

# Lab 7: Parking Garage Case Study

Andres Calvo (ac228)

Wed., Mar. 27

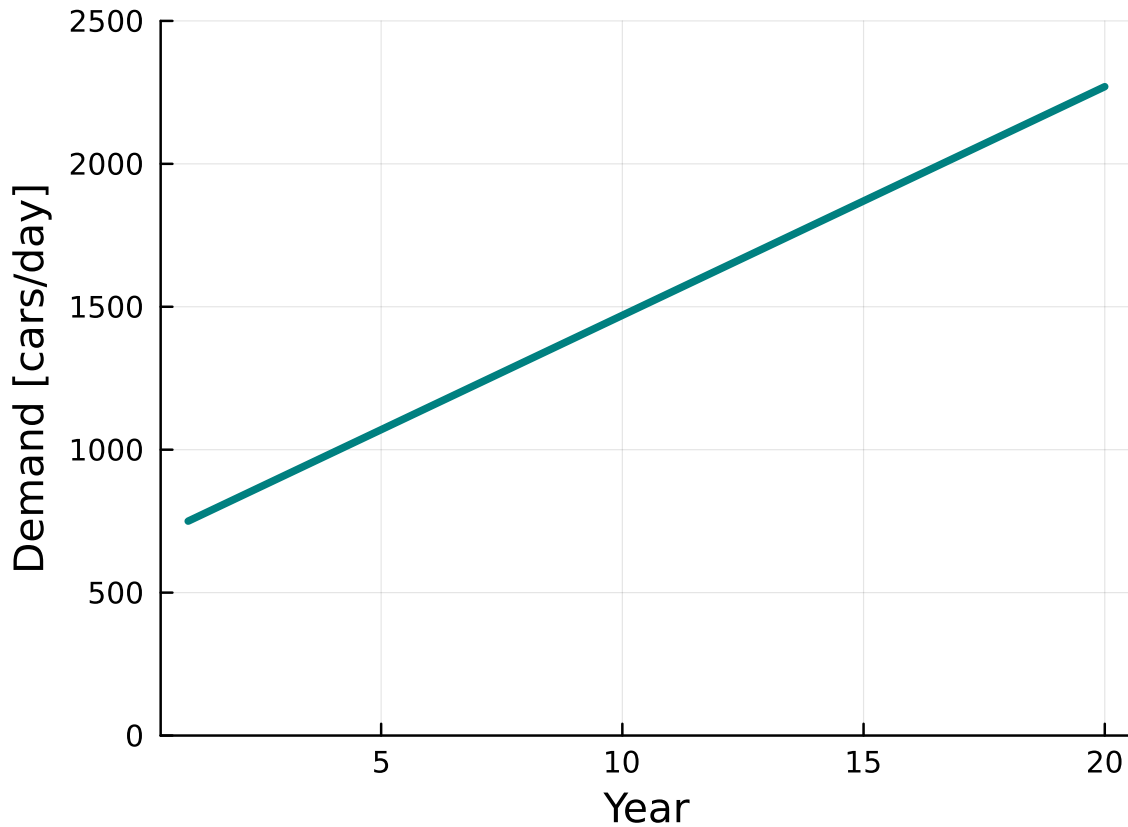
```
1 using Revise
2 using ParkingGarage
3
4 using Plots
5 using Distributions
6
7 Plots.default(; margin=5Plots.mm)
```

## 1 Deterministic analysis

The deterministic analysis uses a **linear** demand model and constant discount rate and time frame. The following is the based demand growth.

```
1 let
2     sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
3     years = 1:(sow.n_years)
4     demand = [
5         ParkingGarage.calculate_demand(year, sow.demand_growth_rate) for year in years
6     ]
7     plot(
8         years,
9         demand;
10        ylabel = "Demand [cars/day]",
11        ylims = [0,2500],
12        xlabel = "Year",
13        legend = false,
14        title = "Demand Growth Rate: $(sow.demand_growth_rate) Cars/Year",
15        size = (500, 400),
16        color = "teal",
17        linewidth = 3,
18    )
19 end
```

## Demand Growth Rate: 80.0 Cars/Year



With the demand growth model, a simulation can be performed using a time horizon of 20 years and a discount rate of 12%. Those are considered as constant variables through the analysis considering that they belong to the financial domain exclusively and are probably standards in the “investment” community or they would want to see the simulation results based on those parameters to make decisions. Nevertheless, for visualization purposes two discount rates are simulated to understand the implications.

The simulation uses a *static policy* that add 0 new levels for every time step regardless of the demand quantities. The following figure shows the Net Present Value (NVP) for every design structure level. For preliminary sensitive analysis, growth rates of 60 and 100 are also simulated.

```
1 let
2   level_Δ_a = 0
3   sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
4   n_levels = 2:12
5   policies = [StaticPolicy(i) for i in n_levels]
6   profits = [simulate(sow, level_Δ_a, policy) for policy in policies]
7   pl = plot()
8   hline!([0]; color = "orangered", linewidth = 3, label = nothing)
9   plot!(
```

```

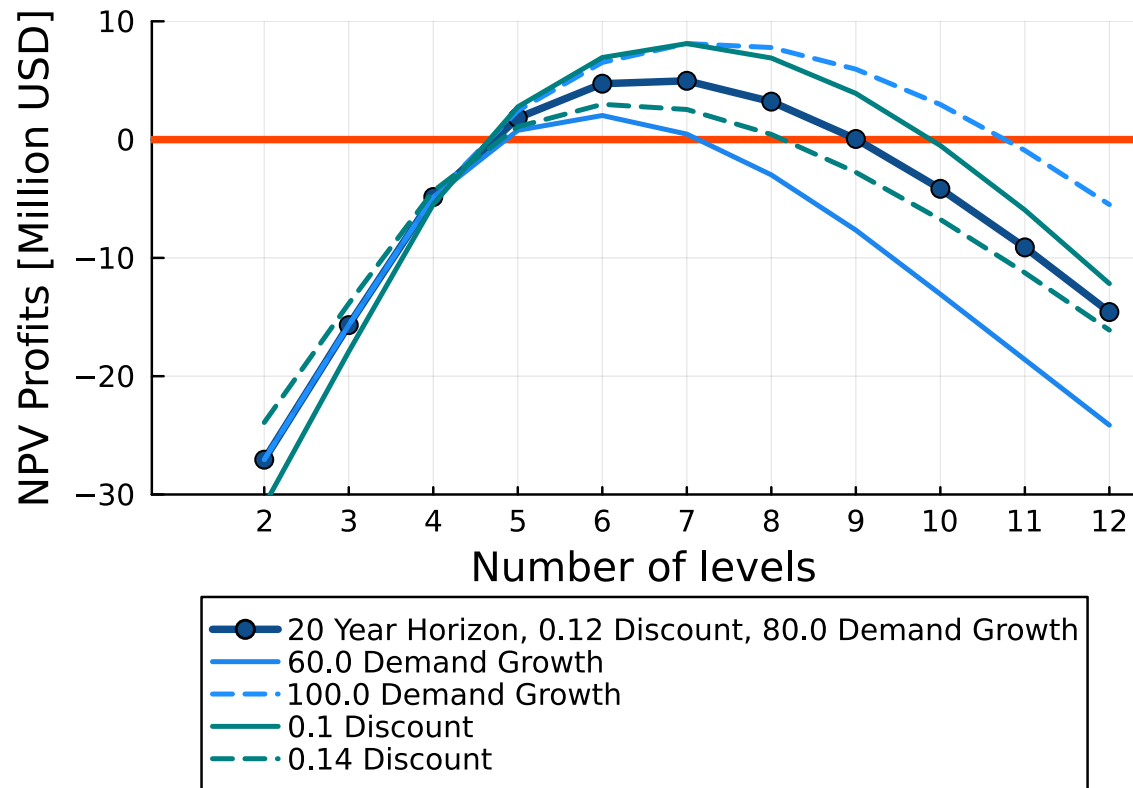
10     n_levels,
11     profits;
12     ylabel = "NPV Profits [Million USD]",
13     ylims = [-30,10],
14     xlabel = "Number of levels",
15     legend = :outerbottom,
16     label = "$ (sow.n_years) Year Horizon, $(sow.discount_rate) Discount, $(sow.demand_grow",
17     size = (500, 400),
18     marker = :circle,
19     xticks = n_levels,
20     color = "dodgerblue4",
21     linewidth = 3,
22 )
23 sow = ParkingGarageSOW(; demand_growth_rate=60.0, n_years=20, discount_rate=0.12)
24 policies = [StaticPolicy(i) for i in n_levels]
25 profits = [simulate(sow, level_A_a, policy) for policy in policies]
26 plot!(
27     n_levels,
28     profits;
29     label = "$ (sow.demand_growth_rate) Demand Growth",
30     color = "dodgerblue2",
31     linewidth = 2,
32 )
33 sow = ParkingGarageSOW(; demand_growth_rate=100.0, n_years=20, discount_rate=0.12)
34 policies = [StaticPolicy(i) for i in n_levels]
35 profits = [simulate(sow, level_A_a, policy) for policy in policies]
36 plot!(
37     n_levels,
38     profits;
39     label = "$ (sow.demand_growth_rate) Demand Growth",
40     style = :dash,
41     color = "dodgerblue",
42     linewidth = 2,
43 )
44 sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.10)
45 policies = [StaticPolicy(i) for i in n_levels]
46 profits = [simulate(sow, level_A_a, policy) for policy in policies]
47 plot!(
48     n_levels,
49     profits;
50     label = "$ (sow.discount_rate) Discount",
51     color = "teal",
52     linewidth = 2,
53 )
54 sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.14)
55 policies = [StaticPolicy(i) for i in n_levels]
56 profits = [simulate(sow, level_A_a, policy) for policy in policies]

