Università della Svizzera italiana

Institute of Computational Science ICS

Evaluating Tensorflow Optimization Techniques for Solving Elliptic Boundary Control Problems

Manav Choudhary
manav.choudhary@usi.ch

Tensorflow Optimization

- Tensorflow is a software library which uses data flow graphs for numerical computations
- Graph has nodes for mathematical operations and edges are the data tensors
- Use Tensorflow for large scale optimal control problems constrained by elliptic partial differential equations

For constrained optimization Tensorflow uses a wrapper over Scipy optimize method with: SLSQP, COBYLA, TNC, L-BFGS-B

- SLSQP uses the Han-Powell quasi-Newton method with a BFGS update. COBYLA, approximates the original problem with linear programming problems. TNC approximates Newtons equations, to update the parameters using conjugate gradient. L-BFGS-B uses an estimation to the inverse Hessian matrix, storing only a few vectors.
- For unconstrained optimization problems Tensorflow provides: Gradient_Descent, Momentum, RMSProp, Adam, Adadelta, AdagradDA

TensorFlow NumPy python SciPy

Tensorflow Implementation

- Forward Problem:
- Solve the forward problem of solving the PDE for some given initial boundary values
- Reverse Problem:
- Solve the reverse control problem of finding the optimal boundary values for the given desired function definition

Algorithm 1 SLSQP constraint optimization

Input: Discretization grid size (N)
Output: Boundary control values

function DIRICHLET_BOUNDARY_CONTROL(N)
    \\ Prepare the A matrix.
    $A \leftarrow block\_diag(J)$
    $A \leftarrow A + I\_lower + I\_upper$
    \\ Initialize vector b and initialize weights i.e left, right, upper and lower boundary values
    $b \leftarrow ones\_vector(N^2)$
    $boundary\_weights \leftarrow zeros\_vector(N)$
    $b \leftarrow b + reshaped\_boundary\_weights$
    $y\_desired \leftarrow 3.0 + 5.0 * domain\_x * (domain\_x - 1) * domain\_y * (domain\_y - 1)$
    loss $\leftarrow$ sum over domain$(y_desired - y_actual)^2$ + sum over boundary$(u_desired - weights)^2$
    optimizer $\leftarrow$ tensorflow Scipy Optimizer Interface(loss, constraints, method='SLSQP')
    while not converged **do**
        $optimizer.minimize(session)$
    end while
    return $boundary\_weights$
end function

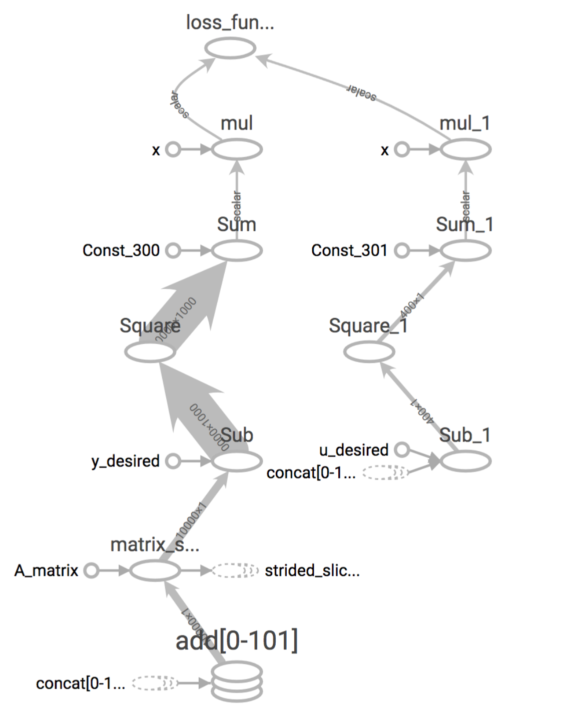


Figure 1: Tensorflow Compuatation Graph

Elliptic boundary control problem benchmark

- Use finite difference discretization technique, tens of thousands of control and state variables

