Complement A

The Seven Habits of Highly Effective Problem Solvers: An Approach to Thinking about and Solving Physics Problems

> . . . it's so simple,
> sublimely simple.
> If you learn not just to see but to observe.
> Put your brain to work, not just the optic nerve
> If you put your mind to use,
> You will find the most abstruse
> Becomes so simple . . .
>
> —Sherlock Holmes, in
> *Baker Street: A Musical*

A.1 Introduction

When we reach problem-based courses in subjects like physics, we lug with us lots of baggage we need to get rid of. Among this junk is a gaggle of *problem-solving techniques* most people pick up in high school. These techniques may serve us when we're tackling simple plug-and-grind problems, but many of them are worse than useless when we advance to more interesting problem situations. By now, many of these habits are almost second nature. I believe they're pernicious, and that you'll have a lot more success and a lot more fun if you work on changing your habits. This handout describes one approach to more thoughtful (and enjoyable) problem solving. One caveat: in a sense, this handout is limited because it's "too general": it lists strategies and tactics, along with several explicit suggestions for their implementation but contains no specifics, no illustrations. *If you decide to try this approach, it's imperative that you seek*

examples at every opportunity. Your courses will provide lots of opportunities: in workshops, on homework, in texts, etc.

Three premises are inherent in this approach. First, you already know much more than you realize; your problem is accessing and using that knowledge effectively. And second, knowledge is *hierarchical and cumulative*, and the most effective ways of accessing and using knowledge take these features into account. Third, problem solving is a lot of work. So it should be as fruitful as possible (not just numbers—anyone can generate number—but *understanding*! Insight! It should be as painless as possible. And it should be (here comes a heretical statement) as much fun as possible. These principles seek to help you approach problems so as to maximize fun and insight and minimize pain, algebra, and tedium.

THE SEVEN PRINCIPLES

- **The Principle of the Staged Assault.**
- **The Principle of Brainstorming.**
- **The Principle of Modeling.**
- **The Principle of Pattern Seeking.**
- **The Principle of Back to Basics.**
- **The Principle of Least Algebra.**
- **The Principle of Alert Awareness.**

A.2 Implementation Strategies

A.2.1 The Principle of the Staged Assault.

> He who hesitates is lost,
> he who doesn't plan
> is in really big trouble.

The basic idea of this approach to problem solving could hardly be simpler. First and foremost, approach problems systematically, strategically, and in a spirit of play.

Approach problems in three stages:

1. **Feed your mind, then cut it some slack.**
2. **Brainstorm the problem.**
3. **Go for it.**

Let me explain.

- When you decide to (or are assigned to) work a problem, *read it immediately.* Just read it: don't try to solve it; don't fret about it. But read it with your mind, not just your eyes.
- If time permits, put the problem aside for a day or so and go about your life while your mind digests and works on the problem. Then take out the problem and brainstorm (see below). Then *and only then* take out your calculator, tables, text, scratch paper, and start work.
- *Don't put yourself under unnecessary stress by letting the problem sit around too long.* (Procrastination isn't immoral, but neither is it cost effective.) Start work early. Give yourself a break: Work on problems during frequent brief sessions, rather than cramming everything into one marathon slugfest with the problem.
- Always be willing to put a problem you're stuck on aside for a while, to return to it later. You'll be surprised what your mind figured out while you were doing other things!

A.2.2 The Principle of Brainstorming.

Plan. Think about what you're going to do, then do it.

Brainstorming is easy. You're not trying to work the problem. You're not trying to get the right answer. You're not doing calculations. You're not rummaging through books looking for relevant equations. You're just playing around with the ideas raised by the problem. Once you're into this approach, you'll find that brainstorming won't take long—but I guarantee that brainstorming will save you *huge amounts of time and agony* when you get into actually working the problem!

A few guidelines are essential to successful brainstorming.

- Jot down as many ways to approach the problem (or parts of it) as you can think of.
- Don't worry at all about practical issues (can I do this? how hard would it be?)
- Brainstorming should involve few or no equations and few or no calculations; rather, you're trying to wrap your mind around the problem. Once you've done that, the equations and calculations will follow—and be lots less trouble than if you'd jumped right to them!

Key questions for a brainstorming session might include

1. What is the problem really asking for? That is, what are the key questions? Do I understand the problem? (To check, try to restate it in your own words *without referring to the original statement.*
2. What are the physical principles involved in the problem situation? (This does not mean "what equations do I use?")
3. Can I simplify the problem with approximations or by modelling the system (see below)? Can I think of more than one such approximation or model?
4. How can I express my understanding of the problem in sketches? Draw lots of *simple, rough, qualitative sketches*—as many as you can think of—to illustrate or clarify the situation posed in the problem.

