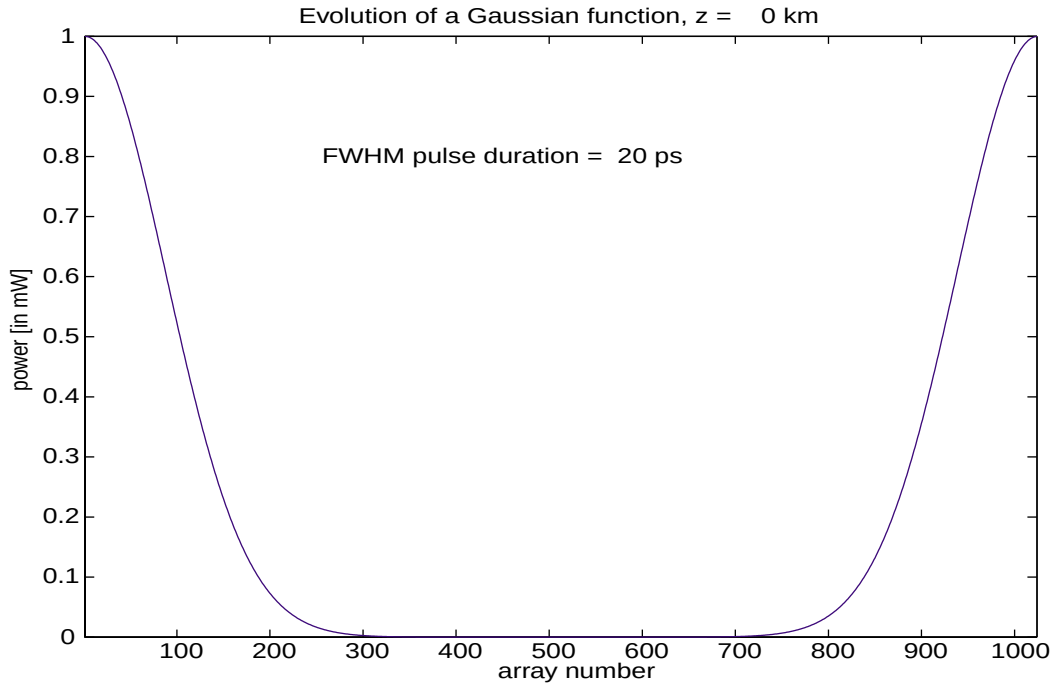
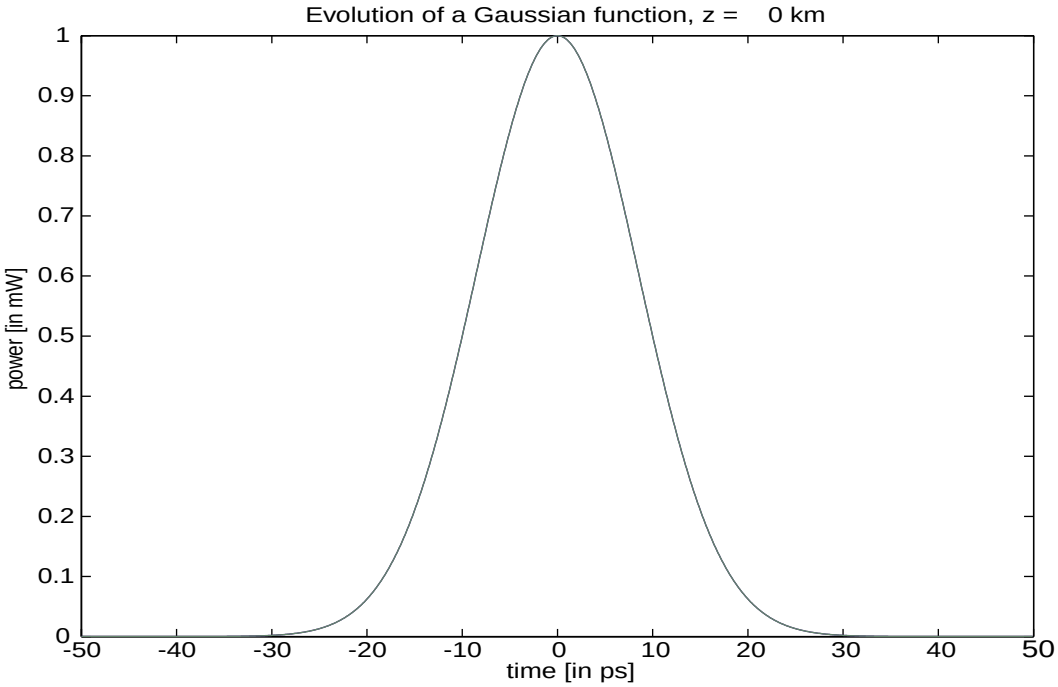
MATLAB code for evolution of a Gaussian pulse in optical fibers
(January 29, 2003 — C. R. Menyuk)

```
% Gauss_evolve
%
% Calculate and plot the evolution in an optical fiber of a Gaussian pulse

% Input parameters
%
N = 1024          % total number of points kept in time window
T_domain = 100    % total time domain kept [in ps]
t_FWHM_0 = 20     % initial pulse FWHM [in ps]
P_0 = 1           % initial peak power [in mW]
D = -1            % dispersion coefficient [in  ps/nm-km]
z_plot = [0 100 200 500]  % z-values to plot [in km]

% Derived parameters
%
Delta_t = T_domain/N;        % node spacing in time
Delta_om = 2*pi/T_domain;    % node spacing in radial frequency
Amp = sqrt(P_0);             % initial amplitude of the Gaussian
t_0 = t_FWHM_0/(2*sqrt(log(2)));   % initial pulse standard deviation
k_0_dp = -1.2*D;  % translation to ps^2/km
                  % WARNING:  Only valid at 1.5 microns

% Input vectors
%
t_vec = Delta_t*(-N/2:1:(N/2)-1);    %create array of time points
om_vec = Delta_om*(-N/2:1:(N/2)-1);  %create array of radial frequency points
