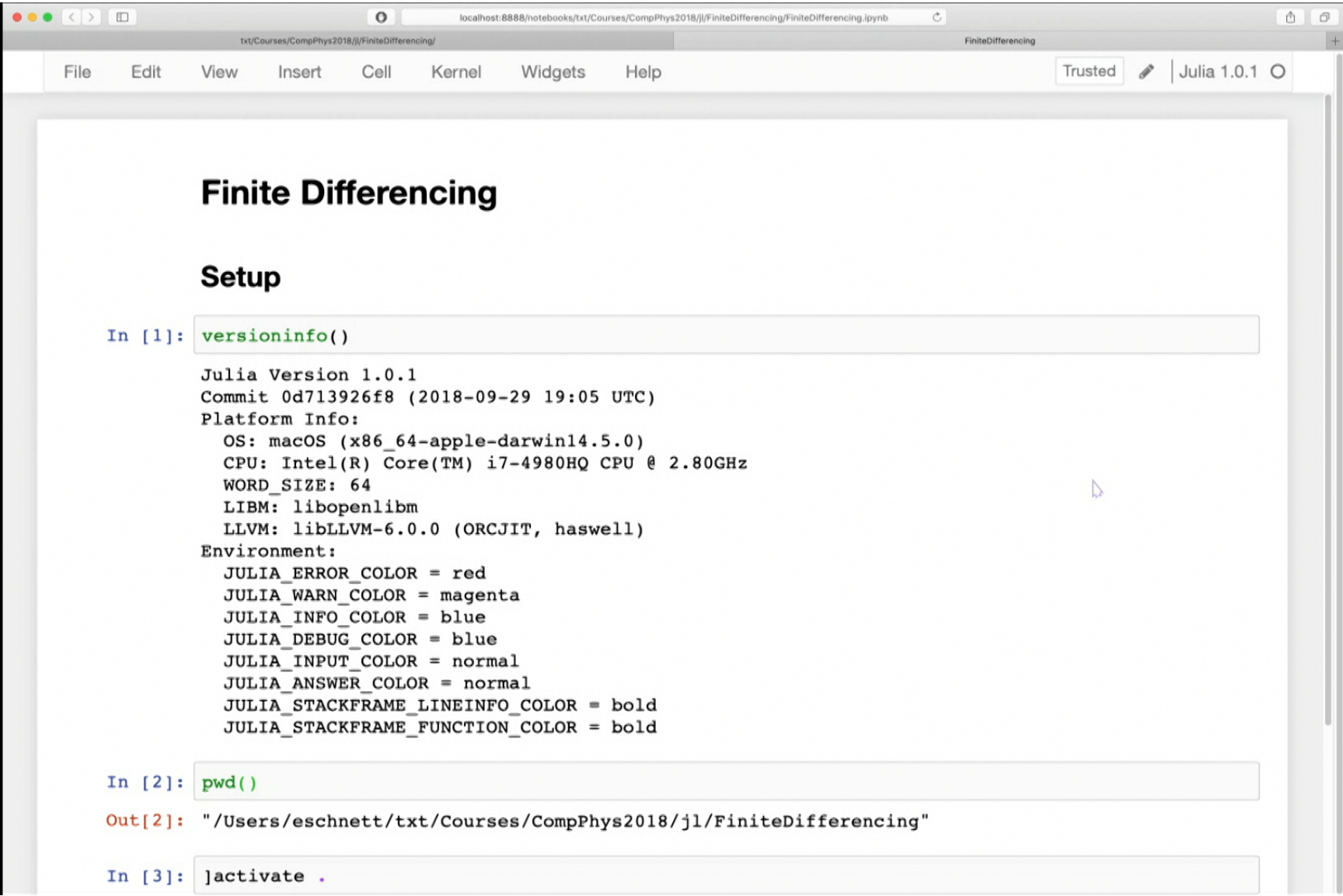
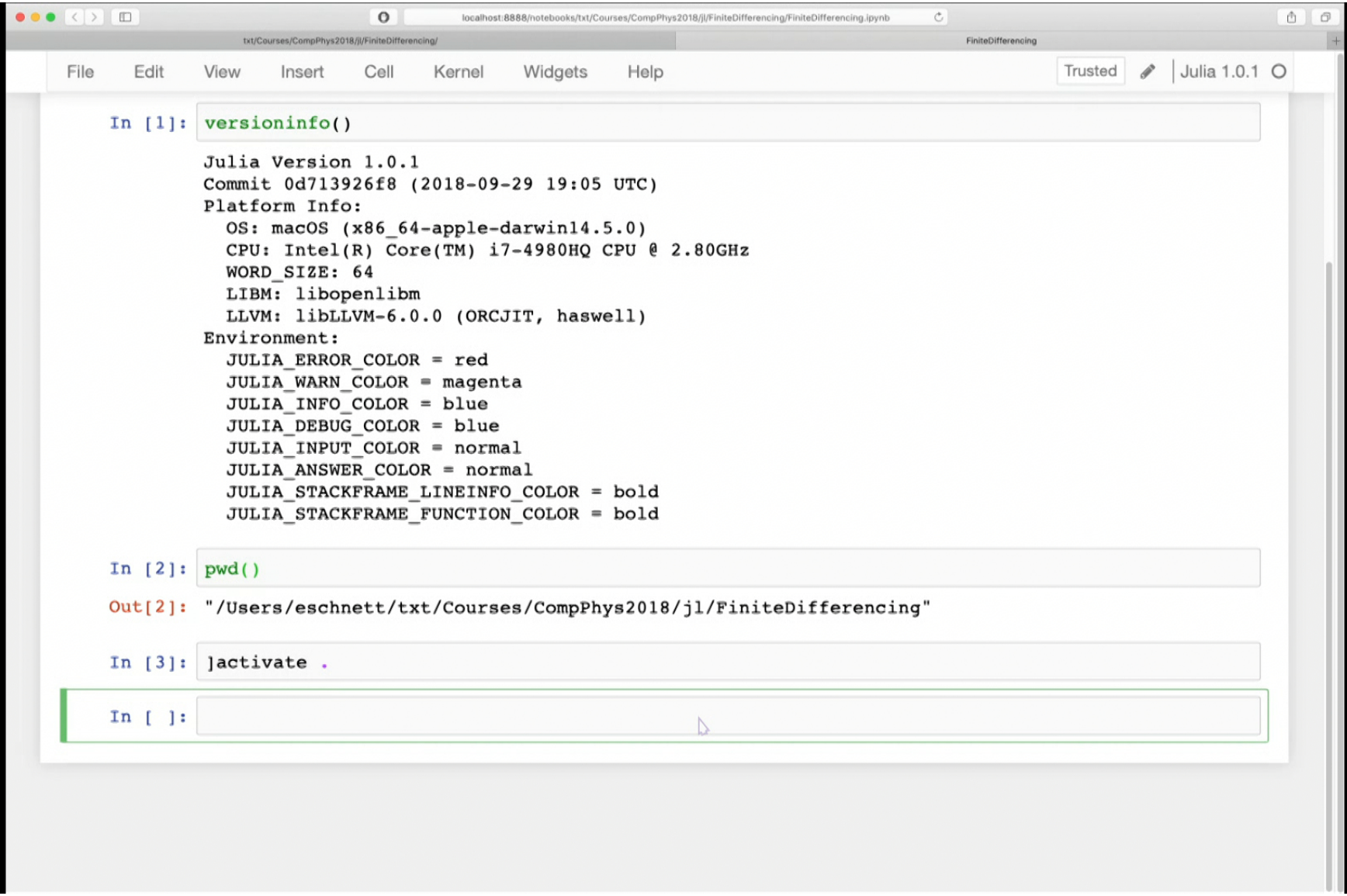
Title: Computational Physics - Lecture 17

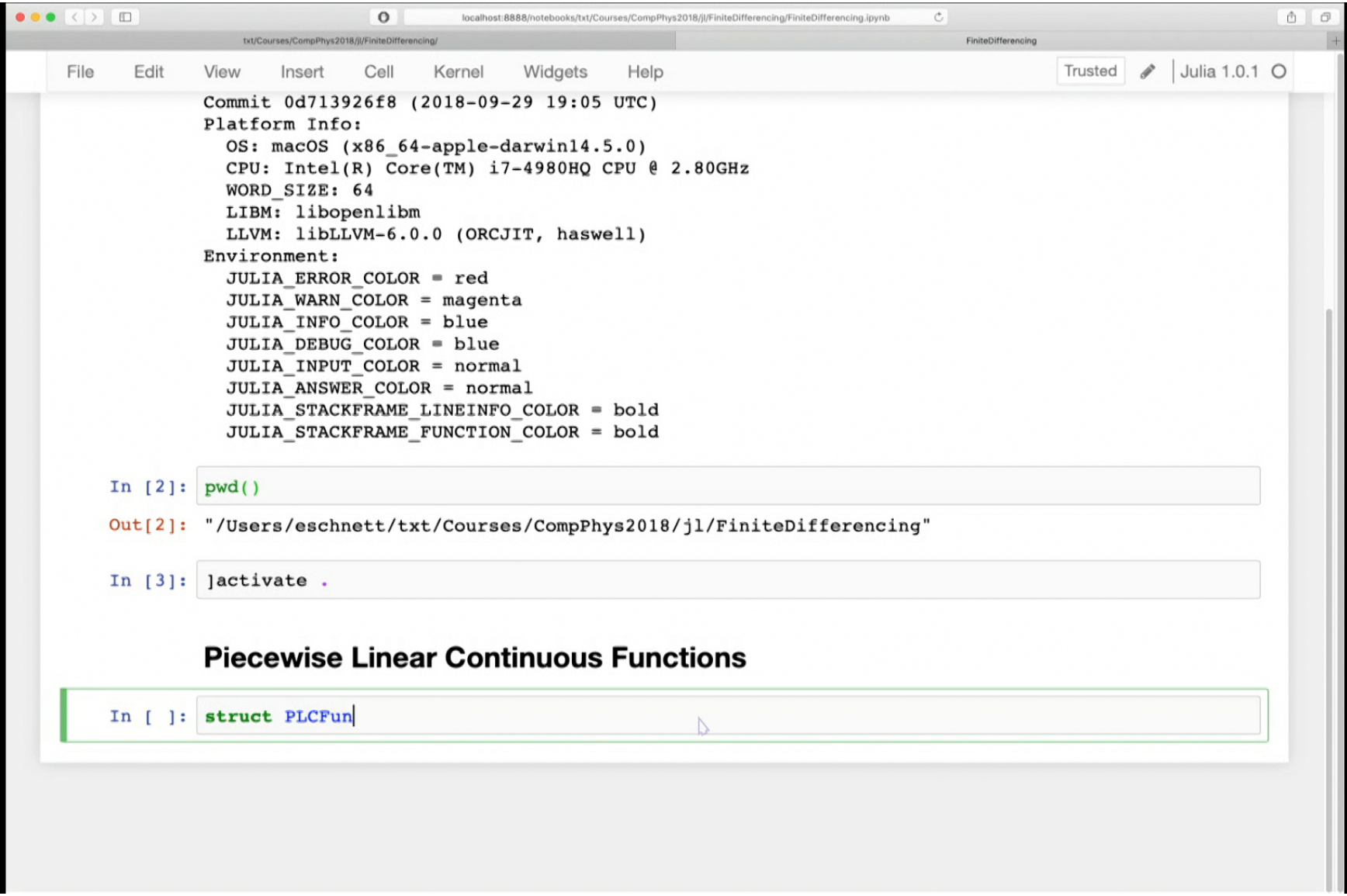
Date: Nov 07, 2018 01:00 PM

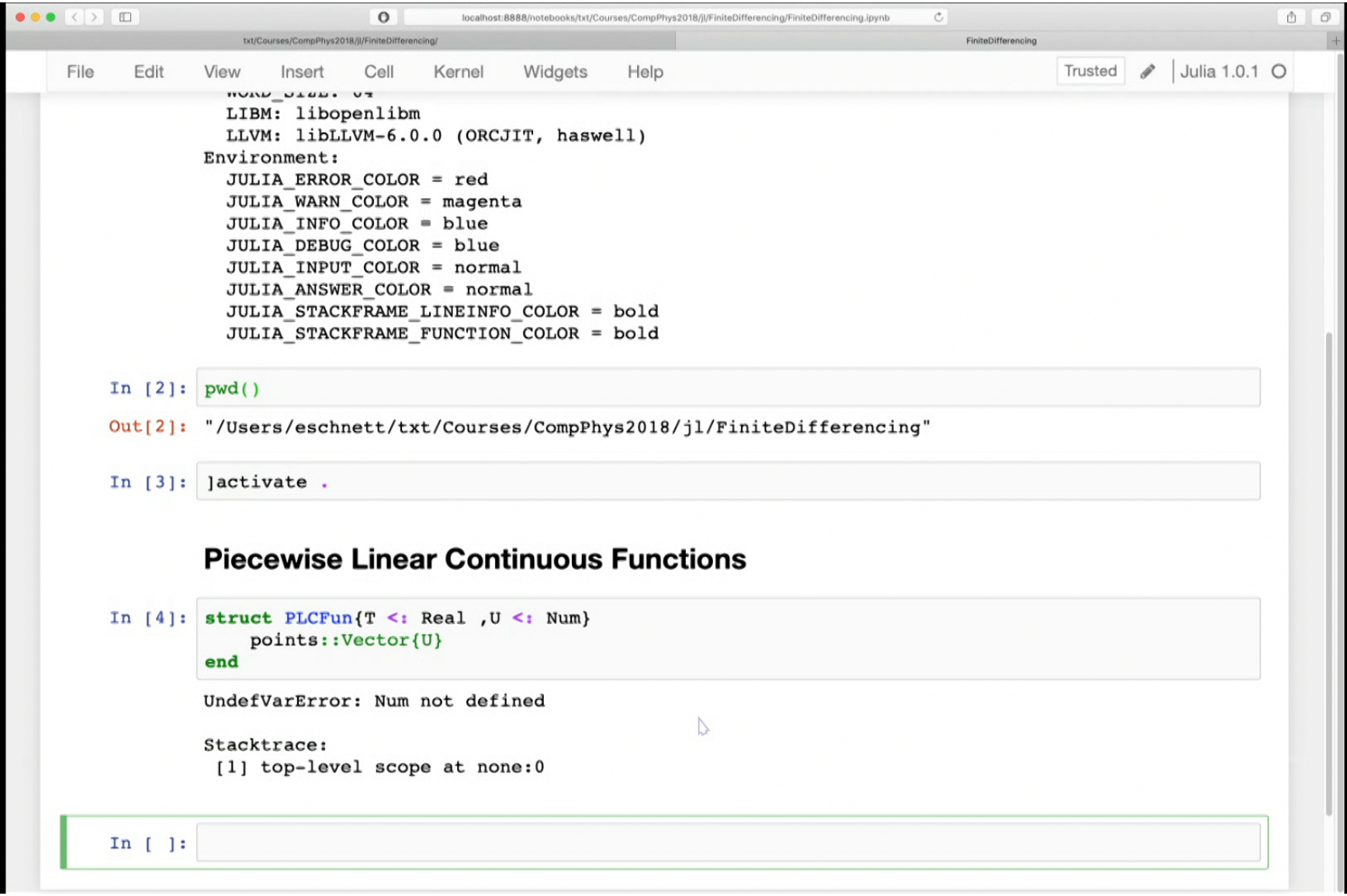
URL: http://pirsa.org/18110043

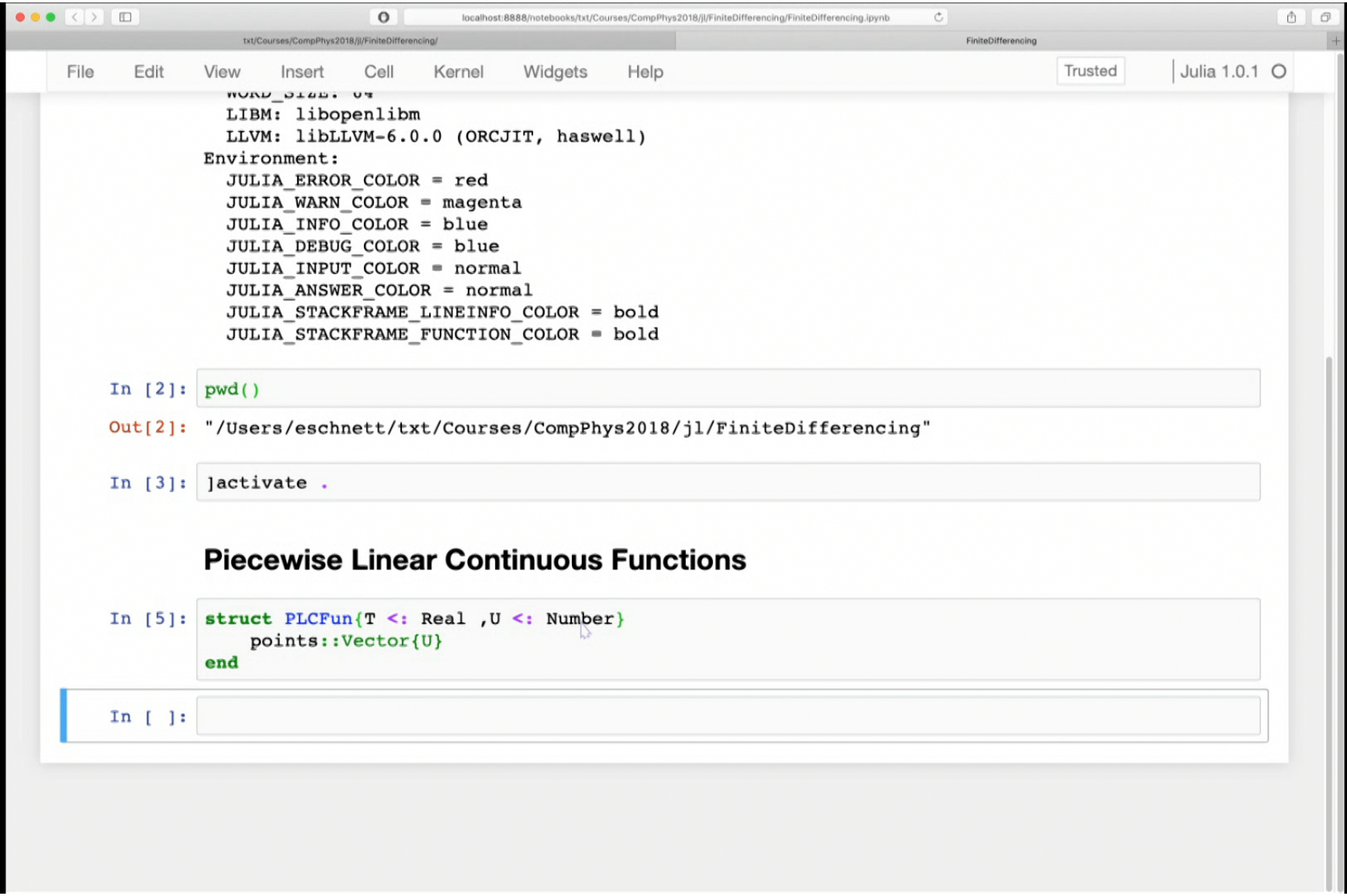
Abstract:

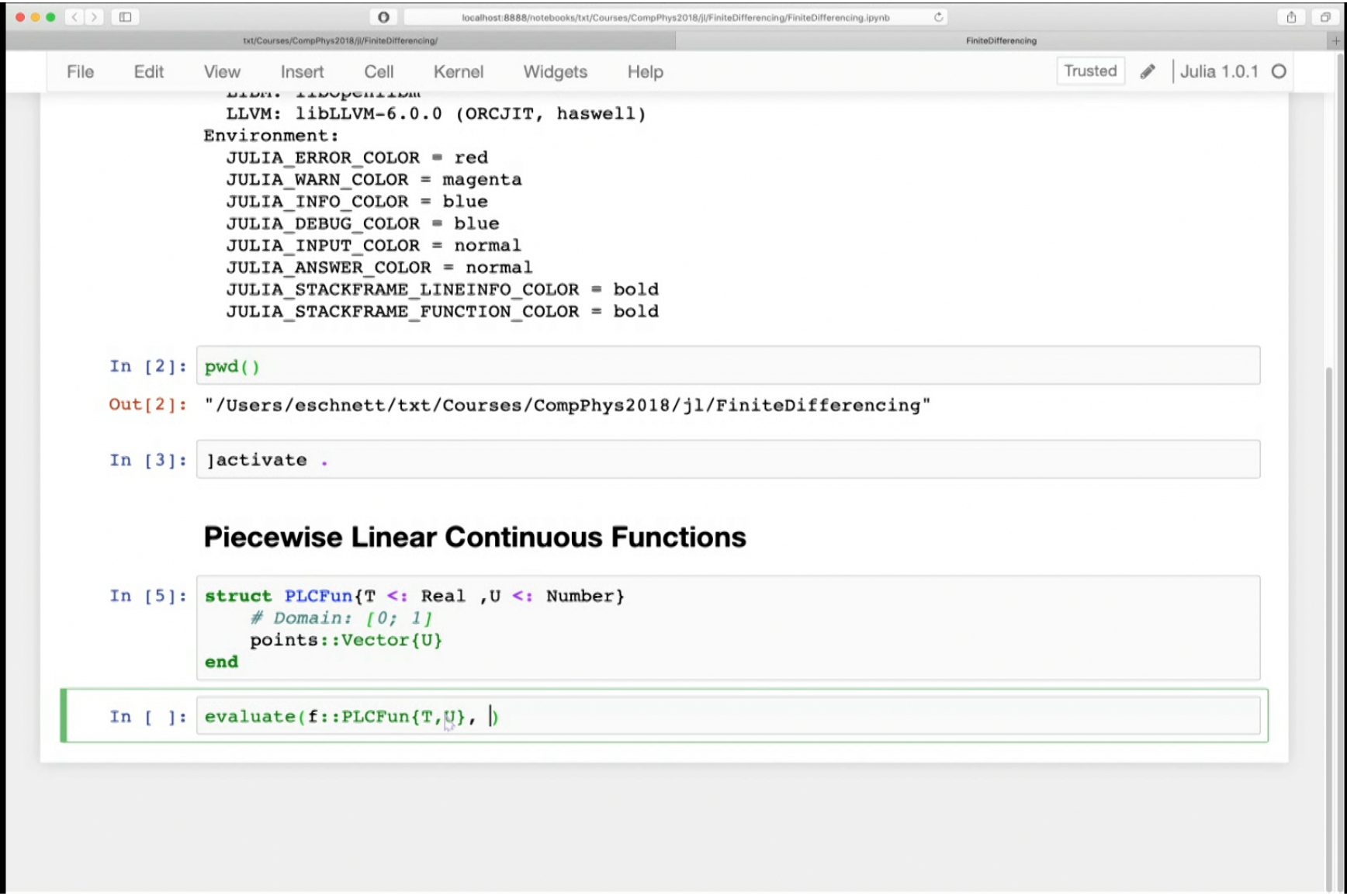

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Finite Differencing
Setup
In [1]: versioninfo()
Julia Version 1.0.1
Commit 0d713926f8 (2018-09-29 19:05 UTC)
Platform Info:
OS: macOS (x86_64-apple-darwin14.5.0)
CPU: Intel(R) Core(TM) i7-4980HQ CPU @ 2.80GHz
WORD_SIZE: 64
LIBM: libopenlibm
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .

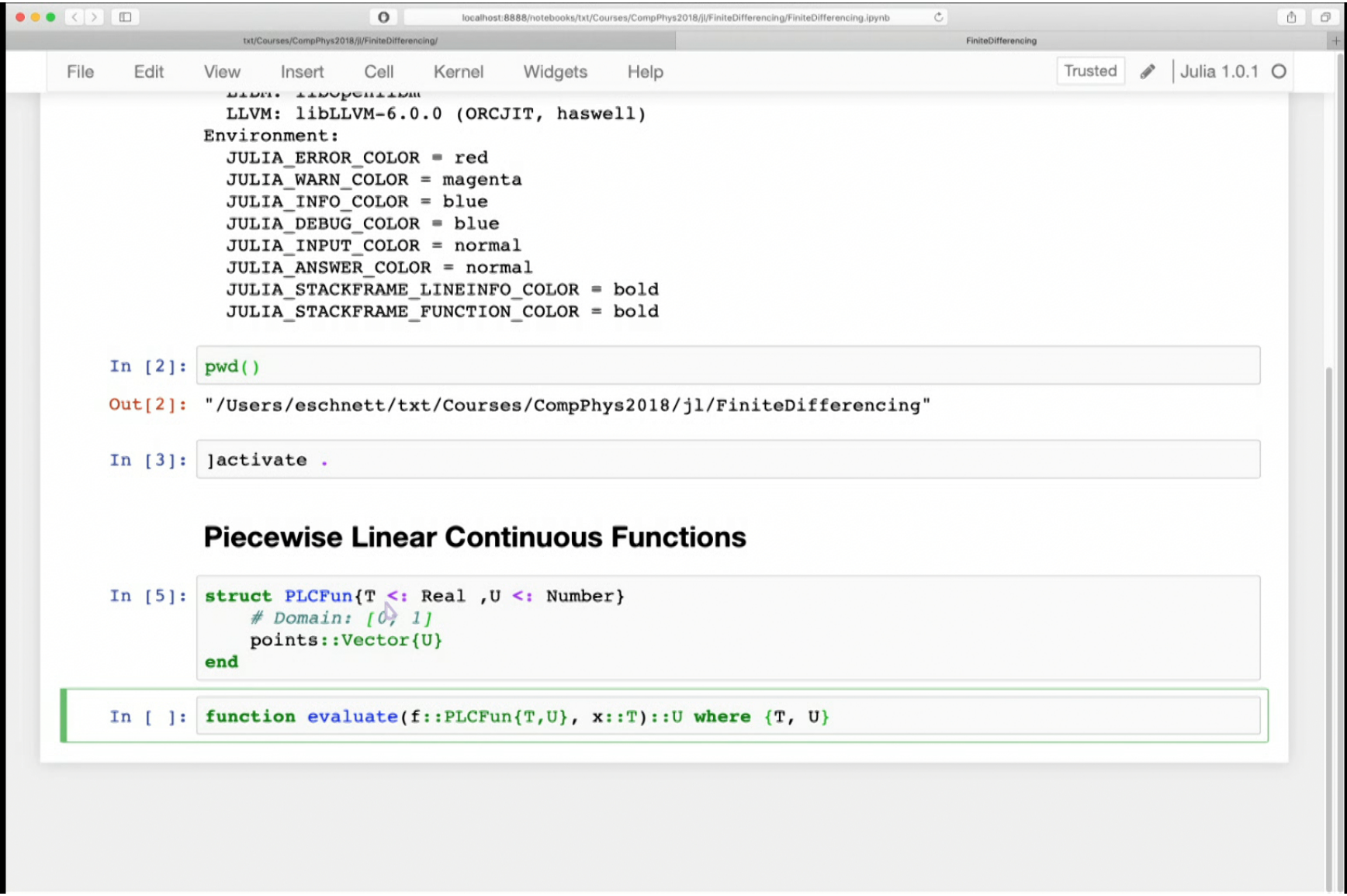

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
In [1]: versioninfo()
Julia Version 1.0.1
Commit 0d713926f8 (2018-09-29 19:05 UTC)
Platform Info:
OS: macOS (x86_64-apple-darwin14.5.0)
CPU: Intel(R) Core(TM) i7-4980HQ CPU @ 2.80GHz
WORD_SIZE: 64
LIBM: libopenlibm
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
In [ ]:

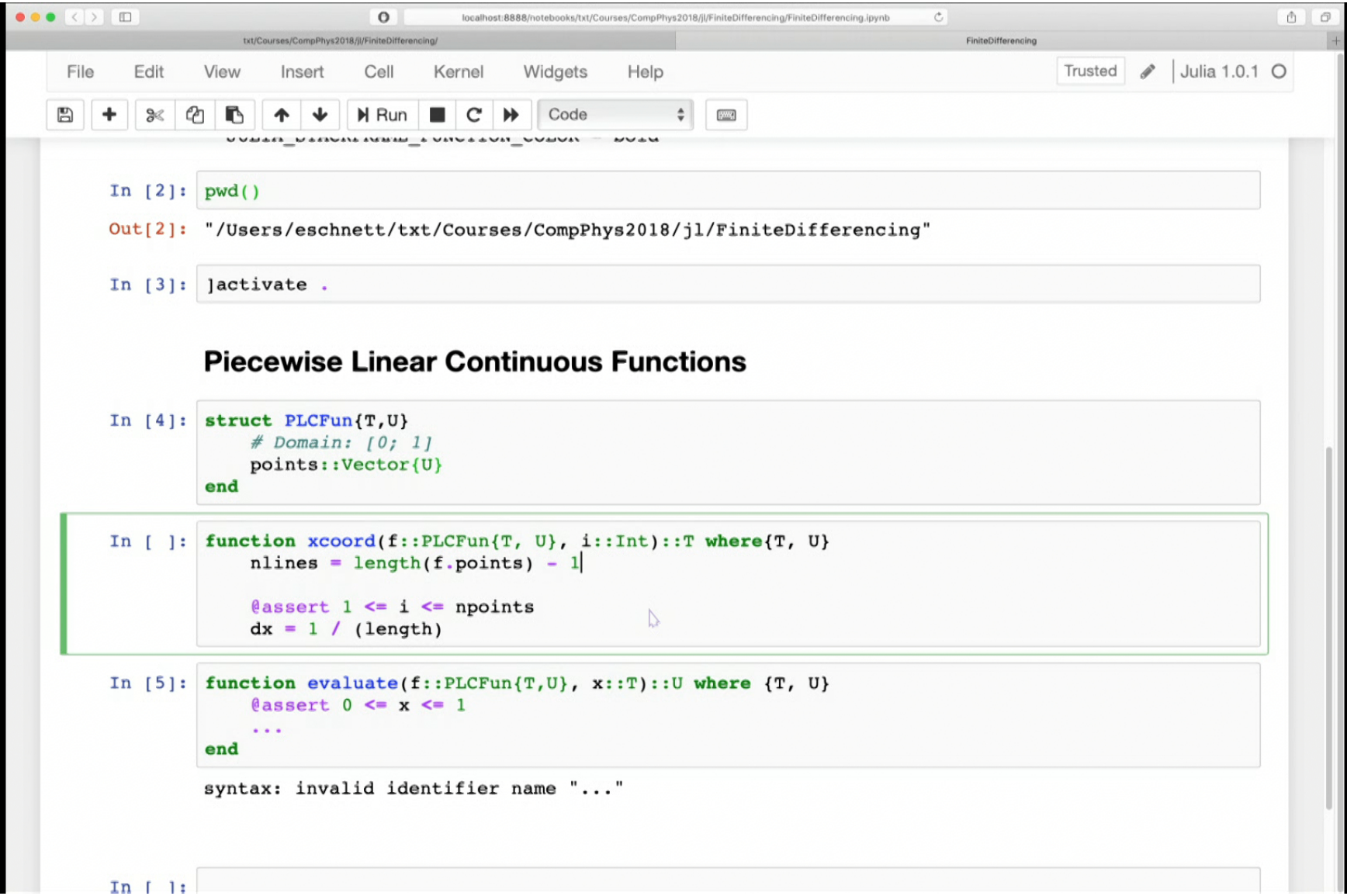

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Commit 0d713926f8 (2018-09-29 19:05 UTC)
Platform Info:
OS: macOS (x86_64-apple-darwin14.5.0)
CPU: Intel(R) Core(TM) i7-4980HQ CPU @ 2.80GHz
WORD_SIZE: 64
LIBM: libopenlibm
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [ ]: struct PLCFun

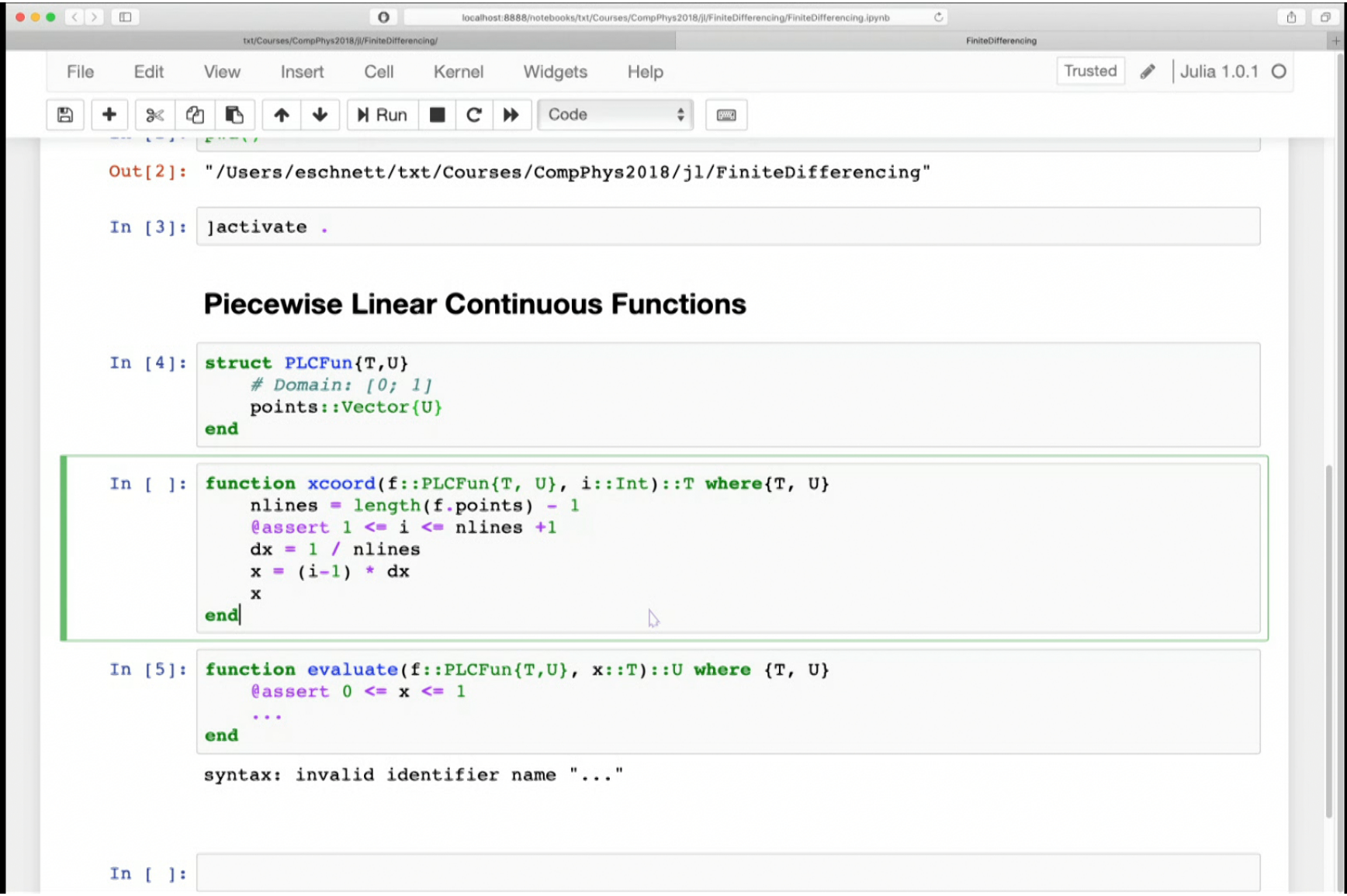

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
LIBM: libopenlibm
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [4]: struct PLCFun{T <: Real ,U <: Num}
points::Vector{U}
end
UndefVarError: Num not defined
Stacktrace:
[1] top-level scope at none:0
In [ ]:

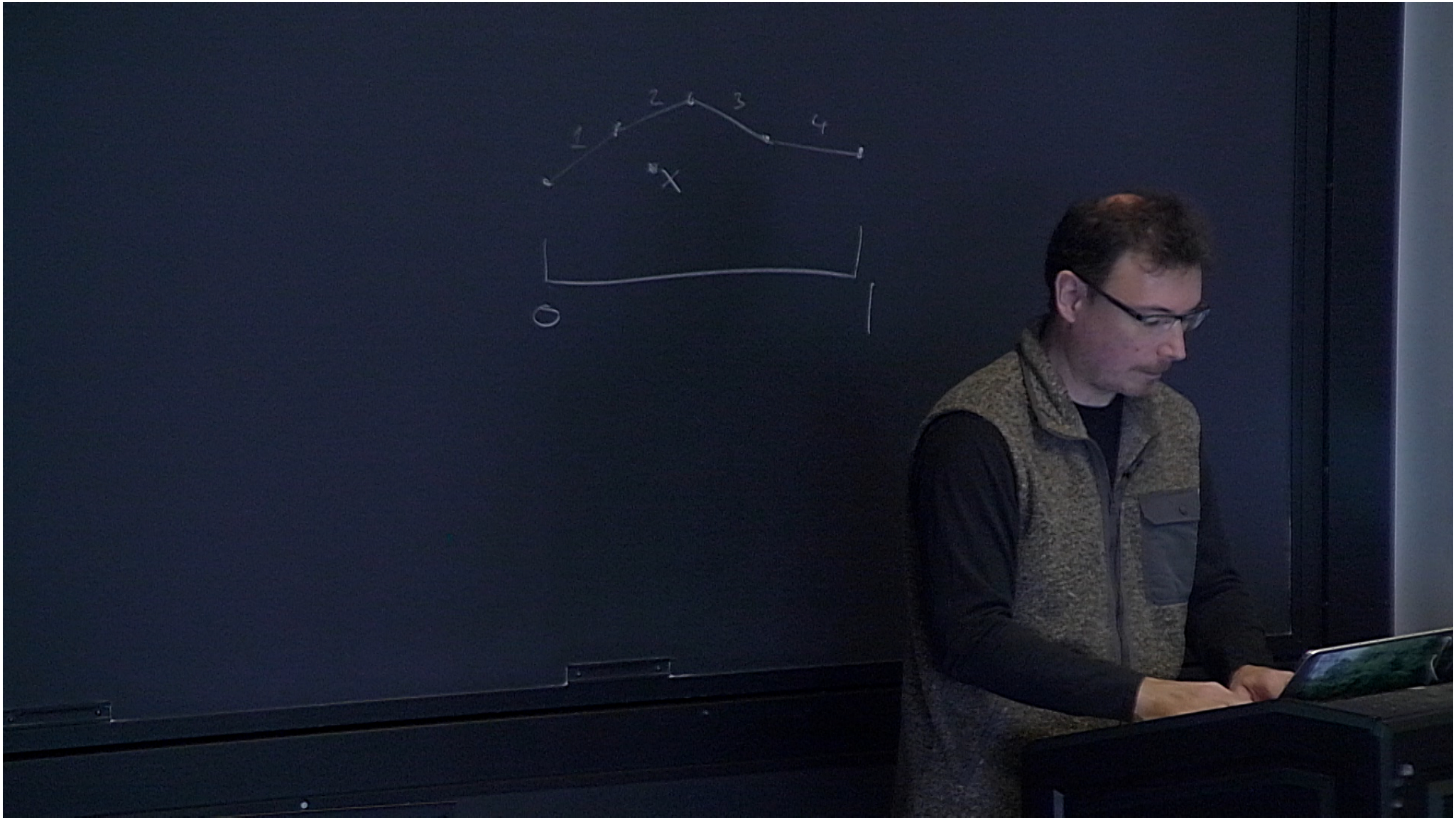

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
LIBM: libopenlibm
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [5]: struct PLCFun{T <: Real ,U <: Number}
points::Vector{U}
end
In [ ]:

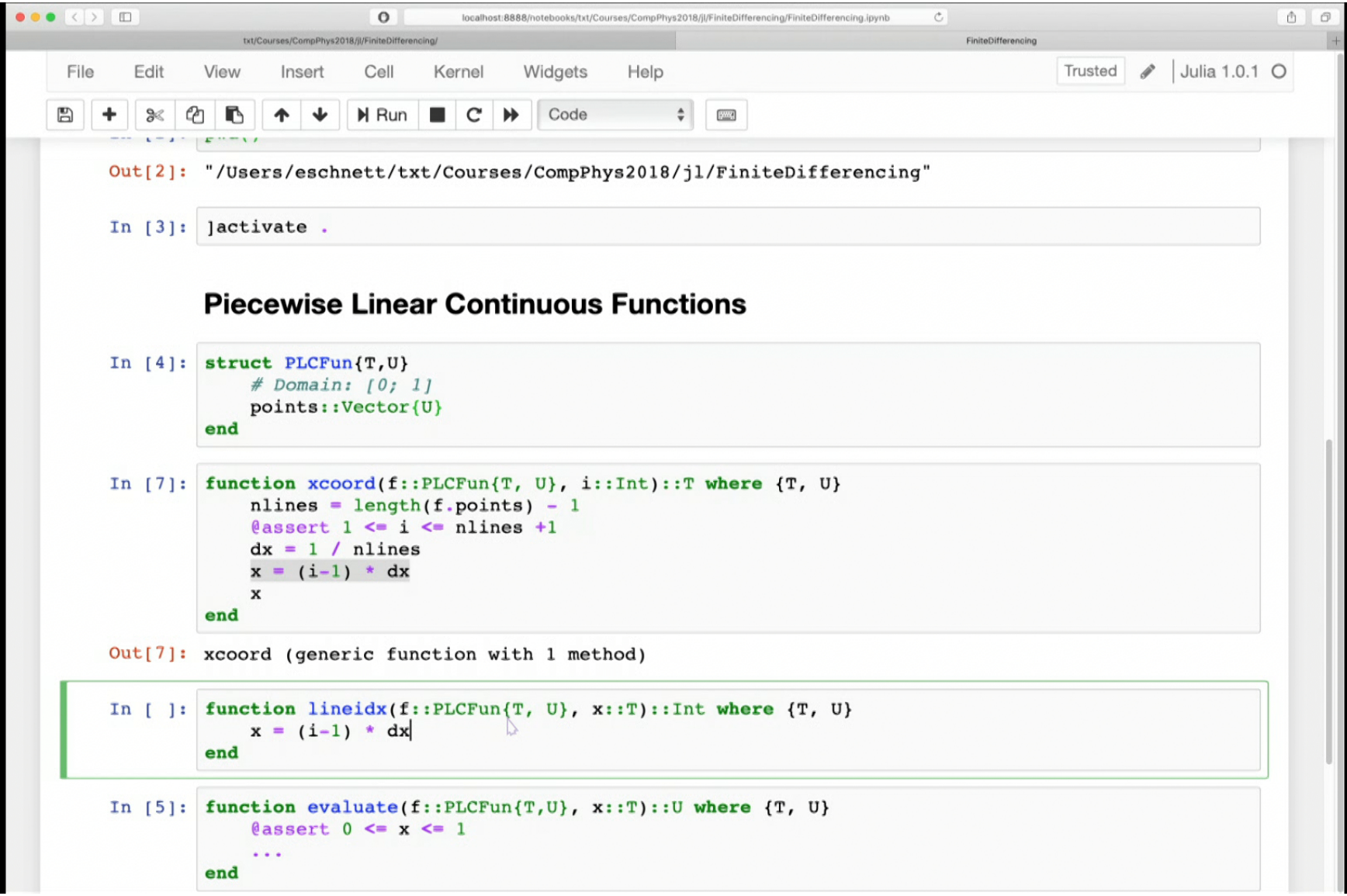

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [5]: struct PLCFun{T <: Real ,U <: Number}
Domain: [0; 1]
points::Vector{U}
end
In [ ]: evaluate(f::PLCFun{T,U},)

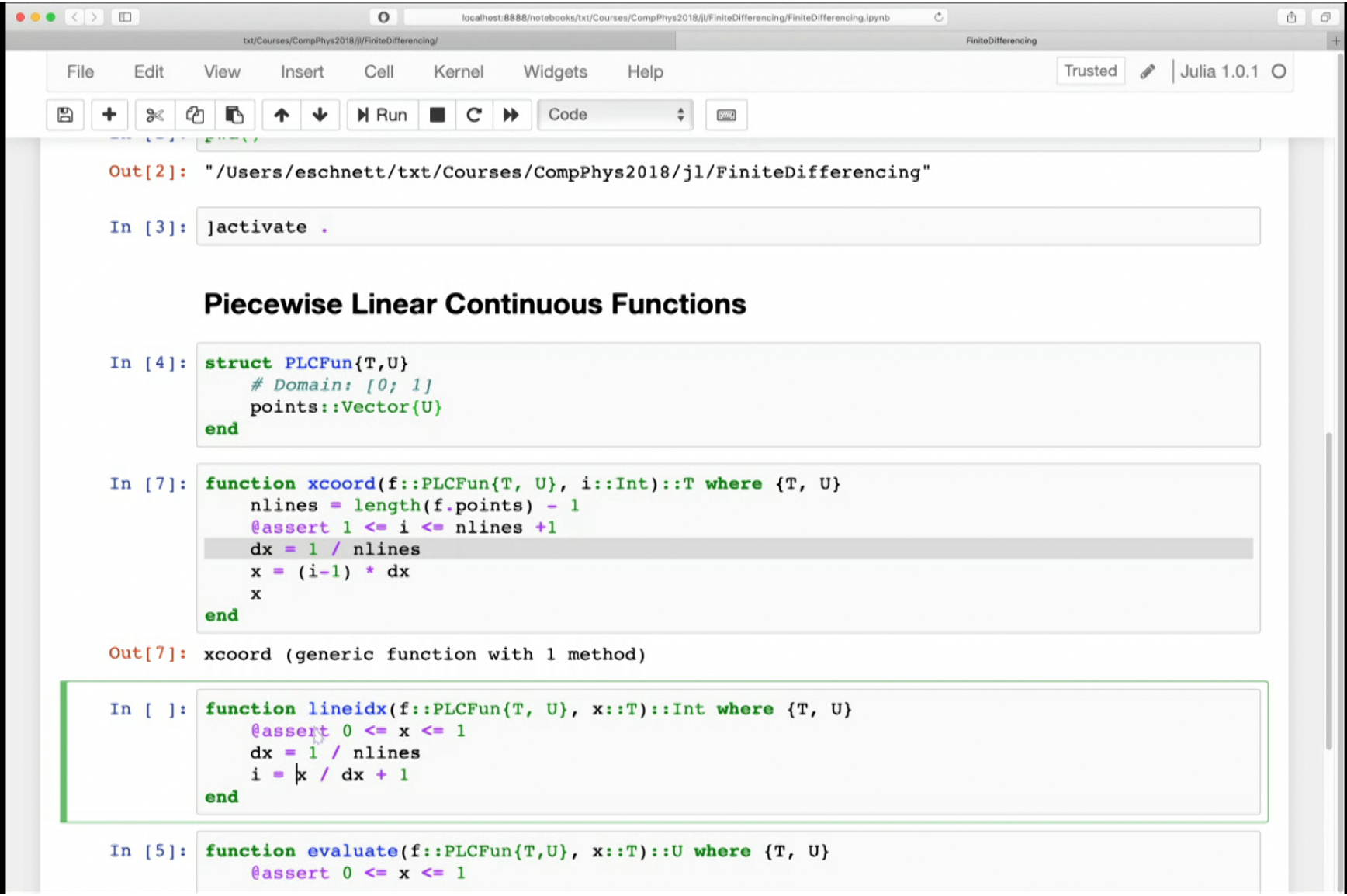

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [5]: struct PLCFun{T <: Real ,U <: Number}
Domain: [0, 1]
points::Vector{U}
end
In [ ]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}

