17. Diff. eq.

$$\frac{\partial h(x,t)}{\partial t} = \kappa \frac{\partial^2 h(x,t)}{\partial x^2}$$

$$h(x,0) = \begin{cases} \frac{H_{max}}{L} & , \ 0 \leq x \leq L \\ H_{max} - \frac{H_{max}(x-L)}{L} & , \ L \leq x \leq 2L \end{cases}$$

b) EB- treatment

$$T_j^{n+1} - T_j^n = D\left(T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}\right)$$

$$\rightarrow T_N^{n+1}(1+2D) - D\,T_{N-1}^{n+1} = T_N^n + D\,T_R$$

$$\underline{T}^n = (T_1^n, T_2^n \dots T_N^n)$$

$$\underline{C} = (D\,T_L, 0, \dots 0, D\,T_R)$$

$$\underline{\underline{M}} = \begin{pmatrix} 1+2D & -D & 0 & & \\ -D & 1+2D & \ddots & & \\ 0 & -D & & \ddots & -D \\ & & \ddots & -D & 1+2D \end{pmatrix}$$

$$D = \frac{\kappa \Delta t}{\Delta x^2} \qquad T = h(x,0), \ \underline{T}^{n+1} = \underline{\underline{M}}^{-1}\left[\underline{T}^n + \underline{C}\right]$$

Untitled1

November 23, 2017

```
In [1]: using PyPlot
```

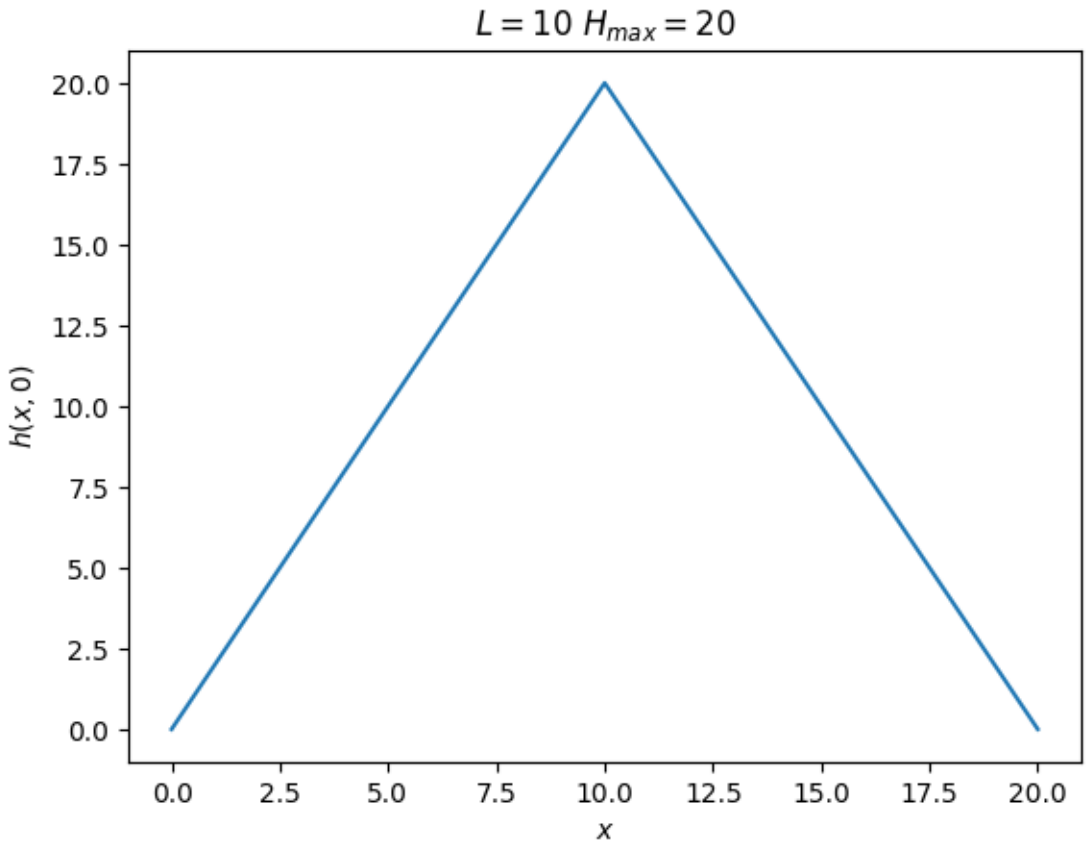
1 16. Diffusion equation: Evolution of a hill slope

