Simulating the Calcination Process for Boron and Nitrogen Co-Doped Carbon in Lithium-Ion Battery Cathode Materials $LiFePO_4$

(Project CSCI 596)

Ning Nie

1. Introduction

Lithium-ion batteries (LIBs) as a cornerstone technology in modern energy storage systems, powering everything from portable electronics to electric vehicles (EVs) and renewable energy storage solutions. The heart of these batteries lies in their cathode materials, which play a pivotal role in defining the battery's overall performance, energy density, and longevity. However, the efficiency and performance of LIBs are significantly constrained by the electrochemical properties of traditional cathode materials. These materials often exhibit low electronic conductivity and poor lithium-ion (Li^+) diffusion, which are major bottlenecks in achieving high-performance batteries. This limitation is particularly pronounced in the case of lithium iron phosphate ($LiFePO_4$), a widely used cathode material known for its stability and safety but criticized for its intrinsic low electronic conductivity.

In recent years, significant advancements have been made in enhancing the properties of $LiFePO_4$ cathodes. Researchers have explored various strategies, such as nano structuring, surface coating, and doping with foreign atoms, to improve its electrical and ionic conductivity. Among these, doping has shown considerable promise. Doping $LiFePO_4$ with elements like carbon, nitrogen, and sulfur has resulted in improved electronic conductivity and battery performance.

2. Current state of the previous work

My previous research, titled "Boron and Nitrogen Co-doped Carbon Layers of LiFePO4 Improve the High-Rate Electrochemical Performance for Lithium-Ion Batteries," represents a significant advancement in the field of lithium-ion battery technology. This study focused on enhancing the electrochemical performance of lithium iron phosphate ($LiFePO_4$) cathodes through an innovative co-doping approach. By integrating nitrogen (N) and boron (B) into the carbon layers of $LiFePO_4$, we achieved notable improvements in the material's electrochemical characteristics.

The research is based mainly on the chemical synthesis process and several characterizations, involved a comprehensive process of material preparation, including hydrothermal synthesis followed by a high-temperature calcination process. Structural and morphological characterizations were conducted using advanced techniques like X-ray diffraction (XRD), high-resolution transmission electron microscopy (HR-TEM), and scanning electron microscopy (SEM) mapping. These analyses confirmed the successful incorporation of N and B dopants into the carbon layers without compromising the structural integrity of $LiFePO_4$.

Electrochemical measurements revealed that both single-element doping (N or B) and combined N+B co-doping significantly elevated the capacity of LiFePO4 at high current rates. Notably, the N+B co-doped $LiFePO_4$ exhibited a synergistic effect, leading to a marked enhancement in electrochemical performance compared to single-element doping. This was evidenced by an increase in discharge capacity from 101.1 mAh g^{-1} to 121.6 mAh g^{-1} at a rate of 20 C for the co-doped sample, compared to the undoped $LiFePO_4$/C. Additionally, the co-doped product based on commercial $LiFePO_4$/C showed a discharge capacity of 78.4 mAhg^{-1}, a significant improvement from the 48.1 mAh g^{-1} observed in the undoped counterpart.

3. Objective

There is a growing recognition of the need for computational simulations in the field of simulating materials preparation. Simulations can play a crucial role in understanding the fundamental mechanisms at the atomic and molecular levels, which often remain elusive in experimental studies. Moreover, computational simulations offer a cost-effective and time-efficient alternative to experimental trial-and-error. They can predict the outcomes of doping, suggest optimal compositions, and even explore new material systems that have not yet been synthesized. This approach is particularly valuable in the context of co-doping strategies, where the interplay between different dopants can lead to a wide range of possible outcomes.

