We thank the reviewers for their time and effort. Below, we address the issues raised and clarify some misconceptions. We begin with two key issues mentioned by more than one reviewer in order to avoid repetition.

Technical contribution. It is indeed true that the broad existing literature on dealing with non-monotonicity by introducing randomness creates the impression that subsampling works almost like a black-box reduction. Starting with [6], these ideas have been used extensively for several constraints and in every possible variant of the problem. Yet, **SampleGreedy is the first random greedy result for a knapsack constraint**. During a seven-year period, there is not a single paper that manages to apply subsampling also for a knapsack constraint, despite the latter being as prevalent in the submodular literature as cardinality or matroid constraints. This fact illustrates perfectly the underlying technical difficulties: while the algorithmic idea itself is immediate, obtaining our results is not. As we mention in Section 1.1 (lines 88-100), the tools needed to show our results differ substantially from other works, which *all* rely on the algorithm making good progress in a step-wise fashion. This being impossible for knapsack constraints, our main technical contribution is the *global* analysis via a comparison with a fuzzy set of high value. So, while employing known *principles*, our work shares only few technical steps with other related work. This is also apparent when dealing with the adaptive setting's subtleties. E.g., monotonicity is crucial in [24] and this cannot be addressed by a naive application of subsampling. Indeed, the whole point of [25] is to make the 'subsampling lemma' work for the case of a uniform matroid. While we share the first step of our analysis with [25], their (somewhat more complex) sampling procedure and its analysis cannot be extended to the knapsack case in any sensible way.

Quality of approximation. Our clear priority is keeping the running time almost linear *and* practical; this is challenging even for the monotone case. The fact is that **SampleGreedy is the first constant factor approximation algorithm asking $O(n \log n)$ queries** for the non-monotone version of the problem. For this reason, using more elaborate building blocks requiring $\Omega(n^2)$ queries in order to improve the approximation, was not really of interest here. (Actually, by enumerating, like in [43], our approximation factor improves to $\frac{2e}{e-1} \approx 3.16$ with $O(n^3 \log n)$ queries.)

Review 1. a) Please see our remarks above. b) Thank you for bringing [YZA] to our attention. [FKK,AEFNS] are only implicitly related (streaming setting and no knapsack constraints); nevertheless we will add all three to our related work. c) Although this could be interesting for future work, it is not immediately clear why the analysis of [YZA] would carry on without too much loss and yield a significantly better factor if we use their algorithm instead. Even if that was the case, it is *impossible* to translate this to the adaptive setting, where we commit to all past choices (see also lines 33-35, 97-100). d) We do mention the state-of-the-art for the monotone case [43,16]. We assume the reviewer refers to the streaming setting, which we do not discuss in detail as it is not directly relevant.

Review 2. a) Please see our remarks above. b) The algorithm in lines 52-55 is a minor variant of the Z^H solution of [45]. However, the reviewer is right about the approximation ratio: the 2 should be 2.8 as follows from [45]. c) The value of the adaptive objective depends only on the state of the elements in the evaluated set (lines 164-165). We can discuss this earlier on as well. Note, however, that we work on the *existing* framework of [24,25] without simplifying it. d) Once a value of p is selected, the variance is very small for large instances. Running the experiment 5 times (which is still very fast) produces slightly better results primarily by 'guessing' a good value of p (see also lines 53-54 herein). So, even in the adaptive case, this could be seen as a dry run to tune p before trying on actual data. For completeness, we can add the expectation/variance results for a single run. e) In practice, lazy evaluations often result in much less than $\log n$ evaluations per element. So, indeed, what we see is much closer to n vs $n \log n$. f) We will add the results for Greedy. They were omitted to highlight that even after 5 runs, SampleGreedy has a much better performance per number of queries ratio to Fantom. The performance of Greedy sits between the two. g) In our conclusions we refer to improving streaming algorithms for knapsack constraints, e.g., [39], whereas [FKK] deals with p-matchoids.

Review 3. a) Please see our remarks above. b) We thank the reviewer for bringing the Ene et al. paper [ENV] to our attention. It should be noted, though, that papers in this line of work do not necessarily end up having practical query complexity. Here [ENV] assumes access to oracles for the multilinear extension of the objective and its gradient. This is not realistic in practice (where a polytime oracle for the objective is often straightforward) but is generally dismissed as it can be resolved via sampling. When it is done explicitly though, it requires a polynomial number of samples for each evaluation; typically $\Omega(n^3)$ (e.g., in the seminal work of Vondrák 'Optimal approximation for the submodular welfare problem in the value oracle model' n^5 samples are used per evaluation). In the case of [ENV] this results in at least $n^4 \log^2 n$ oracle queries in the worst case. c) While we agree, there is not much flexibility about the name; 'adaptive submodular maximization' was established even before 'adaptive complexity of submodular maximization'.

Review 4. a) Please see our remarks above. b) The choice of p reflects the gap between the worst case in theory and what happens in practice: the best singleton is expected to be small and non-monotone objectives are not expected to be—loosely speaking—arbitrarily far from monotonicity. If one was to parameterize, not only by $\max_i v(i)$ (Thm. 3), but also by $\min_{i,T} v(i \,|\, T)$, then it would be clear that as the former decreases and the latter increases, the optimal value of p gets closer to 1. While it is easy to construct *synthetic* instances where $p = 0.41$ and SampleGreedy performs better than FANTOM and *arbitrarily* better than Greedy, we felt that this would be a biased comparison (in our favor).