Convergence, Stability, and Consistency of Finite Difference Schemes in the Solution of Partial Differential Equations

by Gilberto E. Urroz, July 2004

Note: The following worksheet is based on class notes for the class COMPUTATIONAL HYDRAULICS, as taught by Dr. Forrest Holly in the Spring Semester 1985 at the University of Iowa.

Finite difference schemes

A finite difference scheme is produced when the partial derivatives in the partial differential equation(s) governing a physical phenomenon are replaced by a finite difference approximation. The result is a single algebraic equation or a system of algebraic equations which, when solved, provide an approximation to the solution of the original partial differential equation at selected points of a solution grid. The solution grid (also referred to as computational grid or numerical grid) is originated by dividing the axes representing the independent variables in the solution domain into a number of intervals. The extreme points of the interval will represent points in the solution grid. If we draw lines perpendicular to a given axes passing through the extreme points of the intervals, the resulting grid is the computational grid.

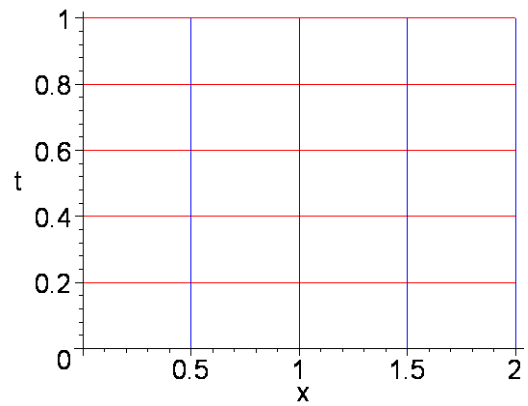
To illustrate a simple computational grid let's assume that the spatial domain corresponds to the values $x = \{0, .5, 1, 1.5, 2.0\}$, while the time domain corresponds to the values $t = \{0, .2, .4, .6, .8, 1.0\}$. The following Maple graph shows the computational grid.

```
> restart:xx:= [seq(0+0.5*j,j=0..4)];tt:=[seq(0+0.2*k,k=0..5)];
```

$$xx := [0., 0.5, 1.0, 1.5, 2.0]$$
$$tt := [0., 0.2, 0.4, 0.6, 0.8, 1.0]$$

```
> with(plots):
Warning, the name changecoords has been redefined
```

```
> pp:=NULL:for j from 1 to 5 do
>      pp := pp, listplot([[xx[j],0],[xx[j],tt[6]]],style = line, color = blue);
> end do:
> for k from 1 to 6 do
>      pp := pp, listplot([[0,tt[k]],[xx[5],tt[k]]],style = line, color = red);
> end do:
> display(pp,labels=["x","t"]);
```



Finite difference approximations result from replacing the partial derivatives in a governing equations by its finite-difference approximation. If the actual solution to a problem in an *x-t* computational domain is given by *u = u(x,t)*, the approximations in the nodes of a computational grid will be given by $u_{i,j} = \mathrm{u}(x_i, t_j)$. The following are finite-difference approximations for the first derivative with respect to the spatial variable *x*:

$$\frac{\partial}{\partial x}\mathrm{u}(x,t) = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + \mathrm{O}(\Delta x),\ \text{forward difference}$$

$$\frac{\partial}{\partial x}\mathrm{u}(x,t) = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + \mathrm{O}(\Delta x),\ \text{backward difference}$$

$$\frac{\partial}{\partial x}\mathrm{u}(x,t) = \frac{u_{i+1,j} - u_{i-1,j}}{2\,\Delta x} + \mathrm{O}(\Delta x^2),\ \text{centered difference}$$

The terms such as $\mathrm{O}(\Delta x)$ or $\mathrm{O}(\Delta x^2)$ indicate that the error incurred when using a particular finite difference approximation to replace the first derivative $\frac{\partial}{\partial x}\mathrm{u}(x,t)$ by that approximation is proportional to either the spatial increment, Δx, or to its square. Typically Δx is a small quantity so that $\Delta x^2 < \Delta x$, therefore, a second order error represents a better approximation. Thus, the centered finite difference approximation shown above for the derivative $\frac{\partial}{\partial x}\mathrm{u}(x,t)$ represents a better approximation than either the forward or backward finite difference approximations for the first derivative. The term $\Delta x = x_{i+1} - x_i = x_i - x_{i-1}$ represents a constant spatial interval. Thus, the spatial grid is said to be *equally-spaced*.

If the spatial grid were not equally spaced, then the proper way to refer to the finite difference approximations would be (error terms are not shown):

$$\frac{\partial}{\partial x}\mathrm{u}(x,t) = \frac{u_{i+1,j} - u_{i,j}}{x_{i+1} - x_i},\ \text{forward difference}$$

$$\frac{\partial}{\partial x}\mathrm{u}(x,t) = \frac{u_{i,j} - u_{i-1,j}}{x_i - x_{i-1}},\ \text{backward difference}$$

$$\frac{\partial}{\partial x} \mathrm{u}(x, t) = \frac{u_{i+1,j} - u_{i-1,j}}{x_{i+1} - x_{i-1}}, \text{ centered difference}$$