Mittelmann Boundary control problems

- Objective Function:

$$F(y,u) = \int_\Omega f(x,y(x))dx + \int_\Gamma g(x,y(x),u(x))dx \tag{1}$$

- State equations:

$$\begin{aligned} -\Delta y(x) + d(x,y(x)) &= 0, && \text{for } x \in \Omega, \\ \partial_\nu y(x) &= b(x,y(x),u(x)), && \text{for } x \in \Gamma, \end{aligned} \tag{2}$$

- Inequality coznstraints:

$$\begin{aligned} C(x,y(x),u(x)) &\leq 0, && \text{for } x \in \Gamma, \\ S(x,y(x)) &\leq 0, && \text{for } x \in \bar{\Omega}, \end{aligned} \tag{3}$$

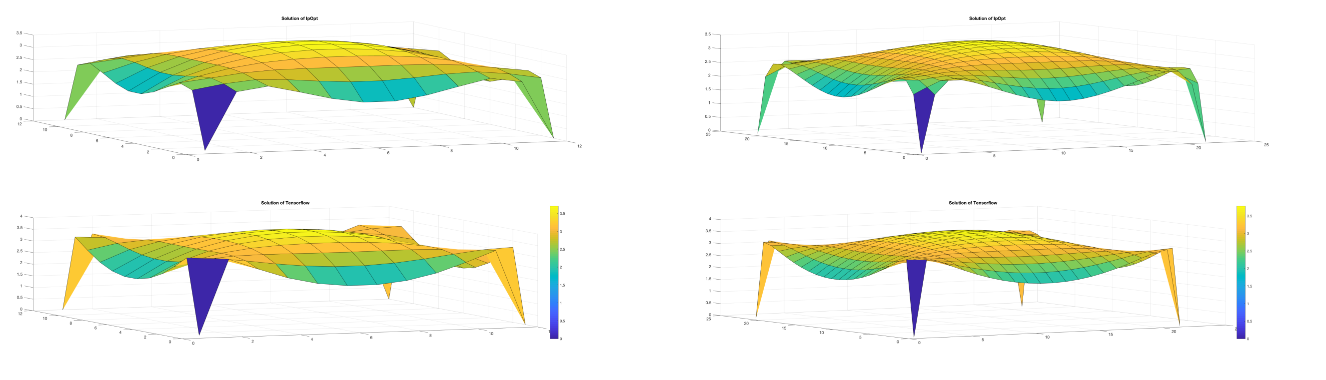
Dirichlet boundary conditions:

- Boundary conditions:

$$\begin{aligned} \text{on } \Omega: \quad -\Delta y(x) &= 20, \\ y(x) &\leq 3.5, \\ y_d(x) &= 3 + 5x_1(x_1-1)x_2(x_2-1), \\ \text{on } \Gamma: \quad y(x) &= u(x), \\ 0 \leq u(x) &\leq 10, \\ u_d(x) &\equiv 0, \alpha = 0.01 \end{aligned} \tag{4}$$

Ipopt Implementation and Results

- **Ipopt (Interior Point Optimizer)** is a software package for large-scale nonlinear optimization.
- **PARDISO package** is a thread-safe, high-performance, robust, memory efficient and easy to use software for solving large sparse symmetric and asymmetric linear systems of equations on shared-memory and distributed-memory multiprocessors



(a) Grid size 10x10 (b) Grid size 20x20

Figure 2: Ipopt (above) and Tensorflow (below) domain values for different grid sizes

