Ignasi Pons Montasell

Grau en Enginyeria de les Tecnologies Industrials

Gener 2016

Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Simulació i anàlisi de sistemes de dosificació i pesatge dinàmic

ANNEX A: Funcionament de la llibreria PyQt4.
ANNEX B: Gràfics obtinguts en fer l'anàlisi de la resposta.
ANNEX C: Com executar el simulador.
...

Autor: Ignasi Pons Montasell
Director/s: Pere Grima Cintas
Convocatòria: Gener 2016

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

UPC

ANNEX A: Funcionament de la llibreria PyQt4

Funcionament general

Per fer funcionar una aplicació amb aquesta llibreria fa falta introduir algunes coses en el codi del programa de l'aplicació que es vol fer. L'estructura del programa principal ha de tenir dues línies de codi per inicialitzar la llibreria, i entre mig pot anar la creació de finestres de l'aplicació i el cos del programa. Aquestes primeres línies criden la llibreria, que ha d'estar prèviament instal·lada en el ordenador de treball i crea la variable app, que és l'aplicació en si. Aquesta treballa en forma de bucle fins que se li dona l'ordre de tancar-se.

```
from PyQt4 import QtGui
import sys

app = QtGui.QApplication(sys.argv)
"""
- Creació i addició de finestres
- Cos del programa
"""
sys.exit(app.exec_())
```

Creació i addició de finestres

Hi ha diverses maneres diferents d'estructurar les finestres en una aplicació amb PyQt4, tot i això, aquí s'explicarà la manera que s'ha trobat més òptima i fàcil per fer-ho i entendre-ho, tot mostrant les dues formes que hi ha de crear finestres per afegir a l'aplicació.

Tota aplicació funciona a partir d'una finestra principal que es diu QMainWindow. Aquesta es configurada des de el fitxer mainwindow.py, i forma part del mòdul PyQt4.QtGui. Aquesta finestra es la que s'obra en iniciar l'aplicacció i allà i podem afegir tots els elements que vulguem. La millor manera de crear aquest element es creant una classe que hereti de l'element *QMainWindow* i afegir-hi tots els mètodes i funcions que facin falta. Aquestes funcions són totalment opcionals i personalitzades i acostumen a definir l'aspecte de la finestra en quan a dimensions i títol. A continuació es mostra la creació de la classe *MainWindow* amb alguns paràmetres de personalització introduïts, que estarien en fitxer mainwindow.py.

```
class MainWindow(QtGui.QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.definirParametres()

    def definirParametres(self):
        self.resize(1000, 600)
        self.setWindowTitle("Aplicació de prova")
        self.setWindowIcon(QtGui.QIcon("Imatges/imatge_de_prova.png"))
```

El que s'ha creat fins ara és el lloc on aniran les finestres que contenen tots els elements (botons, texts, etc...) i tota aplicació que es crea requereix com a mínim una finestra. Les finestres es poden crear de dues maneres, la primera és creant una classe manualment que hereti de *QWidget* del mòdul *PyQt4.QtGui* i afegint tots els elements necessaris a mà. Això implica programar un per un cada element, la seva posició, la seva mida, el seu color i tot els seus paràmetres. Aquest mètode és poc eficient i complex, a més que no permet crear interfícies complexes.

Un altre mètode possible, que ha estat l'utilitzat en el treball consisteix en crear la interfície a partir de un fitxer .ui. Aquest fitxer es crea a partir d'un dissenyador anomenat Qt Designer, que permet configurar les finestres i els seus paràmetres de forma visual i lògica. Un cop creat aquest fitxer, tan sols hem de passar el fitxer.ui a .py, amb unes línees de codi que ho fan automàticament, i a continuació hem de programar les interaccions i les funcions que cridaran cada vegada que es seleccioni alguna cosa del programa. Les línees de codi que cal introduir per a incorporar aquest fitxer i utilitzar-lo com a finestra dins de la finestra principal són les següents:

```
import widget_principal

finestra = QtGui.QWidget()
classe_finestra = widget_principal.Ui_WidgetPrincipal()
classe_finestra.setupUi(finestra)

        ui = MainWindow()
        ui.setCentralWidget(finestra)
        ui.show()
```

Amb la finestra creada pel mètode que sigui ja es pot afegir a la *MainWindow* i mostrar aquesta. Per fer això abans d'acabar l'execució de PyQt4 s'ha de cridar la classe i afegir-li com a finestra central la finestra creada amb el mètode *setCentralWidget* de la instància *ui.* Finalment es crida el seu mètode *show* per visualitzar-la. Aquestes línies de codi serien el final del cos del programa de l'aplicació.

