18.311 — MIT (Spring 2013)

Rodolfo R. Rosales (MIT, Math. Dept., 2-337, Cambridge, MA 02139).

April 26, 2013.

Problem Set # 07.0 - Due: never.

IMPORTANT: These are practice problems, which I **strongly encourage you to do,** before the lecture on Tuesday April 30.

Contents

1 Introduction. Experimenting with Numerical Schemes.

Consider the numerical schemes that follow after this introduction, for the specified equations. **YOUR TASK here is to** *experiment (numerically) with them so as to answer the question:* **Are they sensible?** Specifically:

(i1) Which schemes give rise to the type of behavior illustrated by the "bad" scheme in the GBNS_lecture script[1] in the 18.311 MatLab Toolkit?

(i2) Which ones behave properly as Δx and Δt vanish?

SHOW THAT they all arise from some approximation of the derivatives in the equations (i.e.: the schemes are CONSISTENT), similar to the approximations used to derive the "good" and "bad" schemes used in the GBNS_lecture script[2] in the 18.311 MatLab Toolkit.

I recommend that you read the *Stability of Numerical Schemes for PDE's* notes in the course WEB page before you do these problems.

Remark 1.1 *Some of these schemes are "good" and some are not. For the "good" schemes you will find that — as Δx gets small — restrictions are needed on Δt to avoid bad behavior (i.e.: fast growth of grid–scale oscillations).*
Specifically: in all the schemes a parameter appears, *the parameter being* $\boldsymbol{\lambda = \Delta t/\Delta x}$ *in some cases and* $\boldsymbol{\nu = \Delta t/(\Delta x)^2}$ *in others. You* will need to keep this parameter smaller than some constant *to get*

[1] Alternatively: check the *Stability of Numerical Schemes for PDE's* notes in the course WEB page.
[2] Alternatively: check the *Stability of Numerical Schemes for PDE's* notes in the course WEB page.

the "good" schemes to behave. That is: $\boldsymbol{\lambda < \lambda_c}$, *or* $\boldsymbol{\nu < \nu_c}$. For the "bad" schemes, it will not matter how small λ (or ν) is. **Figuring out these constants is ALSO PART OF THE PROBLEM.** *For the assigned schemes the constants, when they exist ("good" schemes), are simple* $O(1)$ *numbers, somewhere between* $1/4$ *and* 2. *That is, stuff like* $1/2$, $2/3$, 1, $3/2$, *etc.*, not things like $\lambda_c = \pi/4$ *or* $\nu_c = \sqrt{e}$. You should be able to, easily, find them by careful numerical experimentation!

Remark 1.2 *In order to do these problems you may need to write your own programs. If you choose to use MatLab for this purpose, there are several scripts in the 18.311 MatLab Toolkit that can easily be adapted for this. The relevant scripts are:*

- *The schemes used by the* GBNS_lecture *MatLab script are implemented in the script* InitGBNS.
- *The two series of scripts* PS311_Scheme_A, PS311_Scheme_B, *... and* PS311_SchemeDIC_A, PS311_SchemeDIC_B, *..., have several examples of schemes already setup in an easy to use format. Most of the algorithms here are already implemented in these scripts. The scripts are written so that modifying them to use with a different scheme involves editing only a few (clearly indicated) lines of the code.*
- *Note that the scripts in the 18.311 MatLab Toolkit are written avoiding the use of "for" loops and making use of the vector forms MatLab allows — things run a lot faster this way. Do your programs this way too, it is good practice.*

Remark 1.3 *Do not put lots of graphs and numerical output in your answers. Explain what you did and how you arrived at your conclusions, and illustrate your points with a few selected graphs.*

We use the notation: $\boldsymbol{x_n = x_0 + n\Delta x}$, $\boldsymbol{t_k = t_0 + k\Delta t}$, **and** $\boldsymbol{u_n^k = u(x_n, t_k)}$.

1.1 Statement: GBNS01 Scheme A. Forward differences for $u_t + u_x = 0$.

Equation: $\boldsymbol{u_t + u_x = 0}$. Scheme: $\boldsymbol{u_n^{k+1} = u_n^k - \lambda\,(u_{n+1}^k - u_n^k)}$, where $\boldsymbol{\lambda = \dfrac{\Delta t}{\Delta x}}$.

1.2 Statement: GBNS02 Scheme B. Backward differences for $u_t + u_x = 0$.

Equation: $\boldsymbol{u_t + u_x = 0}$. Scheme: $\boldsymbol{u_n^{k+1} = u_n^k - \lambda\,(u_n^k - u_{n-1}^k)}$, where $\boldsymbol{\lambda = \dfrac{\Delta t}{\Delta x}}$.

THE END.