Here in this project, I will focus on bridging the gap: understanding how the nitrogen and boron atoms perform and how their state change during the high temperature calcination process in this doping strategy, the initial research is about the material preparation towards improving the electrochemical performance on the cathode material LiFePO4. See the previous publications on ACS website: (https://pubs.acs.org/doi/10.1021/acsami.5b05398). Given the complexity of the doping process, especially in the context of high-temperature calcination and the insertion of dopants like nitrogen and boron into the carbon layers, there is a critical need for detailed simulations. These simulations can provide insights into how these dopants integrate into the LiFePO4 structure and influence its electrochemical properties.

4. Techniques Proposals

A. Density Functional Theory (DFT) for Electronic Structure Analysis

Density Functional Theory (DFT) is a quantum mechanical method used to investigate the electronic structure of many-body systems. In the context of the co-doped process simulation research, DFT is instrumental in understanding how nitrogen and boron co-doping affects the electronic properties of carbon materials in $LiFePO_4$ cathodes.

To investigate the electronic properties of nitrogen and boron co-doped carbon materials, we implement the algorithm Kohn-Sham equations to solve the electronic density and energy of the system. This involves setting up a supercell model of the doped carbon material and calculating the electronic band structure and density of states. As part of the implementation, we consider utilize software packages like Quantum ESPRESSO or VASP for DFT calculations. These packages allow for the implementation of pseudopotentials and plane-wave basis sets, essential for studying complex materials. Some formula in exchange-correlation functionals, such as the Generalized Gradient Approximation (GGA) or the Perdew-Burke-Ernzerhof (PBE) functional, can be used to accurately model electron interactions.

Algorithm design:

1. Setup supercell model: Initializes the atomic structure of the LFP material, including the positions and types of atoms (carbon, boron, nitrogen, and possibly lithium iron phosphate ($LiFePO_4$) atoms). This model is then broadcasted to all nodes for parallel processing.

2. Prepare DFT input: Sets up the computational parameters for DFT calculations, including pseudopotentials, basis sets, and exchange-correlation functionals. This step is crucial for accurate simulations and can be parallelized for efficiency.

3. Run DFT calculation: Performs the actual DFT calculations, distributing different parts of the calculation (like k-point grid calculations) across multiple nodes. Within each node, OpenMP is used to further parallelize the process.

4. Analyze output: After the DFT calculations, the results are gathered and analyzed. This can include various data types depending on the specific goals of the simulation.

High-Level Pseudocode for DFT Simulation with MPI and OpenMP:

```
#include <mpi.h>
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

// Constants and global variables
const int numAtoms = 10000; // Number of atoms in the supercell
double latticeParameters[3]; // Lattice parameters for the supercell
double atomicPositions[numAtoms][3]; // Positions of atoms

// Function to set up the supercell model
void setupSupercellModel() {
    // Define lattice parameters, atomic positions, etc.
    // setup might be executed by the master node and then distributed to
worker nodes
    if (world_rank == 0) {
        // Initialize lattice parameters and atomic positions
        // latticeParameters = {a, b, c};
        // atomicPositions = {...}; here will add boron, nitrogen, carbon
atoms mainly
    }
    // Distribute the model data to all nodes
    MPI_Bcast(latticeParameters, 3, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    MPI_Bcast(atomicPositions, numAtoms * 3, MPI_DOUBLE, 0,
MPI_COMM_WORLD);
}

// Function to prepare DFT calculation input
void prepareDFTInput() {
    // Set up pseudopotentials, basis set, exchange-correlation
functional, etc.
    // Could be parallelized if different parts of the system are being
prepared by different processes
    // May define pseudopotentials for each element, according to previous
XPS characterization peak data
    // Parallelize the setup for large systems
#pragma omp parallel
    {
        // Setup pseudopotentials and basis sets for boron, nitrogen,
carbon, and LFP atoms
    }
}

// Function to run DFT calculation
void runDFTCalculation() {
    // Parallel execution of DFT calculations
    // MPI is used to distribute different parts of the calculation across
multiple nodes
    // OpenMP will be used within each node to parallelize calculations at
a finer level
    // Divide the k-point grid among MPI processes
    // Each MPI process handles a subset of k-points here
#pragma omp parallel
    {
        // OpenMP to parallelize calculations for each k-point
        // Perform DFT calculations (e.g., solving Kohn-Sham equations)
    }
}

// Function to analyze the output
void analyzeOutput() {
    // Analysis of DFT results, which involve gathering data from all
nodes
    // MPI functions can be used to collect and aggregate data from all
processes
    // Gather band structure data (can also be potential, cv, resistance
data) from all nodes
    MPI_Gather(datas); // Gather results from all processes
    if (world_rank == 0) {
        // Master node processes and analyzes the gathered data
    }
}
```

```
int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    setupSupercellModel();
    prepareDFTInput();

    runDFTCalculation();

    analyzeOutput();

    MPI_Finalize();
    return 0;
}
```