```

```

57 plot!(
58     n_levels,
59     profits;
60     label = "$(sow.discount_rate) Discount",
61     color = "teal",
62     style = :dash,
63     linewidth = 2,
64 )
65 end

```



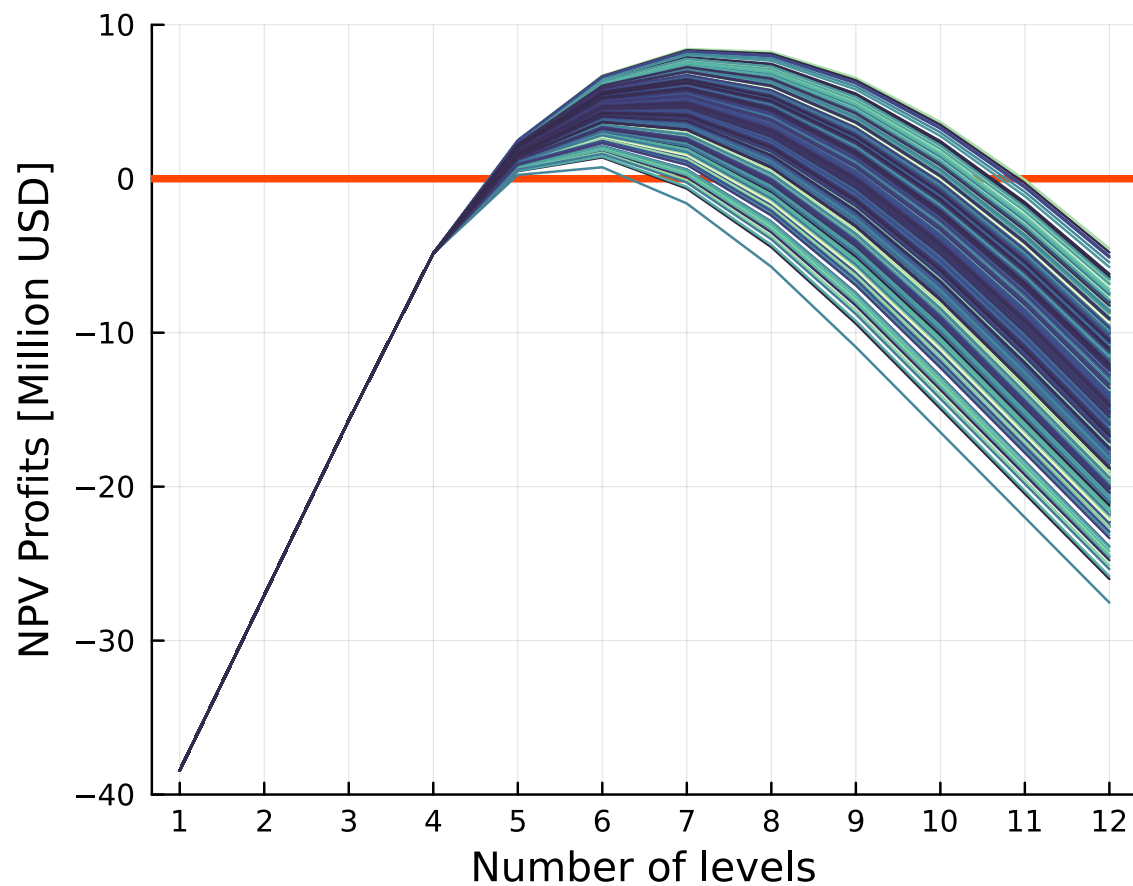
## 2 Uncertainty consideration

The uncertainty is to be propagated for the demand growth model. The annual rate is to be sampled from a *Normal* distribution with of 80 and a standard deviation as a parameter of the coefficient of variation (COV). This way, the level of uncertainty can be propagated in the model. As mentioned before, the discount rate and the time horizon are kept constant (12% and 20 years, respectively). The following is a simulation for a  $COV = 10\%$ .

```

1 function draw_growth_rate(COV)
2     = 80.0
3     return rand(Normal( , COV * ))
4 end
5 let
6     level_Δ_a = 0
7     COV = 0.10
8     n_levels = 1:12
9     N_samples = 1000
10
11     pl = plot(;
12         ylabel = "NPV Profits [Million USD]",
13         ylims = [-40,10],
14         xlabel = "Number of levels",
15         legend = false,
16         size = (500, 400),
17         xticks = n_levels,
18         linewidth = 1,
19         alpha = 0.2,)
20     hline!([0]; color = "orangered", linewidth = 3, label = nothing)
21     for s in 1:N_samples
22         sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(COV),
23                                 n_years = 20,
24                                 discount_rate = 0.12)
25         policies = [StaticPolicy(i) for i in n_levels]
26         profits = [simulate(sow, level_Δ_a, policy) for policy in policies]
27
28         plot!(n_levels,
29             profits;
30             palette = :deep
31         )
32     end
33     pl
34 end

```



In order to compare to the deterministic/static case, three different levels of uncertainty (i.e. COV = 10, 25, and 50%) are used. This way, it is clear that the greater the uncertainty of the demand model is the lower expected returns are simulated. The following figure represent the average NPV for a 1000 sampling simulation.

```

1  let
2      level_Δ_a = 0
3      covs = [0.10, 0.25, 0.50]
4      N_samples = 1000
5      n_levels = 2:12
6
7      pl = plot(;
8          ylabel = "NPV Profits [Million USD]",
9          y_ticks = -30:5:10,
10         ylims = [-30,10],
11         xlabel = "Number of levels",
12         legend = :outerbottom,
13         size = (500, 400),
14         xticks = n_levels,
15         linewidth = 3,

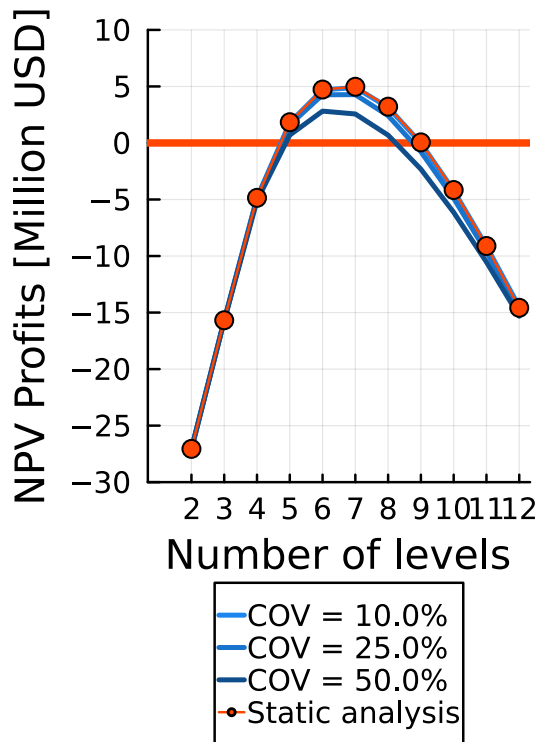
```

```

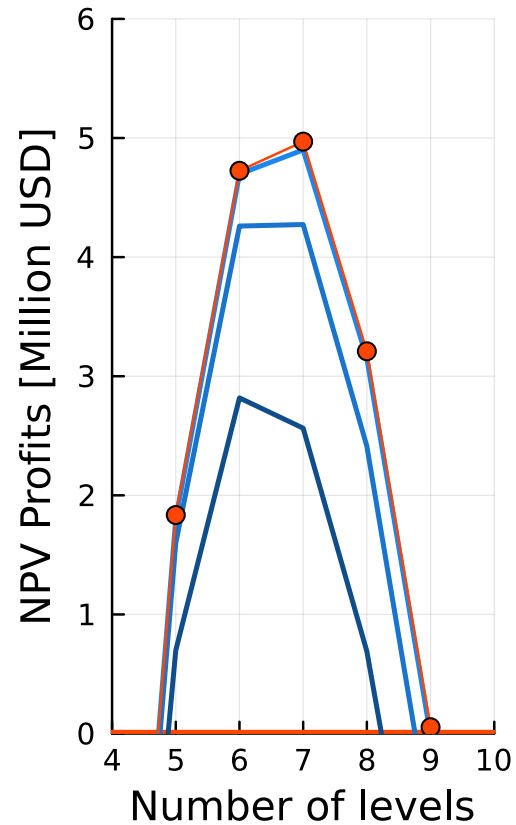
16         title = "Average NPV (N = $N_samples)")
17     hline!([0]; color = "orangered", linewidth = 3, label = nothing)
18     for s in 1:length(covs)
19         profits = zeros(length(n_levels),1)
20         for n in 1:N_samples
21             sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs[s]),
22                                     n_years = 20,
23                                     discount_rate = 0.12)
24             policies = [StaticPolicy(i) for i in n_levels]
25             profits .+= [simulate(sow,level_Δ_a, policy) for policy in policies]
26         end
27         plot!(n_levels,
28               profits/N_samples;
29               label = "COV = $(100*covs[s])%",
30               color = "dodgerblue$(s+1)",
31               linewidth = 2,
32               )
33     end
34     sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
35     policies = [StaticPolicy(i) for i in n_levels]
36     profits = [simulate(sow,level_Δ_a, policy) for policy in policies]
37     plot!(
38         n_levels,
39         profits;
40         label = "Static analysis",
41         marker = :circle,
42         color = "orangered",
43     )
44     pl_det = plot(pl;
45                  ylims = [0,6],
46                  y_ticks = 0:1:6,
47                  xlims = [4,10],
48                  title = "Detail",
49                  legend = false)
50     plot(pl, pl_det, layout = 2)
51 end

