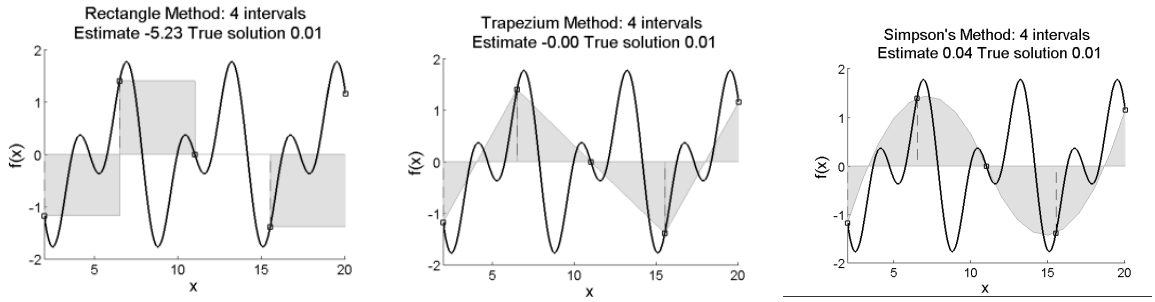
Computational Methods

Coursework 2 Solution

Mohammed AbuSadeh, K21036444

Q1) Please describe in less than 100 words which of the Newton-Cotes methods you would use to integrate function f(x)=cos(x)+sin(2x) over [0,2π] and why.

Ans: The Newton Cotes method with the closest estimation of the true value of the integral for the function f(x)=cos(x)+sin(2x) over [0,2π] should be the Trapezoid Method of integration. Since the function f(x) is an oscillatory function, the Rectangle method would be inaccurate in this case to have an accurate representation of the function. The Simpsons method could potentially yield an accurate and representative result close to the true value of the integral, however, it requires more intervals than the ones that have been given in this case. Thus, the Trapezoid method is the fastest upon the three methods to converge to a value very close to the true value.



Source: Lecture 9 Slides, Computational Methods, 5CCYB070.

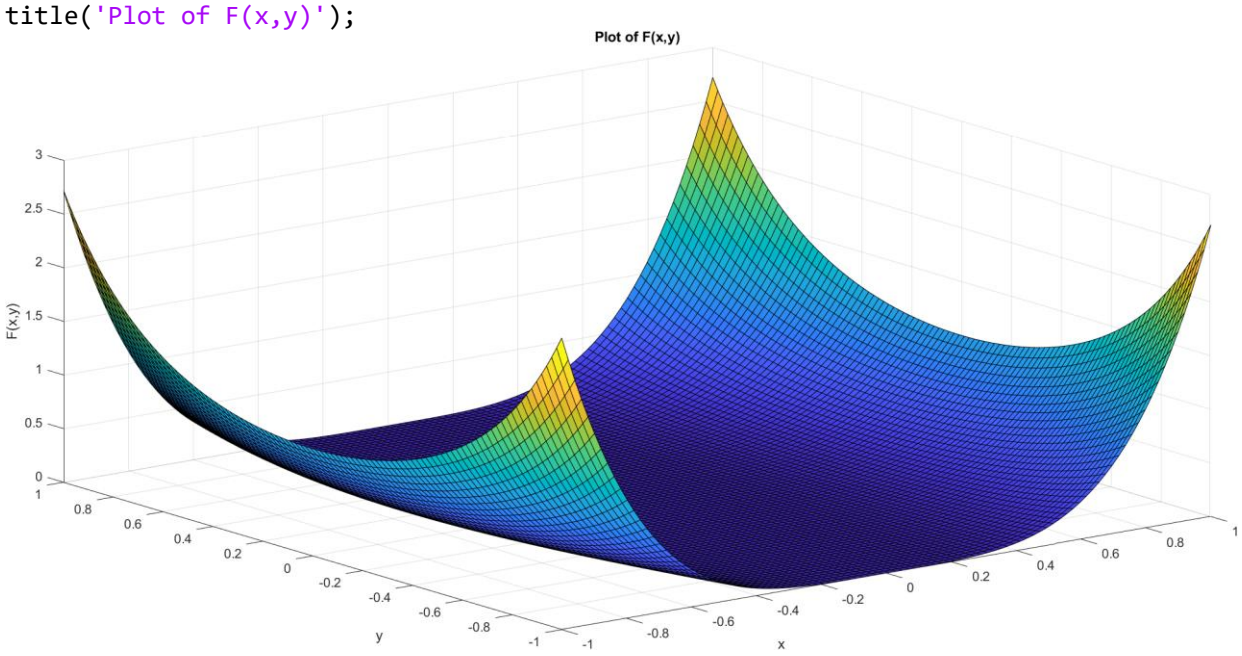
Q2)

1)

CODE :

```
% Define the function
F = @(x,y) (x.^4).*exp(y.^2);

% Setting up the plot
xmin = -1;
xmax = 1;
ymin = -1;
ymax = 1;
[X,Y] = meshgrid(linspace(xmin,xmax), linspace(ymin,ymax));
Z = F(X,Y);

% Create the Graph
figure;
surf(X,Y,Z);
xlabel('x');
ylabel('y');
```

```
zlabel('F(x,y)');
title('Plot of F(x,y)');
```



2)

Number of Gauss Points in each direction	Gaussian Integration
1	0.3398
3	3.1910
5	3.2897

Code is provided in the MATLAB Folder

3) Code: (Includes the Gaussian Integration Algorithm)

```
close all; clearvars; clc;


% Define the function
F = @(x,y) (x.^4).*exp(y.^2);

% Setting up the plot
xmin = -1;
xmax = 1;
ymin = -1;
ymax = 1;
[X,Y] = meshgrid(linspace(xmin,xmax), linspace(ymin,ymax));
Z = F(X,Y);

% Create the Graph
figure;
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('F(x,y)');
title('Plot of F(x,y)');

%% Q2) 3)

%define integration method
sel_meth = -1;
disp('Integration Method:');
disp('1) Rectangular,     2)  Trapezium,     3)  Simpsons     4)  Gaussian');

%ensure that correct number is provided
while (sel_meth~=1 && sel_meth~=2 && sel_meth ~=3 && sel_meth ~=4 )

    sel_meth = input('Please Select integration method: 1,2,3 or 4  ');

end

%Inputing the lower limits of integration for x and y
x_min=inf; y_min=inf;
disp('Set the lower limits for integration:');

while (isinf(x_min) && isnumeric(x_min))
    x_min = input('In the x direction: ');
end

while (isinf(y_min) && isnumeric(y_min))
    y_min = input('In the y direction: ');
end

%Inputing the upper limits of integratin for x and y
x_max=-inf; y_max=-inf;
disp('Set the upper limits for integration:');

while (isinf(x_max) && isnumeric(x_max) && (x_max<x_min) )
    x_max = input('In the x direction: ');
end

while (isinf(y_max) && isnumeric(y_max) && (y_max<y_min) )
```

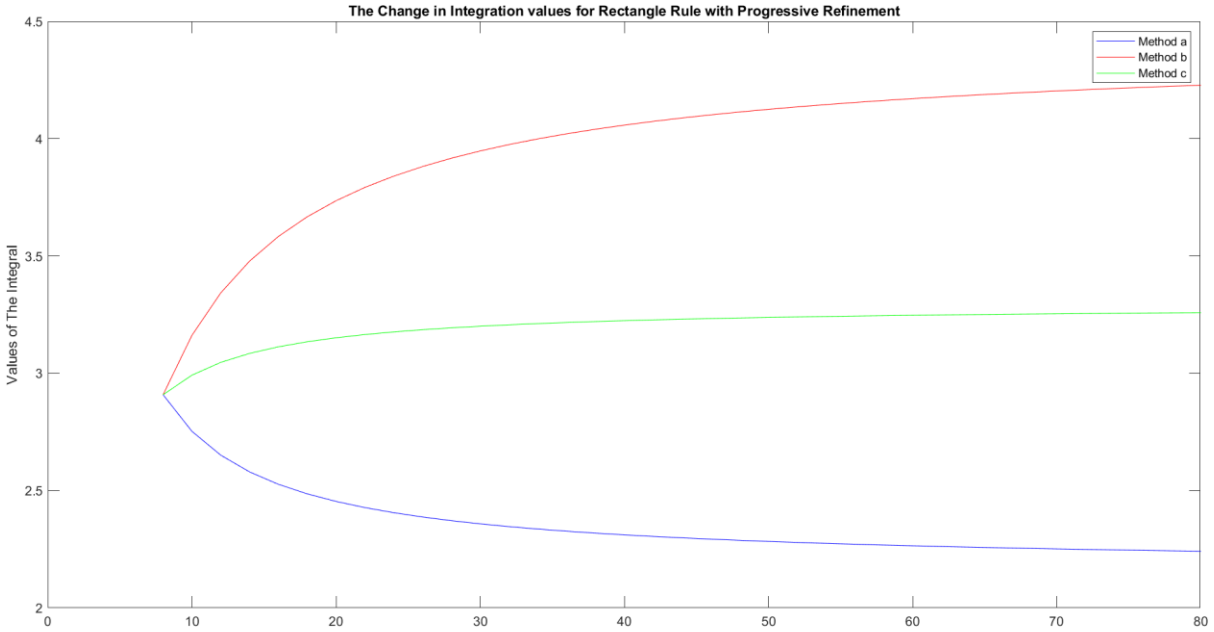
```
    y_max = input('In the y direction: ');
end

%x_min=-1; x_max=0; y_min=0; y_max=2; (Integration Limits in this question)

%Switch Statement that corresponds to the inputed desired method of
%integration
switch sel_meth

    %Finding the required values of integration using the three methods of
    %refinement for the Rectangle Method
    case 1
        x=0;
        for i = 8:2:80
            x= x+1;
            N = i;
            M = 8;
            INTEGRAL_Rec_a(x)=Rect_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);

            M = i;
            N = 8;
            INTEGRAL_Rec_b(x)=Rect_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);

            M = i;
            N = i;
            INTEGRAL_Rec_c(x)=Rect_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);
        end

        %Plotting the three lines corresponding to each type of refinement
        %on the same figure for comparison (Rectangle Method)
        figure;
        plot (8:2:80,INTEGRAL_Rec_a, "b-"); hold on;
        plot (8:2:80,INTEGRAL_Rec_b, "r-"); hold on;
        plot (8:2:80,INTEGRAL_Rec_c, "g-");
        xlabel('Number of Intervals');
        ylabel('Values of The Integral');
        legend ('Method a','Method b','Method c')
        title('The Change in Integration values for Rectangle Rule with
Progressive Refinement');
        METHOD= 'Rectangle';

    %Finding the required values of integration using the three methods of
    %refinement for the Trapizium Method
    case 2
        x=0;
        for i = 8:2:80
            x = x+1;

            N = i;
            M = 8;
            INTEGRAL_Trap_a(x)=Trap_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);

            M = i;
            N = 8;
            INTEGRAL_Trap_b(x)=Trap_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);

            M = i;
            N = i;
            INTEGRAL_Trap_c(x)=Trap_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);
```

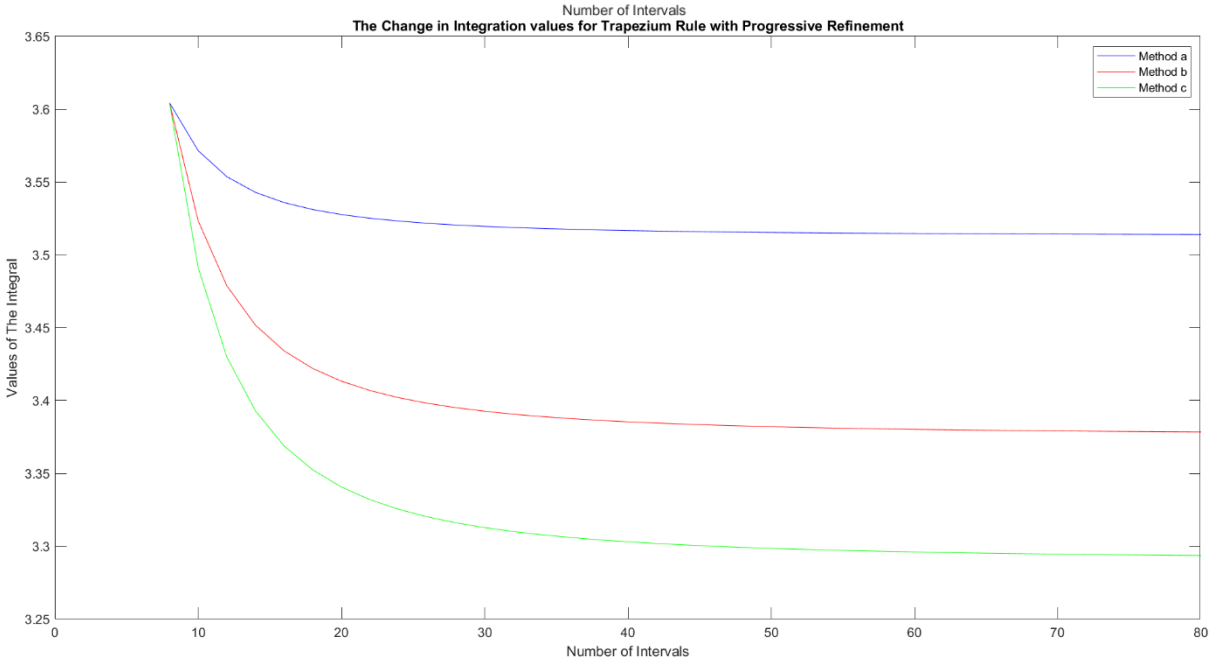
```
        end

        %Plotting the three lines corresponding to each type of refinement
        %on the same figure for comparison (Trapezium Method)
        figure;
        plot (8:2:80,INTEGRAL_Trap_a, "b-"); hold on;
        plot (8:2:80,INTEGRAL_Trap_b, "r-"); hold on;
        plot (8:2:80,INTEGRAL_Trap_c, "g-");
        xlabel('Number of Intervals');
        ylabel('Values of The Integral');
        legend ('Method a','Method b','Method c')
        title('The Change in Integration values for Trapezium Rule with
Progressive Refinement');
        METHOD = 'Trapezium';

    %Finding the required values of integration using the three methods of
    %refinement for the Simpson's Method
    case 3
        x=0;
        for i = 8:2:80
            x = x+1;

            N = i;
            M = 8;
            INTEGRAL_Simp_a(x)=simp_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);

            M = i;
            N = 8;
            INTEGRAL_Simp_b(x)=simp_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);

            M = i;
            N = i;
            INTEGRAL_Simp_c(x)=simp_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);
        end

        %Plotting the three lines corresponding to each type of refinement
        %on the same figure for comparison (Simpson's Method)
        figure;
        plot (8:2:80,INTEGRAL_Simp_a, "b-"); hold on;
        plot (8:2:80,INTEGRAL_Simp_b, "r-"); hold on;
        plot (8:2:80,INTEGRAL_Simp_c, "g-");
        xlabel('Number of Intervals');
        ylabel('Values of The Integral');
        legend ('Method a','Method b','Method c')
        title('The Change in Integration values for Simpsons Rule with Progressive
Refinement');
        METHOD = 'Simpsons';

    %Finding the value of using the Gaussian integration method
    case 4

        %Asking for inputing the Gaussian points in x direction
        N = -1;
        disp('Set Number of Gaussian points:');
        while (N<1 || uint8(N)~=N ) % so N has to be < 255 (2^8=256-> max unsigned
8-bit integer : 255)
            N = input('Please provide a positive integer ');
        end
        %Asking for inputing the Gaussian points in y direction
```

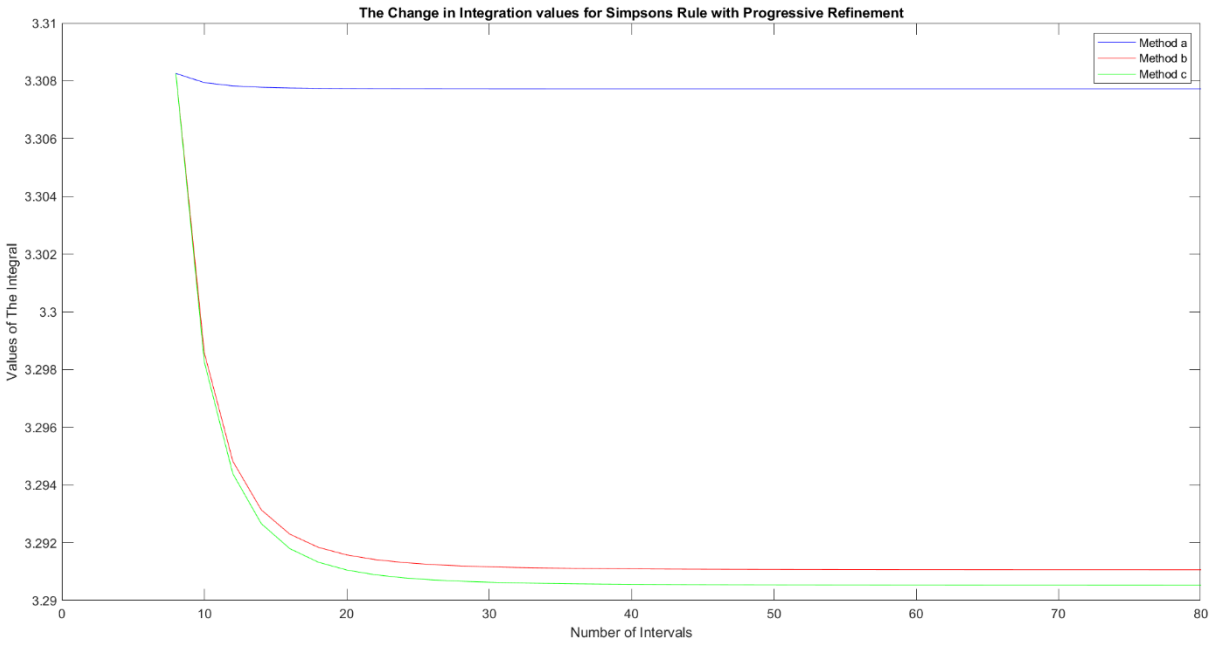
```
        M = -1;
        disp('Set Number of Gaussian points:');
        while (M<1 ||uint8(M)~=M ) % also M <255
            M = input('Please provide a positive integer ');
        end

        INTEGRAL=Gauss_2D_analytic(F,x_min,x_max,y_min,y_max,N,M);
        METHOD = 'Gauss';

end
```

The Rest of the Different Method's Functions are in the MATLAB Folder

4)







5)

The more intervals that are used, the smaller the trapezoids, rectangles or the parabolic arcs will be and the closer the approximations to the true area under the curve will be to estimate the true value of the integral.

In contrast, if fewer intervals are used, the trapezoids, rectangles and parabolic arcs will be larger and the approximations to the true integral value will be less accurate. Therefore, using a refinement strategy that divides the region of integration into more intervals will generally produce a more accurate approximation to the true integral value when using any of the Newton-Cotes rules.

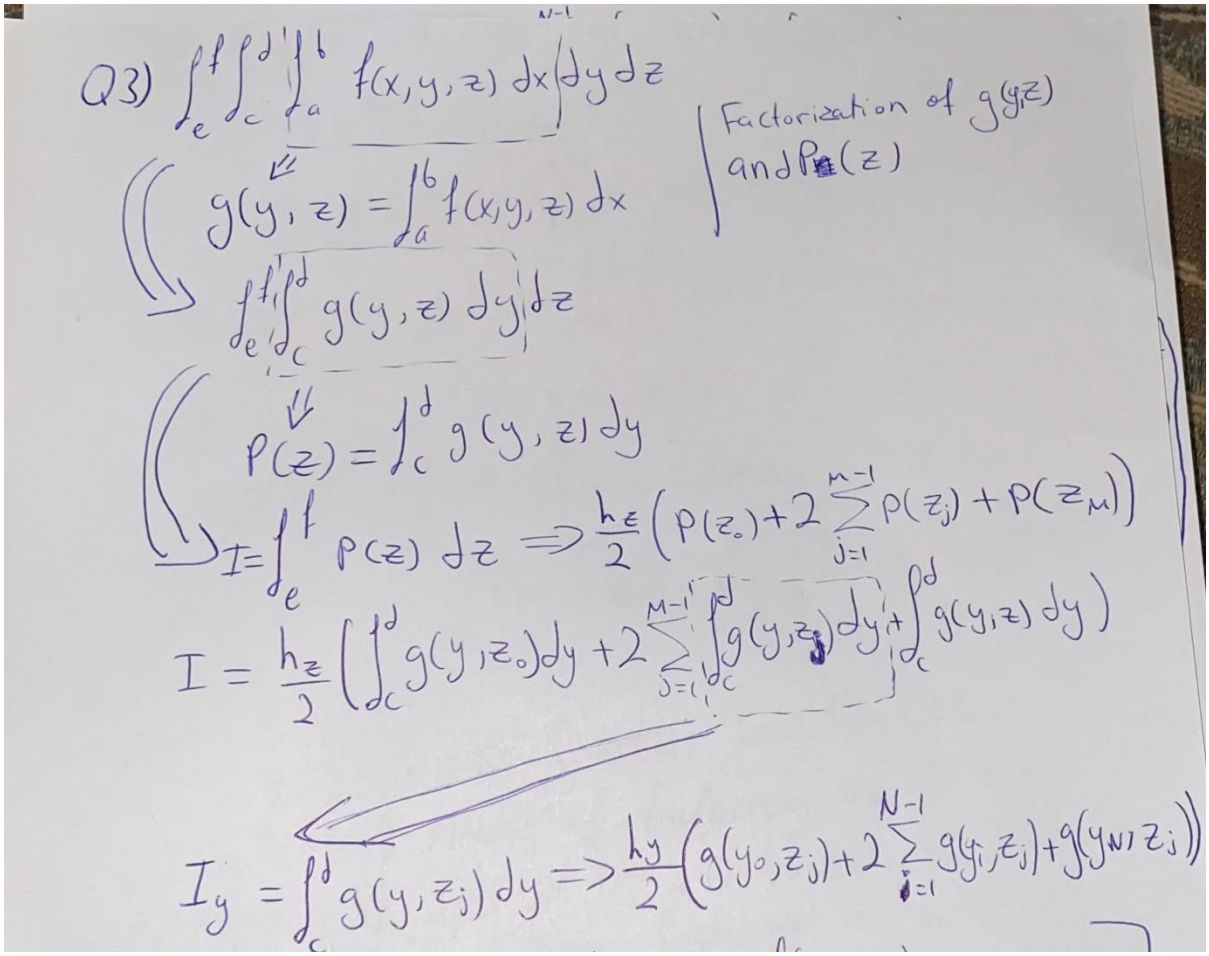
Rectangle Method Graph: As the number of intervals increases for the Rectangle, the Rectangle Method is a first-order method, meaning that the error in the approximation decreases at a rate of O(1/N), where N is the number of intervals. For functions similar to f(x,y) in this question which are more curvy, the rectangular method is not a very accurate method. Method A undershoots the real value of the integral, Method B overshoots the real value, Method C is the most accurate among the three, and in the Rectangle Method, it takes around 30 intervals to stabilize.

Trapezium Method Graph: As the number of intervals increases for the trapezoid, the trapezoidal rule is a first-order method, meaning that the error in the approximation decreases at a rate of O(1/N), where N is the number of intervals. In the context of this question, implementing the incrementally increasing array of values in the x-direction in Method A which overshoots by a lot more than in the y-direction in Method B. Method C is the most accurate and among the three, and it takes between 40-50 intervals to stabilize.

Simpson’s Method Graph: As the number of intervals increase for the Simpson’s Method the error in the approximation decreases at a rate of O(1/N^2), where N is the number of intervals, this is because Simpson's rule is a second-order method. Method A is way off from true value comparing to Method B and C. However, Method C reaches the true value faster than Method B, and stabilizes in between 20 and 30 intervals.

Concluding, the Simpson's Method seems to converge slightly faster than the other two Methods, which proposes that it could give the closest value to the true value of the integral with the least amount of intervals. However, the Simpson's method is known to be the most computationally expensive among the three, so depending on what factors are expected to be used, the answer could change slightly.

Q3)



$$I = \frac{h_z}{2}\left[\begin{array}{l} \frac{h_y}{2}\left(g(y_0,z_0) + 2\sum_{i=1}^{N-1} g(y_i,z_0) + g(y_N,z_0) + \right. \\ 2\sum_{j=1}^{M-1}\left(\frac{h_y}{2}\left(g(y_0,z_j) + 2\sum_{i=1}^{N-1} g(y_i,z_j) + g(y_N,z_j)\right)\right) + \\ \left.\frac{h_y}{2}\left(g(y_0,z_M) + 2\sum_{i=1}^{N-1} g(y_i,z_M) + g(y_N,z_M)\right)\right. \end{array}\right]$$

Substituting $g(y,z) = \int_a^b f(x,y,z)\,dx$

$$I = \frac{h_z}{2}\left[\begin{array}{l} \frac{h_y}{2}\left(\int_a^b f(x,y_0,z_0) + 2\sum_{i=1}^{N-1}\int_a^b f(x,y_i,z_0) + \int_a^b f(x,y_N,z_0) + \right. \\ 2\sum_{j=1}^{M-1}\left(\frac{h_y}{2}\left(\int_a^b f(x,y_0,z_j) + 2\sum_{i=1}^{N-1}\int_a^b f(x,y_i,z_j) + \int_a^b f(x,y_N,z_j)\right.\right. \\ \left.\left.+\frac{h_y}{2}\left(\int_a^b f(x,y_0,z_M) + 2\sum_{i=1}^{N-1}\int_a^b f(x,y_i,z_M) + \int_a^b f(x,y_N,z_M)\right.\right.\right. \end{array}\right]$$

- Factorising $g(y,z)$ & $p(z)$ from $f(x,y,z)$, as the same process of factorising the 1D to 2D relations to be applied to 2D to 3D. the difference is just taking an extra factorisation to get a single integral and apply the matrix weighting factors then. Finally to derive the full formula, the tensor product should be taken and have a 6x6 matrix that has different weights on the corners, edges and in the center of the 3d shape.