```
% Partie Labo : C-F Mateo: 20266263



N = 3;

% Paramètres pour l'intégration
a = 0;
b = 1;
f = @(x) ((x - 1/2).^2) .* exp(2 * x - 1); % Fonction intégrée

% Valeur exacte de l'intégrale:
%valeur_exacte = -0.625 * exp(-1) + 0.125 * exp(1);
valeur_exacte =  (exp(1)/8) - (5/(8*exp(1)));


% Initialisation de la précision
precision = 4 * eps;
erreur_relative = inf;

% Tableau pour stocker les erreurs relatives pour chaque N
erreurs_relatives = [];

% Boucle pour trouver le nombre minimal de nœuds:
while erreur_relative > precision

    % Étape 1 : Initialisation des nœuds pour le nombre actuel de N
    x_0 = zeros(1, N+1);
    for i = 0:N
        x_0(i+1) = (1 - (1 / (8 * (N + 1)^2)) + (1 / (8 * (N + 1)^3))) * cos(((4 *
i + 3) / (4 * N + 6)) * pi);
    end
    x_k = x_0;  % Copie des nœuds initiaux pour itérer

    % Boucle pour les itérations de Newton
    diff_max = inf;
    while diff_max >= eps
        P = zeros(N+2, N+1);
        P_prime = zeros(N+2, N+1);

        % Initialiser les polynômes de base
        P(1, :) = ones(1, N+1);
        P(2, :) = x_k;

        for m = 2:N+1
            % Calcul des polynômes avec la récurrence de Bonnet
            P(m+1, :) = ((2*(m-1) + 1) * x_k .* P(m, :) - (m-1) * P(m-1, :)) / m;
            % Calcul des dérivées des polynômes
```

```
            P_prime(m+1, :) = ((m) * P(m, :) - (m) * x_k .* P(m+1, :)) ./ (1 -
x_k.^2);
        end

        % Mise à jour des nœuds avec Newton
        P_N1_values = P(N+2, :);
        P_N1_prime_values = P_prime(N+2, :);
        x_k1 = x_k - P_N1_values ./ P_N1_prime_values;
        diff_max = max(abs(x_k1 - x_k));
        x_k = x_k1;
    end

    % Calcul des poids ωi:
    omega = zeros(1, N+1);
    for i = 1:N+1
        P_N1_prime = P_prime(N+2, i);
        omega(i) = 2 / ((1 - x_k(i)^2) * P_N1_prime^2);
    end

    % Changement de variable pour évaluer l'intégrale sur [-1,1] avec notre
    % nouvelle fonction g defini telle que:
    g = @(x) f((b - a) / 2 * x + (a + b) / 2);

    % Calcul de l'intégrale approximée avec les poids et nœuds via la
    % quadrature de Gauss:
    integrale_approx = (b - a) / 2 * sum(omega .* g(x_k));


    % Calcul de l'erreur relative
    erreur_relative = abs(integrale_approx - valeur_exacte) / abs(valeur_exacte);

    erreurs_relatives = [erreurs_relatives; N, erreur_relative];

    % Incrément du nombre de nœuds si la précision n'est pas atteinte
    N = N + 1;
end


N_star_plus_1 = N;

% (a) Affichage du nombre minimum de nœuds
fprintf('Nombre de nœuds minimum pour atteindre la précision (N* + 1) : %d\n',
N_star_plus_1); % On a N* + 1 = 5 et donc N* = 4
```

```
Nombre de nœuds minimum pour atteindre la précision (N* + 1) : 5
```

```
% (b) Affichage des erreurs relatives pour chaque N de 3 à N*
fprintf('Erreur relative pour chaque nombre de nœuds (N, Erreur relative) :\n');
```

```
Erreur relative pour chaque nombre de nœuds (N, Erreur relative) :
```

```
disp(erreurs_relatives);
```

```
     3.000000000000000e+00     2.033561797536314e-05
     4.000000000000000e+00     2.361801263239659e-06
```

```
% (c) Affichage des nœuds et des poids pour N* dans un tableau au format long
N_final = N_star_plus_1 - 1;
x_final = x_k;
omega_final = omega;

% Création d'un tableau pour afficher les nœuds et les poids de 0 jusqu'a
% N*
resultats = table((0:N_final)', x_final', omega_final',...
    'VariableNames', {'Indice', 'Noeud_xi', 'Poids_omega'});

% Format long pour plus de précision
format long e
disp('Tableau des nœuds et poids pour N* :');
```

```
Tableau des nœuds et poids pour N* :
```

```
disp(resultats);
```

Indice	Noeud_xi	Poids_omega
0.00000000000000e+00	9.06179845938821e-01	2.36926038736404e-01
1.00000000000000e+00	5.38469310105683e-01	4.78628670411301e-01
2.00000000000000e+00	0.00000000000000e+00	5.68888888888889e-01
3.00000000000000e+00	-5.38469310105683e-01	4.78628670411301e-01
4.00000000000000e+00	-9.06179845938821e-01	2.36926038736404e-01