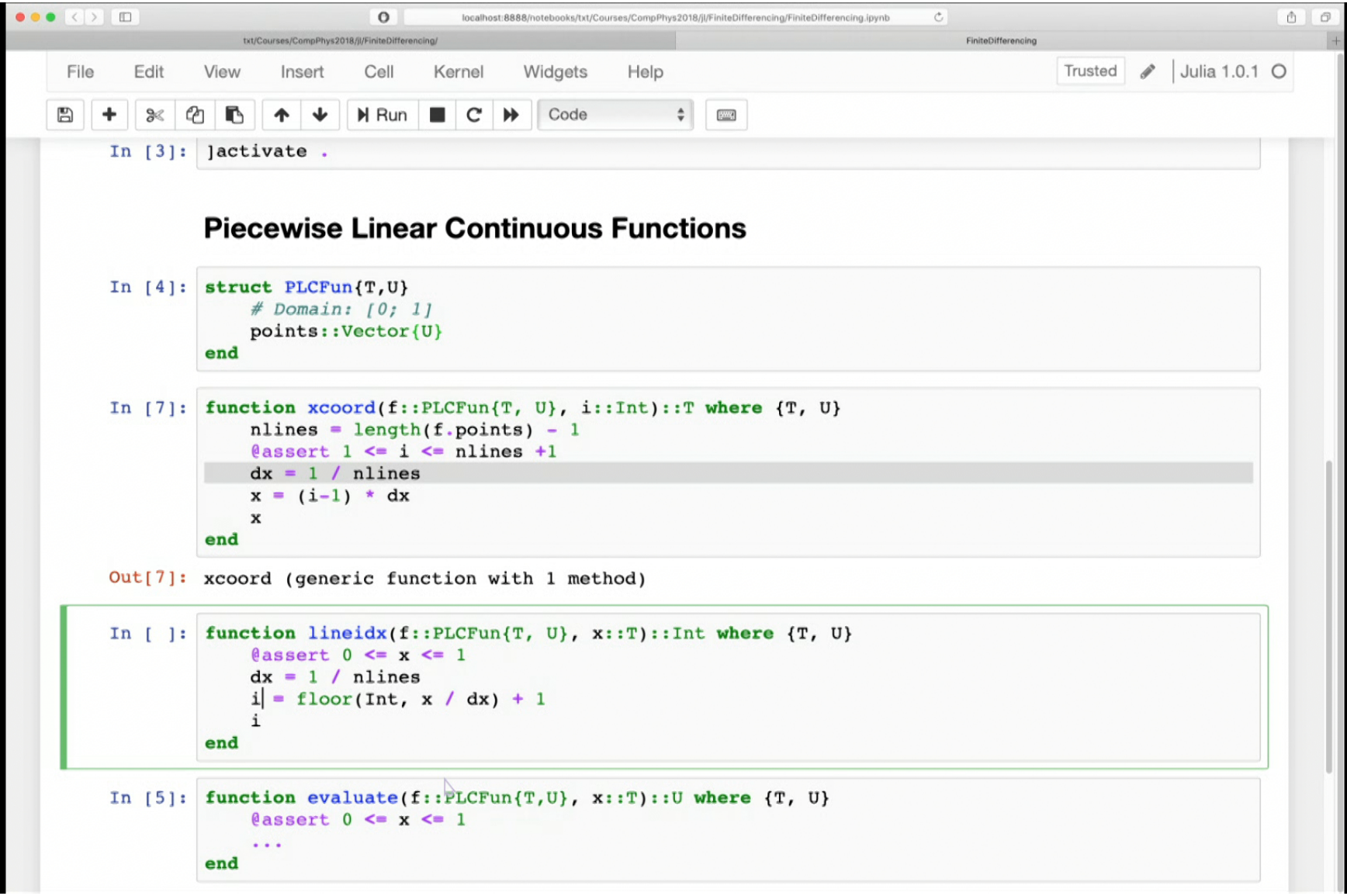

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
In [2]:
pwd()
Out[2]:
"/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:
]activate .
Piecewise Linear Continuous Functions
In [4]:
struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [ ]:
function xcoord(f::PLCFun{T, U}, i::Int)::T where{T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= npoints
dx = 1 / (length)
In [5]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
...
end
syntax: invalid identifier name "..."
In [ ]:

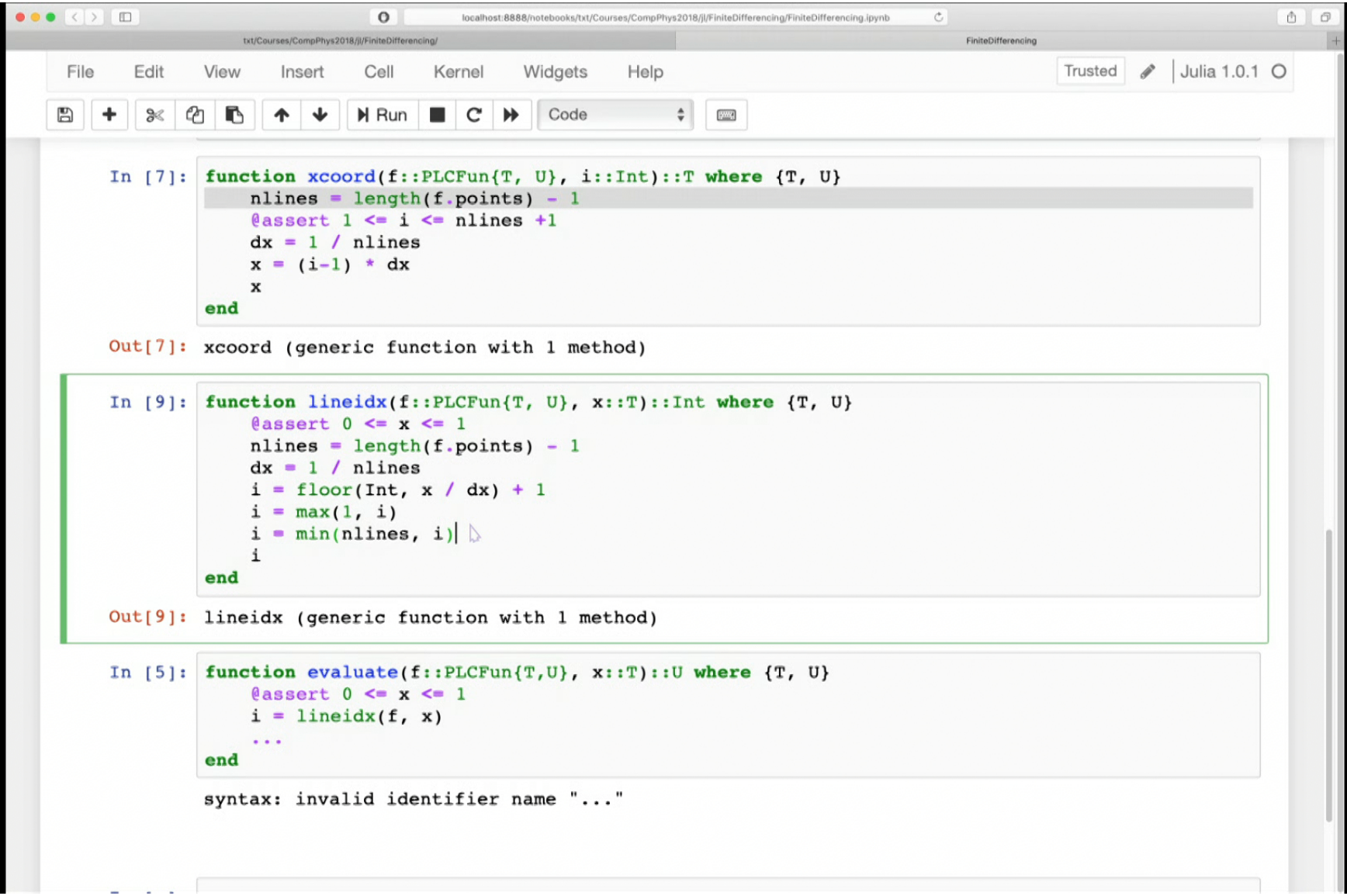

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [4]: struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [ ]: function xcoord(f::PLCFun{T, U}, i::Int)::T where{T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= nlines +1
dx = 1 / nlines
x = (i-1) * dx
x
end
In [5]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
...
end
syntax: invalid identifier name "..."
In [ ]:

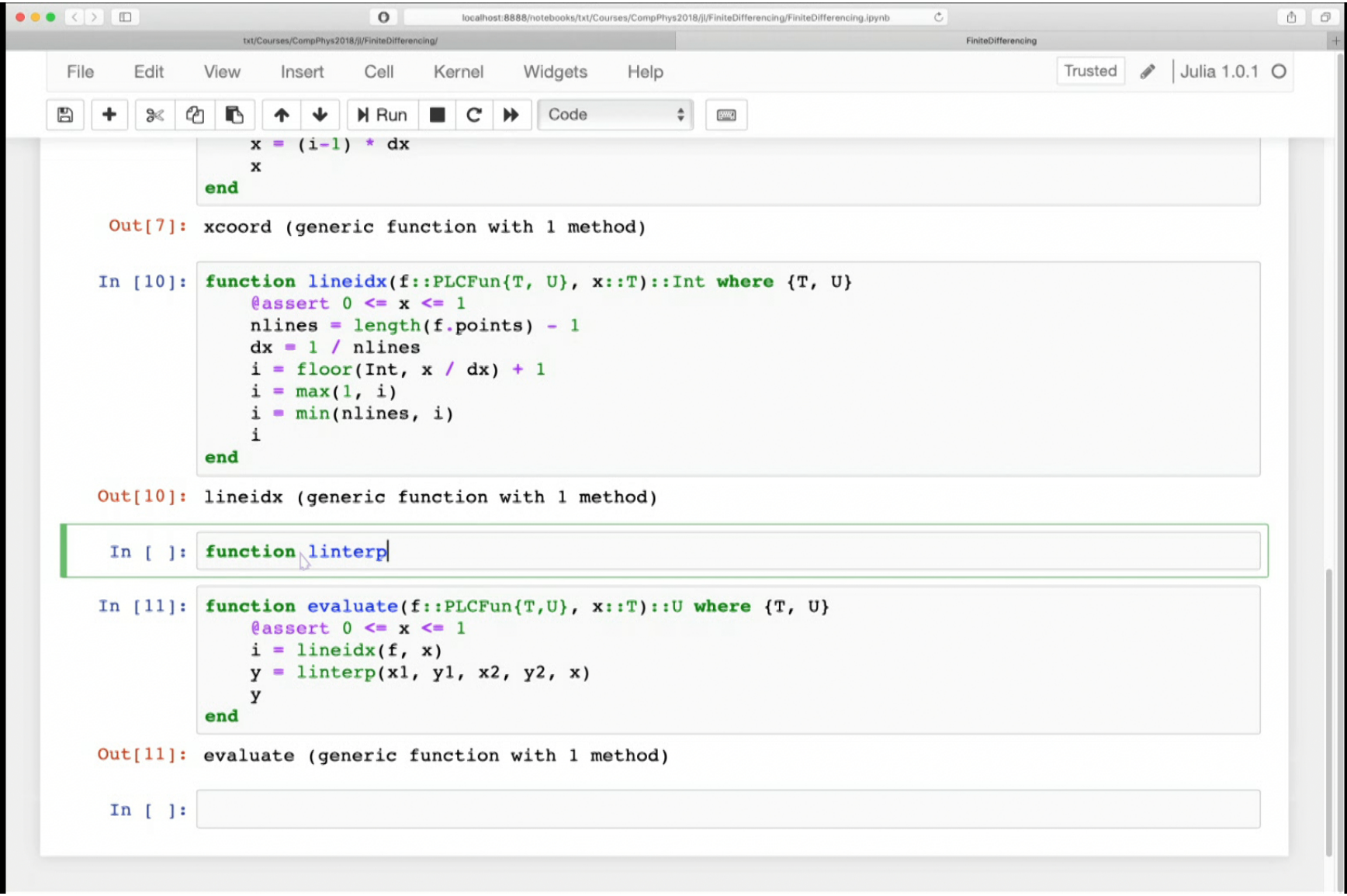

1
2
3
4
X
0
1

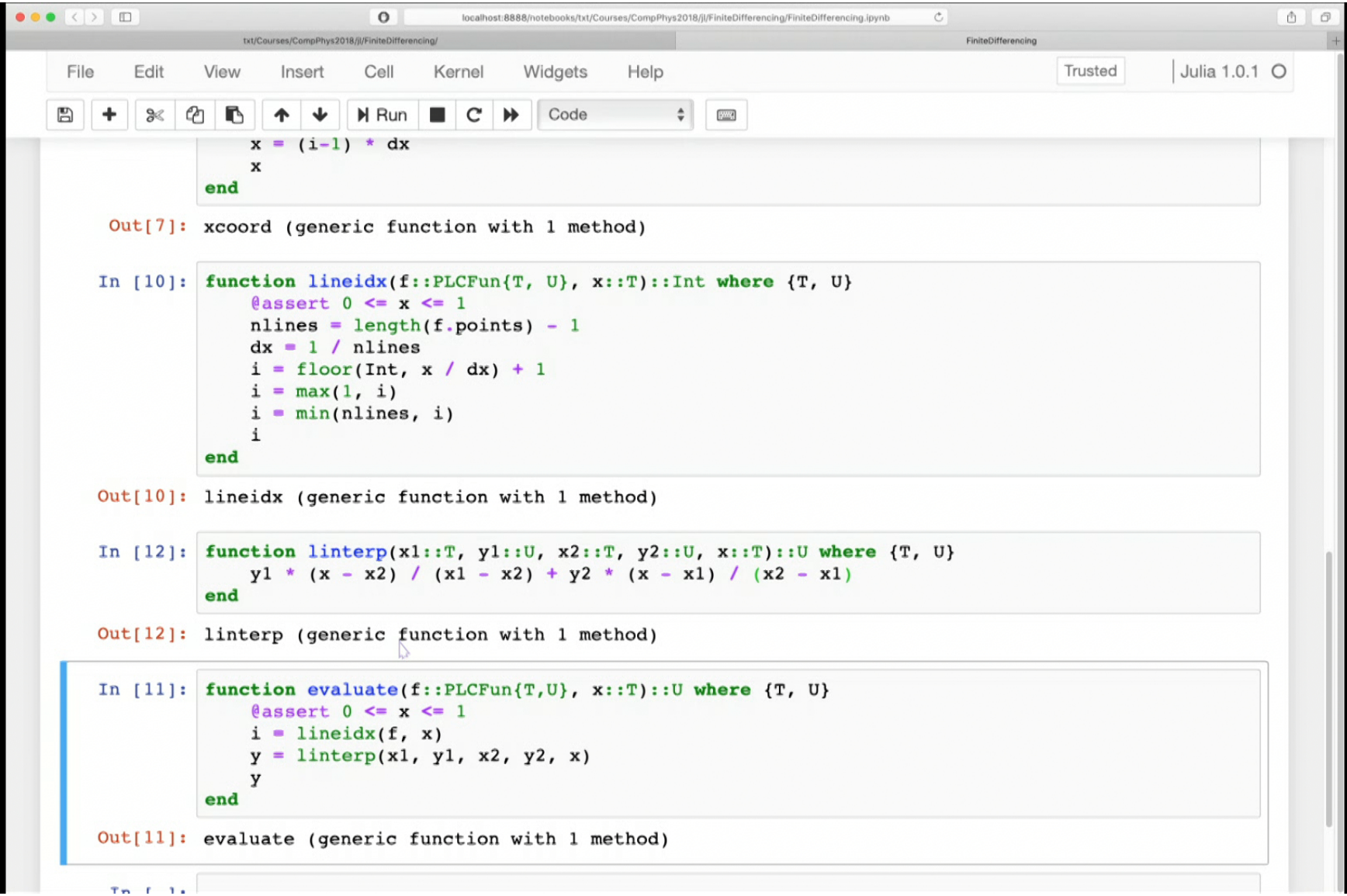

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [4]:
struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [7]:
function xcoord(f::PLCFun{T, U}, i::Int)::T where {T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= nlines +1
dx = 1 / nlines
x = (i-1) * dx
x
end
Out[7]: xcoord (generic function with 1 method)
In [ ]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
x = (i-1) * dx
end
In [5]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
...
end

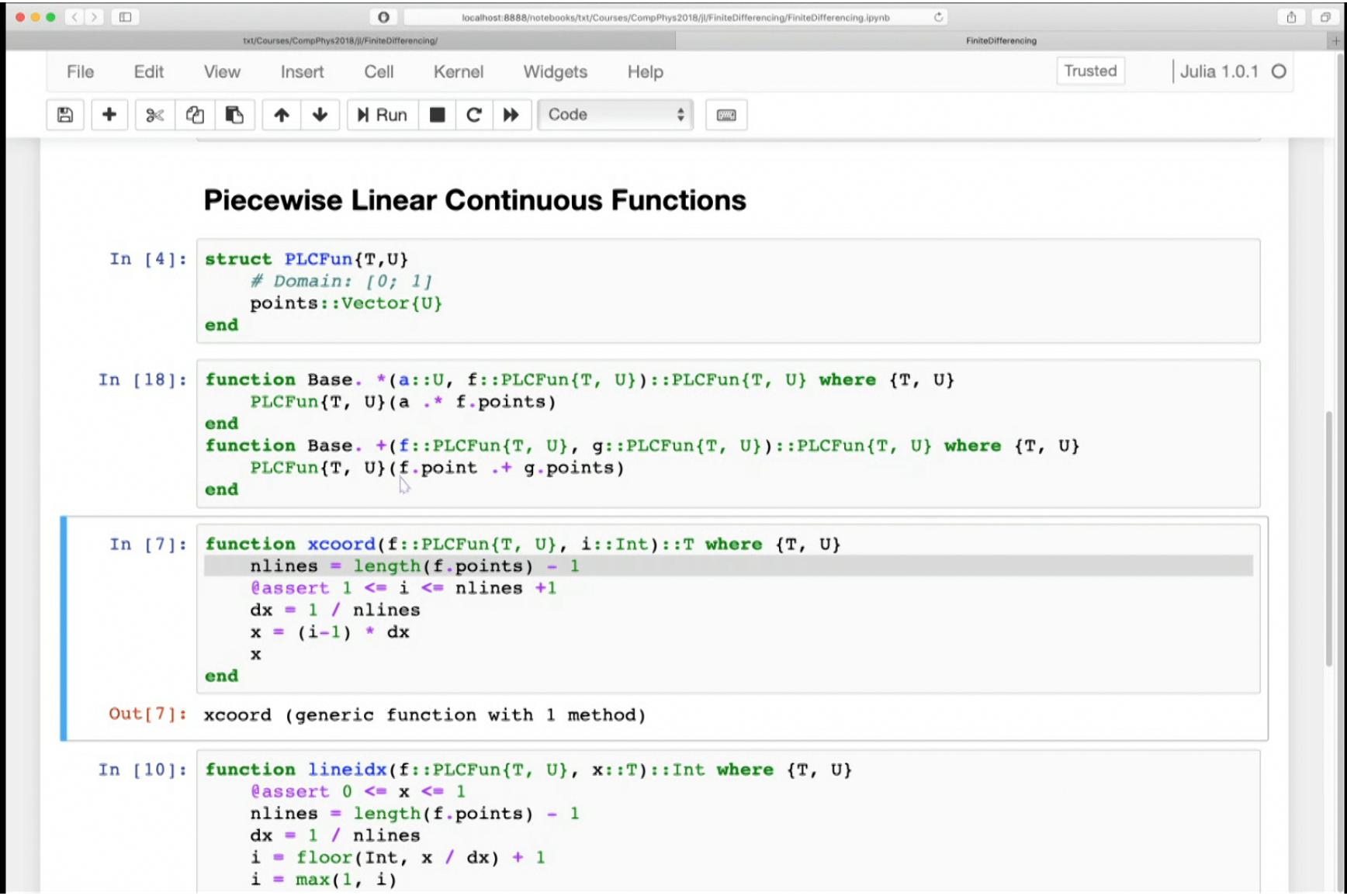

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
Piecewise Linear Continuous Functions
In [4]:
struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [7]:
function xcoord(f::PLCFun{T, U}, i::Int)::T where {T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= nlines +1
dx = 1 / nlines
x = (i-1) * dx
x
end
Out[7]: xcoord (generic function with 1 method)
In [ ]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
dx = 1 / nlines
i = x / dx + 1
end
In [5]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1

