作业：二维声波方程有限差分数值模拟（acoustic wave propagation in 2D）

1. 算法原理介绍：

选用x-z平面（其中z表示深度），均匀介质下的二维声波方程有如下形式：

为了进行有限差分计算，需要将离散化为

根据有限差分原理，可以利用三个点的值来计算二阶导数，故可写作如下形式：

选用常规的二维网格结构，故

源函数选择雷克子波，

本程序中对雷克子波进行了偏移，偏移量为，即震源函数表达成：

为保证程序运行稳定，需要设置一系列的初始条件：

1. 声波在介质中的传播速度，设为3000m/s
2. 空间离散间隔设为5m，时间离散间隔设为0.001s，
3. ，取计算得采样频率，所用雷克子波的主频为，为20HZ
4. 空间上的差分网格点数设为200，时间上的离散点数设为500
5. 程序代码：

%设计一个函数对给定不同的参数都可以生成相应的二维声波响应

p = acoustic2D(20,500,200,200,3000,100,100,5,5,0.001);

%函数主体

function p1 = acoustic2D( f, nt, nx, nz, c, sx, sz, dx, dz, dt )

%对变量进行初始化，提高计算的效率

source = zeros(nt,1);

pnew = zeros(nx,nz);

pold = zeros(nx,nz);

p = zeros(nx, nz);

dx2 = zeros(nx, nz);

dz2 = zeros(nx, nz);

for i = 1:nt

source(i)=(1-(2\*pi\*f\*(i\*dt-1/f))^2)\*exp(-(pi\*f\*(i\*dt-1/f))^2); %随时间的推移，源的幅值会发生变化

for j = 2:nx-1

for k = 2:nz-1

dx2(j,k) = (p(j+1,k)-2\*p(j,k)+p(j-1,k))/(dx^2);%space derivative

dz2(j,k) = (p(j,k+1)-2\*p(j,k)+p(j,k-1))/(dz^2);

pnew(j,k) = c^2\*(dt^2)\*(dx2(j,k)+dz2(j,k))+2\*p(j,k)-pold(j,k);%迭代计算

end

end

pnew(sx,sz) = pnew(sx,sz)+(dt^2)\*source(i);% add source term : sx means the x position of source,the same as sz

pold = p;%将现在的P值赋给下一循环的pold

p = pnew;%将pnew的值赋给下一循环的p

imagesc(p);%对p这一二维矩阵中相应元素的值按照颜色深浅显示出来(实时)

axis equal;

title([ 'FD for 2D acousic wave propagation', newline,'t = ',num2str(i\*dt),'s']);

xlabel('X distance/m');

ylabel('Z distance/m');

drawnow

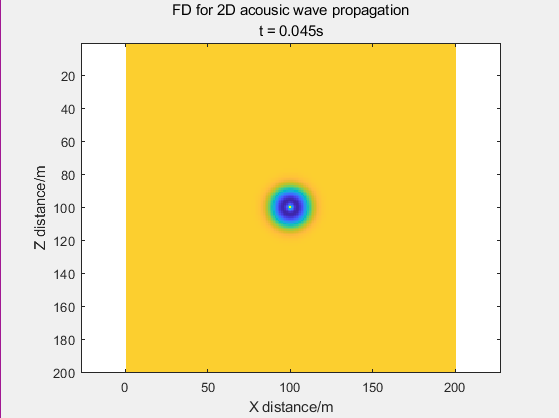
p1 = p;

