Optimization Algorithms
Game Theoretic Optimization
Puzzle 1: Prisoner’s Dilemma

- If both of them remain silent they will get 1 year of Jail Time.
- If one of them confesses and the other one does not, one who confesses get out immediately while the other one gets 20 years of Jail Time.
- If they both confess they will get 5 years of Jail Time each.
- What will they do?

Ref: tps://optimization.mccormick.northwestern.edu/index.php/Applying_Optimization_in_Game_Theory
● The **Nash Equilibrium** here is the action profile that both confess and take 5 years of Jailtime.

● But optimum solution is both remain silent and get only 1 year of Jailtime.

● Why did they deviate?

● Neither of them has the incentive to deviate from this profile because the person who remains silent would be worse off.

● Formally speaking,

\[
\text{if } x_i \in S_i \text{ is the strategy profile for player } i, \ x_{-i} \text{ is the strategy profiles for all the players except player } i, \text{ and } f_i \text{ is the player’s payoff function, then a strategy profile that contains the strategies of all players } x^* \text{ is a Nash Equilibrium so long as } \\
\forall i, x_i \in S_i : f_i (x_i, x^*_i) \geq f_i (x_i, x^*_{-i}).
\]

● One of the algorithms that states whether there exists a Nash Equilibrium and if it exists solves for it is **Iterated Removal of Strictly Dominated Strategies (IRSDS)** or **Iterated Elimination of Strictly Dominated Strategies (IESDS)**.

● **Strictly Dominated Strategies (Def):** Strategy X strictly dominates strategy Y for a player if X gives a bigger (more preferred) payoff than Y no matter what the other players do. Players never rationally choose strictly dominated strategies.

● IESDS iteratively removes Dominated Strategies. In the first step, at most one dominated strategy is removed from the strategy space of each of the players since no rational player would ever play these strategies. This results in a new, smaller game. Some strategies—that were not dominated before—may be dominated in the smaller game. The first step is repeated, creating a new even smaller game, and so on.

● If, after completing this process, there is only one strategy for each player remaining, that strategy set is the unique Nash equilibrium.

● Let’s apply IESDS on Puzzle 1.

Ref: https://en.wikipedia.org/wiki/Strategic_dominance
Player 1 will think if Player 2 confesses he will choose to confess as the penalty is -5. If Player 2 chooses to keep quiet Player 1 will choose to confess as its penalty is 0. Thus keeping quiet is a dominated strategy and we remove it as no rational player will choose this.
Now Player 2 will think if Player 1 confesses he has to choose confession as it’s penalty is -5. If Player 1 keeps quiet he has to choose confess as the penalty is even greater. Thus Keeping Quiet is a dominated strategy and must be removed.
Thus both players or prisoners will confess.
Procedure: Iterated Removal of Strictly Dominated Strategies (IRSDS)

Input: \( D = (D_1, \ldots, D_n) \): profile of domains

Output: Updated domains, or failure

repeat

\( \text{changed} \leftarrow \text{false} \)

for all \( i \in N \) do

for all \( a_i \in d_i \in D_i \) do

for all \( a'_i \in A_i \) do

if \( \forall a_{-i} \in d_{-i} \in D_{-i}, u_i(a_i, a_{-i}) < u_i(a'_i, a_{-i}) \)

then

\( D_i \leftarrow D_i - \{d_i \in D_i : a_i \in d_i\} \)

\( \text{changed} \leftarrow \text{true} \)

if \( D_i = \emptyset \) then

return failure

until \( \text{changed} = \text{false} \)

return \( D \)
Two mountain hotels in the same district, Krakonosˇ and Trautenberg, compete for tourists from three different countries: Germany, Czech Republic and Poland. The capacity of both hotels is sufficient for the accommodation of all tourists in only one of them. Both hotels have financial resources for the advertising campaign in only one country, the effectiveness of their campaigns is the same. If only one hotel runs the campaign in a given country, it gains all tourists from this country and hence the following profit: in the case of Germany 150 thousand EUR, in the case of the Czech Republic 90 thousand EUR and in the case of Poland 72 thousand EUR. If both firms run the campaign in the same country, receives each of them the half of the profit from this country’s customers, similarly in the case that no firm runs the campaign in a specified country. What are the optimal strategies for both hotels?
### Application 1: Business Advertising