♦ **Tip**

Try to keep your brainstorming notes informal but neat enough that you can read them. One part of these notes should be really clear and neat. Prepare a clear *written* list—of about half a page—of the key elements of the problem (quantities, physical principles, data supplied, etc.). When you start work on the problem, this list is your guide, your map out of any swamps of deadly algebra or tedium into which you might sink.

♦ **Tip**

Once you've come up with one way to approach the problem, stop. Reconsider. Ask yourself, "What is another way to solve this problem? Can I think of several ways?" If so, jot them down. Then ask, "which way seems easiest and clearest and most direct?" That's the place to start!

A.2.3 The Principle of Modelling.

Depending on the nature of the problem you're interested in, a vital part of your preliminary brainstorming may be designing a **model** of the physical system involved. The model follows naturally from your ideas about what physical principles and quantities are most important for the problem–ideas you developed in your brainstorming session. Key questions for model building include

1. *Precisely* what quantities do I need to evaluate?
2. What physical features of the system will affect these quantities?[1]
3. Which of these features will *least* affect the quantities I want?
4. How accurately do I need to know these quantities?
5. Based on this analysis, which features of the physics *must I retain* to get the accuracy I desire in the quantities I'm interested?

In your first stab at a model, jettison every feature of the physics that didn't meet the criteria you've established. That is, once you've answered these questions, *throw out as much of the complicating physical aspects of the problem as possible.* What's left is your model.

[1] For example, in many such problems you can get insight into this matter by thinking about the physical meaning of the terms in the potential energy.

♦ **Tip**

Don't haul out the equations too soon! Only when you can write down a *verbal* description of your model (perhaps with rough sketches) and identify the approximations inherent in the model (with justification)... only then are you ready to translate your ideas into mathematics (equations and data).

A.2.4 The Principle of Seeking Patterns.

> You know a lot of stuff. Use what you know. Familiarity Breeds Success. Before you start work, try to relate what you know to what you want to know. Seek patterns over and over as you work.

At several intervals during your work on the problem, as yourself questions like the following.

- What is *the main thing* this problem is asking? What do I already know about this thing?
- What *patterns and relationships* do I see in this problem?
- *How does this problem resemble problems I've already solved?* That is, seek analogies between problems you've solved and the one you want to solve. *Then list the similarities and differences between the two problems!* Often this step alone will trigger your mind to find the road to an efficient, effective approach.
- How can I adapt known results from other physical systems to the context of this problem?

♦ **Tip**

If you're having trouble formulating the problem you've been asked to solve, solve another problem! That is, devise a simpler problem that involves the same physical principles and quantities. The simpler the better! Then seek patterns between the simpler problem(s) and the one you want to solve.

Now that you've completed the three preliminary stages, we're ready to start work. The following strategies "interact." That is, as you work the problem you'll move back and forth among them. As you work, repeatedly refer to and update your brainstorming notes, model design, and

related problems and patterns! They're your guide to efficient, effective problem solving!

A.2.5 The Principle of Back to Basics.

> Whenever you feel yourself getting lost or swamped, stop, look at your list of key elements. Try to navigate back to the basic physical concepts, quantities, and questions of the problem. This may mean temporarily abandoning whatever tributary you've gone down, but that's okay; you can always return later.

As you dig into even a moderately complicated problem, you'll probably find yourself moving deeper and deeper into a mire of symbols, explicit equations, numbers, units, and algebra. What do you do if you get really lost... if you get that sinking feeling that you're just treading quicksand—working hard but getting nowhere? Don't panic. We've all been there—and you can pull yourself out.

Usually people get lost in solving problems for one of three reasons:

1. They don't know where they want to go and haven't a clue how to get there. (This won't happen to you, because you've brainstormed the problem.)
2. They're using equations that are more complicated than they need to be. (This won't happen to you if you keep your attention focused on the physical principles involved, rather than just trying to find any old equation that relates the physical quantities given in the problem.)
3. They're trying to do too many things at once (juggle equations, solve equations, plug numbers into equations, etc.)

You can save yourself lots of work and even more grief, if you'll follow these guidelines. When you feel yourself loosing your way, the first thing to do is

STOP!

Take a deep breath. Go get a coke (or whatever). Now ask yourself

1. Am I following one of the approaches I laid out in my brainstorming, or have I gone off on a detour? If the latter, is this a productive detour, or am I just goofing around?
2. Do I *really* have to use these complicated equations? Are there more variables in these equations than appear in my key list? Can I find a simpler equation that mathematically articulates the physical principles I've identified?

