REA3064: Smittemodellering

Sørensen M.

Amalie Skram VGS.

Februar 2, 2023

Innholdsfortegnelse

1 Enkel modell $I_{t+1} = I_t + \alpha I_t$

Den første modellen tar kun for seg muligheten til å bli smittet. Uttrykket $I_{t+1} = I_t + \alpha I_t$ representerer utviklingen til modellen, der α er koeffisienten som uttryker antall personer et gjennomsnittlig, smittet individ møter på en uke, eller rettere sagt, representerer $\frac{7}{\alpha}$ den gjennomsnittlig antall dager det tar før en smittet person smitter en frisk person. Vi har da at vekstfarten til smittetallet er proporsjonalt med både den gjennomsnittlige tiden det tar for en smittet person å smitte videre, og, fra uttrykket, at den er proporsjonal med antall personer som allerede er smittet.

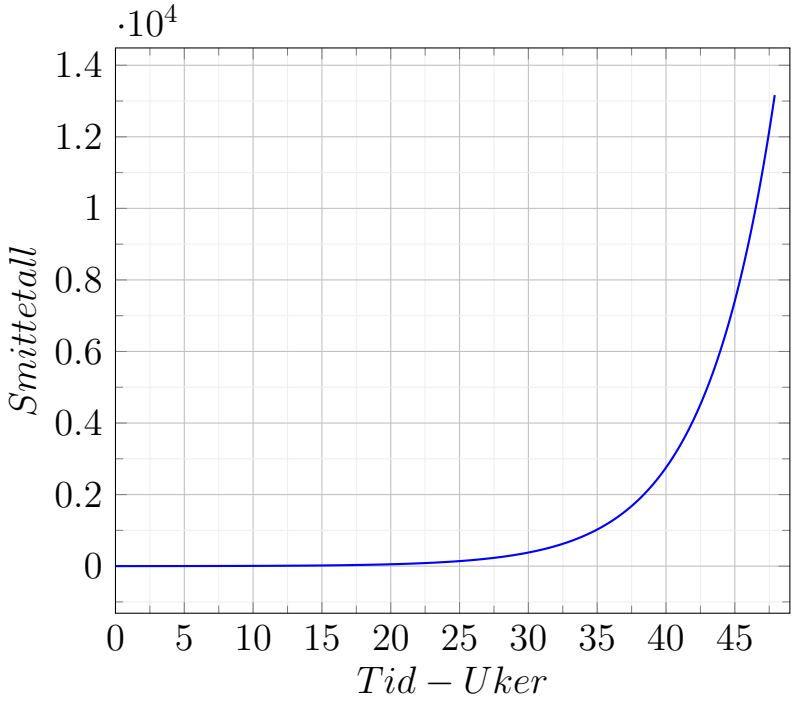
Modellen har kun et begrenset gyldighetsrom, ettersom at den ikke tar for seg at et smittet individ kan bli frisk. Dette gjør at antallet smittede hoper seg opp, og vil derfor ikke represente smittetallet i lengden. I tillegg antar modellen at vi har et uendelig antall individer, ettersom at vekstfarten kun er proporsjonal med antallet smittet og hvor ofte en person smitter videre.

Under er et program som simulerer modellen med en simuleringstid på 48 uker og et initial smittetall på 1. Simuleringen er delt inn i mindre, diskrete, tidssteg for å gjøre simuleringen mer kontinuerlig.

