Technion - Israel Institute of Technology

Numerical Methods in Aeronautical Engineering (086172)

GRADE	OUT OF	CHAPTER
	2	ABSTRACT
	2	CONTENTS, STYLE &C.
	4	PHYSICAL PROBLEM
	4	MATHEMATICAL MODEL
	26	NUMERICAL METHODS
	20	INFLUENCE OF NUMERICAL METHODS
	20	RESULTS
	2	SUMMARY & CONCLUSIONS
	20	COMPUTER PROGRAM
	100	**TOTAL**

Daniel Engelsman, ID 300546173 @ May 16, 2019

= Homework Assignment 2 =

Contents

List of Figures

1 Abstract

In this assignment we are asked to solve a model of an infinite oscillating cylinder in an incompressible Newtonian fluid at rest. Given one initial time condition and one boundary condition, we shall later see how to obtain another initial condition. Luckily, the PDE happened to resolved using an explicit finite differences method, sparring the need for a more elaborated methods (θ, C-N, etc.), which was technically easier.

Along the report I will examine the influence of each numeric parameter on the robustness of the solution, and will try to suggest an optimality. The final solution presents the propagation of the cylinder's motion along time, and we'll examine the oscillations's sensitivity with respect to the cylinder's radius and time.

2 The physical problem

In this problem the infinite cylinder undergoes an harmonic oscillation in a direction normal to its radial axis in an incompressible Newtonian fluid at rest. The fluid is characterized by a linear viscous stress that arises from its flow, and is proportional to the local strain rate. The center of the cylinder is at rest, but distant to the center, we shall examine how disturbances develop and propagate.

3 The mathematical model

Given the PDE equation :

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} - \frac{u}{r^2}, \qquad 0 \le r < 1, \qquad 0 \le t \le 10 \tag{3.1}$$

$$b.c.: \qquad u(1,t) = sin(\omega t), \qquad u(r,0) = 0 \tag{3.2}$$

We'll extract another condition by multiplying Eqn. 3.1 by r^2 and rearrange the equation :

$$r^2\frac{\partial u}{\partial t} = r^2\frac{\partial^2 u}{\partial r^2} + r\frac{\partial u}{\partial r} - u(r,t)\Big|_{r\to 0} \quad \to \quad u(0,t) = 0$$

4 The numeric method

I chose to use the finite difference method on the $u_{i,j} = u(r_0 + ih, t_0 + jk)$ plane, such that indices are calculated due to the following sub-intervals sizes :

$$j = 0 : dt : \frac{10}{dt}, \quad k \equiv dt; \qquad i = \epsilon : dr : \frac{1.0}{dr}, \quad h \equiv dr; \tag{4.1}$$

We'll do the following discretization, note that (t_j - forward F.D, r_i - central F.D) :

$$u(r,t) = R(r)T(t) = u_{i,j}, \qquad \frac{\partial u}{\partial t} = \frac{u_{i,j+1} - u_{i,j}}{k} + O(k)$$

$$\frac{\partial u}{\partial r} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + O(h^2), \qquad \frac{\partial^2 u}{\partial r^2} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + O(h^2) \tag{4.2}$$

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + \frac{1}{r_i}\frac{u_{i+1,j} - u_{i-1,j}}{2h} - \frac{u_{i,j}}{r_i^2} + O(k + h^2) \tag{4.3}$$

Let us multiply the equation by k and rearrange by using auxiliary terms :

$$Define: \quad R \equiv \frac{k}{h^2}, \qquad Q \equiv \frac{k}{2h} \tag{4.4}$$

$$Plug: \quad u_{i,j+1} = u_{i,j} + R\big(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\big) + \frac{Q}{r_i}(u_{i+1,j} - u_{i-1,j}) - \frac{k}{r_i^2}u_{i,j}$$

$$sort: \quad u_{i,j+1} = u_{i-1,j}[R - \frac{Q}{r_i}] + u_{i,j}[1 - 2R - \frac{k}{r_i^2}] + u_{i+1,j}[R + \frac{Q}{r_i}] \tag{4.5}$$