Cos del programa

En el cos del programa pot anar-hi qualsevol funció i operació que es vulgui realitzar amb la interfície funcionant. La part més important són les connexions, que s'encarreguen d'interactuar amb d'interfície i cridar les funcions corresponents.

Entenem per conexions una funció de python que identifica quan s'ha clicat un botó i hi fa alguna acció en conseqüència. La manera de crear les connexions és agafant l'objecte que es vol connectar, seguit de l'esdeveniment que ha d'ocórrer, i seguit del mètode *connect* passant com a paràmetre la funció a connectar. A la imatge següent es crida la classe que representa la finestra central amb l'esdeveniment *clicked* i es connecta amb la funció *funcioDelBoto1*.

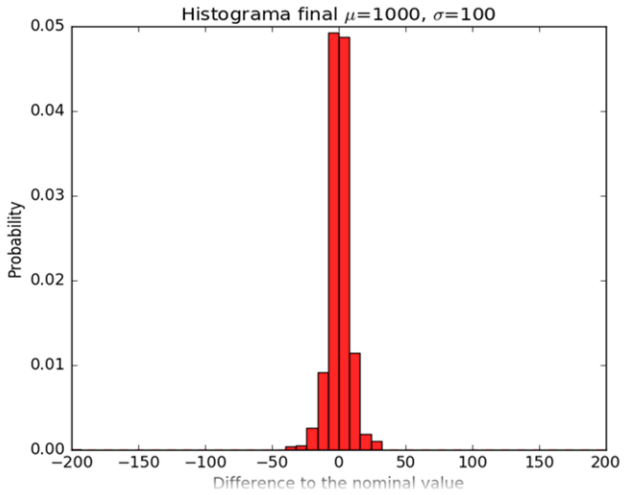
```
def funcioDelBoto1():
    print "S'ha clicat el botó!"

classe_finestra.boto1.clicked.connect(funcioDelBoto1)
```

Les funcions que es criden en clicar un botó no poden no haurien de tenir paràmetres ja que complica molt la seva programació. Tot i això, amb les connexions fetes a l'inici del cos del programa es poden crear totes les funcions que es trobin necessàries per definir la manera de treballar de l'aplicació amb interfície, ja que des de dins la funció cridada al clicar, es pot cridar qualsevol tipus de funció amb qualsevol tipus d'atribut.

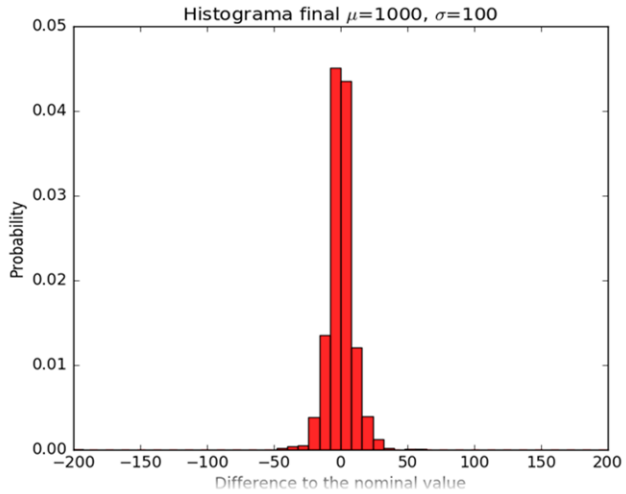
ANNEX B: Gràfics obtinguts en fer l'anàlisi de la resposta

En aquesta part de l'annex, s'hi poden trobar alguns dels gràfics que s'han obtingut en els exemples analitzats durant el treball. No s'han posat tots per reduir espai, però es poden obtenir fàcilment utilitzant el simulador.

