

1D FDTD RLGC Simulations

Eric J. Lundquist
October 2008

Introduction

This report shall discuss the use of the finite-difference time-domain (FDTD) method in order to simulate the behavior of an electrical signal along a line, incorporating the RLGC (resistance, inductance, conductance, and capacitance) parameters of the line.

Methods

It was first necessary to develop a 1D FDTD code that would allow the signal to propagate through a line of which the RLGC parameters could be adjusted at any point in the line. This was accomplished by creating RLGC vectors which could be programmed to certain desired features of the line.

First, a code was developed in order to test a signal along a constant line. This code was used to develop boundary conditions for the end load, which, if matched properly, could be used to minimize reflections. This code can be observed in **Appendix A**.

In order to test the accuracy of the code, a low-pass filter was also designed and programmed, according to the following design:

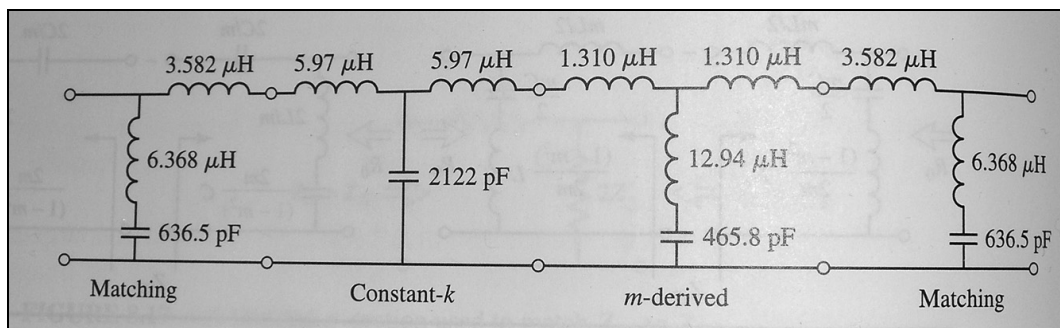


Figure 1: Low-pass composite filter. [1]

This code can be observed in **Appendix B**. Note also that measures were taken in order to program the filter design as a traditional RLGC element. This was necessary in order to combine the inductors and capacitors which are connected in series in both the matching sections and the m -derived section of the filter. The derivation took place as follows:

$$\begin{aligned}
j\omega L + \frac{1}{j\omega C} &= \frac{j^2\omega^2}{j\omega} L + \frac{1/C}{j\omega} \\
&= \frac{1/C - L\omega^2}{j\omega} \\
\frac{1}{j\omega C^c} &= \frac{1/C - L\omega^2}{j\omega} \\
C^c &= \frac{1}{1/C - L\omega^2} \\
C^c &= \frac{C}{1 - LC\omega^2} \tag{1}
\end{aligned}$$

Where C^c is the equivalent combined capacitive impedance of the LC filter sections. Additionally, note that this value is indeed frequency-dependent.

Finally, the code was verified by measuring the reflection coefficient and comparing the simulated results to those of the analytic equations. This code can be observed in **Appendix C**. The reflection coefficient can be measured as follows:

$$\Gamma_{analytic} = \frac{Z_L - Z_0}{Z_L + Z_0} \tag{2}$$

$$\Gamma_{simulated} = \frac{|V_{reflected}|}{|V_{source}|} \tag{3}$$

The reflections were measured by first running the code for a longer length of line, and the re-running the simulation for a shorter length and allowing the signal to reflect from the load. Thus, the original signal could be subtracted from the second signal in order to attain the reflected signal. The amplitude of the reflections could then be obtained.

Results

The signal propagated through the 2 MHz low-pass filter is shown below. Note that the signal is somewhat attenuated at frequencies below 2 MHz, but extremely attenuated at 2 MHz and above.