We can use the designed large-scale DFT calculation to analyze our complex material in many attributes to reveal the doping that affects material properties at the atomic level. In the analysis of electronic band structure: we can reveal how co-doping with boron and nitrogen alters the band structure of the carbon material, which is directly related to its electrical conductivity and suitability as a cathode material. On density of states (DOS): we can understand the availability of electronic states at different energy levels, which influences the material's electronic properties. This analysis can show how electron density is distributed around the doped atoms (the density of charge for battery material), indicating changes in bonding and electronic characteristics due to doping. And also show the stability and reactivity of the doped material.

B. Molecular Dynamics (MD) Simulations for Diffusion Analysis

Molecular Dynamics (MD) simulations are an effective technique for simulating the movement and interaction of nitrogen and boron atoms within the carbon matrix at high temperatures, offering valuable insights into the doping process during calcination. These simulations implement classical Newtonian mechanics to calculate the trajectories of atoms over time, providing a dynamic view of atomic interactions and movements. The use of appropriate force fields, such as the Lennard-Jones potential, models the interactions between carbon, nitrogen, and boron atoms. Additionally, principles of statistical mechanics are applied to analyze diffusion coefficients and activation energies, offering a quantitative measure of atom mobility under various temperature conditions.

Algorithm design:

1. Initialize molecular system: Sets up the initial conditions of the simulation, including the positions and velocities of nitrogen, boron, carbon, and possibly lithium iron phosphate ($LiFePO_4$) particles.

2. Distribute initials: The master node distributes the initial conditions to the worker nodes for parallel processing, facilitating efficient computation across multiple nodes.

3. Calculate forces: The simulation implements a CUDA-based kernel to calculate forces between atoms using the Lennard-Jones potential. This approach allows for efficient parallel computation of interatomic forces on a GPU, significantly enhancing the simulation's performance.

4. Parallel force calculations: OpenMP is utilized to parallelize force calculations within each node, complementing the CUDA implementation for computationally intensive parts of the simulation.

5. Time integration loop: The positions and velocities of the atoms are updated over time, simulating their movement and interaction within the carbon matrix. This step is crucial for understanding the dynamic behavior of the system under study.

6. Gather results: The master node collects results from all worker nodes, which can include data on atomic positions, velocities, and other relevant parameters, providing a comprehensive view of the system's dynamics.