2 a)

```
In [2]: function h(hmax,L)
            x=Array{Float64}(trunc(Int64,2*L/0.01)+1)
            for (i,val) in enumerate(0:0.01:2*L)
                if(val>=L)
                    x[i]=hmax-hmax*(val-L)/L
                else
                    x[i]=hmax*val/L
                end
            end
            plot(0:0.01:2L,x)
            title(L"$L=10$ $H_{max}=20$")
            ylabel(L"$h(x,0)$")
            xlabel(L"$x$")
        end
```

```
Out[2]: h (generic function with 1 method)
```

```
In [3]: h(20,10)
```




```
Out[3]: PyObject <matplotlib.text.Text object at 0x7f94de5e4950>
```

3 b) & c)

```
In [4]: function h2(hmax,L,x)
            if(x>=L)
                return(hmax-hmax*(x-L)/L)
            else
                return(hmax*x/L)
            end
        end
```

```
Out[4]: h2 (generic function with 1 method)
```

```
In [5]: function thomasalgo(d,a,b,y)
            N=length(d)
            #A=a' and Y=y''
            A=Array{Float64}(N)
            Y=Array{Float64}(N)
            x=Array{Float64}(N)
```

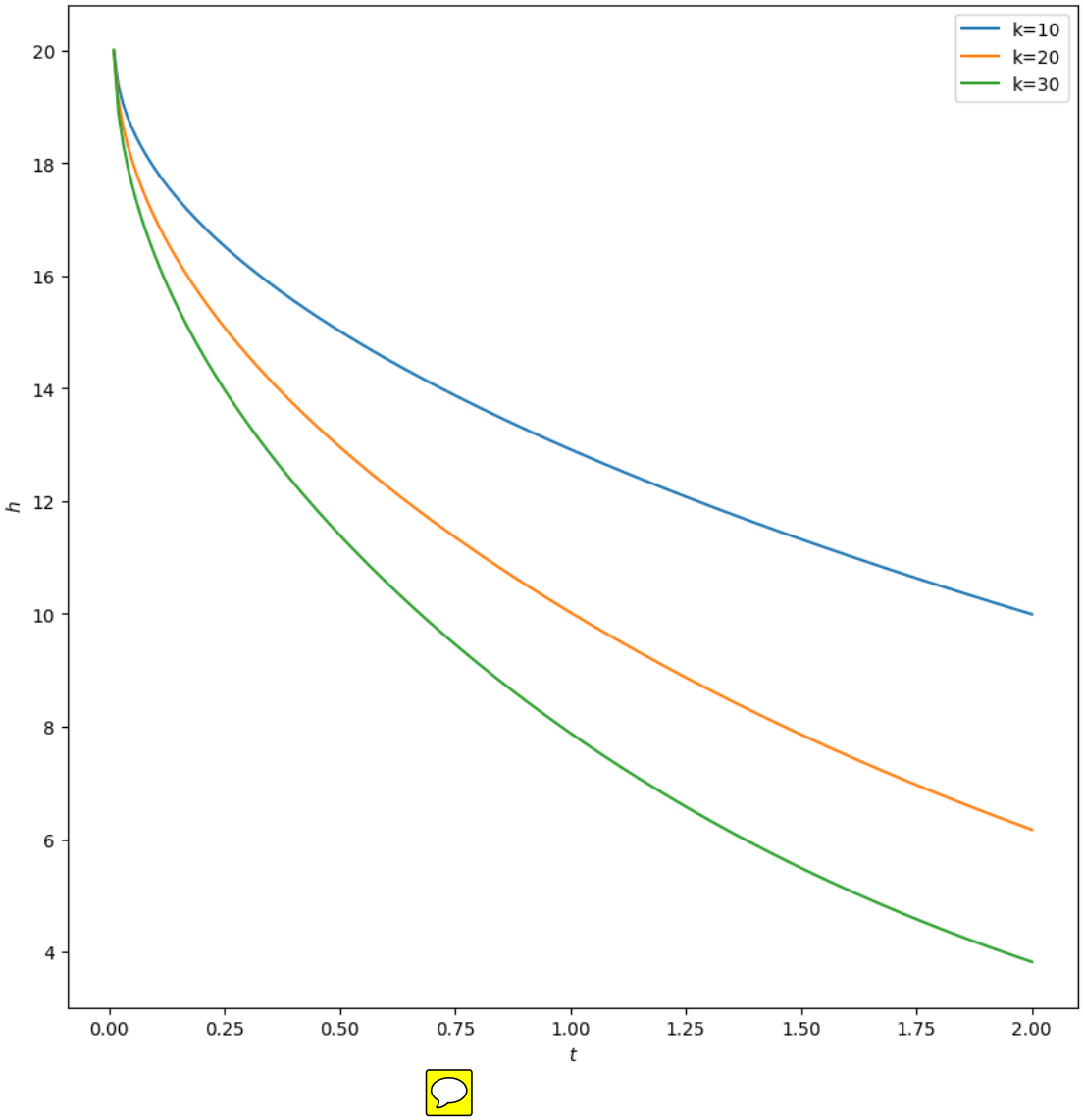
```
        #timestep1
        A[1]=a[1]/d[1]
        Y[1]=y[1]/d[1]
        #steps 2 to N-1
        for i in 2:(N-1)
            A[i]=a[i]/(d[i]-b[i]*A[i-1])
            Y[i]=(y[i]-b[i]*Y[i-1])/(d[i]-b[i]*A[i-1])
        end
        #N-th value
        Y[N]=(y[N]-b[N]*Y[N-1])/(d[N]-b[N]*A[N-1])

        #calculating the x-Vector
        x[N]=Y[N]
        for i in (N-1):-1:1
           x[i]=Y[i]-A[i]*x[i+1]
        end
        return(x)
    end
```

```
Out[5]: thomasalgo (generic function with 1 method)
```

```
In [12]: L=10
         hmax=20
         dx=0.1