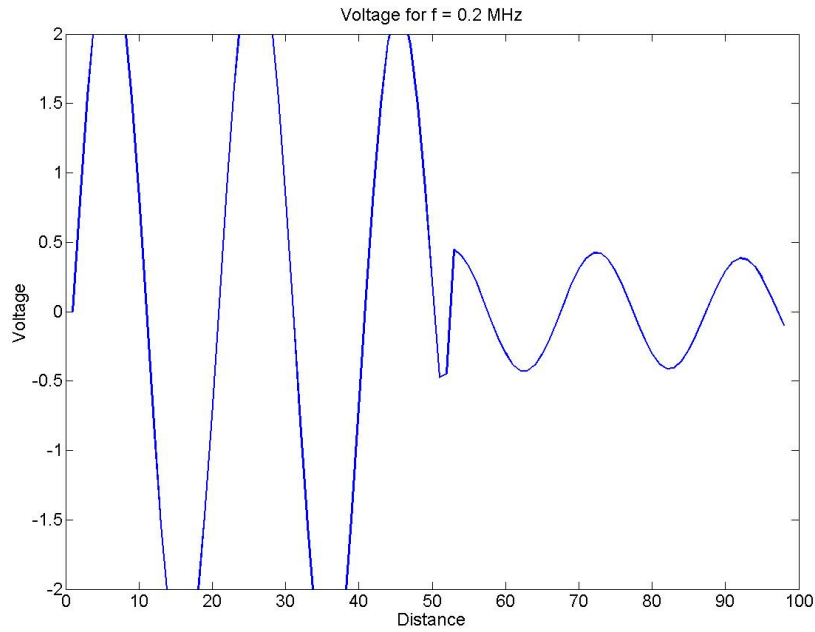


Figure 2: Voltage signal passed through low-pass filter at 0.2 MHz.

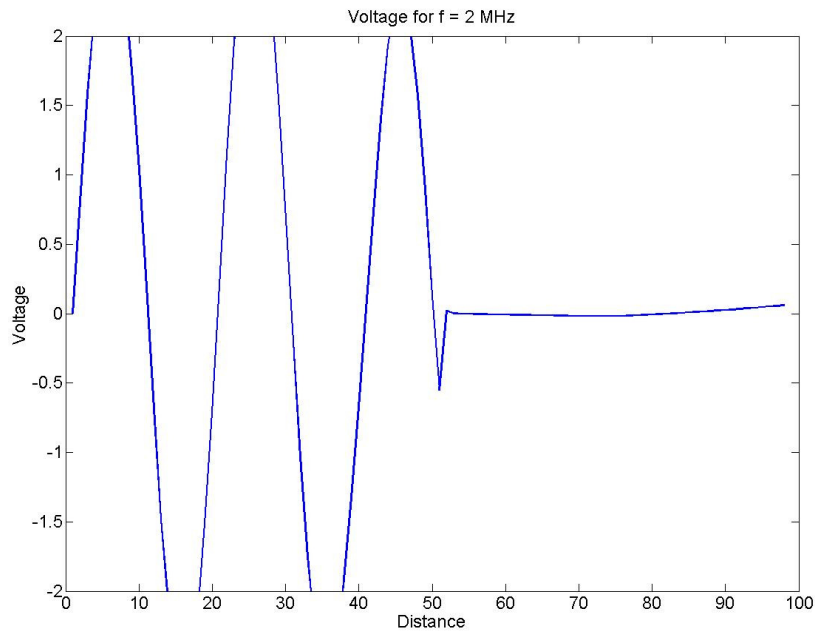


Figure 3: Voltage signal passed through low-pass filter at 2 MHz.

Results from the reflected signal are shown below. In this case, the load is matched and thus the signal is rather low in magnitude.

