

hw03

Jun ZHU

April 14, 2021

1 TRAVEL TIME TABLE (FLATTENING MODEL)

1.1 FORMULA

[X, T]=Ray1D(p,α)
give p

$$X = 2 \sum_{j=1}^N Th_j \frac{p}{\eta_j}$$

$$T = 2 \sum_{j=1}^N \frac{Th_j}{\eta_j} \frac{1}{\alpha_j^2}$$

end

1.2 SUBROUTINE

```
#include <math.h>
```

```
int layerxt(float p, float h, float utop, float ubot, float *dx, float *dt)
{
    int irtr;
    float eta1, eta2, tau1, tau2, dtau, x1, x2, u1, u2, v1, v2, b;
    if (p > utop)
    {
        *dx=0.0;
        *dt=0.0;
        irtr=0;
        return irtr;
    }
    else if (h == 0.0)
    {
        *dx=0.0;
        *dt=0.0;
    }
}
```

```

        irtr=-1;
        return irtr;
    }

    u1=utop;
    u2=ubot;
    v1=1./u1;
    v2=1./u2;
    b=(v2-v1)/h;

    eta1=sqrt(u1*u1-p*p);

    if (b ==0)
    {
        *dx=h*p/eta1;
        *dt=h*u1*u1/eta1;
        irtr=1;
        return irtr;
    }

    x1=eta1/(u1*b*p);
    tau1=(log((u1+eta1)/p)-eta1/u1)/b;

    if (p > ubot)
    {
        *dx=x1;
        dtau=tau1;
        *dt=dtau+p>(*dx);
        irtr=2;
        return irtr;
    }

    irtr=1;

    eta2=sqrt(u2*u2-p*p);
    x2=eta2/(u2*b*p);
    tau2=(log((u2+eta2)/p)-eta2/u2)/b;

    *dx=x1-x2;
    dtau=tau1-tau2;

    *dt=dtau+p(*dx);

    return irtr;
}

```

1.3 MAIN FUNCTION

```
#include <stdio.h>
#include <stdlib.h>
#include "layer.h"
#include "numc.h"

#define N 100000
#define LN 130
int main(int argc, char *argv[])
{
    //read ak135
    float d, v, dep[136], h[135], vp[136];
    int i=0;
    FILE *fp;
    if ((fp=fopen("../.. / data/AK135CSV/ak135","r")) == NULL)
    {
        printf("\nerror on open the file");
        getchar();
        exit(0);
    }
    while(!feof(fp))
    {
        fscanf(fp,"%f %f",&d,&v);
        dep[i]=d;
        vp[i]=v;
        i++;
        printf("%f %f\n",d,v);
    }
    for (i=0;i<135;i++)
    {
        h[i]=dep[i+1]-dep[i];
        printf("%f\n",h[i]);
    }

    int irtr,j;
    float dx=0,dt=0,p[N],x[N]={0.0},t[N]={0.0},tau[N]={0.0};

    linspace(0.05434,0.08979,N,p);

    for (i=0;i<N;i++)
    {
        for (j=0;j<LN;j++)
        {
            irtr = layerxt(p[i],h[j],1.0/vp[j],1.0/vp[j+1],&dx,&dt);
            x[i] += dx;
        }
    }
}
```

```

        t[i] += dt;
    }
    tau[i] = t[i]-p[i]*x[i];
}
printf("param delta tt tau\n");
fp=fopen("../output/ttable.txt","w");
fprintf(fp, "param delta tt tau\n");
for(i=0;i<N;i++)
{
    printf("%f      %f      %f      %f\n",p[i],x[i]/111,t[i],tau[i])
    fprintf(fp, "%f %f      %f      %f\n",p[i],x[i]/111,t[i],tau[i])
}
fclose(fp);
return 0;
}

