Part 4

Setting up a convex problem

We first consider the 2D case as this allows for pretty visuals. We then move over to a bigger problem.

A general equation of a paraboloid is stated with three terms:

$$P(x) = x^T Ax + Cx + d$$

for $A \in \mathbb{R}^{n\times n} \succ 0$, $C \in \mathbb{R}^n$, and $d \in \mathbb{R}$

Using this rule, we generate data:

For projected gradient descent, the problem may be formulated as follows:

$$\min_{x\in K} P(x) \text{ s.t. } ||x|| \leq R$$

where K is a convex set. The gradient writes as

$$\nabla P = 2Ax + C$$

For this project we consider 3 different convex sets.

- If K is the ℓ_2-ball, the projection may be expressed as:

$$\Pi_A(x) = \frac{R}{max(||x||_2, R)} x$$

- Projection onto the simplex with sum R
- Projection onto the ℓ_1-ball with radius R

The last 2 projections have algorithms proposed by Laurent Condat in his paper *Fast Projection onto the Simplex and the l1 Ball*

(See Laurent Condat. Fast Projection onto the Simplex and the l1 Ball. Mathematical Programming, Series A, 2016, 158 (1), pp.575-585. 10.1007/s10107-015-0946-6. hal-01056171v2)

For the Franke-Wolfe / Conditional gradient

When the convex set K is the ℓ_2-ball with radius R, we wish to solve:

$$s_{k+1} \in \underset{s\in B(0,R)}{\operatorname{argmin}} \Big\{ f(x_k) + \langle \nabla f(x_k), s - x_k \rangle \Big\} \tag{1}$$

which is known to have explicit solution:

$$s_{k+1,j} = -R \frac{\text{sign}(\nabla f_j(x_k)) |\nabla f_j(x_k)|}{||\nabla f(x_k)||_2} \tag{2}$$

The weighting of the convex combination may be determined by exact linesearch:

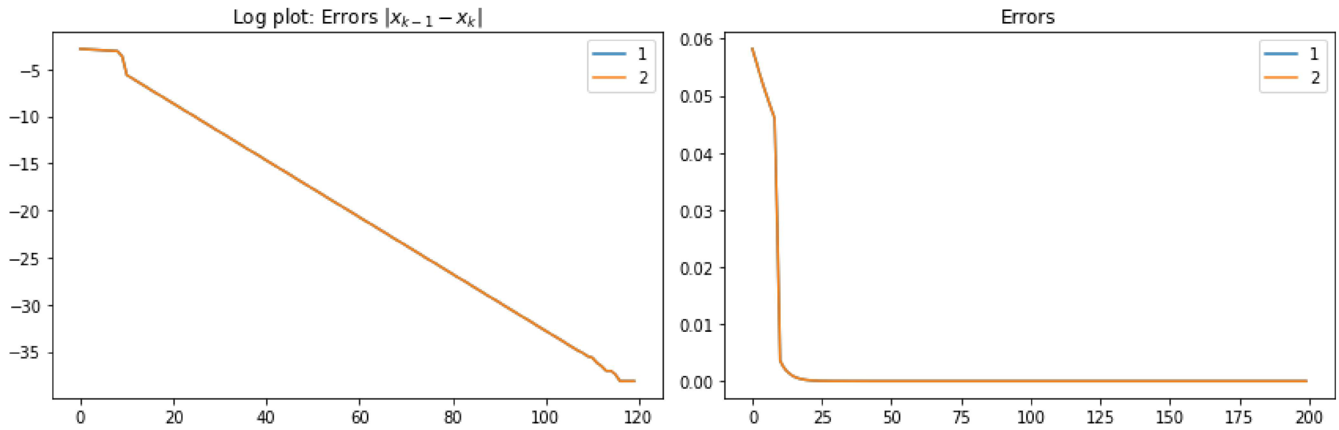
$$\theta_k \in \operatorname{argmin}_{\theta \in [0,1]} f(\theta_k s_{k+1} + (1-\theta) x_k) \tag{3}$$

or by making use of a prescribed step size such as

$$\theta_k = \frac{2}{k+2} \tag{4}$$

We choose an arbitrary starting point x_0 on the boundary of the two balls with radius R and run an iteration of PGD for each:

We plot the successive differences between iterates on normal and log scale

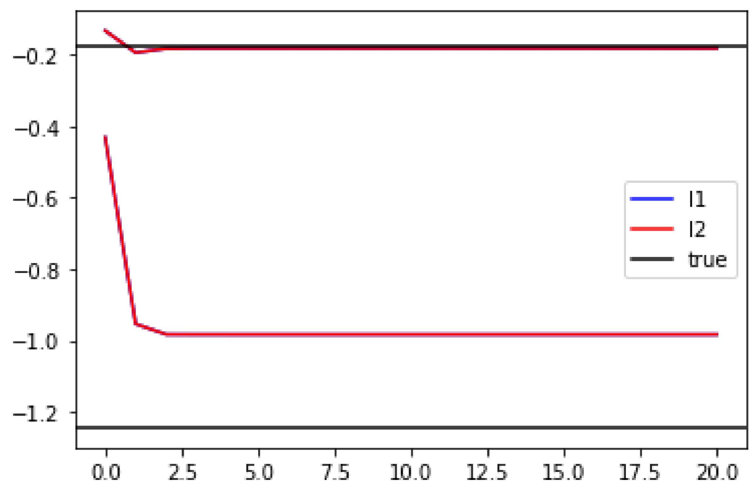


The two balls render almost exactly the same results.

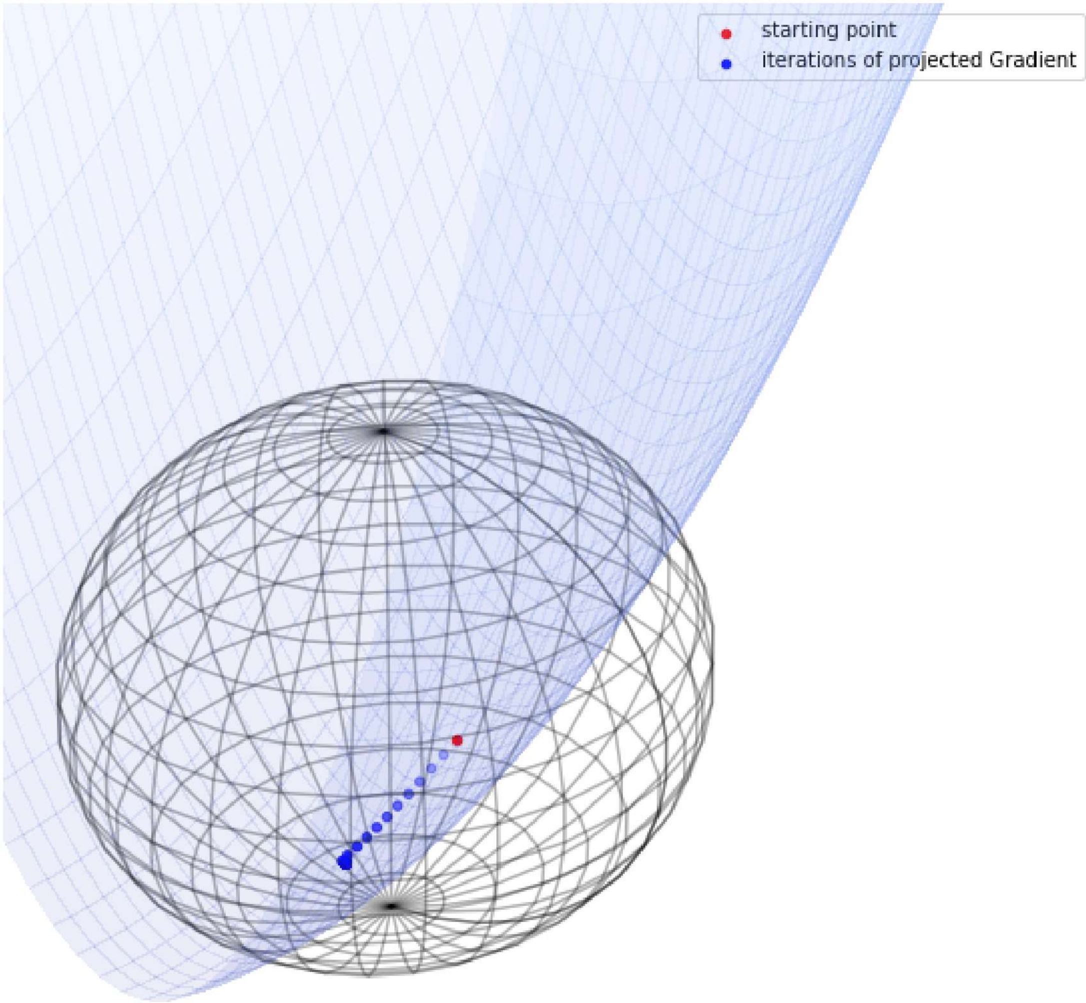
For a problem of small dimension as here, we may determine the exact solution by making use of scipy's *optimize* library. The optimal solution x^* is:

```
array([-0.17323659, -1.24429379])
```

