%%%%%% Euler Method and Analytical Solution Ray Tracing %%%%%%

clear

close all

%%%%%%%% Create Velocity and Slowness Models %%%%%%%%

v0 = 0.5;

vf = 5;

dv = .01;

nv = floor((vf - v0)/dv);

s0\_2 = 1/v0^2;

sf\_2 = 1/vf^2;

gradSlo2 = (sf\_2 - s0\_2) / nv;

nx = nv;

nz = nv;

vals = [1:nv];

%%%%%%%% Euler's Method Implementation %%%%%%%%%

t0 = 0;

tf = 100:10:250;

nt = nv;

slope = 1:numel(tf);

d\_t = 1:numel(tf);

error = 1:numel(tf);

xPos0 = 0;

zPos0 = 0;

xPos = vals;

zPos = vals;

xDir = vals;

zDir = vals;

tVec = vals;

xPos(1) = xPos0;

zPos(1) = zPos0;

alpha=30;

gVx = 0;

gVz = gradSlo2;

xDir0 = sind(alpha);

zDir0 = cosd(alpha);

xDir(1) = xDir0;

zDir(1) = zDir0;

%%%%%%% Analytical Solution Parameters

xPlt = [1:nx];

zPlt = [1:nz];

for itf=1:numel(tf)

dt = (tf(itf) - t0)/nt;

d\_t(itf) = dt;

for ix=1:nx-1

xPos(ix+1) = xPos(ix) + xDir(ix)\*dt;

xDir(ix+1) = xDir(ix) + gVx\*dt;

zPos(ix+1) = zPos(ix) + zDir(ix)\*dt;

zDir(ix+1) = zDir(ix) + gVz\*dt;

if zPos(ix+1) < 0

zPos(ix+1) = 0;

end

end

%%%%%%%%%%% Analytical Solution %%%%%%%%%%%%%%

for it=1:nt % fill tVec with times

tVec(it) = t0 + (it-1) \* dt;

end

for it=1:nt

t = tVec(it);

xPos\_ = xPos0 + xDir0\*t + gVx \* t\*t \* 0.5;

zPos\_ = zPos0 + zDir0\*t + gVz \* t\*t \* 0.5;

if zPos\_ < 0

zPos\_ = 0;

end

xPlt(it) = xPos\_;

zPlt(it) = zPos\_;

end

diffX = xPos-xPlt;

diffZ = zPos-zPlt;

magDiff = sqrt(diffX.^2 + diffZ.^2);

maxmD = max(magDiff);

slope(itf) = maxmD/find(magDiff==maxmD);

%%%%% for comparison %%%%%

error(itf) = -gradSlo2 \* dt\*dt \* 0.5;

end

subplot(3,1,1)

scatter(d\_t,slope)

title('Error Values for Different Time Steps')

xlabel('Time Step (dt)')

ylabel('Error')

subplot(3,1,2)

scatter(d\_t,error)

title('Expected Error for First Order Scheme')

xlabel('Time Step (dt)')

ylabel('g0\*dt^2\*0.5')

subplot(3,1,3)

scatter(log(d\_t),log(slope))

title('Natural Log of Error')

xlabel('Natural Log of dt')

ylabel('Natural Log of Error')

hold off

%%%%%% Euler Method and Analytical Solution Ray Tracing %%%%%%

clear

close all

%%%%%%%% Create Velocity and Slowness Models %%%%%%%%

v0 = 0.5;

vf = 5;

dv = .01;

nv = floor((vf - v0)/dv);

s0\_2 = 1/v0^2;

sf\_2 = 1/vf^2;

gradSlo2 = (sf\_2 - s0\_2) / nv;

% slo2 = [1:nv];

%

% for is=1:nv

% slo2(is) = s0\_2 + is\*gradSlo2;

% end

%

% slo = sqrt(slo2);

nx = nv;

nz = nv;

vals = [1:nv];

%%%%%%%% Structs for Evaluating the Difference Between Solutions %%%%%%%%

allPosVec = struct('x\_Pos',vals,'z\_Pos',vals);

allPltVec = struct('x\_Plt',vals,'z\_Plt',vals);