u_vec = Amp*exp(-t_vec.^2/(2*t_0^2));  %create array of initial amplitudes

% Shift u-vec for the DFT
%
a_vec(1:N/2) = u_vec((N/2)+1:N);
a_vec((N/2)+1:N) = u_vec(1:N/2);

% Shift and square om_vec for use in calculating z-evolution
%
om_vec_square_shift(1:N/2) = om_vec((N/2)+1:N).^2;
om_vec_square_shift((N/2)+1:N) = om_vec(1:N/2).^2;

% Carry out IFFT (appropriate for physics convention)
%
a_tilde_vec = ifft(a_vec,N);

% Enter loop to determine and plot z-evolution
%
n_plot = length(z_plot);   %determine number of z-values
for i_plot = 1:n_plot;

    % Determine the z-value and the exponential factor for evolution
    %
    z_val = z_plot(i_plot);
    factor = exp((sqrt(-1)/2)*k_0_dp*om_vec_square_shift*z_val);
       % NOTE:  I use sqrt(-1) for i to avoid problems with re-definition

    % Carry out the z-shift and transform to time domain
    %
    a_tilde_vec_z = a_tilde_vec.*factor;
    a_vec_z = fft(a_tilde_vec_z,N);

    % Calculate powers as a function of array number and time
    %
    P_a_z = a_vec_z.*conj(a_vec_z);   % power as a function of array number
    P_u_z(1:N/2) = P_a_z((N/2)+1:N);  % shift to obtain power as a ...
