ABSTRACT

The use of partial differential equations spread rapidly in many different fields. People try to simulate phenomena and model them to describe their behavior in partial differential equations. But to make use of the partial differential equations in describing phenomena, we need to program them as a soft ware by using descretization techniques. My field is aerospace engineering and we need the partial differential equations very much in calculating aerodynamics fields and in flight control parameters.

The most important equations we are using are the flow equation, the wave equation and the heat equation. But to be able to solve these equations they need to be programmed by using any programming language. Here I am using C- programming language to program the three equations with some default initial and boundary conditions which can be changed just by changing the input of the program.

ACKNOWLEDGEMENT

"In the name of Allah (God), Most Gracious, Most Merciful. Read, In the name of thy lord and Cherisher, Who created man from a [leech - like] clot. Read, and thy Lord Is Most Bountiful, He Who taught [the use of] the pen. Taught man that which he know not. Nay, but man doth Transgress all bounds. In that he looketh upon himself as self- sufficient. Verily, to thy Lord is the return [of all]. “ (The Holy QURAN, Surah No. 96).

Above and first of all, I thank and pray to Allah for His guidance and protection throughout my life including the years of this study. I am happy to have had a chance to glorify His name, in the sincerest way, through his small accomplishment, and I ask Him, with hope in Him, to accept my efforts. And secondary my peace upon His Prophet, MOHAMMED (salla Allah alihe wa sallam). I wish to thank my direct supervisor **Dr. Siddiqi** for teaching us the course of partial differential equations, and for his support, comments, suggestions, constructive criticism, encouragement, knowledge and help through this semester.

TABLE OF CONTENTS

INTRODUCTION:

The use of partial differential equations spread rapidly in many different fields. People try to simulate phenomena and model them to describe their behavior in partial differential equations. But to make use of the partial differential equations in describing phenomena, we need to program them as a soft ware by using descretization techniques. My field is aerospace engineering and we need the partial differential equations very much in calculating aerodynamics fields and in flight control parameters.
The most important equations we are using are the flow equation, the wave equation and the heat equation. But to be able to solve these equations they need to be programmed by using any programming language. Here I am using C- programming language to program the three equations with some default initial and boundary conditions which can be changed just by changing the input of the program.

BACKGROUND INFORMATION:

➔Wave equation:

The one dimensional wave equation (with no source term) is

$$\frac{1}{c^2}\frac{\partial^2 \phi}{\partial t^2} = \frac{\partial^2 \phi}{\partial x^2},$$

where *c* is the wave speed. (From purely dimensional arguments, *c* must have dimensions $m\ s^{-1}$ and hence must represent a speed.)

The general solution, known as d'Alembert's solution, is

$$\phi(x,t) = F(x - ct) + E(x + ct)$$

which can be easily verified, using the chain rule. It is, however, so easy to derive that it is worthwhile going through the derivation: Introduce new variables

$$u = x - ct \qquad , \qquad v = x + ct$$

Then the wave equation can be written

$$\frac{\partial^2 \phi}{\partial u \partial v} = 0$$

Now, this immediately integrates to

$$\frac{\partial \phi}{\partial u} = h(u)$$

and then

$$\phi(u,v) = \int h(u)du + E(v) = F(u) + E(v) = F(x-ct) + E(x+ct) .$$

This can also be written in the form

$$\phi(x,t) = \hat{F}(t - x/c) + \hat{E}(t + x/c)$$

which is slightly more convenient in what follows.

The important points are that:

- $F(x\text{-}ct)$ represents a wave of shape $F(x)$ (at time t=0) moving in the direction of increasing x at constant speed c. That is, $F(x\text{-}ct)$ represents the same shape as $F(x)$, but with the origin moved to x=ct, ie the same shape as $F(x)$ translated by ct units in the positive x direction.
- $E(x+ct)$ represents a wave of shape $E(x)$ (at time t=0) moving in the direction of decreasing x at constant speed c. That is, $E(x+ct)$ represents the same shape as $E(x)$, but with the origin translated to x=$-ct$.

