Conditioning 3D Object Based Models

Jeff B. Boisvert

Object based modeling is commonly used for generating facies or rock type models that better reproduce complex realistic geology. A drawback of object based modeling is the difficulty of conditioning to dense data. This is the usual argument for considering training images in multiple point statistical (MPS) simulations based on object based modeling. Object based models have other uses as well, but their use as training images has become very prevalent in MPS workflows; however, if the object could be conditioned to dense data, they would likely be used directly as facies models for complex deposits. The proposed methodology is to consider an object as defined by a set of parameters. Optimization of this object is based on the mismatch with data at the well locations. No gradients are used and any object that can be defined by a set of parameters could be conditioned to well data. Four different optimization schemes are reviewed for conditioning (1) quasi-Newton method (2) Hook Jeeves (3) implicit filtering (4) simplex search method. An example of a fluvial deposit with a constant cross section and occasional crevasse splays is presented. Conditioning is considered on dense data up to 100 wells and performs well. The methodology is not limited to fluvial deposits and can be easily expanded to any object that can be parameterized by a maximum of 100 variables.

Introduction and Background

The detail of object based models (OBM) has progressed significantly (for example, see Pyrcz et al 2012 and the many references listed therein). It is difficult to match the level of geological realism in OBM with statistical techniques such as sequential Indicator simulation, Truncated Gaussian simulation, etc. Often, OBM are generated without consideration of local data (unconditional) and are used in multiple point statistical (MPS) simulation as TIs. If conditioning of the OBM could be considered directly, MPS would not be required for conditioning.

Recent advances in the computational speed of personal computers, paralization with multiple processors and better optimization techniques justifies revisiting the computational difficultly of conditioning OBM. Parallelization is not considered in this work; however, the methodology of conditioning *n* objects when only *l* objects are required (*n*>>*l*) results in perfect parallization up to the use of ~*n* processors.

Descriptions of the four optimization techniques considered follow. Note that the goal of this optimization is to find a local minimum. Finding the global minimum is not required as there would only be one optimum channel and the goal is to generate multiple channels conditioned to well data.

1) **Fminunc: Function in Matlab.**

Fminunc uses a quasi-Newton method with an approximated hessian matrix. Specifically the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is used (Broyden 1970, Fletcher 1970, Goldfarb 1970, Shanno 1970). The formula for updating the inverse hessian is shown in equation 1.

$$\mathbf{B_{k+1}} = \mathbf{B_k} + \left(\frac{1 + \mathbf{q_k}^{\mathrm{T}}\mathbf{B_k}\mathbf{q_k}}{\mathbf{q_k}^{\mathrm{T}}\mathbf{p_k}} \right) \frac{\mathbf{p_k}\mathbf{p_k}^{\mathrm{T}}}{\mathbf{p_k}^{\mathrm{T}}\mathbf{q_k}} - \frac{\mathbf{p_k}\mathbf{q_k}^{\mathrm{T}}\mathbf{B_k} + \mathbf{B_k}\mathbf{q_k}\mathbf{p_k}^{\mathrm{T}}}{\mathbf{q_k}^{\mathrm{T}}\mathbf{p_k}} \qquad (1)$$

2) **Hooke-Jeeves optimization**, implemented by Kelley (1998).

The objective value is assessed in different directions for different input variables. The direction with the minimum objective value is approximated, and a step is taken. The procedure is repeated until a user defined stopping criteria. The interested reader is referred to Hooke and Jeeves (1961) for additional details on the algorithm.

3) **Implicit filtering:** Minimization of noisy functions subject to explicit bound constraints.

Implicit filtering is the only technique that includes constraints on the input variables. The algorithm uses coordinate searches as well as gradient approximation to generate parameters with a lower objective function value. More details can be found in Kelley (2011).

4) **Simplex search method.**

The Matlab function, fminsearch, implements Lagarias (1998) version of the simplex algorithm.

