8.6 Optimization Checklist

The early stages of an optimal design project are critical for the success of the entire effort. Selecting the proper framework to define the project requires care and rigor. It is extremely helpful to express the optimization problem in words before describing it mathematically. With this in mind, we now suggest the following tersely listed items.

Problem Identification

1. Describe the design problem verbally. What goals do you want to achieve? Do they compete?
2. Identify quantities you could vary and others you should consider fixed. What specific trade-offs result from trying to satisfy competing goals?
3. Verbally express the trade-offs in optimization terms.
4. Search the literature for optimization studies related to your design problem.
5. Verify, in the final description of your problem, that the trade-offs are meaningful enough to justify further optimization study. Recognize that constraint-bound problems have obvious solutions.

Initial Problem Statement

1. Verbally define the objective function and constraints. If several objectives exist, decide how you will translate the multiobjective problem to a model with a single objective.
2. If you chose one objective and treat other objectives as constraints consider how you will set proper bounds. Expect to do a post-optimal parametric study on these bounds. If you chose a weighted scalar objective, consider how you will chose the weights. Expect to perform a post-optimal parametric study on these weights to generate a Pareto solution set.
3. List all modeling assumptions, updating the list as you proceed through the study. Remember that these will have to be checked at the end of your study to see whether they are satisfied by the final design.
4. List separately which design quantities have been classified as variables, parameters, and constants. You may decide to update or modify this classification as you proceed with the problem. Recall that only what *you* can choose can be a design variable in *your* problem statement.
5. Evaluate the availability of mathematical analysis model(s) for describing the objective and constraints. Will the models be explicitly (algebraically) or computationally defined?
6. When the analysis models are separate codes, "black box" simulations, or legacy codes, consider how the different software components will communicate with the numerical optimizer.

Analysis Models

1. Identify the system to be modeled, its boundary, and its links to the environment. Use appropriate system diagrams, for example, control volumes or free-body diagrams. Describe the system's function verbally.
2. Identify the system's hierarchical level and choose the limits of your decision-making capability. This will allow you to define more clearly the design variables and parameters in your model.
3. Verbally list inputs, outputs, and relating functions for each analysis model. If your model is a "black box" simulation you still need to understand what are the exact inputs/outputs of the simulation.
4. State the natural laws and principles of engineering science governing the system.
5. List all modeling assumptions, continuing from the initial problem statement.
6. Develop the analysis model(s) if needed. For readily available models, state the mathematical form and take results directly from the reference source. Recheck that you know explicitly the assumptions made in the reference.
7. Gather any empirical information from available data or generate empirical data if you have to.
8. Use curve-fitting techniques to develop appropriate analytical expressions from empirical data available in tables or graphs.
9. Distinguish carefully among variables, parameters, and constants for the analysis model(s). Compare these lists with the input/output list of (3).
10. Collect all results from analysis and curve fitting and assemble all equations relating the system quantities. Verify that you have a complete capability for computing outputs from inputs.
11. Check the model by calculating, for simple cases, desired quantities for selected values of parameters and constants.
12. Do a preliminary evaluation of model efficiency, accuracy, and operation under the stated assumptions.
13. For an implicit model or simulation, estimate typical computation time for each analysis run. Is computing time too long?
14. For computationally expensive models, consider creating a metamodel. Use such a metamodel instead of the full one in your optimization model. Recognize that building metamodels is also computationally expensive.

Optimal Design Model

1. State the design model, listing objective and constraint functions and deriving their mathematical forms.
2. Carefully define and list all design variables, parameters, and constants.