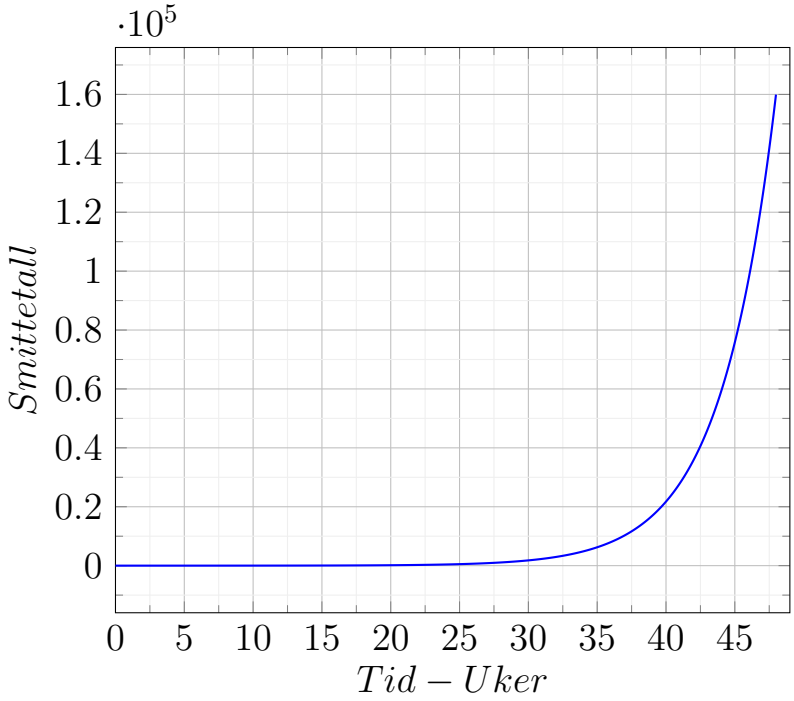
```
# Variabler for simulering
simulerings_tid = 48 # Antall uker modellen simuleres
dt = 0.01 # Tidssteget i simuleringen - brøkdel av en uke

def smitte_modell(smitte_koeffisient):
    tid = np.arange(0, simulerings_tid, dt) # Tids array
    I = np.zeros(len(tid)) # Smitte array
    I[0] = 1 # Initialiserer antall smittede ved t = 0 til 1

    for t in range(1, len(tid)):
        # Regner ut vekstfarten og legger det til forrige smittetall
        smitte = smitte_koeffisient * I[t - 1] * dt
        I[t] = I[t - 1] + smitte

    plt.plot(tid, I)
    plt.show()
```

Når man kjører denne koden, og setter smitte-koeffisienten til henholdsvis 0.2, 0.25 og 0.48 får man følgende resultater:

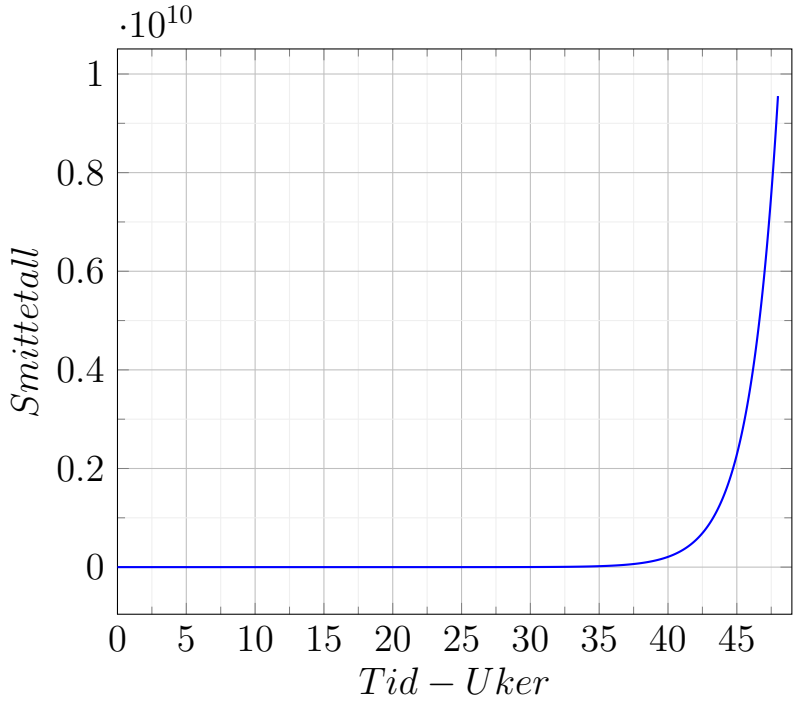
Smittekoeffisient: 0.2



Smittekoeffisient: 0.25



Smittekoeffisient: 0.48



Ser man på resultatet ser man at når smitte-koeffisienten øker, øker det maksimale antallet smittede eksponensielt, og at koeffisienter over 0.25 gir et resultat høyere enn populasjonsstørrelsen, og at den over 0.48 gir et maksimalt resultat større enn antallet mennesker på jorden. Det er viktig at kontaktraten - smitte-koeffisienten - har en enhet fordi det er antallet smittet per tidsenhet, og må derfor ha enheten uke^{-1}.