Hereafter, the optimization techniques will be referred to as Option 1, 2, 3 and 4. Note that none of the techniques require the specification of a gradient. Any type of object can be optimized with these techniques so long as the objective function (discussed below) can be calculated. This amounts to being able to determine the well intersection of all different facies in an object. Note that most of the implementations of these techniques suggest that the practical maximum number of variables that can be used for object parameterization is 100.

The following methodology assumes that the wells have been coded into gross facies categories. Small, inconsistent facies definitions are unlikely to be conditioned properly.

Methodology

The proposed methodology is to generate a large number of objects that are consistent with the well data. Once this library is complete, realizations will be generated by selecting an appropriate number of objects from the library. This is the same methodology presented in CCG paper 115, 2011 for 2D objects. Overall the steps are:

1) Generate an algorithm that constructs an object from a set of parameters OBJ(x_1, x_2, …, x_p) where p<100. The practical restriction on the number of parameters is set by the algorithms implemented.
2) Define the objective function to minimize (discussed below, but this is essentially a summation of the well-data mismatch).
3) Start with an initial set of parameters for the object.
4) Apply one of the algorithms discussed above. The only inputs are the object generating function (step 3) and the objective value calculation (step 2).
5) The result of step 4 is a conditioned object. Add this object to the library of objects.
6) Repeat steps 3-5, *n* times.
7) Add *l* objects from the library of *n* objects to constitute a realization. Add objects until local proportions and all conditioning data are honored. Note, this step will not be considered in this work and is the subject of future work.

Steps 1 (object construction) and 2 (the objective function) are discussed. Each step is demonstrated for an example below. Steps 5 and 6 are straightforward and not discussed. Step 7 will be considered in future work.

Step 1: Object Construction

As discussed above, all of the algorithms can be applied to any object that can be parameterized. For the purposes of demonstrating the methodology, a parameterization of a channel with crevasse splays will be presented here. It must be stressed that the focus of this work is not to generate an OBM generation methodology for fluvial deposits; the goal is to condition those objects. Any object definition could be used.

Channels will be represented by a parabola (height=width^2) (Figure 1) extruded along a center line. Any channel cross section could be used, but this will simplify the example. Channel center lines are defined by a cubic spline fit to a number of initial control points (cp_i) (x,y). The elevation of the channel is considered constant as it is assumed that modeling is being conducted in depositional space, however, the extension to considering channels with variable heights would be trivial and is accomplished by considering x, y and z coordinates for the initial control points (note that this was originally implemented in the example, but constant elevation was deemed more appropriate).

The channel is assumed to have a constant width and constant thickness as the width controls channel thickness, height=width^2.

Crevasse splays are added to locations along the channel length at locations where the derivative of the center line=0. The crevasse splays are added if there is a well indicating a crevasse splay near to this inflection point. The radius of the crevasse splay is constant at 10 units and is assumed to have a constant thickness. Two parameters define a crevasse splay, (1) length along the center line to add a crevasse splay and (2) the thickness of the crevasse splay.

The total number of variables that are required to define a channel of this type is cp_i*2+5, the 5 additional variables include: thickness, x_{shift}, y_{shift}, z_{shift}, CS_{thick} and CS_{loc}. Note that the shift parameters shift the entire channel in the x, y, or z directions. These parameters are redundant with the control points along the channel, but it has been found to help the optimization algorithms if the flexibility for moving the entire channel is included.

Step 2: Objective Function

The basic objective function deals with matching the known well data. This is a mismatch between the current channel object and the well data. The same objective function is used as described in CCG paper 115, 2011:

$$f(o,v) = \sum_{i=1}^{n} d(o,v_i) \qquad (2)$$

where $d(o,v_i)$ is a function that returns the distance between the object (*o*) at its current location and intersection *i* in the conditioning data (v_i). Possible well intersections showing the distances *d* are illustrated in Figure 1.

