REU: Numerical optimal control

Hoang Nguyen*†
Final report

July 21, 2018

Contents

*Advised by Professor Molei Tao
†Email:hoang.nguyen@minerva.kgi.edu, student of Minerva schools at KGI

1 Overview

In summer 2018, I had the privilege to go to Georgia Tech for a research experience for undergraduates (REU) in numerical optimal control under the tutelage of Professor Molei Tao. I spent six weeks learning in-depth about calculus of variations, numerical methods and potential research directions in optimization and control theory. I also took part in other social and professional activities such as hiking, board game evening and talks on how to apply to graduate schools.

Project goal

Our original problem is to optimize the trajectory of a drone with respect to Newtonian dynamics such that it minimizes the cost (such as energy or distance traveled). We can formulate the problem as minimizing:

$$\int_0^T \mathcal{L}(q)^2 dt$$

We can try to solve this problem by using calculus of variations. However, it usually leaves us with a differential equation too complex to solve. Thus, we need to employ numerical methods and the goal of this project is to research effective numerical methods to solve such problems, especially infinite dimensional problems.

Existing research

Our project builds on the work of Weinan E, Weiqing Ren, and Eric Vanden-Eijnden. "Minimum action method for the study of rare events." Communications on pure and applied mathematics 57.5 (2004): 637-656.

2 Summary of my study

2.1 Calculus of variations

In the first 3 weeks, I learned about calculus of variations and its applications on the Euler-Lagrange equations. This will be the basis for the numerical methods that we applied in the later weeks of the REU. Some major results that I learned in this area include:

Fundamental lemma of calculus of variations

Lemma: If a continuous function f on an open interval (a, b) satisfies the equality $\int_a^b f(x)h(x)\ \mathrm{d}\,x = 0$ for all compactly supported smooth functions h on (a, b), then f is identically zero.

This lemma is a strong tool to solve functional derivative problems. In the subsequent Euler-Lagrange equations that we will encounter throughout the project, we mostly use calculus of variations to tackle easy and hard problems alike.

Euler-Lagrange equation

Calculus of variations can be used to prove the Euler-Lagrange equation.

Let

$$S(y) = \int_a^b \mathcal{L}(x, y, y')dx$$

And h is a perturbation of y We have:

$$\delta S = S(y + \epsilon h) - S(y) = \int_a^b \mathcal{L}(x, y + \epsilon h, y' + \epsilon h')dx - \int_a^b \mathcal{L}(x, y, y')dx(3a)$$

From the decomposition rule of functional derivative, we have:

$$\frac{d\mathcal{L}}{\delta x} = \frac{\delta\mathcal{L}}{\delta x} + y'\frac{\delta\mathcal{L}}{\delta y} + y''\frac{\delta\mathcal{L}}{y'}$$

$$(3a) \Leftrightarrow \int_a^b \frac{\delta\mathcal{L}}{\delta x} + \delta\frac{\mathcal{L}}{\delta y}(y' + \epsilon h) + \frac{\delta\mathcal{L}(x, y, y')}{\delta y'}(y' + \epsilon h')dx - \int_a^b \mathcal{L}(x, y, y')dx(*)$$

$$= \int_a^b \mathcal{L}(x, y, y') + \epsilon\frac{\delta\mathcal{L}}{\delta y}h + \frac{\delta\mathcal{L}}{\delta y'}h'dx - \int_a^b \mathcal{L}(x, y, y')$$

$$\Leftrightarrow \delta S = \epsilon\int_a^b \frac{\delta\mathcal{L}}{\delta y}h + \frac{\delta\mathcal{L}}{\delta y'}h'dx$$

$$\Leftrightarrow \delta S = \epsilon\int_a^b h\left(\frac{\delta\mathcal{L}}{\delta y} - \frac{d}{dx}\frac{\delta\mathcal{L}}{\delta y'}dx\right)$$

Choose y such that it is the function so that $S(y)$ is minimal, we have:

$$\lim_{x\to 0}\frac{\delta S}{\epsilon} = 0$$

$$\Leftrightarrow \int_a^b h\left(\frac{\delta\mathcal{L}}{\delta y} - \frac{d}{dx}\frac{\delta\mathcal{L}}{\delta y'}\right)dx = 0\forall h$$

From the fundamental lemma of calculus of variation, we have $\frac{\delta\mathcal{L}}{\delta y} - \frac{d}{dx}\frac{\delta\mathcal{L}}{\delta y'} = 0$.
Hence proved.