2 Begrenset smittemottakelige $I_{t+1} = I_t + \alpha I_t S_t$

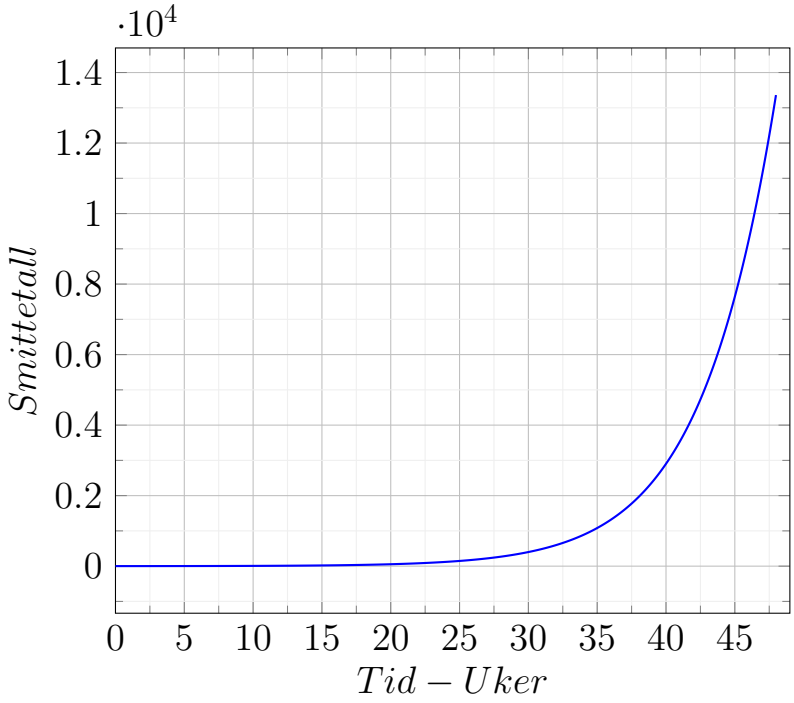
Endringen her er at endringen i smittetall nå er proporsjonal med prosentandelen av befolkningen som er smittbare ("Suseptables"), og fungerer fordi at dette er en makroskopisk modell som ser på gjennomsnittlige endringer i helebefolkningen, altså ikke på individ nivå. Når en person nå blir smittet, vil hen fjernes fra antallet som er smittbare, og vi regner så ut den nye prosentandelen ved at $S(t) = \frac{P-I(t)}{P}$. Dette gjør at smittetallet ikke vokser seg større enn befolkningen, og gjør modellen mer realitisk.

Modellen har likevel en begrenset antall sammenhenger hvor den er nyttig, ettersom at et individ som blir smittet, aldri blir frisk, og dette er en klar begrensning. Den er likevel nyttig for å modellere smittespredning av sykdommer som har lang eller livslang sykdomstid, som f.eks. seksuelle sykdommer.