This aspect of the objective function controls the level of data conditioning mismatch. There are other general considerations for placing objects. Typically, some benefit should be given to wells that match larger sections of well conditioning data. The second part of the objective function rewards objects that match more conditioning data:

$$g(o,v) = -\sum_{i=1}^{n} w(o,v_i) \qquad (3)$$

where $w(o,v_i)$ is the length of matching conditioning data that is within an object (Figure 1).

The final aspect of the objective function is a penalty to objects that are not intersected by a well, but come very close to the well. A common bias for conditioning objects is that the object is moved just outside the well data but is very close (Case f in Figure 1). There is a clear bias for placing objects very close to wells where the conditioning data is not honored. This aspect of the objective function penalizes objects placed very close to conditioning data:

$$h(o,v) = r_w \sum_{i=1}^{n} \left(1 - \frac{b(o,v_i)}{c_b}\right) \qquad \text{if } b(o,v_i) < c_b \qquad (3)$$

where $b(o,v_i)$ is the distance from the object to the well (Case f, Figure 1) and c_b is a user defined constant that controls how close to a well an object can be before being penalized. This would apply a uniform penalty to all wells for having an object within a distance of c_b; this could also introduce a bias as wells would tend to be exactly c_b away from conditioning data. Each well is assigned a random weight that varies the penalty for each well, r_w.

The final objective function is then a weighted combination of the three aspects, mismatch $c_f \cdot f(o,v)$, conditioning $c_g \cdot g(o,v)$ and closeness $c_h \cdot \boldsymbol{h}(o,v)$:

$$obj(o,v) = c_f \cdot f(o,v) + c_g \cdot g(o,v) + c_h \cdot \boldsymbol{h}(o,v) \qquad (4)$$

where c_f, c_g and c_h are weights for determining the level of importance to assign to each aspect of the objective function. The parameters specified by the user are the weights to use for each aspect: c_b, c_f, c_g and c_h. The effects these weights have on the optimization are discussed in the example below.

Step 3: Initial Object Generation

The initial state for object generation should ensure that the initial object is valid. If there are certain combinations of parameters that generate objects that are not realistic, the initial state should reflect this. The initial state for the channel example below is generated randomly. Note that option 3 is the only optimization implementation that provides bounds on the object parameters and can be used if there are ranges of values for which parameters are invalid, otherwise no constraints on the parameters are considered.

Step 4: Optimization algorithm: The different options for optimization were discussed above.

Example

The following example considers the optimization of multiple channels with different levels of conditioning data. Control points are created by generating an azimuth line (s_i) for the object and placing control points at a random distance from the line (Figure 2). For this example $s_i \in [45°, 135°]$ passing through the center of the domain (25,25).

To demonstrate the methodology, a small number of conditioning data (7) will be used. Consider Figure 3 where the optimized channels are shown in plan view as well as cross sections through each well. Each conditioned channel is shown on the cross section, perpendicular to its own centerline, with all channels arranged to the left of the well. Note that this is not a true cross section through the well and is only intended to highlight the conditioning of the algorithm. The restriction on the starting configuration of the wells to pass through (25,25) and have initial centerline azimuths $s_i \in [45°, 135°]$, cause the clustering of channels in Figure 3. Relaxing this initial starting condition would allow for more channels that would condition wells 2 and 6. Note that well 3 is the only well with a crevasse splay, and is shown as a rectangle in cross section.

It is difficult to show the conditioning for a large number of wells, Figure 4 shows 10 channels conditioned to 47 wells. Similar results are seen for a larger number of wells (100+). Wells 41-47 contain no channel facies to highlight the issue of generating conditioned objects that are placed very close to non-channel facies. This is most apparent for wells 43 and 44 where there are a number of channels nearly intersecting the non-channel facies. Weights for the objective function for channels in Figure 4 are $c_f = 1.0$, $c_g = 0.25$ and $c_h = 0.0$. To correct for the closeness of channels to wells 41, 43, 44 and 46, a weight of $c_h = 0.3$ was used (Figure 4 below). This effectively penalizes wells that are close to, but not intersecting conditioning data of a different facies. With weight given to c_h there is only one channel that is unusually close to wells 41 through 47 (Figure 4 below).

