Proyecto 5: Procesos M/M/s

Diego Valladares Porras, B77867; Jorge Sandi Delgado, B77173; David Naranjo Loría, B95529;
IE0405-Modelos Probabilísticos de Señales y Sistemas, I-2021 - Grupo 12
Escuela de Ingeniería Eléctrica, Universidad de Costa Rica
Profesor: Ing. Fabián Abarca Calderón.

I. Sección teórica

Teniendo en cuenta las siguientes consideraciones:

- Es menester tomar una decisión con respecto al número de "servidores" necesarios para cierto proceso de vacunación, de forma tal que el sistema no exceda 100 personas en fila durante el 95% del tiempo de servicio.
- La tasa de llegada λ es constante durante ocho horas de funcionamiento y tiene un valor de $\lambda = 7$ personas por minuto.
- La tasa de servicio ν de cada "servidor" (es decir, personal de salud tomando datos y vacunando) es constante durante ocho horas de funcionamiento y tiene un valor de $\nu = 0.25$ personas por minuto.

Se procede a determinar teóricamente el número s de servidores necesarios para cumplir el requisito.

Expresando el primer item:

$$1 - \sum_{i=0}^{100}(\phi_i) \leq 0.05 \tag{1}$$

Esta ecuación representa el vector de probabilidad del estado estable para cada estado i, teniendo un sistema M/M/s, entonces:

$$\phi_i = \frac{s^s \rho}{s!}\phi_0 \tag{2}$$

Donde

$$\phi_0 = \left[\sum_{k=0}^{s-1}\frac{(s\rho)^k}{k!} + \frac{(s\rho)^s}{s!(1-\rho)}\right]^{-1} \tag{3}$$

Y

$$\rho = \frac{\lambda}{sv} \tag{4}$$

ρ debe ser mayor a 1, utlizando la ecuación 4 y el valor de v y λ:

$$\frac{7}{0.25s} < 1 \tag{5}$$

Despejando la ecuación 5, s debe de ser mayor a 28
Ingresando el valor de s, ϕ_0 y ϕ_1 a 1:

$$1 - \sum_{i=0}^{100}\frac{s^s\rho}{s!}\cdot\left[\sum_{k=0}^{s-1}\frac{(s\rho)^k}{k!} + \frac{(s\rho)^s}{s!(1-\rho)}\right]^{-1} \leq 0.05 \tag{6}$$

Resolviendo la ecuación 6 se obtiene un valor de s=28.4, por tanto se necesita un mínimo de 29 servidores para cumplir con las especificaciones.

II. Sección programada

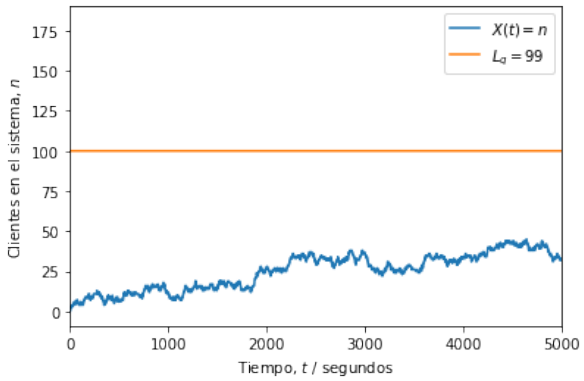

Fig. 1. Potencia real trifásica contra el par mecánico.

Datos de la figura 1: Parámetro λ = 7.0 Parámetro ν = 6.75 Tiempo con más de 99 solicitudes en fila: 26.78%, por tanto no cumple con la especificación. Simulación es equivalente a 8.26 horas.