We plot the true values of the iterates as black lines and for each of the ℓ_1, ℓ_2 balls, plot the evolution of the iterates.

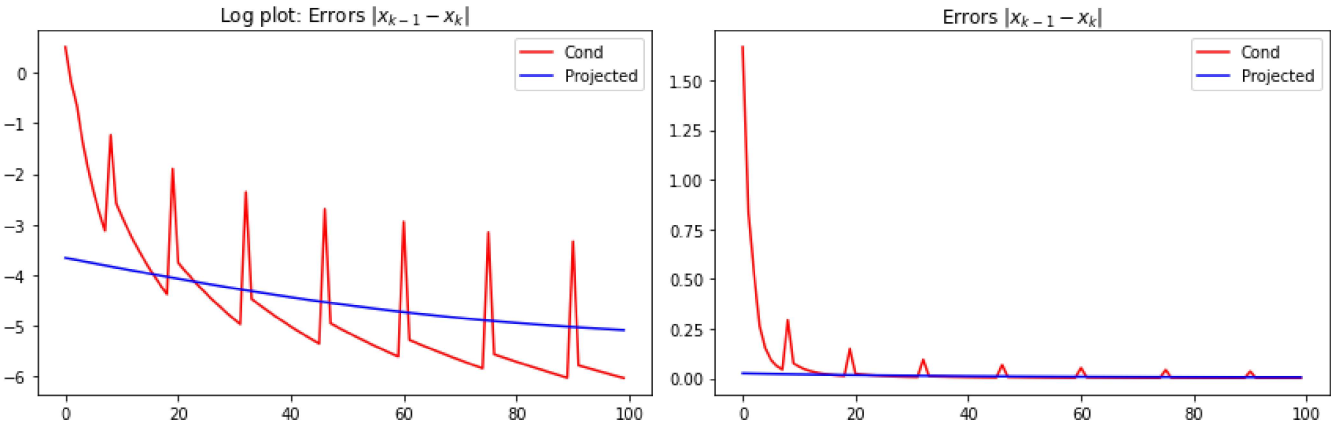


As we first consider a 2D small problem, we may plot the evolution of the iterates in the ℓ_2 ball in 3D space



We now compare the two methods on two different balls.