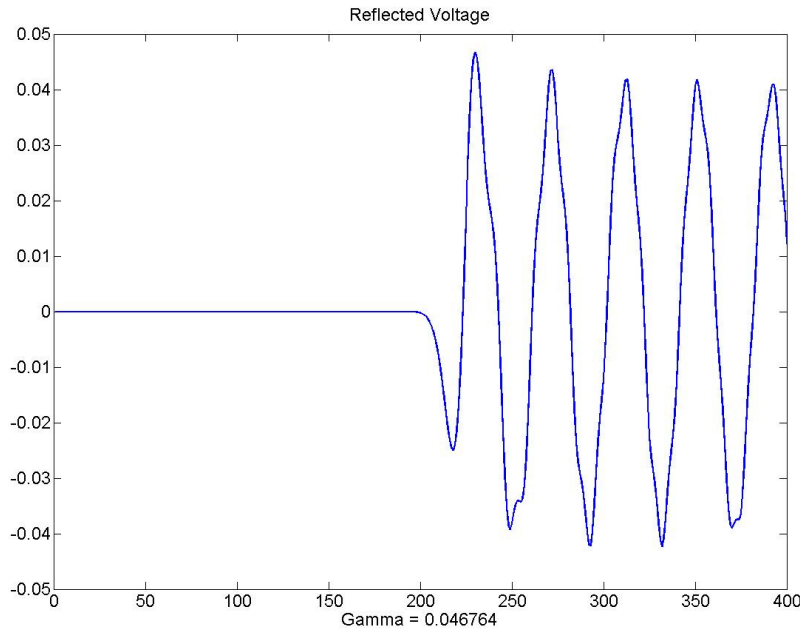


Figure 4: Reflected signal from matched load ($Z_L=Z_0$).

A comparison of the simulated and analytically computed reflection coefficients is shown below.

Table 1: Simulated and Analytically Calculated Reflection Coefficients

Z_L (in terms of Z_0)	Γ (Simulated)	Γ (Analytic)
0.25	-0.6	-0.6
0.3	-0.5381	-0.5385
0.5	-0.3315	-0.3333
0.75	-0.1405	-0.1429
1.0	0.0361	0
1.5	0.1981	0.2000
2.0	0.3330	0.3333
3.0	0.5020	0.5000
4.0	0.6031	0.6000

Conclusion

The finite-difference time-domain (FDTD) method has proven effective in the simulation the signal behavior along a line which incorporates the RLGC parameters of the line. An error of approximately 0.0125 (unitless) was found in the reflection coefficient of the simulations.

Sources

[1] Pozar, David M., *Microwave Engineering* (John Wiley and Sons: Danvers, Massachusetts [2005]), p. 389.

Appendix A

FDTD_Matching.m

```
% Eric J. Lundquist
% 1D FDTD Plots

clear all
close all
clc

% General Parameters
f = 1e9;
w = 2*pi*f;
eo = 8.854e-12;
u0 = 4e-7*pi;

% Copper Conductor
sigma_c = 5.8e7;
u_c = u0;

% Parameters for Surrounding Medium
medium = 'air' ; % enter air, teflon, or sea water
switch medium
    case 'air'
        sigma = 0 ;
        E_r = 1 ;
        u_r = u0 ;
    case 'teflon'
        sigma = 0 ;
        E_r = 2.1 ;
        u_r = u0 ;
    case 'sea water'
        sigma = 5 ;
        E_r = 81 ;
        u_r = u0 ;
    otherwise
        warning('Surrounding medium has not been specified! Air selected by default.')
        sigma = 0 ;
        E_r = 1 ;
        u_r = u0 ;
end

maxT = 400 ; % maximum number of time steps (2*maxZ before wave hits the end of grid)
maxZ = 100 ; % cells in the transmission line
z = maxZ-1 ;

% Compute Constants

cable = 3 ;
switch cable
    case 1 % Measured 3300 Cable
        R0 = 3.6996;
        L0 = 2.7630e-007 ;
        G0 = 4.5602e-004 ;
        C0 = 9.0846e-011 ;
    case 2 % RG58 Coaxial Cable
        a = 0.445e-3 ;
        b = 1.765e-3 ;
        rs = sqrt(pi*f*u_c/sigma_c) ;
