UNIVERSITÀ DEGLI STUDI DI PADOVA

3D Data Processing
Ceres-Solver Tutorial

Alberto Pretto

Goal

Solve robustified **non-linear least squares** problems of the form

Residual Block

$$\min_{\mathbf{x}} \quad \frac{1}{2}\sum_i \rho_i\left(\left\|f_i\left(x_{i_1},\ldots,x_{i_k}\right)\right\|^2\right)$$

Loss Function (Huber, Cauchy ...)

Cost Function

Parameter Block/s

Example

Find the minimum of the function

$$\frac{1}{2}(10 - x)^2$$

Solve it with Ceres:

1) Write a functor that will evaluate the residual
2) Build the non-linear least squares problem
3) Setup and run the solver

```cpp
struct CostFunctor {
   template <typename T>
   bool operator()(const T* const x, T* residual) const {
     residual[0] = 10.0 - x[0];
     return true;
   }
};
```

```cpp
int main(int argc, char** argv) {
  google::InitGoogleLogging(argv[0]);

  // The variable to solve for with its initial value.
  double initial_x = 5.0;
  double x = initial_x;

  // Build the problem.
  Problem problem;

  // Set up the only cost function (also known as residual). This uses
  // auto-differentiation to obtain the derivative (jacobian).
  CostFunction* cost_function =
      new AutoDiffCostFunction<CostFunctor, 1, 1>(new CostFunctor);
  problem.AddResidualBlock(cost_function, nullptr, &x);

  // Run the solver!
  Solver::Options options;
  options.linear_solver_type = ceres::DENSE_QR;
  options.minimizer_progress_to_stdout = true;
  Solver::Summary summary;
  Solve(options, &problem, &summary);

  std::cout << summary.BriefReport() << "\n";
  std::cout << "x : " << initial_x
            << " -> " << x << "\n";
  return 0;
}
```

Example

Find the minimum of the function

$$\frac{1}{2}(10 - x)^2$$

OUTPUT

```
iter      cost      cost_change  |gradient|   |step|    tr_ratio  tr_radius  ls_iter  iter_time  total_time
   0  4.512500e+01    0.00e+00    9.50e+00   0.00e+00   0.00e+00  1.00e+04        0    5.33e-04    3.46e-03
   1  4.511598e-07    4.51e+01    9.50e-04   9.50e+00   1.00e+00  3.00e+04        1    5.00e-04    4.05e-03
   2  5.012552e-16    4.51e-07    3.17e-08   9.50e-04   1.00e+00  9.00e+04        1    1.60e-05    4.09e-03
Ceres Solver Report: Iterations: 2, Initial cost: 4.512500e+01, Final cost: 5.012552e-16, Termination: CONVERGENCE
x : 5.0 -> 10
```

Automatic differentiation

- Ceres can compute automatically the derivatives wrt the parameters vector while computing residuals
- The parameters can be divided into "blocks", as for example done in bundle adjustment ("camera" blocks and "point" blocks), for simplify managing the sparsity

Adding residuals

For each residual, we need to add a corresponding "residual block" to the optmization probkem:

```
ceres::Problem problem;
for( /* iterate for each data point */ )
{
  ceres::CostFunction* cost_function = ...;

  problem.AddResidualBlock( cost_function,
                param_block1, param_block2, ...);

}
```

Adding residuals

We need to define a functor (just a class or struct which defines the operator()) that computes the resiudual:

```
struct Functor
{
  template <typename T> bool operator()(const T* const param_block1,
                                      const T* const param_block2,
                                      ...,
                                      T* residuals) const
  {
    // Compute the residuals given the input parameters blocks
    return true; // Success
  }
}
```

Adding residuals

Then we use this functor to construct the const function:

```
Functor funct = new Functor(...);
ceres::CostFunction* cost_function = new
ceres::AutoDiffCostFunction<Functor,Nr, Nb1, Nb2,... >(funct);
```