```

# Average NPV (N = 1000)



# Detail



## 3 Sequential decisions (adaptive policy/flexible/options)

The following function was written in the `sim` file of the `ParkingGarage` package to set the sequential policy:

```

1  let
2    function get_action(x::ParkingGarageState, policy::AdaptivePolicy, Δ)
3      if x.year == 1
4        return ParkingGarageAction(policy.n_levels_init)
5      else
6        if calculate_capacity(x) < x.demand
7          return ParkingGarageAction(Δ)
8        else
9          return ParkingGarageAction(0)
10       end
11     end
12   end
13 end

```



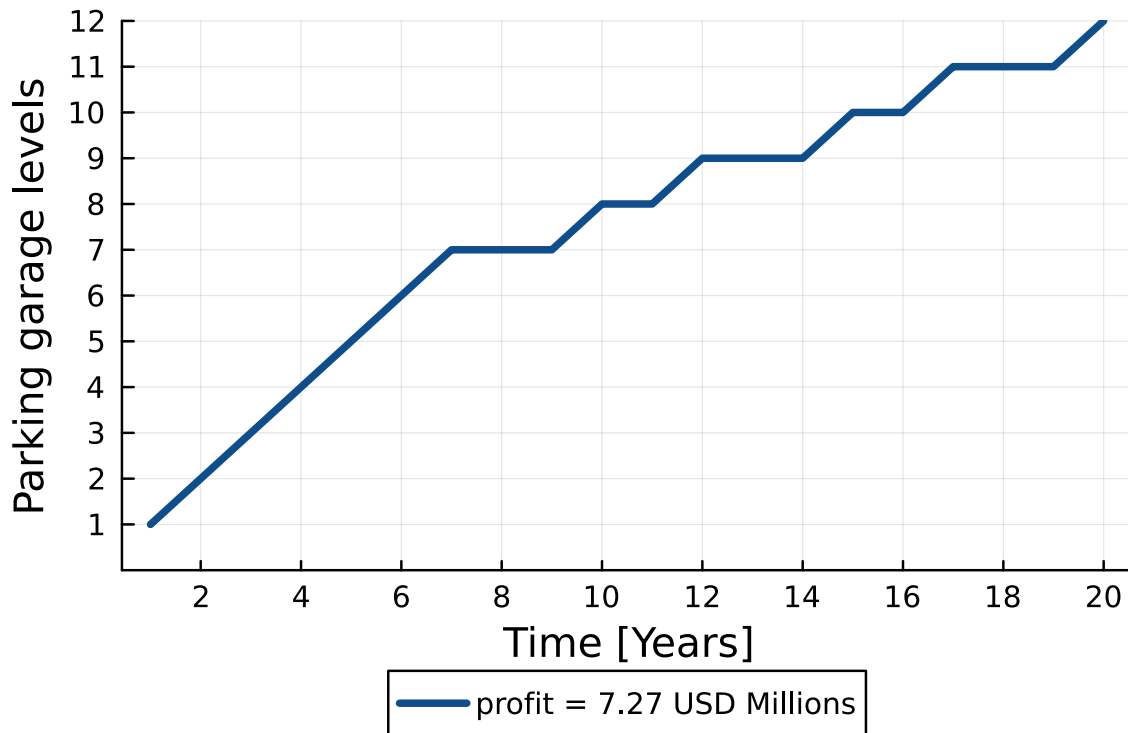
The policy consists in building a new level *every time* that the demand surpass the capacity of the parking building.

In order to understand how the sequential decisions are being selected, a new function `simulatelevels` is written in the `sim` file of the `ParkingGarage` package that returns the levels of the parking garage at every time step.

For example, the following are the level changes when the first level decision is one.

```
1 let
2   years = 20
3   level_Δ_a = 1
4   sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
5   n_levels = 1
6   policies = [AdaptivePolicy(i) for i in n_levels]
7   level_policy = [simulatelevels(sow, level_Δ_a, policy) for policy in policies]
8   profits = [simulate(sow, level_Δ_a, policy) for policy in policies]
9   pl = plot(1:years, level_policy[1];
10            title = "Initial level: $n_levels",
11            xlabel = "Time [Years]",
12            ylabel = "Parking garage levels",
13            label = "profit = $(trunc(profits[1], digits = 2)) USD Millions",
14            legend = :outerbottom,
15            size = (500, 400),
16            ylims = [0,12],
17            y_ticks = 1:1:12,
18            x_ticks = 2:2:20,
19            linewidth = 3,
20            color = "dodgerblue4",
21            )
22   pl
23 end
```

## Initial level: 1



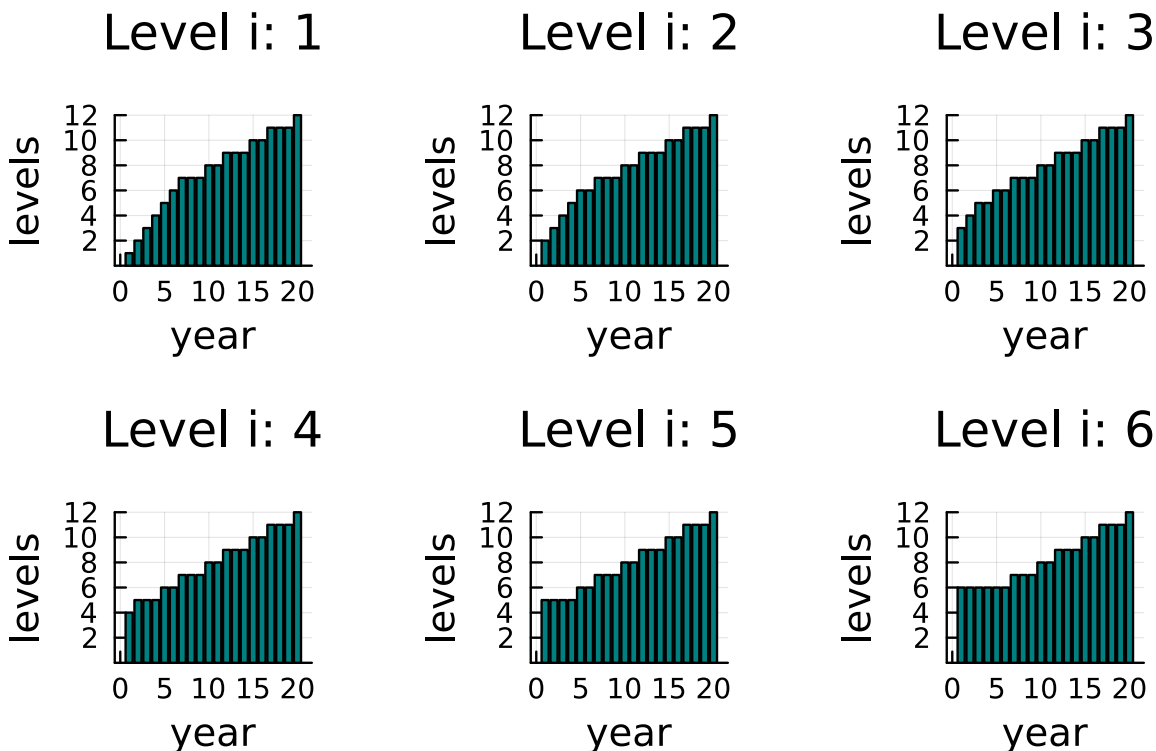
The following are the sequential decisions for different initial levels (Level i)

```
1 let
2   years = 20
3   level_Δ_a = 1
4   sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
5   n_levels = 1:6
6   policies = [AdaptivePolicy(i) for i in n_levels]
7   level_policy = [simulatelevels(sow, level_Δ_a, policy) for policy in policies]
8   profits = [simulate(sow, level_Δ_a, policy) for policy in policies]
9   l = @layout [grid(2,3)]
10  plot(level_policy;
11       layout = l,
12       seriestype = [:bar],
13       label = nothing,
14       color = "teal",
15       y_ticks = [2,4,6,8,10,12],
16       ylims = [0,12],
17       title = ["Level i: $i" for j in 1:1, i in 1:12],
18       xlabel = "year",
```

```

19     ylabel = "levels",)
20 end