        R0 = (rs/(2*pi))*(1/a + 1/b) ;
        L0 = (u_r/(2*pi))*log(b/a) ;
        G0 = (2*pi*sigma)/log(b/a) ;
        C0 = (2*pi*E_r*eo)/log(b/a) ;
    case 3 % Lossless Line
        R0 = 0 ;
        L0 = 1e-6 ;
        G0 = 0 ;
        C0 = L0/2500 ; % 2500 = (50 ohms)^2
    otherwise
        warning('No cable selected! Lossless line selected by default.')
        R0 = 0 ;
```

```

        L0 = 1e-6 ;
        G0 = 0 ;
        C0 = L0/2500 ; % 2500 = (50 ohms)^2
end

Z0 = sqrt(L0/C0)
R(1:maxZ) = R0 ;
L(1:maxZ) = L0 ;
G(1:maxZ) = G0 ;
C(1:maxZ) = C0 ;

% Compute Constants for End Terminal

loadZ = maxZ-1;
R(loadZ)=0;
L(loadZ)=0;
G(loadZ)=Z0;
C(loadZ)=0;

% Constants computed in previous section
wire_cell = 10 ; % number of cell where velocity of propagation and lambda will be
calculated (should be on wire segment)
gamma = sqrt((R+j.*w.*L).*(G+j.*w.*C)) ;
alpha = real(gamma) ;
beta = imag(gamma) ;
vp = w./beta ;
lambda = vp./f ;
dz = lambda(wire_cell)/20 ;
dt = 0.5*dz/vp(wire_cell) ;

% Additional Constants
A = -1./(dz.*(R./2 + L./dt));
B = -1.*((R./2 - L./dt)./(R./2 + L./dt));
D = -1./(dz.*(G./2 + C./dt));
E = -1.*(G./2 - C./dt)./(G./2+C./dt);

% Initialize voltage and current values of transmission line.
V = zeros(1,maxZ) ;
I = zeros(1,maxZ) ;
ttitle = sprintf(['Voltage Signal \n\n time_T_O_T_A_L = ' num2str(maxT*dt) ' seconds']) ;

% FDTD loops
for n=1:maxT;
    V(1) = sin(2*pi*f*n*dt); % sine wave source
    for k=2:maxZ; % find voltage everywhere on the line
        V(k)= D(k)*(I(k)-I(k-1))+E(k)*V(k);
    end
    for k=1:maxZ-1; % find current everywhere on the line
        I(k)= A(k)*(V(k+1)-V(k))+B(k)*I(k);
    end
    I(loadZ-1) = V(loadZ-1)/Z0 ;
    %V(loadZ) = I(loadZ-1)*Z0;
plot(V) % plot the voltage all along the line at time
N
ttitle = ['Voltage for n = ' int2str(maxT) ' Time Steps'] ;
title(ttitle)
xlabel('Distance')
ylabel('Voltage')
axis([0 maxZ -2 2]) % control the axis for uniform pictures
pause(.0001); % give the program time to plot to screen
end

