**1-D minimization:**

The four algorithms are implemented and tested on the following function

Which has a minimum at

By comparing the four algorithms we get

1. **Fibonacci:**
2. Optimal point = 5
3. Iterations = 15
4. Fibonacci number = 17
5. Time = 0.005717 seconds
6. **Golden:**
7. Optimal point = [4.9845,5.0155]
8. Iterations = 4
9. Time = 0.007444
10. **Quadratic interpolation:**
11. Optimal point = 5
12. Iterations = 2
13. Time = 0.000399
14. **Cubic interpolation:**

Because cubic interpolation doesn’t converge on quadratic functions I tried on the following function

1. Optimal point = 4.9965
2. Iterations = 17
3. Time = 0.008834

**Rosen Brock:**

**Fletcher and Reeves conjugate gradient method:**

1. Optimal = [1.0026 , 1.0052]
2. Iterations = 1574
3. Time = 0.048640

**Marquardt:**

1. Optimal = [0.9974 , 0.9948]
2. Iterations = 22
3. Time = 0.013231

**DFP:**

I couldn’t make it work using DFP because my matlab crashes whenever I uses the symbolic toolbox so I had to approximate the difference using finite difference in the cubic interpolation function, but since the rosenbrock function is very steep the errors accumulate and leads to wrong directions after a small number of iterations which makes the function has no optimal point in the chosen direction and the algorithm gets stuck. I tried most of the heuristics in the book but none work. Also the quadratic interpolation doesn’t work because the rosenbrock function is cubic when substituted in any direction which makes the quadratic interpolation useless and leads to the same problem of stuck algorithm.

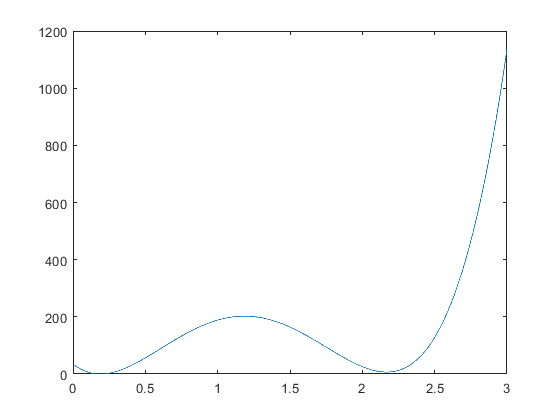


Figure 1: Rosenbrock in lamda

**Powell’s quartic function:**

**Fletcher and Reeves conjugate gradient method:**

1. Optimal =
2. Iterations = 93
3. Time = 0.007221

**Marquardt:**

1. Optimal =
2. Iterations = 14
3. Time = 0.005904

**DFP:**

1. Optimal =
2. Iterations = 19
3. Time = 0.529679