The following C program solves the wave equation

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$$

with boundary values

$$u(t,0)=u(t,1)=0$$

and initial conditions

$$\left.\frac{\partial u}{\partial t}\right|_{t=0} = 0 \;\; u(0,x) = \frac{1}{6}\sin \pi x \quad 0 \le x \le 1$$

```
#include <stdio.h>
#include <math.h>

void main(void)           /* Numerical solution of wave equation */
{                         /* CTJ Dodson, UMIST */
int x,t;                  /* Counters in solution array */
double u[21][11];         /* Array to hold solution values u(t,x) */
FILE *fp;

for (t=1;t<=20;t++)       /* Boundary values u(t,0)=u(t,1)=0 */
u[t][0]=u[t][10]=0;

for (x=0;x<=10;x++)                    /* Initial values u(0,x) */
u[0][x]=0.125*sin(0.1*x*3.14159);
for (x=0;x<=8;x++)
u[1][x+1]=0.5*(u[0][x]+u[0][x+2]);  /* Initial flow: du/dt = 0 at t=0
for all x */

for (t=1;t<=19;t++)       /* Explicit difference equation
approximation */
for (x=1;x<=9;x++)
u[t+1][x]=u[t][x-1]+u[t][x+1]-u[t-1][x];

fp=fopen("wave.out","w");   /* Open output file */
fprintf(fp, "\n\tNumerical solution u(t,x) to wave equation:
d^2u/dt^2=d^2u/dx^2\n");
fprintf(fp, "\tInitial rate: du/dt = 0 at t=0  for all x\n");
fprintf(fp, "Time step dt=0.1 s\t Space step dx=0.1\n");
fprintf(fp, "t sec\tPosition x->");
for (t=0;t<=20;t++)            /* Print values u(t,x) */
fprintf(fp,           /* You can break a line, but not inside a
format list*/
"\n%1.2f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f
%+2.4f%+2.4f %+2.4f",
t*0.1,u[t][0],u[t][1],u[t][2],u[t][3],u[t][4],u[t][5],u[t][6],u[t][7]
,u[t][8],u[t][9],u[t][10]);
fclose(fp);
/* Close output file */
}
```

The output file for this program should look something like this:

```
    Numerical solution u(t,x) to wave equation: d^2u/dt^2=d^2u/dx^2
    Initial rate: du/dt = 0 at t=0  for all x
Time step dt=0.1 s   Space step dx=0.1
t sec   Position x->
0.00 +0.0000 +0.0386 +0.0735 +0.1011 +0.1189 +0.1250 +0.1189 +0.1011 +0.0735+0.0386
+0.0000
0.10 +0.0000 +0.0367 +0.0699 +0.0962 +0.1131 +0.1189 +0.1131 +0.0962 +0.0699+0.0367
+0.0000
0.20 +0.0000 +0.0312 +0.0594 +0.0818 +0.0962 +0.1011 +0.0962 +0.0818 +0.0594+0.0312
+0.0000
0.30 +0.0000 +0.0227 +0.0432 +0.0594 +0.0699 +0.0735 +0.0699 +0.0594 +0.0432+0.0227
+0.0000
0.40 +0.0000 +0.0119 +0.0227 +0.0313 +0.0367 +0.0386 +0.0367 +0.0312 +0.0227+0.0119
+0.0000
0.50 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000+0.0000
+0.0000
0.60 +0.0000 -0.0119 -0.0227 -0.0312 -0.0367 -0.0386 -0.0367 -0.0313 -0.0227-0.0119
+0.0000
0.70 +0.0000 -0.0227 -0.0432 -0.0594 -0.0699 -0.0735 -0.0699 -0.0594 -0.0432-0.0227
+0.0000
0.80 +0.0000 -0.0312 -0.0594 -0.0818 -0.0962 -0.1011 -0.0962 -0.0818 -0.0594-0.0312
+0.0000
```

```
0.90 +0.0000 -0.0367 -0.0699 -0.0962 -0.1131 -0.1189 -0.1131 -0.0962 -0.0699-0.0367
+0.0000
1.00 +0.0000 -0.0386 -0.0735 -0.1011 -0.1189 -0.1250 -0.1189 -0.1011 -0.0735-0.0386
+0.0000
1.10 +0.0000 -0.0367 -0.0699 -0.0962 -0.1131 -0.1189 -0.1131 -0.0962 -0.0699-0.0367
+0.0000
1.20 +0.0000 -0.0312 -0.0594 -0.0818 -0.0962 -0.1011 -0.0962 -0.0818 -0.0594-0.0312
+0.0000
1.30 +0.0000 -0.0227 -0.0432 -0.0594 -0.0699 -0.0735 -0.0699 -0.0594 -0.0432-0.0227
+0.0000
1.40 +0.0000 -0.0119 -0.0227 -0.0312 -0.0367 -0.0386 -0.0367 -0.0313 -0.0227-0.0119
+0.0000
1.50 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000 +0.0000+0.0000
+0.0000
1.60 +0.0000 +0.0119 +0.0227 +0.0313 +0.0367 +0.0386 +0.0367 +0.0312 +0.0227+0.0119
+0.0000
1.70 +0.0000 +0.0227 +0.0432 +0.0594 +0.0699 +0.0735 +0.0699 +0.0594 +0.0432+0.0227
+0.0000
1.80 +0.0000 +0.0312 +0.0594 +0.0818 +0.0962 +0.1011 +0.0962 +0.0818 +0.0594+0.0312
+0.0000
1.90 +0.0000 +0.0367 +0.0699 +0.0962 +0.1131 +0.1189 +0.1131 +0.0962 +0.0699+0.0367
+0.0000
2.00 +0.0000 +0.0386 +0.0735 +0.1011 +0.1189 +0.1250 +0.1189 +0.1011 +0.0735+0.0386
+0.0000
```

