Designing a Program and Subroutines

Summary by: Emmanuel J Rodriguez

Note: Subroutines are commonly called, depending on the programming language, modules, subprograms, methods, and functions.

Top-down design (sometimes called stepwise refinement) is used to break down an algorithm into subroutines.

Top-Down Design Process:

- The overall task of the program is broken down into a series of subtasks.
- Each of the subtasks is examined to determine whether it can be further broken down into more subtasks. This step is repeated until no more subtasks can be identified.
- Once all of the subtasks have been identified, they are written in code.

Three main tools for designing a program and its subroutines:

1. **Hierarchy Chart** – or a structure chart, a top-level visual representation of the main program and the relationships between subroutines.
2. **Flowcharts** – a diagram that graphically depicts the steps that take place in a program.
3. **Pseudocode** – or "fake code" is an informal language that has no syntax rules, it is a "mock-up" program. Each statement in the pseudocode represents an operation that can be performed in any high-level language.

Source: Gaddis, T., *Programming Logic and Design* (New Jersey: Pearson, 2013) pp. 79 – 112.

6/24/21 19:16 @ Ft. Hancock

Top-Down Design
Program: Solve the Heat Equation Numerically (Solving_PDEs_numerically_Parabolic_Crank_Nicolson_Hand written_2021-07-02_152842.pdf)

Overall Task:
Develop a general-purpose MATLAB program to solve the one-dimensional heat equation (partial differential equation - PDE) using the Crank-Nicolson method.

Steps that must be taken to perform the task:
1. Mathematical modeling – represent all the important features of the system; see the figure below for the model.
2. Derivation of governing equations. Re-write the second-order PDE as a finite difference equation.
3. Solution of the governing equations. Solve the system of equations using MATLAB, **to obtain a series of spatial distributions corresponding to the state (Temp.) of the node at each time**.
4. Interpretation of results. The solution of the governing equations gives the temperature distribution of the element of the system. Plot these responses.

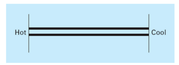


FIGURE 30.1
A thin rod, insulated at all points except at its ends.

1. Hierarchy Chart

Main Program
(Output: temperature(space, time); Input: coefficient of thermal diffusivity, domain, element size, temporal step size, boundary conditions, initial conditions – heat input?)

Note: Hierarchy charts does not show the steps that are taken inside a subroutine; they do not reveal any details about how subroutines work.

Analysis Procedure Background:
Step 1: Mathematical Modeling And Step 2: Derivation of Governing Equations

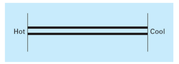
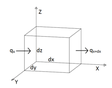


FIGURE 30.1
A thin rod, insulated at all points except at its ends.



Conservation of heat principle is used to develop a heat balance for the differential element:
(rate of heat in) = (rate of heat out) + (rate of heat stored)

FIGURE 30.1 A thin rod, insulated at all points except at its ends.

$$q(x)\Delta y\,\Delta z\,\Delta t = q(x+\Delta x)\,\Delta y\,\Delta z\,\Delta t + \Delta x\,\Delta y\,\Delta z\rho C\,\Delta T$$

Dividing by the volume of the element ($= \Delta x\,\Delta y\,\Delta z$) and Δt gives

$$\frac{q(x)-q(x+\Delta x)}{\Delta x} = \rho C\frac{\Delta T}{\Delta t}$$

Taking the limit yields

$$-\frac{\partial q}{\partial x} = \rho C\frac{\partial T}{\partial t}$$

Substituting Fourier's law of heat conduction [Eq. (29.4)] results in

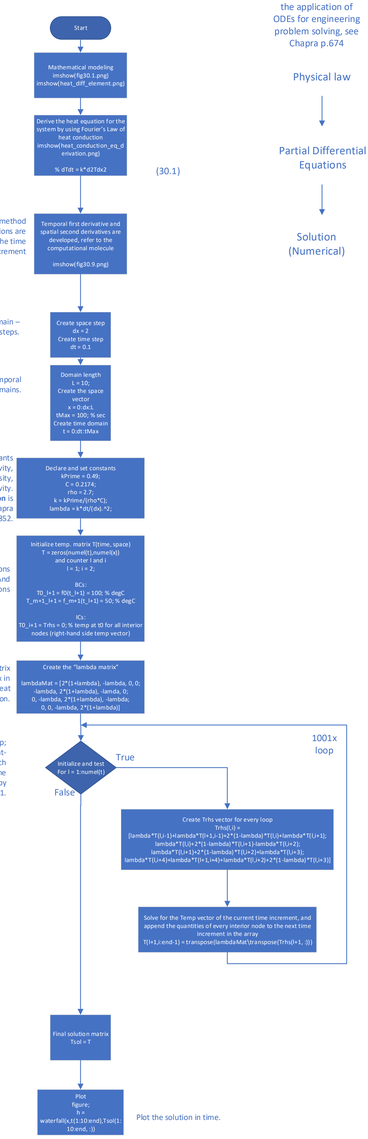
$$k\frac{\partial^2 q}{\partial x^2} = \frac{\partial T}{\partial t}$$

which is the *heat-conduction equation*.