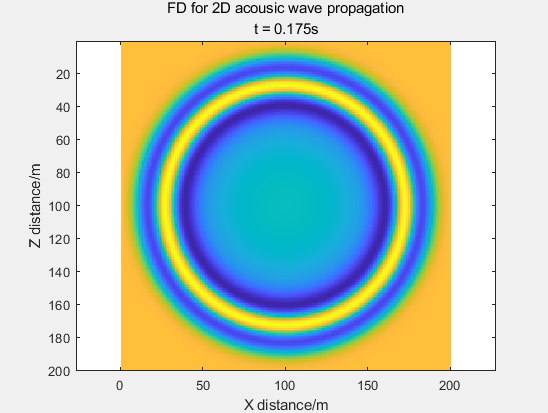
end

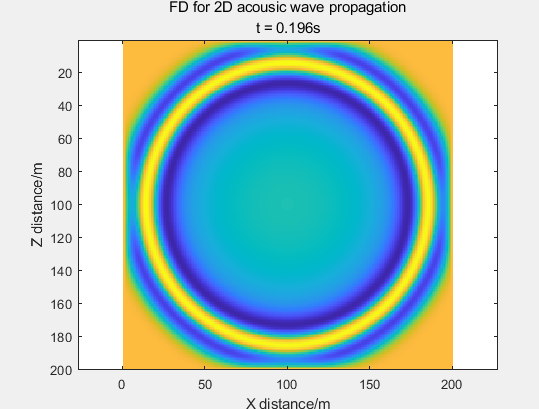
end

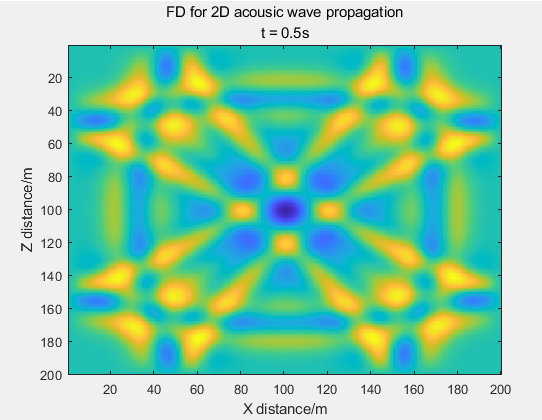
1. 结果分析

在二维平面的正中央设置源，随时间的变化，逐渐向外扩散，当波传播到边界上时，由于未设置吸收边界条件，会发生反射。以下分别抓取了四个时刻的传播情况，如图所示。









1. 补充

发现不同的编码方式的计算效率有区别，如下采用了三段循环，先计算所有的值，再截取不同的时刻进行成像。如下的编码方式比较耗时，历时33.286s。

程序代码如下：

%二维声波方程有限差分法数值模拟

clc;

clear;

tic

%分别设置空间和时间的采样点数

nx=500;

ny=500;

nt=1000;

dx=10; %x方向和y方向的步长相同，用dx代替

dt=0.001; %时间步长

c=3000; %声波在此介质中传播速度设置为3000m/s

F=20; %震源主频设为20HZ

A=(dt\*c)^2/dx^2;%计算时的常量系数

p=zeros(nx,ny,nt);%初始化数组，提高计算效率

%计算P值的循环体

for k = 2:nt-1

for i = 3:nx-1

for j = 3:ny-1

if i==250 && j==250

p(i,j,k) = (1-2\*(pi\*F\*(k\*dt-1/F))^2)\*exp(-(pi\*F\*(k\*dt-1/F))^2);%在(250m，250m)处，即在二维平面的正中央选用雷克子波作为震源

else

p(i,j,k+1)=A\*(p(i+1,j,k)+p(i-1,j,k)+p(i,j+1,k)+p(i,j-1,k)-4\*p(i,j,k))-p(i,j,k-1)+2\*p(i,j,k)+dt^2\*(1-2\*(pi\*F\*(k\*dt-1/F))^2)\*exp(-(pi\*F\*(k\*dt-1/F))^2);

end

end

end

end

%将计算的值进行成像

for k=1:10:nt

pcolor(p(:,:,k));

shading interp;

colormap('copper');

axis equal;%横纵坐标设置成等长刻度

axis([0,500,0,500]);

set(gca,'Ydir','reverse'); %改变坐标轴的方向

xlabel('x/m');

ylabel('y/m');

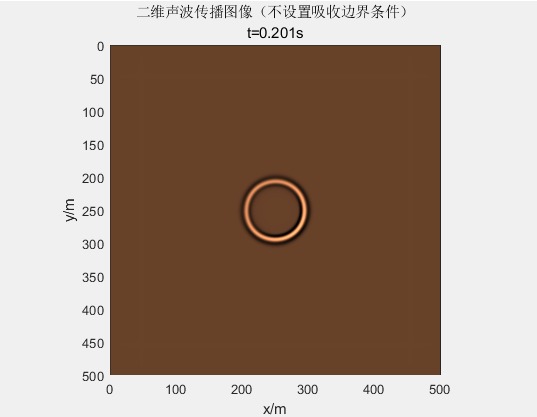
title(['二维声波传播图像（不设置吸收边界条件）',newline,'t=',num2str(k\*dt),'s']);