         dt=0.01
         k=[10 20 30]
         t=zeros(Float64,trunc(Int64,2.*L/dx)+1)
         #altitude array
         x=Array{Float64}(3,100000000)
         #just a zero vector, can stay like that
         c=zeros(Float64,trunc(Int64,2.*L/dx)+1)
         tc=zeros(Float64,trunc(Int64,2.*L/dx)+1)
         a=zeros(Float64,trunc(Int64,2.*L/dx)+1)
         b=zeros(Float64,trunc(Int64,2.*L/dx)+1)
         d=zeros(Float64,trunc(Int64,2.*L/dx)+1)
         #n counts steps until hmax/2
         n=0
         for (j,kval) in enumerate(k)
             D=kval*dt/(dx^2)
             for (i,ival) in enumerate(0:dx:2*L)
                 t[i]=h2(hmax,L,ival)
                 a[i]=-D
                 b[i]=-D
                 d[i]=1.+2.*D
             end
             b[1]=0.
             a[length(a)]=0.
             #L and R=0
```

```
    t[1]=0.
    t[length(t)]=0.
    i2=0
    x[j,1]=t[trunc(Int64(L/dx+1))]
     #calculating timesteps until hmax of smallest k is reached
    while (t[trunc(Int64(L/dx))+1]>(hmax/2.))||(i2<n)
        tc=t+c
        #println(t[trunc(Int64(L/dx))+2])
        #println(d)
        #println(a)
        #println(b)
        #println(inv(diagm(d,0)+diagm(a[1:length(a)-1],1)+diagm(b[1:length(b)-1],-1))
        t=thomasalgo(d,a,b,tc)
        if j==1
            n+=1
        end
        i2+=1
        x[j,i2+1]=t[trunc(Int64(L/dx+1))]
    end
end
i=1:n+1
figure(1,figsize=(10,10))
plot(i*dt,x[1,i],label="k="*string(k[1]))
plot(i*dt,x[2,i],label="k="*string(k[2]))
plot(i*dt,x[3,i],label="k="*string(k[3]))
ylabel(L"$h$")
xlabel(L"$t$")
legend()
```



Out[12]: PyObject <matplotlib.legend.Legend object at 0x7f94d8a33450>