*function similar\_sequence = generator\_FFT(n,H)     
%--------------------------------------------------------------------------    
% GENERATOR\_FFT Use fast fourier transform to generate normalized FGN     
%     and FBM. Then use Norrs method to generate normalized    
%     similar\_sequence. Finally, the average of similar\_sequence was set to \_x0005\_    
%     1 through normaliztion.   
%       
%     Note:     
%     1. The input argument n is the number of point of sequence. It must |     
%     be even. H is the objective similarity you want.   
%     2. The output argument similar\_sequence is a similar\_sequence with   
%     average equal to 1. The FGN and FBM are normalized FGN and FBM     
%     respectively.     
%     3. This routine is a matlab version of paxson's R routine. For more     
%     details, see "Fast, approximate synthesis of fractional Gaussian     
%     noise for generating self-similar network traffic  
%--------------------------------------------------------------------------    
  
%--------------------------------------------------------------------------     
%    
%    generator\_FFT     
%    Edit by Chu Chen, 07/07/2007  
%    Should you have any suggestion for improving the code, please contact:    
%    [email]chuch@scut.edu.cn[/email].  
%--------------------------------------------------------------------------  
  
if mod(n,2) ~= 0   
    error('The input argument "n" must be even');  
else     
    % Returns a Fourier-generated sample path of a "self similar" process     
    % Consisting of n points(n should be even) and Hurst paramenter H     
    n = n/2;     
    lambda = [1:n]\*pi/n;    
  
    % Approxiamte ideal power spectrum.    
    f = FGNspectrum(lambda,H);  
    
    % Adjust for estimating power spectrum via periodogram     
    f = f.\*exprnd(1,1,n);   
    
    % Construct corresponding complex numbers with randm phase    
    alpha = 2\*pi.\*unifrnd(0,1,1,n);  
    a = sqrt(f).\*cos(alpha);     
    b = sqrt(f).\*sin(alpha);  
    z = complex(a,b);   
    
    % Last element should have zero phase  
    z(n) = abs(z(n));  
    
    % Expand z to correspond to a Fourier transform of a real-valued signal. \_x0012\_    
    zprime = [0,z,conj(fliplr(z(1:n-1)))];   
    
    % Inverse FFT gives sample path.  
    FGN = real(ifft(zprime));     
        
    % Standardize FGN and create FBM.     
    FGN = (FGN-mean(FGN))/std(FGN);  
    FBM = cumsum(FGN);     
  
    % Use Norrs method to generate normalized similar\_sequence  
    similar\_sequence = FGN;     
    \_x0012\_    
    % M = 30;     
    % a = 5;     
    % similar\_sequence = M + sqrt(a\*M)\*similar\_sequence;     
    % similar\_sequence = max(0,similar\_sequence);    
    % similar\_sequence = similar\_sequence\*2\*n/sum(similar\_sequence);     
end;     
%----------------------------subfunction1----------------------------------    
function f = FGNspectrum(lambda,H)   
% Returns an approximation of the power spectrum of FGN at the given     
% frequencies lambda and the given Hurst parameter H. \_x0005\_\_x0012\_    
f = 2\*sin(pi\*H)\*gamma(2\*H+1).\*(1-cos(lambda)).\*(lambda.^(-2\*H-1) + FGNest(lambda,H));    
    
%----------------------------subfunction2----------------------------------    
function est = FGNest(lambda,H)    
% Returns the estimate for B(lambda,H).    
d = -2\*H-1;     
dprime = -2\*H;     
a1 = 2\*1\*pi+lambda;     
b1 = 2\*1\*pi-lambda; \_x0005\_    
a2 = 2\*2\*pi+lambda; `    
b2 = 2\*2\*pi-lambda;     
a3 = 2\*3\*pi+lambda;     
b3 = 2\*3\*pi-lambda;    
a4 = 2\*4\*pi+lambda;     
b4 = 2\*4\*pi-lambda;    
est = a1.^d + b1.^d + a2.^d + b2.^d + a3.^d + b3.^d + (a3.^dprime+b3.^dprime+a4.^dprime+b4.^dprime)/(8\*pi\*H)*