➔**The heat equation:**

The **heat equation** is an important partial differential equation which describes the variation of temperature in a given region over time. In the special case of heat propagation in an isotropic and homogeneous medium in the 3-dimensional space, this equation is

$$u_t = k(u_{xx} + u_{yy} + u_{zz})$$

Where:

- $u(t, x, y, z)$ is temperature as a function of time and space;
- u_t is the rate of change of temperature at a point over time;
- u_{xx}, u_{yy}, and u_{zz} are the second spatial derivatives (*thermal conductions*) of temperature in the x, y, and z directions, respectively
- k is a material-specific constant called *thermal conductivity* .

The heat equation is a consequence of Fourier's law of cooling..

To solve the heat equation, we also need to specify boundary for u.

Solutions of the heat equation are characterized by a gradual smoothing of the initial temperature distribution by the flow of heat from warmer to colder areas of an object. As a consequence, to reverse the solution and conclude something about earlier times or initial conditions from the present heat distribution is very inaccurate except over the shortest of time periods.

The heat equation is the prototypical example of a parabolic partial differential equation.

Using the Laplace operator, the heat equation can be generalized to

$$u_t = k\Delta u,$$

Where the Laplace operator is taken in the spatial variables.

The heat equation governs heat diffusion, as well as other diffusive processes, such as particle diffusion. Although they are not diffusive in nature, some quantum mechanics problems are also governed by a mathematical analog of the heat equation. It also can be used to model some processes in finance.

The heat equation is, technically, in violation of special relativity, because its solutions involve instantaneous propagation of a disturbance. The part of the disturbance outside the forward light cone can usually be safely neglected, but if it is

necessary to develop a reasonable speed for the transmission of heat an equation with a second-order time derivative must be used.

The C program solves the heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

for boundary values

$$u(t,0)=u(t,1)=0$$

and initial conditions

$$u(0,x) = 2x \quad 0 \le x \le \frac{1}{2}$$

$$u(0,x) = 2(1-x) \quad \frac{1}{2} \le x \le 1$$

```
#include <stdio.h>
#include <math.h>

void main(void)            /* Numerical solution of heat equation */
{                          /* CTJ Dodson, UMIST */
int x,t;                  /* Counters in array for solution */
double u[21][11];
FILE *fp;                 /* File pointer for output */

for (t=1;t<=20;t++)       /* Boundary values:  u(t,0)=u(t,1)=0 */
u[t][0]=u[t][10]=0;

for (x=0;x<=5;x++)        /* Initial values */
u[0][x]=0.2*x;
for (x=6;x<=10;x++)       /* Initial values */
u[0][x]=2-0.2*x;

for (t=0;t<=19;t++)       /* Explicit difference equation
approximation */
for (x=1;x<=9;x++)
u[t+1][x]=0.1*u[t][x-1]+0.8*u[t][x]+0.1*u[t][x+1];   /* dt=0.001,
dx=0.1 */
fp=fopen("heat.out","w");  /* Open output file */
fprintf(fp, "\n\tNumerical solution u(t,x) to heat equation
du/dt=d^2u/dx^2 for rod\n");
fprintf(fp, "Time step dt=0.001 s \t Space step dx=0.1\n");
fprintf(fp, "t sec \t Position x->");
for (t=0;t<=20;t++)
fprintf(fp,          /* You can break a line, but not inside a format
list*/
"\n%1.3f %1.4f %1.4f %1.4f %1.4f %1.4f %1.4f %1.4f %1.4f %1.4f
%1.4f",0.001*t,
u[t][0],u[t][1],u[t][2],u[t][3],u[t][4],u[t][5],u[t][6],u[t][7],u[t][
8],u[t][9],u[t][10]);
        fclose(fp);           /* Close output file */
}
```