$N_{r:}$dimension of a single residual

$N_{b1:}$dimension of parameters block 1

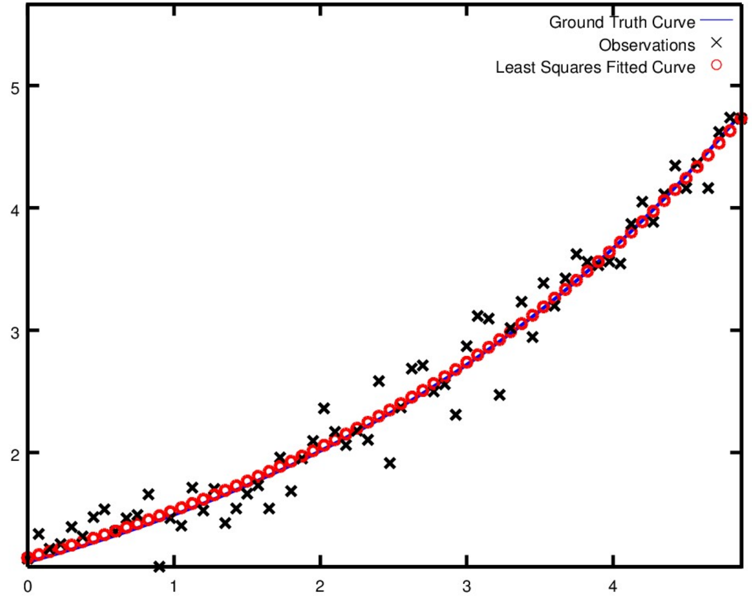
$N_{b2:}$dimension of parameters block 2

....

Curve Fitting

Given a set of observed data points, find the best fitting exponential curve

$$y = e^{mx+c}$$




```
struct ExponentialResidual {
  ExponentialResidual(double x, double y)
      : x_(x), y_(y) {}

  template <typename T>
  bool operator()(const T* const m, const T* const c, T* residual) const {
    residual[0] = y_ - exp(m[0] * x_ + c[0]);
    return true;
  }

 private:
  // Observations for a sample.
  const double x_;
  const double y_;
};
```

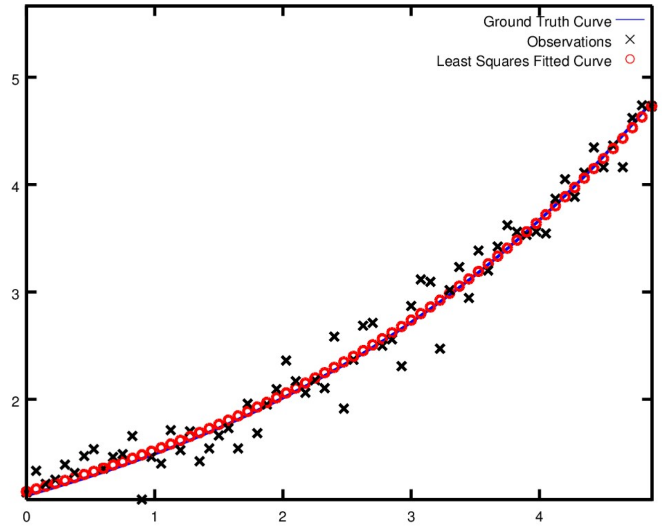
```
double m = 0.0;
double c = 0.0;

Problem problem;
for (int i = 0; i < kNumObservations; ++i) {
  CostFunction* cost_function =
       new AutoDiffCostFunction<ExponentialResidual, 1, 1, 1>(
           new ExponentialResidual(data[2 * i], data[2 * i + 1]));
  problem.AddResidualBlock(cost_function, nullptr, &m, &c);
}
```

