- 第一章作业总结

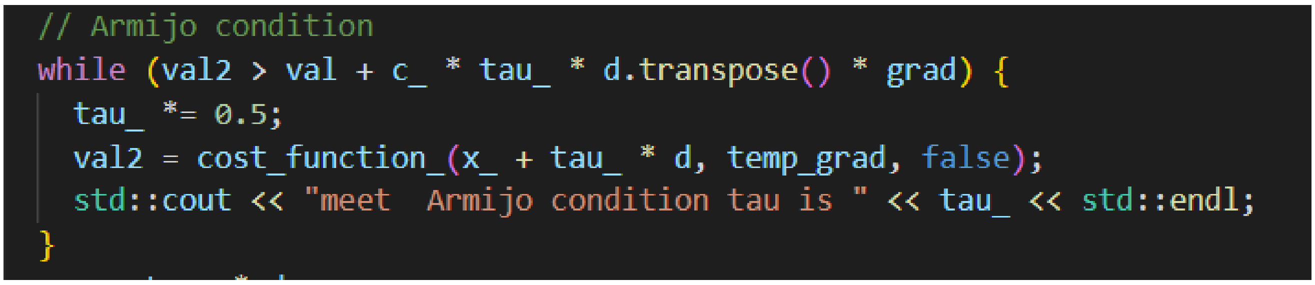
第一章作业总结

1. 目标函数,根据代码编写习惯进行改写

$$f(X) = f(x_0, x_1, ..., x_{N-1}) = \sum_{i=0}^{N/2} [100(x_{2i}^2 - x_{2i+1})^2 + (x_{2i} - 1)^2]$$

2. Armijo condition

$$\tau \in \{\alpha | f(x^k) - f(x^k + \alpha d) \geqslant -c * d^T \Delta f(x^k)\}$$

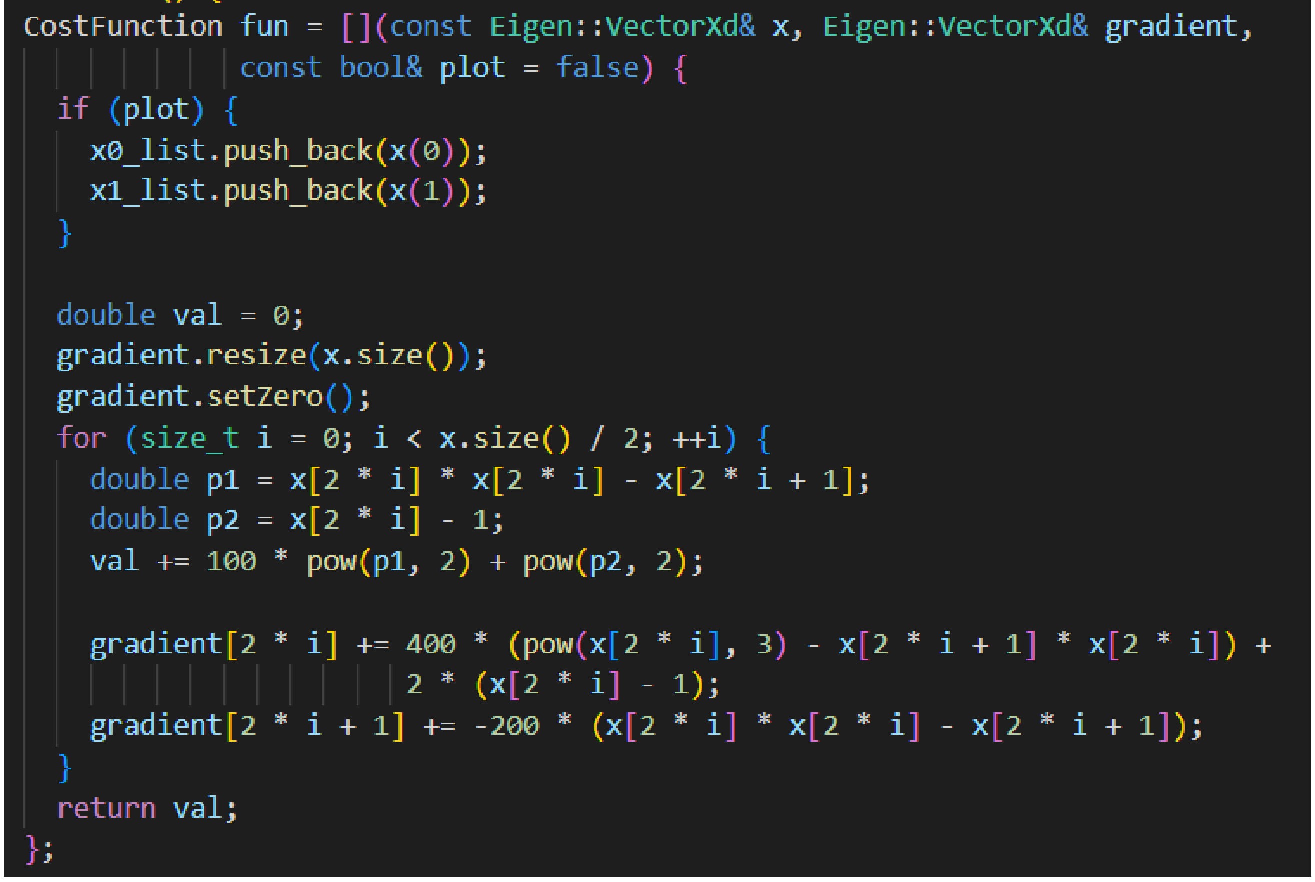


```
// Armijo condition
while (val2 > val + c_ * tau_ * d.transpose() * grad) {
  tau_ *= 0.5;
  val2 = cost_function_(x_ + tau_ * d, temp_grad, false);
  std::cout << "meet  Armijo condition tau is " << tau_ << std::endl;
}
```