<table>
<thead>
<tr>
<th>Country</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany</td>
<td>150</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>90</td>
</tr>
<tr>
<td>Poland</td>
<td>72</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>312</strong></td>
</tr>
</tbody>
</table>

**Trautenberg**

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Germany</th>
<th>Czech Rep</th>
<th>Poland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany</td>
<td>(156, 156)</td>
<td>(186, 126)</td>
<td>(195, 117)</td>
</tr>
<tr>
<td>Krakonoš Czech Rep.</td>
<td>(126, 186)</td>
<td>(156, 156)</td>
<td>(165, 147)</td>
</tr>
<tr>
<td>Poland</td>
<td>(117, 195)</td>
<td>(147, 185)</td>
<td>(156, 156)</td>
</tr>
</tbody>
</table>

Puzzle 2: Pirates’ Treasure

- There are five rational pirates (in strict order of seniority A, B, C, D and E) who found 100 gold coins. They must decide how to distribute them.
- The pirate world's rules of distribution say that the most senior pirate first proposes a plan of distribution. The pirates, including the proposer, then vote on whether to accept this distribution. If the majority accepts the plan, the coins are dispersed and the game ends.
- If the majority rejects the plan, the proposer is thrown overboard from the pirate ship and dies, and the next most senior pirate makes a new proposal to begin the system again.
- In case of a tie vote, the proposer has the casting vote.
- The process repeats until a plan is accepted or if there is one pirate left.

Ref: https://en.wikipedia.org/wiki/Pirate_game
Puzzle 2: Pirates’ Treasure

- Prisoners’ Dilemma is an example of a ‘static’ game where the players enacted simultaneously.
- Whereas Pirate’s game is an example of ‘dynamic’ game where the players will act sequentially and where each player’s information about earlier moves are recorded in detail.
- These types of games can be represented as a directed tree and is known as Extensive Game.

- A subgame of an extensive game is the subtree of the game tree that includes all the information sets containing the node of the subtree.
- This type of games induces Nash Equilibrium in every subgame and termed as subgame perfect equilibrium. Note that this equilibrium is not unique.
- Backward Induction is used to solve the games with subgame perfect equilibrium.

Ref: https://en.wikipedia.org/wiki/Pirate_game
Algorithm 1 Subgame Perfect Equilibrium

**Input:** An Extensive game

**Output:** A subgame perfect Nash Equilibrium of the game

1: begin
2: Consider, in increasing order of inclusion, each subgame of the game
3: Find a Nash Equilibrium of the subgame
4: Replace the subgame by a new terminal node that has equilibrium payoffs
5: end

This algorithm while applied to puzzle 2 gives
Puzzle 2: Pirates’ Treasure

- The final possible scenario would have all the pirates except D and E thrown overboard. Since D is senior to E, he has the casting vote; so, D would propose to keep 100 for himself and 0 for E.
- If there are three left (C, D and E), C knows that D will offer E 0 in the next round; therefore, C has to offer E one coin in this round to win E's vote. Therefore, when only three are left the allocation is C:99, D:0, E:1.
- If B, C, D and E remain, B can offer 1 to D; because B has the casting vote, only D's vote is required. Thus, B proposes B:99, C:0, D:1, E:0.

(In the previous round, one might consider proposing B:99, C:0, D:0, E:1, as E knows it won't be possible to get more coins, if any, if E throws B overboard. But, as each pirate is eager to throw the others overboard, E would prefer to kill B, to get the same amount of gold from C.)

- With this knowledge, A can count on C and E's support for the following allocation, which is the final solution:

(A:98, B:0, C:1, D:0, E:1)

Ref: https://en.wikipedia.org/wiki/Pirate_game
Optimization Algorithms
Steepest Descent
Quadratic form

**Definition:** A quadratic form is a scalar function $f: \mathbb{R}^n \to \mathbb{R}$ with the form

$$f(x) = \frac{1}{2} x^T A x - b^T x + c$$

where $A$ is a matrix, $x$ and $b$ are column vectors and $c$ is a scalar constant.