The method (Eqn. 4.5) is expressed explicitly with a truncation error of $\underline{\underline{O(k + h^2)}}$. The radius is given between $\bar{r} \in [r_0, r_{N+1}]$, and every j_{th} time iteration solves the $u(\bar{r}, j)$ profile :

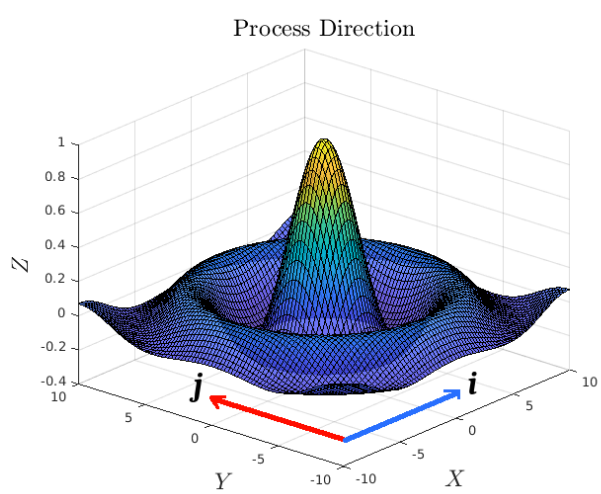

Figure 1: General demonstration of the numerical approach

The initial and boundary conditions can be expressed numerically :

$$u_{0,j} = u(0,t) = 0, \qquad u_{N+1,j} = u(1,t) = sin(\omega \cdot t_j) \tag{4.6}$$

Arranging (Eqn. 4.5) in a matrix form, and applying the boundary conditions (Eqn. 4.6) in vector b, we get a system of linear equations - $\underline{u}_{i,j+1} = \underline{A} \cdot \underline{u}_{i,j} + \underline{b}$:

$$\begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ ... \\ ... \\ ... \\ u_{N,j+1} \end{bmatrix} = \begin{bmatrix} 1-2R-\frac{k}{r_1^2} & R+\frac{Q}{r_1} & 0 & ... & 0 \\ R-\frac{Q}{r_2} & ... & ... & ... & ... \\ 0 & ... & ... & ... & ... \\ ... & ... & ... & ... & 0 \\ ... & ... & ... & ... & R+\frac{Q}{r_{N-1}} \\ 0 & ... & 0 & R-\frac{Q}{r_N} & 1-2R-\frac{k}{r_N^2} \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ ... \\ ... \\ ... \\ u_{N,j} \end{bmatrix} + \begin{bmatrix} u_{0,j}[R-\frac{Q}{r_1}] \\ 0 \\ 0 \\ 0 \\ 0 \\ u_{N+1,j}[R+\frac{Q}{r_N}] \end{bmatrix}$$

5 Influence of the numerical methods

The exlipcit method's main benefit is the formulation's simplicity, and relatively short time until reaching the validation phase. Contrarily, one cannot ignore the arisen drawbacks :

∘ Non-physical solution - the method suffers from a conditional stability that may distort the physical solution, caused by the $R = \frac{k}{h^2}$ parameter (Eqn. 4.4).

∘ Accuracy cost - in order to satisfy sufficient R value, the time axis becomes directly dependent on the problem's duartion and oppositely to the time k step size.

∘ Computational cost - as time varies from $t_j \in [0, 10]$. Small k values would yield huge time axis with apprx. $j \sim 10^6$ steps, that forces tough computational calculations for $u_{i,j}$.

It is therefore required to empirically examine the permissible range of R that will satisfy the solution space, considering the rule of thumb of $R \leq \frac{1}{2}$. Additionally, I shall compare computation times between different values of h and k in search of optimality.

Given no validation reference and having failed to solve the problem analytically, no "ground truth" validation exists for the sake of validation. It is therefore prompted to empirically look for the solution's sensitivity under different values of dt and dr.