Including additional aspects in the objective function beyond just matching conditioning data has the effect of increasing the well/channel mismatch. A balance must be struck between the three components of the objective function. Alternatively, the effect of c_h could be ignored and channels that violate this constraint could be rejected in Step 7 of the methodology where the final OBM realization is created.

Thus far, the example has considered optimizing with option 2 (Hook- Jeeves) without consideration for CPU time. Figure 5 compares the generation of objects based on time as well as based on the ability of the optimization algorithms to condition to well data (setting c_g to 0 and 0.25 for comparison). Option 3 performs quite poorly in regard to the average mismatch between the conditioning data and the well data (top row of plots in Figure 5), option 3 will not be considered further. The comparison of the other three options with no weight is given to the conditioning are very similar in performance; in regards to time and conditioning ability there is little difference. There is some difference in the performance of the algorithms when weight is given to the conditioning data, $g(o, v\quad)$; in this case option 1 seems to better match the constraints of the well data (top right plot on Figure 5) however, it is not able to obtain the same level of well conditioning as options 2 and 4. Option 4 performs nearly as well as option 2 for well conditioning (top and bottom right plots in Figure 5), but takes considerably longer for the optimization.

Going forward, option 2 will be implemented in a FORTRAN type program that will be used to test the conditioning of more complex OBM techniques. It is expected that there will be significant CPU speed increases with the FORTRAN implementation but more complex objects may be more difficult.

Conclusion

The proposed objective function is simplistic but captures many of the aspects important for placing objects. Current optimization techniques are able to generate objects efficiently (order of seconds) and many objects that match conditioning data can be generated. Dense data slightly increases CPU time, but the proposed algorithms are effective for 100+ wells. Previous views have been that conditioning objects with dense data is not computationally efficient; this was not found to be the case for the object tested here. The examples tested thus far have been fairly simplistic. Channels are generated that honor one intersection in the well data. Future work will consider wells with multiple intersections and objects with more complex facies relationships.

References

Chiles, J.-P., Delfiner, P., 1999. Geostatistics: modeling spatial uncertainty. John Wiley & Sons, Inc., 720pp.

Broyden, C.G., 1970. The Convergence of a Class of Double-Rank Minimization Algorithms, Journal Inst. Math. Applic.,6, 76-90.

Fletcher, R., 1970 A New Approach to Variable Metric Algorithms, Computer Journal, 13, 317-322.

Goldfarb, D., 1970 A Family of Variable Metric Updates Derived by Variational Means, Mathematics of Computing, 24, 23-26.

Hooke and Jeeves 1961. Direct search: solution of numerical and statistical problems. *Journal of the Association for Computing Machinery (ACM)* **8** (2): 212–229

Kelley, C (1998). Hooke-Jeeves optimization. Accessed Aug 15, 2012. http://www4.ncsu.edu/~ctk/darts/hooke.m

Kelley, C (2007). Minimization of noisy functions subject to explicit bound constraints. Accessed Aug 15, 2012. http://www4.ncsu.edu/~ctk/MATLAB_IMFIL_v1/imfil.m

Kelley, C (2011). Implicit Filtering. SIAM. 184.

Lagarias, Reeds, Wright and Wright, 1998, Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, SIAM Journal of Optimization, 9(1):112-147

Pyrcz, McHargue, Clark, Sullivan, Strebelle, 2012, Event-based geostatistical modeling: Description and applications. In Abrahamsen et al. (eds) Geostatistics Oslo 2012. 27-38.

Shanno, D.F., 1970 Conditioning of Quasi-Newton Methods for Function Minimization, Mathematics of Computing, 24, 647-656.

