We thank the reviewers for their effort and their very detailed reviews.

Reviewer 1

Q: Comparison to probabilistic programming languages.
A: You are right, PPLs also allow flexible modeling and problem solving. We will add a review/comparison to PPLs and discuss the differences to our approach in the related work section.
Q: Why is GENO so much better than other solvers.
A: Other general purpose approaches/solvers (CVXPY) need to transform any problem instance into some standard form like an LP, QP, or SDP in standard from. The transformation increases in the problem size in terms of optimization variables and/or constraints. The transformed problems are typically addressed by Newton-type solvers that are very general but do not scale with the problem dimension.
Q: Running times, what is reported? Table 2 / comparison to CVXPY?
A: We will make this more clear. The reported timings include only the running times of the individual solvers. Generating the solvers by GENO takes only a few milliseconds and is not included into the timings. Anyway, the solver has to be generated only once. CVXPY was way to slow to be included with the other experiments in Figure 2. Hence, we ran a few much smaller problems for CVXPY and present the results in Table 2.
Q: Distinction between well-engineered and recent state-of-the-art solvers.
A: We consider solvers like LIBLINEAR and LIBSVM that are maintained for more than ten years well-engineered. Such solvers often outperform recently published solvers that implement new algorithmic ideas or adapt to some problem structure. We refer to the latter as recent state-of-the-art solvers.
Q: GENO does not transform a problem but a whole problem class?
A: In a problem class, parameters like a data matrix remain abstract while they are concrete (numerical values) in any problem instance. Traditional modeling languages take a problem instance and transform it into a problem instance of either a standard LP, QP, or SDP. Essentially, GENO takes a problem class that is specified by an objective function and constraints and transforms it into another problem class with smooth objective function but without constraints.
Q: Workflow example.
A: Initially, we had a workflow example but removed it due to space limitations. We will add it again to the appendix.

Reviewer 2

Q: Special structure in the problem.
A: The solver can exploit some special structure in the data. One can specify that a matrix is symmetric and/or sparse.
Q: Source code availability and license (also asked by Reviewer 3).
A: There is a link on the bottom of the anonymized GENO webpage that points to the github repository that contains the anonymized source code along with installation instructions. It seems, the reviewer has missed this link. We plan to make the original code available via github under the GPLv3 license.

Reviewer 3

Q: Limited practical impact. Highly efficient solvers have existed for years.
A: The purpose of GENO is to provide highly efficient solvers for new problems. Actually, our work on GENO is motivated from our repeated experience that there was no efficient readily available solver for some model that we were considering. Hence, we wanted GENO to be flexible but at the same time the generated code to be highly efficient. Only to prove our point that the generated code is indeed efficient we compared it to solvers that have existed for years. We also compared GENO to recently published solvers (NeurIPS 2018 and ICML 2019) for special problems (non-linear least-squares, compressed sensing). GENO outperforms these specialized solvers.
Q: Exploit convex duality for efficiency for efficiency.
A: Convex duality can only be exploited for convex problems. For this, the problem needs to be transformed into a standard form. The solvers Gurobi and Mosek exploit convex duality. However, they are a few orders of magnitude slower than our approach. It is usually believed that in order to be very efficient one needs a specialized solver. We challenged this claim by showing that one approach can be as efficient as specialized, well-established solvers.
Q: Deep learning problems out of scope.
A: As detailed in the introduction, there are a number of well-established, efficient deep learning frameworks. Such a framework was missing for classical ML. We want to fill this gap with GENO.
Q: GENO seems to be more convenient than, e.g., CVXPY (problem dimensions need not be specified).
A: This is not only a matter of convenience. A key aspect of GENO is that it generates a new solver for every problem class, while existing approaches like CVXPY transform a problem instance (where the dimensions and values are specified) into some standard form. This fundamentally different approach leads to an increase in efficiency of a few orders of magnitude.