    P_u_z((N/2)+1:N) = P_a_z(1:N/2);  % function of time

    % Calculate analytical formula for the power for comparison
    %
    t_z = sqrt(t_0^2 + (k_0_dp*z_val)^2/t_0^2);  % new standard deviation
```

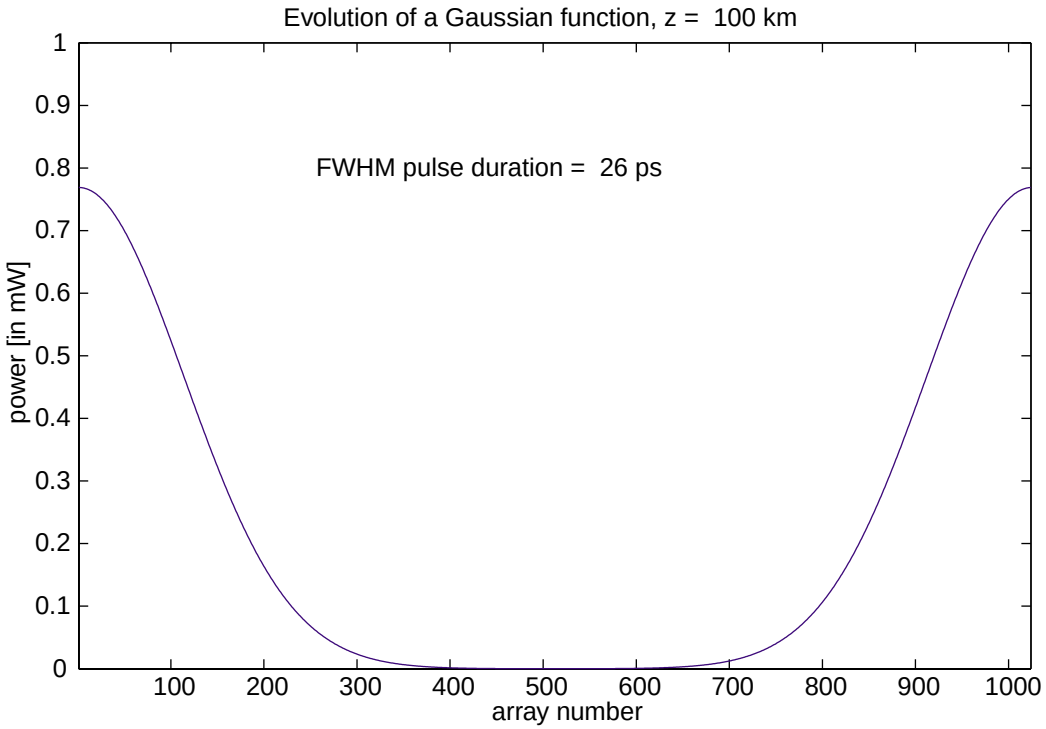
```
    t_FWHM_z = 2*sqrt(log(2))*t_z;               % new FWHM duration
    P_comp_z = P_0*(t_0/t_z)*exp(-t_vec.^2/t_z^2);

    % Plot results
    %
    plot(P_a_z)                      % power as a function of array number
    axis([1 N 0 P_0])
    xlabel('array number')
    ylabel('power [in mW]')
    titlabel = sprintf('Evolution of a Gaussian function, z = %4.0f km'...
        ,z_val);
    title(titlabel)
    timetext = sprintf('FWHM pulse duration = %3.0f ps',t_FWHM_z);
    text(0.25*N, 0.8*P_0,timetext)
    pause
    %
    plot(t_vec,[P_u_z;P_comp_z])     % power as a function of time
    axis([-T_domain/2 T_domain/2 0 P_0])
    xlabel('time [in ps]')
    ylabel('power [in mW]')
    titlabel = sprintf('Evolution of a Gaussian function, z = %4.0f km'...