For an equally-spaced first derivative in time, the corresponding expressions are shown next:

$$\frac{\partial}{\partial t} \mathrm{u}(x, t) = \frac{u_{i,j+1} - u_{i,j}}{\Delta t} + \mathrm{O}(\Delta t), \text{ forward difference}$$

$$\frac{\partial}{\partial t} \mathrm{u}(x, t) = \frac{u_{i,j} - u_{i,j-1}}{\Delta t} + \mathrm{O}(\Delta t), \text{ backward difference}$$

$$\frac{\partial}{\partial t} \mathrm{u}(x, t) = \frac{u_{i,j+1} - u_{i,j-1}}{2\,\Delta t} + \mathrm{O}(\Delta t^2), \text{ centered difference}$$

A centered difference for the second derivative in ***x*** is given by

$$\frac{\partial^2}{\partial x^2} \mathrm{u}(x, t) = \frac{u_{i+1} - 2\,u_{i,j} + u_{i-1,j}}{\Delta x^2} + \mathrm{O}(\Delta x^2).$$

Notice that the derivatives shown above for the spatial variable, namely, $\frac{\partial}{\partial x} u$ and $\frac{\partial^2}{\partial x^2} u$ are calculated at time step ***j***, i.e. at $t = t_j$. It is possible to "weight" the contribution of time steps ***j*** and ***j+1*** for the derivatives by utilizing a formulation as shown below:

$$\frac{\partial}{\partial x} \mathrm{u}(x, t) = \frac{\theta}{\Delta x} (u_{i+1,j+1} - u_{i,j+1}) + \frac{1-\theta}{\Delta x} (u_{i+1,j} - u_{i,j}), \text{ first order derivative}$$

$$\frac{\partial^2}{\partial x^2} \mathrm{u}(x, t) = \frac{\theta}{\Delta x} (u_{i+1,j+1} + 2\,u_{i,j+1} - u_{i-1,j+1}) + \frac{1-\theta}{\Delta x} (u_{i+1,j} + 2\,u_{i,j} - u_{i-1,j}), \text{ second order derivative}$$

The "weighing factor" θ is selected to be between 0 and 1. Thus, when $\theta = .5$ the two time steps, namely ***j*** and ***j+1***, have equal weight. Also, when $\theta = 0$ the resulting finite difference approximations correspond to time t_j, and when $\theta = 1$ the finite difference approximation corresponds to time t_{j+1}. The second-order finite difference approximation shown above is known as the *Crank-Nicholson formulation*.

First-order partial differential equation for analysis

We will use the following partial differential equation (PDE) to illustrate the ideas of consistency, convergence, and stability of a particular finite-difference numerical scheme. The equation represents the simple linear advection of a scalar quantity, $\mathrm{u}(x, t)$, at a constant speed ***a***:

$$\left(\frac{\partial}{\partial t} \mathrm{u}(x, t)\right) + a \left(\frac{\partial}{\partial x} \mathrm{u}(x, t)\right) = 0,$$

and it's subject to the following initial and boundary conditions:

$$u(x, 0) = u_0(x), \text{ initial condition (i.e., at } t = 0)$$

$$u(0, t) = u_1(t), \text{ boundary condition (i.e., at } x = 0).$$

The approximations that we will select for the derivatives will be the following:

$$\frac{\partial}{\partial t} u(x, t) = \frac{u_{i,j+1} - u_{i,j}}{\Delta t} \text{ , time derivative, forward}$$

$$\frac{\partial}{\partial x} u(x, t) = \frac{u_{i+1,j} - u_{i-1,j}}{2 \Delta x} \text{ , spatial derivative, centered}$$

Thus, the finite difference scheme for this governing equation will be given by the following algebraic equation

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} + a \frac{u_{i+1,j} - u_{i-1,j}}{2 \Delta x} = 0. \quad \text{[A]}$$

Alternatively, we could write this scheme as

$$u_{i,j+1} - u_{i,j} + \frac{a \Delta t}{\Delta x} \left(\frac{u_{i+1,j} - u_{i-1,j}}{2} \right). \quad \text{[B]}$$

Consistency

A finite difference scheme or operator is consistent if the operator reduces to the original differential equation as the increments in the independent variables vanish. For the present case, the finite differences shown in [A], above, will reproduce the original equation as Δx -> 0 and Δt -> 0 . This can be shown by noticing that

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \left(\frac{\partial}{\partial t} u(x, t) \right) + O(\Delta t),$$

and

$$\frac{u_{i+1,j} - u_{i-1,j}}{2 \Delta x} = \left(\frac{\partial}{\partial x} u(x, t) \right) + O(\Delta x^2).$$

Thus, equation [A] can be written as

$$\left(\frac{\partial}{\partial t} u(x, t) \right) + O(\Delta t) + \left(\frac{\partial}{\partial x} u(x, t) \right) + O(\Delta x^2) = 0.$$

Thus, as Δx -> 0 and Δt -> 0, the scheme shown above reduces to the original equation, namely,

$$\left(\frac{\partial}{\partial t}\mathrm{u}(x,t)\right)+0+\left(\frac{\partial}{\partial x}\mathrm{u}(x,t)\right)+0=0.$$

and, therefore, the proposed scheme for the numerical solution of this equation is *consistent*.

It should be point out that any finite difference scheme based on reasonable approximations of the derivatives should be consistent. However, we should also check the scheme for convergence and stability, as shown next.