Figures

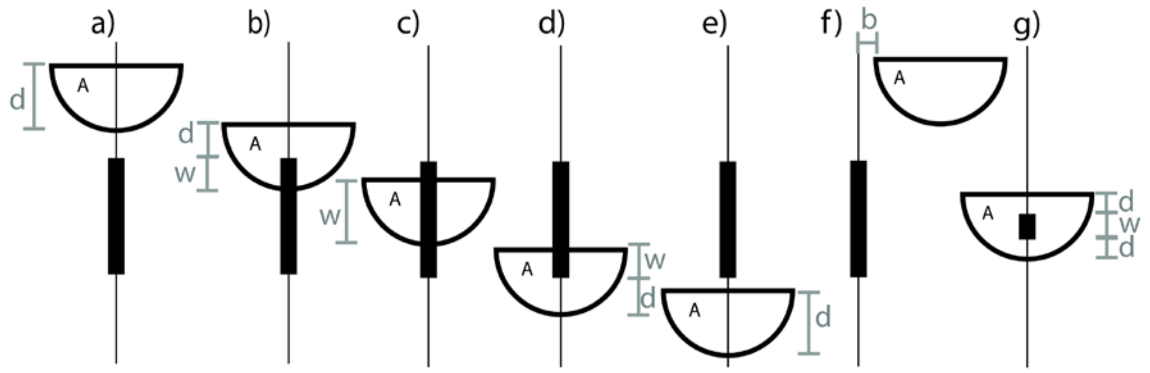


Figure 1: Distances used for calculation of the objective function.

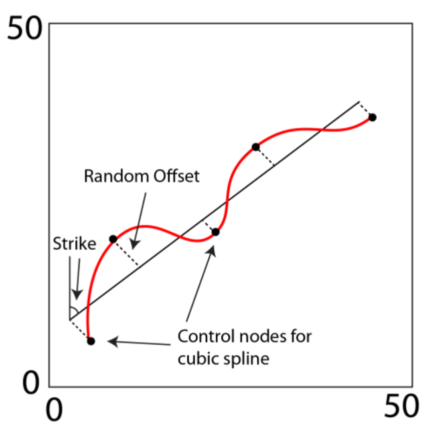


Figure 2: Initial centerline construction. Initial control points for the cubic spline are generated as offsets from a random strike. Plan view of one channel.