```

Appendix B

FDTD_Filter.m

```
% Eric J. Lundquist
% 1D FDTD Plots
% 2 MHz Low-pass Composite Filter Design

clear all
close all
clc

set(0,'DefaultAxesFontSize',18)
set(0,'DefaultLineLineWidth',2)
set(0,'DefaultTextFontSize',18)

movieon = 1 ; % enter 1 to create movie (turning on movie does slow down simulation)

% General Parameters
f = 2e6;
w = 2*pi*f;
eo = 8.854e-12;
u0 = 4e-7*pi;

% Copper Conductor
sigma_c = 5.8e7;
u_c = u0;

% Parameters for Surrounding Medium
medium = 'air' ; % enter air, teflon, or sea water
switch medium
    case 'air'
        sigma = 0 ;
        E_r = 1 ;
        u_r = u0 ;
    case 'teflon'
        sigma = 0 ;
        E_r = 2.1 ;
        u_r = u0 ;
    case 'sea water'
        sigma = 5 ;
        E_r = 81 ;
        u_r = u0 ;
    otherwise
        warning('Surrounding medium has not been specified! Air selected by default.')
        sigma = 0 ;
        E_r = 1 ;
        u_r = u0 ;
end

maxT = 300 ; % maximum number of time steps (2*maxZ before wave hits the end of grid)
maxZ = 100 ; % cells in the transmission line

% Compute Constants
cable = 3 ;
switch cable
    case 1 % Measured 3300 Cable
        R0 = 3.6996;
        L0 = 2.7630e-007 ;
        G0 = 4.5602e-004 ;
        C0 = 9.0846e-011 ;
    case 2 % RG58 Coaxial Cable
        a = 0.445e-3 ;
        b = 1.765e-3 ;
        rs = sqrt(pi*f*u_c/sigma_c) ;
        R0 = (rs/(2*pi))*(1/a + 1/b) ;
        L0 = (u_r/(2*pi))*log(b/a) ;
        G0 = (2*pi*sigma)/log(b/a) ;
        C0 = (2*pi*E_r*eo)/log(b/a) ;
    case 3 % Lossless Line
        R0 = 0 ;
```

```

        L0 = 1e-6 ;
        G0 = 0 ;
        C0 = L0/(75^2) ; % filter is designed for a 75 ohm line
    otherwise
        warning('No cable selected! Lossless line selected by default.')
        R0 = 0 ;
        L0 = 1e-6 ;
        G0 = 0 ;
        C0 = L0/2500 ; % 2500 = (50 ohms)^2
end

Z0 = sqrt(L0/C0)

R(1:maxZ) = R0 ;
L(1:maxZ) = L0 ;
G(1:maxZ) = G0 ;
C(1:maxZ) = C0 ;

% Filter Design

c1=636.5e-12;
l1=6.368e-6;
C(50) = c1/(1-l1*c1*w^2);

L(51) = 3.582e-6 + 5.97e-6 ;
C(51) = 2122e-12;

c2=465.8e-12;
l2=12.94e-6;
L(52) = 5.97e-6 + 1.31e-6 ;
C(52) = c2/(1-l2*c2*w^2);

L(53) = 1.31e-6 + 3.582e-12 ;
C(53) = C(50);

% Compute Constants for End Terminal
loadZ = maxZ-1;
R(loadZ)=0;
L(loadZ)=0;
G(loadZ)=Z0;
C(loadZ)=0;

% Constants computed in previous section
wire_cell = 10 ; % number of cell where velocity of propagation and lambda will be
                % calculated (should be on wire segment)
gamma = sqrt((R+j.*w.*L).*(G+j.*w.*C)) ;
alpha = real(gamma) ;
beta = imag(gamma) ;
vp = w./beta ;
lambda = vp./f ;
dz = lambda(wire_cell)/20 ;
dt = 0.5*dz/vp(wire_cell) ;

% Additional Constants
A = -1./(dz.*(R./2 + L./dt));
B = -1.*((R./2 - L./dt)./(R./2 + L./dt));
D = -1./(dz.*(G./2 + C./dt));
E = -1.*(G./2 - C./dt)./(G./2+C./dt);

% Initialize voltage and current values of transmission line.
V = zeros(1,maxZ) ;
I = zeros(1,maxZ) ;
%title = sprintf(['Voltage Signal \n\n time_T_O_T_A_L = ' num2str(maxT*dt) ' seconds'])
;

% FDTD loops
for n=1:maxT;
    V(1) = sin(2*pi*f*n*dt); % sine wave source
    for k=2:maxZ; % find voltage everywhere on the line
        V(k)= D(k)*(I(k)-I(k-1))+E(k)*V(k);
    end
    for k=1:maxZ-1; % find current everywhere on the line
        I(k)= A(k)*(V(k+1)-V(k))+B(k)*I(k);
    end
end