```



The NPV behavior for this sequential decision proofs to yield to higher profits. It is debatable weather or not it is possible to build a new level every year. If that is the case, for a 80 growth rate demand (static), building 4 levels initially and increasing the number of levels up to 12 (according to the previous figure) yields to >10 USD millions NPV. For that case, starting with one level and keeping growing (~one level every 2 years) is profitable.

```

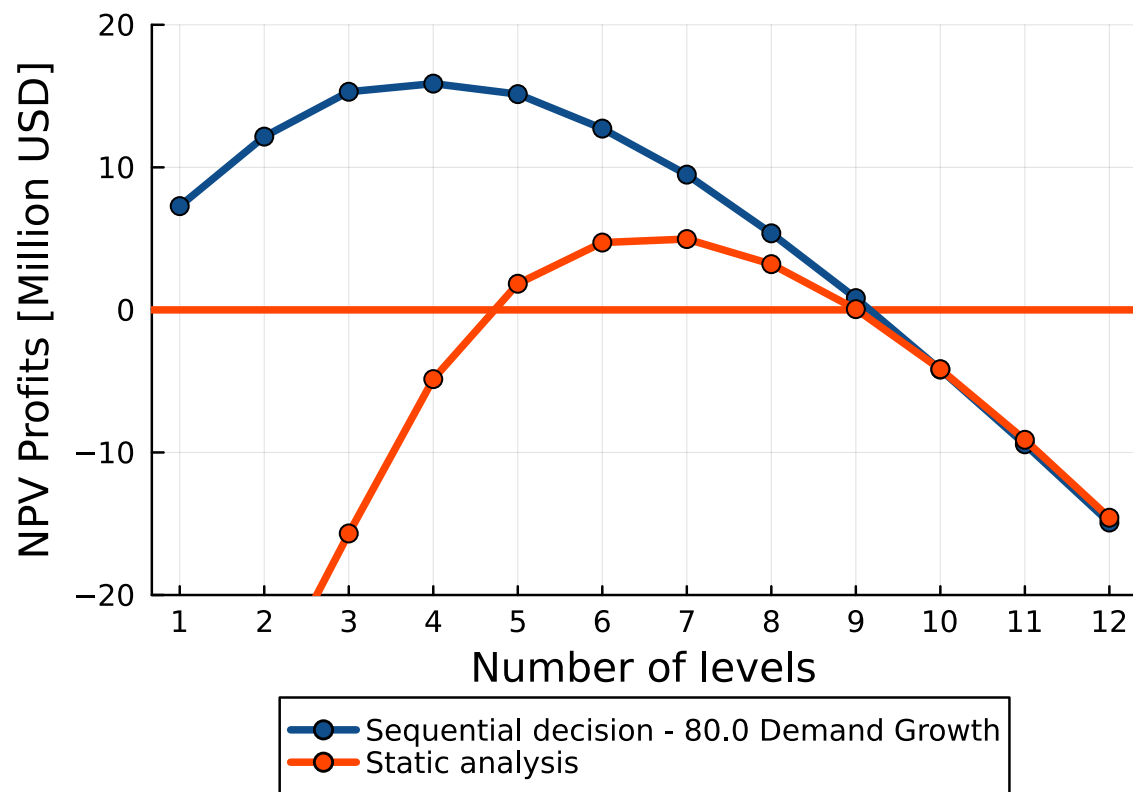
1  let
2      level_Δ_a = 1
3      years = 20
4      sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
5      n_levels = 1:12
6      policies = [AdaptivePolicy(i) for i in n_levels]
7      profits = [simulate(sow, level_Δ_a, policy) for policy in policies]
8      pl = plot()
9      hline!([0]; color = "orangered", linewidth = 3, label = nothing)
10     plot!(
11         n_levels,
12         profits;
13         ylabel = "NPV Profits [Million USD]",
14         ylims = [-20,20],
15         xlabel = "Number of levels",
16         legend = :outerbottom,

```

```

17     label = "Sequential decision - $(sow.demand_growth_rate) Demand Growth",
18     size = (500, 400),
19     marker = :circle,
20     xticks = n_levels,
21     color = "dodgerblue4",
22     linewidth = 3,
23 )
24 level_Δ_a = 0
25 sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
26 policies = [StaticPolicy(i) for i in n_levels]
27 profits = [simulate(sow,level_Δ_a, policy) for policy in policies]
28 plot!(
29     n_levels,
30     profits;
31     label = "Static analysis",
32     marker = :circle,
33     linewidth = 3,
34     color = "orangered",
35 )
36 )
37 end

```



When propagating uncertainty in the demand growth, the average NPV shows a similar trend. The following are 1000 simulations for a 50% COV (high uncertainty) and its average NPV. In the background are shown probable results from the simulations.

```

1  let
2      level_Δ_a = 1
3      covs = 0.5
4      N_samples = 1000
5      n_levels = 1:12
6
7      pl = plot(
8          ylabel = "NPV Profits [Million USD]",
9          y_ticks = -30:5:20,
10         ylims = [-30,20],
11         xlabel = "Number of levels",
12         legend = :outerbottom,
13         size = (500, 400),
14         xticks = n_levels,
15         linewidth = 3,
16         title = "Average NPV N = $N_samples")

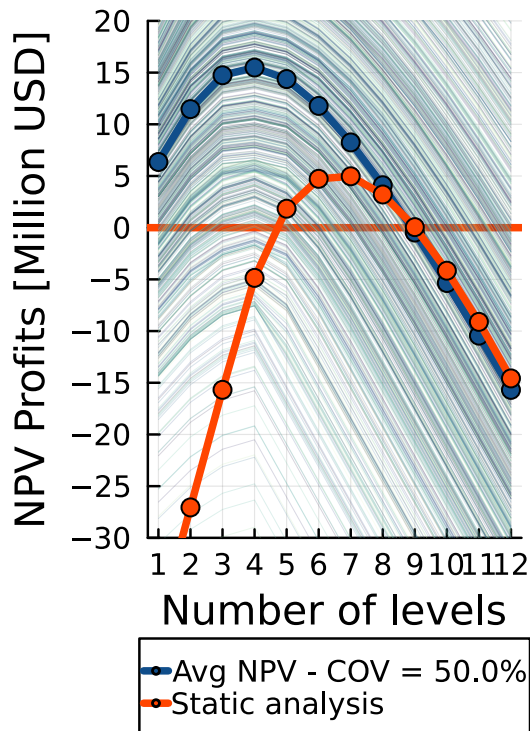
```

```

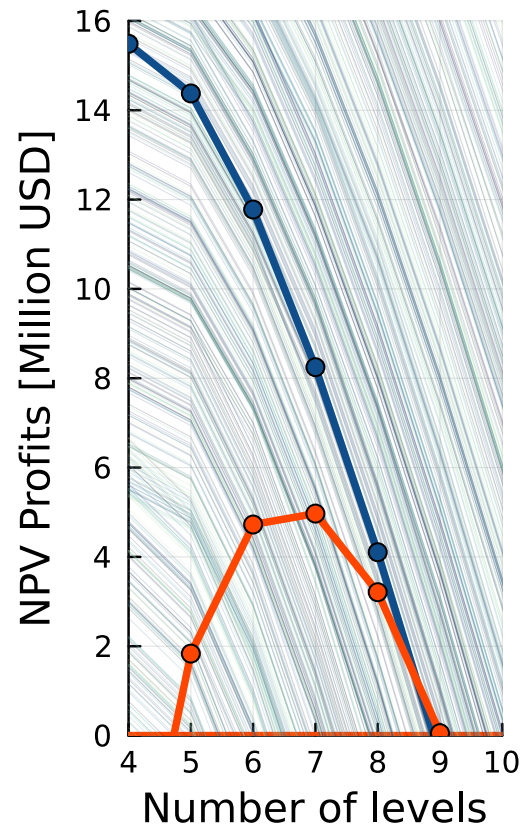
17     hline!([0]; color = "orangered", linewidth = 3, label = nothing)
18
19     profits = zeros(length(n_levels),1)
20     for n in 1:N_samples
21         sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs),
22                                 n_years = 20,
23                                 discount_rate = 0.12)
24         policies = [AdaptivePolicy(i) for i in n_levels]
25         profit_n = [simulate(sow, level_Δ_a, policy) for policy in policies]
26         plot!(n_levels,
27               profit_n;
28               label = nothing,
29               palette = :deep,
30               linewidth = 0.2,
31               alpha = 0.2
32             )
33         profits .+= profit_n
34     end
35     plot!(n_levels,
36           profits/N_samples;
37           label = "Avg NPV - COV = $(100*covs)%",
38           color = "dodgerblue4",
39           marker = :circle,
40           linewidth = 3,
41         )
42     level_Δ_a = 0
43     sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
44     policies = [StaticPolicy(i) for i in n_levels]
45     profits = [simulate(sow,level_Δ_a, policy) for policy in policies]
46     plot!(
47         n_levels,
48         profits;
49         label = "Static analysis",
50         marker = :circle,
51         linewidth = 3,
52         color = "orangered",
53     )
54
55     pl_det = plot(pl;
56                 y_ticks = 0:2:16,
57                 ylims = [0,16],
58                 xlims = [4,10],
59                 title = "Detail",
60                 legend =false)
61     plot(pl, pl_det, layout = 2)
62 end

