```
function [P, Q, V, delta, J11] = solveForPowerFlow(PSpecified, QSpecified, ...
    V, delta, ...
    ybus, BMatrix, E, nPQ, nPV, ...
    listOfPQBuses, listOfNonSlackBuses, ...
    numIterations, toleranceLimit, powerFlowMethod, ...
    displayTables, printJacobians, printMismatches, ...
    printPowerFlowConvergenceMessages)

    N = size(ybus, 1);
    if strcmp(powerFlowMethod, 'NRPF')
        for itr = 1:numIterations
            desiredOutput = 'both';
            [PMismatch, QMismatch, P, Q] = ...
                computeMismatches(PSpecified, QSpecified, ...
                V, delta, ybus, BMatrix, E, ...
                listOfPQBuses, listOfNonSlackBuses, powerFlowMethod);

            mismatch = [PMismatch; QMismatch];
            if displayTables && printMismatches
                fprintf("Iteration Number %i Mismatches:\n", itr);
                disp(mismatch)
            end
            [J, JTable] = constructJacobian(P, Q, ...
                V, delta, N, ybus, BMatrix, E, ...
                nPQ, nPV, listOfPQBuses, listOfNonSlackBuses, powerFlowMethod,
desiredOutput);

            if displayTables && printJacobians
                fprintf("Iteration Number %i Jacobian:\n", itr);
                JTable %#ok<NOPRT>
            end

            correction = solveUsingLU(J, mismatch, 2*nPQ+nPV);

            delta = [delta(1); delta(listOfNonSlackBuses) + correction(1:nPQ+nPV)];
            V(listOfPQBuses) = V(listOfPQBuses).*( ones(nPQ, 1) +
correction(nPQ+nPV+1:end) );

            if mean(abs(correction)) < toleranceLimit
                fprintf("Convergence using %s achieved in %i iterations.\n",
powerFlowMethod, itr);
                break;
            else
                if printPowerFlowConvergenceMessages
                    fprintf("Convergence still not achieved in %i iterations as %f
is greater than %f\n", itr, mean(abs(correction)), toleranceLimit);
                end
            end
        end
```

```
    elseif strcmp(powerFlowMethod, 'Decoupled NRPF') || strcmp(powerFlowMethod,
'Fast Decoupled NRPF')
        for itr = 1:numIterations
            desiredOutput = 'P';
            [PMismatch, ~, P, Q] = ...
                computeMismatches(PSpecified, QSpecified, ...
                V, delta, ybus, BMatrix, E, ...
                listOfPQBuses, listOfNonSlackBuses, powerFlowMethod);

            if displayTables && printMismatches
                fprintf("Iteration Number %i P Mismatches:\n", itr);
                disp(PMismatch)
            end
            [J11, J11Table] = constructJacobian(P, Q, ...
                V, delta, N, ybus, BMatrix, E, ...
                nPQ, nPV, listOfPQBuses, listOfNonSlackBuses, powerFlowMethod,
desiredOutput);

            if displayTables && printJacobians
                fprintf("Iteration Number %i Jacobian J11:\n", itr);
                J11Table %#ok<NOPRT>
            end

            correctionDelta = solveUsingLU(J11, PMismatch, nPQ+nPV);

            delta = [delta(1); delta(listOfNonSlackBuses) + correctionDelta];

            desiredOutput = 'Q';
            [~, QMismatch, P, Q] = ...
                computeMismatches(PSpecified, QSpecified, ...
                V, delta, ybus, BMatrix, E, ...
                listOfPQBuses, listOfNonSlackBuses, powerFlowMethod);

            if displayTables && printMismatches
                fprintf("Iteration Number %i Q Mismatches:\n", itr);
                disp(QMismatch)
            end
            [J22, J22Table] = constructJacobian(P, Q, ...
                V, delta, N, ybus, BMatrix, E, ...
                nPQ, nPV, listOfPQBuses, listOfNonSlackBuses, powerFlowMethod,
desiredOutput);

            if displayTables && printJacobians
                fprintf("Iteration Number %i Jacobians:\n", itr);
                J22Table %#ok<NOPRT>
            end

            correctionDeltaVByV = solveUsingLU(J22, QMismatch, nPQ);
```

```
            V(listOfPQBuses) = V(listOfPQBuses).*( ones(nPQ, 1) +
correctionDeltaVByV );

            correction = [correctionDelta; correctionDeltaVByV];
            if mean(abs(correction)) < toleranceLimit
                fprintf("Convergence using %s achieved in %i iterations.\n",
powerFlowMethod, itr);
                break;
            else
                if printPowerFlowConvergenceMessages
                    fprintf("Convergence still not achieved in %i iterations as %f
is greater than %f\n", itr, mean(abs(correction)), toleranceLimit);
                end
            end
        end
    end
end
```