%%%%%%%% Euler's Method Implementation %%%%%%%%%

t0 = 0;

tf = 200;

nt = nv;

dt = (tf-t0)/nt;

xPos0 = 0;

zPos0 = 0;

xPos = vals;

zPos = vals;

xDir = vals;

zDir = vals;

tVec = vals;

xPos(1) = xPos0;

zPos(1) = zPos0;

a0 = 0;

af = 90;

da = 4;

na = floor((af - a0)/da);

gVx = 0;

gVz = gradSlo2;

hold on

for a=1:na+1

alpha = a0 + (a-1)\*da;

xDir0 = sind(alpha);

zDir0 = cosd(alpha);

xDir(1) = xDir0;

zDir(1) = zDir0;

for ix=1:nx-1

xPos(ix+1) = xPos(ix) + xDir(ix)\*dt;

xDir(ix+1) = xDir(ix) + gVx\*dt;

zPos(ix+1) = zPos(ix) + zDir(ix)\*dt;

zDir(ix+1) = zDir(ix) + gVz\*dt;

if zPos(ix+1) < 0

zPos(ix+1) = 0;

end

end

allPosVec(a).x\_Pos = xPos;

allPosVec(a).z\_Pos = zPos;

plot(xPos,zPos)

end

title('Euler''s Method Ray Tracing','fontsize',16)

ylabel('Depth of Ray')

xlabel('Horizontal Distance Traveled')

%axis([0,120,0,50])

axis ij

hold off

%%%%%%%%%%% Analytical Solution %%%%%%%%%%%%%%

figure

for it=1:nt % fill tVec with times

tVec(it) = t0 + (it-1) \* dt;

end

xPlt = [1:nx];

zPlt = [1:nz];

hold on

for a=1:na+1

alpha = a0 + (a-1)\*da;

xDir0 = sind(alpha);

zDir0 = cosd(alpha);

for it=1:nt

t = tVec(it);

xPos = xPos0 + xDir0\*t + gVx \* t\*t \* 0.5;

zPos = zPos0 + zDir0\*t + gVz \* t\*t \* 0.5;

if zPos < 0

zPos = 0;

end

xPlt(it) = xPos;

zPlt(it) = zPos;

end

allPltVec(a).x\_Plt = xPlt;

allPltVec(a).z\_Plt = zPlt;

plot(xPlt,zPlt)

end

title('Analytical Solution','fontsize',16)

ylabel('Depth of Ray')

xlabel('Horizontal Distance Traveled')

%axis([0,120,0,50])

axis ij

%colorbar(1/slo)

hold off

diffX = struct('difference',[1:nv]);

diffZ = struct('difference',[1:nv]);

magDiff = struct('magDif',[1:nv]);

for ia=1:na

diffX(ia).difference = allPosVec(ia).x\_Pos - allPltVec(ia).x\_Plt;

diffZ(ia).difference = allPosVec(ia).z\_Pos - allPltVec(ia).z\_Plt;

magDiff(ia).magDif = sqrt((diffX(ia).difference).^2 + (diffZ(ia).difference).^2);

end

figure

plot(magDiff(1).magDif)

title('Magnitude of the Difference Between Rays')

%%%%%%%%%%% Euler's Variation Changing Depth of Rays for Final Plot %%%%%%%

% for ix=1:nx-1

% xPos(ix+1) = xPos(ix) + xDir(ix)\*dt;

% xDir(ix+1) = xDir(ix) + gVx\*dt;

% end

% for ix=1:nx-1

% zPos(ix+1) = zPos(ix) + zDir(ix)\*dt;

% zDir(ix+1) = zDir(ix) + gVz\*dt;

%

% if zPos(ix+1) < 0

% zPos(ix+1) = 0;

% end

% if zPos(ix+1) > 50

% zPos(ix+1:nx) = 50;

% break

% end

%

% end

%%%%%%%%%%% Analytic Variation Changing depth of Rays %%%%%%%%%%%%

% for it=1:nt

% t = tVec(it);

% xPos = xPos0 + xDir0\*t + gVx \* t\*t \* 0.5;

% xPlt(it) = xPos;

% end

% for it=1:nt

% t = tVec(it);

% zPos = zPos0 + zDir0\*t + gVz \* t\*t \* 0.5;

% if zPos < 0

% zPos = 0;

% end

% if zPos > 50

% zPlt(it:nt) = 50;

% break

% else

% zPlt(it) = zPos;

% end

% end

Euler Second Order Equation

Euler First order equation

Analytic Solution

Euler Half Steps