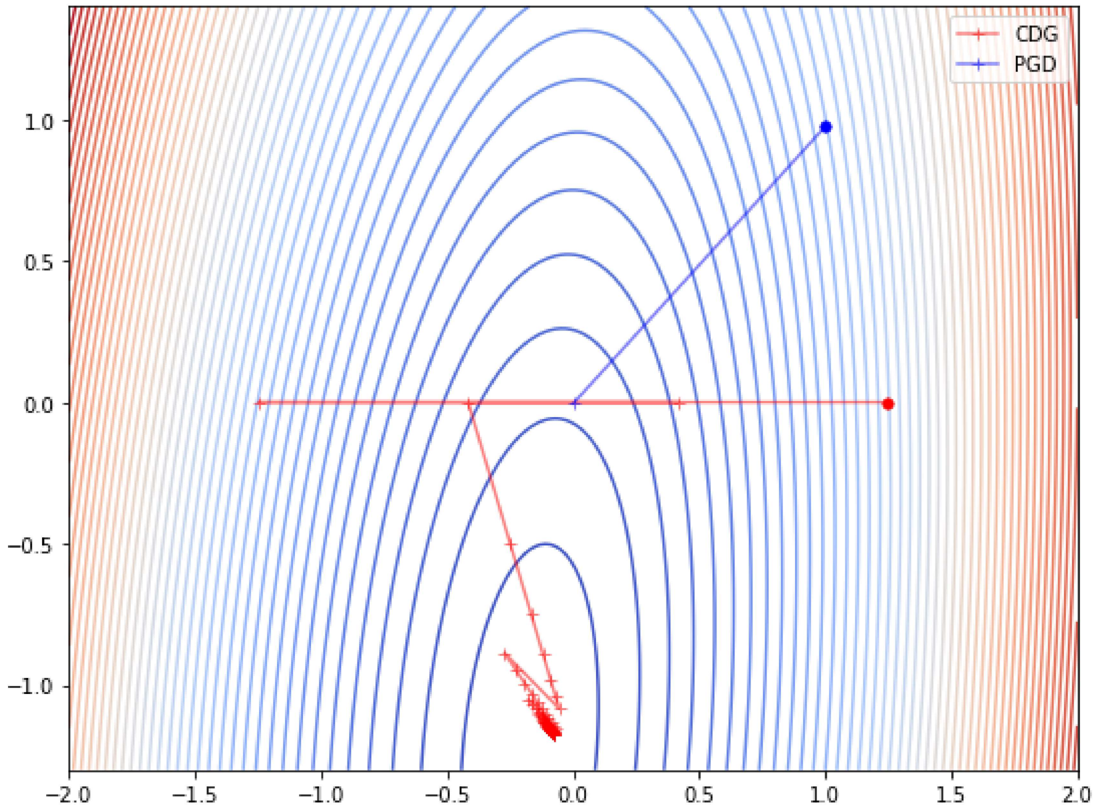
For the ℓ_1-ball



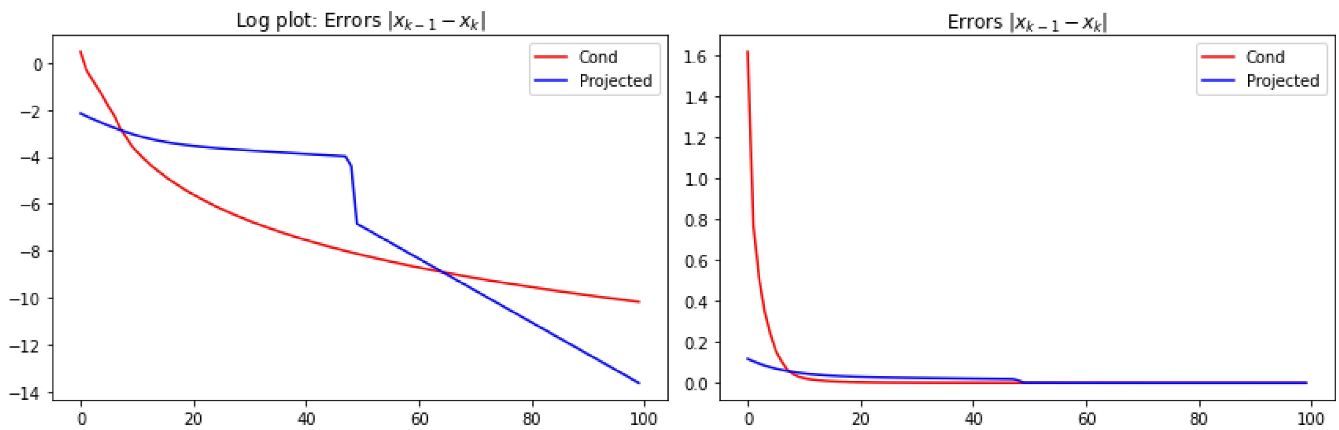
Both iterates converge fairly quickly when considering successive differences in the norm:

```
Final Difference:
Cond: 0.002388635446043774
Proj: 0.006167361498751566
```