Curve Fitting

Given a set of observed data points, find the best fitting exponential curve

$$y = e^{mx+c}$$



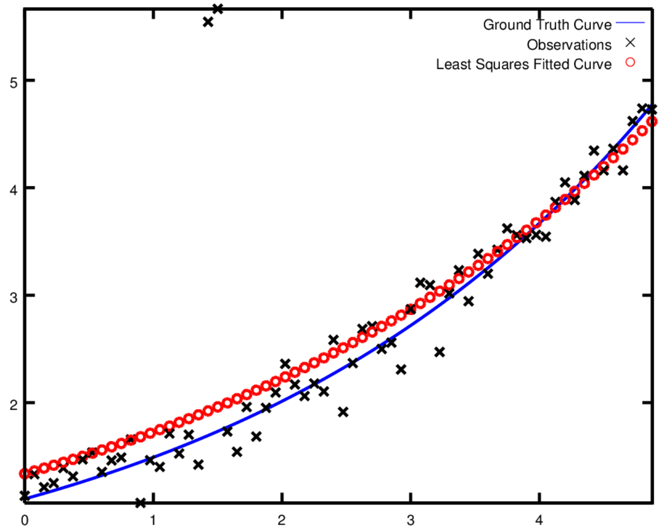
iter	cost	cost_change	|gradient|	|step|	tr_ratio	tr_radius	ls_iter	iter_time	total_time
0	1.211734e+02	0.00e+00	3.61e+02	0.00e+00	0.00e+00	1.00e+04	0	5.34e-04	2.56e-03
1	1.211734e+02	-2.21e+03	0.00e+00	7.52e-01	-1.87e+01	5.00e+03	1	4.29e-05	3.25e-03
2	1.211734e+02	-2.21e+03	0.00e+00	7.51e-01	-1.86e+01	1.25e+03	1	1.10e-05	3.28e-03
3	1.211734e+02	-2.19e+03	0.00e+00	7.48e-01	-1.85e+01	1.56e+02	1	1.41e-05	3.31e-03
4	1.211734e+02	-2.02e+03	0.00e+00	7.22e-01	-1.70e+01	9.77e+00	1	1.00e-05	3.34e-03
5	1.211734e+02	-7.34e+02	0.00e+00	5.78e-01	-6.32e+00	3.05e-01	1	1.00e-05	3.36e-03
6	3.306595e+01	8.81e+01	4.10e+02	3.18e-01	1.37e+00	9.16e-01	1	2.79e-05	3.41e-03
7	6.426770e+00	2.66e+01	1.81e+02	1.29e-01	1.10e+00	2.75e+00	1	2.10e-05	3.45e-03
8	3.344546e+00	3.08e+00	5.51e+01	3.05e-02	1.03e+00	8.24e+00	1	2.10e-05	3.48e-03
9	1.987485e+00	1.36e+00	2.33e+01	8.87e-02	9.94e-01	2.47e+01	1	2.10e-05	3.52e-03
10	1.211585e+00	7.76e-01	8.22e+00	1.05e-01	9.89e-01	7.42e+01	1	2.10e-05	3.56e-03
11	1.063265e+00	1.48e-01	1.44e+00	6.06e-02	9.97e-01	2.22e+02	1	2.60e-05	3.61e-03
12	1.056795e+00	6.47e-03	1.18e-01	1.47e-02	1.00e+00	6.67e+02	1	2.10e-05	3.64e-03
13	1.056751e+00	4.39e-05	3.79e-03	1.28e-03	1.00e+00	2.00e+03	1	2.10e-05	3.68e-03

```
Ceres Solver Report: Iterations: 13, Initial cost: 1.211734e+02, Final cost: 1.056751e+00, Termination: CONVERGENCE
Initial m: 0 c: 0
Final   m: 0.291861 c: 0.131439
```

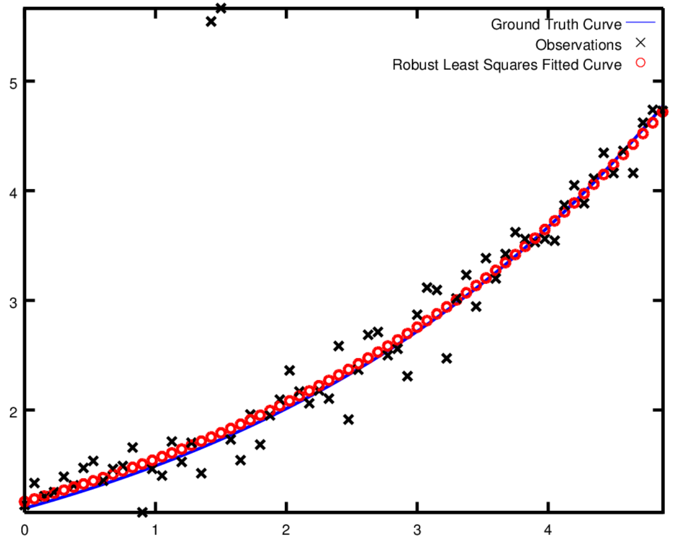
Robust Curve Fitting

$$y = e^{mx+c}$$

Without Loss Function



With Loss Function



Exploit loss functions for reducing the influence of outliers

```
problem.AddResidualBlock(cost_function, new CauchyLoss(0.5) , &m, &c);
```

References

- http://ceres-solver.org/nnls_tutorial.html
- http://ceres-solver.org/nnls_tutorial.html#bundle-adjustment