Go with What You Know

W. Garrett Mitchener
Department of Mathematics
Duke University
P.O Box 90320
Durham NC 27708
wgm@math.duke.edu

A problem doesn't have to be fancy to be difficult.

—Michael Reed

The Mathematical Contest in Modeling is very different from most math contests. Having participated as a student and advised teams as a faculty member, I offer the following suggestions about what skills should be present in an MCM team. My main point is:

Good teams have a mixture of mathematical, computational, and writing skills.

I suggest that one member become the team's specialist in each of these areas, though all members usually contribute skills to all three. Unlike other contests, in the MCM problem-solving tricks are not as useful. Instead, teams must

- make creative use of what talents they have,
- know how to best use them in the limited time of the contest, and
- know how make good judgments about what tools to use and learn in the limited time of the contest.

What Mathematics to Use

I find that the most useful mathematical areas are probability and dynamical systems, because almost every modeling problem involves uncertainty, variability, and changing quantities. Specific problems may require knowledge from other areas. For example, graph theory might be useful for problems concerning networks, PDEs are often appropriate for materials-science problems, and an image-analysis problem may require signal processing. Obviously, no one team member should plan to be an expert in all things mathematical, nor should anyone plan to become an expert in PDEs in one weekend.

Some research will no doubt be necessary, and if you decide to read books or articles to learn a specialized topic, do the necessary reading but know how to stop before reading becomes a time sink. With that in mind, perhaps *the most useful mathematical skill is the ability to improvise*: You should be as creative as possible with what you *do* know. Improvising can improve your paper by

giving the problem a fresh perspective that will not be found through research. Additionally, the problems generally don't come with enough information for you to solve them completely. In real life, you would ask the problem poser for the information, but it may not be available, and it will certainly not be available during the contest, so very often you must improvise. For example, you might make additional assumptions that allow you to simulate the missing information. From there, you can show what you would do with the actual information if it were available.

Use what you know, but improvise as needed.

Computation

Computation and simulation are indispensable tools for applied mathematics. Someone on the team should be skilled in software development, and know how to

- write good programs rapidly;
- estimate the total time required to write, debug, run, and analyze a proposed computation; and
- propose alternatives to computations that will take too much time.

I prefer programming languages such as Python that are high-level and easy to use. A Python program that takes two hours to develop and runs in fifteen minutes is often better than an equivalent C++ program that runs in seconds but takes two days to write and debug. I remember several situations where I wrote a program in C++ and spend tremendous amounts of time fixing memory bugs and compiler issues. In the end, despite the speed of C++ the program turned out to be so time-consuming that it never finished running. With 20/20 hindsight I can say: There simply isn't enough time in a weekend for that. You don't want to end up with a paper that just says you designed a simulation but couldn't run it, with no results or analysis.

Mathematica, Maple, and Matlab can be indispensible time-savers, since they include high-level programming languages with built-in solving and graphics commands. However, if you plan to use one of these, start learning it well ahead of time; you don't want to waste a lot of time during the contest trying to learn a complex software package. As with mathematics, I recommend above all that you make the most of what you already know. If you discover during the contest that there are better tools and methods available, see if you have time to learn them, but consider leaving them for next year.

Program in what you know, but in advance learn a useful language well.

Writing

Writing skills at all levels of detail are crucial. It's surprisingly easy to spend all weekend on a problem and write an incomprehensible paper—without realizing it. You understand the problem and your team's solution thoroughly (well, you hope so); but to write a good paper, you must step back and adopt the perspective of someone who knows nothing about what you did.

The paper needs to have consistent notation, a list of references, and appropriate graphics. It should be organized, coherent, and readable. Identify vague points in the paper. For example, if you find yourself writing that a certain result "looks better," figure out specifically what makes it look better and use that as an opportunity to improve the analysis of your result and make a stronger, more correct, and more precise statement in your paper.

The judges are looking for specific features, such as

- clearly-stated assumptions,
- a thorough understanding of the problem (including natural questions that weren't specifically asked in the problem statement), and
- a clear description and analysis of your model.

Someone on the team needs to be in charge of making sure that all of these features are present and easy for the judges to find. Given that the task of writing will often be split among all three team members, someone should be in charge of assembling fragments written by each member and checking them for consistency and redundancy. Most models come in phases of increasing complexity, and it's crucial that the paper clearly state which assumptions and conclusions are true of which phases. With three contributors, confusion among phases and inconsistent notation can easily creep in. Someone must be responsible for finding and eliminating all these potential sources of errors so that the explanation of the model and its behavior are clear.

On a more mechanical level, someone needs to be proficient with the typesetting or writing software you use, such as LaTeX or Word, so when you need to add page numbers, merge revisions, or import graphics in a particular way, you don't waste time looking through manuals or online help. Again, go with what you know; and if you don't know how to do a particular task, spend a reasonable amount of time trying to figure it out, but know when to improvise and go with Plan B.

Write what you know, as if your reader doesn't know about it.

Conclusion

In conclusion, the MCM challenges teams to balance creativity against existing knowledge, and quantity against quality. Team members should be aware

of their own strengths and weaknesses, and make certain that mathematical, computational, and expository skills are all covered. They should be prepared to explore new tools during the contest, but with the realization that it's often best to go with what you already know.

About the Author

I'm a graduate of the North Carolina School of Science and Mathematics (NCSSM) and Duke University, and I participated in the MCM at both schools. Once at NCSSM and twice at Duke, my teams earned the Outstanding designation. I earned my Ph.D. at Princeton University in applied and computational mathematics, and I have returned to Duke as a post-doctoral research associate. Currently, I help organize Duke's MCM teams. My research is in mathematical modeling for linguistics. In my spare time, I play oboe for the Durham Community Concert Band, dabble in gardening and calligraphy, and lately I've been learning to juggle. I live in Durham NC and attend Blacknall Presbyterian Church.