```

2 RAY TRACING

2.1 BENDING

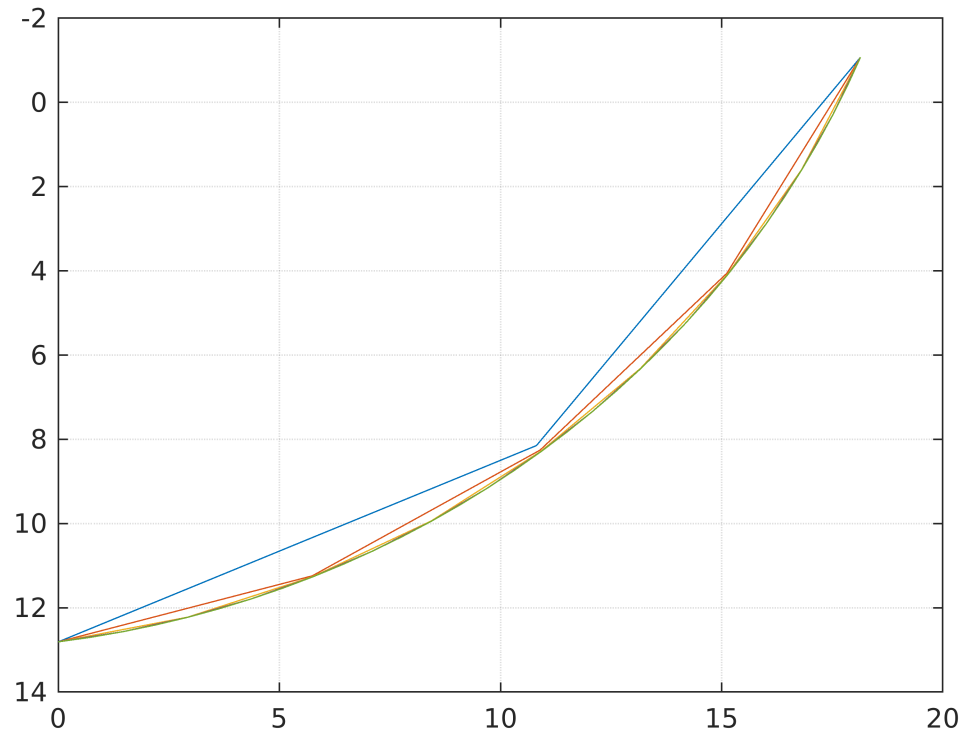


Figure 1: Ray tracing by bending method: NO RANDOM VELOCITY FIELD

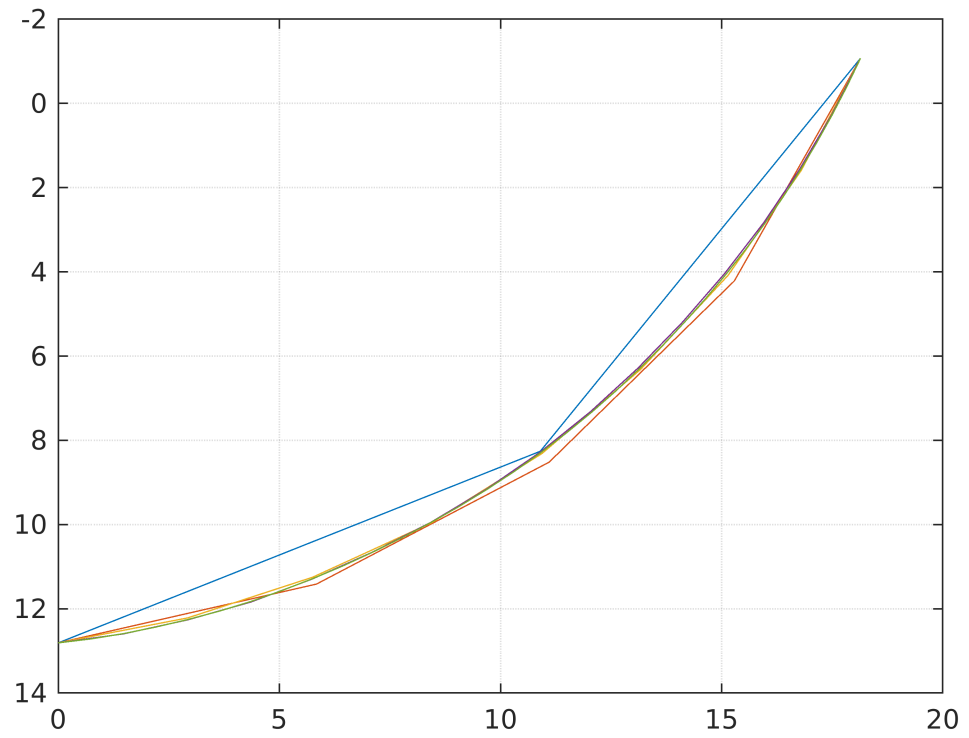


Figure 2: Ray tracing by bending method: 10% RANDOM VELOCITY FIELD LOADED

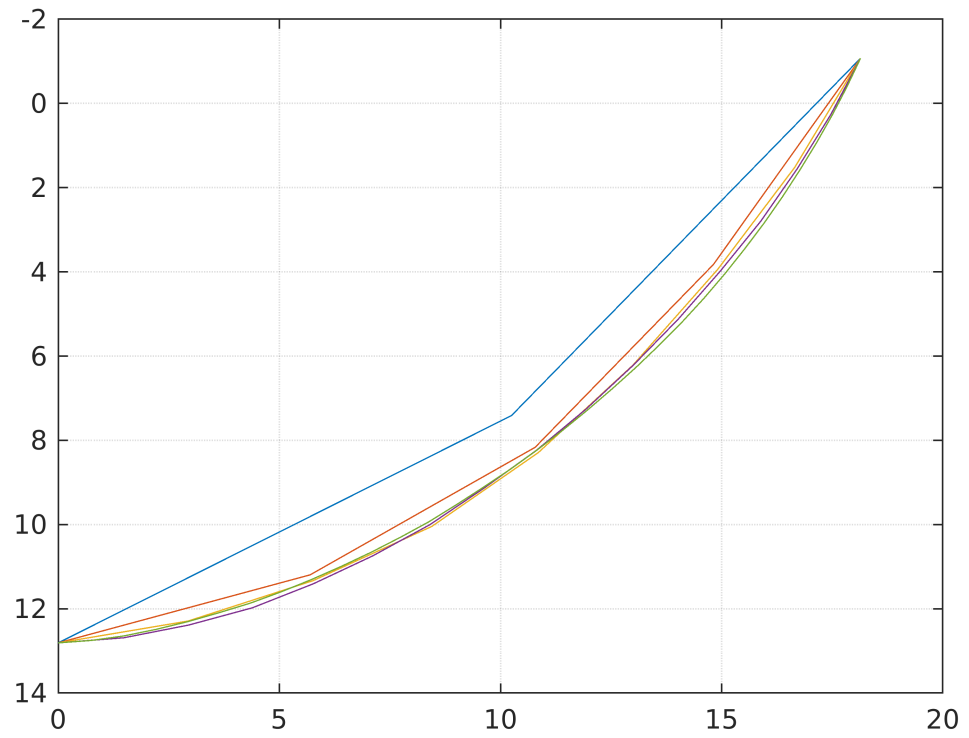