**Definition:** A $n \times n$ real matrix $A$ is **positive-definite** if $x^T A x > 0$ for all nonzero $x$.

**Proposition:** If $A$ is symmetric and positive-definite, the corresponding quadratic form $f(x)$ is minimized by the solution to $A x = b$. 
Quadratic form

\[ z = x^2 + y^2 \]

\[ z = x^2 \]

\[ z = x^2 - y^2 \]
Quadratic form

**Definition:** The gradient of a function $f(x)$ with multiple variables is defined to be

$$f'(x) = \left( \frac{\partial}{\partial x_1} f(x), \frac{\partial}{\partial x_2} f(x), \ldots, \frac{\partial}{\partial x_n} f(x) \right)^T.$$

**Proposition:** The gradient points to the direction of greatest increase of $f(x)$.

**Proposition:** If $A$ is symmetric, the gradient of the quadratic form is $f'(x) = Ax - b$. 
The method of Steepest Descent

We want to minimize the quadratic form.

The steepest descent is an iterative algorithm.

1. Start at an arbitrary point
2. Repeat the following until we get a satisfying solution
3. Choose a direction in which $f$ decreases most quickly
4. Take a step in that direction
The method of Steepest Descent

Let $x_{(i)}$ denote the solution we have at the $i$th iteration and $x$ be the actual solution.

Let the error be $e_{(i)} = x_{(i)} - x$ and the residual be $r_{(i)} = -Ae_{(i)} = -f'(x_{(i)}) = b - Ax_{(i)}$

(the direction of steepest descent).

Let $\alpha_{(i)}$ be the step we take at each iteration.

1. Start at an arbitrary point $x_{(0)}$

2. Repeat the following until we get a satisfying solution

3. $x_{(i+1)} = x_{(i)} + \alpha_{(i)}r_{(i)}$
Line search

1. Start at an arbitrary point $x_{(0)}$

2. Repeat the following until we get a satisfying solution

3. Choose $\alpha_{(i)}$ to minimize $f$ along the line $r_{(i)}$ is on.

4. $x_{(i+1)} = x_{(i)} + \alpha_{(i)} r_{(i)}$

Just let $\frac{d}{d\alpha_{(i)}} f(x_{(i+1)}) = 0$. Implying that $r_{(i+1)}$ and $f'(x_{(i)})$ are orthogonal.

And $\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T Ar_{(i)}}$. 
Steepest Descent

1. Start at an arbitrary point $x_{(0)}$

2. Repeat the following until we get a satisfying solution

3. $r_{(i)} = b - Ax_{(i)}$

4. $\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T Ar_{(i)}}$

5. $x_{(i+1)} = x_{(i)} + \alpha_{(i)} r_{(i)}$

From 5, if we multiply both sides by $-A$ and add $b$, we’ll have $r_{(i+1)} = r_{(i)} - \alpha_{(i)} Ar_{(i)}$. This can reduce the number of matrix multiplications because we can calculate $Ar_{(i)}$ for both 4 and the update of $r$. But the accumulation of floating point round-off error may cause $x_{(i)}$ to converge to some point near $x$. 
Steepest Descent

Given the inputs $A$, $b$, a starting $x$, a max number of iterations $i_{\text{max}}$, and an error tolerance $\varepsilon < 1$.

This algorithm terminates when the maximum number of iterations $i_{\text{max}}$ has been exceeded, or when $\|r_{(i)}\| \leq \varepsilon \|r_{(0)}\|$.