The output file for this program should look something like this:

```
    Numerical solution u(t,x) to heat equation du/dt=d^2u/dx^2 for rod
Time step dt=0.001 s     Space step dx=0.1
t sec    Position x->
0.000 0.0000 0.2000 0.4000 0.6000 0.8000 1.0000 0.8000 0.6000 0.4000 0.2000 0.0000
0.001 0.0000 0.2000 0.4000 0.6000 0.8000 0.9600 0.8000 0.6000 0.4000 0.2000 0.0000
0.002 0.0000 0.2000 0.4000 0.6000 0.7960 0.9280 0.7960 0.6000 0.4000 0.2000 0.0000
0.003 0.0000 0.2000 0.4000 0.5996 0.7896 0.9016 0.7896 0.5996 0.4000 0.2000 0.0000
0.004 0.0000 0.2000 0.4000 0.5986 0.7818 0.8792 0.7818 0.5986 0.4000 0.2000 0.0000
0.005 0.0000 0.2000 0.3998 0.5971 0.7732 0.8597 0.7732 0.5971 0.3998 0.2000 0.0000
0.006 0.0000 0.2000 0.3996 0.5950 0.7643 0.8424 0.7643 0.5950 0.3996 0.2000 0.0000
0.007 0.0000 0.1999 0.3992 0.5924 0.7551 0.8268 0.7551 0.5924 0.3992 0.1999 0.0000
0.008 0.0000 0.1999 0.3986 0.5893 0.7460 0.8125 0.7460 0.5893 0.3986 0.1999 0.0000
0.009 0.0000 0.1998 0.3978 0.5859 0.7370 0.7992 0.7370 0.5859 0.3978 0.1998 0.0000
0.010 0.0000 0.1996 0.3968 0.5822 0.7281 0.7867 0.7281 0.5822 0.3968 0.1996 0.0000
0.011 0.0000 0.1993 0.3956 0.5783 0.7194 0.7750 0.7194 0.5783 0.3956 0.1993 0.0000
0.012 0.0000 0.1990 0.3942 0.5741 0.7108 0.7639 0.7108 0.5741 0.3942 0.1990 0.0000
0.013 0.0000 0.1986 0.3927 0.5698 0.7025 0.7533 0.7025 0.5698 0.3927 0.1986 0.0000
0.014 0.0000 0.1982 0.3910 0.5653 0.6943 0.7431 0.6943 0.5653 0.3910 0.1982 0.0000
0.015 0.0000 0.1977 0.3892 0.5608 0.6863 0.7333 0.6863 0.5608 0.3892 0.1977 0.0000
0.016 0.0000 0.1970 0.3872 0.5562 0.6784 0.7239 0.6784 0.5562 0.3872 0.1970 0.0000
0.017 0.0000 0.1963 0.3851 0.5515 0.6708 0.7148 0.6708 0.5515 0.3851 0.1963 0.0000
0.018 0.0000 0.1956 0.3828 0.5468 0.6632 0.7060 0.6632 0.5468 0.3828 0.1956 0.0000
0.019 0.0000 0.1948 0.3805 0.5420 0.6559 0.6975 0.6559 0.5420 0.3805 0.1948 0.0000
0.020 0.0000 0.1939 0.3781 0.5373 0.6487 0.6891 0.6487 0.5373 0.3781 0.1939 0.0000
```

➔ **Flow equation:**

The flow equation is the partial differential equation that describes the flow pattern. We have many types of it used in different fields like the groundwater flow equation, the current flow equation, the pressure driven flow equation and many others.

