```
% Constants and parameters
h = 6.626e-34; % Planck's constant
hbar = h/(2 * pi); % Reduced Planck's constant
tau = 1.4e-6; % Interaction time (constant)
t1 = 2e-9; % Duration of SWAP operator S1
t2 = 2e-9; % Duration of SWAP operator S2
J1_pi = 250e6; % Ideal J1 for perfect pi-pulse
J3_pi = 250e6; % Ideal J3 for perfect pi-pulse
Q = 21; % Exchange oscillation quality factor

sigma_Bz = 18e6; % Standard deviation for hyperfine noise
Bz_mean = [0, 20e6, 0, 50e6] + 3.075e9; % Mean hyperfine fields
num_realizations =128; % Number of Monte Carlo realizations
B_uniform = 3.075e9; % Uniform magnetic field gradient (Hz)

% Readout error parameters
tm_ST1 = 4e-6;
tm_ST2 = 6e-6;
T1_ST1 = 60e-6;
T1_ST2 = 50e-6;
fm_ST1 = 0.99;
fm_ST2 = 0.95;

% Compute relaxation probabilities (r) for each ST qubit
r_ST1 = 1 - exp(-tm_ST1 / T1_ST1);
r_ST2 = 1 - exp(-tm_ST2 / T1_ST2);

% Compute misidentification probabilities (q) for each ST qubit
q_ST1 = 1 - fm_ST1;
q_ST2 = 1 - fm_ST2;

% Sweeping parameter
epsilon_values = linspace(-0.6, 0.6, 175); % Sweep range for fractional errors
J2_values = linspace(0, 5, 175) * 1e6; % J2 range (Hz)

% Initial state setup
g = [0; 1; 0; 0]; % |g> = |↑↓>
e = [0; 0; 1; 0]; % |e> = |↓↑>
T_plus = [1; 0; 0; 0]; %
T_minus = [0; 0; 0; 1]; %
sigma_x=[0,1;1,0];
sigma_y=[0,1j;-1j,0];
sigma_z=[1,0;0,-1];
% Coefficients for initial state
fg = 0.9;
fg2 = 0.95;
sa1 = sqrt(fg); % Coefficient for |g>
sa2 = sqrt((1 - fg) / 3); % Coefficient for |e>
sa3 = sa2; % Coefficient for |T+>
sa4 = sa2; % Coefficient for |T->
```

```
sb1 = sqrt(fg2); % Coefficient for |g>
sb2 = sqrt((1 - fg2) / 3); % Coefficient for |e>
sb3 = sb2; % Coefficient for |T+>
sb4 = sb2; % Coefficient for |T->

% Initial state for a single ST qubit
psi_i_ST1 = sa1 * g + sa2 * e + sa3 * T_plus + sa4 * T_minus; % ST qubit 1
psi_i_ST2 = sb1 * g + sb2 * e + sb3 * T_plus + sb4 * T_minus; % ST qubit 2
psi_i = kron(psi_i_ST1, psi_i_ST2); % Two-qubit system initial state

% Projector of ST-1 state measurement operator
P_g = kron(g * g', eye(4));
I4 = eye(4);
I2 = eye(2);
sz1 = kron(kron(kron(sigma_z, I2), I2), I2);
sz2 = kron(kron(kron(I2, sigma_z), I2), I2);
sz3 = kron(kron(kron(I2, I2), sigma_z), I2);
sz4 = kron(kron(kron(I2, I2), I2), sigma_z);

sx1 = kron(kron(kron(sigma_x, I2), I2), I2);
sx2 = kron(kron(kron(I2, sigma_x), I2), I2);
sx3 = kron(kron(kron(I2, I2), sigma_x), I2);
sx4 = kron(kron(kron(I2, I2), I2), sigma_x);

sy1 = kron(kron(kron(sigma_y, I2), I2), I2);
sy2 = kron(kron(kron(I2,sigma_y),I2),I2);
sy3 = kron(kron(kron(I2,I2),sigma_y),I2);
sy4 = kron(kron(kron(I2, I2), I2), sigma_y);
I4 = eye(4);
I2 = eye(2);
% Interaction between qubits
H_12 = interaction_H12(sz1,sz2,sy1,sy2,sx1,sx2);
H_23 = interaction_H23(sz2,sz3,sy2,sy3,sx2,sx3);
H_34 = interaction_H34(sz3,sz4,sy3,sy4,sx3,sx4);



