Statistická Práce
Robotický Vysavač

Vojtěch Pröschl

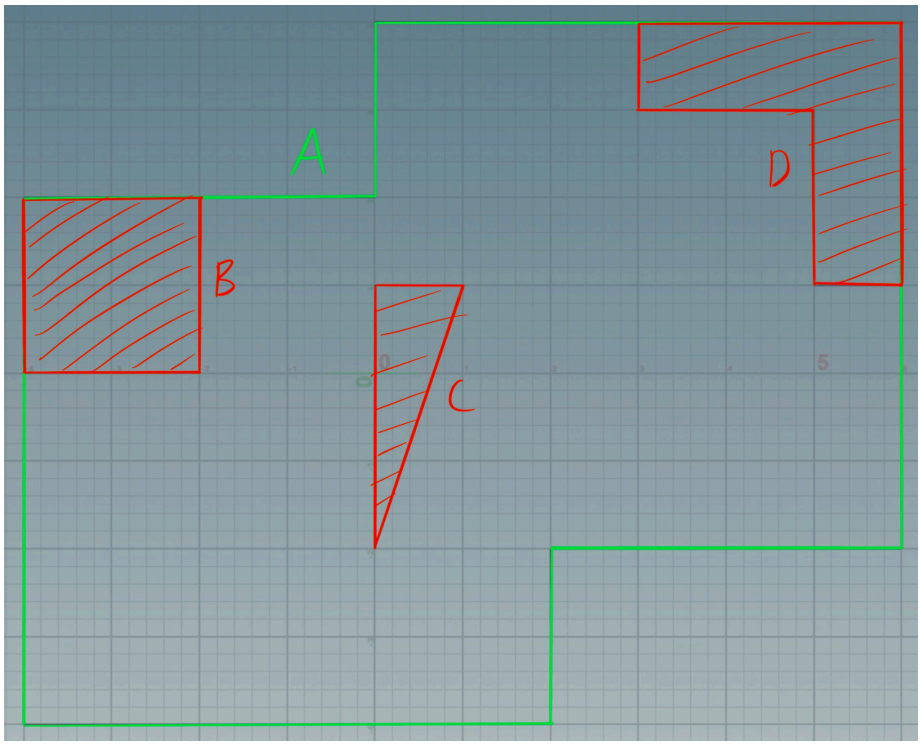
25. srpna 2024

1 Prostředí experimentu

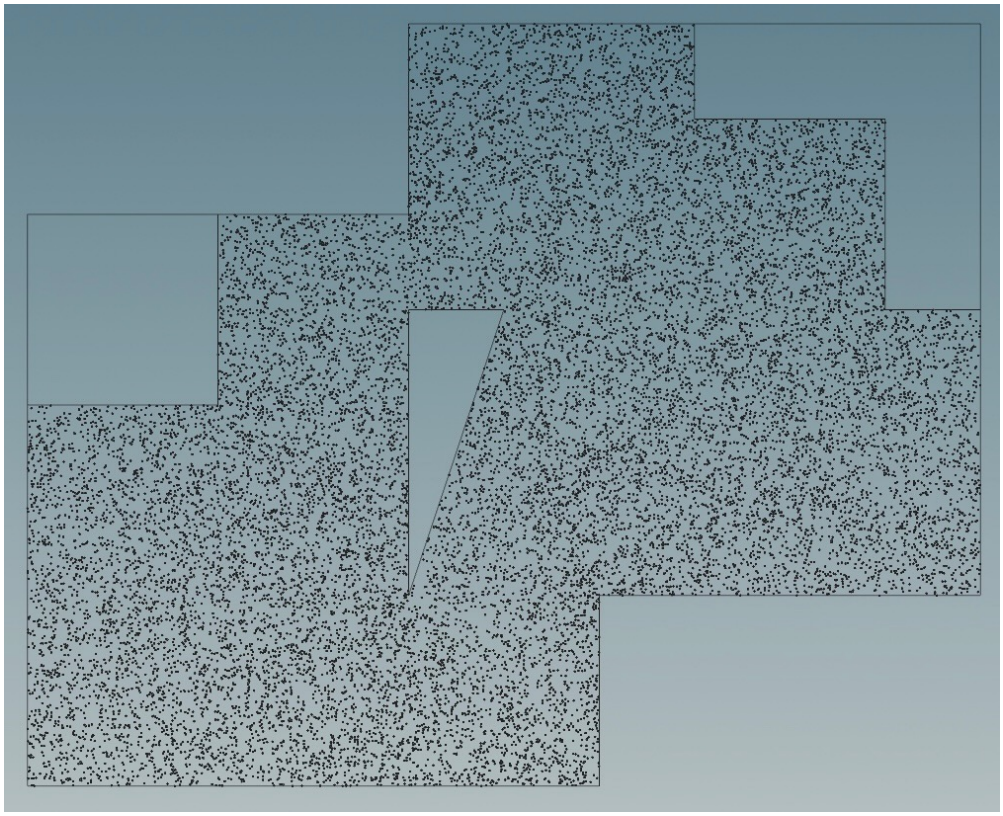
Stručné shrnutí: Máme robota, který se v pokoji pohybuje tak, že jde rovně, dokud nenarazí do zdi. Pokud narazí do zdi, otočí se o náhodný úhel a pokračuje rovně. Ve zbytku této sekce je detailněji popsán model světa, ve kterém se experiment odehrává, a konkrétní „mechanismus“ fungování robota.