```

## Average NPV N = 1000



## Detail



### 3.1 Number of levels per decision

In order to explore some realities, different sequential policies are used. First, by modifying the `sim` file, it is possible to use as an input the number of levels to build per action. It could be more realistic to assume that building new levels comes with several complexities and problems. It is preferable to performed few alterations to the structure in its timeframe. Here are the comparison of choosing to build 1, 2, and 3 levels when the demand surpasses the capacity.

```

1 let
2   years = 20
3   n_levels = 4
4   level_Δ_a = [1,2,3]
5   pl = plot(
6     title = "Initial level: $n_levels",
7     xlabel = "Time [Years]",
8     ylabel = "Parking garage levels",
9     legend = :outerbottom,
10    size = (700, 400),
11    ylims = [3,15],
12    y_ticks = 3:1:15,

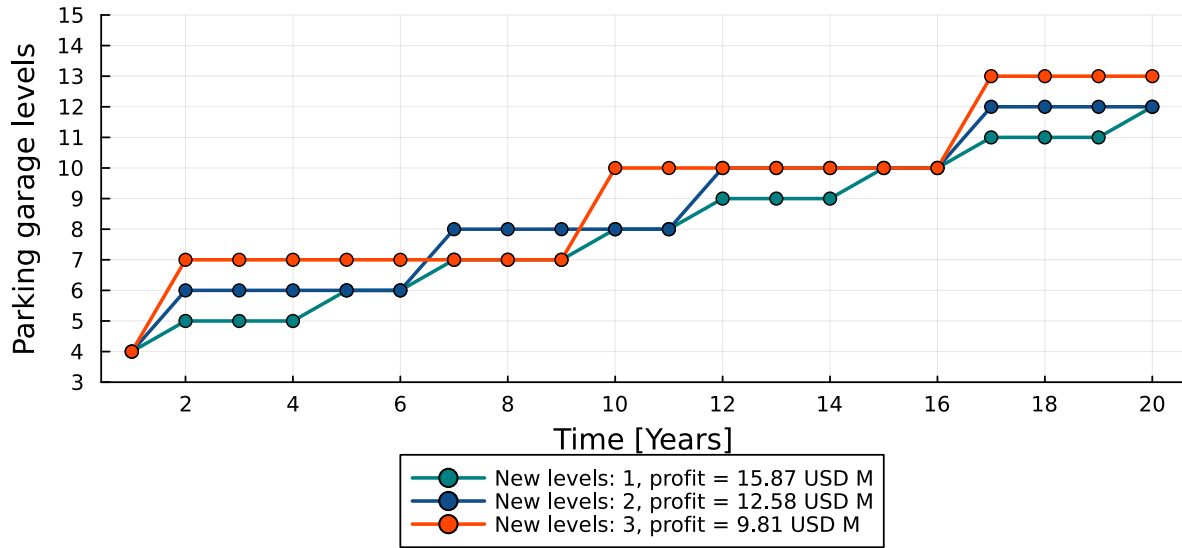
```

```

13         x_ticks = 2:2:20,
14         linewidth = 3,
15     )
16     color_plot = ["teal", "dodgerblue4", "orangered"]
17     for Δ in level_Δ_a
18         sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
19
20         policies = [AdaptivePolicy(i) for i in n_levels]
21         level_policy = [simulatelevels(sow, Δ, policy) for policy in policies]
22         profits = [simulate(sow, Δ, policy) for policy in policies]
23         pl = plot!(pl, 1:years, level_policy[1];
24             label = "New levels: $Δ, profit = $(trunc(profits[1], digits = 2)) USD M",
25             color = color_plot[Δ],
26             linewidth = 2,
27             marker = :circle,
28         )
29     end
30     pl
31 end

```

Initial level: 4



### 3.2 Random decision +1 level

Another decision policy would be considering that, for different reasons, the owner could or not decide to build a new level *even* if the demand surpasses the capacity (in the paper the decision was based on two consecutive years surpassing the capacity). A new `get_action` function is written in `sim` file to build one additional level or not (binary variable) randomly when the demand surpasses the capacity.



```

1 function get_actionbin(x::ParkingGarageState, policy::AdaptivePolicy)
2     if x.year == 1
3         return ParkingGarageAction(policy.n_levels_init)
4     else
5         if calculate_capacity(x) < x.demand
6             return ParkingGarageAction(trunc(Int, round(rand(1)[1])))
7         else
8             return ParkingGarageAction(0)
9         end
10    end
11 end