Convergence and Stability (Von Neumann analysis)

These two characteristics of a numerical scheme in the solution of partial differential equations can be analyzed simultaneously as illustrated next. *Convergence* means that the finite-difference solution approaches the true solution to the partial differential equation as the increments $\Delta x, \Delta t$ go to zero. *Stability* means that the error caused by a small perturbation in the numerical solution remains bound.

The basic idea for the convergence and stability analysis for a linear partial differential equation (i.e., one whose derivatives and terms are of first order) consists in writing the solution to the equation as a complex Fourier series and analyzing a generic component of the solution. Consider the solution to be

$$\mathrm{u}(x,t)=\sum_{m=-\infty}^{\infty} A_m\,\mathbf{e}^{(I(\sigma_m x-\beta_m t))},$$

where $I=\sqrt{-1}$ (the unit imaginary number), A_m is the amplitude of the *m-th* component, $\beta_m=\frac{2\pi}{T_m}$ = angular frequency of the *m-th* component, T_m = period of the *m-th* component, $\sigma_m=\frac{2\pi}{L_m}$ = wave number of the *m-th* component, and L_m = wave length.

Let's now replace the generic component, namely, $u_m(x,t)=A_m\,\mathbf{e}^{(I(\sigma_m x-\beta_m t))}$, into the differential equation:

```
> restart:PDE:=diff(u(x,t),t)+a*diff(u(x,t),x)=0;u:=(x,t)->A[m]*exp(I*(sigma[m]*x-beta[m]*t));
```

$$PDE:=\left(\frac{\partial}{\partial t}\mathrm{u}(x,t)\right)+a\left(\frac{\partial}{\partial x}\mathrm{u}(x,t)\right)=0$$

$$u:=(x,t)\to A_m\,\mathbf{e}^{((\sigma_m x-\beta_m t)I)}$$

```
> PDE;
```

$$-A_m\,\beta_m\,\mathbf{e}^{((\sigma_m x-\beta_m t)I)}\,I+a\,A_m\,\sigma_m\,\mathbf{e}^{((\sigma_m x-\beta_m t)I)}\,I=0$$

```
> simplify(PDE/u(x,t));
```

$$(-\beta_m+a\,\sigma_m)\,I=0$$

This last result indicates that

$$\beta_m = a\,\sigma_m,$$

or that,

$$a = \frac{\beta_m}{\sigma_m} = \frac{L_m}{T_m}.$$

This result indicates that the twavelength, L_m, for any component of the solution, and the corresponding time period, T_m, when divided, should produce the constant speed of advection *a*. The result shown above suggest that all components of the Fourier series solution move with the same speed through space.

The *m-th* component of the solution, namely,

$$u_m(x, t) = A_m\, \mathbf{e}^{(I(\sigma_m x - \beta_m t))}$$

can also be written as

$$u_m(x, t) = A_m\, \mathbf{e}^{(-I\beta_m t)}\, \mathbf{e}^{(I\sigma_m x)},$$

which suggest that each component consist of an amplitude, A_m, multiplied by a time variation term (or *amplification factor*) given by $\mathbf{e}^{(-I\beta_m t)}$, and a space variation term given by $\mathbf{e}^{(I\sigma_m t)}$. For a physical process represented by the general component $u_m(x, t)$ we require that σ_m, β_m, and a be all real constants. Thus, from Euler's formula, i.e., $\mathbf{e}^{(I\theta)} = \cos(\theta) + I\sin(\theta)$, it follows that

$$\mathbf{e}^{(-I\beta_m t)} = \cos(\beta_m t) - I\sin(\beta_m(t)).$$

Also, the magnitude (or absolute value) of this amplification factor should be 1.0, i.e.,

$$\left|\mathbf{e}^{(-I\beta_m t)}\right| = \left|\cos(\beta_m t) - I\sin(\beta_m(t))\right| = \sqrt{\cos(\beta_m t)^2 + \sin(\beta_m t)^2} = 1.0.$$

Therefore, $u_m(x, t)$ is neither amplified nor damped with time.

Earlier on we found that all components of the Fourier series solution move with the same speed through space. The later result indicates that the components not only move with the same speed, but also that they travel without damping. Thus, any initial distribution $u_0(x)$ would simply move downstream, at speed a, without changing its shape.

How does the numerical solution behaves for the *m-th* component of the solution? To answer this question we write the *m-th* component of the solution at point *i,j* as

$$u_{m_{i,j}} = A_m\, \mathbf{e}^{(-I\beta_m j\,\Delta t)}\, \mathbf{e}^{(I\sigma_m i\,\Delta x)}.$$

Here, $x_i = i\,\Delta x$ and $t_j = j\,\Delta t$. The values A_m, β_m, and σ_m where described earlier. Next, we replace the value of