2.2 Finite difference method

Since most Euler-Lagrange equations are impossible to solve analytically, we looked into numerical methods to solve such equations. In this REU, we specifically worked on two finite difference methods: the explicit method and the implicit method (Crank-Nicolson method) and applied these methods to minimize the energy of the heat equation $\int_0^T (q')^2 dt$, optimize the trajectory of a drone that satisfy Newtonian dynamics.

2.2.1 Explicit method

In the explicit method, the finite difference method moves forward in time, that is the future value is equal to a function with past value inputs.

Example: Minimize $\int_0^1 (q')^2 dt$ such that $q(0) = 0, q(1) = 1$.

Solution: From calculus of variations, we know that $\delta\int_0^1 (q')^2 dt = 2\int_0^1 q'\delta q' dt = -2\int_0^1 q''\delta q dt$. From the partial differential equation

$$d_\tau q = -d_t^2 q$$

We can discretize it to obtain the following relation:

$$\frac{q_{i,j+1} - q_{i,j}}{\Delta\tau} = \frac{q_{i+1,j} - 2q_{i,j} + q_{i-1,j}}{(\Delta t)^2}$$

(We have to move in the opposite of the gradient in order to minimize the cost function)

$$\Leftrightarrow q_{i,j+1} = q_{i,j} + \frac{h}{k^2}(q_{i+1,j} - 2q_{i,j} + q_{i-1,j})$$

While it is easy to implement, the explicit method has many drawbacks, one of which is the time step usually has a bound, which means it limits the speed of convergence. In our heat equation example, we must have $\Delta\tau \le \frac{(\Delta t)^2}{2}$ and the time step will have to be much smaller when we deal with higher orders of differential (as shown below in the von Neumann analysis part).

2.2.2 Implicit method

In contrast to the explicit method, the implicit method moves backward in time, that is we write the finite differences as a system of equations where the future values are the unknowns.

Example: Minimize $\int_0^1 (q')^2 dt$ such that $q(0) = 0, q(1) = 1$.

Solution:Instead of writing the equation forward, we write the finite differences as following:

$$\frac{q_{i,j+1} - q_{i,j}}{\Delta\tau} = \frac{q_{i+1,j+1} - 2q_{i,j+1} + q_{i-1,j+1}}{(\Delta t)^2}$$

Since we know that $q_{0,j} = 0$ and $q_{n,j} = 1 \forall j$, let i runs from $1 \to n-1$, we have $n-1$ equations with $n-1$ unknowns.

2.3 Advanced gradient descent methods

Solving for the optimal function in the Euler-Lagrange equation is actually an infinite dimensional optimization problem.

2.3.1 Fictitious time method

Since the optimal control problem is an infinite dimensional optimization problem, a simple gradient descent method would be insufficient to tackle problem as it was used to solve finite dimensional optimization problem. Hence, we introduced a fictitious time variable τ to resemble the number of iteration in the conventional gradient descent and combined with calculus of variations to perform a gradient descent on the optimal control problem.

Example

Problem: Solve the heat equation $\int_0^1 (q')^2 dt$ given $q(0) = 0, q(1) = 1$.

Solution: Using calculus of variations, we have that:

$$\delta \int_0^1 (q')^2 dt$$

$$= 2 \int_0^1 q' \delta q' dt$$

$$= 2q'q\Big|_0^1 - 2 \int_0^1 q'' \delta q dt$$

$$= -2 \int_0^1 q'' \delta q dt$$

This gives us the following partial differential equation:

$$d_\tau q = -d_t^2 q$$

Discretize it, we have:

$$\frac{q_{i,j+1} - q_{i,j}}{h} = \frac{q_{i+1,j} - 2q_{i,j} + q_{i-1,j}}{(\Delta t)^2}$$

$$\Leftrightarrow q_{i,j+1} = q_{i,j} + h\frac{q_{i+1,j} - 2q_{i,j} + q_{i-1,j}}{(\Delta t)^2}$$

Which is the explicit method.

