```
% ### EXgaussSUM.m ###  2016.01.08 CB

% re PHYS 2030 HW1, add the first N (e.g., 100) integers. Done two differents
ways
% o Method A - via a simple use of sum.m
% o Method B - via a for loop
% o Method C - simply a closed form solution (Gauss' insight)

% Refs:
% o http://en.wikipedia.org/wiki/Carl_Friedrich_Gauss
% o http://mathcentral.uregina.ca/QQ/database/QQ.02.06/jo1.html

clear
% ---------------
N= 51;     % upper limit
% ---------------

disp(['Adding up the first ',num2str(N),' integers']);
% ===
valA= sum(1:N);
disp(['Method A yields ',num2str(valA)]);

% ===
valB= 0;    % initialize to zero
for n=1:N
    valB= valB+n;
end
disp(['Method B yields ',num2str(valB)]);

% ===
valC= (N+1)* N/2;
disp(['Method C yields ',num2str(valC)]);
```

```
% ### EXdiff3.m ###      2016.01.08

% modified version of EXdiff1.m to deal w/ a tanh function


clear all;
% --------------
% User parameters
np = 30; % number of points along curve
Nfact= 0.01; % scale factor for additive noise
% --------------


    func = @(t)tanh(t); % function
    dfunc = @(t)1- (tanh(t)).^2; % derivative of function
    tStart= -2*pi;  tEnd= 2*pi; loc= 'NorthWest';

% ----
t = linspace(tStart,tEnd,np);  % determine time array
y = func(t)+ Nfact*randn(numel(t),1)';  % determined 'sampled' points (which
can be noisy)
dydt = diff(y)./diff(t);    % compute derivative via numerical difference
% ----
% visualize
tt = linspace(tStart,tEnd,np*100);    % plot actual curve sans noise
(oversample time here)
yy = func(tt);
figure(1); clf;
plot(tt,yy,'m-','LineWidth',2); hold on; grid on;
yy = dfunc(tt);
plot(tt,yy,'k--','LineWidth',2);
plot(t,y,'bs','LineWidth',2);    % plot points and connecting lines
% central difference - plot numerical derivative at midpoint
ttCD = t(1:end-1) + diff(t)./2;
plot(ttCD,dydt,'r+','LineWidth',2,'MarkerSize',8);
xlabel('t'); ylabel('y or dy/dt'); title('Example of numerical
differentiation');
legend('y(t)','dy/dt','measured y(t) (w/ noise)','dy/dt
(diff.m)','Location',loc);
% ----
% compare to Matlab's built-in gradient.m (which uses centered differences)
if 1==1
    slope= gradient(y,t);
    plot(t,slope,'o','LineWidth',2,'Color',[0 0.75 0])
    legend('y(t) (no noise)','dy/dt (analytic)','yn(t) (i.e., w/
noise)','dyn/dt (diff.m)','dyn/dt (gradient.m)','Location',loc);
end
```