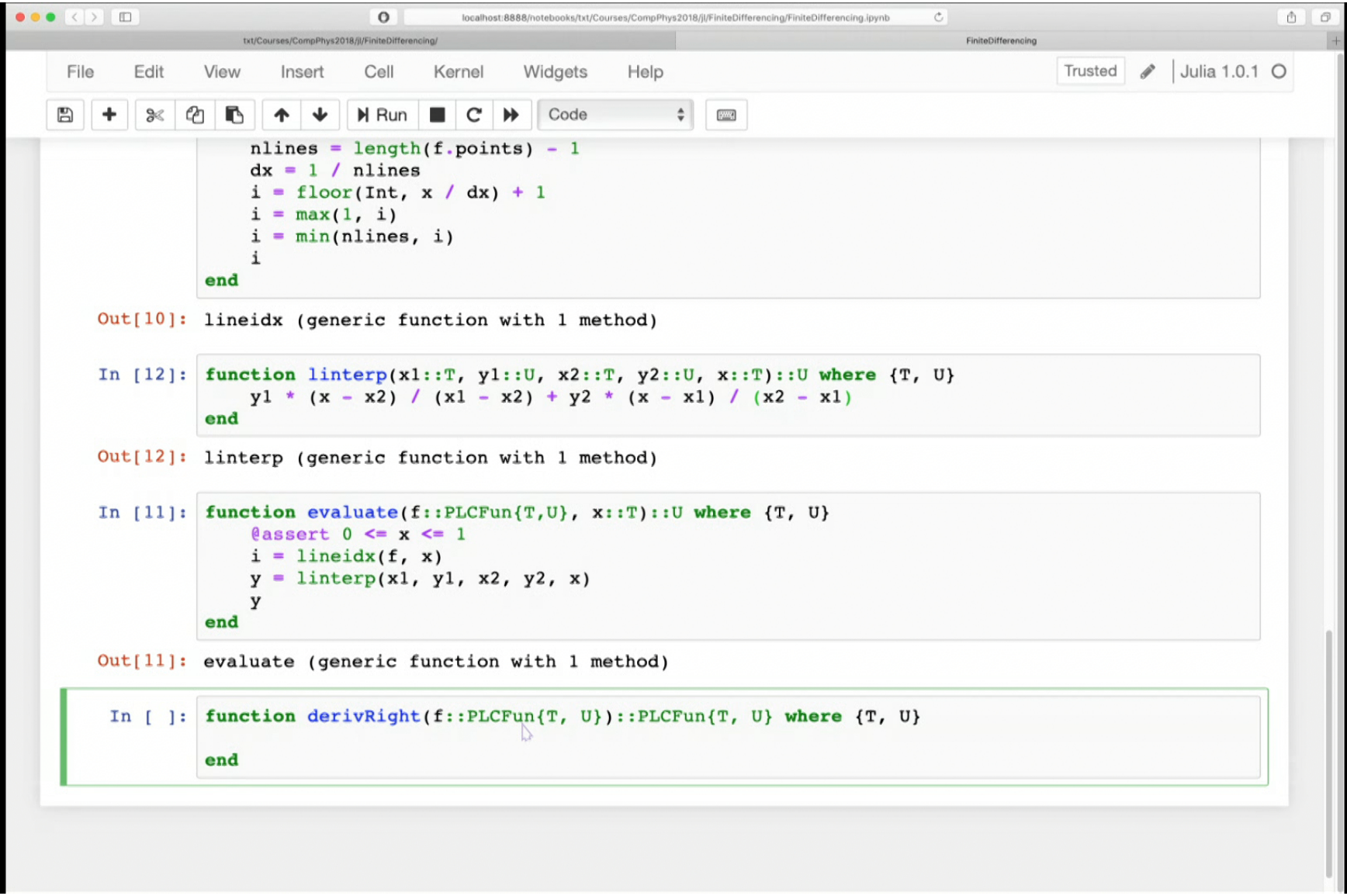

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
In [3]:
]activate .
Piecewise Linear Continuous Functions
In [4]:
struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [7]:
function xcoord(f::PLCFun{T, U}, i::Int)::T where {T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= nlines +1
dx = 1 / nlines
x = (i-1) * dx
x
end
Out[7]:
xcoord (generic function with 1 method)
In [ ]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i
end
In [5]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
...
end

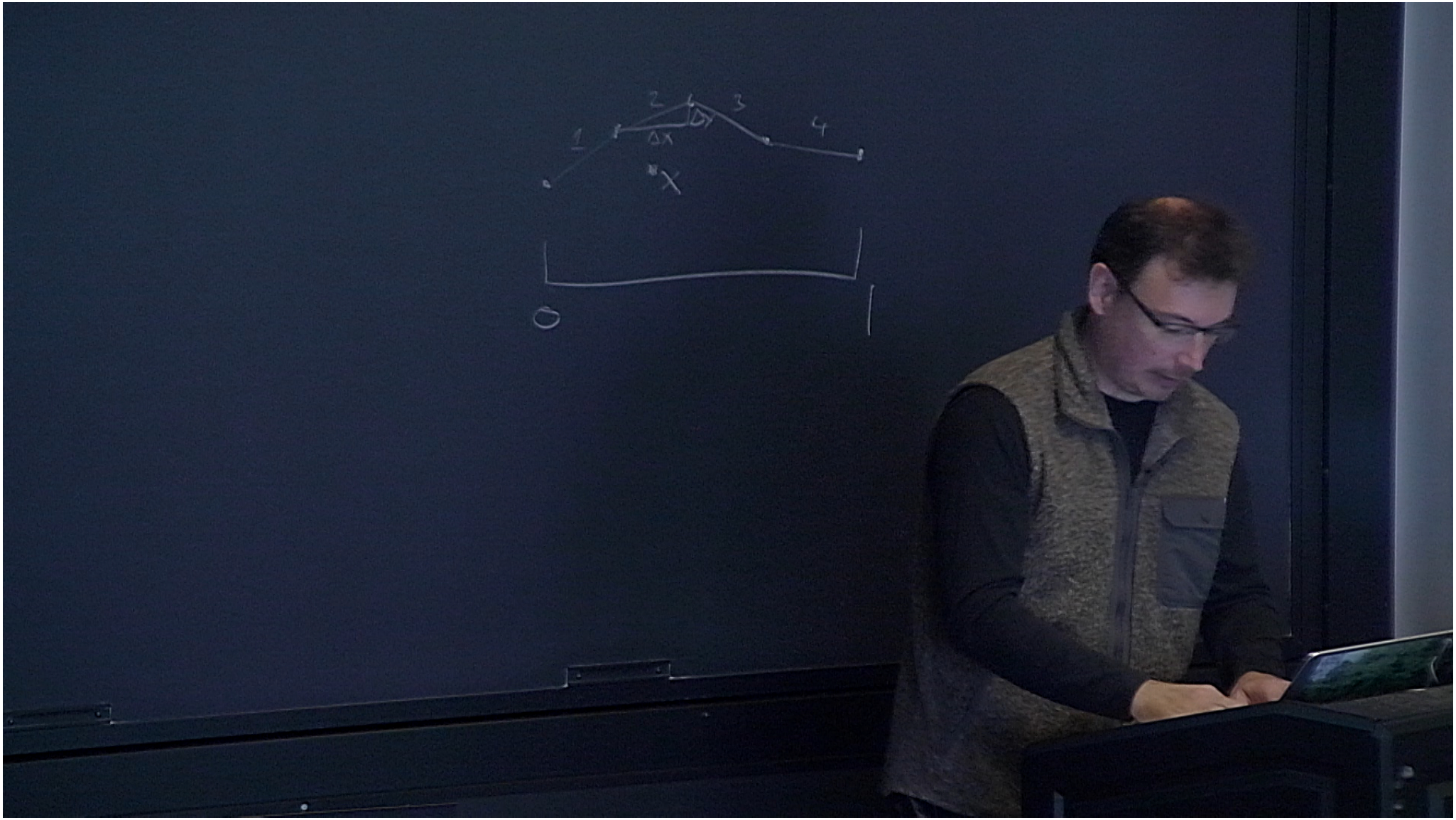

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
In [7]:
function xcoord(f::PLCFun{T, U}, i::Int)::T where {T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= nlines +1
dx = 1 / nlines
x = (i-1) * dx
x
end
Out[7]:
xcoord (generic function with 1 method)
In [9]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)
i = min(nlines, i)
i
end
Out[9]:
lineidx (generic function with 1 method)
In [5]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
...
end
syntax: invalid identifier name "..."

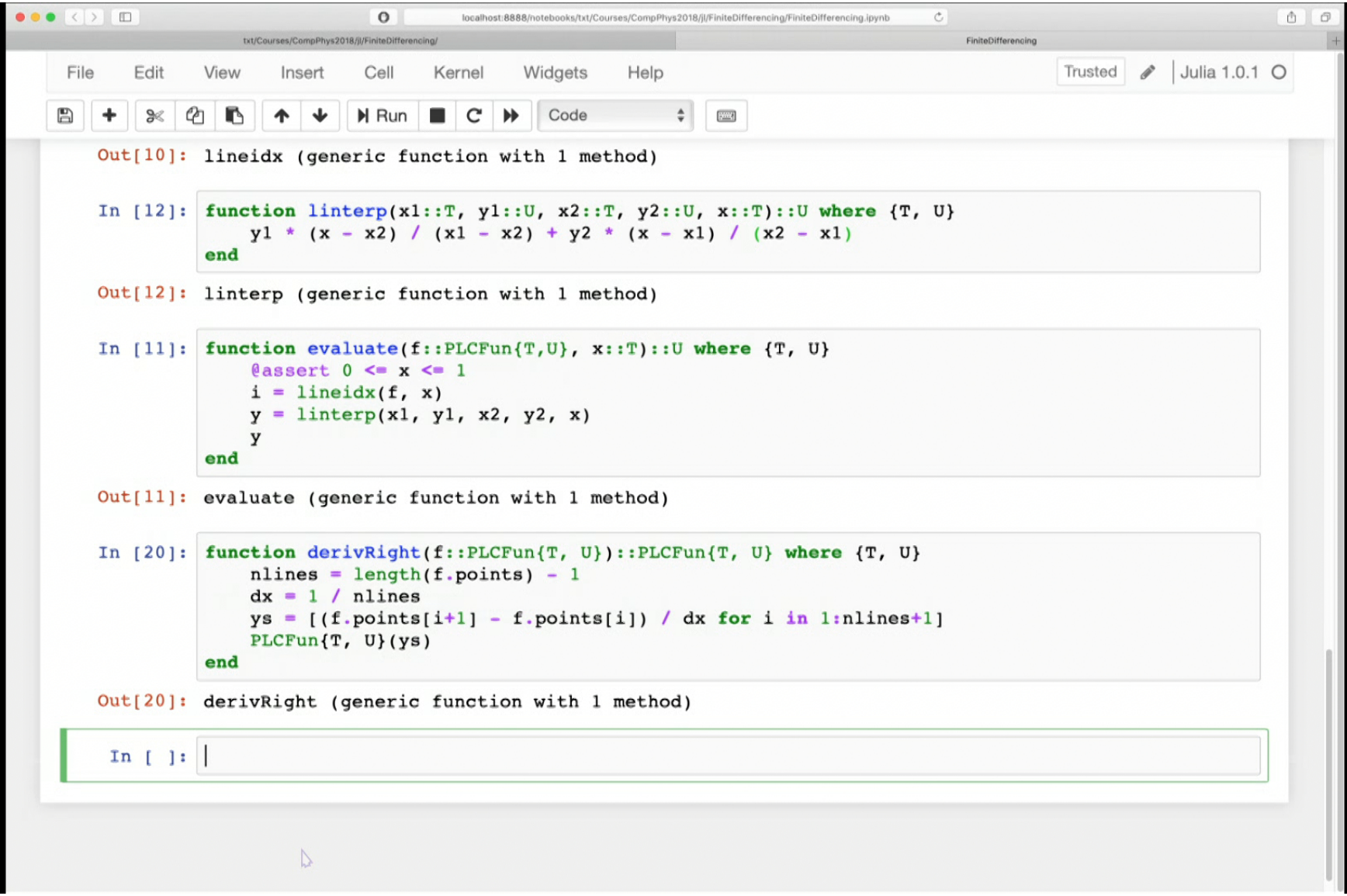

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
x = (i-1) * dx
x
end
Out[7]: xcoord (generic function with 1 method)
In [10]: function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)
i = min(nlines, i)
i
end
Out[10]: lineidx (generic function with 1 method)
In [ ]: function linterp
In [11]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)
In [ ]:

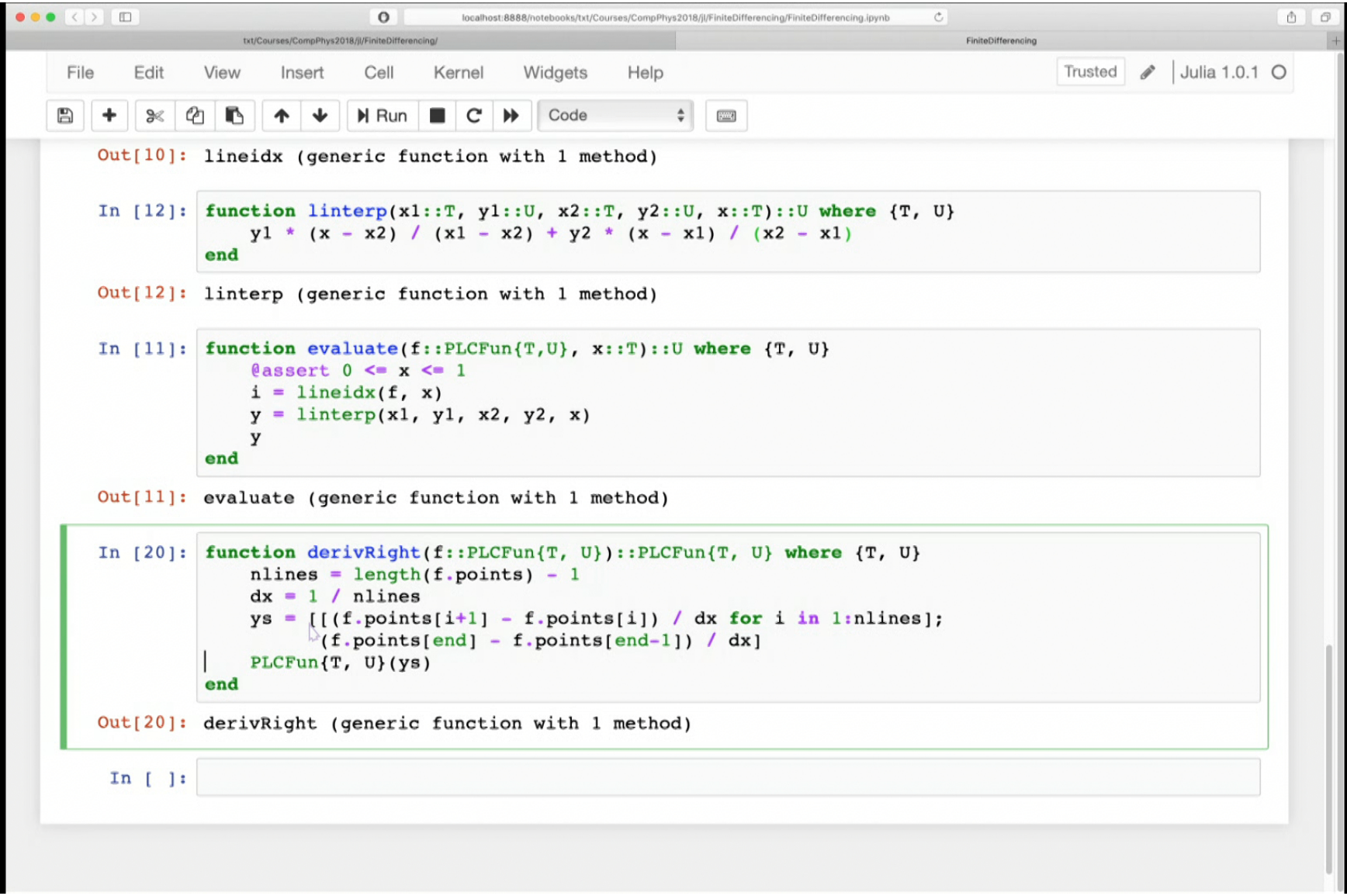

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
x = (i-1) * dx
x
end
Out[7]: xcoord (generic function with 1 method)
In [10]: function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)
i = min(nlines, i)
i
end
Out[10]: lineidx (generic function with 1 method)
In [12]: function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)

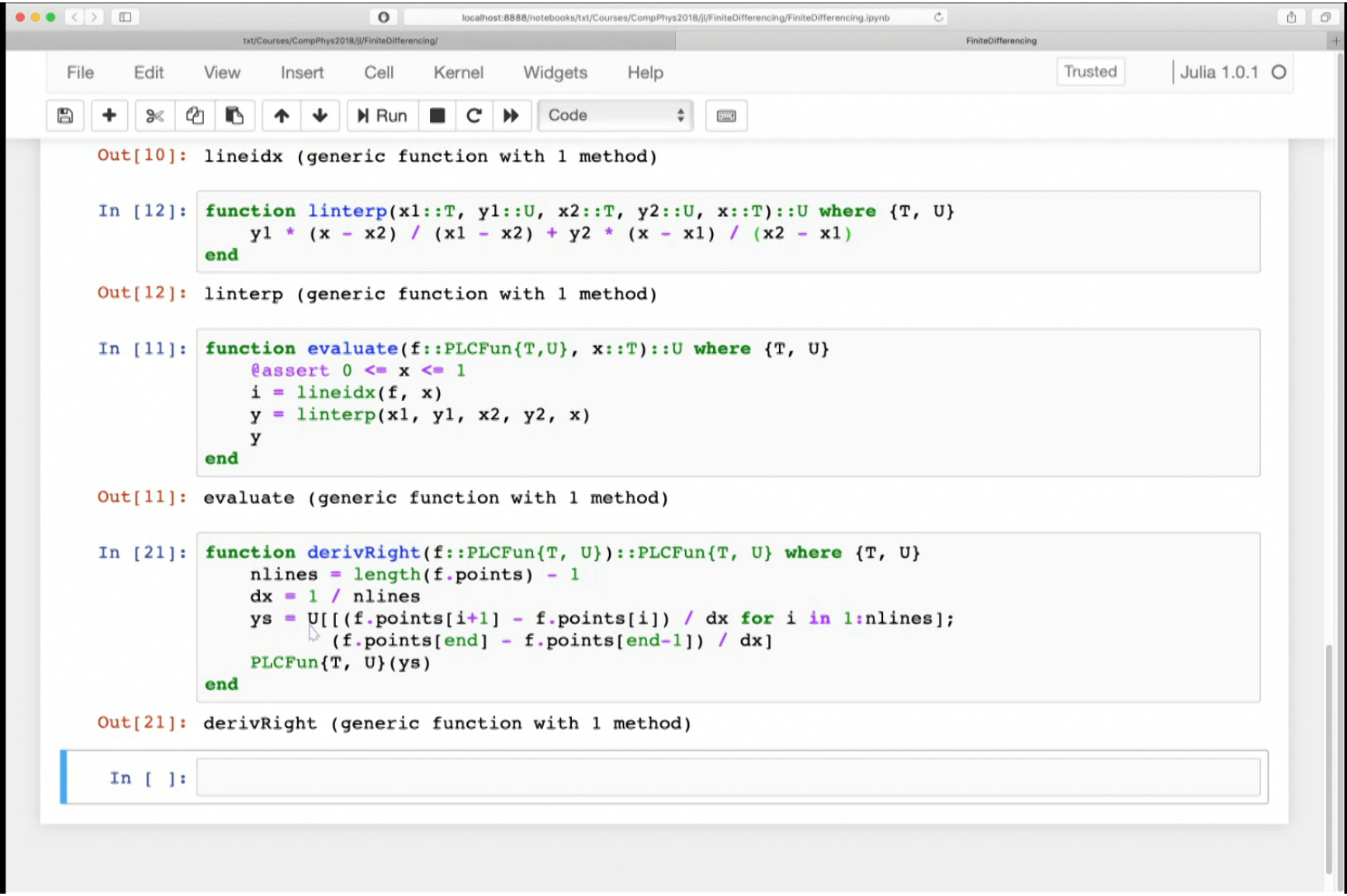

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Piecewise Linear Continuous Functions
In [4]:
struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [18]:
function Base. *(a::U, f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
PLCFun{T, U}(a .* f.points)
end
function Base. +(f::PLCFun{T, U}, g::PLCFun{T, U})::PLCFun{T, U} where {T, U}
PLCFun{T, U}(f.point .+ g.points)
end
In [7]:
function xcoord(f::PLCFun{T, U}, i::Int)::T where {T, U}
nlines = length(f.points) - 1
@assert 1 <= i <= nlines +1
dx = 1 / nlines
x = (i-1) * dx
x
end
Out[7]:
xcoord (generic function with 1 method)
In [10]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)

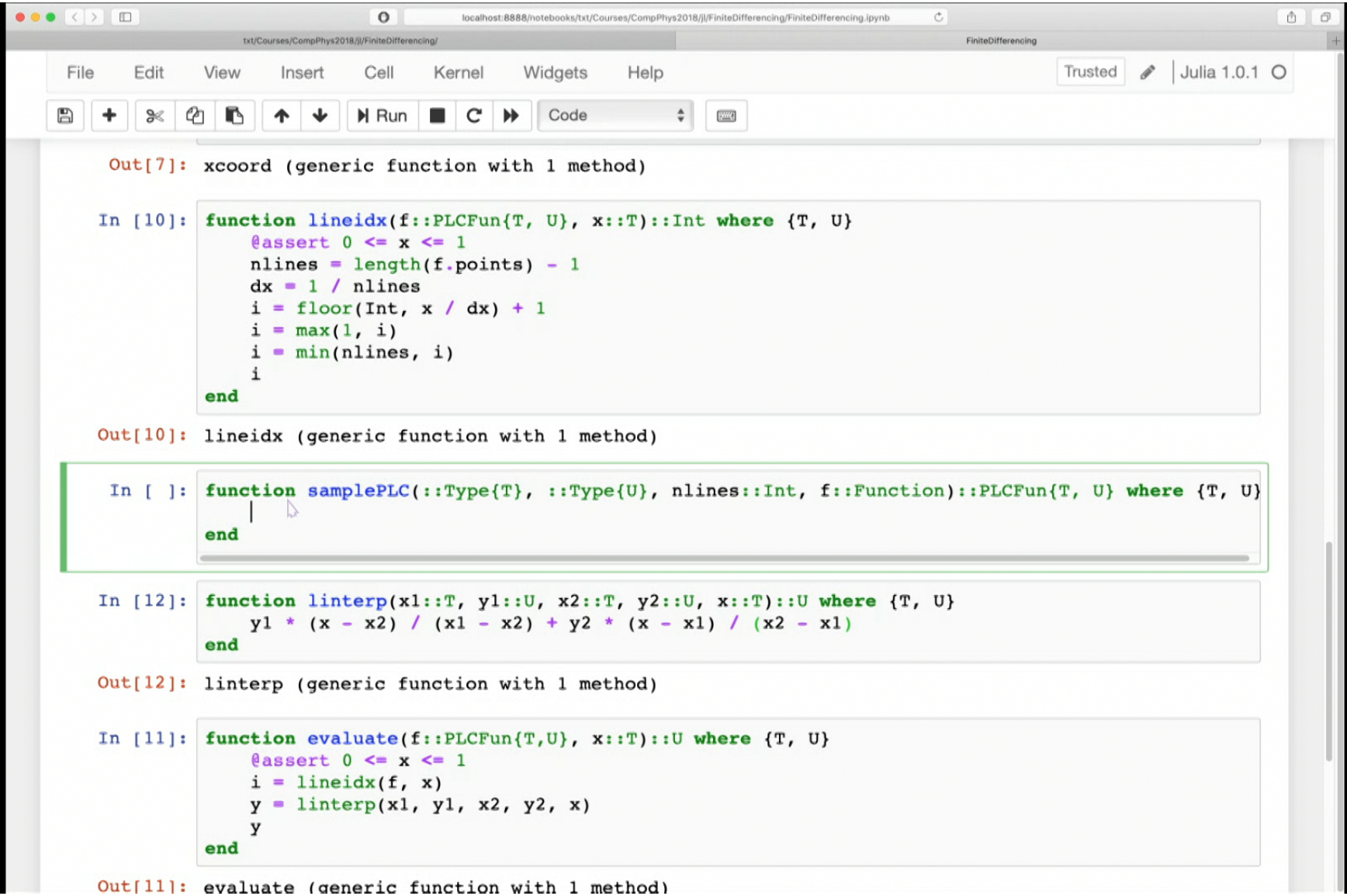

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)
i = min(nlines, i)
i
end
Out[10]: lineidx (generic function with 1 method)
In [12]: function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)
In [ ]: function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
end