2.3.2 Gradient descent with momentum

In the original gradient descent method, the object follows the steepest slope to the local extremum and stays there. While this is straight, it does not reflect the physical dynamic of an descending object as well as it performs poorly in high dimensional problems. Inspired from the Langevin dynamics, we now add a momentum quantity to the dynamic. We have the deterministic Langevin equation:

$$\begin{cases} \delta_\tau q = p \\ \delta_\tau p = -\nabla V(q) - \gamma p \end{cases}$$

2.3.3 Accelerated gradient descent

Equation: The accelerated gradient descent can be modeled with an ODE. From the sequences, we have:

$$x_{k+1} - x_k = \frac{k-1}{k+2}(x_k - x_{k-1}) - s\nabla f(y_k)$$

We can derive the following ODE to model the algorithm:

$$\ddot{X} + \frac{3}{t}\dot{X} + \nabla f(X) = 0$$

The ODE gives us the continuous form of the algorithm in which we can implement the implicit method. Substitute $Y = \dot{X}$, we have the system:

$$\begin{cases} \dot{Y} + \frac{3}{t}Y + \nabla f(X) = 0 \\ \dot{X} = Y \end{cases}$$

From here, we can rewrite the system into a PDE using space-time separation method. We have:

$$\begin{cases} \dot{Y} + \frac{3}{t}Y + \nabla f(X) = 0 \\ \dot{X} = Y \end{cases}$$

This system can be transformed to a PDE as following:

$$\partial_\tau^2 X - \frac{3}{\tau}\partial_\tau X = \partial_t f(X)$$

Where the LHS is the time dimension and the RHS is the space dimension. From this partial differential equation, we can discretize it implicitly as following:

$$X_{i+1} = \left[\frac{L}{k^2} - \frac{(i+3)I_{N-2}}{ih^2}\right]^{-1}\left[\frac{-2iX_i + (i-3)X_{i-1}}{ih^2} + b\right]$$

Where $b(N-1) = \frac{1}{k^2}$. As we can see, the implicit scheme is only dependent on the fictitious time step size.

Algorithm: The accelerated gradient descent is defined by the following sequences:

$$\begin{cases} x_k = y_{k-1} - s\nabla f(y_{k-1}) \\ y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1}) \end{cases}$$

Where $s \leq \frac{1}{L}$ for L is Lipschitz constant of ∇f.

While gradient descent algorithm helps us find the local minimum, it is not the optimal algorithm. In the project, I have also researched about the accelerated gradient descent method, which had been proven to be the optimal algorithm asymptotically. Unlike the original gradient descent algorithm, accelerated gradient descent algorithm does not have the relaxation property, meaning the algorithm is not monotonic after each iteration, but turns out it converges faster than any other algorithm precedes it.

2.4 Stability analysis

In the 5th week, I researched different stability analysis methods including von Neumann analysis and Lyapunov stability analysis. For the sake of the research, I mostly applied von Neumann analysis to the finite difference methods. I find that implicit methods are usually unconditionally stable while explicit methods impose some constraints to the time step.

2.5 Matlab

While I had a rich programming experience prior to the REU, Matlab was a new programming tool that I haven't used yet. In this REU, Professor Tao encouraged me to learn Matlab and it has opened a new window to the world mathematical programming for me.

3 Notable results

3.1 Euler-Lagrange equation

3.1.1 Generalized results

From calculus of variations and chain rule, we can obtain a generalized Euler-Lagrange equation for n-th order derivative as following:

$$\sum_{i=0}^{n}(-1)^i \frac{d^i}{dx^i}\left(\frac{\delta \mathcal{L}}{\delta f^{(i)}}\right) = 0$$

However, not all problems can be rewritten as a solvable Euler-Lagrange equation (for example: $\int_0^1 (q'-q)^2 dt$ gives us $\frac{d}{dt}(q^2 - q'^2) = 0$ which is not trivial to solve). Thus, it is recommended to use calculus of variations instead.

3.1.2 Non-linear and generalized problems

We also look into more complex equations, including problems with arbitrary cost, problems with non-linear terms.

Problem 1: Solve the Euler-Lagrange equation of $\int_0^T (q'' - f(q))^2 dt$ where f is arbitrary.