$u_{m_{i,j}}$ in the following expression for the numerical scheme

$$u_{i,j+1} - u_{i,j} + \frac{r}{2}\,(\,u_{i+1,j} - u_{i-1,j}\,) = 0,$$

where $r = \frac{a\,\Delta t}{\Delta x}$. This expression is based on equation [B], shown above. The following Maple statements produce the substitution and simplification of the resulting expression:

```
> restart:FDE:=u(i,j+1)-u(i,j) + r/2 *(u(i+1,j)-u(i-1,j)) = 0;
```

$$FDE := \mathrm{u}(i, j+1) - \mathrm{u}(i, j) + \frac{1}{2}\, r\, (\mathrm{u}(i+1, j) - \mathrm{u}(i-1, j)) = 0$$

```
> u:=(i,j)->A[m]*exp(-I*beta[m]*j*Delta*t)*exp(I*sigma[m]*i*Delta*x);
```

$$u := (i, j) \to A_m\, \mathbf{e}^{(-I\, \beta_m\, j\, \Delta\, t)}\, \mathbf{e}^{(\sigma_m\, i\, \Delta\, x\, I)}$$

```
> FDE;
```

$$A_m\, \mathbf{e}^{(-I\, \beta_m\, (j+1)\, \Delta\, t)}\, \mathbf{e}^{(\sigma_m\, i\, \Delta\, x\, I)} - A_m\, \mathbf{e}^{(-I\, \beta_m\, j\, \Delta\, t)}\, \mathbf{e}^{(\sigma_m\, i\, \Delta\, x\, I)} + \frac{1}{2}\, r\, (A_m\, \mathbf{e}^{(-I\, \beta_m\, j\, \Delta\, t)}\, \mathbf{e}^{(\sigma_m\, (i+1)\, \Delta\, x\, I)} - A_m\, \mathbf{e}^{(-I\, \beta_m\, j\, \Delta\, t)}\, \mathbf{e}^{(\sigma_m\, (i-1)\, \Delta\, x\, I)}) = 0$$

```
> simplify(FDE/u(i,j));
```

$$r \sin(\Delta\, \sigma_m\, x)\, I + \mathbf{e}^{(-I\, \Delta\, t\, \beta_m)} - 1 = 0$$

From this result we conclude that

$$\mathbf{e}^{(-I\, \beta_m\, \Delta\, t)} = 1 - I\, r \sin(\sigma_m\, \Delta\, x).$$

Thus, an amplification factor for $j = 1$ (i.e., first time step) and ***i* = *1*** (i.e., first space increment), has a magnitude of

$$\left|\mathbf{e}^{(-I\, \beta_m\, \Delta\, t)}\right| = \left|1 - I\, r \sin(\sigma_m\, \Delta\, x)\right| = \sqrt{1 + r^2 \sin(\sigma_m\, \Delta\, x)^2}\;.$$

This magnification factor is always larger than 1.0. Thus, any small perturbation in the solution will be amplified, and the scheme, although consistent, is *always unstable*. This means that any and all components of the solution will grow without bound as time progresses. Notice also that the result shown above for the amplification factor, namely, that $\left|\mathbf{e}^{(-I\, \beta_m\, \Delta\, t)}\right| > 1$, requires β_m to be an imaginary number, i.e., $\beta_m = \alpha_m + I\, \delta_m$, where $\alpha_m = \Re(\beta_m)$ and $\delta_m = \Im(\beta_m)$.

In terms of the celerity of each component of the numerical solution we find that the celerity must be defined as $\frac{\alpha_m}{\sigma_m} = \frac{\Re(\beta_m)}{\sigma_m}$. To find an expression for this result, we start from

$$\mathbf{e}^{(-I\, \beta_m\, \Delta\, t)} = 1 - I\, r \sin(\sigma_m\, \Delta\, x),$$

or

$$\mathbf{e}^{(-I\,\alpha_m\,\Delta t)}\,\mathbf{e}^{(\delta_m\,\Delta t)} = 1 - I\,r\,\sin(\sigma_m\,\Delta x).$$

Using Euler's formula we can write

$$(\cos(\alpha_m\,\Delta t) - I\sin(\alpha_m\,\Delta t))\,\mathbf{e}^{(\delta_m\,\Delta t)} = 1 - I\,r\,\sin(\sigma_m\,\Delta x).$$

Equating the real and imaginary parts of this equation we produce two equations, namely,

$$\sin(\alpha_m\,\Delta t)\,\mathbf{e}^{(\delta_m\,\Delta t)} = r\,\sin(\sigma_m\,\Delta x).$$

and

$$\cos(\alpha_m\,\Delta t)\,\mathbf{e}^{(\delta_m\,\Delta t)} = 1.$$

Dividing term-by-term we find that

$$\tan(\alpha_m\,\Delta t) = r\,\sin(\sigma_m\,\Delta x).$$

Thus,

$$\alpha_m = \frac{\operatorname{atan}(r\,\sin(\sigma_m\,\Delta x))}{\Delta t}$$

and, the celerity of the *m-th* component in the numerical solution is

$$a_m = \frac{\alpha_m}{\sigma_m} = \frac{\operatorname{atan}(r\,\sin(\sigma_m\,\Delta x))}{\sigma_m\,\Delta t},$$

i.e., not a constant value but a function of r, $\frac{\Delta x}{L_m} = \sigma_m\,\Delta x$, and Δt. Thus, every component of the numerical solution will propagate with different celerity not equal to the convective celerity a. This fact results in what is called *numerical dispersion*, i.e., a phenomenon similar to physical dispersion but caused by the numerical scheme itself.

Thus, the suggested numerical scheme is *always unstable*, and *always dispersive*, i.e., basically useless for calculating a reliable numerical approximation to the partial differential equation of interest to our analysis.