3. 将公式展开，方便进行梯度计算

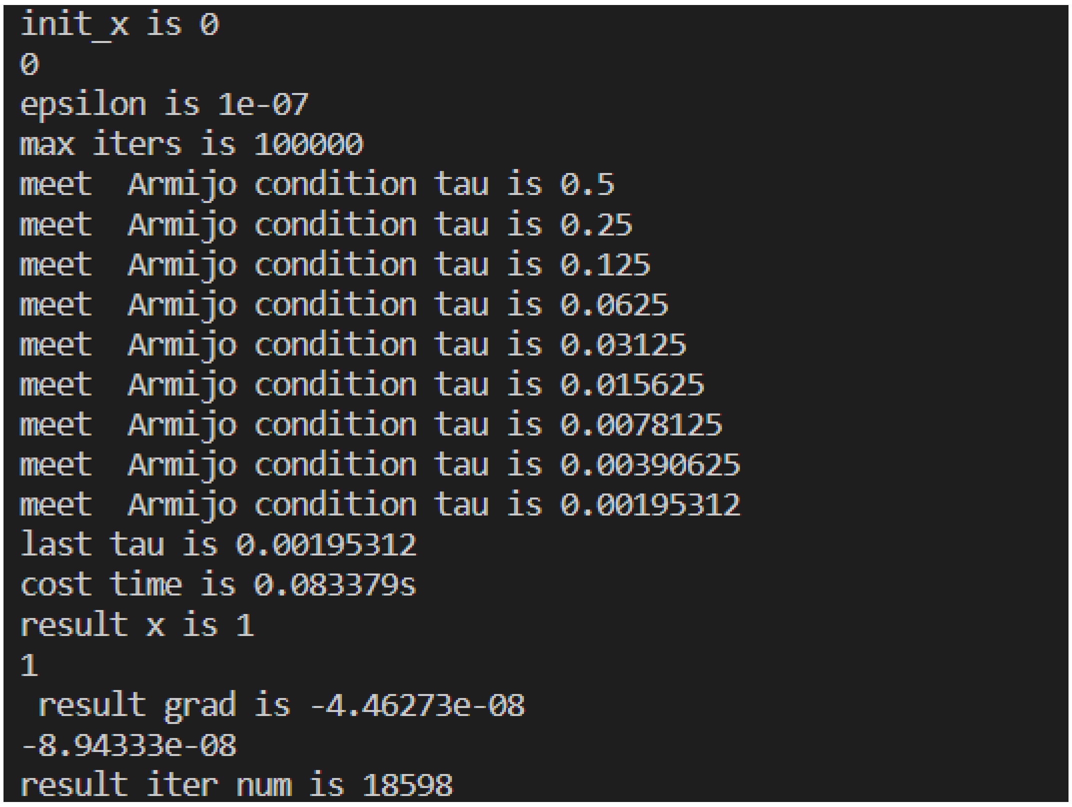
$$f(X) = \sum_{i=0}^{N/2} [100 * x_{2i}^4 - 200x_{2i}^2 * x_{2i+1} + 100x_{2i+1}^2 + x_{2i}^2 - 2 * x_{*i}]$$

