Post-questions for HeartDown Expert Study

1. How different is writing in HeartDown from writing a paper or web page? *

Not very different from writing LaTeX/markdown.

2. In what ways is it more difficult? *

I would say the only thing I found uncomfortable at the beginning was I had to enter space to convert a command to an unicode character. Although I got used to it pretty fast, but it is proven to be an issue in code generation.

3. In what ways is it easier? *

1. Since we generally only write linear equations, I don't have to worry about LaTeX env. IHeartLA makes typing equations easier and faster.
2. HeartDown is an excellent tool to share tutorial online -- it highlights the vector dimension and variable meaning. And trust me, following all the vectors/matrices/their dims is the hardest part of reproducing a paper.

4. Did you run into limitations? *

1. no index offset: since I can't do vec[i-1], I had to make a copy of the same vec but shifted one index. This is too costly for memory.
2. constant loop range: I can't compute all physical energy in one go. I had to change vector length to make the same code work for different energies.
3. unicode char in generated code will cause error in compilation (C++ w/ gcc on Linux, and unicode is not a part of C++ standard so I assume other compilers will throw errors too).

5. Are there things you wish it did? *

1. loop range definition. It is very important to define the summation range in geometry and simulation.
2. index offset. If I want to do a prefix sum, it wouldn't be possible in IHeartLA. And vec[i-1] is definitely a common access pattern.
3. block wise addition: for (int i = 0; i < n; i++) mat[i, i] = block, where block is a 2x2 matrix.
4. save the changes done in web browser editor to local.

6. Do you have ideas for how to improve it? *

1. For C++ backend, there is definitely a better way to generate more expressive code. For example, convert std::vector of Eigen::VectorXd to Eigen::MatrixXd -- Eigen::MatrixXd is way more standard in geometry/simulation and have better performance.
2. Customized loop range. Since IHeartLA supports index wise equation (A_i = b_ix_i), then it makes sense to give the loop a range instead of just loop through the whole vector. In geometry/simulation, we usually don't need to iterate through the whole vector/matrix due to boundary conditions.
3. I understand the design choice of single letter variable naming convention, and it's good, keep it up -- but I would suggest to limit the variable names to only English alphabet letters and numbers. For Greek letters, maybe treat Greek letters like other alias LaTeX variables ("kappa" instead of "κ").

A general comment: note that we actually do not want all the equations written in the paper to be a part of the code -- some are impossible to convert to code, and some are just there to demonstrate math insights. But when we read the paper, we still want them to be defined and linked with other equations so we can follow them. There should be parsed/visualized only code section in IHeartLA.

7. How did you use the code it generated? *

I used the generated code to verify energy behavior -- usually we would expect they become stable with an energy conserved time integrator. I compared the results with the ground truth and it checked out.

8. What risks and benefits do you see in the reading environment? *

Overall it is a good design. It highlights the variables and equations in a friendly way (friendly to all kinds of readers TBH), which is the core contribution IMO. Some recent HCI study shows this has something to do attention span...
However, the design of "scope" limits paper writing. I think we would have to segment the equations in a logical order so they group under the same reading env.