We’ll use the fast recursive formula for the residual, but we calculate the exact residual occasionally to remove the accumulated floating point error.

\[
i \leftarrow 0 \\
r \leftarrow b - Ax \\
\delta \leftarrow r^T r \\
\delta_0 \leftarrow \delta \\
\text{While } i < i_{\text{max}} \text{ and } \delta > \varepsilon^2 \delta_0 \text{ do} \\
\quad q \leftarrow Ar \\
\quad \alpha \leftarrow \frac{\delta}{r^T q} \\
\quad x \leftarrow x + \alpha r \\
\quad \text{If } i \text{ is divisible by 50} \Rightarrow \quad r \leftarrow b - Ax \\
\quad \text{else} \quad r \leftarrow r - \alpha q \\
\quad \delta \leftarrow r^T r \\
i \leftarrow i + 1
\]
Steepest Descent

An example of steepest descent in 2D
Convergence

Define $\|e(i)\|_A^2 = e(i)^T A e(i)$ to be the energy norm of error.

We hope to bound $\frac{\|e(i+1)\|_A^2}{\|e(i)\|_A^2}$ by a constant $w < 1$. Then we will have $\frac{\|e(0)\|_A^2}{\|e(0)\|_A^2} \leq w^i$ and $\frac{f(x(i)) - f(x)}{f(x(0)) - f(x)} \leq w^{2i}$. 
Convergence

**Proposition:** An $n \times n$ symmetric matrix has $n$ orthogonal unit eigenvectors.

**Proposition:** The eigenvalues of a positive-definite matrix are positive.

Since we are only considering the case when $A$ is symmetric and positive-definite, we can express $e_{(i)}$, $r_{(i)}$, $\alpha_{(i)}$ in terms of the eigenvectors and eigenvalues. Further analysis show that

$$\frac{\|e_{(i+1)}\|_A^2}{\|e_{(i)}\|_A^2} \leq \frac{k - 1}{k + 1} < 1$$

where $k$ is the ratio between the max and min eigenvalues, which is also called the condition number.
Quadratic form

\[ \frac{\|e_{(i+1)}\|_A^2}{\|e_{(i)}\|_A^2} \leq \frac{k - 1}{k + 1} < 1 \]

Convergence of Steepest Descent (per iteration) worsens as the condition number of the matrix increases.

A large condition number means that the matrix is close to being singular.

Positive-definite matrix is always nonsingular.
Convergence

**Proposition:** An $n \times n$ symmetric matrix has $n$ orthogonal unit eigenvectors.

**Proposition:** The eigenvalues of a positive-definite matrix are positive.

Since we are only considering the case when $A$ is symmetric and positive-definite, we can express $e_{(i)}$, $r_{(i)}$, $a_{(i)}$ in terms of the eigenvectors and eigenvalues. Further analysis show that

$$\frac{\|e_{(i+1)}\|^2_A}{\|e_{(i)}\|^2_A} \leq \frac{k - 1}{k + 1} < 1$$

where $k$ is the ratio between the max and min eigenvalues, which is also called the condition number.
Optimization Algorithms
Conjugate Gradient Method
Why do we need Conjugate Gradient Method at all?
Steepest Descent often finds itself taking steps in the same direction as earlier steps.

Intuition of Conjugate Gradient Method:
Each time when we take a step on a search direction, we only want to take exactly one step on that direction, and never back to it again. “And that step will be just the right length to line up evenly with x.”

Thus, if we have n search directions, $d_{(0)}, d_{(1)}, \ldots, d_{(n-1)}$, after n steps, we are done.
What if the search directions, $d_{(0)}, d_{(1)}, \ldots, d_{(n-1)}$, are orthogonal to each other?

For each step, we are computing:

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)}d_{(i)}$$

Notice that $e_{(i+1)}$ should be orthogonal to $d_{(i)}$ (so that we will never step back into $d_{(i)}$ again). Thus, we have:

$$d_{(i)}^T e_{(i+1)} = 0$$

(by Equation 29)

$$d_{(i)}^T (e_{(i)} + \alpha_{(i)}d_{(i)}) = 0$$

$$\alpha_{(i)} = -\frac{d_{(i)}^T e_{(i)}}{d_{(i)}^T d_{(i)}}$$

From this equation, to compute $\alpha_{(i)}$, we need to know $e_{(i)}$. But if we already know $\alpha_{(i)}$, the problem has already been solved. Thus, orthogonal search directions don’t help anything.
Instead of orthogonal search directions, **A-orthogonal (conjugate)** search directions are used.