1.1 Pokoj

Geometrie: Podlaha pokoje je popsána polygonem A, překážky jsou popsány polygony B, C a D. Body, které leží uvnitř polygonu A a zároveň neleží uvnitř polygonů B, C nebo D reprezentují plochu, kterou má robot uklidit.



Úklidové body: Na ploše podlahy, která není zablokována žádnou z překážek, jsem uniformně náhodně vygeneroval 20 tisíc bodů, které budou reprezentovat nepořádek v pokoji.



1.2 Robot

Robot je z geometrického pohledu kruh v prostoru, který je schopen uklidit body, které do něj v daný moment náleží. Robot má přiřazenou určitou orientaci. Je schopen se pohybovat rovně ve směru této orientace a otáčet se, aby změnil svou orientaci.

Radius:

$$r := 0.15$$

Startovni pozice:

$$S_P = (-2, -2)$$

Startovni orientace:

$$S_o = 0$$

Poznámka: toto reprezentuje směr robota $(\cos(0), \sin(0))$

Rychlost pohybu:

$$v := 0.3 \ \left[\frac{m}{s}\right]$$

Rychlost otáčení:

$$\omega := \frac{\pi}{5} \left[\frac{\text{rad}}{s}\right]$$

1.3 Dostupné instrukce:

Move forward: Robot půjde rovně, dokud nenarazí do zdi. Doba trvání této instrukce se počítá jako

$$t_m = \frac{d_e(P_0, P_1)}{v}$$

kde P_0 je pozice, na které robot začal, a P_1 je pozice, na které robot narazil do zdi.

Rotate by α: Robot se otočí o úhel α kolem své osy. Doba trváni této instrukce je

$$t_r = \frac{\alpha}{\omega}$$

1.4 Algoritmus robota

Posloupnost instrukcí robota je funkce $I : \mathbb{N} \to \text{Instructions}$

$$I(k) = \begin{cases} \text{Move forward} & \iff k \,\%\, 2 = 0 \\ \text{Rotate by } \alpha(k) & \iff k \,\%\, 2 = 1 \end{cases}$$

