```
% a)

function [l] = HermiteMoments (N, V0, X0, ...
    kappa, sigma, theta, r, rho, T, v_min, v_max, mu_w, sigma_w)
% Computes first N Hermite moments via moment formula

% N: first Hermite moments to compute
% V0,X0: parameters of the basis vector B for the first Hermite moment
% kappa, sigma, theta, r, rho: parameters of the Jacobi model
% v_min, v_max: parameters of the quadratic form
% mu_w, sigma_w: mean ans standard deviation of the w(x) gaussian
density
% l : first N Hermite moments using the moment formula


% from (m.n) to the position of the corresponding basis element in Hn
Ind = @(m,n) (m+n+1)*(m+n)/2 + (n+1);

% Polynomial basis evaluation in initial points
M = 1/2 * (N+1) * (N+2);        % given in the problem
h = zeros(1,M);                 % vector initialization

% m,n >= 0 and m+n <= N
for m = 0:N
    for n = 0:N-m

        % probabilistic standard Hermite polynomial
        H = @(n,x) 2^(-0.5*n) * hermiteH(n,x/sqrt(2));

        % polynomial evaluation: v^m * H_n(x)
        BN(Ind(m,n)) = V0^m * H(n,(X0 - mu_w) / sigma_w)/sqrt
(factorial(n));

    end
end


% Matrix of the generator of the Jacobi model

G = zeros(M,M);                         % matrix initialization
D = (sqrt(v_max)-sqrt(v_min))^(-2);     % useful

% m,n >= 0 and m+n <= N
for m = 0:N
    for n = 0:N-m

        ColInd = Ind(m,n);              % position

        if m > 1
```

```
            G(Ind(m-2,n),ColInd) = ...
                - 0.5 * sigma^2 * m * (m-1) * v_max * v_min * D;
        end
        if m > 0 && n > 0
            G(Ind(m-1,n-1),ColInd) = ...
                - sigma * rho * m * sqrt(n) * v_max * v_min * D /↙
sigma_w;
        end
        if m > 0
            G(Ind(m-1,n),ColInd) = kappa * theta * m + ...
                0.5 * sigma^2 * m * (m-1) * (v_max+v_min) * D;
        end
        if n > 0
            G(Ind(m,n-1),ColInd) = r * sqrt(n) / sigma_w + ...
                sigma * rho * m * sqrt(n) * (v_max+v_min) * D /↙
sigma_w;
            G(Ind(m+1,n-1),ColInd) = - 0.5 * sqrt(n) / sigma_w - ...
                sigma * rho * m * sqrt(n) * D / sigma_w;
        end
        if n > 1
            G(Ind(m+1,n-2),ColInd) = 0.5 * sqrt(n * (n-1)) /↙
sigma_w^2;
        end

        % for each m and n:
        G(ColInd,ColInd) = - kappa * m - 0.5 * sigma^2 * m * (m-1) *↙
D;

    end

end

L = BN * expm(T * G);           % Hermite moments matrix ( moment↙
formula )
l = zeros(N+1,1);               % initialization moment vector

for n = 0:N
    % e_{phi(0.n)} is translated in taking only Ind(0,n)
    l(n+1) = L( Ind(0,n) );
end

end
```