High-Level Pseudocode for MD Simulation:

```
#include <mpi.h>
#include <omp.h>
#include <cuda_runtime.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Constants and global variables
const int numParticles = 10000; // Number of particles
const double sigma = 1.0; // Lennard-Jones parameter for particle size
const double epsilon = 1.0; // Lennard-Jones parameter for depth of
potential well
const double dt = 0.001; // Time step for integration
const int num_steps = 1000; // Number of simulation steps
const int blocks = 256; // Number of blocks for CUDA
const int threads = 256; // Number of threads per block

double positions[numParticles][3]; // Positions of particles
double velocities[numParticles][3]; // Velocities of particles
double forces[numParticles][3]; // Forces on particles

// Function to calculate Lennard-Jones force between two particles
void calculateLJForce(double pos1[3], double pos2[3], double force[3]) {
    double r[3], distSquared = 0.0, distSixth, distTwelfth, fMagnitude;
    for (int i = 0; i < 3; ++i) {
        r[i] = pos1[i] - pos2[i];
        distSquared += r[i] * r[i];
    }
    distSixth = distSquared * distSquared * distSquared;
    distTwelfth = distSixth * distSixth;
    fMagnitude = 24 * epsilon * (2 * pow(sigma, 12) / distTwelfth -
pow(sigma, 6) / distSixth) / distSquared;

    for (int i = 0; i < 3; ++i) {
        force[i] = fMagnitude * r[i];
    }
}

// Function to calculate forces
void calculateForces() {
    // Initialize forces to zero
    for (int i = 0; i < numParticles; ++i) {
        for (int j = 0; j < 3; ++j) {
            forces[i][j] = 0.0;
        }
    }

    // Calculate forces using Lennard-Jones potential
    double force[3];
    for (int i = 0; i < numParticles; ++i) {
        for (int j = i + 1; j < numParticles; ++j) {
            calculateLJForce(positions[i], positions[j], force);
            for (int k = 0; k < 3; ++k) {
                forces[i][k] += force[k];
                forces[j][k] -= force[k]; // Newton's third law
            }
        }
    }
}

// CUDA Kernel for force calculations using Lennard-Jones potential
__global__ void calculateForcesCUDA(double* positions, double* forces, int
numParticles) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx >= numParticles) return;

    double pos[3] = {positions[idx * 3], positions[idx * 3 + 1],
positions[idx * 3 + 2]};
    double force[3] = {0.0, 0.0, 0.0};

    for (int i = 0; i < numParticles; ++i) {
        if (i == idx) continue; // Skip self

        double r[3], distSquared = 0.0, distSixth, distTwelfth,
fMagnitude;
        double pos_i[3] = {positions[i * 3], positions[i * 3 + 1],
positions[i * 3 + 2]};

        // Calculate distance between particles
        for (int j = 0; j < 3; ++j) {
            r[j] = pos[j] - pos_i[j];
            distSquared += r[j] * r[j];
        }

        // Lennard-Jones potential calculations
        distSixth = distSquared * distSquared * distSquared;
        distTwelfth = distSixth * distSixth;
        fMagnitude = 24 * epsilon * (2 * pow(sigma, 12) / distTwelfth -
pow(sigma, 6) / distSixth) / distSquared;

        // Accumulate force
        for (int j = 0; j < 3; ++j) {
            force[j] += fMagnitude * r[j];
        }
    }

    // Write the calculated force back to global memory
    for (int j = 0; j < 3; ++j) {
        forces[idx * 3 + j] = force[j];
    }
}


// Function to initialize the molecular system
void initializeSystem() {
    // Initialize positions and velocities of particles
    // This mainly contains Nitrogen, Boron, Carbon, and LFP particles
    // For simplicity, I'll place particles randomly and assign random
velocities
    for (int i = 0; i < numParticles; ++i) {
        for (int j = 0; j < 3; ++j) {
            positions[i][j] = (double)rand() / RAND_MAX; // Random
position
            velocities[i][j] = (double)rand() / RAND_MAX - 0.5; // Random
velocity
        }
    }
}

// Function to update positions and velocities
void updateParticles() {
    for (int i = 0; i < numParticles; ++i) {
        for (int j = 0; j < 3; ++j) {
            positions[i][j] += velocities[i][j] * dt + 0.5 * forces[i][j]
* dt * dt;
            velocities[i][j] += forces[i][j] * dt;
        }
    }
}

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Initialize the molecular system
    initializeSystem();

    // Main simulation loop
    for (int step = 0; step < num_steps; step++) {
        // Calculate forces
        calculateForces();
        // Update positions and velocities of particles
        updateParticles();
    }

    MPI_Finalize();
    return 0;
}
```

The MD simulations are expected to provide dynamic insights into the interactions of nitrogen and boron atoms with the carbon matrix during high-temperature calcination. This contrasts with previous research methods such as XRD, SEM, and TEM, which primarily offered static views of the material's structure. The MD approach allows for a deeper exploration of the doping mechanism, revealing how atomic mobility and reaction kinetics are influenced by the doping process. The simulation can calculate diffusion coefficients and activation energies, providing quantitative measures of atom mobility. Additionally, the observation of atomic trajectories will reveal the effectiveness of the doping process and its impact on the material's microstructure. By simulating different temperature conditions, the MD simulations can also predict the stability of the doped material and its phase behavior, crucial for assessing the material's suitability for battery applications.