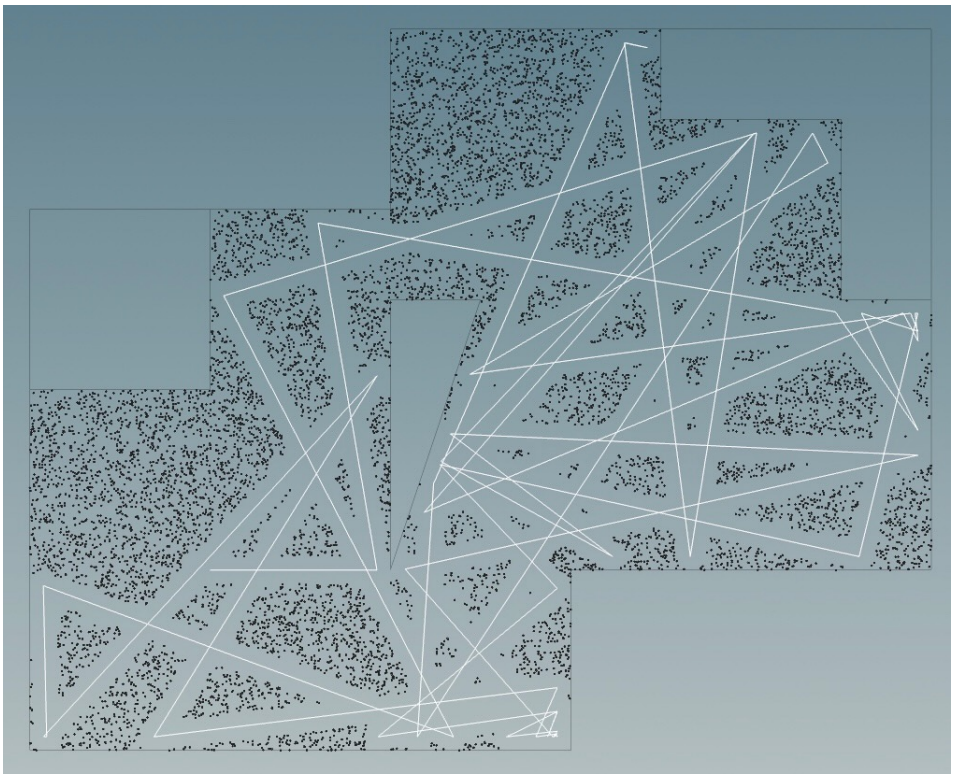
Kde $\alpha : \mathbb{N} \to [-\pi, \pi]$ je konkrétní pozorování posloupnosti uniformně rozdělených, nezávislých náhodných veličin na výše uvedeném intervalu.

1.5 Proces uklidu

Pokud je v nějakém okamžiku vzdálenost středu robota od nějakého úklidového bodu menší než jeho poloměr, tento bod se označí za uklizený. Procento uklizené plochy můžeme aproximovat pomocí úklidových bodů jako:

$$\frac{\#\text{Uklizených}}{\#\text{Úklidových bodu}} \cdot 100$$

Aby se experiment dal realizovat v rozumném čase, přepočítávám počet uklizených bodů vždy po dokončení instrukce.



2 Experiment

Cílem experimentu je nasbírat data o chování robota při úklidu modelového prostředí.

2.1 Jedna iterace

Iteraci si rozdělíme na jednotlivé instrukce a budeme sbírat data o stavu úklidu po vykonání k-té instrukce. Celkem v jedné iteraci provedeme 5000 instrukcí. Data, která budeme sbírat, jsou následující:

- T_I - Jak dlouho robotovi trvala k-tá instrukce?
- T_C - Jak dlouho robotovi celkem trvalo vykonat instrukce 0 až k?
- C_I - Kolik dosud neuklizených bodů robot při vykonávání k-té instrukce uklidil?
- C_C - Kolik bodů je po vykonání k-té instrukce celkem uklizeno?
- P - Kolik procent bodů je po vykonání k-té instrukce uklizeno?

2.2 Opakované iterace

Provedeme $2 \cdot 500$ iterací našeho experimentu. V každé iteraci pro generování náhodných čísel pro rotace použijeme jiný seed. Výsledné data jsem rozdělil do dvou skupin, prvních 500 běhů jsem uložil do složky `/Data/DataSet1` a posledních 500 běhů jsem uložil do složky `/Data/DataSet2`.

V následující sekci provedu explorační analýzu dat z první množiny, pomocí které zformuluju hypotézu. Tuto hypotézu pak budu testovat na druhé množině dat.

2.3 Jak jsem prováděl simulace?

Simulace jsem prováděl v softwaru SideFX Houdini, v repozitáři je uložený soubor s experimentem `/Houdini/main.hiplc`. Houdini je program na procedurální generování geometrie a simulace a má solidní sadu nástrojů na výpočetní geometrii, pomocí čehož jsem vytvořil simulátor. Nekomerční verze softwaru se dá zdarma stáhnout na stránce SideFX.

3 Explorační analýza dat - Experiment 1

Výpočty z této částí jsou popsány ve zdrojáku `ExperimentExploration.py`. Cílem je prozkoumat chování robota během jednoho běhu. Běh, který zde analyzuji je `/Data/DataSet1/Experiment1.csv`.