NOTE: The procedure presented above for analyzind convergence and stability of a numerical scheme, known as von Neumann (*) analysis, applies only to linear differential equations with periodic boundary conditions (to ensure applicability of the Fourier series components). Thus, the analysis would not apply to an equation such as $\left(\frac{\partial}{\partial t}\mathrm{u}(x,t)\right) + \mathrm{u}(x,t)\left(\frac{\partial}{\partial x}\mathrm{u}(x,t)\right) = 0$ because the second term in the equation is non-linear. Linearizing the equation by some means can help in the application of von Neumann's analysis for convergence and stability. If a linearized scheme is not convergent, most likely the original non-linear scheme will not converge either.

(*)**John von Neumann** (1903-1957), a mathematician and chemical engineer, made significant contributions to the sciences of quantum physics, mathematical logic, and meteorology. His contributions to computer science and game theory are also monumental. He wrote about 150 papers on a number of subjects: physics, set theory, mathematical logic, topological groups, measure theory, ergodic theory, operator theory, continuous geometry, statistics, numerical analysis, shock waves, flow problems, hydrodynamics, aerodynamics, ballistics, problems of detonation of explosives, meteorology, mathematical games and computer logic.

Amplitude and phase portraits

The stability analysis shown earlier for a simple linear partial differential equation and a specific numerical scheme produced two main results: (1) an amplification factor; and (2) celerity, for each component of the numerical solution. We will define an *amplification parameter* R_1 as the ratio of the magnitudes of the numerical amplification factor to the true amplification factor (which happens to be 1.0), i.e., for this case,

$$R_1 = \frac{\left|\mathbf{e}^{(-I\,\beta_m\,\Delta t)}\right|}{1.0} = \left|\,1 - I\,r\,\sin(\sigma_m\,\Delta x)\right| = \sqrt{1 + r^2\,\sin(\sigma_m\,\Delta x)^2}\ .$$

A phase parameter, R_2, is defined as the ration of the numerical celerity to that of the true (or analytical) celerity. Thus, for this case,

$$R_2 = \frac{a_m}{a} = \frac{\operatorname{atan}(r\,\sin(\sigma_m\,\Delta x))}{a\,\sigma_m\,\Delta t}\ .$$

Plots of the parameters R_1 and R_2 versus the dimensionless parameter $\frac{L_m}{\Delta x} = \frac{2\,\pi}{\sigma_m\,\Delta x}$ are referred to as *amplitude portrait* and *phase portrait*, respectively. These "portraits" can be used to show graphically the stability, or lack thereof, of a numerical scheme. Typically, the "portraits" will show plots corresponding to different values of the parameter $r = \frac{a\,\Delta t}{\Delta x}$. While the parameter L_m represents the characteristic wave length of the *m-th* component of the numerical solution, it can be taken to be the length of the solution domain, i.e., $0 < x < L_m$. Thus, the dimensionless parameter $L_s = \frac{L_m}{\Delta x}$ relates the length of the solution domain to the grid size. The smallest the grid size, Δx, the largest the value of L_s.

Consider the amplitude and phase portraits of the scheme analyzed above. First, the expression for R_1 can be written as a function of $L_s = \frac{L_m}{\Delta x} = \frac{2\,\pi}{\sigma_m\,\Delta x}$, by writting $\sigma_m\,\Delta x = \frac{2\,\pi}{L_s}$. With this result, the amplification parameter is given by

$$R_1 = \sqrt{1 + r^2\,\sin\left(\frac{2\,\pi}{L_s}\right)^2}\ .$$

Also, the phase portrait can be plotted by using $\sigma_m\,\Delta x = \frac{2\,\pi}{L_s}$, and $a\,\Delta t = r\,\Delta x$, so that the phase parameter becomes

$$R_2 = \frac{\operatorname{atan}\left(r \sin\left(\frac{2\,\pi}{L_s}\right)\right)}{\frac{r\,2\,\pi}{L_s}} = \frac{L_s}{2\,\pi\,r} \operatorname{atan}\left(r \sin\left(\frac{2\,\pi}{L_s}\right)\right).$$

Thus, by letting L_s be between 0 and $2\,\pi$, we can produce the amplitude and phase portraits for values of r = 0.1, 0.5, 0.7, and 1.0. The plots are shown below. First, the amplitude portrait:

```
> restart:
> R1:=(r,Ls) -> sqrt(1+r^2*sin(2*Pi/Ls));
```

$$R1 := (r, Ls) \rightarrow \sqrt{1 + r^2 \sin\left(\frac{2\,\pi}{Ls}\right)}$$

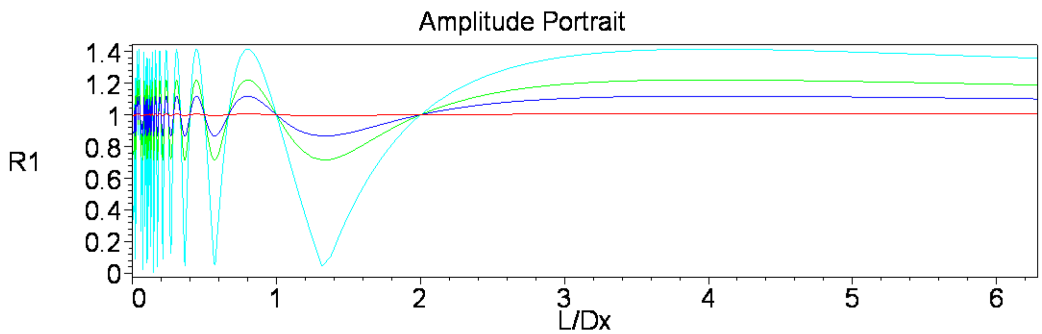
```
> R2:=(r,Ls) -> (Ls/(2*Pi*r))*arctan(r*sin(2*Pi/Ls));
```