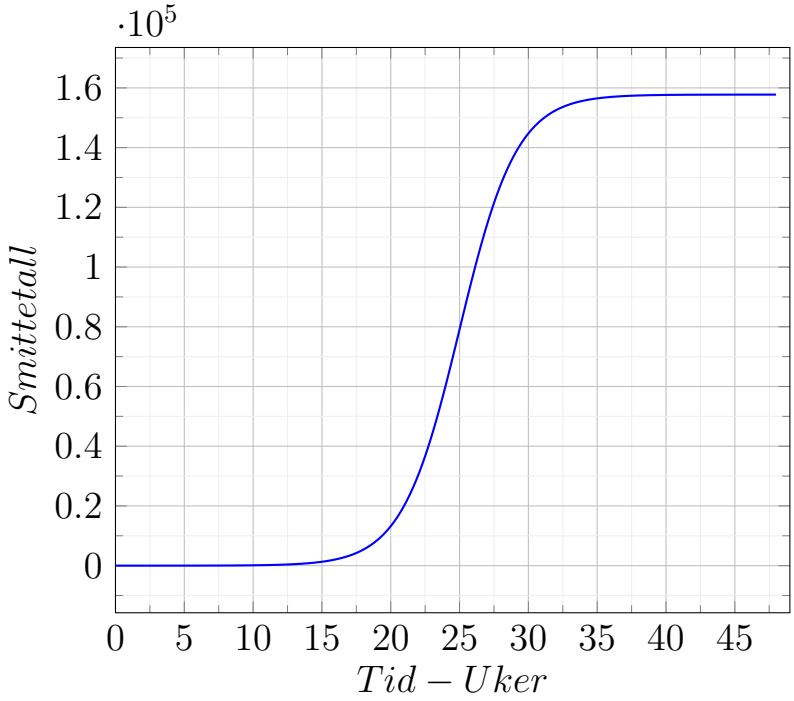
```
# Variabler for simulering
P = 157759 # Populasjons størrelsen

def smitte_modell(smitte_koeffisient):
    # Arrayer for simuleringsdata
    tid = np.arange(0, simulerings_tid, dt) # Tids array
    I = np.zeros(len(tid)) # Smitte array
    S = np.zeros(len(tid)) # Mottakelige ("suseptables") array
    I[0] = 1 # Initialiserer antallet smittede ved t = 0 til 1
    S[0] = P - I[0] # Initialiserer antallet mottakelige til 157758

    for t in range(1, len(tid)):
        # Regner ut vekstfarten og legger det til forrige smittetall
        # og trekker det fra antallet mottakelige
        smitte = smitte_koeffisient * S[t - 1]/P * I[t - 1] * dt
        I[t] = I[t - 1] + smitte
        S[t] = S[t - 1] - smitte

    plt.plot(tid, I)
    plt.show()
```

Resultatet av koden ovenfor gir følgende resultater:

Smittekoeffisient: 0.2



Smittekoeffisient: 0.48



Vi ser at modellen nå ikke "eksploderer" eksponensielt slik den gjorde i den forrige modellen, og at den ved en Smittekoeffisient på 0.48 gjevner seg ut rundt P.

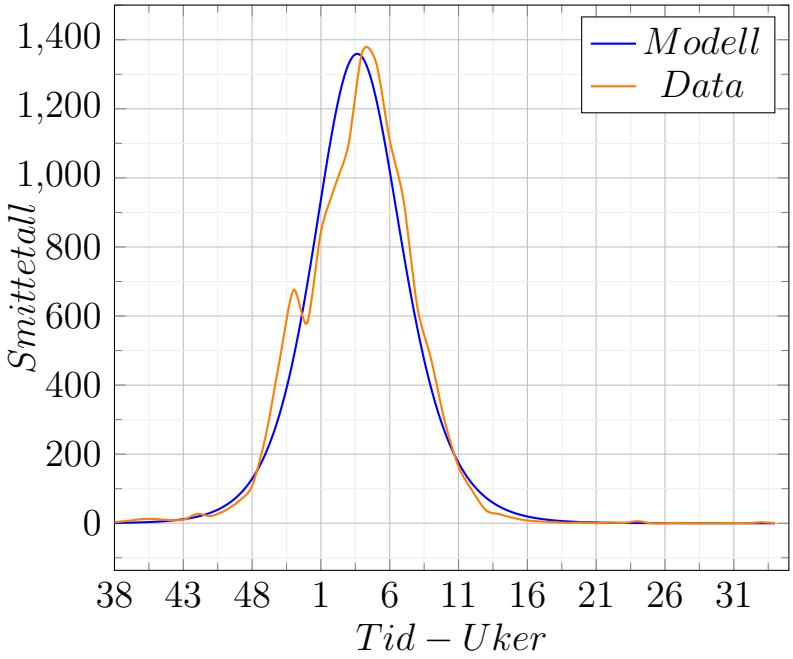
3 Mulighet for å bli frisk $I_{t+1} = I_t + \alpha I_t S_t - \beta I_t$

Man kan forbedre den forrige modellen ved å legge til muligheten for å bli frisk. Dette gjøres ved å holde styr på antallet individer som blir friske, uttrykket ved at endringen i antall friske $friske_{t+1} = friske_t + \beta I_t$, og trekke fra βI_t, både i antallet som kan smittes ("suseptables") og fra endringen i antall smittede I.