3. Am I trying to do several things at once?

♦ **Tip**

The last point is worth emphasizing. Particularly when you're facing a new type of problem or one that is otherwise unfamiliar, *approach the problem in stages* (see above). Select the relevant equations. Relate these equations. Try to simplify them if necessary. Plug explicit expressions into them (see below). Plug numbers in, if necessary. Interpret the equations physically and/or graphically. *But never never do all this at once.*

♦ **Tip**

Interact with your key element list as you work. Update and annotate new quantities you introduce, equations involving these quantities, and intermediate or auxiliary results. Keep it neat. You need this map! It may save your life (metaphorically speaking, of course).

A.2.6 The Principle of Least Algebra.

> There is no problem which,
> however difficult,
> cannot be made more difficult
> if you approach it right.

We now come to my personal choice as the single most important principle of effective problem solving. The basic idea is simple.

> Clean up your act! Before you start manipulating equations, do everything you can to make them look as simple as possible (lots of suggestions follow). As you work, repeatedly try to simplify their mathematical structure. *Above all, at each stage of your work on a problem, use the least explicit forms of physical expressions and equations you possibly can.*

- Keep it simple—but not too simple: *At each stage of your solution, use the least explicit expressions you can and still make progress.*
- *Keep your focus on the mathematical structure of the expressions and equations you're using.*
- *Once you've formulated the problem, express everything in terms of dimensionless quantities*, in order to reduce clutter and expose the underlying structure of the expressions in the problem.

- *At frequent intervals, clear away the algebraic underbrush.* Try to eliminate algebraic clutter (e.g., by introducing auxiliary constants, variables, and functions) so you can see clearly the structure of your expressions.

A.2.7 The Principle of Alert Awareness.

Okay, I lied. This principle is even more important than the last one. If you follow this alone—and ignore all my other advice—you'll still markedly improve your effectiveness as a problem solver. (And you'll have less pain and more fun.)

> *Stay alert. Keep your brain engaged! Don't let the problem lull you into a mindless stupor! Never solve problems on autopilot.*

Stop frequently to ask yourself questions like the following,

- What do these quantities I'm working with mean physically?
- If I don't know *the* answer, then what *would* an answer to this question be like?
- Is this problem becoming more complicated than I think it should?
- Do my answers or intermediate results make sense? Are they of sensible order of magnitude? Are they dimensionally correct? Do they "feel wrong" somehow?

♦ **Tip**

Don't forget that your work on a problem isn't through until you've checked *dimensions*, *orders of magnitude*, and *simple limits*—and anything else you can think of!

ADDENDUM

Habits of Ace Computational Physicists.

Although the overall strategy of tackling physics problems you want to solve on the computer is similar to the "Seven Habits" discussed above, successfully using the computer differs *tactically* from solving problems with paper and pencil. Several of the steps listed below are equivalent, in a computational context, to "brainstorming" a paper-and-pencil problem. I guarantee that carrying out these steps will save you (a) lots of grief; (b) lots of time, and (c) lots of tedium. First and foremost:

> The computer isn't a toy. Nor is it an end in itself. It's just a tool—a souped up hand calculator. It will do what you tell it—*precisely* what you tell it, whether or not that's what you *want* it to do. Like you and me, it makes mistakes. Unlike you and me, it can't think.
> **Never use the computer unless you have to.**

The Plan

No matter how much computation you think a problem will involve, *your first step should be to sit down in a quiet place with paper and pencil and make a plan*. Write your plan on a **help sheet** you'll keep by your side at the computer. Here are some suggestions.

1. **The first step.** Write down the *equation(s)* you want to solve and their *initial or boundary conditions*. Write this information down *very carefully and in as clean a form as possible.* Illuminate the *mathematical structure* of the equations, not their details. *Write all auxiliary definitions clearly and completely on your help sheet.*

⧫ **Tip**

Use The Principle of Least Algebra. Get rid of algebraic clutter! Use only dimensionless quantities! Judiciously define auxiliary quantities (variables, functions) so as to to absorb numerical debris (superscripts, constants, subscripts, arguments) that obscures the all-important *mathematical form* of the equation.

⧫ **Tip**

Before you sit down at the computer, double check all signs, exponents, and multiplicative factors you've not been able to absorb into dimensionless variables and other auxiliary functions.

---

2. **Know your algorithm.** Be sure you know *precisely* what quantities the *numerical algorithm(s)* you'll use require and how you're going to introduce these quantities. Write this information down on your help sheet.

3. **The Game Plan.** *Develop a rough game plan—a pencil-and-paper sketch of what precisely you want to calculate, laid out step by step.* This is the final part of your help sheet. Annotate your plan with ideas that occur to you as you plan your assault on the problem.