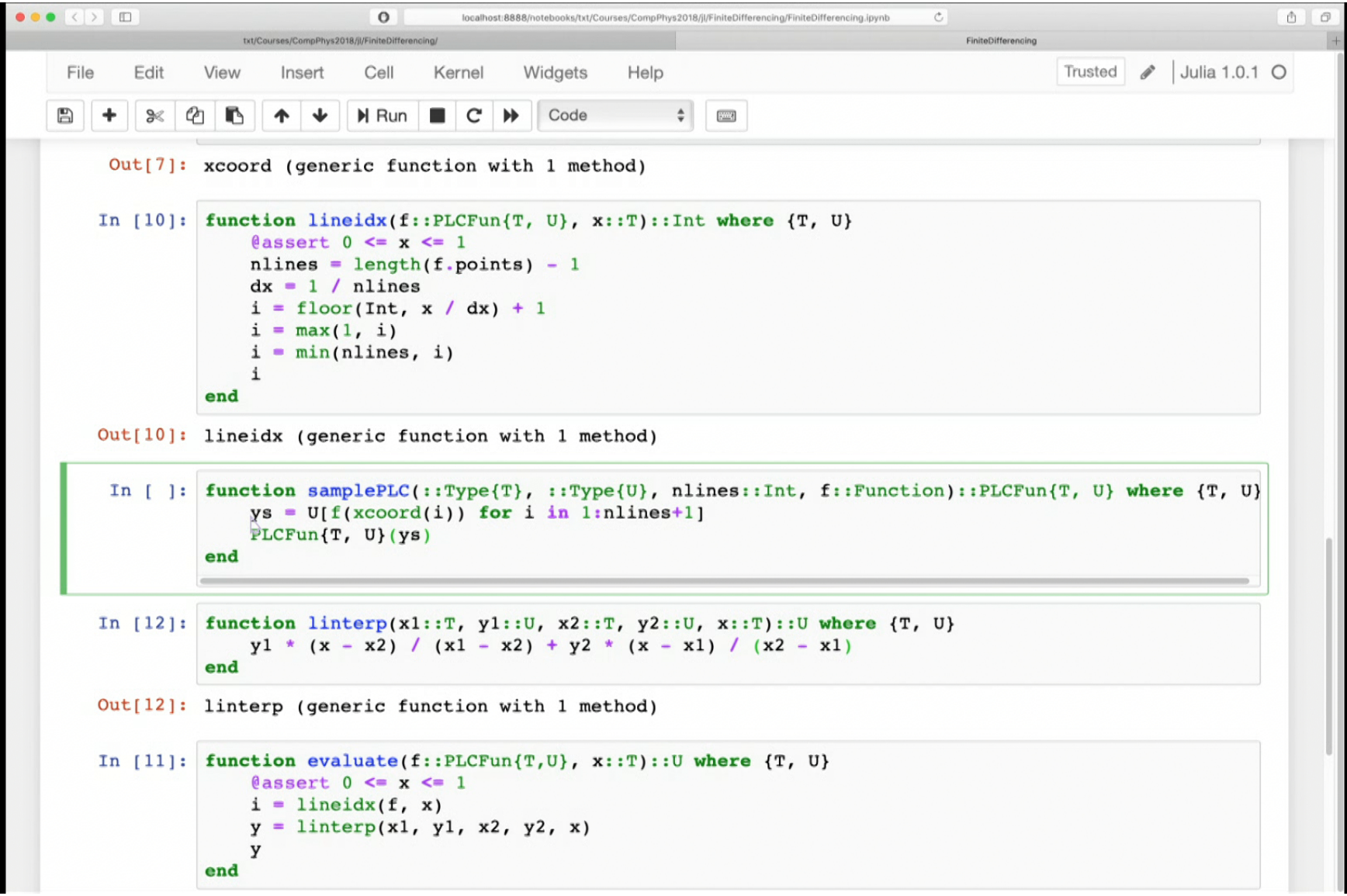

2
3
4
1
Δx
Δy
x
0

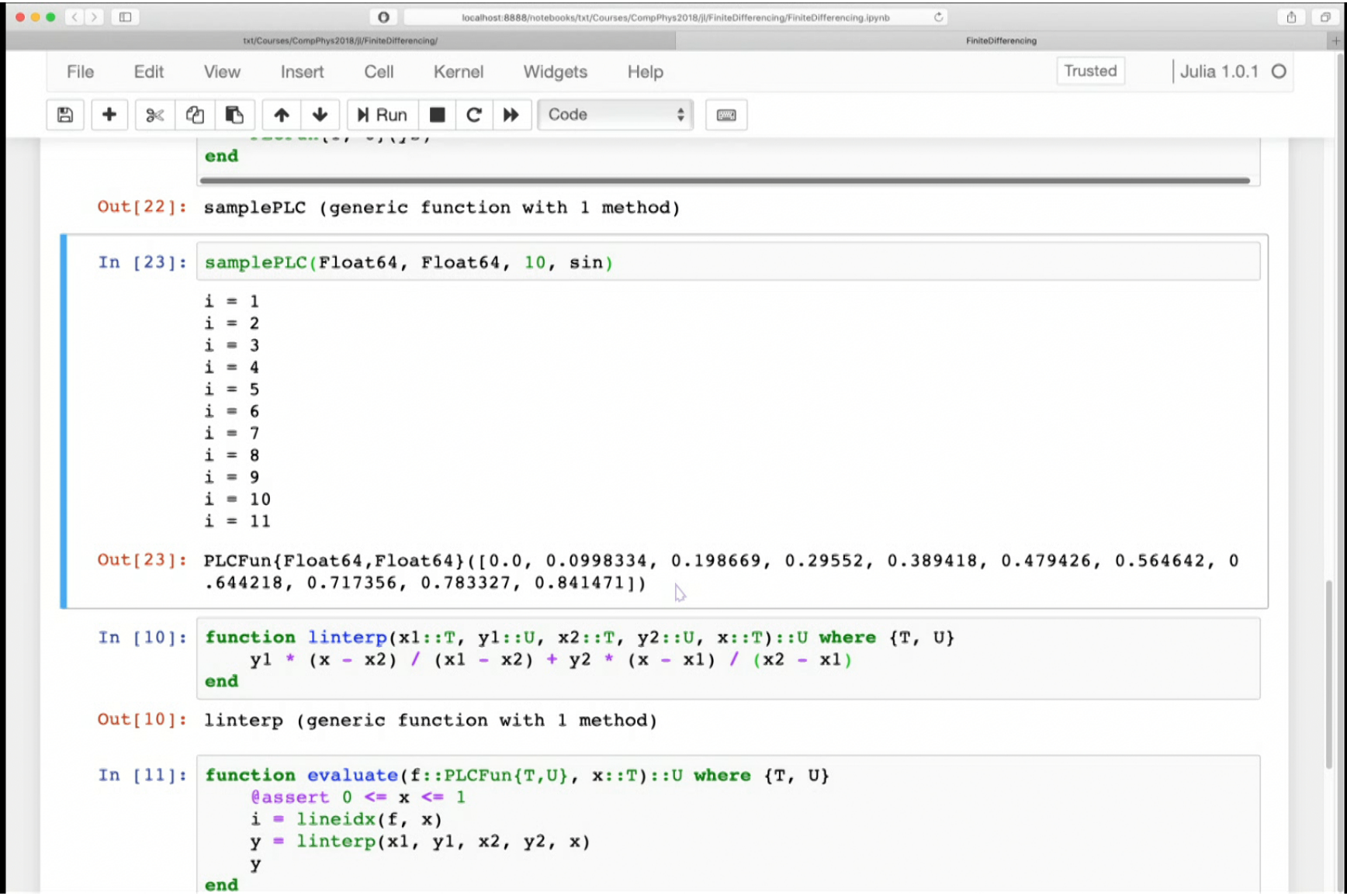

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[10]: lineidx (generic function with 1 method)
In [12]:
function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)
In [20]:
function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = [(f.points[i+1] - f.points[i]) / dx for i in 1:nlines+1]
PLCFun{T, U}(ys)
end
Out[20]: derivRight (generic function with 1 method)
In [ ]:

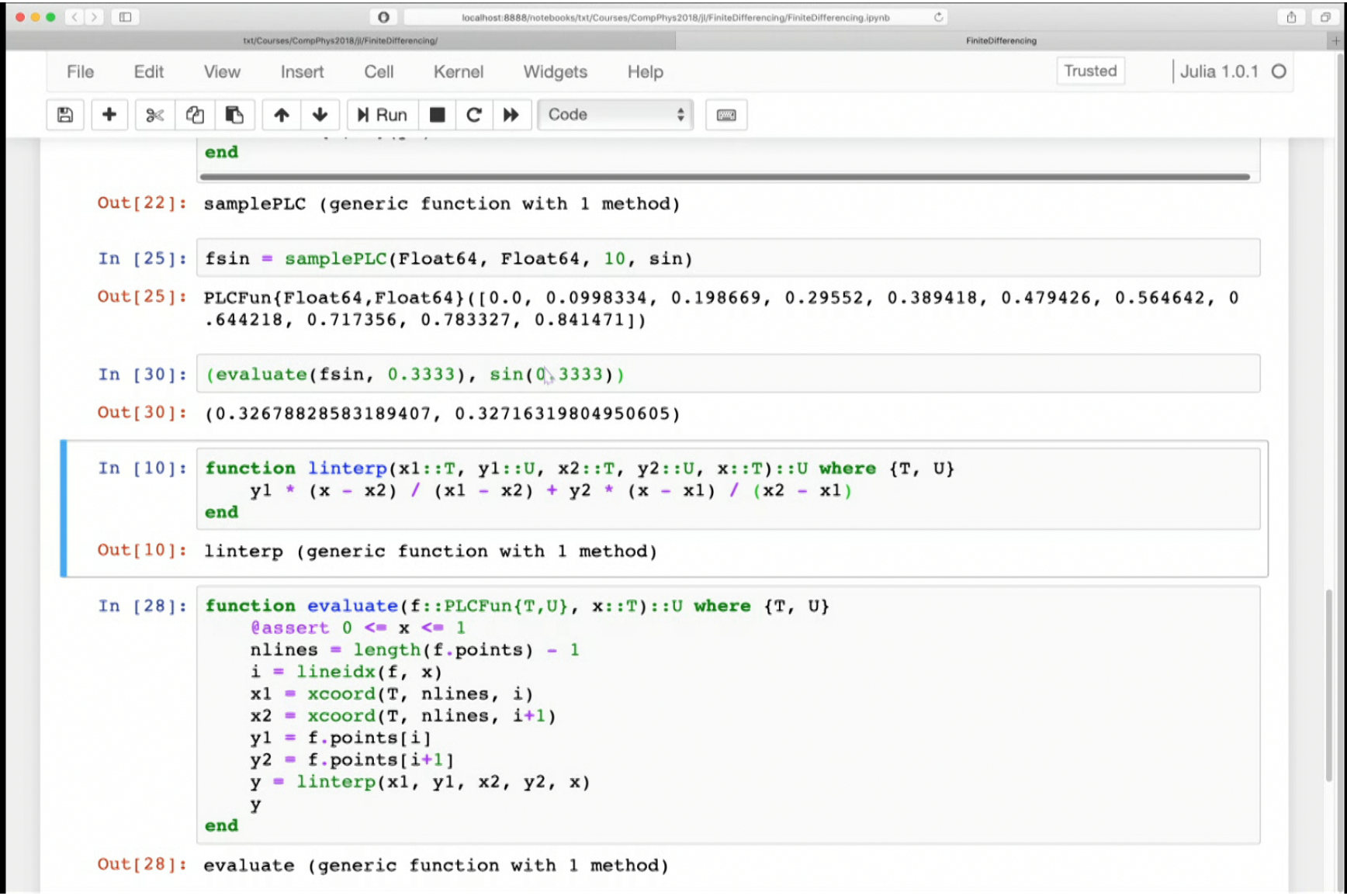

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[10]: lineidx (generic function with 1 method)
In [12]:
function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)
In [20]:
function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = [[(f.points[i+1] - f.points[i]) / dx for i in 1:nlines];
(f.points[end] - f.points[end-1]) / dx]
PLCFun{T, U}(ys)
end
Out[20]: derivRight (generic function with 1 method)
In [ ]:

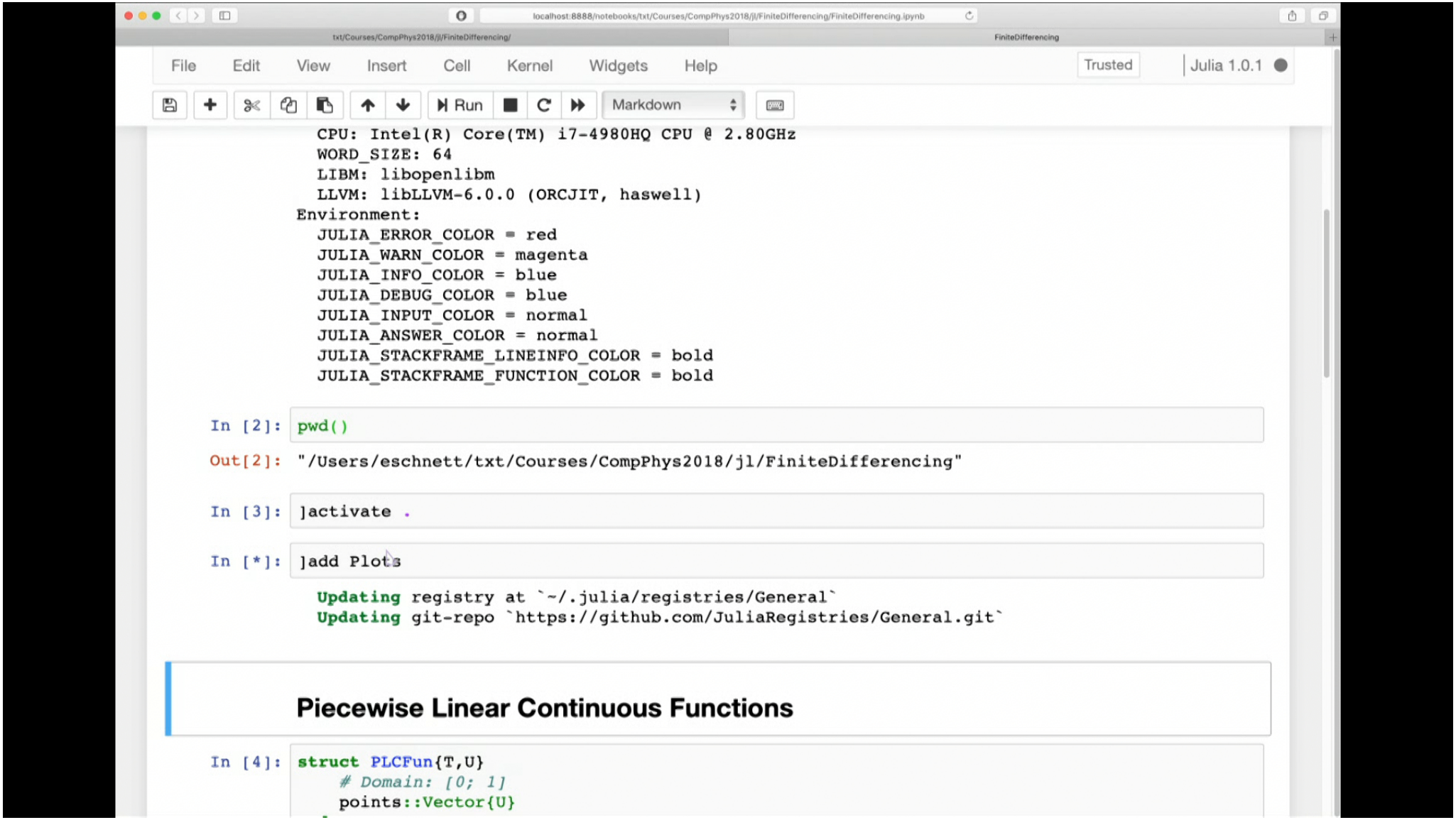

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[10]: lineidx (generic function with 1 method)
In [12]:
function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)
In [21]:
function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = U[[(f.points[i+1] - f.points[i]) / dx for i in 1:nlines];
(f.points[end] - f.points[end-1]) / dx]
PLCFun{T, U}(ys)
end
Out[21]: derivRight (generic function with 1 method)
In [ ]:

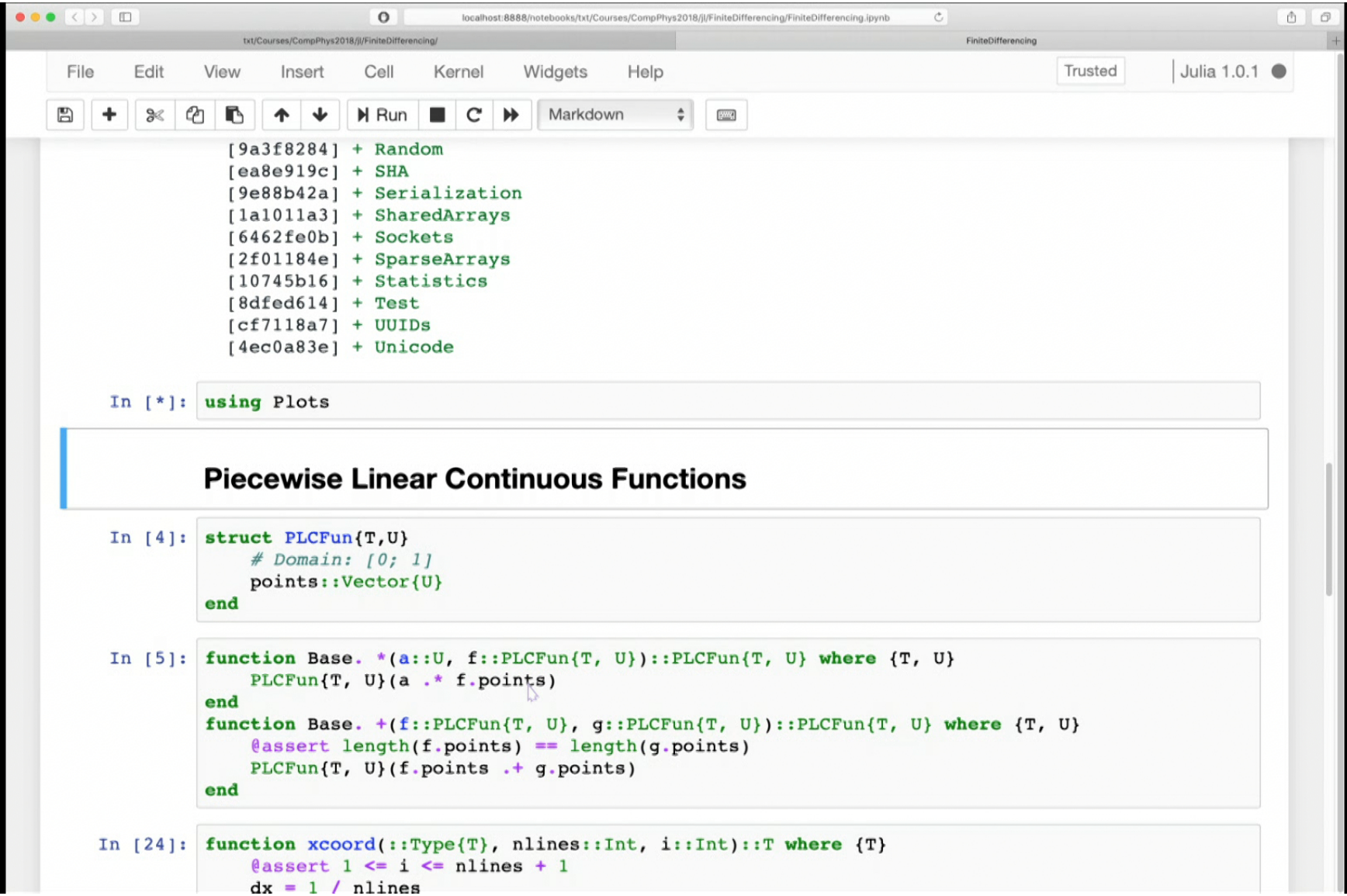

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[7]: xcoord (generic function with 1 method)
In [10]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)
i = min(nlines, i)
i
end
Out[10]: lineidx (generic function with 1 method)
In [ ]:
function samplePLC(::Type{T}, ::Type{U}, nlines::Int, f::Function)::PLCFun{T, U} where {T, U}
end
In [12]:
function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end
Out[11]: evaluate (generic function with 1 method)

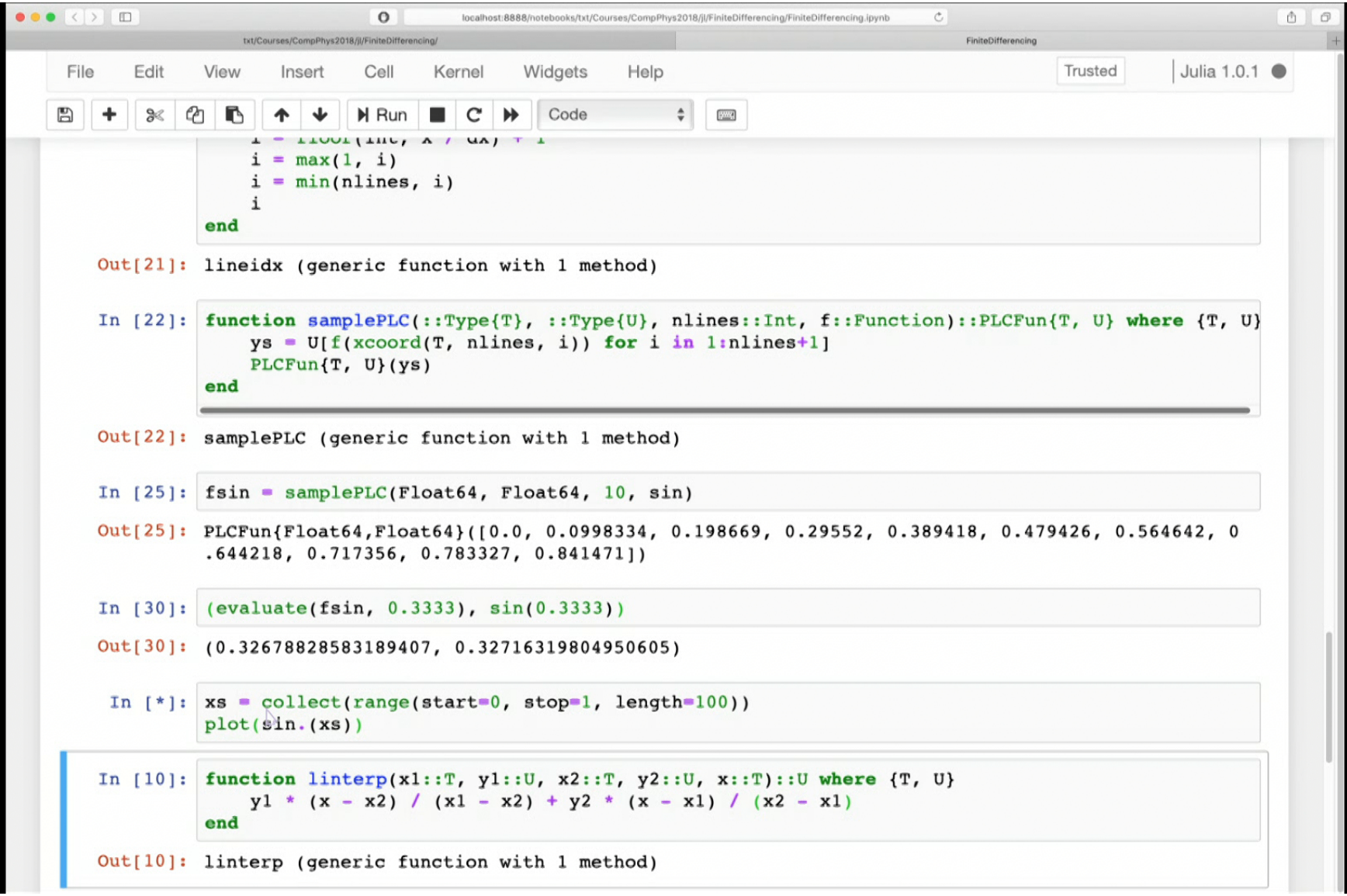

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[7]: xcoord (generic function with 1 method)
In [10]:
function lineidx(f::PLCFun{T, U}, x::T)::Int where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
dx = 1 / nlines
i = floor(Int, x / dx) + 1
i = max(1, i)
i = min(nlines, i)
i
end
Out[10]: lineidx (generic function with 1 method)
In [ ]:
function samplePLC(::Type{T}, ::Type{U}, nlines::Int, f::Function)::PLCFun{T, U} where {T, U}
ys = U[f(xcoord(i)) for i in 1:nlines+1]
PLCFun{T, U}(ys)
end
In [12]:
function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[12]: linterp (generic function with 1 method)
In [11]:
function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end