Thefollwing C program solves the flow equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

for boundary values

$$u(t,0)=2t$$

and initial conditions

$$u(0,x) = x(x-2) \quad 0 \le x \le 2$$

$$u(0,x) = 2(x-2) \quad 2 \le x1$$

```
#include <stdio.h>
#include <math.h>

void main(void)           /* Numerical solution of flow equation */
{                         /* CTJ Dodson, UMIST */
int x,t;                  /* Counters in solution array */
double u[11][10];         /* Array to hold solution values u(t,x) */
FILE *fp;
for (x=0;x<=2;x++)       /* Initial values u(0,x) */
u[0][x]=0.25*x*(0.25*x-2);
for (x=3;x<=9;x++)       /* Initial values u(0,x) */
u[0][x]=2*(0.25*x-2);

for (t=0;t<=10;t++)       /* Boundary values u(t,0)=2t */
u[t][0]=0.25*t;

for (t=0;t<=9;t++)       /* Explicit difference equation
approximation */
for (x=1;x<=9;x++)

u[t+1][x]=0.375*u[t][x-1]+0.75*u[t][x]-0.125*u[t][x+1];

fp=fopen("flow.out","w");   /* Open output file */
fprintf(fp, "\n\tNumerical solution u(t,x) to fluid flow equation:
du/dt+du/dx=0\n");
fprintf(fp, "\tBoundary u(t,0)=2t\n Time step dt=0.125 s\t Space step
dx=0.25\n");
fprintf(fp, "t sec\tPosition x->");
for (t=0;t<=10;t++)            /* Print values u(t,x) */
fprintf(fp,          /* You can break a line, but not inside a format
list*/
```

```
"\n%2.3f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f %+2.4f
%+2.4f %+2.4f",
t*0.125,
u[t][0],u[t][1],u[t][2],u[t][3],u[t][4],u[t][5],u[t][6],u[t][7],u[t][
8],u[t][9]);
fclose(fp);
/* Close output file */
}
```

The output file for this program should look something like this:

```
    Numerical solution u(t,x) to fluid flow equation: du/dt+du/dx=0
    Boundary u(t,0)=2t
 Time step dt=0.125 s    Space step dx=0.25
t sec   Position x->
0.000 +0.0000 -0.4375 -0.7500 -2.5000 -2.0000 -1.5000 -1.0000 -0.5000 +0.0000 +0.5000
0.125 +0.2500 -0.2344 -0.4141 -1.9062 -2.2500 -1.7500 -1.2500 -0.7500 -0.2500 +0.3438
0.250 +0.5000 -0.0303 -0.1602 -1.3037 -2.1836 -2.0000 -1.5000 -1.0000 -0.5117 +0.1016
0.375 +0.7500 +0.1848 +0.0315 -0.7649 -1.8766 -2.1313 -1.7500 -1.2485 -0.7715 -0.2095
0.500 +1.0000 +0.4159 +0.1885 -0.3273 -1.4279 -2.0835 -1.9557 -1.4962 -1.0206 -0.5714
0.625 +1.2500 +0.6634 +0.3383 +0.0037 -0.9332 -1.8536 -2.0610 -1.7280 -1.2551 -0.9675
0.750 +1.5000 +0.9240 +0.5020 +0.2463 -0.4668 -1.4825 -2.0249 -1.9120 -1.4684 -1.3838
0.875 +1.7500 +1.1927 +0.6922 +0.4313 -0.0724 -1.0338 -1.8356 -2.0098 -1.6453 -1.8073
1.000 +2.0000 +1.4643 +0.9125 +0.5921 +0.2367 -0.5731 -1.5132 -1.9900 -1.7617 -2.2224
1.125 +2.2500 +1.7341 +1.1595 +0.7567 +0.4712 -0.1519 -1.1010 -1.8397 -1.7897 -2.6087
1.250 +2.5000 +1.9994 +1.4253 +0.9434 +0.6561 +0.2004 -0.6528 -1.5690 -1.7061 -2.9402
```

CONCLUSION:

In this paper there were very help initiated programs to solve the wave equation, the heat equation and the flow equation. Just by changing the initial and boundary conditions we can solve different problems by using the same programs. The results are very accurate and can be used in industrial purposes and in different manufacturing processes.

References:

1. http://www.geocities.com/aboutwaves/node71.html

2. http://en.wikipedia.org/wiki/Heat_equation