**What is A-orthogonal?**

Two vectors $d_{(i)}$ and $d_{(j)}$ are A-orthogonal if,

$$d_{(i)} \Rightarrow d_{(i)}^T A d_{(j)} = 0.$$ 

Now, the new requirement is that $e_{(i+1)}$ be A-orthogonal to $d_{(i)}$, i.e.

$$d_{(i)}^T A e_{(i+1)} = 0.$$ 

Now to compute $\alpha_{(i)}$ we have

$$\alpha_{(i)} = \frac{d_{(i)}^T A e_{(i)}}{d_{(i)}^T A \alpha_{(i)}} = \frac{d_{(i)}^T r_{(i)}}{d_{(i)}^T d_{(i)}}.$$ 

Thus $\alpha_{(i)}$ can be computed, without the knowledge of $e_{(i)}$.

Seems like $e(i)$ and $r(i)$ are “equivalent”? Not True
Now the question becomes:

**Why follow these** \( n \) **A-orthogonal search directions,** \( d_{(0)}, d_{(1)}, \ldots, d_{(n-1)} \), **the computation of** \( x \) **will be done in** \( n \) **steps?**

Express the error term as a linear combination of the search directions:

\[
e_{(0)} = \sum_{j=0}^{n-1} \delta_j d_{(j)}
\]

Then multiply both sides by \( d_{(k)}^T A :\)

\[
\delta_{(k)} = \frac{d_{(k)}^T A e_{(0)}}{d_{(k)}^T A d_{(k)}}
\]

\[
d_{(k)}^T A e_{(0)} = \sum_j \delta_j d_{(k)}^T A d_{(j)} = \frac{d_{(k)}^T A (e_{(0)} + \sum_{i=0}^{k-1} \alpha_{(i)} d_{(i)})}{d_{(k)}^T A d_{(k)}}
\]

Fact: \( \alpha_{(i)} = -\delta_{(i)} \)

\[
e_{(i)} = e_{(0)} + \sum_{j=0}^{i-1} \alpha_{(j)} d_{(j)} = \sum_{j=0}^{n-1} \delta_{(j)} d_{(j)} - \sum_{j=0}^{i-1} \delta_{(j)} d_{(j)} = \sum_{j=i}^{n-1} \delta_{(j)} d_{(j)}.
\]
Now, how do we get a set of A-orthogonal search directions $d_{(0)}, d_{(1)}, \ldots, d_{(n-1)}$? The method is called \textit{conjugate Gram-Schmidt process}.

Suppose we have a set of $n$ linearly independent vectors $u_0, u_1, \ldots, u_{n-1}$, e.g. the coordinate axes. “To construct $d_{(i)}$, take $u_i$ and subtract out any components that are not A-orthogonal to the previous $d$ vectors.” Set $d_{(0)} = u_0$, and for $i > 0$ set

$$d_{(i)} = u_i + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}$$

$$d_{(i)}^T A d_{(j)} = u_i^T A d_{(j)} + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}^T A d_{(j)}$$

$$0 = u_i^T A d_{(j)} + \beta_{ij} d_{(j)}^T A d_{(j)}, \quad i > j$$

$$\beta_{ij} = -\frac{u_i^T A d_{(j)}}{d_{(j)}^T A d_{(j)}}$$

Follow this way, we have to compute every $\beta_{ij}$, which can be time and space consuming. Can we do better?
Fact: \( r_{(i)} \) is orthogonal to \( D_i \), where \( D_i \) is the subspace spanned by \( d_{(0)}, d_{(1)}, \ldots, d_{(i-1)} \).