Figure 3: Ray tracing by bending method: 30% RANDOM VELOCITY FIELD LOADED

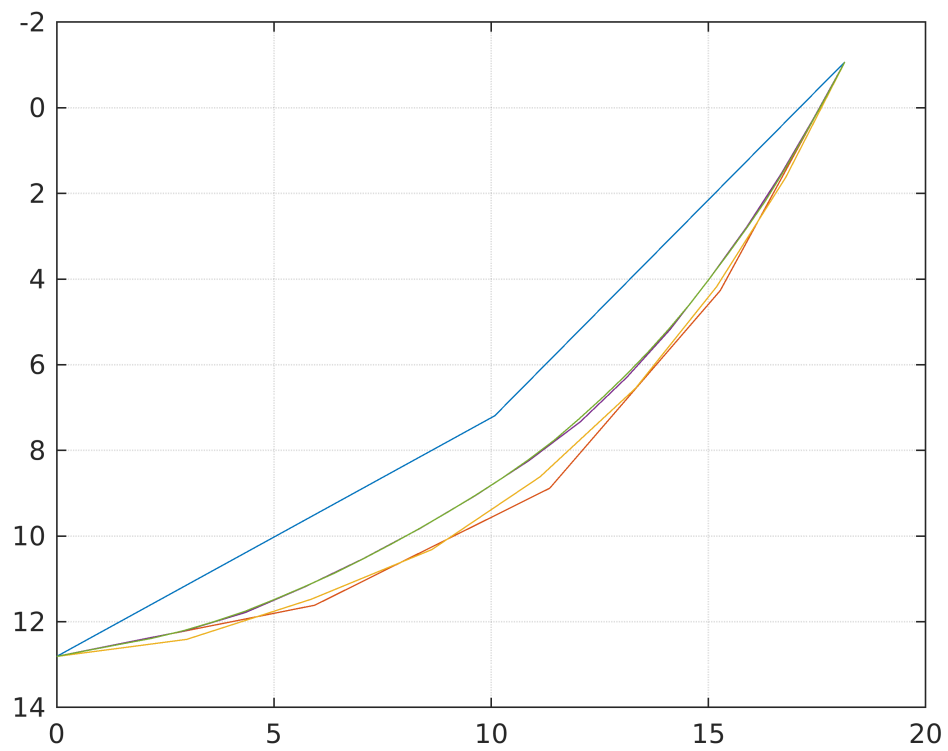


Figure 4: Ray tracing by bending method: 50% RANDOM VELOCITY FIELD LOADED

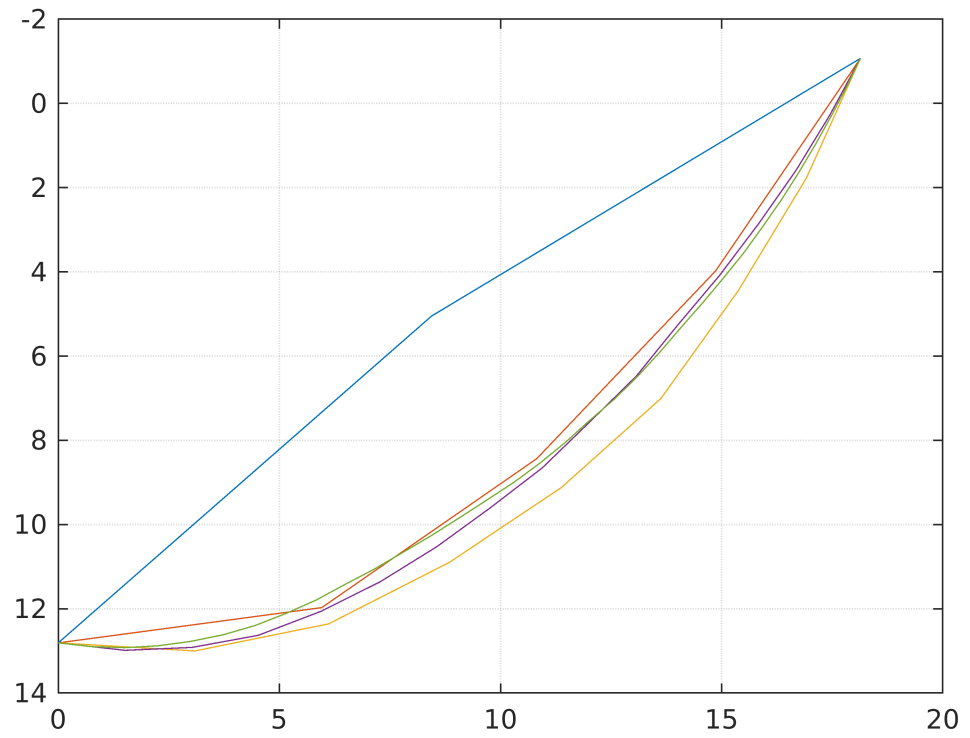


Figure 5: Ray tracing by bending method: 80% RANDOM VELOCITY FIELD LOADED

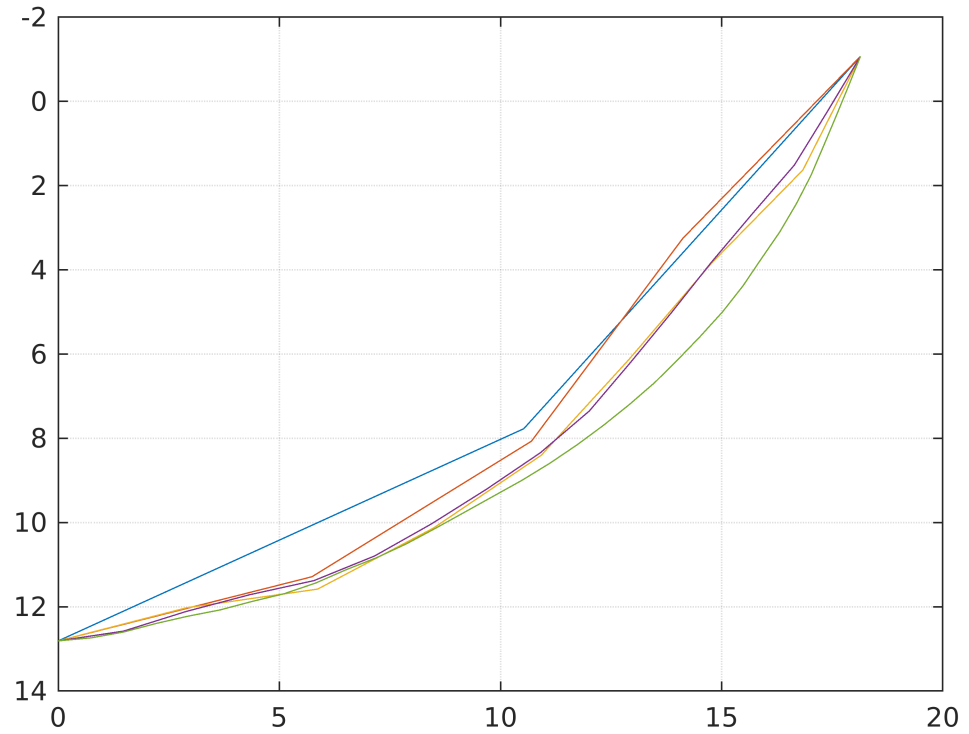


Figure 6: Ray tracing by bending method: 100% RANDOM VELOCITY FIELD LOADED

2.2 SUBROUTINE

```
function getRay(maxiter, thred_t)
%max iteration number
%threshold of time
global ray npts
while(1)
    niter=0;
    while(1)
        niter=niter+1;
        if(niter>=maxiter)
            break;
        end
        t0 = travelttime(npts,ray);
        perturbRay();
    end
end
```

```

        t1 = traveltime(npts,ray);
        if (abs(t0-t1)<thred_t )
            break;
        end
    end
    if (2*npts-1>=maxiter)
        break;
    end
    doublePath();
    t2 = traveltime(npts,ray);
    if (abs(t2-t1)<thred_t )
        break;
    end
    plotpath(ray);hold on
end

```