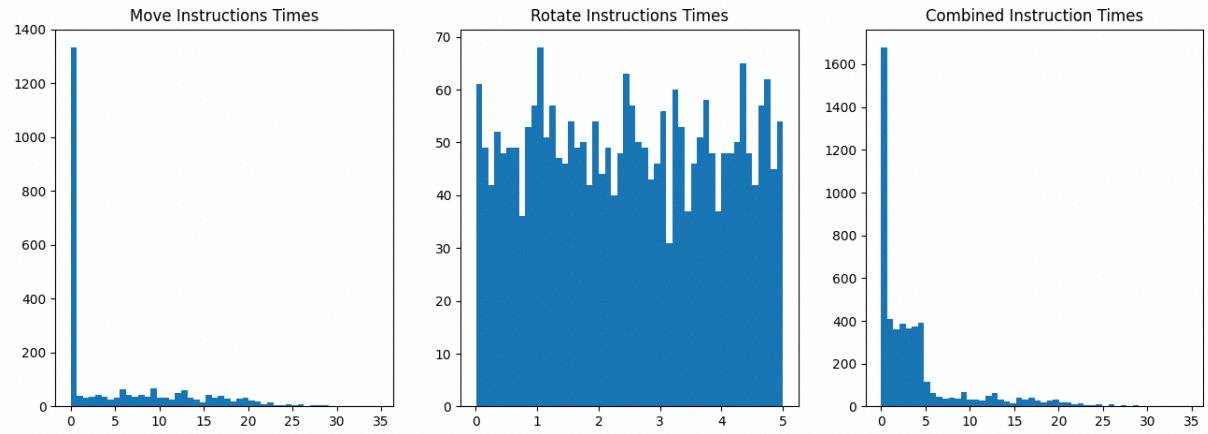
3.1 O jednotlivých instrukcích

Pojďme se nejprve podívat na to, jak dlouho trvaly jednotlivé instrukce.

- Nejkratší instrukce $\approx 1.98 \cdot 10^{-7}$ $[s]$
- Nejdelší instrukce ≈ 34.5 $[s]$
- Průměrná instrukce ≈ 3.9 $[s]$
- Průměrná `Move` instrukce ≈ 5.3 $[s]$
- Průměrná `Rotate` instrukce ≈ 2.49 $[s]$

Protože se robot otáčí vždy o úhel $\alpha \in [-\pi, \pi]$, který si vybírá uniformě náhodně, můžeme odvodit, že průměrné otočení trvá $(\pi/2)/(\pi/5) = 5/2$ sekund - tohle poměrně dobře odpovídá pozorovaným datům. Z toho rovněž pomocí rychlosti pohybu v můžeme odvodit průměrnou vzdálenost, kterou robot při pohybové instrukcí urazí jako $5.3 \cdot v = 1.674$

3.2 Rozdělení trvání instrukcí



Z rozdělení instrukcí můžeme vypozorovat pár zajimavých vlastností robota

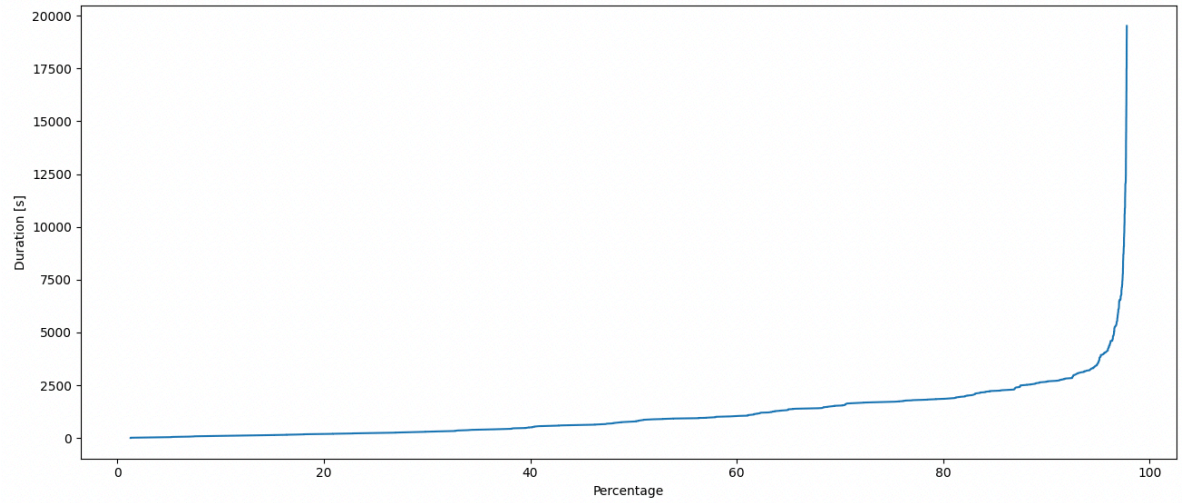
Opakované nárazy do zdi: V momentě, kdy robot narazí do zdi, se často stává, že po otočení robot neujde moc velkou vzdálenost a opět narazí do té samé zdi (spike v prvním histogramu). Tohle nám ale z hlediska doby úklidu moc nevadí, protože takové instrukce jsou velmi rychlé.