C. Thermodynamic and Kinetic Modeling

The thermodynamics and kinetics of the calcination process is essential for optimizing the doping efficiency and material properties. This modeling will provide insights into the reaction rates, activation energies, and phase stability of the doped materials, directly correlating with the findings and objectives of the efficiency and material properties. Employ Arrhenius equations and reaction kinetics models to simulate the calcination process. This involves calculating reaction rates, activation energies, and equilibrium constants. Use thermodynamic data (enthalpy, entropy, Gibbs free energy) to model the phase transformations and stability of the doped materials under different temperature regimes. This part of code implementation will be the custom scripts and software for modeling. Use thermodynamic data (enthalpy, entropy, Gibbs free energy) to model the phase transformations and stability of the doped materials under different temperature regimes.

Algorithm design:

1. Calculate reaction rate using Arrhenius equation: calculates the reaction rate based on the Arrhenius equation, which is crucial for understanding the kinetics of the calcination process. It uses OpenMP for parallel processing within a node.

2. Calculate Gibbs free energy: This function computes the Gibbs free energy of the system, a key thermodynamic property that indicates the stability and phase transformations of the doped materials under various temperature conditions.

3. CUDA kernel for intensive calculations: The CUDA kernel is designed for computationally intensive operations, which could include modeling the doping process at an atomic level or other complex calculations.

4. MPI for data distribution and gathering: MPI is used to distribute data among different nodes and gather results, facilitating parallel processing across multiple nodes.

5. Integration of MPI, OpenMP, and CUDA: The code effectively combines these three technologies to leverage the strengths: MPI for distributed computing, OpenMP for shared-memory parallelism, and CUDA for GPU-accelerated computations.

High-Level Pseudocode for Thermodynamic and Kinetic Modeling:

```
#include <mpi.h>
#include <omp.h>
#include <cuda_runtime.h>
#include <math.h>

// Constants
const double R = 8.314; // Universal gas constant in J/(mol*K)

// CUDA Kernel for intensive calculations
__global__ void intensiveCalculationsCUDA(double* input, double* output,
int dataSize) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx < dataSize) {
        output[idx] = dopingOperation(input[idx]);
    }
}

// Device function for intensive operations
__device__ double dopingOperation(double value) {
    // operations
}

// Calculate reaction rate using Arrhenius equation
double calculateReactionRate(double A, double Ea, double T) {
    double rate = 0.0;
#pragma omp parallel
    {
#pragma omp for reduction(+:rate)
        for (int i = 0; i < some_large_number; ++i) {
            rate += F(Arrhenius)(A, Ea, T, i);
        }
    }
    return rate;
}

// Function to calculate Gibbs free energy
double calculateGibbsFreeEnergy(double enthalpy, double entropy, double T)
{
    double gibbsEnergy = 0.0;
#pragma omp parallel
    {
        gibbsEnergy = enthalpy - T * entropy;
    }
    return gibbsEnergy;
}

// Host function to launch the CUDA kernel
void runIntensiveCalculations(double* hostInput, double* hostOutput, int
dataSize) {
    double *devInput, *devOutput;
    cudaMalloc((void**)&devInput, dataSize * sizeof(double));
    cudaMalloc((void**)&devOutput, dataSize * sizeof(double));
    cudaMemcpy(devInput, hostInput, dataSize * sizeof(double),
cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocks = (dataSize + threadsPerBlock - 1) / threadsPerBlock;
    intensiveCalculationsCUDA<<<blocks, threadsPerBlock>>>(devInput,
devOutput, dataSize);

    cudaMemcpy(hostOutput, devOutput, dataSize * sizeof(double),
cudaMemcpyDeviceToHost);
    cudaFree(devInput);
    cudaFree(devOutput);
}

// Main simulation loop with MPI, OpenMP, and CUDA integration
int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Define parameters for the calcination process
    double A, Ea; // Parameters for Arrhenius equation
    double enthalpy, entropy; // Thermodynamic data
    double T; // Temperature

    // MPI: Distribute data among nodes
    if (world_rank == 0) {
        // Master node work
    } else {
        // Worker nodes work
    }

    // Calculate reaction rate and Gibbs free energy in parallel
    double reactionRate = calculateReactionRate(A, Ea, T);
    double gibbsFreeEnergy = calculateGibbsFreeEnergy(enthalpy, entropy,
T);

    // Prepare data for CUDA calculations
    double *hostInput, *hostOutput;
    int dataSize = 1000; // the size of data, say 1000 here
    // Allocate and initialize hostInput and hostOutput

    // Run CUDA calculations
    runIntensiveCalculations(hostInput, hostOutput, dataSize);

    // MPI: Gather results from all nodes
    if (world_rank == 0) {
        // Master node gathers results
    }
    // Free host memory
    free(hostInput);
    free(hostOutput);

    MPI_Finalize();
    return 0;
}
```

