1;
```

Execution

for k= 3000 N/m, xf = 0.1 m, $\Delta x = 10^{-4}$ 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 4 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 capacitors voltage. As the capacitor voltage increases the voltage drop across the resistor is less and therefore the current flowing is less.

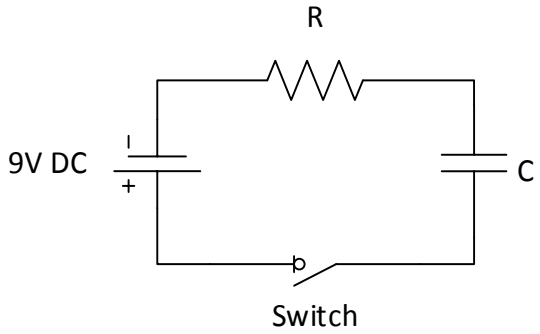


Figure 4: Schematic diagram of charging circuit. The capacitor is initially fully discharged (Q_0 = 0). At the start time (t = 0) the switch is closed and charging begins. As the capacitor increases in charge the voltage drop across the resistor will decrease.

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.

$$\frac{dV}{dt} = \frac{V_s - V}{RC} \tag{7}$$

where: V = Voltage across the capacitor (Volts)
V_s = 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 (= $V_n - V_{n-1}$). 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.

$$V_i = V_{i-1} + \frac{(V_s - V_{i-1})}{RC}\Delta t \tag{8}$$

where: V_i = 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. Note: You should be able to solve equation 7 to obtain equation 9.

$$V_a = V_s\left(1 - e^{-t/RC}\right) \qquad (9)$$

where: V_a = the capacitor voltage from this analytic solution

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

Function Problem: Hand draw a flowchart and develop a function to calculate the voltage across the capacitor, given the supply voltage (V_s), the resistance(R), the capacitance (C), and the charging time (t) as inputs. Base the program on the setup shown in Figure 5. 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.

2. 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
t	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)

3. 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

4. Solution Steps (order of these two parts may be varied):

(1) 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)

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

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)

Figure 5: 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 in Table 1 with your code.

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.

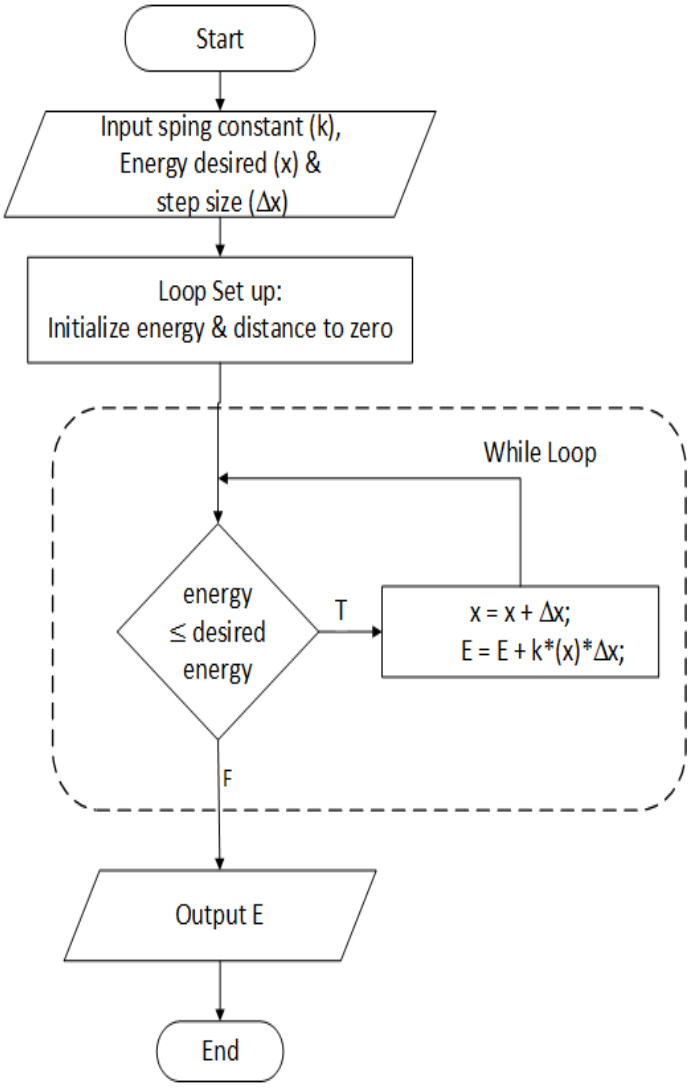
a. Review Appendix for background on while loops.

b. 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.**

c. **Example Code:** The program **elastic2.m** is shown in Figure 6, 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.

d. **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.

e. **Function Problem:** For the charging capacitor, develop a program that will determine the time required to reach a specific voltage based on the recursive formula. Use a while loop. To figure out how to do this program review the elastic2.m function that solves a similar spring program.



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
```

Figure 6: 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.

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

1. Voltage across a capacitor in an RC circuit: Develop a function that matches the setup in Figure 5 for calculating the voltage across a capacitor as a function of time. Include a clear flowchart, the hand solution to the test case in Figure 5 using equation 8, well commented code and the validation of this function using the Figure 5 test case plus one other test case.

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

2. 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 is the planned steps for # 4 on the worksheet
   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), V_s = 9 Volts, V_c = 8 Volts.

   The Program Development Worksheet is 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	Validation	Program execution provided showing match to known correct results	
10		Includes required test case Includes complete and accurate Table 1.	

2. 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	Validation	Program execution provided showing match to known correct results	
10		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).

2. English: while ____ is true do this

3. General form:

initial definition of loop variable(s)
while *logical expression*
statements (including change of loop variable(s))
end

4. Flowchart: