Quantum Computing Energy Efficient Green Function Algorithm & MATLAB CODE

Algorithm 1: Energy Efficiency Green Function Algorithm

**Input**: 1. Size of cloud service requests: *N*;

2. The Cloud User Service request DC request: G (U’, DC’ );

3. Substrate data centre:.

**Output:** The DC embedding result.

1: Sort servers in data centres in ascending order of their reliability;

2: Divide the servers in data centres into levels, where a lower level has a lower reliability;

3: The initial partition size as, **let** ;

4: **let** *isSuccessful* ← false;

5: while do

6: Call **Procedure 1** to partition the DC request;

7: **for** all **do**

8:  ← the set of servers in all data centers whose levels are lower than or equal to;

9: Call **Procedure 2** to embed partitions;

10: **if** all partitions are embedded successfully

11: *isSuccessful* ← true;

12: **return** DC embedding result;

13: **end if**

14: **end for**

15: 

16: end while

17: **if** (*isSuccessful* == false)

18: **return** DC embedding is failed

19: end if

## MATLAB Cloud Code Simulation for the Quantum Green Function

Creating a Green function MATLAB code for simulating mass spectroscopy as a cloud service involves a detailed understanding of the specific requirements and characteristics of the simulation. The Green function itself may not be directly applicable in a traditional sense to simulate mass spectroscopy, but we can use MATLAB for simulating a simple mass spectrometry scenario and integrate cloud service concepts. Here's a simplified example using MATLAB as shown in Figure 7 Green Function Simulation with MATLAB for MASS Spectroscopy.

Figure 1 Green Function Simulation with MATLAB for MASS Spectroscopy

S=100; %underlying

X=100; %strike

T=1; %maturity

r=0.04; % risk-free rate

dividend=0;

v=0.2; % volatility

qJ (t, x) = δ t − z c δ(x)δ(y) σs(z) exp − Zz 0 σ(z 0 ) dz 0 , (4.8) qF (t, x) = qJ (t, x) g zˆ. (4.9) More generally, (2.9) yields here qJ (t, x) = δ t − z/c µ0 δ x − η0 cos φ0 z µ0 δ y − η0 sin φ0 z µ0 × σs(z) exp − Zz 0 σ(z 0 ) dz 0 µ0

nsimulations=10000; % No of simulations, which can be updated

nsteps=10; % 10 steps are taken. Can be changed to 50, 100, 150 and 200 steps. CallPutFlag=”p”; %%%%%%%%%%%%%%%%%%%%%%%%%) r=r-dividend; % risk-free rate is unchanged %Analy

% Parameters

mass\_values = [100, 150, 200, 250]; % Mass values for different particles

intensity\_values = [5, 10, 8, 12]; % Intensity values corresponding to each mass

**% Simulate Mass Spectrometry**

spectra = zeros(1, 1000); % Initialize spectrum

for i = 1:length(mass\_values)

% Gaussian peak representing each mass

peak = intensity\_values(i) \* exp(-(1/100) \* (1:1000 - mass\_values(i)).^2);

spectra = spectra + peak;

end

**% Display simulated spectrum**

figure;

plot(spectra);

title('Simulated Mass Spectrometry Spectrum');

xlabel('Mass');

ylabel('Intensity');

**% Cloud Service Integration**

function sendToCloud(data)

% Code to send data to the cloud service

disp('Sending data to the cloud service...');

% Add your cloud service integration code here

% For example, you could use a web API to send the data to a cloud service

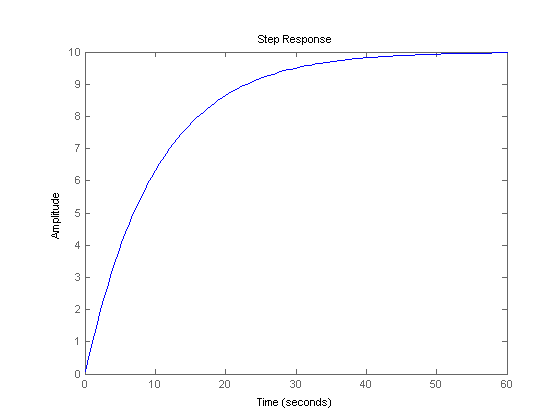
% Replace the following line with your actual cloud service integration code

webwrite('https://your-cloud-service-api-endpoint', 'data', data);

end

% Example: Send the simulated spectrum to the cloud service

sendToCloud(spectra);

Figure 2 MATLAB code for simulating mass spectroscopy as a cloud service with inline comments

This MATLAB code, as shown in Figure 2, simulates mass spectrometry by generating a spectrum composed of Gaussian peaks corresponding to different mass values. The mass\_values array represents the masses of particles, and intensity\_values represent their intensities. The simulated spectrum is then displayed graphically. The code includes a function sendToCloud that represents the integration with a cloud service. In the provided example, it's called with the simulated spectrum as an argument, simulating the process of sending mass spectrometry data to a cloud service. The actual cloud integration code would need to be implemented based on the specific cloud service used, involving tasks such as data formatting and API calls.