Assignment 1 CS4745/6025 Winter 2025

Georgiy Krylov

January 23, 2025

Due Date: Monday, February 3, 2025 - 8:30 am
Purpose: practice problem to help you learn Julia.
Fast Sweeping Method to Solve the Eikonal Equation.

Write Julia functions that solve the Eikonal equation for a given domain using the sequential Fast Sweeping Method, as described in Chapter 7 of the textbook (also see slides in D2L). Don't forget that Julia uses 1-based array indexing! The implementations must include the following functions:

```
# Solves the Eikonal equation using the Fast Sweeping Method
# Input is in file with name stored in source
# Input domain has ni x nj values, with a spacing of h
# Default tolerance is 1E−6
# Returns ni x nj solution matrix without extra layer of points
function solveEikonal(ni, nj, h, source, tol = 1E−6)

# Perform one set of sweeps on matrix U using directions specified in
# i1,ib,ja,jb, and speed function F, and current value of maximum
# absolute error (err = (U[i, j]−Unew)/U[i, j]), where Unew
# is new value of U[i,j] calculated using solveQuadratic()
# h is the spacing between points
# Returns updated value of maxerr
function sweep!(U, ia, ib, ja, jb, F, h, maxerr)

# Solve the discretized Eikonal equation at (i,j), given
# speed function F, and spacing between points h
# Returns the new value of U[i,j]
function solveQuadratic(U, i, j, F, h)
```

`solveEikonal()` must also use the following function, which initializes the F and U matrices:

```
# Reads from source a speed function F defined a ni x nj grid
# as well as coordinates of boundary of front
# (input is terminated with a negative number on last line)
# Also initializes solution matrix U
# U and F are (ni+2) x (nj+2) 2D matrices
# Requires the DelimitedFiles package
function initialize!(F, U, ni, nj, source)
```

```
    temp = readdlm(source)
    for j in 1:nj, i in 1:ni
        F[i+1, j+1] = Float64(temp[i, j])
    end
    for i in ni+1:size(temp, 1)-1 # skip last line that has -1 terminator
    # +2: skip border and convert input from 0-based indexing
    U[temp[i,1]+2, temp[i,2]+2] = 0
    end
    nothing
end
```

```
julia> include("eikonal.jl")
2 iterations, maxerr = 0.0
7×7 view(::Matrix{Float64}, 2:8, 2:8) with eltype Float64:
 4.75515  4.04804  3.44223  3.0  3.44223  4.04804  4.75515
 4.04804  3.25244  2.54533  2.0  2.54533  3.25244  4.04804
 3.44223  2.54533  1.70711  1.0  1.70711  2.54533  3.44223
 3.0      2.0      1.0      0.0  1.0      2.0      3.0
 3.44223  2.54533  1.70711  1.0  1.70711  2.54533  3.44223
 4.04804  3.25244  2.54533  2.0  2.54533  3.25244  4.04804
 4.75515  4.04804  3.44223  3.0  3.44223  4.04804  4.75515
```

Figure 1: Example, for illustrative 7x7 problem in book (and slides)

Your solveEikonal() function should print the number of iterations and maxerr, as in the example above. The example also gives the hint that a view can be used to return the result without the outer layer. Two input files are provided for testing: `ex1.txt`, and `test100.txt` (a 100x100 domain similar to the example in the book and slides with a horizontal barrier). For the second test case, report your result using a heatmap: heatmap(1:100,1:100,U,color=:gist_yarg), available in the `Plots` package. **To pass in the assignment:** Submit a `.zip` file with your source code (`.eikonal.jl`) and a single pdf document containing listings of your code and output from your testing, and submit it via D2L. Please note that a screenshot of your code and/or output is not acceptable. Name your document LastName_FirstName A1.pdf (LastName and FirstName are of course substituted with your last and first name).