

Engineering Sketch Pad (ESP)



Advanced User Training Session 02 Using Miniopt for Packaging

John F. Dannenhoffer, III

john@geocentrictech.com

Geocentric Technologies LLC

updated for v1.29+

- Optimization basics
- Optimization techniques
- The Miniopt tool
- Using EVALUATE DIST in ESP
- Homework exercises

- Design parameters
 - values prescribed by the customer that “define” the requirement(s) of the product being designed
- Objective function (objfun)
 - the quantity that is to be minimized (or maximized)
- Design variables (optvar)
 - the values that can be changed (by the designer) to achieve the minimum/maximum
- Constraints (constr)
 - explicit and implicit limits on how the design variables can be changed

- We want to design a 10 foot long, cantilevered, steel beam (with a rectangular cross-section) that deflects less than $1/4$ inch under a tip load of 10000 lb. How light can we make it if the width-to-height ratio has to be between 0.10 and 10.0?

- Design parameters
 - Length: $L = 10$ ft
 - Force: $F = 10,000$ lb
 - Density: $\rho = 15$ slug/ft³ (for steel)
 - Modulus: $E = 30 \times 10^6$ lb/in² (for steel)
- Objective function (objfun)
 - minimize the mass (M) of the bar
- Design variables (optvar)
 - beam height: h
 - beam width: w
- Constraints (constr)
 - deflection: $d \leq 1/4$ in
 - aspect ratio: $0.10 \leq h/w \leq 10$
 - positivity: $w > 0, h > 0$

- Mass of beam

$$M = \rho whL$$

- Moment of inertia

$$I = \frac{wh^3}{12}$$

- Deflection of beam

$$d = \frac{FL^3}{3EI}$$

- Define design variables and sequence
 - h is the height in inches
 - w is the width in inches
- Define the variable to be minimized
 - if minimization problem, $obj = +value$
 - if maximization problem, $obj = -value$
 - for this beam problem, $obj = mass$

- Linear vs. nonlinear
 - for linear: can find global minimum
 - for nonlinear: at best can find local minimum
- Unconstrained vs. constrained
 - for constrained:
 - one can directly handle constraints, or
 - one can approximately minimize constraint violations with penalty functions or barrier functions
- Continuous vs. discrete
 - for discrete: often use stochastic methods such a genetic algorithm or simulate annealing

- ESP has a tool named `Miniopt` that performs general non-linear, constrained, continuous optimization
- Uses the Method of Moving Asymptotes (MMA) from `NLopt`
 - introduced by Krister Svanberg in 1987
 - is an iterative scheme that successively approximates the original non-linear constraints and objective function with simpler, convex approximations

- Once you start **Tools**→**Miniopt**, provide (in a semicolon-separated list)
 - the name of the single-valued OUTPMTR whose value should be minimized
 - the name of the (possibly-)multi-valued DESPMTR whose values can be changed to achieve the minimum
 - LBOUNDS and UBOUNDS should be prescribed for these DESPMTRs
 - the name of the (possible-)multi-valued OUTPMTR whose values will be non-positive when the constraints are satisfied
 - (the default semicolon-separated list is `objfun;optvar;constr`)
- Then press **Minimize** to start the optimization
 - this can be done multiple times, always starting from the previous “best” solution

- If you are satisfied with the results, press **Miniopt**→**Save** to save the results and write the `miniopt.despmtrs` file
- If you select **Miniopt**→**Quit**, ESP is returned to the state that it was in before **Miniopt** was selected
- These special parameters influence the way **Miniopt** functions:
 - `optmethod` - set to either `$mma` (the default) or `$slsqp`
 - `optfreq` - frequency at which `miniopt_XXXXX.despmtrs` files are written
 - `optoutlevel` - sets the output level during the optimization (default 0)

Finding Distance Between “Neighboring” Bodys (1)

- For “neighboring” Bodys, use
`EVALUATE DIST ibody1 ibody2`
where `ibody1` and `ibody2` are the two Body (of any type) numbers
- If the two Bodys are disjoint, the returned values are:
 - `@edata[1]` is the minimum distance
 - `@edata[2]` through `@edata[4]` are the entity and parametric coordinates on `ibody1`
 - `@edata[5]` through `@edata[7]` are the *xyz*-coordinates of the closest point on `ibody1`
 - `@edata[8]` through `@edata[10]` are the entity and parametric coordinates on `ibody2`
 - `@edata[11]` through `@edata[13]` are the *xyz*-coordinates of the closest point on `ibody2`



Finding Distance Between “Neighboring” Bodys (2)

- If the two Bodys overlap, the returned values are:
 - `@edata[1]` (a negative value) is maximum intrusion of `ibody2` into `ibody1` (and is computed approximately)
 - `@edata[2]` through `@edata[4]` are the entity and parametric coordinates on `ibody1`
 - `@edata[5]` through `@edata[7]` are the *xyz*-coordinates of the closest point on `ibody1`
 - `@edata[8]` through `@edata[10]` are the entity and parametric coordinates on `ibody2`
 - `@edata[11]` through `@edata[13]` are the *xyz*-coordinates of the closest point on `ibody2`

Finding Distance Between “Enclosing” Body1 and Body2 (1)

- For “neighboring” Bodys, use
`EVALUATE DIST -ibody1 ibody2`
where `ibody1` is the enclosing SolidBody number and `ibody2` is the enclosed (of any type) Body number
- If `ibody2` is completely within `ibody1`
 - `@edata[1]` is the minimum distance
 - `@edata[2]` through `@edata[4]` are the entity and parametric coordinates on `ibody1`
 - `@edata[5]` through `@edata[7]` are the *xyz*-coordinates of the closest point on `ibody1`
 - `@edata[8]` through `@edata[10]` are the entity and parametric coordinates on `ibody2`
 - `@edata[11]` through `@edata[13]` are the *xyz*-coordinates of the closest point on `ibody2`

Finding Distance Between “Enclosing” Body1 and Body2 (2)

- If the two BODYS overlap, the returned values are:
 - @edata[1] (a negative value) is maximum protrusion of ibody2 out of ibody1 (and is computed approximately)
 - @edata[2] through @edata[4] are the entity and parametric coordinates on ibody1
 - @edata[5] through @edata[7] are the *xyz*-coordinates of the closest point on ibody1
 - @edata[8] through @edata[10] are the entity and parametric coordinates on ibody2
 - @edata[11] through @edata[13] are the *xyz*-coordinates of the closest point on ibody2

- If the returned value (in `@edata[1]`) is non-negative:
 - the values returned are on the configuration (and not limited to the tessellation points)
 - the computation is relatively fast
 - fortunately, this situation will dominate in the final optimization steps
- If the returned value (in `@edata[1]`) is negative:
 - only tessellation points on the two Bodies are used in the computation
 - the results are “approximate”
 - this computation is relatively slow
- Bottom line: choosing an initial configuration that satisfies the constraints is always faster

- Overall problem:
 - Put the smallest nacelle that is possible around an engine that has a clearance of at least 0.1 at all points.
- Identify the design variables, objective function, and constraints
- Modify `exercises/session02/nacelle.csm` to include parameters named `optvar`, `objfun`, and `constr`
- Perform the optimization in `serveESP` by using the **Tools**→**Miniopt** option. Once complete, **Miniopt**→**Save** the results.
- Since your initial result probably has concavities, add more constraints to ensure that the final result is convex. This can be done by constraining the design variables to be at least as large as the value predicted by a straight line between its neighbors.