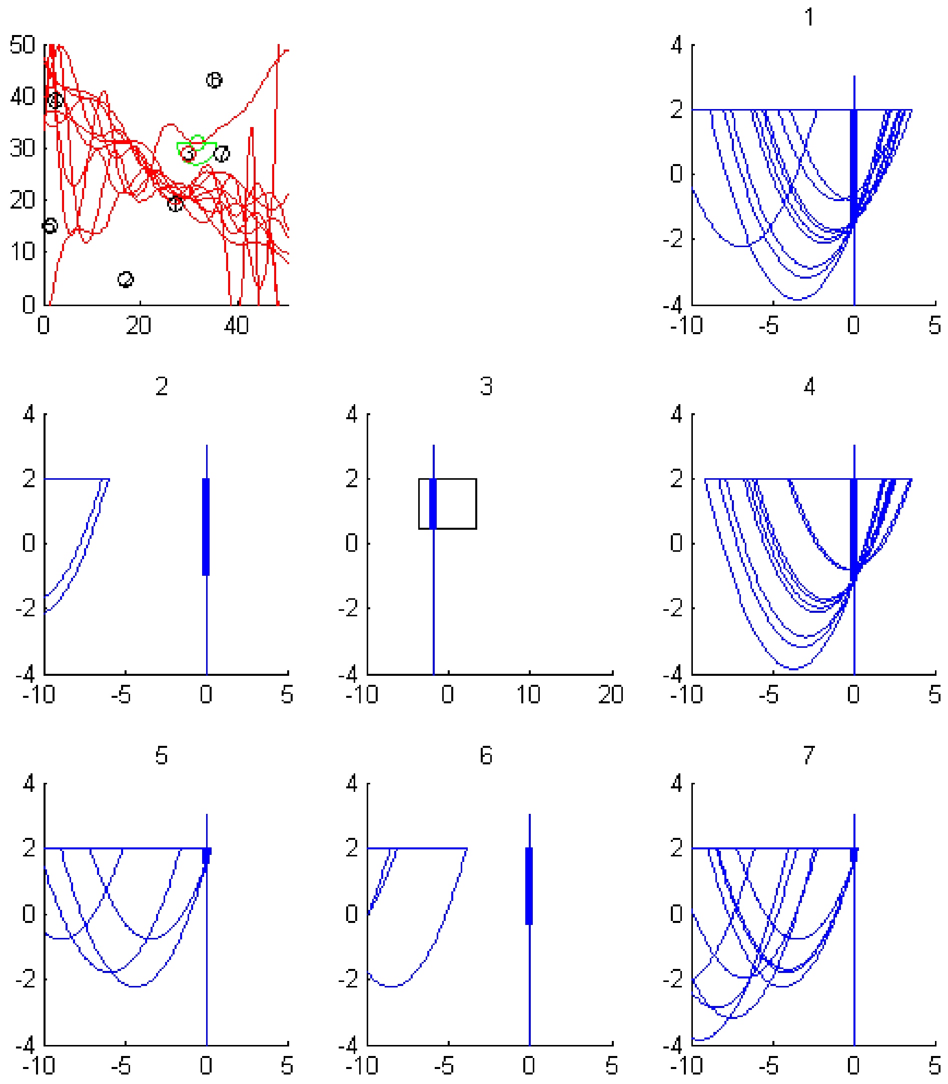
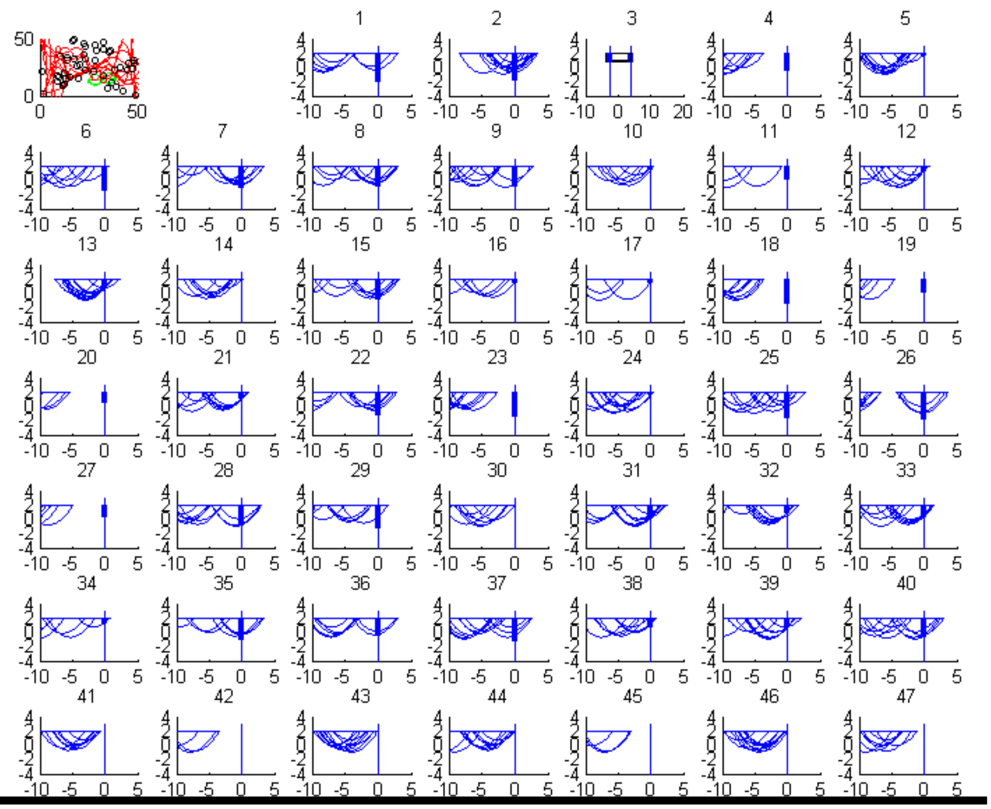


Figure 3: Conditioning to 7 wells. Upper left plot shows the plan view of the channel centerlines as well as the crevasse splay near well 3. Vertical axis is depth (units) and the horizontal axis is set perpendicular to the channel centerline.