We note, however, that CGD converges faster for this ball

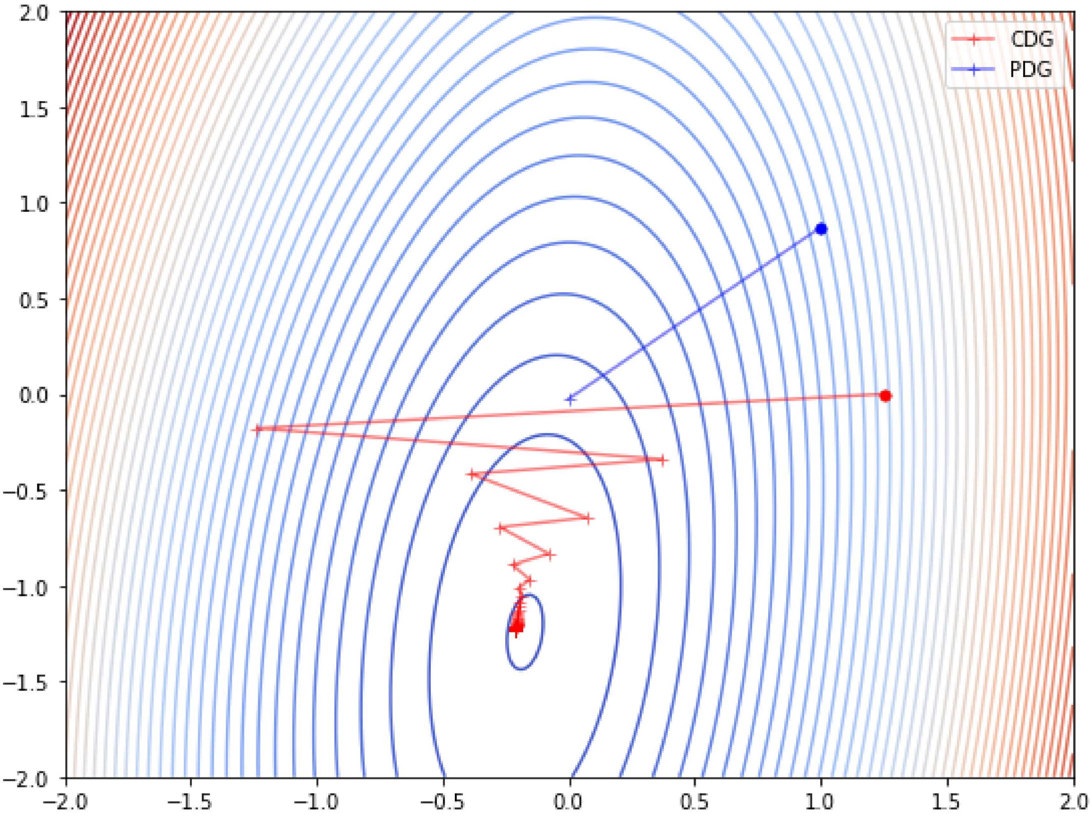


For the ℓ_2-ball



Here PGD seems to converge almost immediately and then stagnates. This happened even for multiple values of the step parameter

```
Final Difference:
Cond: 3.806120191684874e-05
Proj: 1.1969725472280687e-06
```


CDG
PDG
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0