Solution:

$$\begin{aligned}
&\delta \int_0^T (q'' - f(q))^2 dt \\
&= 2\int_0^T (q'' - f(q))^T \delta(q'' - f(q)) dt \\
&= 2\int_0^T (q'' - f(q))^T \delta q'' - (q'' - f(q))^T \delta f(q)) dt \\
&= 2\int_0^T -(q''' - f'(q)q')^T \delta q' - (q'' - f(q))^T f'(q) \delta q dt \\
&= 2\int_0^T \left[(q^{(4)} - f'(q)q'^2 - f'(q)q'')^T - (q'' - f(q))^T f'(q)\right] \delta q = 0
\end{aligned}$$

From the fundamental lemma and taking transpose both sides, we have:

$$q^{(4)} - f'(q)q'^2 - f'(q)q'' - f'(q)^T(q'' - f(q)) = 0$$

$$\Leftrightarrow q^{(4)} - \left[f'(q) + f'(q)^T\right] q'' - f''(q)q'^2 + f'(q)^T f(q) = 0$$

Note that we have $f''(q)$ is a 3-tensor, which does not have clear definition on how to perform matrix multiplications. Professor Tao recommended to expand everything by elements to avoid confusion.

Problem 2: Heat equation with non-linear terms $d_\tau = d_t^2 q + \sin u$

To solve this problem numerically using implicit method, we only need to write the differentials forward in time while keeping the non-linear term in the current time frame. We have:

$$\frac{q_{i,j+1} - q_{i,j}}{h} = \frac{q_{i+1,j+1} - 2q_{i,j+1} + q_{i-1,j+1}}{k^2} + \sin q_{i,j}$$

We can also use Taylor expansion to linearize the non-linear terms in the equation, such as the following problem:

Problem 3:Solve the partial differential equation

$$\partial_\tau u = \partial_t^2 (\sin u)$$

From the Taylor expansion, we have $\sin u \approx u$ for $u \to 0$. Thus, we can use linear algebra to implement the implicit method.

3.2 Stability and computational complexity of finite difference methods

When I implemented the numerical methods to solve Euler-Lagrange equations, I paid a special attention to computational complexity and the stability of the methods. Surprisingly, the implicit methods are usually unconditionally stable while explicit methods have certain constraints on the time step.

Unsurprisingly, this affects the time complexity of the explicit significantly. In the one dimensional Newtonian problem, von Neumann stability analysis gives us $h \leq \frac{k^4}{8}$ where h, k are the time step and the space step respectively (proof below). Hence, the time complexity of the explicit method can be as high as $O(k^5)$, while the implicit method only costs at most $O(k^3)$.

3.2.1 Stability analysis

Problem: Perform stability analysis on the finite difference method concerning the equation $d_\tau q = -d_t^4 q$.

Lemma: A finite difference scheme is stable if the growth factor G satisfies $|G| \leq 1$.

Explicit method

Back to our finite difference model, let $u_{l,j} = e^{ik\Delta tj}$, we have:

$$q_{l,j+1} = \left(1 - h(e^{2ik\Delta t} + e^{-2ik\Delta t} - 4(e^{ik\Delta t} + e^{-ik\Delta t}) + 6)\right) e^{ik\Delta tj}$$

$$= (1 - 2h(\cos 2k\Delta t - 4\cos k\Delta t + 3))e^{ik\Delta tj}$$

$$= (1 - 2h(2\cos k\Delta t^2 - 4\cos k\Delta t + 2))e^{ik\Delta tj}$$

$$= (1 - 4h(\cos k\Delta t - 1)^2)e^{ik\Delta tj}$$

Thus, we have the growth factor $G = 1 - 4h(\cos k\Delta t - 1)^2$
Since $h > 0, (\cos k\Delta t - 1)^2 \geq 0$, we have $G < 1 \forall h > 0$. To have $|G| \leq 1$, we need to have $G = 1 - 4h(\cos k\Delta t - 1)^2 \geq -1 \Leftrightarrow h \leq \frac{1}{2(\cos k\Delta t - 1)^2}$.
To ensure the inequality holds $\forall k$, we must have $h \leq \frac{k^4}{8}$