        ,z_val);
    title(titlabel)
    % --- suppress plotting of t_FWHM when it will overlap plot
    if t_0/t_z < 0.8
        timetext = sprintf('FWHM pulse duration = %3.0f ps',t_FWHM_z);
        text(-0.25*T_domain, 0.8*P_0,timetext)
    end

    pause
end
```

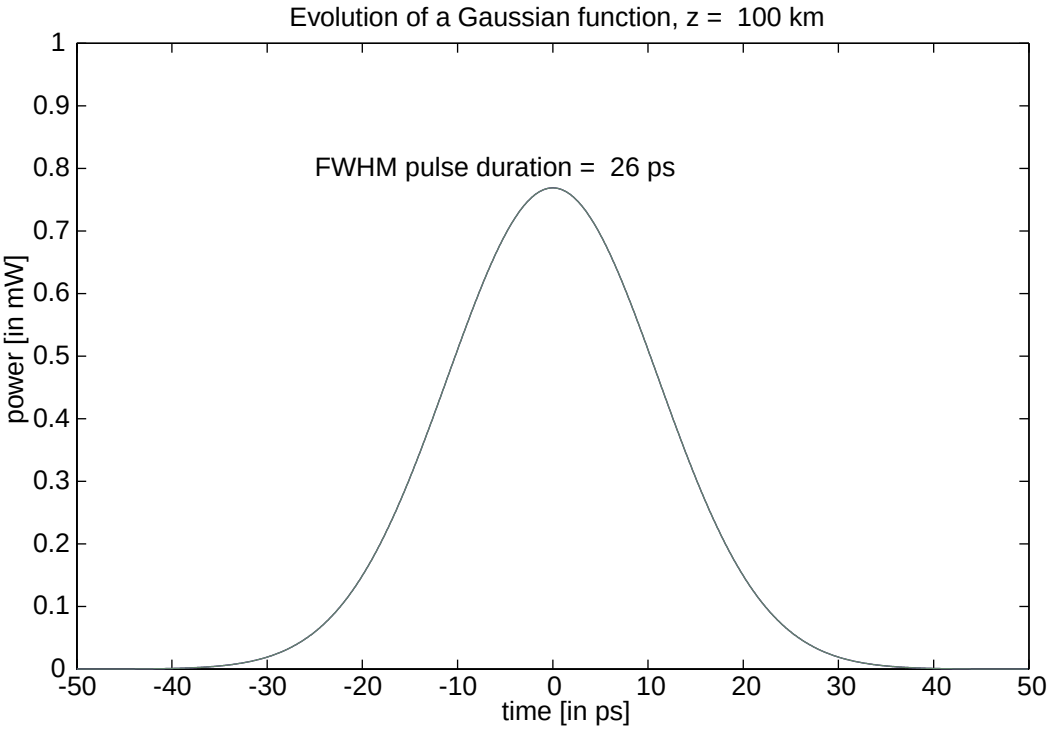
MATLAB output *z* = 0 km



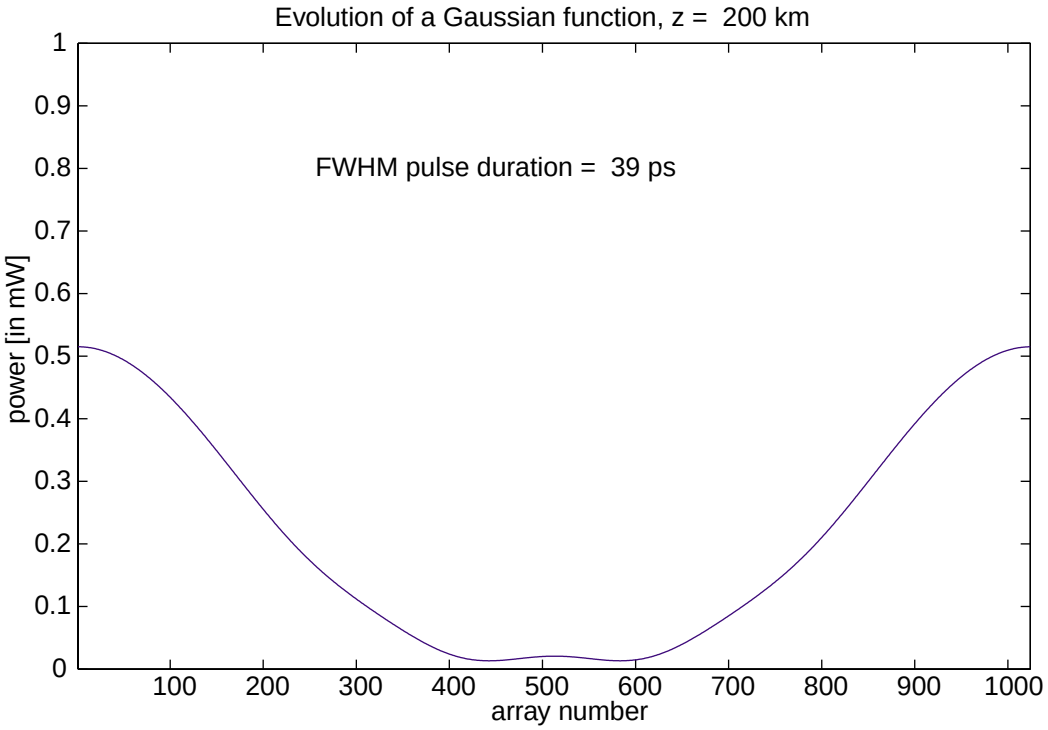


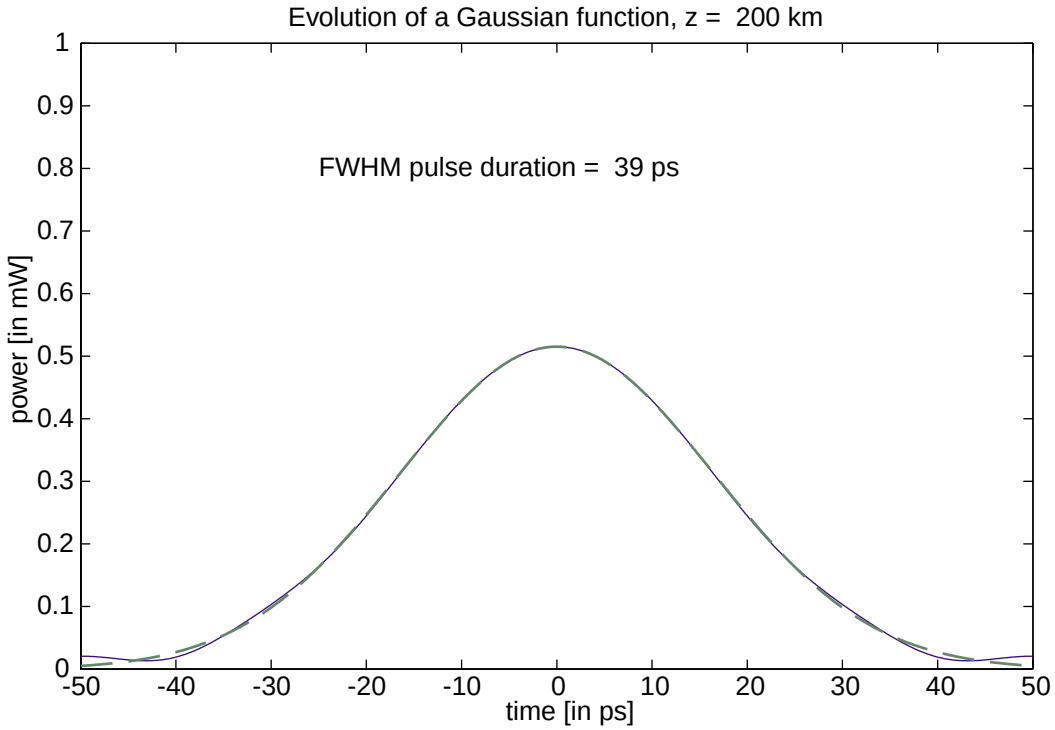
MATLAB output *z* = 100 km





MATLAB output *z* = 200 km





MATLAB output *z* = 500 km