$$R2 := (r, Ls) \rightarrow \frac{1}{2} \frac{Ls \arctan\left(r \sin\left(\frac{2\,\pi}{Ls}\right)\right)}{\pi\,r}$$

```
> rr:=[0.1,0.5,0.7,1.0];cc:=[red,blue,green,cyan];
```

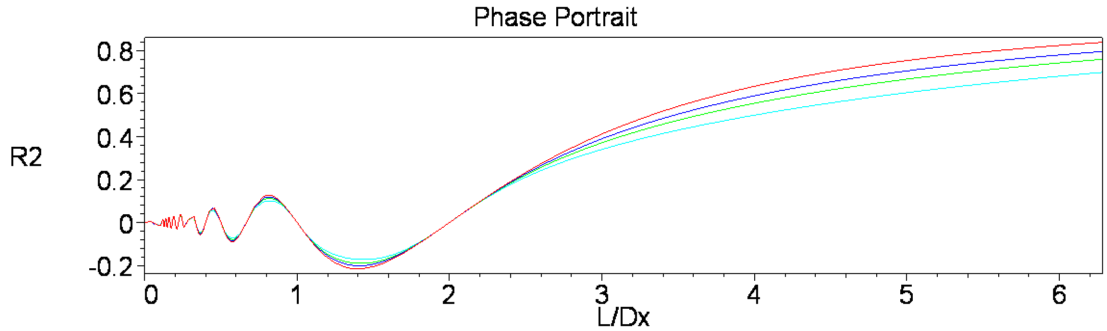
$$rr := [0.1, 0.5, 0.7, 1.0]$$
$$cc := [red, blue, green, cyan]$$

```
> pp:=NULL:for j from 1 to 4 do
>      pp:=pp,plot(R1(rr[j],Ls),Ls = 0..2*Pi,color=cc[j]);
> end do:
> plots[display](pp,labels=["L/Dx","R1"],axes=boxed,title="Amplitude Portrait");
```



The following plot represents the phase portrait:

```
> ppp:=NULL:for j from 1 to 4 do
>      ppp:=ppp,plot(R2(rr[j],Ls),Ls = 0..2*Pi,color=cc[j]);
> end do:
> plots[display](ppp,labels=["L/Dx","R2"],axes=boxed,title="Phase Portrait");
```

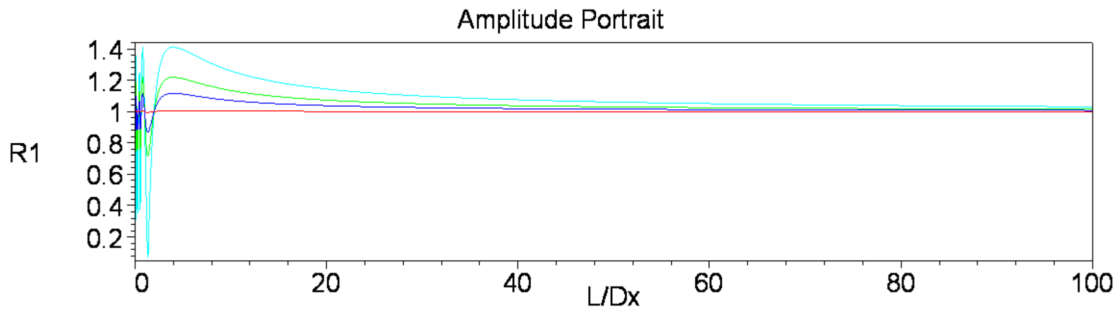



```
>
```

In most cases the values of $L_s = \dfrac{L_m}{\Delta x}$ would be larger than $2\,\pi$, therefore, more detailed amplitude and phase portraits for the current scheme will include a larger range for L_s, say, between 0 and 100. Here are the corresponding phase portraits for r = 0.1, 0.5, 0.7, and 1.0.

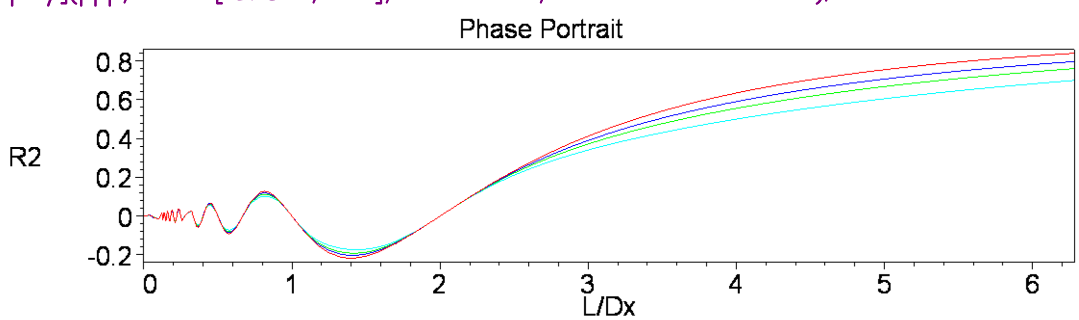
Amplitude portrait:

```
> pp:=NULL:for j from 1 to 4 do
>      pp:=pp,plot(R1(rr[j],Ls),Ls = 0..100,color=cc[j]);
> end do:
> plots[display](pp,labels=["L/Dx","R1"],axes=boxed,title="Amplitude Portrait");
```