Otáčení zabírá hodně času: Problém ale nastává v tom, že po nárazu do zdi, se robot opět otočí o náhodný úhel, což průměrně trvá 2.5s. Ke každé zanedbatelné instrukci pohybu, při které robot narazí do zdi hned po otočení, si tedy musíme přičíst čas, jaký trvalo samotné otočení. Algoritmus by se tedy dal dost zrychlit, kdybychom zamezili opakovaným nárazům do zdi (nebo chytřeji volili rotaci po opakovaném nárazu)

Rozdělení času: Rozdělení doby trvání instrukcí je dost složité a pravděpodobně by se dalo modelovat kombinací několika známých rozdělení. V tomhle projektu se tímhle směrem nevydáme, a raději budeme měřit dobu celkového úklidu.

3.3 O celkovém průběhu

Robotovi v experimentu 1 trvalo vykonat 5000 instrukci priblizne 19518 sekund a uklidil pri tom priblizne 97.79% plochy pokoje.



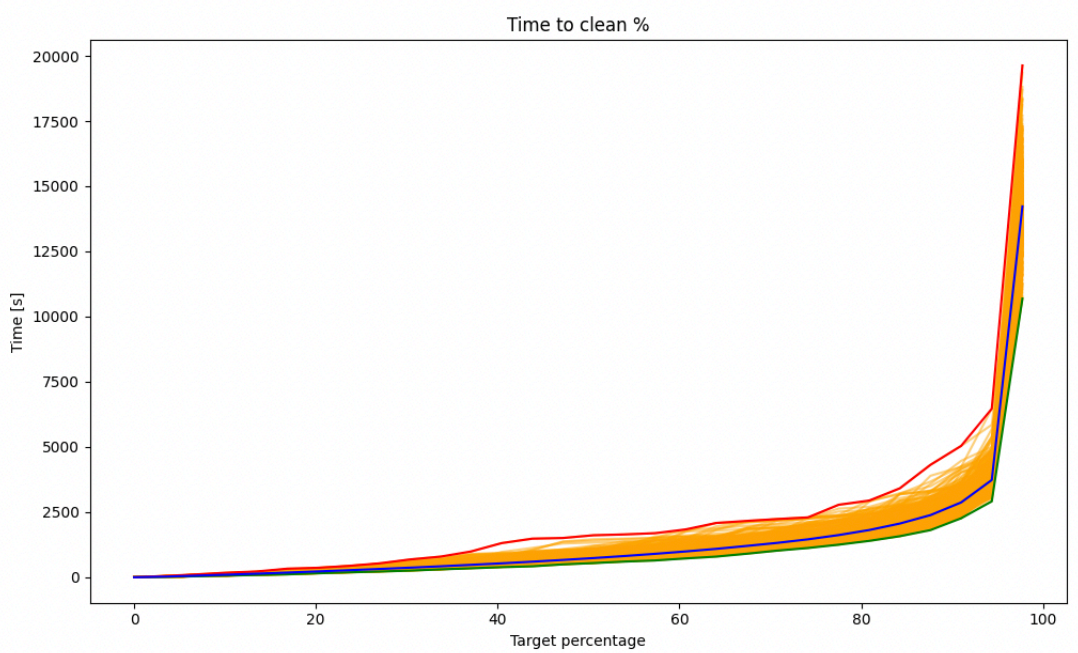
Z tohoto grafu můžeme vyčíst, že zřejmě časová náročnost pro uklizení dalšího procenta po překročení hranice přibližně 90ti procent opravdu strmě roste. Z praktického pohledu by nám mohlo stačit, aby robot uklidil právě 90% plochy pokoje. V další části tedy prozkoumáme, jak vypadá rozdělení časů, kdy robot překročil hranici 90%.