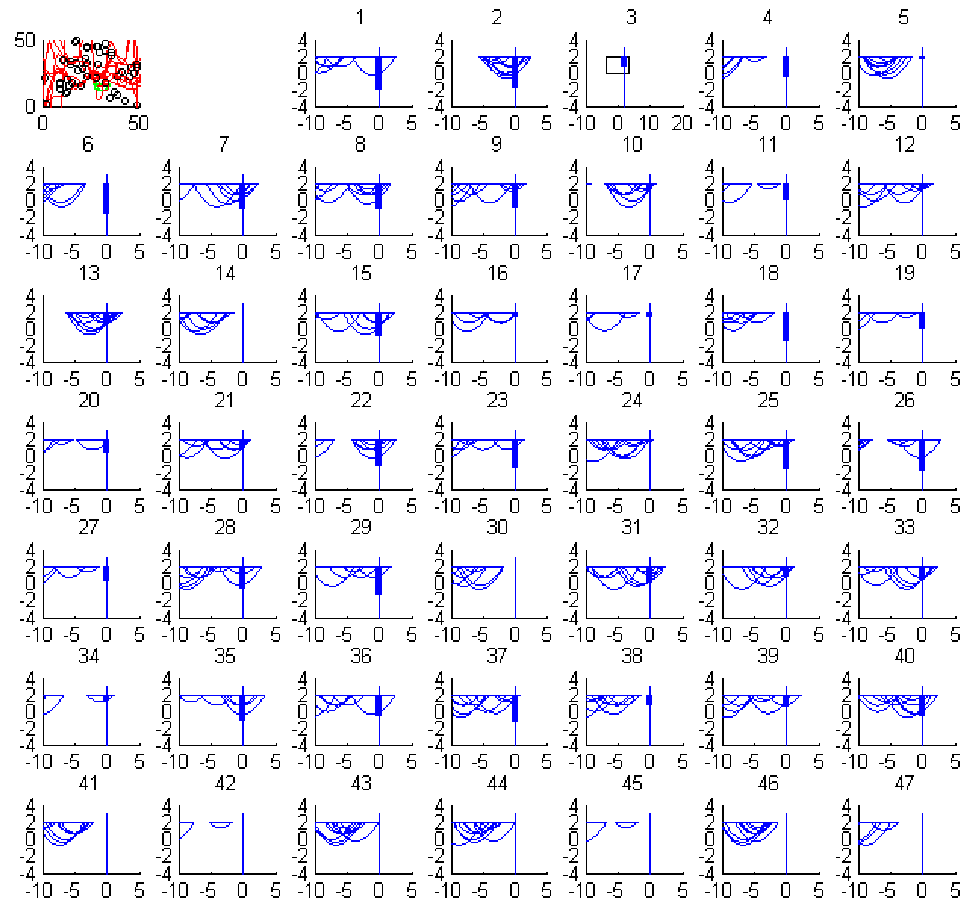


Figure 4 (previous page): Conditioning 10 objects for 47 wells. Upper left plot shows the plan view of the channel centerlines as well as the crevasse splay near well 3. Wells 41-47 do not have any channel facies present. Plots above, a weight of 0.25 is given to conditioning data in the objective function resulting in an average mismatch between channel and well data of 0.1 units for the objects generated. Using a weight of 0 for conditioning data results in a mismatch of 0 units. Note that only well 3 contains a crevasse splay intersection, all other wells have channel intersections. Vertical axis is depth (units) and horizontal axis is set perpendicular to the channel centerline. Plots below consider a weight of 0.3 for the buffer aspect of the objective function.