Hereby is presented a table that concentrates a grid of $R = f(h, k)$ in sought of permissible range. $R > \frac{1}{2}$ values (*red*) have been ignored for the diverged solution they produced :

R		h_size											
		0.0050	0.0066	0.0086	0.0113	0.0149	0.0195	0.0256	0.0336	0.0442	0.0580	0.0762	0.1000
k_size	0.00001	0.40000	0.23201	0.13457	0.07806	0.04527	0.02626	0.01523	0.00883	0.00512	0.00297	0.00172	0.00100
	0.00003	1.20213	0.69727	0.40444	0.23458	0.13607	0.07892	0.04578	0.02655	0.01540	0.00893	0.00518	0.00301
	0.00009	3.61280	2.09553	1.21547	0.70500	0.40892	0.23719	0.13757	0.07980	0.04628	0.02685	0.01557	0.00903
	0.00027	10.8577	6.2978	3.6529	2.1188	1.2289	0.7128	0.4135	0.2398	0.1391	0.0807	0.0468	0.0271
	0.00082	32.6309	18.9268	10.9781	6.3676	3.6934	2.1423	1.2426	0.7207	0.4180	0.2425	0.1406	0.0816
	0.00245	98.0666	56.8814	32.9928	19.1368	11.0999	6.4382	3.7344	2.1660	1.2564	0.7287	0.4227	0.2452
	0.00737	294.72	170.95	99.15	57.51	33.36	19.35	11.22	6.51	3.78	2.19	1.27	0.74
	0.02214	885.74	513.75	297.99	172.84	100.25	58.15	33.73	19.56	11.35	6.58	3.82	2.21
	0.06655	2661.94	1544.00	895.56	519.45	301.30	174.76	101.37	58.80	34.10	19.78	11.47	6.65
	0.20000	8000.00	4640.23	2691.46	1561.12	905.50	525.21	304.64	176.70	102.49	59.45	34.48	20.00

Figure 2: $R = \frac{k}{h^2}$ table

Afterwards, amidst the valid R's, I compared for the best computation time. Note that some values (*orange*) exhibited divergence despite satisfying $R < \frac{1}{2}$:

Run-time		h_size											
		0.0050	0.0066	0.0086	0.0113	0.0149	0.0195	0.0256	0.0336	0.0442	0.0580	0.0762	0.1000
k_size	0.00001	5.51083	5.34402	4.03206	2.89277	2.49072	1.98357	1.79395	1.70873	1.68522	1.65462	1.56655	1.46872
	0.00003	0.00000	0.00000	1.73196	0.91579	0.79600	0.68063	0.64037	0.55672	0.51174	0.52955	0.53677	0.52150
	0.00009	0.00000	0.00000	0.00000	0.00000	0.27324	0.25029	0.23013	0.20058	0.18855	0.17461	0.17281	0.17444
	0.00027	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0693	0.0691	0.0630	0.0637	0.0614	0.0681
	0.00082	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.00245	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.00737	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.02214	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.06655	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.20000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 3: $arg\min\ \ G(t)$ table

The solution is highly sensitive to small values of k's, as it imposes a calculation of huge dimensions $j \sim \frac{10}{10^{-5}} \approx 10^6$. On the other hand, the relatively "diagonal" of that table shows the biggest permissible values $R \to \frac{1}{2}$, that any further k reduction would lead into divergence. The optimal values are gained around $k^* \sim 3 \cdot 10^{-4}$, while the dr step size showed stagnation from a certain level of $h^* > 0.2$. That might imply on a reasonable convergence region, for its insensitivity to dr's size.

We shall see a 3D representation of that data (*red* = impermissible). Note the logarithmic plane's axes, and the grayscale colorbar that denotes the runtimes (darker = faster) :

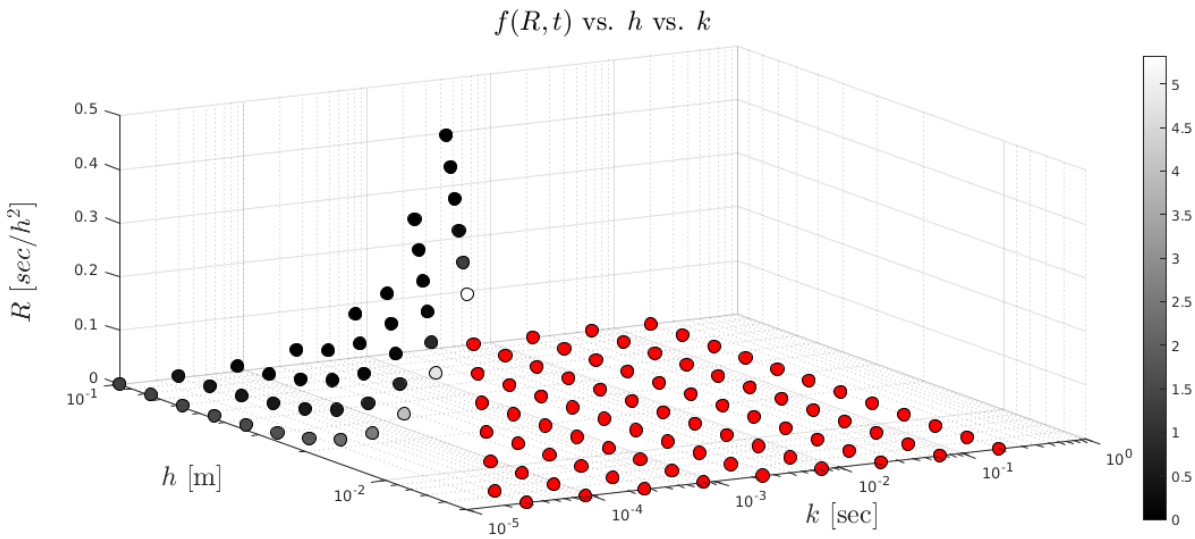

Figure 4: Solution space

Optimal h, k values are picked automatically at each run of the main script, by extracting the minimal runtime performance.

6 Results

Following are the results of the chosen optimal pair (h^*, k^*), after downsizing the surface resolution (see subsection 6). Initially, at $u(r, 0)$, the cylinder's radius is at complete rest, while its center $(r = 0)$ exhibits a subtle sinusoidal motion on the $u(0, t)$ profile. As time goes by, the disturbances propagate in that wavelike harmonic motion, and we can cautiously say that is <u>somewhat</u> proportional to the radius.

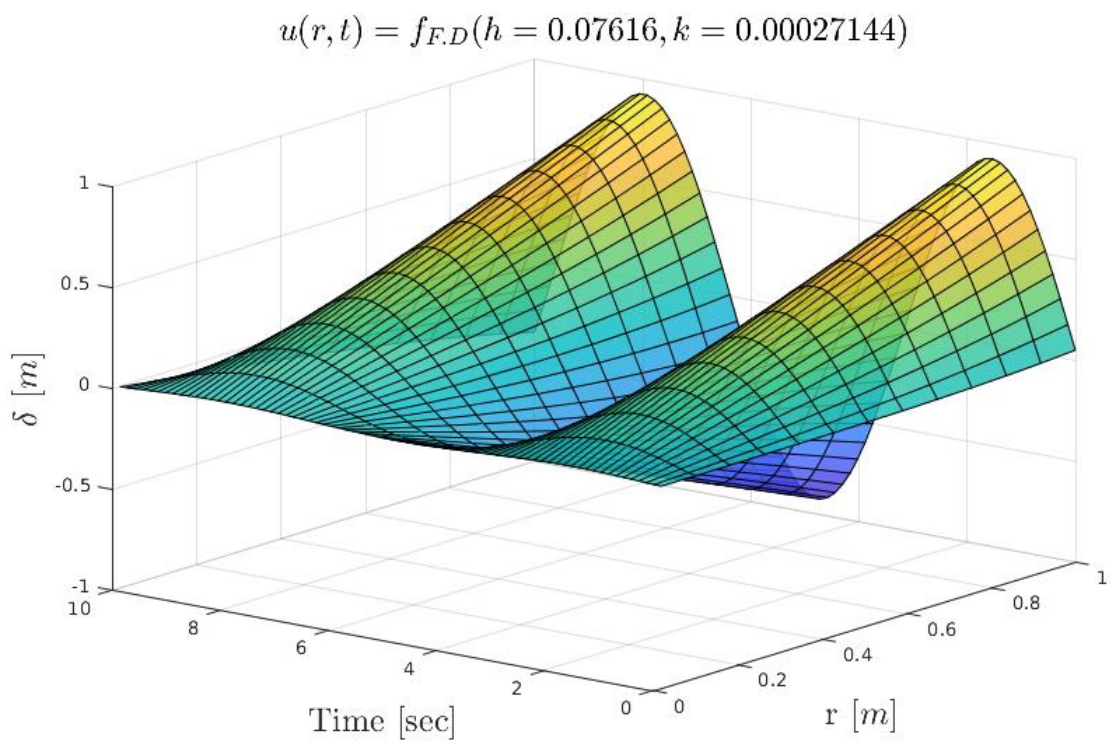

Figure 5: Solution for $u_{i,j}(\omega = 1)$

Using $\omega = 1$ yields ~ 1.5 sinusoidal cycles along all the radial axis. Additionally, the solution is bounded between $|u| \leq 1$ implying that the problem is normalized A priori, and no influential coefficients are exist to change that balance and cause acute violations.

∘ Although not given in the problem, I presumed physical units of a MKS system.

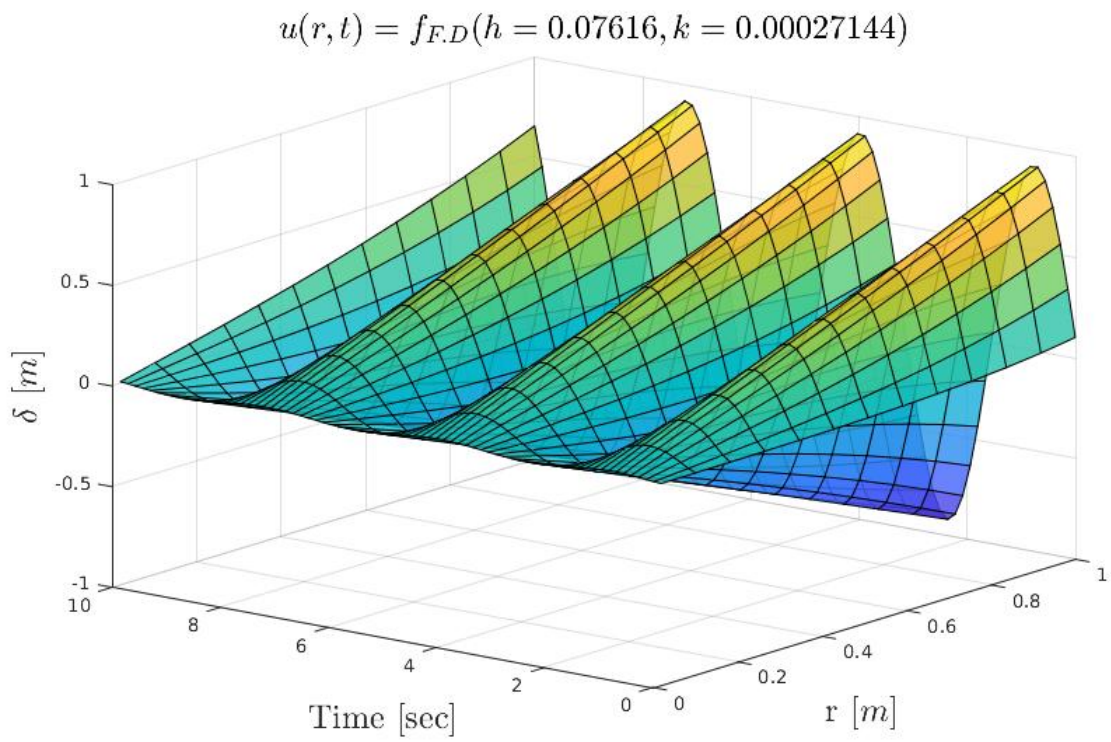

Figure 6: Solution for $u_{i,j}(\omega = 2)$

Similarly to the mentioned above, but now using a doubled ω frequency, the solution results expectedly in a close manner. The frequency stands proportionally to smaller periods that allow the moving disturbances along the radius to complete ~ 3 cycles. Other than that, no any further notable difference has been found.

Problematics of the mesh size

Using some of the values imposed a technical difficulty to plot the 3D matrics $(h = 0.005, k = 10^{-}5) \rightarrow u \sim \mathbb{R}^{200\times10^6}$. Unsurprisingly, my computer failed to convey that 3D visualization due to a weak graphic card. Eventually, I found a resourceful solution - partition the grand matrix into set of submatrices by a desired factor, and then calculating the mean each submatrix.

The following graphs were plotted using that downsizing method - $u_{reduced} \sim \mathbb{R}^{50\times150}$. **Note** that a slight inaccuracies are expected to take place due to that averaging. See the *downSize.m* algorithm attached in the code section for a detailed decription.

7 Summary and conclusion

∘ The problem was solved using an explicit finite differences method with a truncation error of $O(k + h^2)$, on a mesh resolution of $dr = 0.07616$, and $dt = 2.7 \cdot 10^{-5}$, that were found optimally.

∘ Avoid using robust methods such as θ, C-N, etc., limited the permissible range of R's, and imposed a technical computation challenge in search of a non-diverged solution.

∘ Given no validation reference, the influence of the numeric method was examined per se, looking for sensitivity region and optimality.

∘ The center of the cylinder is at rest, but as the radius grows, the disturbances' propagation magnitude, exacerbates within time.

∘ The solution's periodical cycle is influenced almost proportionally to the ω's frequency.

8. Computer program

The main script :

```
clc; close all; clear; set(0, 'defaultfigurecolor', [1 1 1]);
% ------------------------ Description ----------------------- %
%                                                               %
%              The main script of the program                   %
%                                                               %
% ----------- Initialize values of the problem ------------- %
% [k, h]     = deal(1e-4, 2e-2);
[r_0, r_f, t_0, t_f ] = deal(h,   1.0, k, 10);
[h_max, k_max, omega] = deal(0.1, 0.2, 1);
marker_size = 70;

% ---------------- Mesh of permissible R's ----------------- %
[mesh_x, mesh_y] = deal(12, 10);
h_mesh = logspace(log10(0.005),    log10(h_max), mesh_x);
k_mesh = logspace(log10(0.00001), log10(k_max), mesh_y);
R = k_mesh'./h_mesh.^2; R_log = (R < 0.5); R_valid = R_log .* R;
Run_time = zeros(mesh_y, mesh_x);

% --------------------- Solution space --------------------- %
for i = 1:mesh_y
    for j = 1:mesh_x
        if ( R_valid(i, j) ~= 0 )           % Calculate current f(h, k)
            [~, u_valid, Run_time(i, j)] = Hw_2_calc_u(h_mesh(j), k_mesh(i));
            if u_valid > 1                  % Sanity check
                Run_time(i, j) = 0;
            end
        end
    end
end

figure; hold on; grid on;
t_max = max(max(Run_time));
for i = 1:mesh_y
    for j = 1:mesh_x
        if ( R_valid(i, j) ~= 0 )
            scatter3(k_mesh(i), h_mesh(j), R_valid(i, j), marker_size, ...
                'MarkerEdgeColor','k','MarkerFaceColor', (Run_time(i, j)/t_max)*[1 1 1
        else
            scatter3(k_mesh(i), h_mesh(j), R_valid(i, j), marker_size, ...
                'MarkerEdgeColor','k','MarkerFaceColor', [1 0 0]);
        end
    end
end

% -------------------- Graph properties -------------------- %
set(gca, 'xScale', 'log');
set(gca, 'YScale', 'log');
colormap(gray(256)); colorbar; caxis([0 t_max]); view(-34, 21);
```

```
ind(1) = title('$f(R, t)$ vs. $h$ vs. $k$');
ind(2) = xlabel('$k$ [sec]');
ind(3) = ylabel('$h$ [m]');
ind(4) = zlabel('$R$ [${sec}/{h^2}$]');
set(ind, 'Interpreter', 'latex', 'fontsize', 18 );

%% -------- Extract optimal R w.r.t time performance -------- %

min_val = min( Run_time(Run_time(:)>0)  );
[m_min, n_min] = find(Run_time == min_val);
[h, k] = deal(h_mesh(n_min), k_mesh(m_min));

for i = 1:2
    omega = i;
    % ----------------- Plot the Optimal graph ----------------- %
    t = t_0:k:t_f; t_N = length(t);
    r = r_0:h:r_f; r_N = length(r); r = r';
    [u, ~, ~] = Hw_2_calc_u(h, k);

    [m, n] = size(u);
    [m_r, n_r] = deal(1, 500);              % Reduction factor
    t_s = linspace(t_0, t_f, floor(n/n_r));
    r_s = linspace(r_0, r_f, floor(m/m_r)); %r_s = r_s';
    u_s = Hw_2_downsize(u, m_r, n_r);       % Down size the matrix' dimensions

    figure;
    surf(r_s, t_s, u_s');                   % Plot the reduced dimension matrix
    alpha(0.8); view([-52 22]);

    % ------------- Labels ------------- %
    ind(1) = title(['$u(r, t) = f_{F.D}(h=', num2str(h), ', k=', num2str(k) ')$']);
    ind(2) = xlabel('r [$m$]');
    ind(3) = ylabel('Time [sec]');
    ind(4) = zlabel('$\delta$ [$m$]');
    set(ind, 'Interpreter', 'latex', 'fontsize', 18 );

end
```

The finite differences numeric algorithm :

```
function [u, u_valid, run_time] = Hw_2_calc_u(h, k)
% ------------------------ Description ---------------------- %
%                                                              %
%           This function calculates the main PDE              %
%                                                              %
% ----------- Initialize vectors of the problem ------------ %
global omega t_0 t_f r_0 r_f
t = t_0:k:t_f; t_N = length(t);
r = r_0:h:r_f; r_N = length(r); r = r';

% --------- Numeric Method == u_{j,i} <=> u_{r, t} ---------- %
u_m1 = eps;
u_p1  = @(t) sin(omega*t);              % B.C is f(t) and changes by j
```

```
u = zeros(r_N, t_N);                    % Fulfill boundary conditions
nOnes = ones(r_N, 1);
[R,   Q] = deal((k/h^2), (k/(2*h)) );
[Rm, Qm] = deal(nOnes * R, nOnes(1:end-1) * Q );

% ------------------ Consruct Matrix From ------------------ %
C_1 = diag(Rm(1:end-1) + Qm./r(1:(end-1)), 1);
C_2 = diag(1  - 2*Rm   -  k./r.^2,         0);
C_3 = diag(Rm(2:end)   - Qm./r(2:end),    -1);
A_r = C_1 + C_2 + C_3; b = zeros(r_N, 1);

% ------- Extract vector u_{j,:} every time step (j) ------- %
tic;
for j = 1:t_N-1
    [b(1), b(end)] = deal(u_m1*(R - Q/r(1)), u_p1(t(j+1))*(R + Q/r(end)) );
    u(:, j+1) = A_r*u(:, j) + b;
end            % --- Final matrix - u_{T, R} --- %
run_time = toc;               % Return elapsed computation time
u_valid = abs(max(max(u)));
```

The dimensionality reduction algorithm :

```
function A_reduced = Hw_2_downsize(u, m_r, n_r)
% ------------------------ Description ---------------------- %
%                                                              %
%            Dimensionallity reduction algorithm               %
%                                                              %
% ----------- Initialize vectors of the problem ------------ %
global t_0 t_f r_0 r_f
[m, n] = size(u);
[i_r, j_r] = deal(floor(m/m_r), floor(n/n_r));   % maximal number to stay in range
A_reduced = zeros(i_r, j_r);                     % New reduced matrix

% --------- Extract mean values from block matrix --------- %
for i = 1:i_r
    for j = 1:j_r
        A_reduced(i, j) = mean( mean( u( 1+m_r*(i-1):m_r*i, 1+n_r*(j-1):n_r*j )));
    end
end
```