4 Explorační analýza dat - Dataset 1

Výpočty z této částí jsou popsány ve zdrojáku `DataSetExploration.py`

4.1 O celkovém průběhu

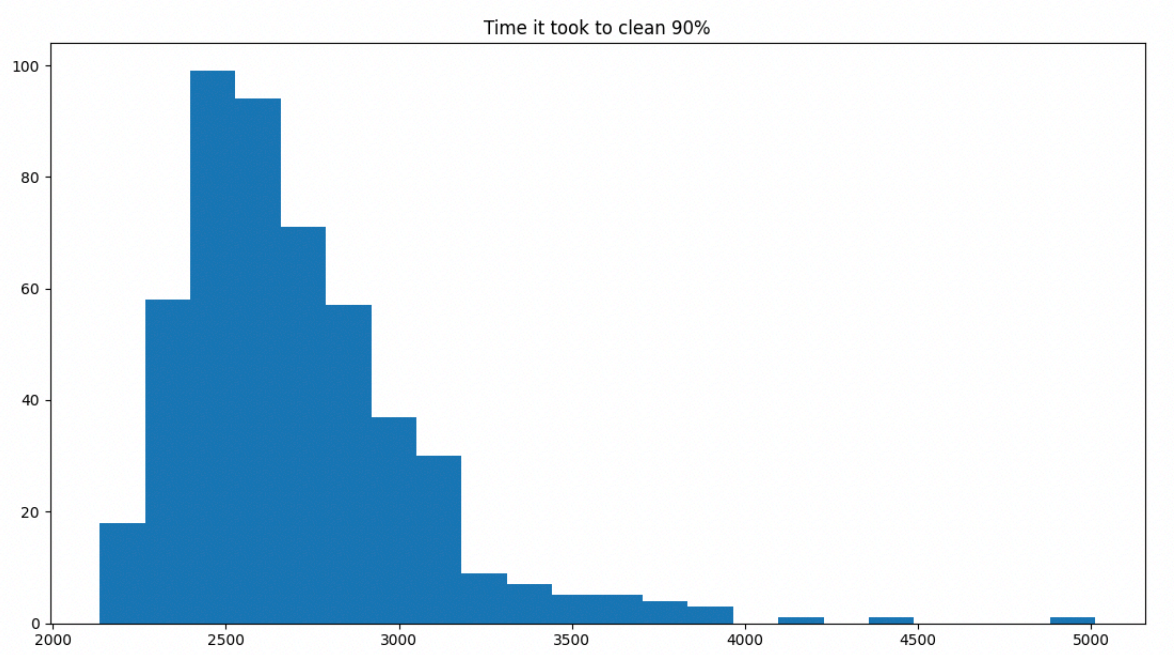
Podobně jako v části s jedním experimentem, podívejme se na to, jakým způsobem vypadá chování mnoha experimentů - v tomto případě celého datasetu 1. Pro sjednocení dat jsem na základě výsledků experimentů provedl lineární interpolaci dat a každý experiment jsem nasamploval na interval $[5, 97]$ procent. Tohle do dat přivádí určitou nepřesnost, vzhledem k tomu, že ale robot typicky v jedné instrukci neuklidí moc velkou plochu, je to přijatelné.



Na teto ilustraci modrá červená křivka je vrchní hranice doby trvání pro určité procento uklizené plochy, modrá je střední hodnota a zelená křivka je minimální doba trvání. Z těchto dat je patrné, že rozptyl doby pro úklid určité plochy roste podobně jako čas. Nicméně v 90% je stále jak střední hodnota doby úklidu, tak rozptyl této doby poměrně přijatelný, pojďme se na to teď podívat jako na rozdělení.