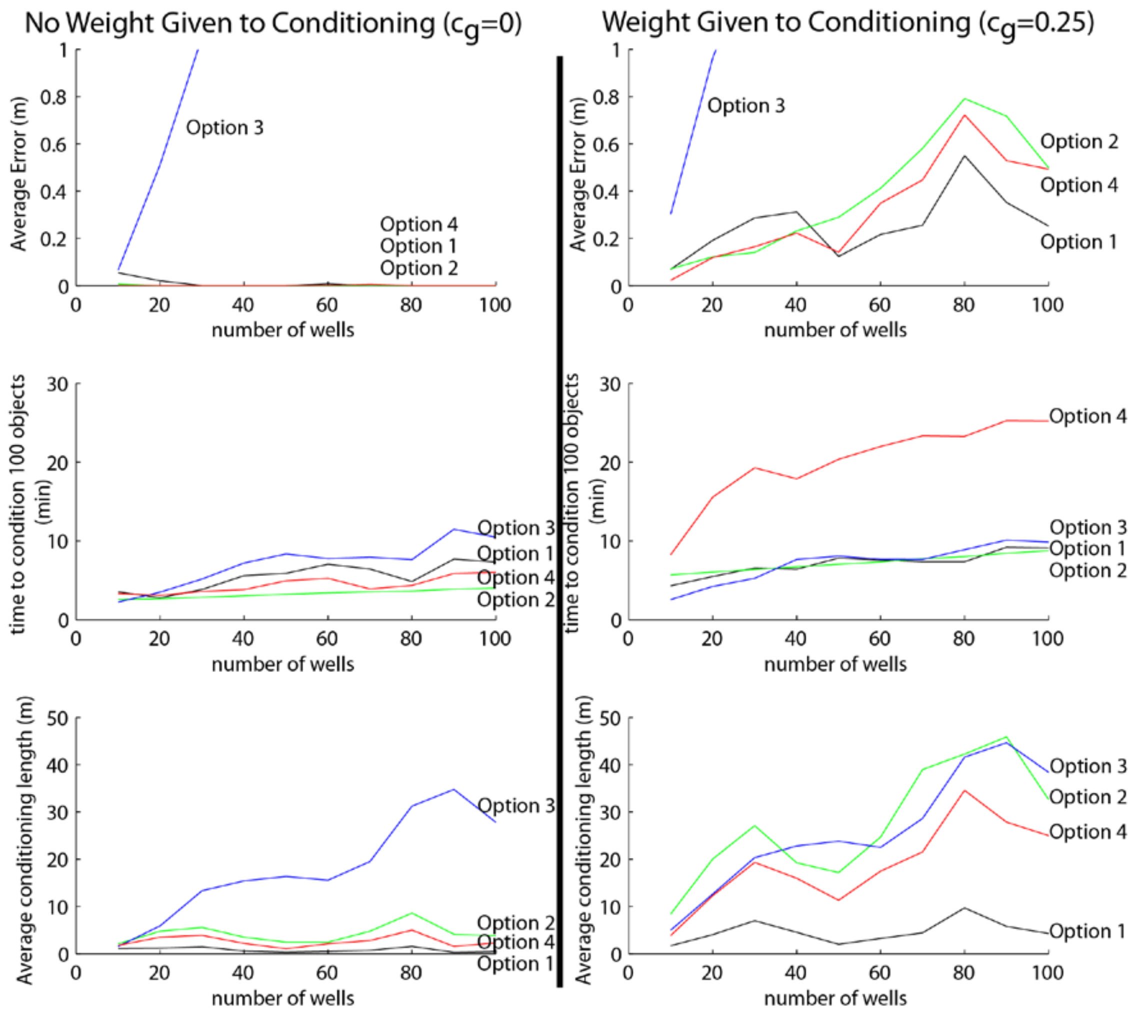


Figure 5: Time trials for conditioning objects.