Proof: Multiply both sides by \(-d_{(i)}^T A\),

\[
c_{(i)} = c_{(0)} + \sum_{j=0}^{i-1} \alpha_{(j)} d_{(j)}
\]
\[
= \sum_{j=0}^{n-1} \delta_{(j)} d_{(j)} - \sum_{j=0}^{i-1} \delta_{(j)} d_{(j)}
\]
\[
= \sum_{j=i}^{n-1} \delta_{(j)} d_{(j)}.
\]

\[-d_{(i)}^T A e_{(j)} = -\sum_{j=i}^{n-1} \delta_{(j)} d_{(i)}^T A d_{(j)}
\]
\[
d_{(i)}^T r_{(j)} = 0, \quad i < j \quad \text{(by } A\text{-orthogonality of } d\text{-vectors)}.
\]

Thus, \( r_{(i)} \) is also orthogonal to all previous \( u \) vectors.

\[
d_{(i)} = u_{(i)} + \sum_{k=0}^{i-1} \beta_{i,k} d_{(k)}
\]
\[
d_{(i)}^T r_{(j)} = u_{(i)}^T r_{(j)} + \sum_{k=0}^{i-1} \beta_{i,k} d_{(k)}^T r_{(j)}
\]
\[
0 = u_{(i)}^T r_{(j)}, \quad i < j
\]
\[
d_{(i)}^T r_{(i)} = u_{(i)}^T r_{(i)}.
\]

A way to compute \( r_{(i)} \) iteratively.

\[
r_{(i+1)} = -A e_{(i+1)}
\]
\[
= -A(e_{(i)} + \alpha_{(i)} d_{(i)})
\]
\[
= r_{(i)} - \alpha_{(i)} A d_{(i)}.
\]
Now, **how do we make improvements to the conjugate Gram-Schmidt process.**

This answer is to construct search directions based on the conjugation of the residuals, i.e. set \( u_i = r_i \).

Now equation 0 \( = u_i^T r_j \) becomes \( r_j^T r_j = 0 \).

From the iterative generation of each new residual \( r_j \) we can view it as a linear combination of the previous residual and \( Ad_{i-1} \). Thus, we have (this **Krylov subspace**),

\[
D_i = \text{span}\{d_0, Ad_0, A^2d_0, \ldots, A^{i-1}d_0\} = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{i-1}r_0\}.
\]

Because \( AD_i \) is included in \( D_{i+1} \), the fact that \( r_{i+1} \) is orthogonal to \( D_{i+1} \) implies that \( r_{i+1} \) is A-orthogonal to \( D_i \). The **Gram-Schmidt process** becomes easy because \( r_{i+1} \) is already A-orthogonal to all previous search directions except \( d_i \).

Now, somehow we can simplify the computation of \( \beta_{ij} \).

\[
\beta_{ij} = -\frac{u_i^T Ad_{j}}{d_i^T Ad_{j}} = \begin{cases} \frac{1}{\alpha_{i(j)}} r_i^T r_j, & i = j; \\ \frac{1}{\alpha_{i(i-1)}} r_i^T r_{i-1}, & i = j + 1; \\ 0, & \text{otherwise}. \end{cases}
\]

\[
\beta(i) = \begin{cases} \frac{r_i^T r_i}{d_{(i-1)}^T r_{i-1}}, & i = j + 1; \\ \frac{r_i^T r_{i-1}}{d_{(i-1)}^T r_{i-1},} & i > j + 1. \end{cases}
\]
The complete Conjugate Gradient Method algorithm.

\[ d_0 = r_0 = b - Ax_0, \]

\[ \alpha(i) = \frac{r^T(i) r(i)}{d^T(i) Ad(i)} \quad \text{(by Equations 32 and 42)}, \]

\[ x(i+1) = x(i) + \alpha(i) d(i), \]

\[ r(i+1) = r(i) - \alpha(i) Ad(i), \]

\[ \beta(i+1) = \frac{r^T(i+1) r(i+1)}{r^T(i) r(i)}, \]

\[ d(i+1) = r(i+1) + \beta(i+1) d(i). \]

Time Complexity:

Steepest Descent: \( \mathcal{O}(mk) \)

Conjugate Gradient: \( \mathcal{O}(m\sqrt{\kappa}) \)

Space Complexity:

Both: \( \mathcal{O}(m) \)

Where \( m \) is the number of non-zero elements in \( A \), \( \kappa \) is the ratio of the largest eigenvalue of \( A \) against the smallest.