4. 目标函数与梯度计算图



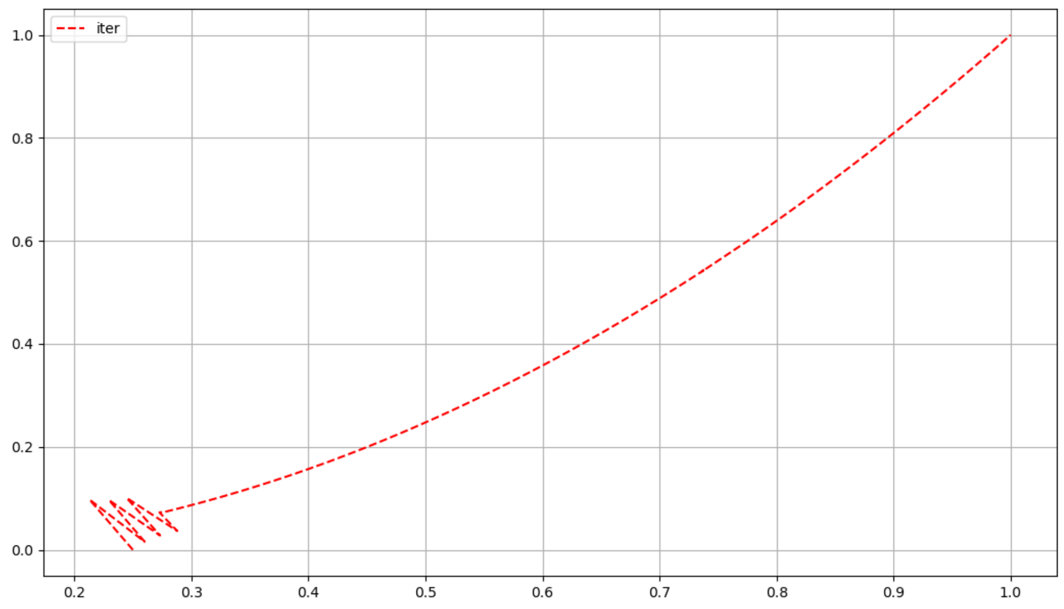
```
CostFunction fun = [](const Eigen::VectorXd& x, Eigen::VectorXd& gradient,
                      const bool& plot = false) {
  if (plot) {
    x0_list.push_back(x(0));
    x1_list.push_back(x(1));
  }

  double val = 0;
  gradient.resize(x.size());
  gradient.setZero();
  for (size_t i = 0; i < x.size() / 2; ++i) {
    double p1 = x[2 * i] * x[2 * i] - x[2 * i + 1];
    double p2 = x[2 * i] - 1;
    val += 100 * pow(p1, 2) + pow(p2, 2);

    gradient[2 * i] += 400 * (pow(x[2 * i], 3) - x[2 * i + 1] * x[2 * i]) +
                       2 * (x[2 * i] - 1);
    gradient[2 * i + 1] += -200 * (x[2 * i] * x[2 * i] - x[2 * i + 1]);
  }
  return val;
};
```

5. 优化结果设置



```
init_x is 0
0
epsilon is 1e-07
max iters is 100000
meet  Armijo condition tau is 0.5
meet  Armijo condition tau is 0.25
meet  Armijo condition tau is 0.125
meet  Armijo condition tau is 0.0625
meet  Armijo condition tau is 0.03125
meet  Armijo condition tau is 0.015625
meet  Armijo condition tau is 0.0078125
meet  Armijo condition tau is 0.00390625
meet  Armijo condition tau is 0.00195312
last tau is 0.00195312
cost time is 0.083379s
result x is 1
1
 result grad is -4.46273e-08
-8.94333e-08
result iter num is 18598
```

6. 迭代结果图



7. 使用绘图需要安装 sudo apt-get install libpython3-dev