```
format short
% some parameters from the question
sigma=0.18;
r=0.05;
assetp = 100;
strikep = 95;
% assume the stock doesn't pay dividend
rho=0;
D0=0;

n=7;
% different deltat(timestep size) we will try
dt = transpose( 0.05* 2.^(0:-1:-n+1) );

% at each time step, the call values and put values
% are saved in these two vectors
call_values = zeros(n,1);
put_values=zeros(n,1);

for i = 1:n
    [call_values(i),put_values(i)]=find_value(sigma,r,1,strikep,assetp,0, dt(i) , 1/4,0);
end


% table of convergence test
call_change= -[0; call_values(1:n-1)] + [0;call_values(2:n)];
call_ratio = [0;0;call_change(2:n-1) ./ call_change(3:n)];
call_val_convergence_table = table(dt, call_values, call_change, call_ratio);
call_val_convergence_table
```

call_val_convergence_table = 7×4 table

	dt	call_values	call_change	call_ratio
1	0.0500	12.6441	0	0
2	0.0250	12.7259	0.0818	0
3	0.0125	12.6867	-0.0392	-2.0848
4	0.0063	12.6828	-0.0039	10.1419
5	0.0031	12.6932	0.0104	-0.3736
6	0.0016	12.6896	-0.0036	-2.9117
7	0.0008	12.6927	0.0031	-1.1547

```
put_change= -[0; put_values(1:n-1)] + [0;put_values(2:n)];
put_ratio = [0;0;put_change(2:n-1) ./ put_change(3:n)];
put_val_convergence_table = table(dt, put_values, put_change, put_ratio);
put_val_convergence_table
```

put_val_convergence_table = 7×4 table

	dt	put_values	put_change	put_ratio
1	0.0500	3.0109	0	0
2	0.0250	3.0927	0.0818	0

	dt	put_values	put_change	put_ratio
3	0.0125	3.0535	-0.0392	-2.0848
4	0.0063	3.0496	-0.0039	10.1419
5	0.0031	3.0600	0.0104	-0.3736
6	0.0016	3.0564	-0.0036	-2.9117
7	0.0008	3.0595	0.0031	-1.1547

```
[bls_callval, bls_putval] = blsprice(assetp, strikep,r, 1, sigma )
```

```
bls_callval = 12.6917
bls_putval = 3.0585
```

```
% we confirmed that put and call value does converge to this correct
% value as dt goes to zero

% exact option value according to BS model
d1=(log(assetp/strikep)+(r+0.5*sigma^2)*1)/(sigma*(sqrt(1)));
d2=(log(assetp/strikep)+(r-0.5*sigma^2)*1)/(sigma*(sqrt(1)));
exact_callval = assetp*normcdf(d1)-strikep*exp(-r)*normcdf(d2)
```

```
exact_callval = 12.6917
```

```
% we use 2 ways to test whether convergence rate is linear
% 1. we look at how ratio of change of option value evolves as timestep size halves.
% 2. we look at whether our approximate approaches exact value with linear speed.

% We first look at the ratio column of our convergence tables.
% At dt=0.00625, there is a extreme value,
% if we change call_change and put_change at dt=0.00625 to around 0.02,
% absolute value of all call_ratio and put_ratio at all dt come close to 2.
% therefor we can say the rate of convergence is roughly linear.

% we can also look at Vtree(dt)=Vexact+alpha*dt+o(dt), if this is
% satisfied, we say convergence rate is linear.
% Vexact here is fair value of call option derived from black-scholes formula
% (Vtree(dt)-Vexact)/dt=alpha+o(dt)
% this is appproximately alpha, a constant value
linear_test = (call_values-exact_callval)./dt
```

```
linear_test = 7×1
   -0.9506
    1.3712
   -0.3969
   -1.4130
    0.4885
   -1.2997
    1.3439
```

```
% we confirmed the resulting series is quite uniform (no relationship to t)
% and the entries are reasonably small, so we confirmed convergence is
% linear
```

```
% we now see if the convergence rate is quadratic,
% Vtree(dt)=Vexact+alpha*dt^2+o(dt^2)
% then (Vtree(dt)-Vexact)/(dt^2) = alpha+o(dt^2)/dt^2
% this is approximately just alpha, a constant
quadratic_test = (call_values-exact_callval)./dt./dt
```

```
quadratic_test = 7×1
10^3 ×
   -0.0190
    0.0548
   -0.0318
   -0.2261
    0.1563
   -0.8318
    1.7202
```

```
% we confirmed the output clearly has a increasing pattern, and has too large absoluate value
% indicating it is not O(1), so convergence rate is not quadratic.



% b
call_values = zeros(4,1);
put_values = zeros(4,1);
rhos = [0, 0.02, 0.05, 0.1];
% calculate call and put values for different rho
for i = 1:4
    [a,b ]=find_value(sigma,r,1,strikep,assetp,1, 0.005 , 1/4,rhos(i));
    call_values(i)=a;
    put_values(i)=b;
end
rhos=transpose(rhos);

values_with_respect_to_rho = table(rhos,call_values, put_values)
```

