```
include("hw4_helpers.jl");

using ArgParse;

s = ArgParseSettings();
@add_arg_table s begin
  "-A"
    help = "Coefficient of 1/2 d(u^2)/dx, like a wave speed"
    arg_type = Float64
    default = 1.0
  "-B"
    help = "Diffusion coefficient"
    arg_type = Float64
    default = 1.0
  "-C"
    help = "Coefficient of du/dt"
    arg_type = Float64
    default = 1.0
  "-H"
    help = "Grid spacing"
    arg_type = Float64
    default = 0.2
  "-K"
    help = "Time step size"
    arg_type = Float64
    default = 0.01
  "--show-plot", "-P"
    help = "show plot of solution"
    action = :store_true
  "--fname", "-f"
    help = "file name of plot"
    default = ""
  "--show-error", "-E"
    help = "show L2 error"
    action = :store_true
end

pa = parse_args(s);

const a = pa["A"];
const b = pa["B"];
const c = pa["C"];
const h = pa["H"];
const k = pa["K"];

# analytical solution
asoln(x, t) = a - c * tanh(c / (2*b) * (x - a*t));

xs = linspace(-1.0, 1.0, Int(round((2.0) / h)));
ts = linspace(0.0, 1.0, Int(round(1.0 / k)));
const M, K = length(xs), length(ts);

u = SharedArray(Float64, (M, K); init = x -> 0);
u[:, 1] = map(x -> asoln(x, 0.0), xs);

const α1 = (k * a) / (4 * h);
const α2 = (k * b) / (h^2);

for n in 1:K-1
  for m=2:M-1
    u[m, n+1] = u[m, n] + (-α1 * (u[m+1, n]*u[m+1, n] - u[m-1, n]*u[m-1, n])
                  + α2 * (u[m+1, n] - 2 * u[m, n] + u[m-1, n])) / c;
  end
  u[1, n+1] = asoln(xs[1], ts[n+1]);
```

```
  u[end, n+1] = asoln(xs[end], ts[n+1]);
end

if pa["show-plot"] || pa["fname"] != ""
  plot_solution(xs, ts, u, asoln; t="\$a = $a, b = $b, c = $c, h = $h, k = $k\$",
                show_plot=pa["show-plot"], fname=pa["fname"]);
end

if pa["show-error"]
  println("Relative L2 error");
  for (t, n) in zip(ts, 1:K)
    u_analytical = map(x -> asoln(x, t), xs);
    println(@sprintf("%5.4lf %5.4lf", t,
                     norm(u[:, n] - u_analytical, 2) / norm(u_analytical, 2)));
  end
end
```