```

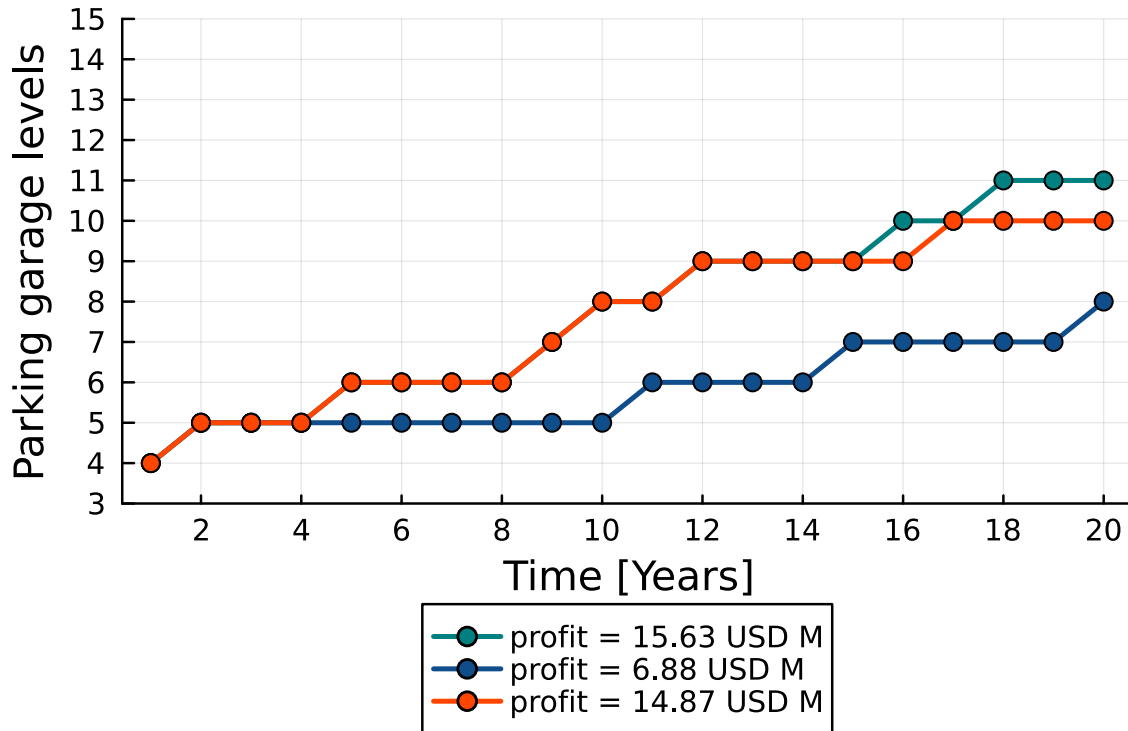
The following are some sequential decisions for the 80 rate demand growth ratio and four initial levels.

```

1 let
2     years = 20
3     n_levels = 4
4     color_plot = ["teal", "dodgerblue4", "orangered"]
5     pl = plot(;
6         title = "Initial level: $n_levels",
7         xlabel = "Time [Years]",
8         ylabel = "Parking garage levels",
9         legend = :outerbottom,
10        size = (500, 400),
11        ylims = [3,15],
12        y_ticks = 3:1:15,
13        x_ticks = 2:2:20,
14        linewidth = 3,
15    )
16    for i in 1:3
17        sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
18        policies = [AdaptivePolicy(i) for i in n_levels]
19        level_policy = [simulatelevelsbin(sow, policy) for policy in policies]
20        profits = [simulatebin(sow, policy) for policy in policies]
21        pl = plot!(pl, 1:years, level_policy[1];
22            label = "profit = $(trunc(profits[1], digits = 2)) USD M",
23            linewidth = 2,
24            marker = :circle,
25            color = color_plot[i]
26        )
27    end
28    pl
29 end

```

## Initial level: 4



This way, uncertainty in the decision is added to the model. The following are 100 simulations where there is large uncertainty even for static growth rate model.

```

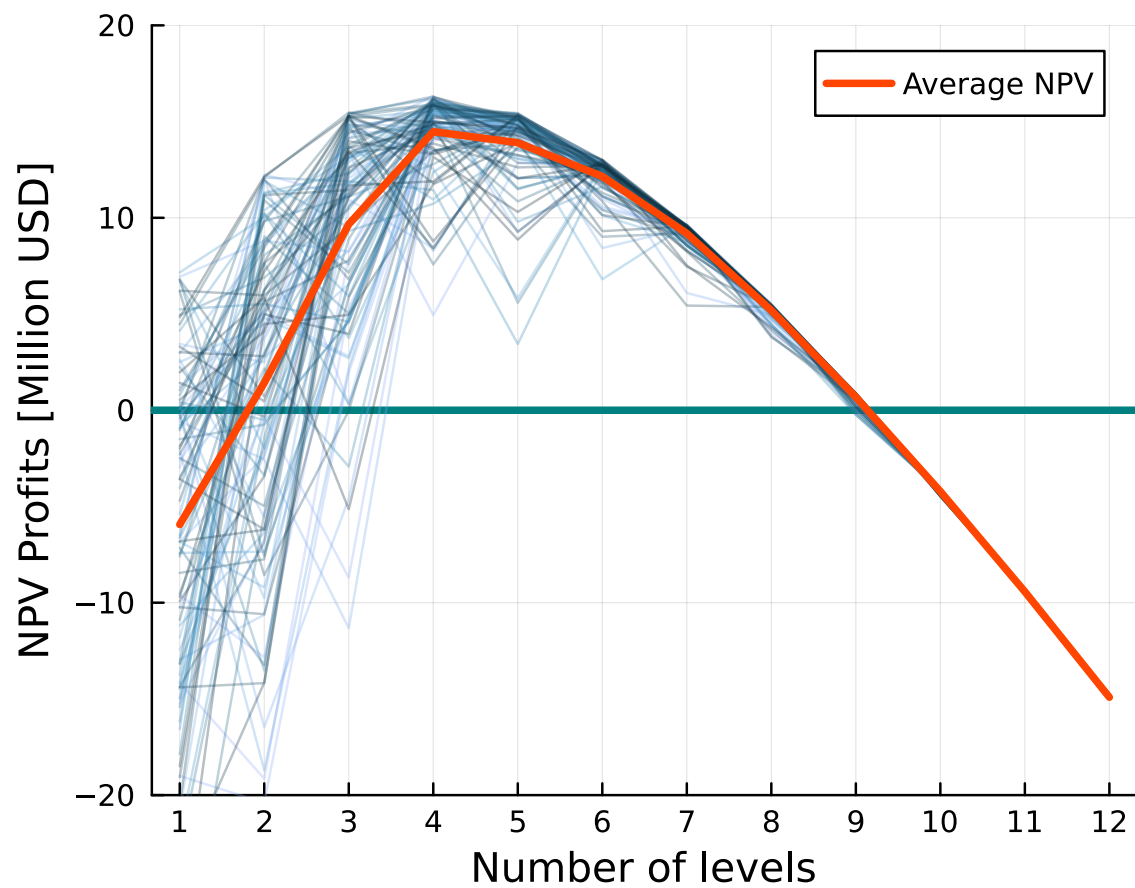
1 let
2     N_samples = 100
3     years = 20
4     n_levels = 1:12
5     profit_avg = zeros(12,1)
6     pl = plot(
7         ylabel = "NPV Profits [Million USD]",
8         ylims = [-20,20],
9         xlabel = "Number of levels",
10        size = (500, 400),
11        xticks = n_levels,
12        palette = :berlin,
13        linewidth = 1,)
14     hline!([0]; color = "teal", linewidth = 3, label = nothing)
15     for i in 1:N_samples
16         sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
17         policies = [AdaptivePolicy(i) for i in n_levels]

```

```

18     profits = [simulatebin(sow, policy) for policy in policies]
19     plot!(pl,
20           n_levels,
21           profits,
22           label = nothing,
23           alpha = 0.3
24         )
25     profit_avg = profit_avg + profits
26 end
27 plot!(pl,
28       n_levels,
29       profit_avg ./ N_samples;
30       linewidth = 3,
31       color = "orangered",
32       label = "Average NPV")
33 pl
34 end

```



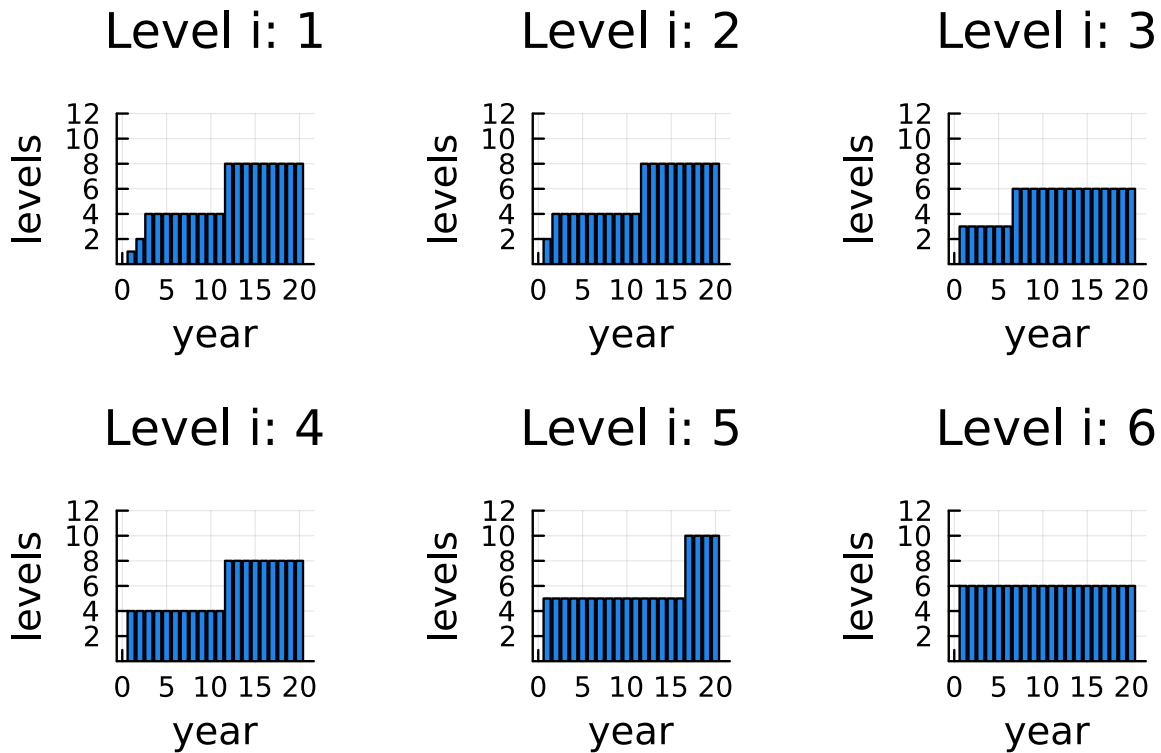
### 3.3 Minimizing new construction

Another policy could be minimizing new construction but also profiting from the new demand. A new `get_action` function is written to double the capacity of the building when the demand doubles the current capacity.

```
1 function get_action_double(x::ParkingGarageState, policy::AdaptivePolicy)
2     if x.year == 1
3         return ParkingGarageAction(policy.n_levels_init)
4     else
5         if calculate_capacity(x) < 0.50 * x.demand
6             return ParkingGarageAction(x.n_levels)
7         else
8             return ParkingGarageAction(0)
9         end
10    end
11 end
```