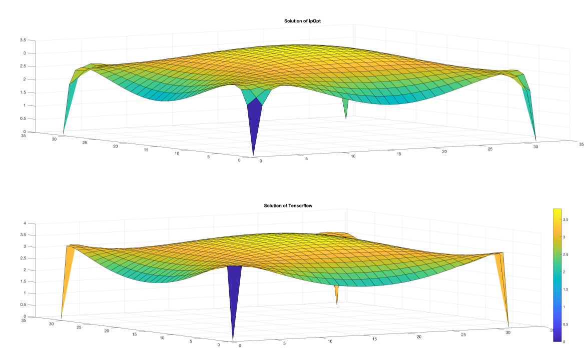


Figure 3: Ipopt (above) and Tensorflow (below) domain values for grid size 30x30

Results

- For N = 100, tensorflow implementation with SLSQP method takes more than 10 hours
- For N = 100, Ipopt takes 0.84 sec, Tensorflow takes 32.324 sec for the forward problem alone

Table 1: Comparison of Ipopt vs Tensorflow performance on time required

Grid Size	Ipopt	SLSQP		COBYLA		TNC		L-BFGS-B	
	time (sec)	time (sec)	relative error	time (sec)	relative error	time (sec)	relative error	time (sec)	relative error
10	0.119	45	0.054	106	0.054	0.507	0.084	0.447	0.084
12	0.073	92	0.054	217	0.054	0.703	0.091	0.752	0.091
14	0.066	189	0.053	448	0.054	1.025	0.096	0.949	0.096
16	0.068	344	0.053	921	0.053	1.739	0.099	1.399	0.099

Table 2: Comparison of Ipopt vs Tensorflow on required number of iterations

Grid Size	Ipopt	SLSQP	COBYLA	TNC	L-BFGS-B
	#iterations	#iterations	#iterations	#iterations	#iterations
10	13	14	1000	9	10
12	13	18	1000	10	13
14	13	22	1000	10	12
16	13	21	1000	10	15

Results and References

- Tensorflow constraint optimization methods do not scale well with large grid size
- Ipopt with Pardiso solver scales very well with large grid size

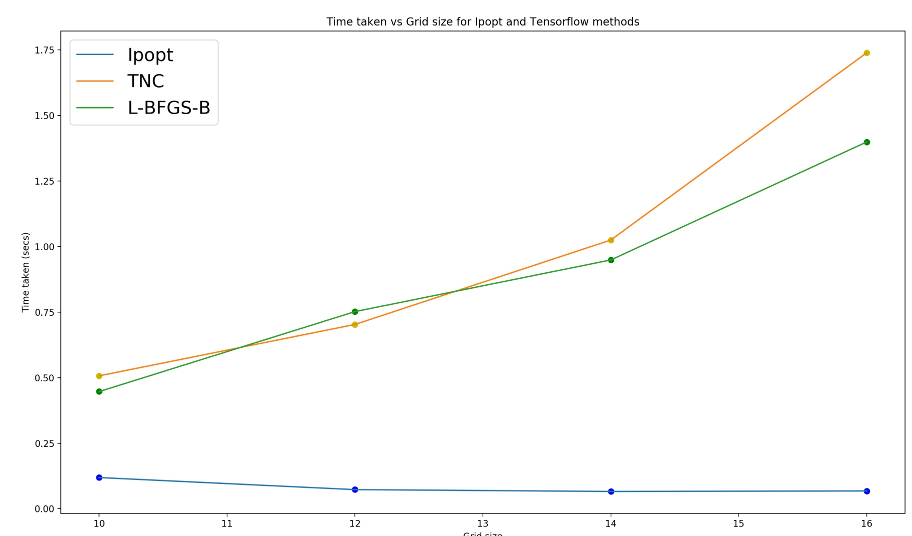


Figure 4: Time taken vs Grid size for Ipopt and tensorflow optimization methods

- D. Kourounis, A. Fuchs, and O. Schenk, Towards the next generation of multiperiod optimal power flow solvers, IEEE Transactions on Power Systems, vol. PP, no. 99, pp. 110, 2018.

This work is based on a master student project within the course "Software Atelier: Simulation, Data Science Supercomputing" at Universita della Svizzera Italiana