3. Distinguish clearly between equality and inequality constraints, keeping them in their original form without attempting to simplify them.
4. If some functional relations are still unavailable, try to state them in a general symbolic form, for example, $f(x_1, x_2, x_3) = x_4$, which at least shows which variables are involved. Attempt to find an approximate relation from an analysis model, experiments, or a consultant's opinion.
5. If the design model requires iteration of a computer-based model, consider using metamodels based on computational experiments, as mentioned earlier.
6. Review all modeling assumptions to see if any new ones have been added to the design modeling stage.
7. For typical values of the parameters, verify that the feasible domain is not empty. Check for inconsistency in the constraints. Finding at least one feasible point should be attempted using your engineering knowledge about the problem or information on a current design.

Model Transformation

1. Count degrees of freedom. Verify this invariant count after each further model transformation or simplification.
2. Identify monotonicities in the objective and constraint functions.
3. Use monotonicity analysis procedures wherever possible. Every monotonic variable in the objective must have an appropriate bound. If not, check for missing constraints, missing terms in the objective, or oversimplified approximations. Also check the hierarchical system level assumptions. Is the model well bounded?
4. Check validity of all transformations. For example, if you divided by something, it cannot be zero. Must this restriction be made explicit?
5. Look for hidden constraints such as asymptotes. For example, you may eliminate asymptotic expressions by case decomposition and additional constraints as appropriate.
6. Check for redundant constraints.
7. Perform a reduction of the feasible domain to tighten up the set constraint and perhaps find explicit bounds based only on natural constraints.
8. Drop all practical constraints not needed for well boundedness. Mark any remaining for further study.
9. For any revised model, (i) make a monotonicity table; (ii) apply monotonicity analysis to identify constraint activity; (iii) perform case decomposition based on activity status and solve each well-bounded case if their number is small.
10. Does the problem have discrete variables? If a discrete variable takes only a few (discrete) values treat it as a parameter and resolve the problem for each parameter value.

11. Does a discrete variable have an underlying continuous form? If so, consider relaxing the discreteness requirement. The continuous relaxation problem solution will be a lower bound on the discrete one. Set up a branch-and-bound strategy if necessary.
12. If several variables are truly discrete (e.g., 0,1) methods beyond those of this book are likely necessary.

Local Iterative Techniques

1. If you have equality constraints, see if you can put them into a function definition format. For example, if the problem is: min $f(x_1, x_2, x_3)$, subject to $h_1(x_1, x_2) - x_3 = 0$, use $x_3 = h_1(x_1, x_2)$ as a definition and compute $f[x_1, x_2, h_1(x_1, x_2)]$ instead of treating $h_1 - x_3 = 0$ as an equality constraint.
2. Watch out for function forms defined over intervals of the variables that may be violated during iterations. Recall that some algorithms will temporarily violate constraints. Make transformations to avoid such violations if they will lead to a computer "crash."
3. Select a starting base case that is feasible. Although most codes accept infeasible starting points, as a designer you should be able to find a feasible start by iterating the analysis models. If not, beware of constraint inconsistency.
4. Scale the problem using the base case values. Rescale as necessary, so that (i) all variables and functions have about the same order of magnitude and/or (ii) the objective function is properly "sensitive" to changes in variables and constraints, so that you avoid getting trapped in flat areas.
5. Carefully choose values of the termination criteria, taking scaling effects into account. Beware of noisy functions!
6. Are the model functions all smooth as required by local methods? If lack of smoothness is modest, Newton-like methods work better. If there is substantial noise, discontinuities, or nondifferentiabilities, consider your options: Look for nongradient-based methods, develop surrogate models, or try to eliminate the sources of trouble in the model.
7. If the model is explicitly algebraic, finite differencing can be avoided by computing gradients symbolically.
8. Do not assume a constraint is active just because its Lagrange multiplier does not vanish completely. Small values may indicate degeneracy or ill-conditioned computations. Monotonicity arguments may be needed for reliable results.
9. Numerical results should be verified analytically if possible, particularly for determining constraint activity. Apparently reliable results obtained by local methods will justify an attempt at global verification.
10. For parametric study, a special purpose program combining case decomposition and local iteration may be needed to generate reliable results efficiently.