Phase portrait:

```
> ppp:=NULL:for j from 1 to 4 do
>      ppp:=ppp,plot(R2(rr[j],Ls),Ls = 0..2*Pi,color=cc[j]);
> end do:
> plots[display](ppp,labels=["L/Dx","R2"],axes=boxed,title="Phase Portrait");
```



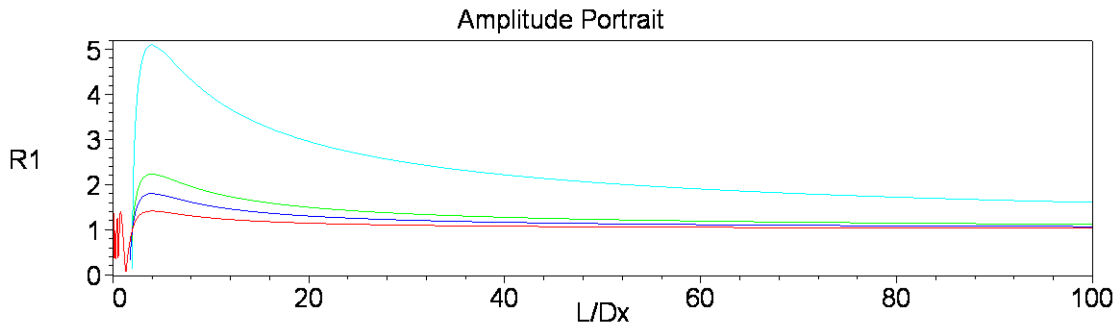
So far, the amplitude and phase portraits shown correspond to relatively low values of the parameter $r = \frac{a \Delta t}{\Delta x}$, i.e., for 0 < r < 1. Next, we show amplitude and phase portraits for larger values of *r*, i.e., for *r* = *1, 1.5, 2,* and *5*:

```
> rr:=[1,1.5,2,5];
```

$$rr := [1, 1.5, 2, 5]$$

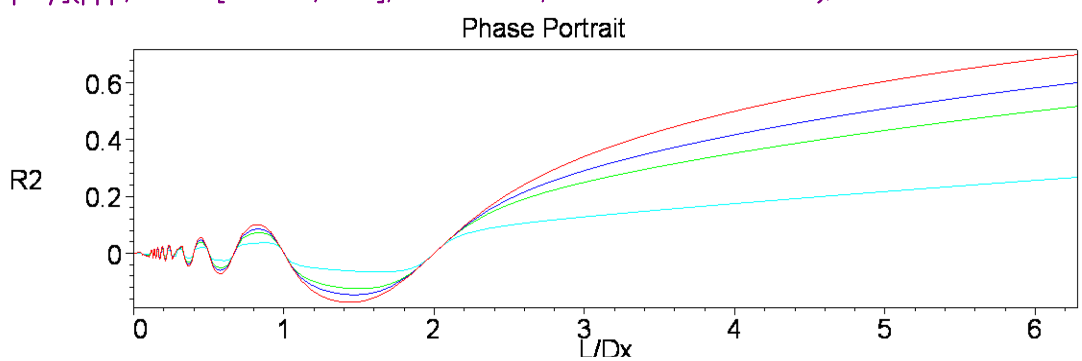
Amplitude portrait:

```
> pp:=NULL:for j from 1 to 4 do
>       pp:=pp,plot(R1(rr[j],Ls),Ls = 0..100,color=cc[j]);
> end do:
> plots[display](pp,labels=["L/Dx","R1"],axes=boxed,title="Amplitude Portrait");
```



Phase portrait:

```
> ppp:=NULL:for j from 1 to 4 do
>       ppp:=ppp,plot(R2(rr[j],Ls),Ls = 0..2*Pi,color=cc[j]);
> end do:
> plots[display](ppp,labels=["L/Dx","R2"],axes=boxed,title="Phase Portrait");
```



From the amplitude portraits shown above we notice that the amplification parameter varies between 0.2 and 1.4 when the values of the parameter $r = \frac{a \Delta t}{\Delta x}$ are smaller than 1.0 (i.e., if $a \Delta t < \Delta x$), the amplification parameter varies widely if the values of $L_s = \frac{L_m}{\Delta x}$ are less than 2π. The amplification parameter peaks at about L_s = *3*, and then decreases as L_s grows past the value $L_s = 3$. As the value of *r* grows larger than 1.0, the amplification parameter reaches larger values. Thus, the scheme proposed herein will produce relatively large amplification parameters particularly for larger values of *r* and for small values of L_s.

The phase portrait shows the phase parameter as an oscillatory signal whose amplitude and wavelength increases with L_s, thus, confirming the observation pointed out earlier that the celerity of the numerical solution components is different and will produce numerical dispersion.