Implicit method

Let G be the growth factor, we have the equation:

$$\frac{q_{l,j+1}-q_{l,j}}{\Delta\tau} = -\frac{h}{2}\left(\frac{q_{i+2,j}-4q_{i+1,j}+6q_{i,j}-4q_{i-1,j}+q_{i-2,j}}{(\Delta t)^4} + \frac{q_{i+2,j+1}-4q_{i+1,j+1}+6q_{i,j+1}-4q_{i-1,j+1}+q_{i-2,j+1}}{(\Delta t)^4}\right)$$

$$\Leftrightarrow \frac{G-1}{\Delta\tau} = -\frac{h(G+1)}{2}\left(\frac{4(\cos k\Delta t-1)^2}{(\Delta t)^4}\right)$$

$$\Leftrightarrow \frac{G-1}{\Delta\tau} = -2h(G+1)\left(\frac{(\cos k\Delta t-1)^2}{(\Delta t)^4}\right)$$

$$\Leftrightarrow G = \frac{1-2h(\cos k\Delta t-1)^2}{1+2h(\cos k\Delta t-1)^2}$$

$$\Leftrightarrow |G|<1$$

Hence we have the implicit method is unconditionally stable $\forall h$.

Multi-dimensional stability analysis

Besides using ideas from Fourier transform to perform stability analysis, we can use use eigenvalues to analyze the stability of linear transformations.

Example: Gradient descent with momentum on heat equation.
We have the dynamic:

$$\begin{cases} p_{j+1} = \left(1+\gamma\Delta\tau-\left(\frac{\Delta\tau}{\Delta t}\right)^2\right)^{-1}\left(p_j+\frac{\Delta\tau}{(\Delta t)^2}q_j\right) \\ q_{j+1} = \left[\left(1+\gamma\Delta\tau\right)I_n-\left(\frac{\Delta\tau}{\Delta t}\right)^2 L\right]^{-1}\left[\Delta\tau p_j+\left(1+\gamma\Delta\tau\right)q_j\right] \end{cases}$$

As we can see, the dynamic involves two vectors and we cannot simply use von Neumann to analyze the stability of the sequences. However, the problem is still linear so we can rewrite the dynamic as:

$$\begin{bmatrix} p_{j+1} \\ q_{j+1} \end{bmatrix} = A\begin{bmatrix} p_j \\ q_j \end{bmatrix}$$

Where A is a square matrix.

3.2.2 Sparse matrix algorithms

We can further optimize the time complexity by using sparse matrix algorithms. In the heat equation problem, we can solve in linear time using the tridiagonal linear algorithm as the matrix of the system only consists of three diagonal sub-matrices. However, as we work with the non-linear terms in the partial differential equations, these algorithms will not be sufficient in optimizing the run time.

3.3 Application of advanced gradient descent methods

In this part, we look at the application of advanced gradient descent methods in infinite dimensional problems.

3.3.1 Gradient descent with momentum

In the project, I applied the gradient descent with momentum method to solve the heat equation problem. Discretize the problem, we have:

$$\begin{cases} \frac{q_{i,j+1}-q_{i,j}}{\Delta\tau} = p_{i,j+1} \\ \frac{p_{i,j+1}-p_{i,j}}{\Delta\tau} = \frac{2(q_{i+1,j+1}-2q_{i,j+1}+q_{i-1,j+1})}{(\Delta t)^2} - \gamma p_{i,j+1} \end{cases}$$

$$\Leftrightarrow \begin{cases} p_{j+1} = \left[(1+\gamma\Delta\tau) I_n - 2\left(\frac{\Delta\tau}{\Delta t}\right)^2 L\right]^{-1} \left[p_j + \frac{2\Delta\tau}{(\Delta t)^2} L(q_j + b)\right] \\ q_{j+1} = \left[(1+\gamma\Delta\tau) I_n - 2\left(\frac{\Delta\tau}{\Delta t}\right)^2 L\right]^{-1} \left[\Delta\tau p_j + (1+\gamma\Delta\tau)(q_j + b)\right] \end{cases}$$

Where $b(n) = \frac{h}{\gamma k^2}$.
In the simulation, the plot exhibits many interesting behaviors, such as oscillating around the optimal solution $f(t) = t$.