```

```

end
I(loadZ-1) = V(loadZ-1)/Z0 ;
if movieon==1
    M(n)=getframe;
end
plot(V) % plot the voltage all along the line at time
N
tttitle = ['Voltage for f = ' num2str(f/1e6) ' MHz' ] ;
title(tttitle)
xlabel('Distance')
ylabel('Voltage')
axis([0 maxZ -2 2]) % control the axis for uniform pictures
pause(.0001); % give the program time to plot to screen
end

if movieon==1
    movie2avi(M,'2 MHz FDTD Filter');
end

```

Appendix C

FDTD_Refl.m (Reflection Coefficient Simulations)

```
% Eric J. Lundquist
% 1D FDTD Plots

clear all
close all
clc

% General Parameters
f = 1e9;
w = 2*pi*f;
eo = 8.854e-12;
u0 = 4e-7*pi;

% Copper Conductor
sigma_c = 5.8e7;
u_c = u0;

% Parameters for Surrounding Medium
medium = 'air' ; % enter air, teflon, or sea water
switch medium
case 'air'
    sigma = 0 ;
    E_r = 1 ;
    u_r = u0 ;
case 'teflon'
    sigma = 0 ;
    E_r = 2.1 ;
    u_r = u0 ;
case 'sea water'
    sigma = 5 ;
    E_r = 81 ;
    u_r = u0 ;
otherwise
    warning('Surrounding medium has not been specified! Air selected by default.')
    sigma = 0 ;
    E_r = 1 ;
    u_r = u0 ;
end

maxT = 400 ; % maximum number of time steps (2*maxZ before wave hits the end of grid)
maxZ = 400 ; % cells in the transmission line

% Compute Constants

cable = 3 ;
switch cable
case 1 % Measured 3300 Cable
    R0 = 3.6996;
    L0 = 2.7630e-007 ;
    G0 = 4.5602e-004 ;
    C0 = 9.0846e-011 ;
case 2 % RG58 Coaxial Cable
    a = 0.445e-3 ;
    b = 1.765e-3 ;
    rs = sqrt(pi*f*u_c/sigma_c) ;
    R0 = (rs/(2*pi))*(1/a + 1/b) ;
    L0 = (u_r/(2*pi))*log(b/a) ;
    G0 = (2*pi*sigma)/log(b/a) ;
    C0 = (2*pi*E_r*eo)/log(b/a) ;
case 3 % Lossless Line
    R0 = 0 ;
    L0 = 1e-6 ;
    G0 = 0 ;
    C0 = L0/2500 ; % 2500 = (50 ohms)^2
otherwise
    warning('No cable selected! Lossless line selected by default.')
    R0 = 0 ;
    L0 = 1e-6 ;
```

```

        G0 = 0 ;
        C0 = L0/2500 ; % 2500 = (50 ohms)^2
end

Z0 = sqrt(L0/C0)
R(1:maxZ) = R0 ;
L(1:maxZ) = L0 ;
G(1:maxZ) = G0 ;
C(1:maxZ) = C0 ;

% Constants computed in previous section
wire_cell = 10 ; % number of cell where velocity of propagation and lambda will be
calculated (should be on wire segment)
gamma = sqrt((R+j.*w.*L).*(G+j.*w.*C)) ;
alpha = real(gamma) ;
beta = imag(gamma) ;
vp = w./beta ;
lambda = vp./f ;
dz = lambda(wire_cell)/20 ;
dt = 0.5*dz/vp(wire_cell) ;

% Additional Constants
A = -1./(dz.*(R./2 + L./dt));
B = -1.*((R./2 - L./dt)./(R./2 + L./dt));
D = -1./(dz.*(G./2 + C./dt));
E = -1.*(G./2 - C./dt)./(G./2+C./dt);

% Initialize voltage and current values of transmission line
maxZ=200;
maxT=400;
V = zeros(1,maxZ) ;
I = zeros(1,maxZ) ;

% FDTD loops
for n=1:maxT;
    V(1) = sin(2*pi*f*n*dt); % sine wave source
    for k=2:maxZ; % find voltage everywhere on the line
        V(k)= D(k)*(I(k)-I(k-1))+E(k)*V(k);
    end
    for k=1:maxZ-1; % find current everywhere on the line
        I(k)= A(k)*(V(k+1)-V(k))+B(k)*I(k);
    end
    Vo(n)=V(90);
end

% Initialize voltage and current values of transmission line
maxZ=100;
maxT=400;
V = zeros(1,maxZ) ;
I = zeros(1,maxZ) ;

% Compute Constants for End Terminal
ZL = Z0
loadZ = maxZ-1;
R(loadZ)=0;
L(loadZ)=0;
G(loadZ)=ZL;
C(loadZ)=0;

% FDTD loops
for n=1:maxT;
    V(1) = sin(2*pi*f*n*dt); % sine wave source
    for k=2:maxZ; % find voltage everywhere on the line
        V(k)= D(k)*(I(k)-I(k-1))+E(k)*V(k);
    end
    V(loadZ+1) = V(loadZ);
    for k=1:maxZ-1; % find current everywhere on the line
        I(k)= A(k)*(V(k+1)-V(k))+B(k)*I(k);
    end
    I(loadZ) = V(loadZ)/ZL ;
    Vr(n)=V(90);
% plot(V)
% tttitle = ['Voltage for n = ' int2str(maxT) ' Time Steps'] ;

```

```

% title(tttitle)
% xlabel('Distance')
% ylabel('Voltage')
% axis([0 maxZ -2 2])           % control the axis for uniform pictures
% pause(.0001);               % give the program time to plot to screen
end

Vrefl = Vr-Vo;
ReflCo = max(Vrefl); % does not need to be divided by max(Vo) because max(Vo)=1 for this
source
tvalue = sprintf(['Gamma = ' num2str(ReflCo)]) ;

figure(2)
plot(Vrefl)
title('Reflected Voltage')
xlabel(tvalue)

if ZL>=Z0
    Simulated = ReflCo-.0108 % ~0.0125 Error
else
    Simulated = -(ReflCo-.0108) % ~0.0125 Error
end
Analytic = (ZL-Z0)/(ZL+Z0)

```