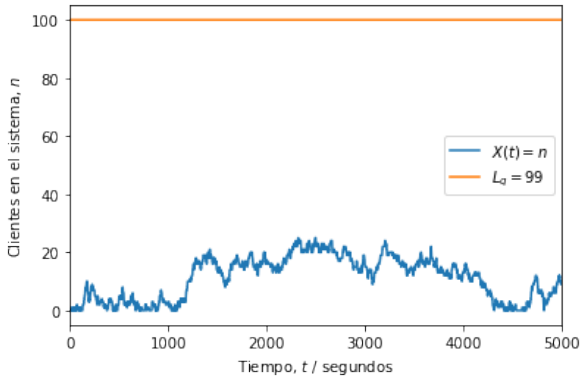

Fig. 2. Potencia real trifásica contra el par mecánico.

Parámetro λ = 7.0 Parámetro ν = 7.25 Tiempo con más de 99 solicitudes en fila: 0.00%. Sí cumple con la especificación. Simulación es equivalente a 8.29 horas.

III. Código

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# N mero de clientes
N = 3200

# Par metro de llegada (clientes/segundos)
lam = 7/60

# Par metro de servicio (servicios/segundos)
nu = (0.25*27)/60

# Distribuci n de los tiempos de llegada entre cada cliente
X = stats.expon(scale = 1/lam)

# Distribuci n de los tiempos de servicio a cada cliente
Y = stats.expon(scale = 1/nu)

# Intervalos entre llegadas (segundos desde  ltimo  cliente)
t_intervalos = np.ceil(X.rvs(N)).astype('int')

# Tiempos de las llegadas (segundos desde el inicio)
t_llegadas = [t_intervalos[0]]
for i in range(1, len(t_intervalos)):
    siguiente = t_llegadas[i-1] + t_intervalos[i]
    t_llegadas.append(siguiente)

# Tiempos de servicio (segundos desde inicio de servicio)
t_servicio = np.ceil(Y.rvs(N)).astype('int')

# Inicializaci n del tiempo de inicio y fin de atenci n
inicio = t_llegadas[0]          # primera llegada
fin = inicio + t_servicio[0]    # primera salida

# Tiempos en que recibe atenci n cada i- simo  cliente (!= que llega)
t_atencion = [inicio]
for i in range(1, N):
    inicio = np.max((t_llegadas[i], fin))
    fin = inicio + t_servicio[i]
    t_atencion.append(inicio)

# Inicializaci n del vector temporal para registrar eventos
t = np.zeros(t_atencion[-1] + t_servicio[-1] + 1)

# Asignaci n de eventos de llegada (+1) y salida (-1) de clientes
for c in range(N):
    i = t_llegadas[c]
    t[i] += 1
    j = t_atencion[c] + t_servicio[c]
    t[j] -= 1

# Umbral de P o m s personas en sistema (hay P - 1 en fila)
P = 101

# Instantes (segundos) de tiempo con P o m s solicitudes en sistema
exceso = 0

# Proceso aleatorio (estados n = {0, 1, 2...})
Xt = np.zeros(t.shape)

# Inicializaci n de estado n
n = 0

# Recorrido del vector temporal y conteo de clientes (estado n)
for i, c in enumerate(t):
    n += c # sumar (+1) o restar (-1) al estado
    Xt[i] = n
    if Xt[i] >= P:
        exceso += 1

# Fracci n de tiempo con P o m s solicitudes en sistema
```

```
fraccion = exceso / len(t)

# Resultados
print('Par metro lambda =', str(lam*60))
print('Par metro nu =', str(nu*60))
print('Tiempo con m s de {} solicitudes en fila:'.format(P-2))
print('\t {:0.2f}%'.format(100*fraccion))
if fraccion <= 0.01:
    print('\t S  cumple con la especificaci n.')
else:
    print('\t No cumple con la especificaci n.')
print('Simulaci n es equivalente a {:0.2f} horas.'.format(len(t)/3600))

# Gr fica de X(t) (estados del sistema)
plt.figure()
plt.plot(Xt)
plt.plot(range(len(t)), (P-1)*np.ones(t.shape))
plt.legend(('$X(t) = n$', '$L_q = $' + str(P-2)))
plt.ylabel('Clientes en el sistema, $n$')
plt.xlabel('Tiempo, $t$ / segundos')
plt.xlim((0, 5000))
plt.show()
```