This way, the number of times the garage is in construction is very limited. The following are the sequential decisions under this policy.

```
1 let
2     years = 20
3     sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
4     n_levels = 1:6
5     policies = [AdaptivePolicy(i) for i in n_levels]
6     level_policy = [simulatelevels_double(sow, policy) for policy in policies]
7     profits = [simulate_double(sow, policy) for policy in policies]
8     l = @layout [grid(2,3)]
9     plot(level_policy;
10         layout = l,
11         seriestype = [:bar],
12         label = nothing,
13         color = "dodgerblue2",
14         y_ticks = 2:2:12,
15         ylims = [0,12],
16         title = ["Level i: $i" for j in 1:1, i in 1:12],
17         xlabel = "year",
18         ylabel = "levels",)
19 end
```



The following would be the NPV behavior for a static 80 growth rate demand. The curve shows a flat behavior given that the policy is very similar regardless of the initial level for the 4 to 7 level range.

```

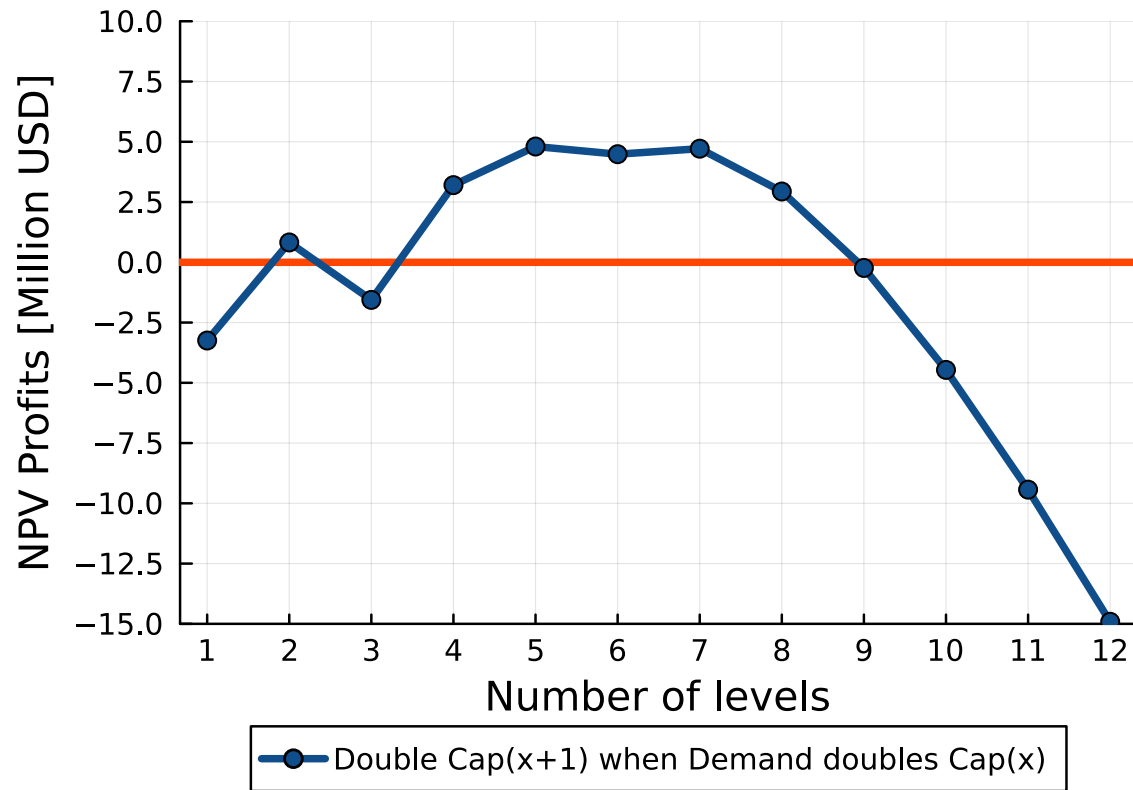
1 let
2   years = 20
3   sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=years, discount_rate=0.12)
4   n_levels = 1:12
5   policies = [AdaptivePolicy(i) for i in n_levels]
6   profits = [simulate_double(sow, policy) for policy in policies]
7   pl = plot()
8   hline!([0]; color = "orangered", linewidth = 3, label = nothing)
9   plot!(
10      n_levels,
11      profits;
12      ylabel = "NPV Profits [Million USD]",
13      ylims = [-15,10],
14      y_ticks = -15:2.5:10,
15      xlabel = "Number of levels",
16      legend = :outerbottom,
17      label = "Double Cap(x+1) when Demand doubles Cap(x) ",
18      size = (500, 400),
19      marker = :circle,
20      xticks = n_levels,
21      color = "dodgerblue4",

```

```

22     linewidth = 3,
23 )
24 end

```



When incorporating the variability from the demand growth model (high uncertainty 50% COV), the following is the average NPV from 1000 simulations.

```

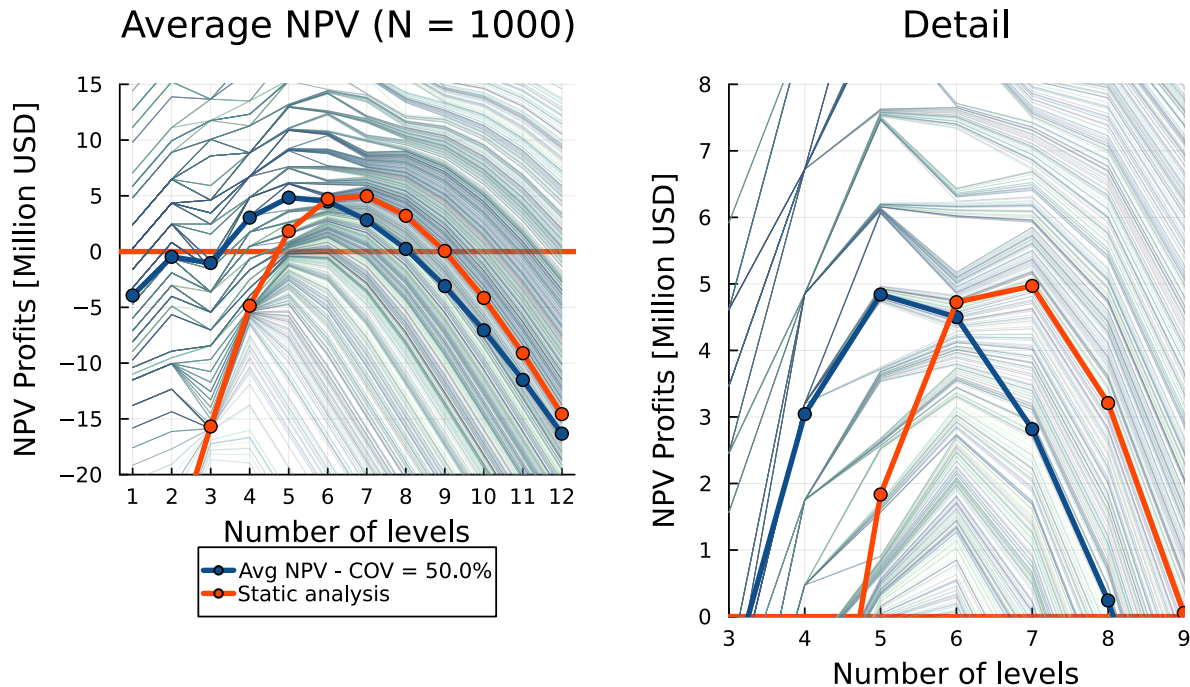
1  let
2      covs = 0.5
3      N_samples = 1000
4      n_levels = 1:12
5
6      pl = plot(;
7          ylabel = "NPV Profits [Million USD]",
8          y_ticks = -20:5:15,
9          ylims = [-20,15],
10         xlabel = "Number of levels",
11         legend = :outerbottom,
12         size = (700, 400),
13         xticks = n_levels,