♦ **Tip**

A good game plan includes at least the following: definitions of intermediate results, expressions, or commands, written down so you can easily refer to them as you solve the problem; lists of arguments you intend to use for any functions you will define; a careful (tentative) layout of each argument list; note of particular commands you intend to use (e.g., plot commands, special functions) and their syntax.

4. **Test Cases.** Design a few carefully selected *test cases* for each major command and group of commands in your game plan. Test cases should include (a) simple tests for which you know the right answer; (b) tests that stress the algorithm(s) you're using; (c) tests that encompass the range of the domains of all functions you define. Add a thorough description of each test case *and its correct solution* to your help sheet.

Now you're ready to go to the computer and make it work for you. What is the single most vital guideline for this stage in the problem? Easy!

NEVER COMPUTE ON AUTOPILOT.
KEEP YOUR BRAIN ENGAGED
AT ALL TIMES.

Here are a few other pointers:

1. Proceed as systematically as possible. *NEVER TRY TO DO MORE THAN ONE STEP AT A TIME.*
2. Check *all* intermediate results, definitions (assignments), modules, and commands when you define them. Test them against *test cases* you've prepared in advance.
3. *Don't trust the computer.* View every result it gives you skeptically. Nearly all these results should be ones too complicated for you to work out on paper (otherwise you'd already have worked them out), so you have to verify results the computer produces wherever possible.
4. *Graph results whenever possible.* Don't be satisfied with an equation that's too complicated to interpret easily or a table of numbers that doesn't easily reveal its secrets. The saying "a picture is worth a thousand words" is a cliché. But it's also true.
5. *Avoid "computer snooze mode."* Sufferers from this deadly affliction lapse into a semi-catatonic state in which their fingers continue to type commands but their brain ceases to function. Something about computers—especially PCs—tends to induce this mode, leaching all fun and insight from computational problem solving.

♦ **Tip**

Remember: your work on a computational problem is never finished until you've written down your interpretation of the physics you have learned from the calculation. What does your result tell you about the problem?

For Those Stalwart Souls who Want to Program.

Although good programming practices are very specific to the language you're using and may be particular to the type of problem you're solving, a few general principles can minimize the pain and maximize the gain:

> **Be clear, not clever.**
> **Modularize.**
> **Test small segments of the code first.**
> **Document now!**

1. **Step by step by step.** Determine the essential numerical step(s) of the numerical algorithm(s) you intend to implement. List these steps, breaking them down into logical sub-steps if appropriate.

2. **Group logically connected steps.** Examine the algorithm carefully to see which steps seem logically connected; note groupings of connected commands—these will probably correspond to single functions or expressions that you will define.

3. **One step, one command.** Carry out each sub-step in a single command on a test case for which your know the answer.

4. **If you use *Mathematica*, use lists.** If you're using *Mathematica*, then keep in mind the following principle: To as great an extent as possible, carry out each steps using list manipulations, i.e., try to think about each step like *Mathematica* thinks. List manipulations are usually the fast, clearest way to code.

5. **Be clear, not clever.** Unless you're writing a research code that devours CPU time, in which case you may need to be clever to save time, *always strive for the clearest, most readable code possible.*

♦ **Tip**

In *Mathematica*, this principle should influence names of *Mathematica* commands, use of local variables in `Modules`, choice of which individual commands to group within modules—in sum, every aspect of your code development.

---

6. **Construct commands around logically connected steps.** As you implement and test each step, collect the logically connected steps into FORTRAN functions or subroutines or, if you're using *Mathematica*, into *Mathematica* commands.

♦ **Tip**

In *Mathematica*, Use `Module` to ensure the local scope of expressions (e.g., variables, lists, functions) generated at each stage.

---

7. **Know how your algorithm can fail—and make it fail!** Be sure you've looked up (or figured out) all error(s) inherent in your algorithm. Do test cases that push the algorithm near the limits of its inherent error. If possible, devise a test case that causes the algorithm to break down numerically. If you know what to look for when it breaks down, you're better equipped to spot such a catastrophe in the making in your actual applications.

8. **DOCUMENT YOUR CODE WHILE YOU WRITE IT.** Writing documentation is a bit of a pain. But it is nowhere near as agonizing as returning months later to your code (or giving it to a friend) and having to spend huge amounts of time deciphering its workings!

♦ **Tip**

Documentation should take two forms: (a). Brief remarks including sources and an example within the code itself (if you're using *Mathematica*, see the `Skeleton.m` package that comes with *Mathematica*) and (b). Paper documentation, including your game-plan help sheet, a brief summary of the algorithm with references and definition of notation, and a test case (sample input and output).

The Final Principle

Physics is work. Like most tasks of the mind, it's best approached seriously but not solemnly, in a spirit of "intellectual play." So go ahead: have fun!