3.3.2 Accelerated gradient descent

For accelerated gradient descent, I have also applied the algorithm to solve the heat equation. In the original formulation of the algorithm, I used the explicit method as following:

$$\begin{cases} x_k = y_{k-1} - s\nabla f(y_{k-1}) \\ y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1}) \end{cases}$$

$$\Leftrightarrow \begin{cases} x_k = y_{k-1} + sL_2 y_k + b \\ y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1}) \end{cases}$$

Where L_2 is the tridiagonal matrix to represent second order differentials. Note that we need the slack value to be $b(N) = \frac{s}{k^2}$ in order for the function to be smooth. If we use the continuous form of the Nesterov gradient descent, we can rewrite the algorithm implicitly using the characteristic partial differential equation as following:

$$\partial_\tau^2 X - \frac{3}{\tau}\partial_\tau X = \partial_t f(X)$$

$$\Leftrightarrow \frac{X_{i+1,j} - 2X_{i,j} + X_{i-1,j}}{\tau^2} - \frac{3}{2i\tau^2}(X_{i+1,j} - X_{i-1,j}) = \partial_t f(X)$$

Where the RHS is written forward in time. From this discretization of the partial differential equation, we can rewrite it implicitly as following:

$$X_{i+1} = \left[\frac{L_2}{k^2} - \frac{(i+3)I_{N-2}}{ih^2}\right]^{-1} \left[\frac{-2iX_i + (i-3)X_{i-1}}{ih^2} + b\right]$$

Where $b(N) = \frac{1}{k^2}$. As we can see, the implicit scheme is only dependent on the fictitious time step size and we can choose any value for the fictitious time step without any restriction (unlike the explicit scheme).

3.3.3 Comparing performances of gradient descent algorithms

In the Nesterov's gradient descent algorithm, we use a rescaled time step. That is, we have the total time $t \approx k\sqrt{s}$ for k is the number of iteration. Apply to the heat equation, we have:

h	k	Normal GD	GDwM	AGD	AGD (rescaled)	AGD (implicit)
0.01	0.01	0.7	1.2	15	0.03	30^+
0.1	0.01	0.7		150	0.03	5
0.5	0.01					3.5

From the result, we can see that the gradient descent with momentum performs no better or even worse than the normal gradient descent. We also find a good gamma in order to have a sufficiently good result. On the other hand, the accelerated gradient descent outperforms both algorithms by a wide margin if we consider the rescaled time step, which is an encouraging result for future research.

Comparing between different implementations of the accelerated gradient descent, different choice of h, k can alter the performance. This could be an area that we will look further.

4 Research directions

4.1 Accelerated gradient descent

Regarding future research directions, Langevin dynamics or Nesterov gradient descent in infinite dimensional problem would be a potential area of research. While it is known that these algorithms perform better in high-dimensional problems, it is still unclear if they outperform in infinite dimensional problems, such as solving the Euler-Lagrange equation. Preliminary results with the heat equation have shown that the accelerated gradient has great potential as it outperforms other gradient descent algorithm by a wide margin. However, we haven't touched on non-linear problems, which are essential to our research problem. This shall be an area of research that we will look into in the near future.

4.2 Spaceship trajectory optimization

In our project, we work specifically on the Euler-Lagrange equations to find the optimal control policy for the drone in finite dimension. During the REU, we worked on the two dimensional Newtonian dynamic problem with a large solar body:

$$x'' = -\frac{x}{(x^2+y^2)^{\frac{3}{2}}} + u_x$$

$$y'' = -\frac{y}{(x^2+y^2)^{\frac{3}{2}}} + u_y$$

Minimize:

$$\int_0^T \left(u_x^2 + u_y^2\right) dt$$

With such a complex dynamic, it is imperative for us to employ numerical methods in order to solve it. The problem also poses serious implementation challenges as it has linear terms in the equations, which makes implementing with implicit method extremely hard.

Similarly, we can add more planets to have a more complicated dynamic. For example, if we wish to launch a drone from Earth to Moon, then at least three bodies must be included: the Sun, Earth and Moon.

5 Appendix

Biweekly reports

`https://github.com/HoangT1215/Numerical-methods/tree/master/Reports`

Keywords

- Calculus of variations
- Partial differential equations
- Finite difference methods
- Stability analysis
- Gradient descent
- Heavy ball method

6 Photos

Here are some photos during my stay at Atlanta.

Figure 1: Stone moutain hike with Professor Tao. Left to right: Hoang, Gabriell, Professor Tao

Figure 2: Stone moutain hike with Professor Tao. Left to right: Hoang, Andrew, Professor Tao

Figure 3: Pizza lunch with Georgia Tech Professors and other REU students