Creating amplitude and phase portraits in Matlab

The following script can be used to produce amplitude and phase portraits for the present finite-difference scheme:

```
======================================================================
% Script to plot amplitude and phase portraits for solving the
% PDE:  diff(u(x,t),t)+ a*diff(u(x,t),x) = 0, using forward
% time derivative and centered space derivative.
% The amplitude portrait consists of plotting the amplification
% parameter R1 vs. Ls = L/Dx, for different values of r = a*Dt/Dx.
% The phase portrait consists of plotting the phase parameter R2
% vs. Ls = L/Dx, for different values of r = a*Dt/Dx.

% (1) First, we produce the AMPLITUDE PORTRAIT for 0.1 < r < 1.0:
Ls = [0:0.1:100]; r = [0.1,0.5,0.7,1.0]; cc = ['r','b','g','k','m'];
n = length(Ls);     m = length(r);
R1 = zeros(n,m);
for j = 1:m
   for i = 1:n
      R1(i,j) = sqrt(1+r(j)^2*sin(2*pi/Ls(i)));
   end;
end;
figure(1);hold;
for j = 1:m
   plot(Ls,R1(:,j),cc(j));
end;
hold;
xlabel('L/Dx');ylabel('R1');title('Amplitude Portrait, 0.1 < r < 1.0');

% (2) Next, we produce the AMPLITUDE PORTRAIT for 1.0 < r < 5.0:
Ls = [0:0.1:100]; r = [1.0,1.5,2.0,5.0]; cc = ['r','b','g','k','m'];
n = length(Ls);     m = length(r);
R1 = zeros(n,m);
for j = 1:m
   for i = 1:n
      R1(i,j) = sqrt(1+r(j)^2*sin(2*pi/Ls(i)));
   end;
end;
figure(2);hold;
for j = 1:m
   plot(Ls,R1(:,j),cc(j));
end;
hold;
xlabel('L/Dx');ylabel('R1');title('Amplitude Portrait, 1.0 < r < 5.0');

% (3) Next, we produce the PHASE PORTRAIT for 0.1 < r < 1.0:
Ls = [0:0.1:100]; r = [0.1,0.5,0.7,1.0]; cc = ['r','b','g','k','m'];
n = length(Ls);     m = length(r);
R2 = zeros(n,m);
```

```
for j = 1:m
   for i = 1:n
      R2(i,j) = (Ls(i)/(2*pi*r(j)))*atan(r(j)*sin(2*pi/Ls(i)));
   end;
end;
figure(3);hold;
for j = 1:m
   plot(Ls,R2(:,j),cc(j));
end;
hold;
xlabel('L/Dx');ylabel('R2');title('Phase Portrait, 0.1 < r < 1.0');

% (4) Next, we produce the PHASE PORTRAIT for 1.0 < r < 5.0:
Ls = [0:0.1:100]; r = [1.0,1.5,2.0,5.0]; cc = ['r','b','g','k','m'];
n = length(Ls);     m = length(r);
R2 = zeros(n,m);
for j = 1:m
   for i = 1:n
      R2(i,j) = (Ls(i)/(2*pi*r(j)))*atan(r(j)*sin(2*pi/Ls(i)));
   end;
end;
figure(4);hold;
for j = 1:m
   plot(Ls,R2(:,j),cc(j));
end;
hold;
xlabel('L/Dx');ylabel('R1');title('Phase Portrait, 1.0 < r < 5.0');
======================================================================
```

Assignment

NOTE: This assignment is based on class notes for the class COMPUTATIONAL HYDRAULICS, as taught by Dr. Forrest Holly in the Spring Semester 1985 at the University of Iowa.

Consider once more the linear advection equation, namely,

$$\left(\frac{\partial}{\partial t}\mathrm{u}(x,t)\right)+a\left(\frac{\partial}{\partial x}\mathrm{u}(x,t)\right)=0,$$

were *a* is a constant. Suppose that we want to solve this equation for *u(x,t)* in the domain *0 < x < L, 0 < t < T* with boundary condition $u(0,t) = u_1(t)$, and initial condition $u(x,0) = u_0(x)$. The numerical solution will utilize the so-called *upwind finite difference approximations* for the derivatives, namely,

$$\frac{\partial}{\partial t}\mathrm{u}(x,t)=\frac{u_{i,j+1}-u_{i,j}}{\Delta t}$$

and

$$\frac{\partial}{\partial x}\mathrm{u}(x,t)=\frac{u_{i,j}-u_{i-1,j}}{\Delta x}.$$

(a) Perform a stability analysis (von Neumann's analysis) on the upwind difference scheme that results from using the finite difference approximations shown above in the linear advection equation. (Follow the approach shown earlier in this document).

(b) Obtain expression for the amplification and phase parameters, $R_1(r, Ls)$ and $R_2(r, Ls)$, respectively, where $L_s = \dfrac{L}{\Delta x}$ and $r = \dfrac{a\,\Delta t}{\Delta x}$ (NOTE: r is sometimes referred to as the Courant number).

(c) Using MATLAB, plot amplitude and phase portraits for Courant numbers r = *0.25, 0.5, 0.75, 1.0, 2.0*, and for $0 < \dfrac{L}{\Delta x} < 30$.

(d) Based on the results shown in your amplitude and phase portraits discuss the stability of the upwind method.