En slik type modell er ganske god på å modellere en ekte smittespredning, men er likevel simplistisk i og med at smitteraten er antatt kontinuerlig. Den tar derfor ikke i betraktning påvirkningen av eventuelle ferier, som vil påvirke antallet som kan smittes, betraktlig. Under kan man se et program som bruker denne modellen for å simulere en smittespredning. Den er plottet opp mot faktisk data fra en influensa sesong i 2004 med samme populasjonsstørrelse som modellene er kjørt på ovenfor. Vi observerer at en smittekoeffisient $\alpha \approx 3.8$ og en bedringskoeffisient $\beta \approx 3.3$ gir en god approximering av smitteutviklingen. Dette svarer til at sykdommen smittes videre omlag hver andre dag og at man blir frisk etter omlag 2 dager.

```
# Influensa data
influensa_data = np.loadtxt("./influensa.txt", delimiter=",")
influensa_tid = np.split(influensa_data, 2, 1)[0]
influensa_smitte = np.split(influensa_data, 2, 1)[1]


def smitte_modell(smitte_koeffisient, bedrings_koeffisient):
    # Arrayer for simuleringsdata
    tid = np.arange(0, simulerings_tid, dt) # Tids array
    I = np.zeros(len(tid)) # Smitte array
    S = np.zeros(len(tid)) # Mottakelige ("suseptables") array
    I[0] = 1 # Initialiserer antallet smittede ved t = 0 til 1
    S[0] = P - I[0] # Initialiserer antallet mottakelige til 157758

    for t in range(1, len(tid)):
        # Regner ut vekstfarten til smittetallet og vekstfarten til
        # hvor mange som blir friske og legger til differansen på
        # smittetallet og trekker smitten fra antallet mottakelige
        smitte = smitte_koeffisient * S[t - 1]/P * I[t - 1] * dt
        frisk = bedrings_koeffisient * I [t - 1] * dt
        I[t] = I[t - 1] + (smitte - frisk)
        S[t] = S[t - 1] - smitte
```

