1-d Wave Equation Exercise

Create a source file `wave1d.f`, and corresponding executable `wave1d`, which uses second-order finite-difference techniques to solve the following one-dimensional wave equation for $u(x,t)$:

$$u_{tt} = u_{xx} \qquad 0 \le x \le 1 \quad t \ge 0 \tag{1}$$

$$u(x,0) = l(x) + r(x) \qquad u_t(x,0) = l'(x) - r'(x) \qquad u(0,t) = u(1,t) = 0 \tag{2}$$

where $l(x)$ and $r(x)$ are, respectively, the left-moving and right-moving components of the solution at $t = 0$ and $'$ denotes differentiation.

`wave1d` should accept 6 arguments as illustrated by the following usage message:

```
usage: wave1d <level> <dt/dx> <ncross> <a left-mover> <a right-mover> <olevel>
```

The arguments have the following interpretation:

- `level`: Discretization level (`integer`). The spatial mesh will have $\texttt{nx} = 2^{\texttt{level}} + 1$ points.
- `dt/dx`: "Courant number" (`real*8`). Ratio of temporal spacing $\triangle t$ to spatial mesh-size $\triangle x$.
- `ncross`: Final integration time in units of "crossing times" (`integer`). A crossing time is the amount of time it takes for a signal to propagate across the solution domain. Since the wave speed in (1) is 1, and the spatial domain is $0 \le x \le 1$, the crossing time in this case is also 1. Note that the number of time steps in the integration, `nt`, is then implicitly defined by `level`, `dt/dx` and `ncross`.
- `<a left-mover>`: The amplitude (`real*8`) of the initially left-moving component of the wave (see below).
- `<a right-mover>`: The amplitude (`real*8`) of the initially right-moving component of the wave (see below).
- `olevel`: Output level (`integer`). Must not be greater than `level`. This parameter defines the frequency of output as in the previous question:

  $$\texttt{ofreq} = 2^{\texttt{level} - \texttt{olevel}}$$

  In addition, in this case `ofreq` also specifies a "spatial" frequency of output, e.g. if `ofreq = 2`, then at output times, every second value $u_1^n, u_3^n, \cdots$ is dumped (see below).

To aid in the development of your program, the following routines are provided in the file `util.f` available from

`ftp://valentina.fis.cinvestav.mx/pub/choptuik/wave1d`

The first routine is

```
subroutine dvgaussian(g,dg,ddg,x,n,amp,x0,del)
   integer        n
   real*8         g(n),     dg(n),     ddg(n),     x(n)
   real*8         amp,      x0,        del
```

which given `x(j) , j = 1 ... n`, `amp`, `x0` and `del` returns a Gaussian profile, `g(j)`, and its first two derivatives, `dg(j)` and `ddg(j)`, evaluated on the finite-difference mesh:

$$\texttt{g(j)} \equiv g(\texttt{x(j)}) = \texttt{amp} \times \exp\left(-\left(\texttt{x(j)} - \texttt{x0}\right)^2 / \texttt{del}^2\right)$$

$$\texttt{dg(j)} \equiv \frac{dg}{dx}\left(\texttt{x(j)}\right)$$

$$\texttt{ddg(j)} \equiv \frac{d^2 g}{dx^2}\,(\texttt{x(j)})$$

This routine should be used to set up the initially left- and right-moving components of the solution, $l(x)$ and $r(x)$ as well as the first and second derivatives of these components, $l'(x), r'(x), l''(x), r''(x)$. Each of the two components should be a Gaussian with `x0 = 0.5d0`, `del = 0.1d0` and an amplitude given by the corresponding command-line argument. The difference values u_j^0 and u_j^1 are to be initialized to $O(\triangle t^3)$ using the Taylor series approach discussed in the lecture.

The second routine provided in `util.f` is

```
subroutine gnuout(u,x,nx,t,stride)
   implicit       none

   integer        nx,           stride
   real*8         u(nx),        x(nx),        t

   integer        j

   do j = 1 , nx , stride
      write(*,*) t, x(j), u(j)
   end do
   write(*,*)

   return

end
```

which writes data to standard output in `gnuplot` `splot`-format (as in the `gpwave` example). Assuming that the grid function `u` has been declared via

```
real*8 u(maxnx,2)
```

then the *only* output (to standard out) from your program should be generated every `ofreq` steps using a call like

```
call gnuout(u(1,np1),x,nx,t,ofreq)
```

When you are satisfied that your program is working, generate some sample output using:

```
wave1d 8 0.5 3 0.5 1.0 5 > out8
```

and then use `splot` in `gnuplot` to produce a surface plot of the results. Save a postscript version of your plot in the file `out8.ps`. Now try

```
wave1d 8 1.00025 3 0.5 1.0 5 > out8uns
```

and use `gnplot` to make a postscript file `out8uns.ps` containing a surface plot of the results. What appears to be happening to the solution in this case?