Final Review

1. Is the solution sufficiently robust? Try different starting points. If you get different results, examine if this indicates sensitivity to the termination criteria or existence of multiple minima.
2. Consider using a multistart strategy, namely, select starting points that will tend to "cover" the feasible domain, in an effort to find all local minima.
3. Is the model accurate enough near the optimum?
4. Are the model validity constraints satisfied? Should some assumptions and simplifications be discarded or modified?
5. Should the classification of variables and parameters be changed?
6. Would a new model be at the same hierarchical level, but more complex and accurate? Should you move up in the hierarchy of a larger system thus increasing your decision-making options?
7. Define and explore a second-generation model.

8.7 Concepts and Principles

Finally, it is time to summarize the key ideas reviewed and developed in this book. The checklist of the previous section mixed them in with the many routine but important steps of a design optimization study. Here we list concepts and principles introduced throughout this book that should by now be familiar to the reader. The aim of the book is to create design optimization practitioners. The list can serve as a self-test to a designer's readiness to face the challenges of real-world problem solving.

Unlike the earlier checklist, this compilation emphasizes function rather than position in the project time schedule. We discern three major functions: building the model, analyzing the model, and searching for local optima. Concepts are listed within these categories without any particular order, along with pertinent section number.

Model Building

Model setup

- Negative null form (1.2)
- Hierachical levels (1.1, 1.2)
- Configuration vs. proportional design (1.2)
- Design variables, parameters, constants (1.1)
- Analysis vs. design models (1.1)
- Degrees of freedom (1.3)

- Model validity constraints (1.4, 2.5)
- Surrogate models
  - Curve fitting (2.1)
  - Families of curves (2.1)
  - Least squares (2.2)
  - Neural nets (2.3)
  - Kriging (2.4)
- Multiple criteria (1.2)
- Pareto set (1.2)
- Natural and practical constraints (2.8)

Model properties
- Feasibility (1.3)
- Activity (1.3, 3.2, 6.5)
- Well boundedness (1.3, 3.1, 3.2)
- Monotonicity (3.3, 3.4)

Anticipated solution
- Interior vs. boundary optima (1.4)
- Constraint-bound optima (1.3)
- Local vs. global optima (1.4)

Model Analysis

Monotonicity
- Monotonicity theorem (3.3)
- First monotonicity principle (3.3)
- Second monotonicity principle (3.7)
- Recognition of monotonicity (3.4)
- Monotonicity table (6.2)
- Regional monotonicity (3.6)
- Directing equalities (1.3, 3.6)
- Functional monotonicity analysis (6.3)

Activity
- Activity theorem (3.2)
- Dominance (1.4, 3.5)
- Constraint relaxation (3.5)

- Semiactivity (3.2, 6.5)
- Weak vs. strong activity (6.5)
- Tight constraints (6.5)

Criticality (3.3)

- Conditional (3.5)
- Multiple (3.5)
- Uncriticality (3.5)

Parametric solution

- Parametric models (6.1)
- Parametric functions (6.1)
- Case decomposition (6.2)
- Branching (6.2)
- Parametric testing (6.1)

Partial minimization (3.2)

Model repair (3.3)

Model reduction and simplification

- Variable transformation (1.2, 2.5, 2.7, 2.8)
- Variable elimination (1.3, 3.2, 3.3)
- Asymptotic substitution (2.8)
- Redundant constraint (1.4)
- Feasible domain reduction (2.8, 6.7)

Discrete variables

- Discrete local optima (6.5)
- Multiple optima (6.5)
- Constraint tightening (6.5, 6.6)
- Constraint derivation (2.8, 6.7)
- Rounding off (6.7)
- Continuous relaxation (6.8)
- Branch-and-bound (6.8)
- Exhaustive enumeration (6.8)

Local Searching

Model forms

- Weierstrass Theorem (4.1)
- Compact set (4.1)