```

```

14         linewidth = 3,
15         title = "Average NPV (N = $N_samples)")
16     hline!([0]; color = "orangered", linewidth = 3, label = nothing)
17
18     profits = zeros(length(n_levels),1)
19     for n in 1:N_samples
20         sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs),
21                                 n_years = 20,
22                                 discount_rate = 0.12)
23         policies = [AdaptivePolicy(i) for i in n_levels]
24         profit_n = [simulate_double(sow, policy) for policy in policies]
25         plot!(n_levels,
26               profit_n;
27               label = nothing,
28               palette = :deep,
29               linewidth = 0.2,
30               alpha = 0.2
31             )
32         profits .+= profit_n
33     end
34     plot!(n_levels,
35           profits/N_samples;
36           label = "Avg NPV - COV = $(100*covs)%",
37           color = "dodgerblue4",
38           marker = :circle,
39           linewidth = 3,
40           )
41     level_Δ_a = 0
42     sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
43     policies = [StaticPolicy(i) for i in n_levels]
44     profits = [simulate(sow,level_Δ_a, policy) for policy in policies]
45     plot!(
46         n_levels,
47         profits;
48         label = "Static analysis",
49         marker = :circle,
50         linewidth = 3,
51         color = "orangered",
52     )
53
54     pl_det = plot(pl;
55                  y_ticks = 0:1:8,
56                  ylims = [0,8],
57                  xlims = [3,9],
58                  title = "Detail",
59                  legend = false)
60     plot(pl, pl_det, layout = 2)

```



## 4 Comparisson

When incorporating the variability from the demand growth model (high uncertainty 50% COV), the following is the average NPV from 1000 simulations. Finally, the following is the comparison for the deterministic model (with and without uncertainty), and the studied sequential decision policies. The figure is generated from 1000 samples and a **COV of 30%**.

```

1 let
2   covs = 0.30
3   N_samples = 1000
4   n_levels = 1:12
5
6   pl = plot(;
7     ylabel = "NPV Profits [Million USD]",
8     y_ticks = -30:5:20,
9     ylims = [-30,20],
10    xlabel = "Number of levels",
11    legend = :outerbottom,
12    size = (700, 400),
13    xticks = n_levels,
14    linewidth = 3,
15    title = "Average NPV N = $N_samples")
16   hline!([0]; color = "orangered", linewidth = 3, label = nothing)
17   # Uncertainty
18   level_Δ_a = 0

```



```

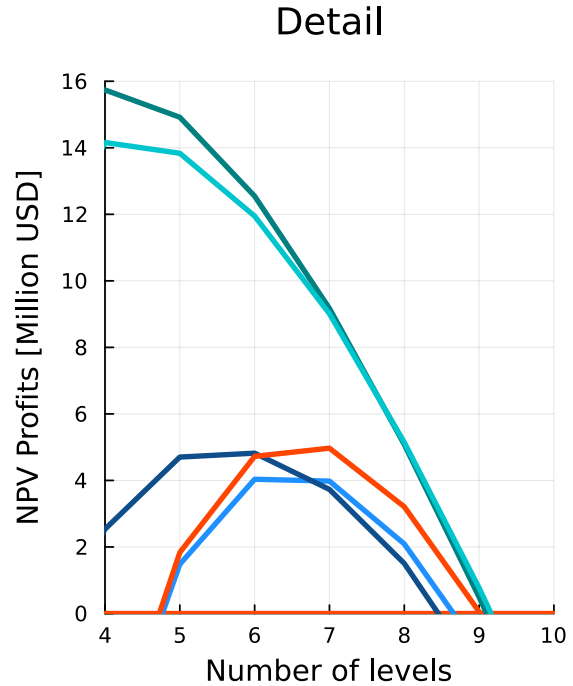
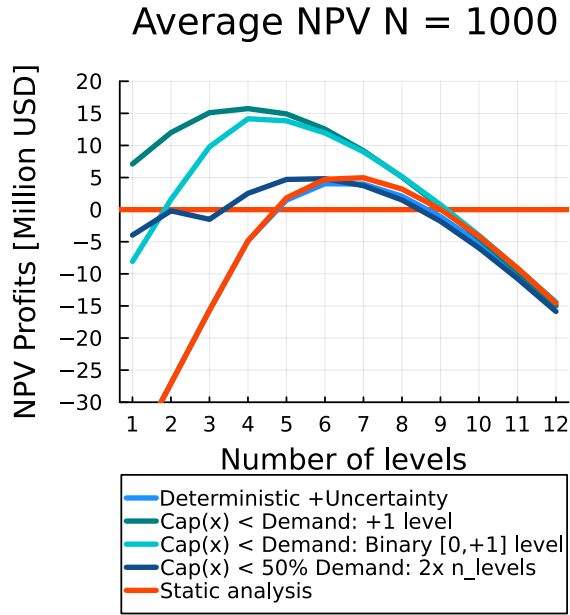
19 profits = zeros(length(n_levels),1)
20 for n in 1:N_samples
21     sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs),
22                             n_years = 20,
23                             discount_rate = 0.12)
24     policies = [StaticPolicy(i) for i in n_levels]
25     profits .+= [simulate(sow,level_Δ_a, policy) for policy in policies]
26 end
27 plot!(n_levels,
28       profits/N_samples;
29       label = "Deterministic +Uncertainty",
30       linewidth = 3,
31       color = "dodgerblue1",
32       )
33 # Building a new level when capacity is surpass by demand
34 level_Δ_a = 1
35 profits = zeros(length(n_levels),1)
36 for n in 1:N_samples
37     sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs),
38                             n_years = 20,
39                             discount_rate = 0.12)
40     policies = [AdaptivePolicy(i) for i in n_levels]
41     profit_n = [simulate(sow, level_Δ_a, policy) for policy in policies]
42     profits .+= profit_n
43 end
44 plot!(n_levels,
45       profits/N_samples;
46       label = "Cap(x) < Demand: +1 level",
47       color = "teal",
48       linewidth = 3,
49       )
50 # Binary decision
51 profits = zeros(length(n_levels),1)
52 for i in 1:N_samples
53     sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs), n_years = 20
54                             policies = [AdaptivePolicy(i) for i in n_levels]
55     profit_n = [simulatebin(sow, policy) for policy in policies]
56     profits .+= profit_n
57 end
58 plot!(n_levels,
59       profits/N_samples;
60       linewidth = 3,
61       color = "turquoise3",
62       label = "Cap(x) < Demand: Binary [0,+1] level")
63 # Building when the demand doubles the capacity. Double the capacity
64 profits = zeros(length(n_levels),1)
65 for n in 1:N_samples

```

```

66         sow = ParkingGarageSOW(; demand_growth_rate = draw_growth_rate(covs),
67                                n_years = 20,
68                                discount_rate = 0.12)
69         policies = [AdaptivePolicy(i) for i in n_levels]
70         profit_n = [simulate_double(sow, policy) for policy in policies]
71         profits .+= profit_n
72     end
73     plot!(n_levels,
74           profits/N_samples;
75           label = "Cap(x) < 50% Demand: 2x n_levels",
76           color = "dodgerblue4",
77           linewidth = 3,
78           )
79     level_Δ_a = 0
80     sow = ParkingGarageSOW(; demand_growth_rate=80.0, n_years=20, discount_rate=0.12)
81     policies = [StaticPolicy(i) for i in n_levels]
82     profits = [simulate(sow,level_Δ_a, policy) for policy in policies]
83     plot!(
84         n_levels,
85         profits;
86         label = "Static analysis",
87         linewidth = 3,
88         color = "orangered",
89
90     )
91     pl_det = plot(pl;
92                  y_ticks = 0:2:16,
93                  ylims = [0,16],
94                  xlims = [4,10],
95                  title = "Detail",
96                  legend = false)
97     plot(pl, pl_det, layout = 2)
98 end

```



The following are some conclusions:

1. regardless of the adaptive policy, the parking garage can have higher NPV when “options” are incorporated. In particular, they all shown large profit even when starting the building with small number of levels (low initial investment).
2. If consecutive construction in the building is feasible, the profits can grow importantly when the capacity is closely adapting to the demand.
3. When multiple constructions are not desired, smaller NPV are possible but much higher for small number of initial levels. Considering options clearly represent a better investment in comparison to a single decision upfront.