See Chapra, p. 840.

Step 1: Mathematical Modeling
Step 2: Derivation of Governing Equations
Step 3: Solution of the Governing Equations

2. Flowchart



Step 4: Interpretation of Results

Sequence of events in the application of ODEs for engineering problem solving, see Chapra p.674

Physical law

↓

Partial Differential Equations

↓

Solution (Numerical)

3. Pseudocode

```
% Start
% Step 1 Mathematical Modeling

% Display image that represents system model
imshow(fig30.1.png)
imshow(heat_diff_element.png)

% Step 2: Derivation of Governing Equations

% Fourier's Law of heat conduction
imshow(heat_conduction_eq_derivation.png)
% dTdt = k*d2Tdx2

% Step 3: Solution of the Governing Equations

% Crank-Nicolson method
% diff approximations are developed at the midpoint of the time increment
imshow(fig30.9.png) % computational molecule

% Domain
dx = 2; % space step
dt = 0.1; % time step
L = 10; % domain length
x = 0:dx:L; % space vector
tMax = 100; % temporal domain
t = 0:dt:tMax-dt; % time vector


% Declare and set constants, aluminum material
kPrime = 0.49; % thermal conductivity
C = 0.2174; % specific heat capacity
rho = 2.7; % density
k = kPrime/(rho*C);

% Function's stability criterion
lambda = k*dt/(dx).^2;


% Boundary conditions and Initial conditions

% Initialize Temp matrix, which will be a function of node and time step, T(time, space), and counters I and i
T = zeros(numel(t), numel(x)) % zeros for all time x all nodes

I = 1; i = 2; % time step number one is initialized at element 1, node number is initialized at node #2, because the spatial second derivative is
determined at the midpoint (current node) by averaging the difference approximations at the beginning and end of the time increment; this
is done through averaging temp quantities for three nodes, and therefore the first approximation will be of nodes 1, 2, and 3 – which is why
we initialize i at 2. Essentially, we are integrating forward in time.

% BCs:
T(:, x(1)) = 100; % degC, T as a function of all time and node 1 (only), that is, the left end is fixed at the value given
T(:, x(end)) = 50; % degC, T will be set to the value for all time to the last (end) node in the system, that is, the right end is fixed

% ICs:
% Temp data for the interior nodes, this is the RHS temp vector
intNodes = numel(x) – 2; % calculates the number of interior nodes
Trhs = zeros(numel(t), intNodes); % Declare, set, and initialize Trhs, to hold the RHS temp vector (INTERIOR) nodes only


% Create the lambda matrix
% The lambda matrix is analogous to the stiffness matrix in FEA, but in the context of heat transfer (obv)
lambdaMat = [2*(1+lambda), -lambda, 0, 0;
             -lambda, 2*(1+lambda), -lambda, 0;
             0, -lambda, 2*(1+lambda), -lambda;
             0, 0, -lambda, 2*(1+lambda)]

% Calculate the solution Temp for all time, and all nodes, then append the temp-node value to the Temp matrix using counter I+1.
% Noting that t=0 sec starts at I=1, and from ICs the interior nodes temp = 0; and t=0.1 sec is I=2.
for I = 1:numel(t)

    % Node 2 temp of Temp matrix, time +1:
    Trhs(I+1, i-1) = lambda*T(I,i-1)+lambda*T(I+1,i-1)+2*(1-lambda)*T(I,i)+lambda*T(I,i+1);
    % Node 3:
    Trhs(I+1, i) = lambda*T(I,i)+2*(1-lambda)*T(I,i+1)-lambda*T(I,i+2);
    % Node 4:
    Trhs(I+1, i+1) = lambda*T(I,i+1)+2*(1-lambda)*T(I,i+2)+lambda*T(I,i+3);
    % Node 5:
    Trhs(I+1, i+2) = lambda*T(I,i+4)+lambda*T(I+1,i+4)+lambda*T(I,i+2)+2*(1-lambda)*T(I,i+3);

    % Solve the system of equations, outputting Temp for each time-node combination, append the quantities of every interior node to
    the next time increment in the array.
    T(I+1,i:end-1) = transpose(lambdaMat\transpose(Trhs(I+1, :)));

end

% Final solution matrix
Tsol = T

% Step 4: Interpretation of the Results

% Plot solution in time
figure;
h = waterfall(x, t(1:10:end), Tsol(1:10:end))

% End program ☺
```