Finite difference computation of the Elliptic Poisson Partial Differential Equation

# Purpose

## Physical context

The purpose is to solve the Elliptic Poisson Partial Differential Equation with help of the Finite Difference method. More particularly we solve it with a five point difference scheme in two dimensional space and Dirichlet Boundary Condition. The explanation about the method has been taken from [[1](#Bur10)] and [[2](#Cha10)].

The problem comes historically from the following Heat Equation of Fourier:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

In two dimensional spaces and for any general real function in may be written as below:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

We note that the function is not anymore dependent of time. For instance, concerning the Heat Equation, it corresponds to the steady-state regime. In the particular case of, the equation (2) is known as the Laplace Equation.

## Discretization

The problem is discretized over a rectangular area with the abscissa: and. The step over the abscissa is given by and over ordinates by. The coordinate of any point in the grid is therefore given by:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

The Taylor series on let us generate the symmetrical difference formula for the second order partial derivation of.

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

And the Poisson Equation may be written as:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

## The finite difference method

### 5-points Finite Difference Scheme

By considering the higher order terms as negligible and by multiplying each side by, the equation (5) lead to the following general scheme in the case of an irregular step-size grid:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

In the particular case of a regular grid of identical step size along the abscissa and the ordinates one obtain the now widespread formula:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

## Error

1. *Data errors* analysis are left to the particular type of data we use (if they are experimental for instance)
2. *Truncation error* is a consequence of the approximation of the Taylor formula and lead to an estimation of. If the grid is regular i.e. then the truncation error is in . See [[3](#LeV07)] for an in-depth analysis of why the truncation error calculated for a particular point is still valid when speaking about the whole are the equation is calculated on.
3. *The Round-off error* is due to the particular tool we use for computation and computer.

# Protocol

## The computation

1. A matrix of dimension and a vector of dimension
   1. An index is calculated for each point. This index is of prime importance for the computational efficiency of the solution and is commented below.
   2. Each line is filled with coefficient of the scheme at the proper index,
2. The corresponding component of the vector is calculated from the value of the function at the boundary if it is a boundary point else its value is 0.
3. The particular system is solved and gives directly the final value of the point at index

## Note on the computation

### The matrix index

The formula of the index used is:

There is many ways of indexing each point in the matrix ( In fact but this one insure us of a resulting banded matrix of width at most [[1](#Bur10)]. But “*this clearly gives the optimal matrix structure for the purpose of applying Gaussian elimination”* [[3](#LeV07)].

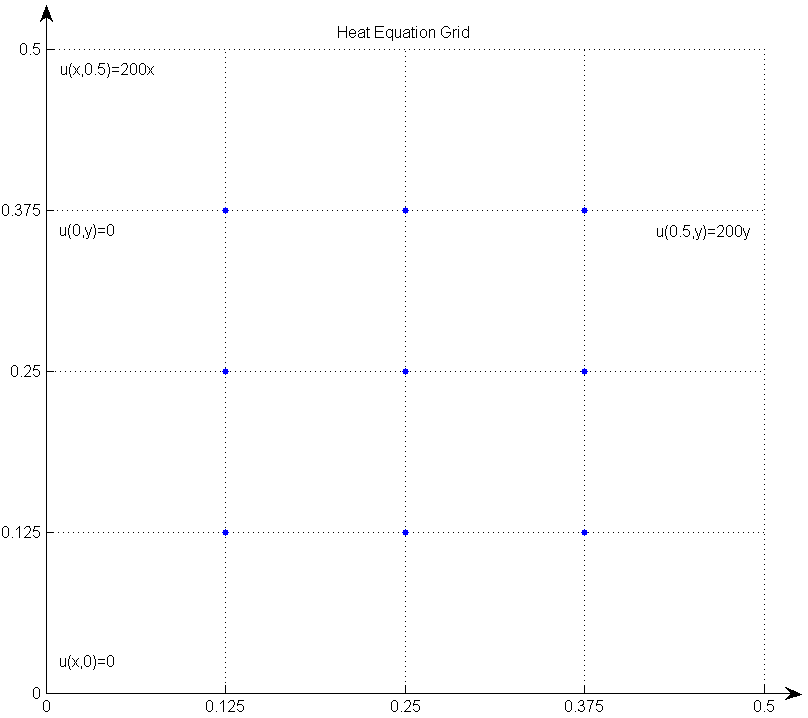
### Solving the system

A lot of methods are available to solve the equation. But as is a big and sparse matrix, just some little system of a hundred points are calculable by direct algebraic methods such as Gauss Elimination. In general, the problems encountered in physics, engineering… give birth to very big sparse matrices that have to be solve by iterative methods such as Jacobi Method, Liebman Method (Gauss-Seidel) [[4](#Kiu05)] or Simultaneous Algebraic Reconstruction Techniques (SART) [[4](#AIR)] currently use for tomography reconstruction.

# Result

## The computation matrix

We perform the algorithm on the grid shown below:



The matrix and vector give:

|  |  |
| --- | --- |
|  |  |
| 4 -1 0 -1 0 0 0 0 0  -1 4 -1 0 -1 0 0 0 0  0 -1 4 0 0 -1 0 0 0  -1 0 0 4 -1 0 -1 0 0  0 -1 0 -1 4 -1 0 -1 0  0 0 -1 0 -1 4 0 0 -1  0 0 0 -1 0 0 4 -1 0  0 0 0 0 -1 0 -1 4 -1  0 0 0 0 0 -1 0 -1 4 | 25  50  150  0  0  50  0  0  25 |

And the values for x are found (see [[1](#Bur10)]) after 250 iterations with SART method:

18.7500 12.5000 6.2500

37.5000 25.0000 12.5000

56.2500 37.5000 18.7500

## Graphical result

# Bibliography

x

|  |  |
| --- | --- |
| [1] | Burden R L. and Faire J D, *Numerical Analysis*, 9th ed.: Brooks/Cole, 2010. |
| [2] | S C Chapra and R P Canale, *Numerical Methods For Engineers*, Sixth Edition ed., Mac Graw Hill, Ed., 2010. |
| [3] | LeVeque R J., *Finite Difference Methods for Ordinary and Partial Differential Equations*. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2007. |
| [4] | P C. Hansen and Saxild-Hansen M, *AIR Tools - A MATLAB Package of Algebraic Iterative Reconstruction Methods*.: Journal of Computational and Applied Mathematics, 2012, vol. Vol. 236, No. 8. |

x