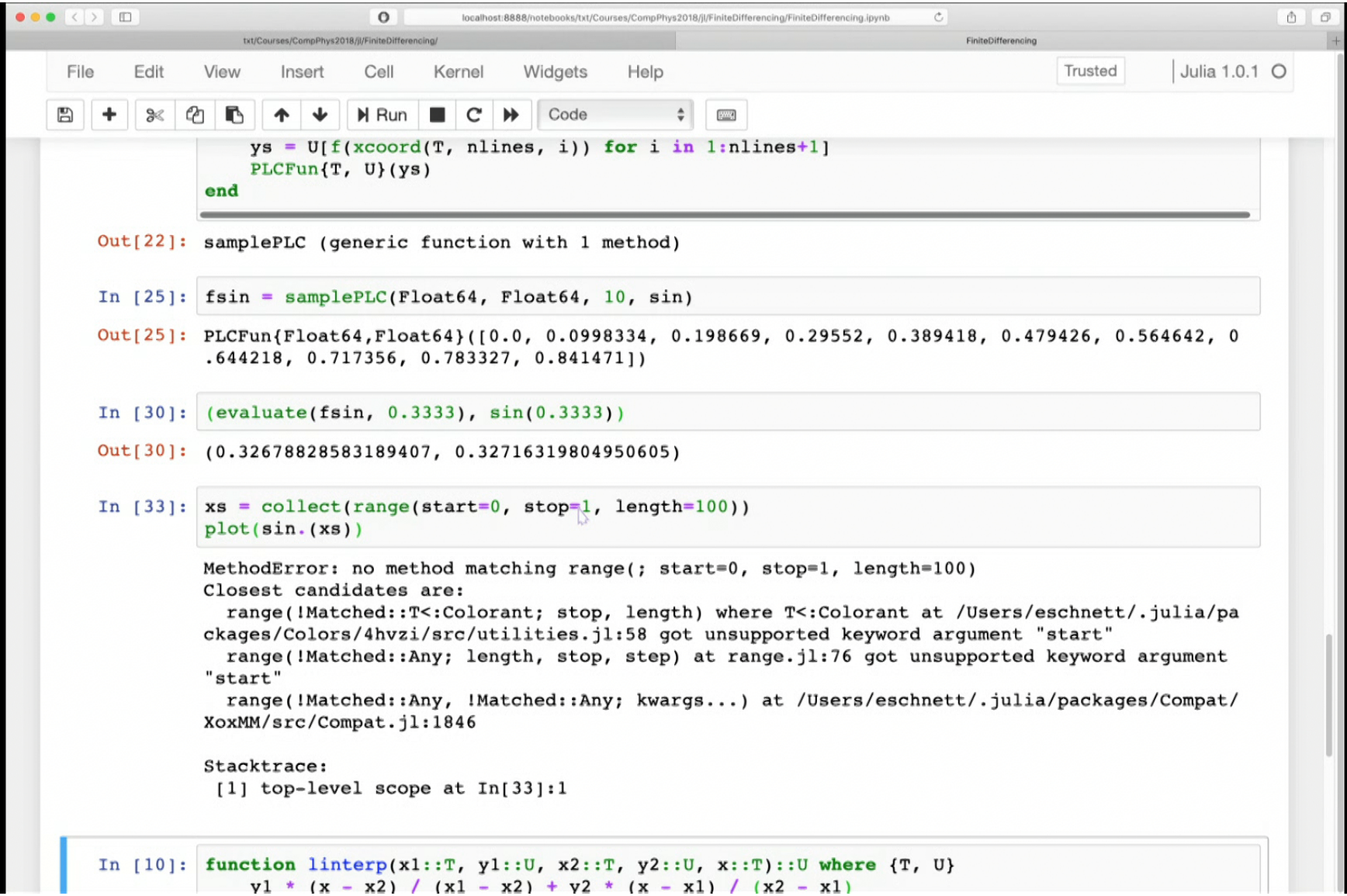

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
end
Out[22]: samplePLC (generic function with 1 method)
In [23]: samplePLC(Float64, Float64, 10, sin)
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
i = 11
Out[23]: PLCFun{Float64,Float64}([0.0, 0.0998334, 0.198669, 0.29552, 0.389418, 0.479426, 0.564642, 0
.644218, 0.717356, 0.783327, 0.841471])
In [10]: function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[10]: linterp (generic function with 1 method)
In [11]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
i = lineidx(f, x)
y = linterp(x1, y1, x2, y2, x)
y
end


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
end
Out[22]: samplePLC (generic function with 1 method)
In [25]: fsin = samplePLC(Float64, Float64, 10, sin)
Out[25]: PLCFun{Float64,Float64}([0.0, 0.0998334, 0.198669, 0.29552, 0.389418, 0.479426, 0.564642, 0.644218, 0.717356, 0.783327, 0.841471])
In [30]: (evaluate(fsin, 0.3333), sin(0.3333))
Out[30]: (0.32678828583189407, 0.32716319804950605)
In [10]: function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[10]: linterp (generic function with 1 method)
In [28]: function evaluate(f::PLCFun{T,U}, x::T)::U where {T, U}
@assert 0 <= x <= 1
nlines = length(f.points) - 1
i = lineidx(f, x)
x1 = xcoord(T, nlines, i)
x2 = xcoord(T, nlines, i+1)
y1 = f.points[i]
y2 = f.points[i+1]
y = linterp(x1, y1, x2, y2, x)
y
end
Out[28]: evaluate (generic function with 1 method)


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Markdown
CPU: Intel(R) Core(TM) i7-4980HQ CPU @ 2.80GHz
WORD_SIZE: 64
LIBM: libopenlibm
LLVM: libLLVM-6.0.0 (ORCJIT, haswell)
Environment:
JULIA_ERROR_COLOR = red
JULIA_WARN_COLOR = magenta
JULIA_INFO_COLOR = blue
JULIA_DEBUG_COLOR = blue
JULIA_INPUT_COLOR = normal
JULIA_ANSWER_COLOR = normal
JULIA_STACKFRAME_LINEINFO_COLOR = bold
JULIA_STACKFRAME_FUNCTION_COLOR = bold
In [2]: pwd()
Out[2]: "/Users/eschnett/txt/Courses/CompPhys2018/jl/FiniteDifferencing"
In [3]:]activate .
In [*]:]add Plots
Updating registry at `~/.julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Piecewise Linear Continuous Functions
In [4]: struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Markdown
[9a3f8284] + Random
[ea8e919c] + SHA
[9e88b42a] + Serialization
[1a1011a3] + SharedArrays
[6462fe0b] + Sockets
[2f01184e] + SparseArrays
[10745b16] + Statistics
[8dfed614] + Test
[cf7118a7] + UUIDs
[4ec0a83e] + Unicode
In [*]: using Plots
Piecewise Linear Continuous Functions
In [4]: struct PLCFun{T,U}
Domain: [0; 1]
points::Vector{U}
end
In [5]: function Base. *(a::U, f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
PLCFun{T, U}(a .* f.points)
end
function Base. +(f::PLCFun{T, U}, g::PLCFun{T, U})::PLCFun{T, U} where {T, U}
@assert length(f.points) == length(g.points)
PLCFun{T, U}(f.points .+ g.points)
end
In [24]: function xcoord(::Type{T}, nlines::Int, i::Int)::T where {T}
@assert 1 <= i <= nlines + 1
dx = 1 / nlines


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
i = max(1, i)
i = min(nlines, i)
i
end
Out[21]: lineidx (generic function with 1 method)
In [22]: function samplePLC(::Type{T}, ::Type{U}, nlines::Int, f::Function)::PLCFun{T, U} where {T, U}
ys = U[f(xcoord(T, nlines, i)) for i in 1:nlines+1]
PLCFun{T, U}(ys)
end
Out[22]: samplePLC (generic function with 1 method)
In [25]: fsin = samplePLC(Float64, Float64, 10, sin)
Out[25]: PLCFun{Float64,Float64}([0.0, 0.0998334, 0.198669, 0.29552, 0.389418, 0.479426, 0.564642, 0
.644218, 0.717356, 0.783327, 0.841471])
In [30]: (evaluate(fsin, 0.3333), sin(0.3333))
Out[30]: (0.32678828583189407, 0.32716319804950605)
In [*]: xs = collect(range(start=0, stop=1, length=100))
plot(sin.(xs))
In [10]: function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
Out[10]: linterp (generic function with 1 method)


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
ys = U[f(xcoord(T, nlines, i)) for i in 1:nlines+1]
PLCFun{T, U}(ys)
end
Out[22]: samplePLC (generic function with 1 method)
In [25]: fsin = samplePLC(Float64, Float64, 10, sin)
Out[25]: PLCFun{Float64,Float64}([0.0, 0.0998334, 0.198669, 0.29552, 0.389418, 0.479426, 0.564642, 0.644218, 0.717356, 0.783327, 0.841471])
In [30]: (evaluate(fsin, 0.3333), sin(0.3333))
Out[30]: (0.32678828583189407, 0.32716319804950605)
In [33]: xs = collect(range(start=0, stop=1, length=100))
plot(sin.(xs))
MethodError: no method matching range(; start=0, stop=1, length=100)
Closest candidates are:
range(!Matched::T<:Colorant; stop, length) where T<:Colorant at /Users/eschnett/.julia/packages/Colors/4hvzi/src/utilities.jl:58 got unsupported keyword argument "start"
range(!Matched::Any; length, stop, step) at range.jl:76 got unsupported keyword argument "start"
range(!Matched::Any, !Matched::Any; kwargs...) at /Users/eschnett/.julia/packages/Compat/XoxMM/src/Compat.jl:1846
Stacktrace:
[1] top-level scope at In[33]:1
In [10]: function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

Run | Code

Out[21]:
```
lineidx (generic function with 1 method)
```

In [22]:
```
function samplePLC(::Type{T}, ::Type{U}, nlines::Int, f::Function)::PLCFun{T, U} where {T, U}
    ys = U[f(xcoord(T, nlines, i)) for i in 1:nlines+1]
    PLCFun{T, U}(ys)
end
```

Out[22]:
```
samplePLC (generic function with 1 method)
```

In [25]:
```
fsin = samplePLC(Float64, Float64, 10, sin)
```

Out[25]:
```
PLCFun{Float64,Float64}([0.0, 0.0998334, 0.198669, 0.29552, 0.389418, 0.479426, 0.564642, 0
.644218, 0.717356, 0.783327, 0.841471])
```

In [30]:
```
(evaluate(fsin, 0.3333), sin(0.3333))
```

Out[30]:
```
(0.32678828583189407, 0.32716319804950605)
```

In [36]:
```
xs = collect(range(0, stop=1, length=100))
plot(sin.(xs))
```

Out[36]:

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

Run | Code

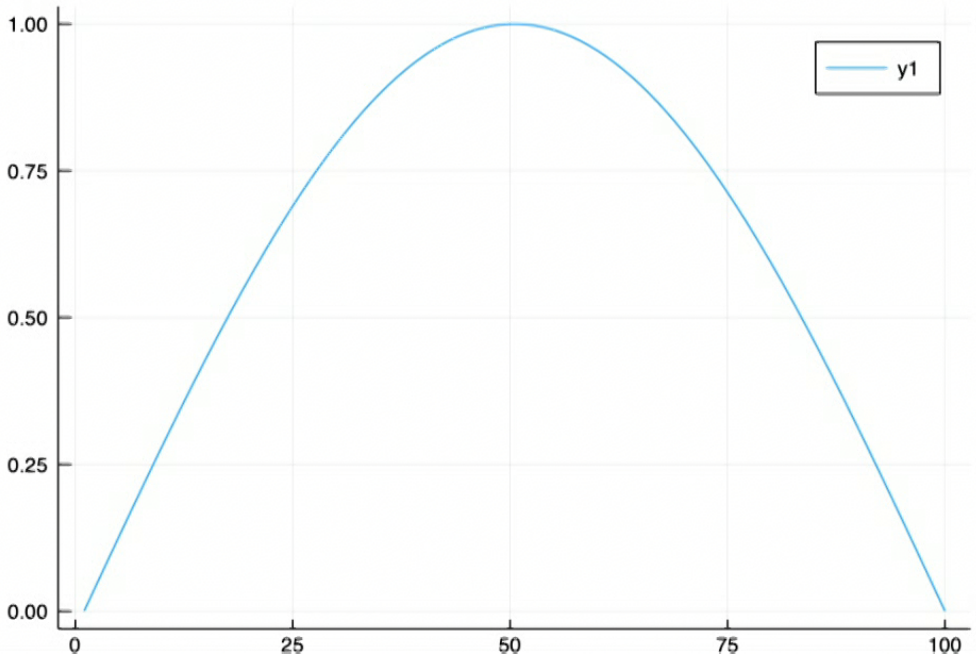
Out[37]:
```
PLCFun{Float64,Float64}([0.0, 0.309017, 0.587785, 0.809017, 0.951057, 1.0, 0.951057, 0.8090
17, 0.587785, 0.309017, 0.0])
```

In [38]:
```
(evaluate(fsin, 0.3333), sin(0.3333))
```

Out[38]:
```
(0.8563161551743761, 0.32716319804950605)
```

In [39]:
```
xs = collect(range(0, stop=1, length=100))
plot(sinpi.(xs),)
```

Out[39]:



localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

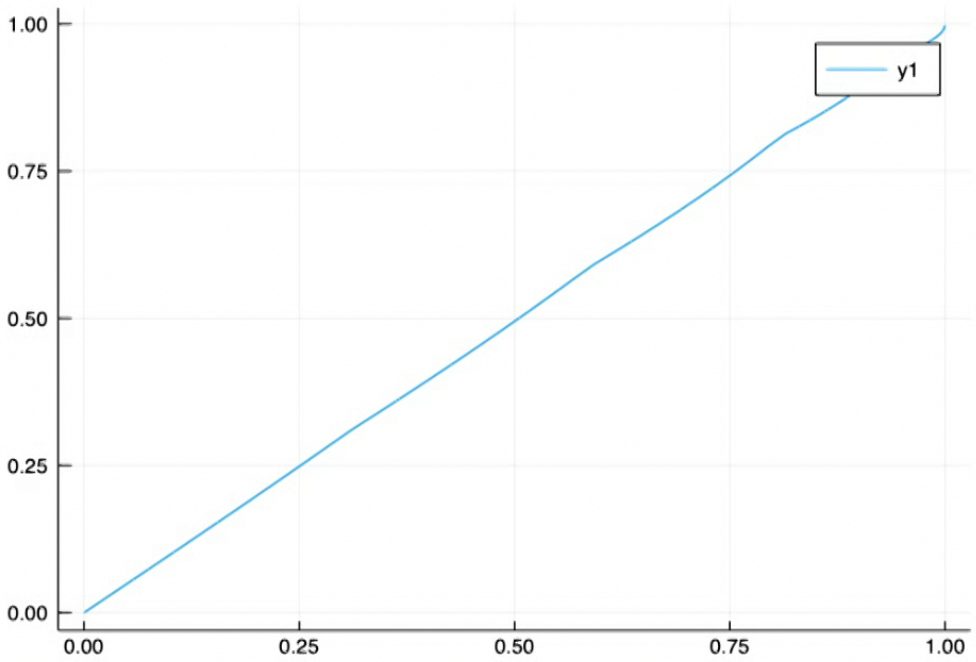
txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

Run | Code

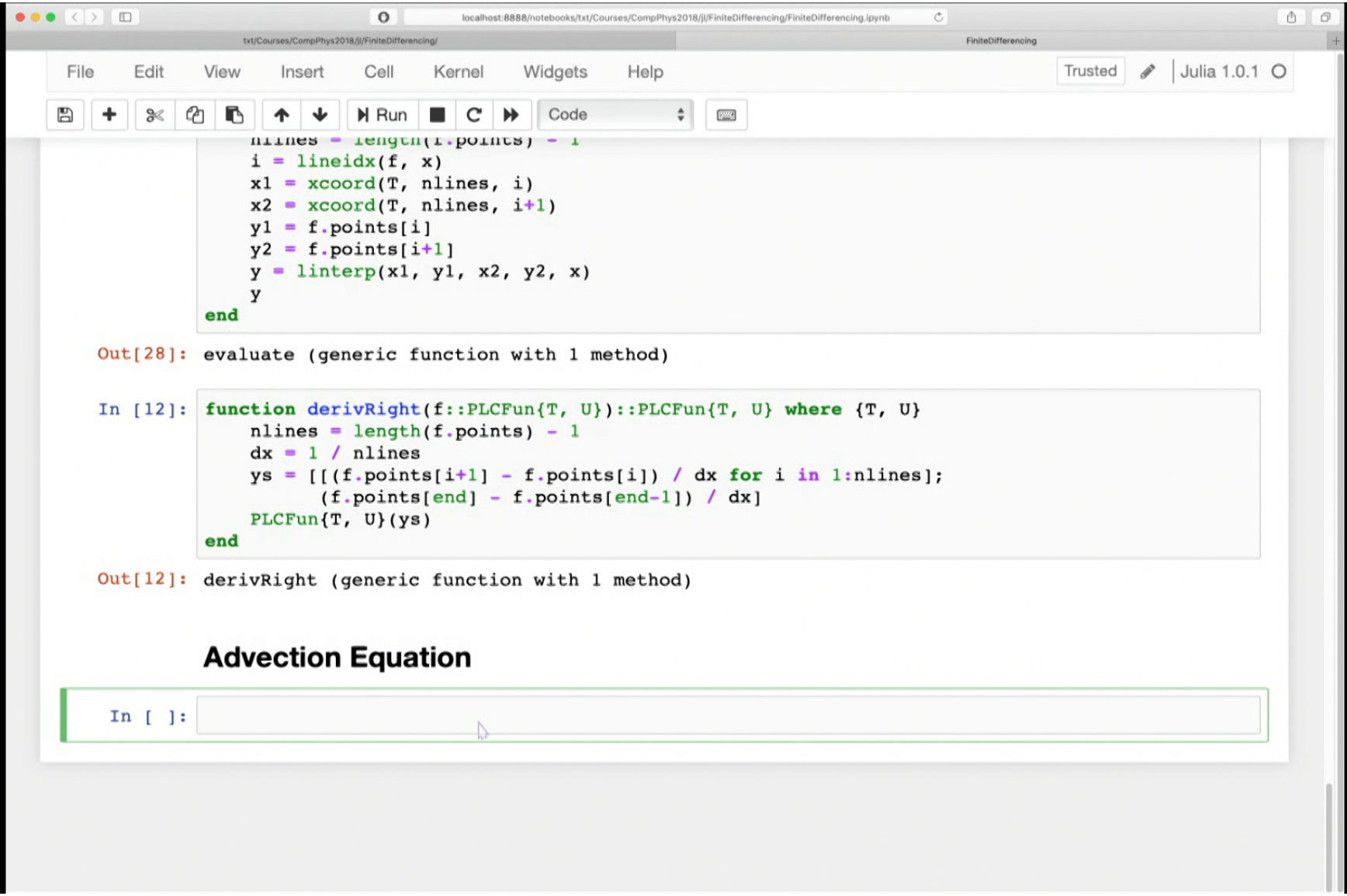
In [40]:
```
xs = collect(range(0, stop=1, length=100))
plot(sinpi.(xs), [evaluate(fsin, x) for x in xs])
```

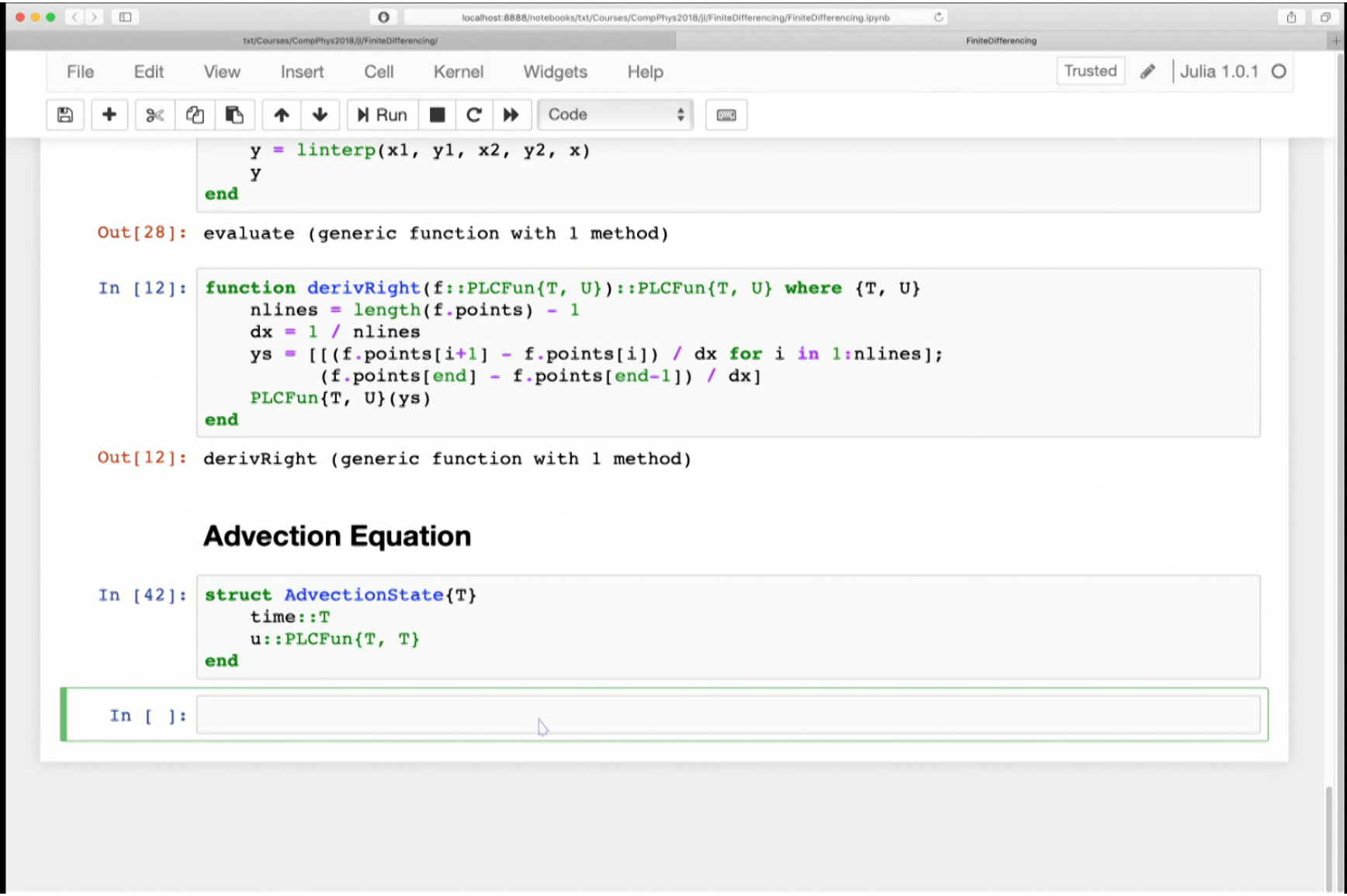
Out[40]:

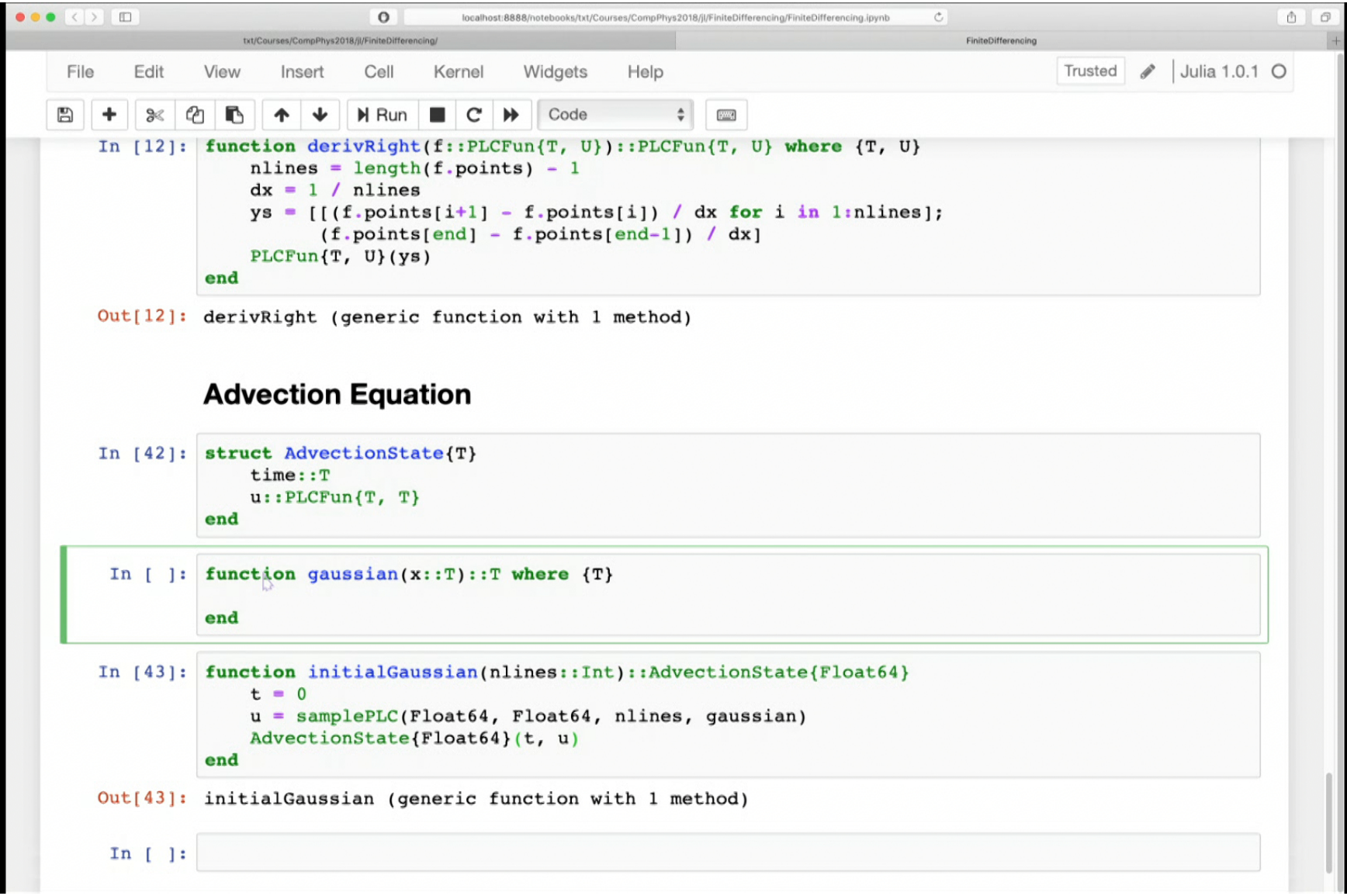


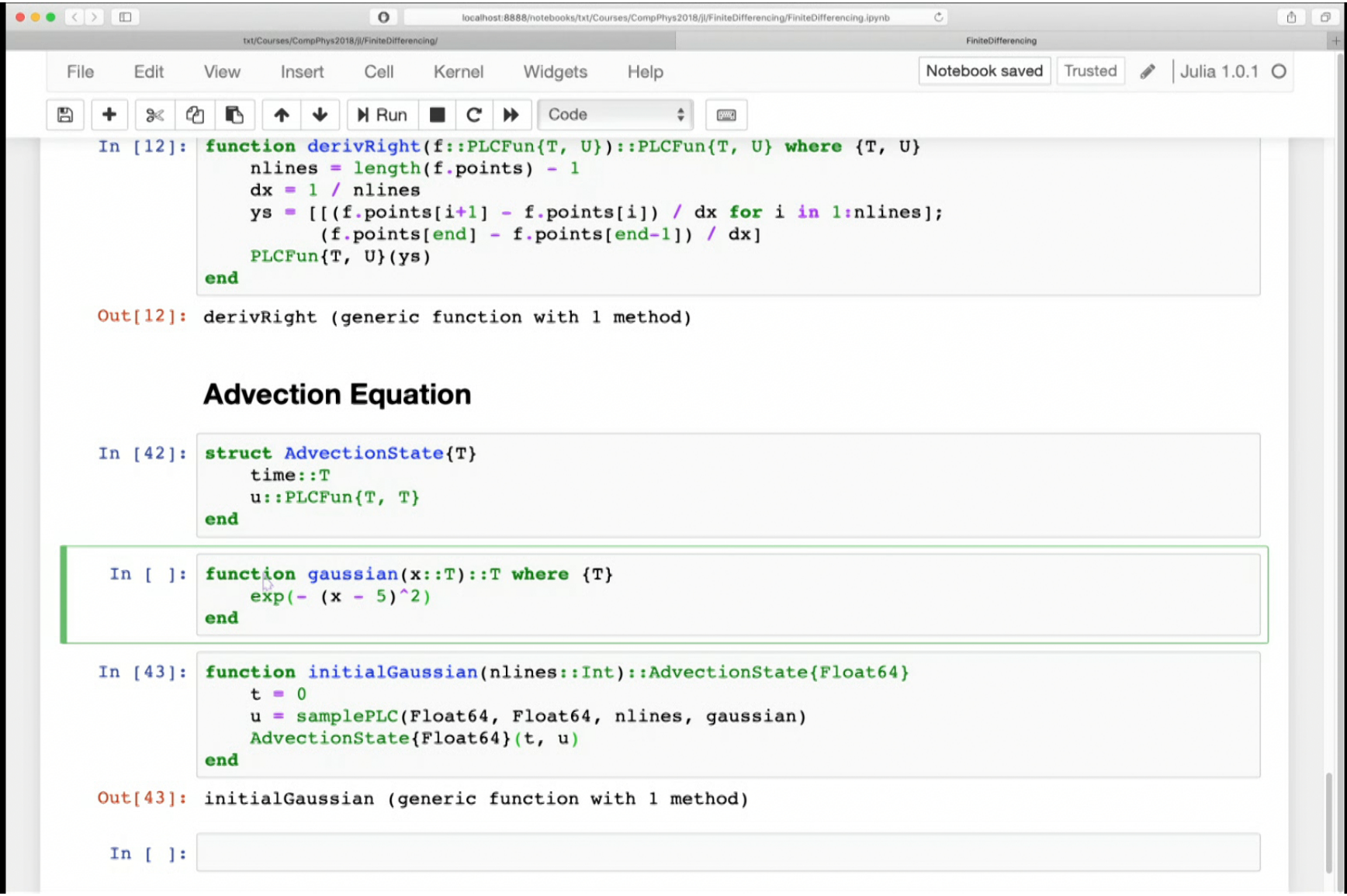
In [10]:
```
function linterp(x1::T, y1::U, x2::T, y2::U, x::T)::U where {T, U}
    y1 * (x - x2) / (x1 - x2) + y2 * (x - x1) / (x2 - x1)
end
```

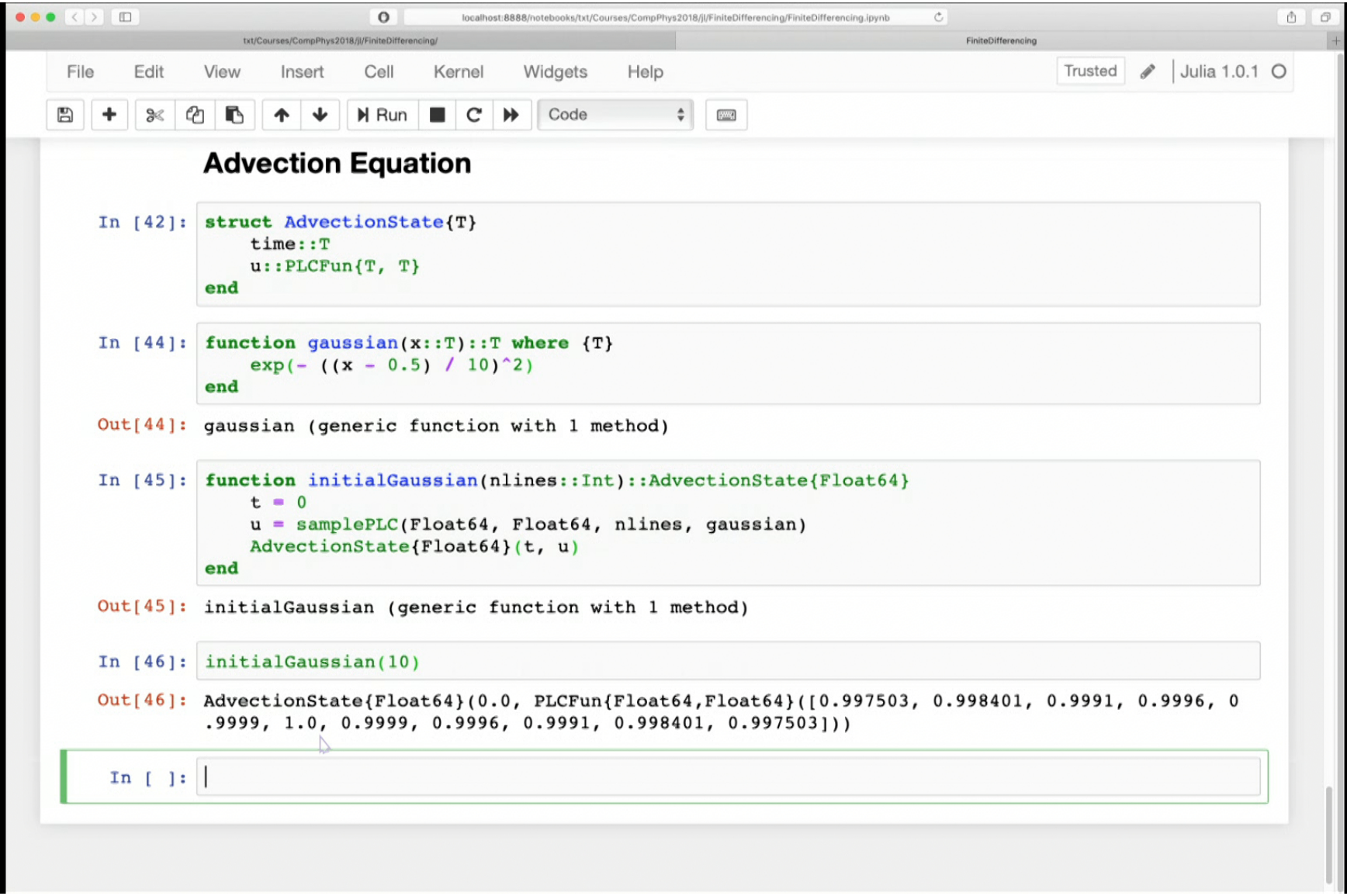
Out[10]: linterp (generic function with 1 method)


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
i = lineidx(f, x)
x1 = xcoord(T, nlines, i)
x2 = xcoord(T, nlines, i+1)
y1 = f.points[i]
y2 = f.points[i+1]
y = linterp(x1, y1, x2, y2, x)
y
end
Out[28]: evaluate (generic function with 1 method)
In [12]: function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = [[(f.points[i+1] - f.points[i]) / dx for i in 1:nlines];
(f.points[end] - f.points[end-1]) / dx]
PLCFun{T, U}(ys)
end
Out[12]: derivRight (generic function with 1 method)
Advection Equation
In [ ]:


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
y = linterp(x1, y1, x2, y2, x)
y
end
Out[28]: evaluate (generic function with 1 method)
In [12]:
function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = [[(f.points[i+1] - f.points[i]) / dx for i in 1:nlines];
(f.points[end] - f.points[end-1]) / dx]
PLCFun{T, U}(ys)
end
Out[12]: derivRight (generic function with 1 method)
Advection Equation
In [42]:
struct AdvectionState{T}
time::T
u::PLCFun{T, T}
end
In [ ]:


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
In [12]:
function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = [[(f.points[i+1] - f.points[i]) / dx for i in 1:nlines];
(f.points[end] - f.points[end-1]) / dx]
PLCFun{T, U}(ys)
end
Out[12]:
derivRight (generic function with 1 method)
Advection Equation
In [42]:
struct AdvectionState{T}
time::T
u::PLCFun{T, T}
end
In [ ]:
function gaussian(x::T)::T where {T}
end
In [43]:
function initialGaussian(nlines::Int)::AdvectionState{Float64}
t = 0
u = samplePLC(Float64, Float64, nlines, gaussian)
AdvectionState{Float64}(t, u)
end
Out[43]:
initialGaussian (generic function with 1 method)
In [ ]:


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Notebook saved Trusted Julia 1.0.1
Run Code
In [12]:
function derivRight(f::PLCFun{T, U})::PLCFun{T, U} where {T, U}
nlines = length(f.points) - 1
dx = 1 / nlines
ys = [[(f.points[i+1] - f.points[i]) / dx for i in 1:nlines];
(f.points[end] - f.points[end-1]) / dx]
PLCFun{T, U}(ys)
end
Out[12]:
derivRight (generic function with 1 method)
Advection Equation
In [42]:
struct AdvectionState{T}
time::T
u::PLCFun{T, T}
end
In [ ]:
function gaussian(x::T)::T where {T}
exp(- (x - 5)^2)
end
In [43]:
function initialGaussian(nlines::Int)::AdvectionState{Float64}
t = 0
u = samplePLC(Float64, Float64, nlines, gaussian)
AdvectionState{Float64}(t, u)
end
Out[43]:
initialGaussian (generic function with 1 method)
In [ ]:


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Advection Equation
In [42]:
struct AdvectionState{T}
time::T
u::PLCFun{T, T}
end
In [44]:
function gaussian(x::T)::T where {T}
exp(- ((x - 0.5) / 10)^2)
end
Out[44]:
gaussian (generic function with 1 method)
In [45]:
function initialGaussian(nlines::Int)::AdvectionState{Float64}
t = 0
u = samplePLC(Float64, Float64, nlines, gaussian)
AdvectionState{Float64}(t, u)
end
Out[45]:
initialGaussian (generic function with 1 method)
In [46]:
initialGaussian(10)
Out[46]:
AdvectionState{Float64}(0.0, PLCFun{Float64,Float64}([0.997503, 0.998401, 0.9991, 0.9996, 0
.9999, 1.0, 0.9999, 0.9996, 0.9991, 0.998401, 0.997503]))
In [ ]:

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

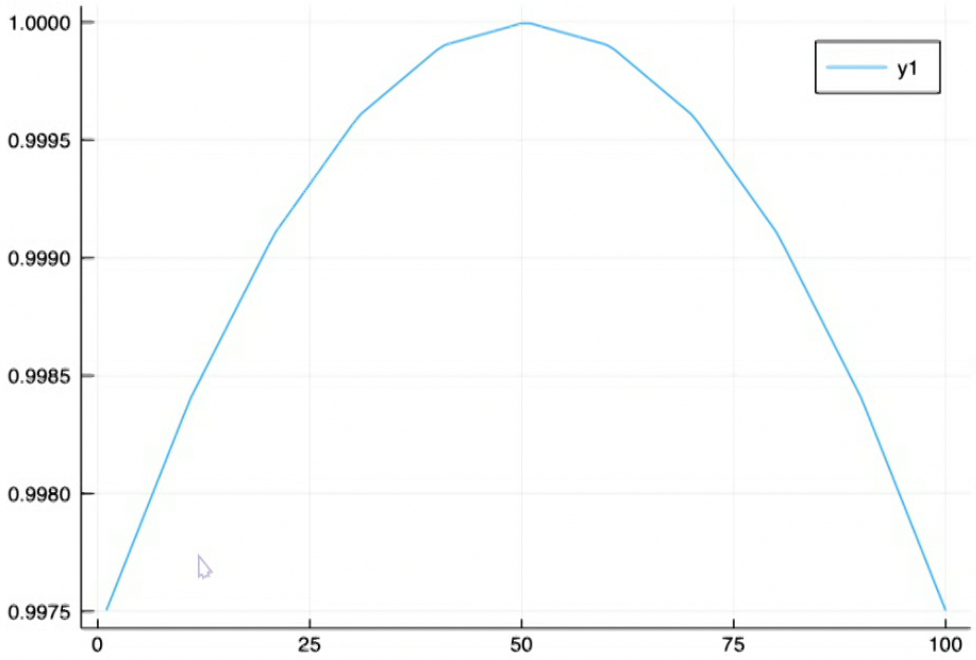
Run | Code

```
In [47]: s0 = initialGaussian(10)
```

```
Out[47]: AdvectionState{Float64}(0.0, PLCFun{Float64,Float64}([0.997503, 0.998401, 0.9991, 0.9996, 0
         .9999, 1.0, 0.9999, 0.9996, 0.9991, 0.998401, 0.997503]))
```

```
In [48]: plot([evaluate(s0.u, x) for x in xs])
```

Out[48]:



localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

Run | Code

In [42]:
```
struct AdvectionState{T}
    time::T
    u::PLCFun{T, T}
end
```

In [44]:
```
function gaussian(x::T)::T where {T}
    exp(- ((x - 0.5) / 10)^2)
end
```

Out[44]: gaussian (generic function with 1 method)

In [45]:
```
function initialGaussian(nlines::Int)::AdvectionState{Float64}
    t = 0
    u = samplePLC(Float64, Float64, nlines, gaussian)
    AdvectionState{Float64}(t, u)
end
```

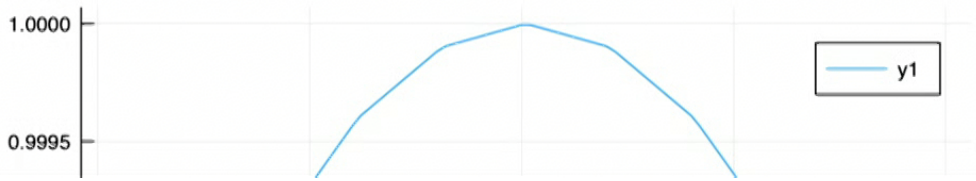
Out[45]: initialGaussian (generic function with 1 method)

In [47]:
```
s0 = initialGaussian(10)
```

Out[47]: AdvectionState{Float64}(0.0, PLCFun{Float64,Float64}([0.997503, 0.998401, 0.9991, 0.9996, 0.9999, 1.0, 0.9999, 0.9996, 0.9991, 0.998401, 0.997503]))

In [48]:
```
plot([evaluate(s0.u, x) for x in xs])
```

Out[48]:



localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

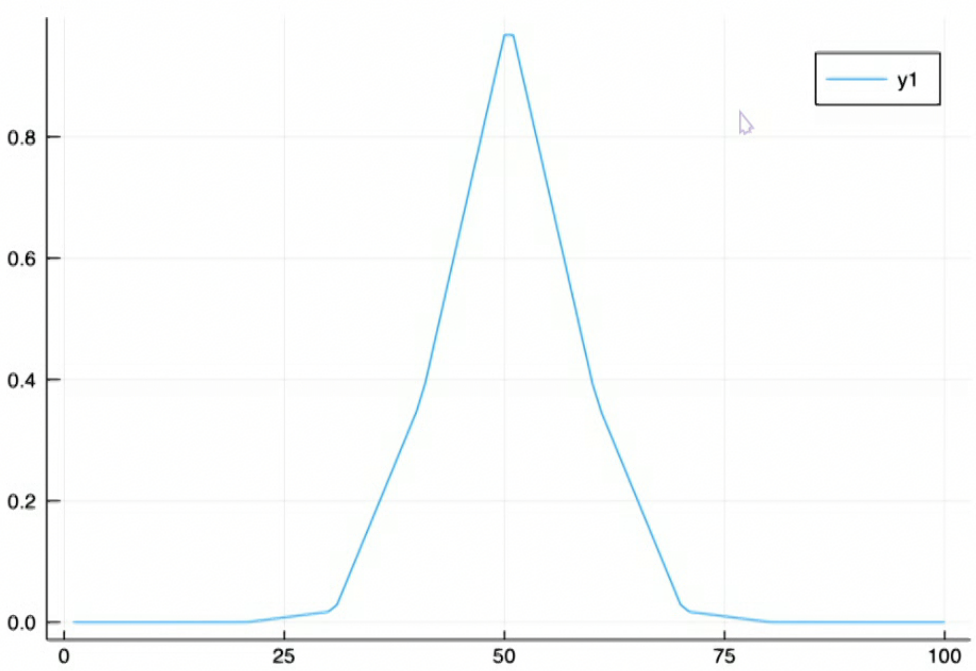
File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

Run | Code

Out[51]: AdvectionState{Float64}(0.0, PLCFun{Float64,Float64}([1.38879e-11, 1.12535e-7, 0.00012341, 0.0183156, 0.367879, 1.0, 0.367879, 0.0183156, 0.00012341, 1.12535e-7, 1.38879e-11]))

In [52]:
```
plot([evaluate(s0.u, x) for x in xs])
```

Out[52]:



In [ ]:

localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb

txt/Courses/CompPhys2018/jl/FiniteDifferencing/ | FiniteDifferencing

File Edit View Insert Cell Kernel Widgets Help | Trusted | Julia 1.0.1

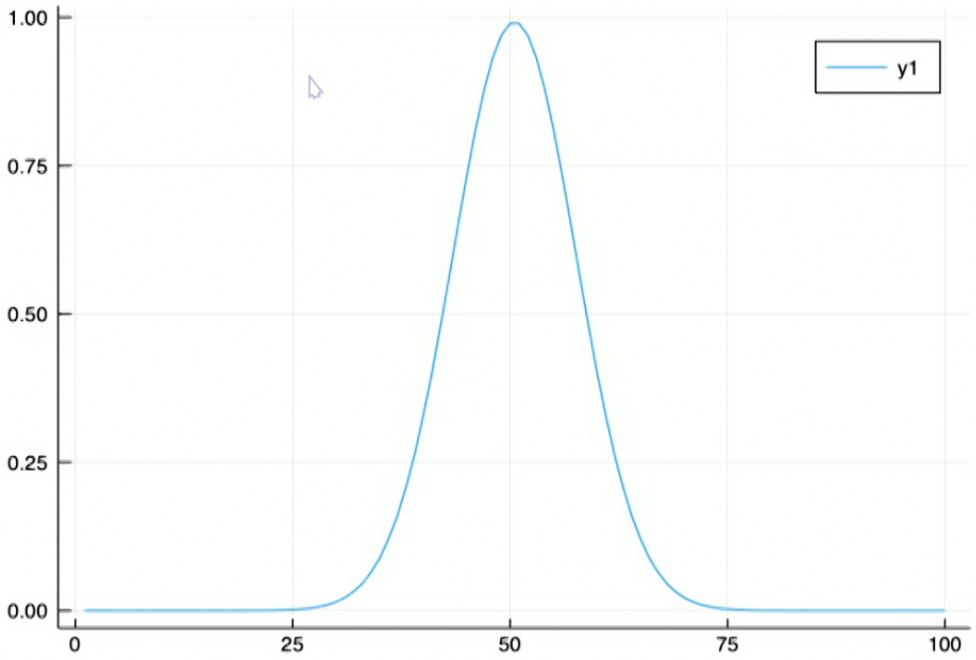
Run | Code

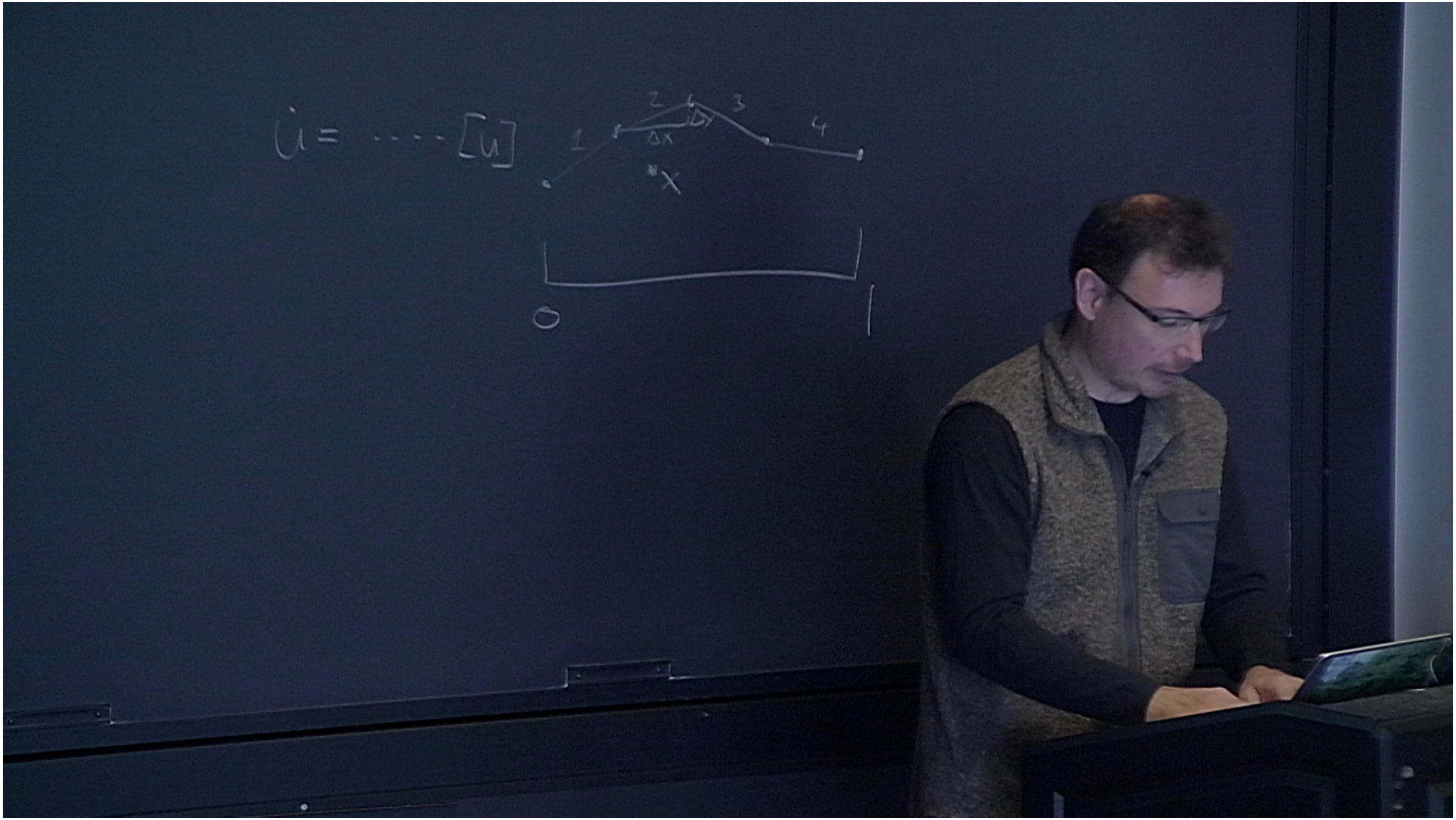
In [57]:
```
s0 = initialGaussian(50)
```

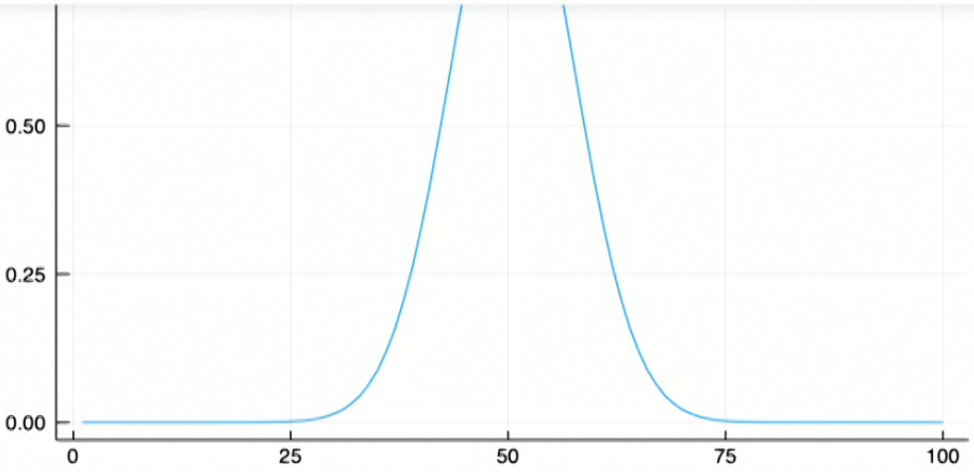
Out[57]:
```
AdvectionState{Float64}(0.0, PLCFun{Float64,Float64}([1.38879e-11, 9.85951e-11, 6.46143e-10
, 3.90894e-9, 2.18296e-8, 1.12535e-7, 5.35535e-7, 2.35258e-6, 9.54016e-6, 3.57128e-5  …  3.
57128e-5, 9.54016e-6, 2.35258e-6, 5.35535e-7, 1.12535e-7, 2.18296e-8, 3.90894e-9, 6.46143e-
10, 9.85951e-11, 1.38879e-11]))
```

In [58]:
```
plot([evaluate(s0.u, x) for x in xs])
```

Out[58]:



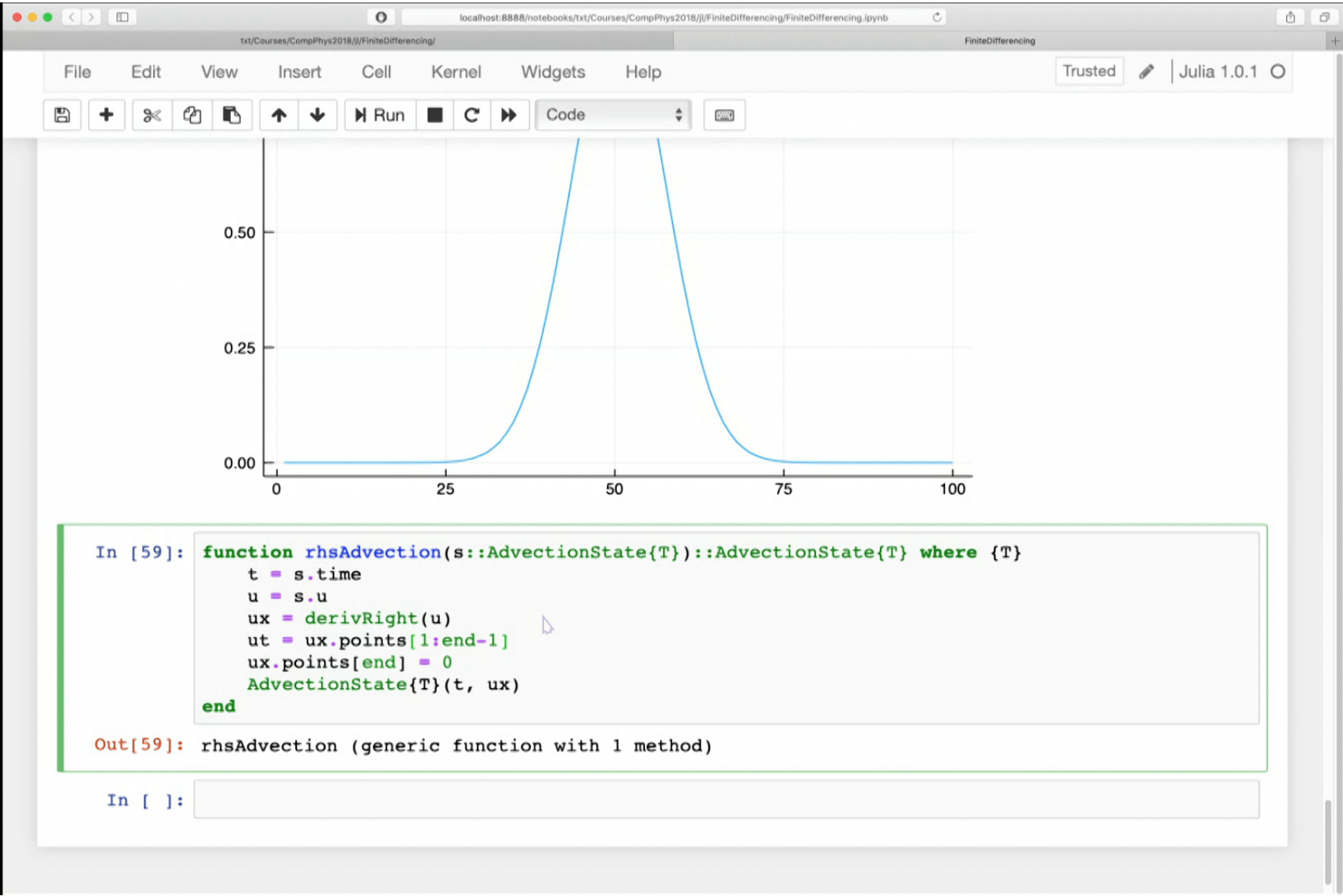

[u]
1
2
3
4
Δx
Δx
x
0
1

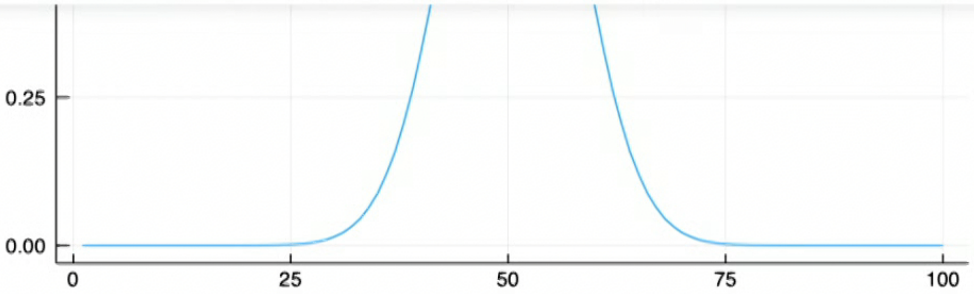


In [59]:
```
function rhsAdvection(s::AdvectionState{T})::AdvectionState{T} where {T}
    t = s.time
    u = s.u
    ux = derivRight(u)
    AdvectionState{T}(t, ux)
end
```

Out[59]: rhsAdvection (generic function with 1 method)

In [ ]:


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
0.50
0.25
0.00
0
25
50
75
100
In [59]:
function rhsAdvection(s::AdvectionState{T})::AdvectionState{T} where {T}
t = s.time
u = s.u
ux = derivRight(u)
ut = ux.points[1:end-1]
ux.points[end] = 0
AdvectionState{T}(t, ux)
end
Out[59]:
rhsAdvection (generic function with 1 method)
In [ ]:



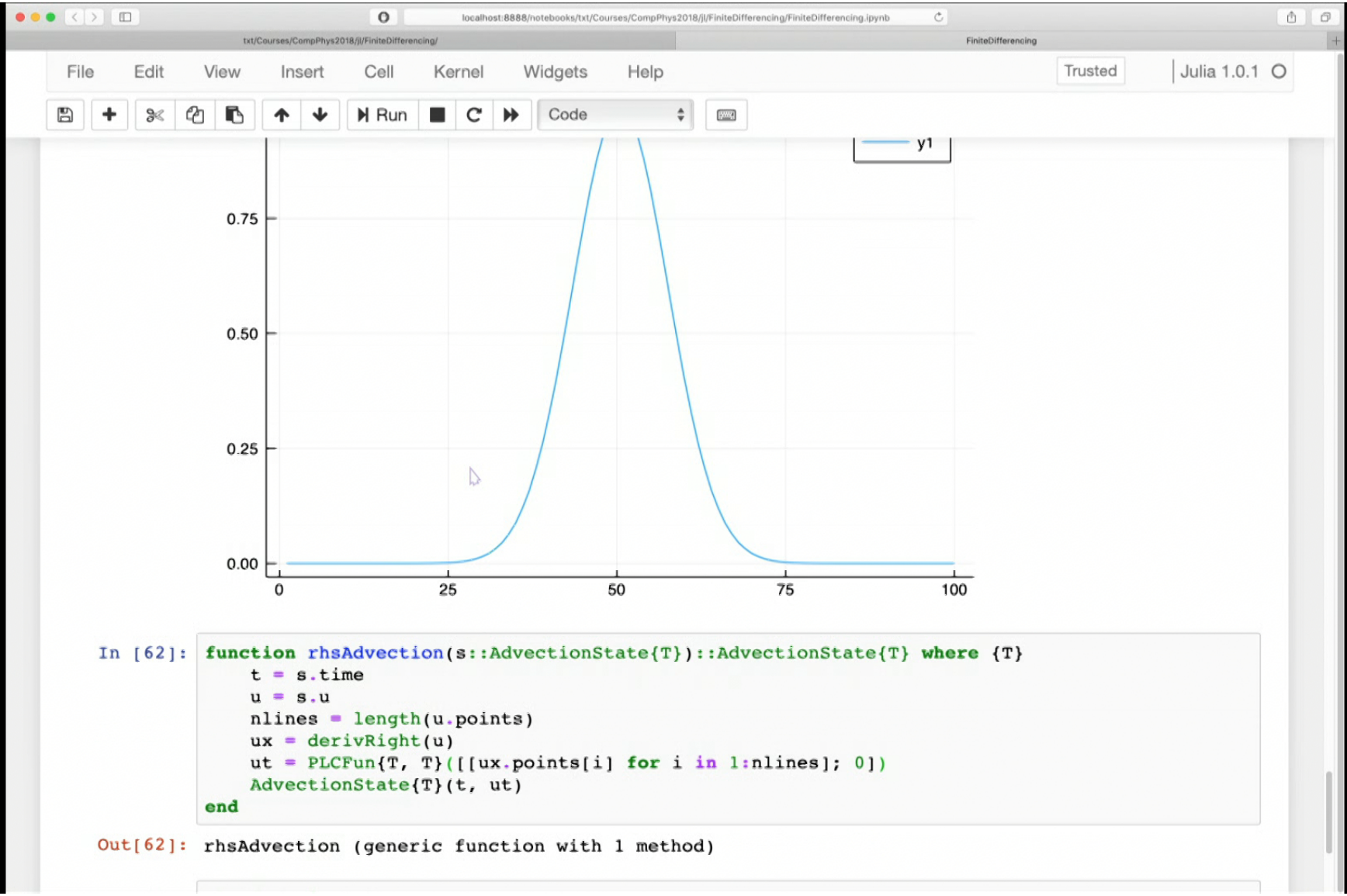
In [60]:
```
function rhsAdvection(s::AdvectionState{T})::AdvectionState{T} where {T}
    t = s.time
    u = s.u
    nlines = length(u.points)
    ux = derivRight(u)
    ut = T[[ux.points[i] for i in 1:nlines]; 0]
    AdvectionState{T}(t, ut)
end
```

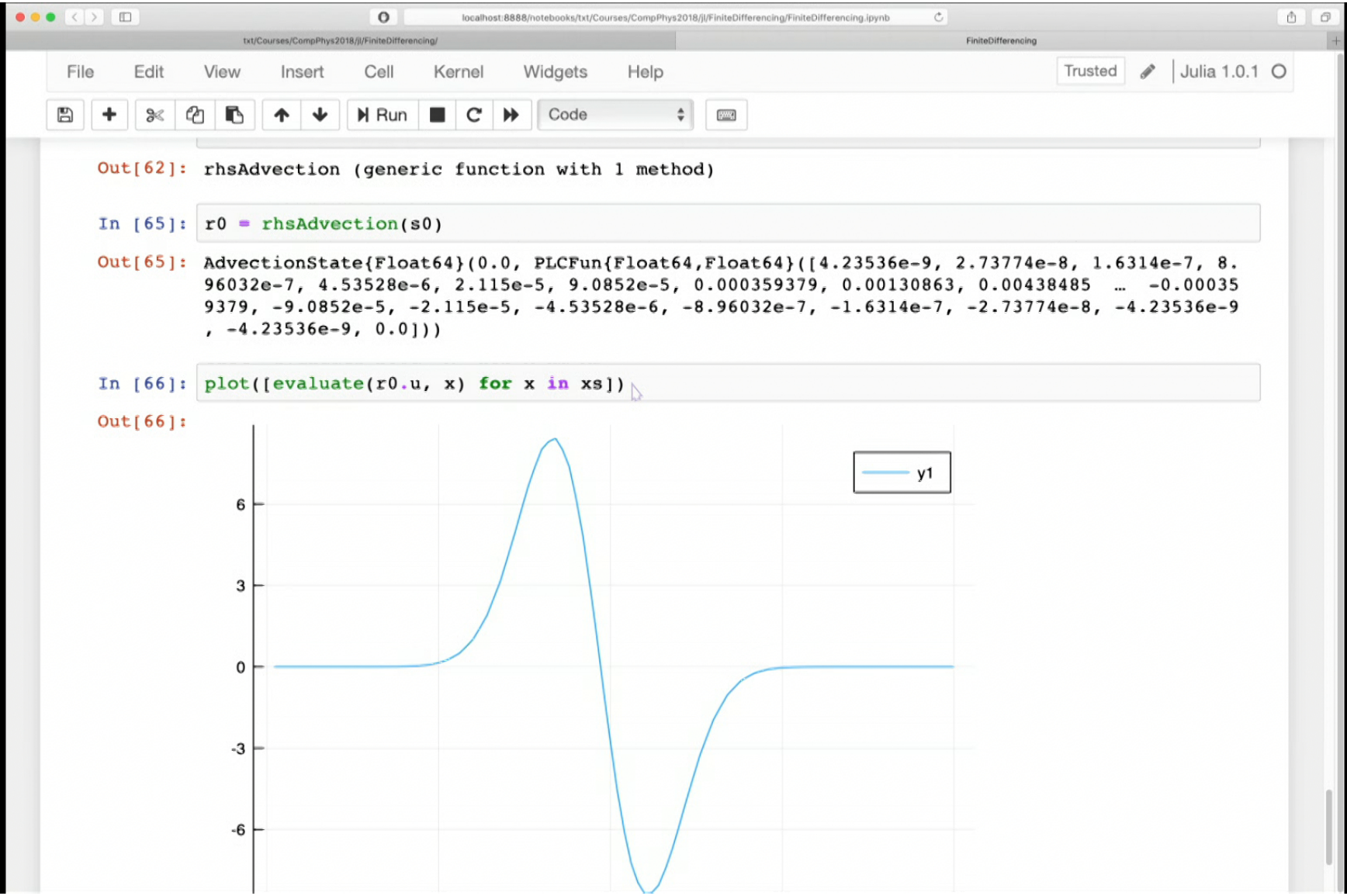
Out[60]:
```
rhsAdvection (generic function with 1 method)
```

In [61]:
```
rhsAdvection(s0)
```

```
MethodError: Cannot `convert` an object of type Array{Float64,1} to an object of type PLCFu
n{Float64,Float64}
Closest candidates are:
  convert(::Type{T}, !Matched::T) where T at essentials.jl:154
  PLCFun{Float64,Float64}(::Any) where {T, U} at In[4]:3

Stacktrace:
 [1] AdvectionState{Float64}(::Float64, ::Array{Float64,1}) at ./In[42]:2
 [2] rhsAdvection(::AdvectionState{Float64}) at ./In[60]:7
```


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
y1
0.75
0.50
0.25
0.00
0
25
50
75
100
In [62]:
function rhsAdvection(s::AdvectionState{T})::AdvectionState{T} where {T}
t = s.time
u = s.u
nlines = length(u.points)
ux = derivRight(u)
ut = PLCFun{T, T}([[ux.points[i] for i in 1:nlines]; 0])
AdvectionState{T}(t, ut)
end
Out[62]:
rhsAdvection (generic function with 1 method)


localhost:8888/notebooks/txt/Courses/CompPhys2018/jl/FiniteDifferencing/FiniteDifferencing.ipynb
txt/Courses/CompPhys2018/jl/FiniteDifferencing/
FiniteDifferencing
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.0.1
Run
Code
Out[62]: rhsAdvection (generic function with 1 method)
In [65]: r0 = rhsAdvection(s0)
Out[65]: AdvectionState{Float64}(0.0, PLCFun{Float64,Float64}([4.23536e-9, 2.73774e-8, 1.6314e-7, 8.96032e-7, 4.53528e-6, 2.115e-5, 9.0852e-5, 0.000359379, 0.00130863, 0.00438485 … -0.000359379, -9.0852e-5, -2.115e-5, -4.53528e-6, -8.96032e-7, -1.6314e-7, -2.73774e-8, -4.23536e-9, -4.23536e-9, 0.0]))
In [66]: plot([evaluate(r0.u, x) for x in xs])
Out[66]:
y1
6
3
0
-3
-6