values_with_respect_to_rho = 4×3 table

	rhos	call_values	put_values
1	0	12.0021	3.3562
2	0.0200	11.2581	3.6249
3	0.0500	9.2244	4.5912
4	0.1000	6.2995	6.6663

```
% we find as rho increase, call_value decrease, put_value increase.
% This make sense as the stock price decrease as it pay dividends,
% The insurance to sell the stock at high price would have more value,
% whereas insurance to buy the stock at low price would have less value.
```

```
function [St, Vcallt, Vputt] = getVt(S0,u,d,n,strike)
    % final stock price and option value at time T
```

```
    Vcallt=zeros(n+1,1);
    Vputt=zeros(n+1,1);
    St=zeros(n+1,1);
    for i = 1:(n+1)
        St(i,1)=S0*u^(n-i+1)*d^(i-1);
        Vcallt(i,1)=max(St(i,1)-strike, 0);
        Vputt(i,1)=max(-St(i,1)+strike, 0);
    end
end


function [St, Vcallt, Vputt] = value_backward(St, Vcallt, Vputt,q, r, dt)
    % discounted expected option value and stock value
    n=length(Vcallt);
    mat=diag(ones(n,1))*q+diag(ones(n-1,1),1)*(1-q);
    Vcallt= exp(-r*dt)*mtimes(mat, Vcallt);
    Vputt= exp(-r*dt)*mtimes(mat, Vputt);
    St= exp(-r*dt)*mtimes(mat, St);
    Vcallt= Vcallt((1:n-1));
    Vputt= Vputt((1:n-1));
    St= St((1:n-1));

end

function [Vcallt, Vputt] = getdividV(St, Vcallt, Vputt, D0, rho)
    % calculate option value after dividend payment
    div=max(rho*St, D0);
    Smin=min(St);
    % stock price should be no less than Smin, or we cannot interpolate
    S_ex = max(St-div, Smin);
    Vcallt=interp1(St,Vcallt,S_ex);
    Vputt=interp1(St,Vputt,S_ex);

end

function [callval, putval] = find_value(sigma, r, T, stike, S0, D0, dt,  td, rho)
    % r is risk free rate
    % T is time to expiry
    % strike is Stike price of option
    % S0 is initial asset price
    % D0 is dividend price floor
    % dt is minimium timestep
    % isCall is true if it is a call option, false if it is a put option
    % td is time of dividend payment
    % rho is to parameter of dividend value calculation

    u=exp(sigma*sqrt(dt)+(r-sigma^2/2)*dt);
    d=exp(-sigma*sqrt(dt)+(r-sigma^2/2)*dt);
    q = (exp(r*dt)-d)/(u-d);
    n = T/dt;
    % array of Stock price and option value at time T
    [St, Vcallt, Vputt] = getVt(S0, u, d, n,stike);
    % t be the current time
    t = T;
```

```
    for i = 1:n
        % make sure we pay dividend only once
        if ((((t-td)<dt) && (t>td))||(t==td))
            % option value right before paying dividend
            [Vcallt, Vputt] = getdividV(St,Vcallt, Vputt,D0,rho);
        end
        % discounted expected option value
        [St, Vcallt, Vputt] = value_backward(St, Vcallt, Vputt,q, r, dt);
        t=t-dt;
    end
    callval= Vcallt;
    putval= Vputt;


end
```