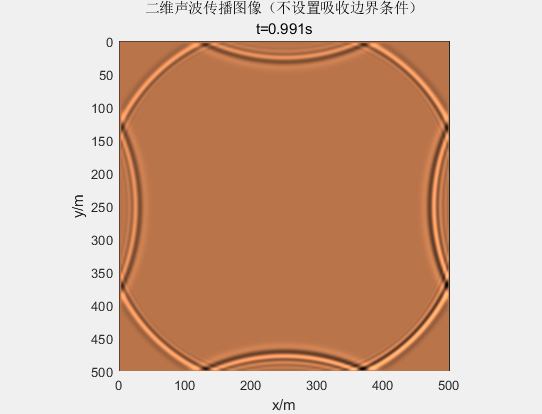
image=getframe(gcf); % 捕获画面

end

toc

其两个时刻的抓取图像如下：





clear

clc

nx=200;nz=200; nt=500;

dx=10; %x方向和z方向的网格长度

dt=0.001; %时间离散间隔

v=3000; %波速

f=20; %震源主频

A=dt\*v/dx;%差分方程系数

p=zeros(nx,nz,nt);%初始化p

for k=2:nt

for i=2:nx-1

for j=2:nz-1

if i==100 && j==100

p(i,j,k)= (1-2\*(pi\*f\*dt\*(k\*dt-1/f))^2)\*exp(-(pi\*f\*dt\*(k\*dt-1/f))^2);

else

p(1,j,k+1)=(2-2\*A-A^2)\*p(1,j,k)+2\*A\*(1+A)\*p(2,j,k)-A^2\*p(3,j,k)+(2\*A-1)\*p(1,j,k-1)-2\*A\*p(2,j,k-1);

p(nx,j,k+1)=(2-2\*A-A^2)\*p(nx,j,k)+2\*A\*(1+A)\*p(nx-1,j,k)-A^2\*p(nx-2,j,k)+(2\*A-1)\*p(nx,j,k-1)-2\*A\*p(nx-1,j,k-1);

p(i,1,k+1)=(2-2\*A-A^2)\*p(i,1,k)+2\*A\*(1+A)\*p(i,2,k)-A^2\*p(i,3,k)+(2\*A-1)\*p(i,1,k-1)-2\*A\*p(i,2,k-1);

p(i,nz,k+1)=(2-2\*A-A^2)\*p(i,nz,k)+2\*A\*(1+A)\*p(i,nz-1,k)-A^2\*p(i,nz-2,k)+(2\*A-1)\*p(i,nz,k-1)-2\*A\*p(i,nz-1,k-1);

p(i,j,k+1)=(1-2\*(pi\*f\*dt\*(k\*dt-1/f))^2)\*exp(-(pi\*f\*dt\*(k\*dt-1/f))^2)\*dt^2+2\*p(i,j,k)-p(i,j,k-1)+A^2\*(p(i+1,j,k)+p(i-1,j,k)+p(i,j+1,k)+p(i,j-1,k)-4\*p(i,j,k));

end

end

end

end

for k=1:10:nt

pcolor(p(:,:,k))

shading interp;

colormap('turbo');

colorbar;

axis equal;

set(gca,'Ydir','reverse');

xlabel('x');

ylabel('z');

title(['二维声波方程(吸收边界)',newline,'t=',num2str(k\*dt),'s']);

f=getframe(gcf); %捕获画面

F=frame2im(f);%从f中返回RGB图像

[F,cmap] = rgb2ind(F,256); %将RGB图像转换为包含256种颜色的索引图, %F为索引矩阵,cmap为颜色图

filename='二维声波方程有限差分.gif';

if k==1

imwrite(F,cmap,filename,'gif', 'Loopcount',inf,'DelayTime',0.05);

else

imwrite(F,cmap,filename,'gif','WriteMode','append','DelayTime',0.1);

end

end