Thermodynamic with kinetic modeling is not pure computational problem, but rather a common way to form the characterization in most of the material preparation research. By simulating the reaction rates and thermodynamic properties, we can analyze into the efficiency and effectiveness of the doping process. Here in our code, a robust framework for simulating and analyzing the thermodynamics and kinetics of the calcination process is provided, the reaction rates and activation energies can indicate how quickly and efficiently the doping process occurs, which is crucial for optimizing material properties. The phase stability through Gibbs free energy calculations can help in predicting the long-term performance and reliability of the doped materials. This part of modeling and simulation take the integration with experimental data I gained before during the chemical characterization results, and combining simulation results with experimental findings can lead to a more comprehensive understanding of the doping process and its impact on electrical performance.

5. Expected Results and Conclusions

Electronic Structure Analysis via DFT is a common analysis method, we just use the MPI and parallel computation in this research. We can get band structure and DOS because the DFT simulations are expected to reveal significant changes in the electronic band structure and density of states due to boron and nitrogen co-doping. This could manifest as a reduced band gap or altered band alignment, potentially enhancing the electronic conductivity of $LiFePO_4$. At the same time the results will give insights into how electron density is redistributed around the doped atoms, indicating changes in bonding characteristics and electronic properties. At the same time, the simulations may show how co-doping affects the stability and reactivity of the material, crucial for assessing its performance in battery applications.

The MD simulations will quantify atom mobility and reaction kinetics, providing a deeper understanding of how doping affects Li-ion diffusion in the material. The results will provide atomic trajectories through the calculations of atomic movement and interactions, and will shed light on the effectiveness of the doping process and its impact on the material's microstructure. In the calcination process, there also exists phase transformation from solid state to liquid and back to solid state after cooling down. By simulating various temperature conditions, MD can predict the stability and phase behavior of the doped material, crucial for high-temperature applications.

The thermodynamic and kinetic modeling is all about the reaction rates & activation energies. These parameters will indicate the efficiency of the B-, N-, and B+N doping process, essential for optimizing the material's electrochemical properties. The calculation of Gibbs free energy will explain and predict the long-term performance and reliability of the doped materials we prepared in the material lab.

Overall, the integration with the experimental data gained through the chemical reaction and characterization and insert into the simulation process is the key: combining these simulation results with experimental findings (e.g., the existence of the N-B impurity, and the C-N, B-N conductive bond) will enhance the understanding of the doping process and its impact on the electrical performance of $LiFePO_4$.