Running a problem on a sparse grid and interpolating onto a fine grid

We consider the optimal control problem:

$$\mathcal{J}(\rho, \mathbf{w}) = \frac{1}{2}||\rho - \widehat{\rho}||^2_{L_2(\Sigma)} + \frac{\beta}{2}||\mathbf{w}||^2_{L_2(\Sigma)}$$

subject to:

$$\frac{\partial \rho}{\partial t} = \nabla^2 \rho - \nabla(\rho \mathbf{w}) + \kappa \nabla \int_\Omega \rho(r)\rho(r')\mathbf{K}(r, r')dr' + f.$$

The question of interest is to investigate whether it is of benefit to run a given problem on a sparse grid first to a certain optimality tolerance and then interpolate the solution onto a finer grid. The idea is that while the sparser grid may not be sufficient for a very accurate result, it will bring us closer to a good initial guess for the fine grid. This would mean that we need to compute less iterations on the fine grid than if we started with a less accurate initial guess. Since finer grids are more computationally expensive, this would mean a saving in computational effort and time.

1 Exact Problem

At first we consider the Neumann flow control problem with an analytic solution on the standard box. We then take the exact $\mathbf{w}$ and perturb it in time by a magnitude of 0.1, using the perturbation function described in the paper. This function will not perturb any boundary in space or time and is smooth. The expectation is that the optimization solver will find the exact $\mathbf{w}$ after a few iterations, within the optimality tolerance.
In order to compare the numerical scheme to a base case, we first compute this on the fine grid only, without using any of the described procedure. We choose the tolerances to be $10^{-7}/10^{-3}$ and the number of points is $n = 20$, and $N = 40$. This is solved within 358 iterations, which takes 5.2829×10^3 seconds. The final error in $\mathbf{w}$, compared against the exact solution, is 2.6611×10^{-4}.

In a next step we use the algorithm described above. For the sparse grid we choose $N = 22$ and n remains at 20. We then set the optimality tolerances of the sparse grid to different choices and give the resulting solution as an initial guess for the fine grid, which has the same choices of parameters as above.
The first chosen tolerances for the sparse grid are $10^{-7}/1$. This is solved in 1 iteration and 1.7913 seconds. The resulting error in $\mathbf{w}$ is 0.0388, which is a considerable decrease from the initial error of 0.1 we caused by perturbing the exact solution. After interpolating this onto the fine grid the error remains the same. Solving the problem on the fine grid now takes 285

iterations and 4.1477×10^3 seconds. The final error in $\mathbf{w}$ is 2.8313×10^{-4}. We can conclude a gain in time, while getting the same accuracy in the result.
Next we run the sparse grid to an optimality tolerance of 10^{-1}. With this configuration, we solve the problem on the sparse grid in 20 iterations, which takes 34.5687 seconds. The error of the resulting $\mathbf{w}$, compared to the exact $\mathbf{w}$ is 0.0299. This resulting $\mathbf{w}$ is then used as an initial guess for the fine grid problem. The interpolated $\mathbf{w}$ has still an error of 0.0299, which means no accuracy is lost at the interpolation step. The fine grid problem converges within 266 iterations, which takes 3.6053×10^3 seconds. The final error in $\mathbf{w}$ is 2.8532×10^{-4}. Again, we can see an improvement in time and iterations.
When letting the sparse grid run up to 10^{-2} optimality tolerance, this takes 190 iterations and 268.3850 seconds. The final error in $\mathbf{w}$ is 0.0028. This doesn't change after $\mathbf{w}$ is interpolated onto the fine grid. Then on the fine grid, the problem takes 98 iterations to be solved and takes 1.2275×10^3 seconds. The final error is 2.8422×10^{-4}. Again, this is an improvement in time, while getting the same accuracy for $\mathbf{w}$.

Finally, letting the sparse grid run all the way to an optimality tolerance of 10^{-3}, which is the desired tolerance, this converges in 357 iterations and 500.4822 seconds. The error in $\mathbf{w}$ is 2.7060×10^{-4}, which is unchanged by the interpolation step. The fine grid now only takes one iteration and 13.2093 seconds to converge. The final error in $\mathbf{w}$ is 1.0792×10^{-4}. This seems to be the best strategy of the ones above in terms of time savings.

2 Example Problem 1

The only difference to the previous example is that we are choosing a problem without analytic solution. The initial configuration for this example is:

$$\begin{aligned} \rho_0 &= \exp(-2((y_1 - 0.5)^2 + (y_2 + 0.5)^2)) \\ \mathbf{w} &= \mathbf{0}. \end{aligned}$$

First we consider the fine only results, to have a baseline to compare our algorithm to. All parameter choices are as above. Solving this problem takes 413 iterations and 1.3926×10^4 seconds.
Now we are looking at running the problem on a sparse grid first, up to a given optimality tolerance. Then we pass the result to the fine grid and run it to the final optimality tolerance of 10^{-3}. At first the sparse grid is running only up to an optimality tolerance of 1. This takes 23 iterations and 77.0391 seconds. Then the solution is passed to the fine grid and solving the problem to 10^{-3} accuracy there takes 318 iterations and 9.7888×10^3 seconds. We can compare to the baseline case of only running the fine grid and see a significant improvement in iterations

and running time.
Next we try running the sparse grid up to 10^{-1}, which takes 101 iterations and 304.3667 seconds. The following fine grid solve takes 234 iterations and 6.6193×10^3 seconds. Again, there's a saving in time and iterations.
Then we run the sparse grid up to 10^{-2}, which takes 240 iterations and 747.4124 seconds. The fine grid problem then takes 118 iterations and 3.3464×10^3 seconds, yet another improvement. Finally, running the sparse grid problem all the way to 10^{-3} takes 412 iterations and 1.0586×10^3 seconds. Then solving the corresponding problem on the fine grid takes 114 iterations and 3.2017×10^3 seconds. So in this case, the final step does not improve the overall running time of the code.

3 Example Problem 2

We choose Example 1 from the MultiShape examples and do the same thing as above. First we run the problem only on the fine grid to get a value to compare the results of the scheme with. Since this is a MultiShape, we will have two shapes with $N = 40$ each and $n = 20$, which makes this a larger problem than the one above. It takes 440 iterations in 6.0083×10^4 seconds.
First we set the sparse grid problem to run to an optimality tolerance of 1.