4.2 Jak dlouho trvá uklidit 90% plochy?

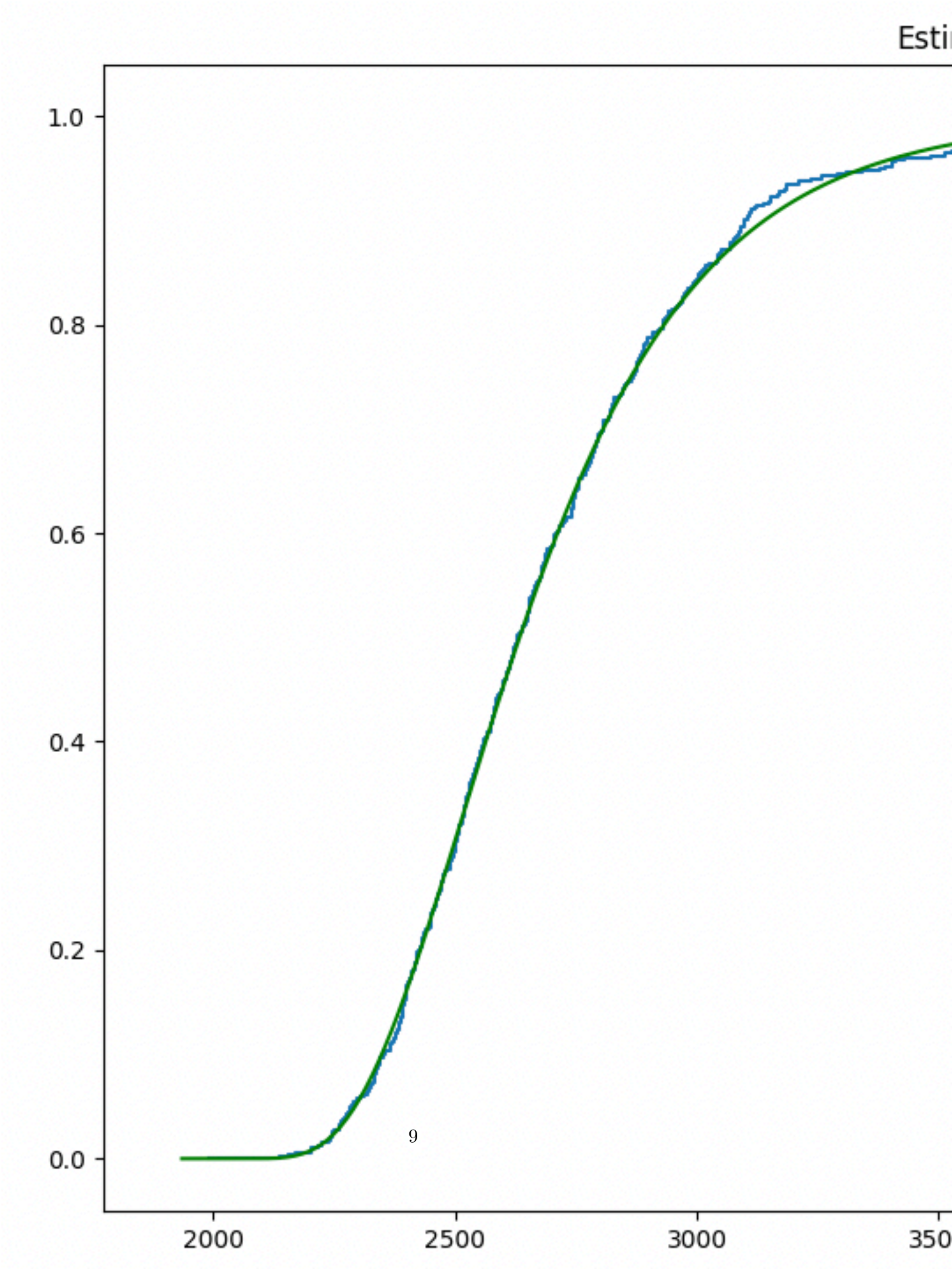
Pojďme se nejprve podívat na histogram o tomto údaji pro všech 500 úklidů z datasetu1.



Z histogramu se může zdát, že jsou doby normálně rozdělené, problém je ale v tom, že je pro vysavač nemožné uklidit dané procento rychleji, než je nějaký minimální čas úrčený jeho parametry a geometrii pokoje. Z moje pohledu tohle poměrně dobře odpovídá log-normálnímu rozdělení.

4.3 ECDF a fitting parametrů

Provedu fitting parametrů log-normálního rozdělení na základě dat z datasetu 1.


Esti
1.0
0.8
0.6
0.4
0.2
0.0
2000
2500
3000
350

5 Testování hypotézy

Výpočty z této částí jsou popsány ve zdrojáku `HypothesisTest.py`