We want to thank the reviewers for their thorough comments and constructive suggestions! They highlighted the importance of the formulated problem (**R1**, **R2**, **R3**) and the relevance of the submitted manuscript to the community (**R1**). Also, reviewers noticed that the proposed method is simple (**R3**), the error bound on the gradient approximation is derived (**R1**, **R4**), and the code is provided (**R3**). All reviewers commend the clear explanation but noted the language issues and typos in formulas (**R4**). We respond to the reviewers' comments below and update the final version of the manuscript.

[R2, R4] Concerns about theoretical error bounds. The error bound derivation was included to demonstrate which factors affect the quality of gradient approximation. Namely, *logarithmic norm* of the right-hand side Jacobian affects the approximation quality and can be further used as a regularizer when training neural ODE models.

[**R1**, **R3**] **Improvement in stability of gradient computations.** We perform additional experiments on reconstruction trajectory of dynamical system that collapses in zero. As a result, we observe that reverse dynamic method (RDM) and our method (IRDM) solve this problem with approximately the same accuracy. However, RDM requires at least 10 times more right-hand side evaluations to solve adjoint IVP in every iteration than IRDM. Thus, IRDM and RDM compute similar gradients, but IRDM computes them much faster.

[**R1**] **Datasets considered for the density estimation task are only synthetic.** We present the performance of our method for the density estimation problem on the real data `miniboone` dataset (Figure 2).

[R2] How do the authors plan to improve the manuscript with respect to weakness 1? We use a 4th order scheme proposed in "Some Practical Runge-Kutta Formulas" by Shampine (1986). It combines the basic Dormand-Prince Runge-Kutta process with their 5th-order formula, modified 4th-order formula, and 4th-order formula for the midpoint along with local quartic Hermite interpolation. Theoretical results in "Interpolation for Runge-Kutta methods" by Shampine (1985) yield that we get 4th-order approximations at pre-defined grid points. If we understand the terminology right, we already use a smooth-interpolant RK formula. A useful paper by Calvo et. al., recommended by **R2**, can be considered as a generalization of a 5th order scheme, proposed in the above paper by Shampine. We will include a detailed description related to this issue and links to the mentioned papers in the final version.

Yes, a scheme in which the solvers are forced to include the predefined grid points as part of the otherwise adaptive mesh, can be applied. However, such a scheme repeatedly increases the size of the optimal step size, which ruins the control of optimal step size, and it can lead to a high increase in computational time and rounding error. We will add an experiment to the final version to demonstrate our method's time savings compared to this approach.

[R2] Comparison with ANODE. ANODE exploits the checkpointing technique in the backward pass (see Figure 1d). In the official repository, we found the implementation that exploits a standard automatic differentiation rather than checkpointing. Therefore, we did not compare with ANODE since the out of memory error was raised during training.

[R2] I would like to see an evaluation of gradient accuracy with respect to grid points and tolerance. We carried out additional experiments to estimate the requested dependencies and conclude that the larger (absolute and relative) tolerances lead to larger error in gradient estimation. Also, the dependence of gradient accuracy w.r.t. number of grid points is non-monotonic, i.e. too small and too large number of points in Chebyshev grid leads to degrading of gradient accuracy. These results will be added to the final version of the manuscript.

[R2] Will the code be released upon acceptance? Sure, the code will be released.

[R2] Barycentric Lagrange interpolation (BLI) should be contrasted with other interpolation techniques. Indeed, other interpolation techniques can be used in the proposed method. We tested piecewise-smooth interpolants (such as piecewise-linear and cubic splines) and found that they provide worse performance than BLI. We will add these experiments to the final version.

[R3] The time improvement is very marginal with N times more memory usage. As we showed in experiments, the values of N are typically small and additional memory is not an issue since NODE has a few parameters. Also, we respectfully disagree that improvement is very marginal.

[R3] The interpolation acts as an approximation for $z(T)$. Is it possible to directly backpropagate through this approximated $z(T)$ to obtain gradients with $\mathcal{O}(N)$ memory? No, it is not possible since $z(T)$ is computed with an ODE solver without exploiting the interpolation technique.

[R4] The boundedness condition on $\frac{\partial f}{\partial \theta}$ and $\frac{\partial^2 f}{\partial \theta^2}$ is very strong. Can this even be verified in the considered applications? In the considered applications, the norms of $\frac{\partial f}{\partial \theta}$ and $\frac{\partial^2 f}{\partial \theta^2}$ can be computed during training. These norms roughly estimate the bounds of gradient and hessian of the right-hand side w.r.t. parameters. The potential approach to ensure the mentioned boundedness is to use proper normalization layers in the right-hand side f.