return_prob_avg = zeros(length(epsilon_values), length(J2_values));

% Monte Carlo simulation
for e = 1:length(epsilon_values)
    epsilon = epsilon_values(e); % Current fractional error

    for j = 1:length(J2_values)
        J2_nominal = J2_values(j); % Interaction strength

        return_prob_realizations = zeros(1, num_realizations); % Store realizations

        for realization = 1:num_realizations
```

```
            % Randomize J1 and J3 with errors
            sigma_J1 = J1_pi / (sqrt(2) * pi * Q); % Standard deviation for J1
            sigma_J2 = J2_nominal / (sqrt(2) * pi * Q); % Standard deviation for J2
            sigma_J3 = J3_pi / (sqrt(2) * pi * Q); % Standard deviation for J3



            J1sd = normrnd(J1_pi, sigma_J1) * (1 + epsilon); % J1 with π-pulse errors
            J2sd = normrnd(J2_nominal, sigma_J2);
            J3sd = normrnd(J3_pi, sigma_J3) * (1 + epsilon); % J3 with π-pulse errors


            % Hyperfine fields
            Bz = normrnd(Bz_mean, sigma_Bz); % Random Bz values
            H_B = Bz(1)*sz1 + Bz(2)*sz2 + Bz(3) * sz3 + Bz(4) *sz4;
            H_B = h / 2 * H_B;



            % Hamiltonians and operators
            H_S1 = h / 4 * J1sd * (H_12) + H_B;
            H_S2 = h / 4 * J3sd * (H_34) + H_B;

            S1 = expm(-1i * H_S1 * t1 / hbar);
            S2 = expm(-1i * H_S2 * t2 / hbar);

            % Interaction Hamiltonian
            H_int = h / 4 * J2_nominal * H_23 + H_B;
            S_int = expm(-1i * H_int * tau / hbar);

            % Full evolution operator
            U = S_int * S2 * S1;

            % Time evolution
            psi_t = U^4 * psi_i;

            % Measurement
            P_g_return = abs(psi_t' * P_g * psi_t);
            % probability of excited state relaxing to ground state during measurements
            P_g_final_ST1 = (1 - r_ST1 - 2 * q_ST1) * P_g_return + r_ST1 + q_ST1;
            P_g_final_ST2 = (1 - r_ST2 - 2 * q_ST2) * P_g_return + r_ST2 + q_ST2;
            return_prob_realizations(realization) = P_g_final_ST1;
        end

        % Average over realizations
        return_prob_avg(e, j) = mean(return_prob_realizations);
    end
end

% Plotting the results
```

```
imagesc(J2_values / 1e6, epsilon_values, return_prob_avg); % J2 in MHz
set(gca, 'YDir', 'normal');
xlabel('J_2 (MHz)');
ylabel('Fractional Error (\epsilon)');
title('Return Probability vs. \pi-Pulse Error and J_2');
colorbar;
caxis([0.4, 0.9]);
%   Define dot product of pauli operator
function H_12 = interaction_H12(sz1,sz2,sy1,sy2,sx1,sx2)
    H_12 = sz1*sz2+sy1*sy2+sx1*sx2;
end

function H_23 = interaction_H23(sz2,sz3,sy2,sy3,sx2,sx3)
    H_23 = sz2*sz3+sy2*sy3+sx2*sx3;
end

function H_34 = interaction_H34(sz3,sz4,sy3,sy4,sx3,sx4)
    H_34 = sz3*sz4+sy3*sy4+sx3*sx4;
end
```