Smittekoeffisient: 3.8, Bedringskoeffisient: 3.3



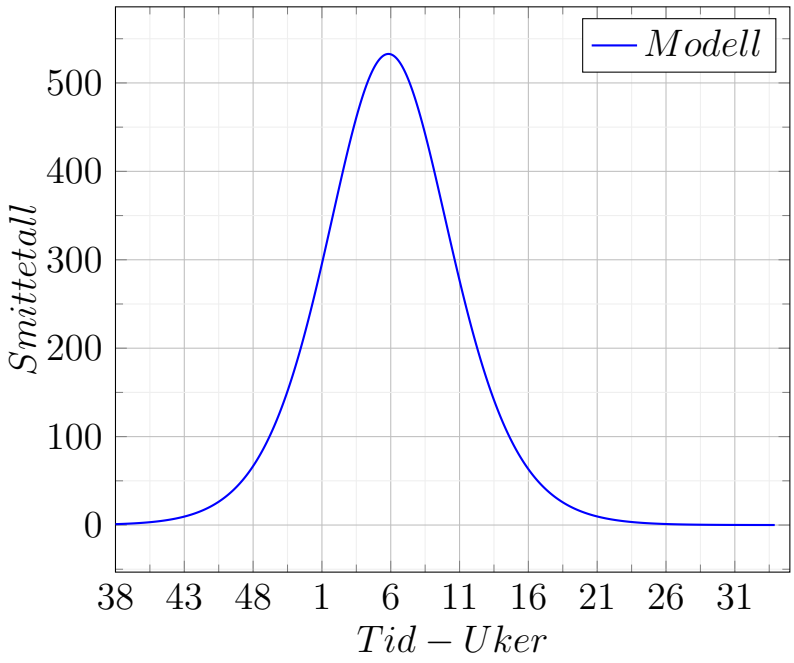
4 Vaksinering

Det er også mulig å utvide modellen videre, ved å legge til vaksinering av befolkningen. Under er et program der en ny størrelse - "Vaksinerte" - blir brukt for å betegne antallet individer som har fått administrert vaksinen, og den blir igjen brukt for å regne ut $S(t)$ ved at $S(t) = (P - F - V)/P$, der P er populasjonsstørrelsen, F er antallet friske/immune og V er antallet vaksinerte. Antallet vaksinerte blir hver uke endret med $\sigma S(t)$, der σ er det maksimale antallet viksiner som kan administreres ukentlig. Resultatet, gitt samme parametre for smittsomhet som i forrige eksempel, gir en "flatere" smittekurve med en lavere smittetopp, noe som er karekteristisk for en slik situasjon.

```
def smitte_modell(alpha, beta, sigma):
    # Arrayer for simuleringsdata
    tid = np.arange(0, simulerings_tid, dt) # Tid
    I = np.zeros(len(tid)) # Smittetall
    F = np.zeros(len(tid)) # Antall immune
    V = np.zeros(len(tid)) # Anatll vaksinerte

    # Initialisering av variabler
    I[0] = 1 # En person er smittet ved t = 0
    immune[0] = 1 # En person er også immun ved t = 0

    for t in range(1, len(tid)):
        dI = alpha * (P - F[t - 1] - V[t - 1])/P * I[t - 1] * dt
        dF = beta * I[t - 1] * dt
        dV = sigma * (P - F[t - 1] - V[t - 1])/P * dt

        # 1. I' = a * S(t) * I(t) - bI(t)
        #    S(t) = (P - F(t) - V(t)) / P
        # 2. F' = I' -> Alle som blir smittet, blir immun
        # 3. V' = sigma * S(t) -> Vaksineringstall, skalert
        I[t] = I[t - 1] + (dI - dF)
        F[t] = F[t - 1] + dI
        V[t] = V[t - 1] + dV

    plt.plot(tid, I)
    plt.show()
```

Smittekoeffisient: 3.8, Bedringskoeffisient: 3.3,

Vaksineringskoeffisient: 500



5 Julekoeffisient

Til slutt, for gøy, kan man legge til muligheten for å modulere antallet smittbare basert på om den nåværende uken ligger innen en høytid, der høytiden i programmet under er juleferien. Den er implementert ved å variere $S(t)$ med en koeffisient Δ i det uken er lik romjulen. Resultatet blir at modellen får en "knekk" i romjulen slik man kan observere i de faktiske dataene vist tidligere.

```
# Koden for å regne ut antal "Susceptibles" S(t)
def S(t, immune, vaksinerte, tid, _julekoeffisient):
    julekoeffisient = 1
    # Hvis den nåværende uken er i romjulen
    if 13 < tid[t] and tid[t] < 14:
        julekoeffisient = _julekoeffisient
    return julekoeffisient * (P - immune[t - 1] - vaksinerte[t - 1])/P
```

```python
# Koden for å kjøre modellen
def smitte_modell(alpha, beta, sigma, delta):
    # Arrayer for simuleringsdata
    tid = np.arange(0, simulerings_tid, dt) # Tid
    I = np.zeros(len(tid)) # Smittetall
    F = np.zeros(len(tid)) # Antall immune
    V = np.zeros(len(tid)) # Anatll vaksinerte

    # Initialisering av variabler
    I[0] = 1 # En person er smittet ved t = 0
    F[0] = 1 # En person er også immun ved t = 0

    for t in range(1, len(tid)):
        S_t = S(t, F, V, tid, delta)
        dI = alpha * S_t * I[t - 1] * dt
        dF = beta * I[t - 1] * dt
        dV = sigma * S_t * dt

        I[t] = I[t - 1] + (dI - dF)
        F[t] = F[t - 1] + dI
        V[t] = V[t - 1] + dV
```

Smittekoeffisient: 3.8, Bedringskoeffisient: 3.3