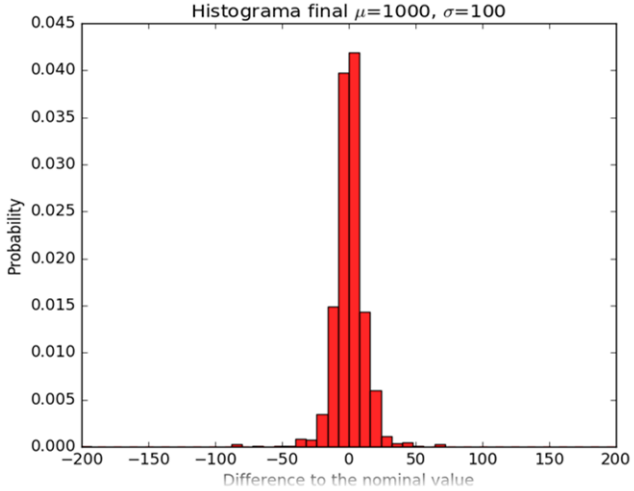
Flux discret. Política d'eliminació: "When the hoppers remain closed in x operations":



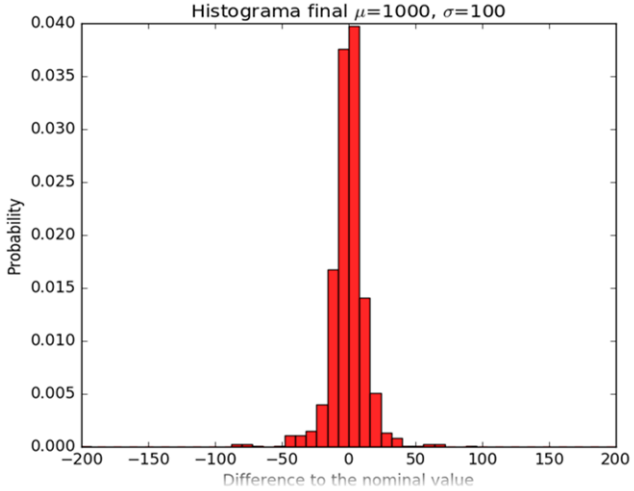
Tolves tancades 2 operacions



Tolves tancades 3 operacions

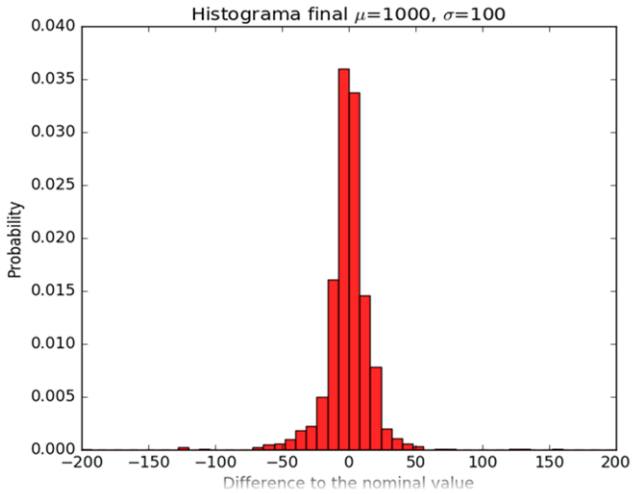


Tolves tancades 5 operacions

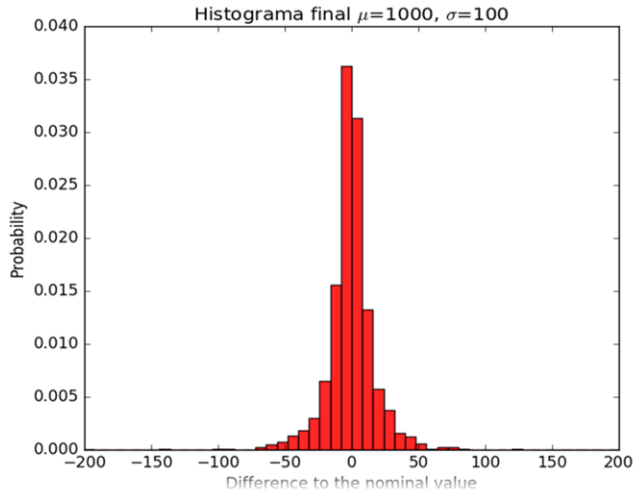


Tolves tancades 7 operacions

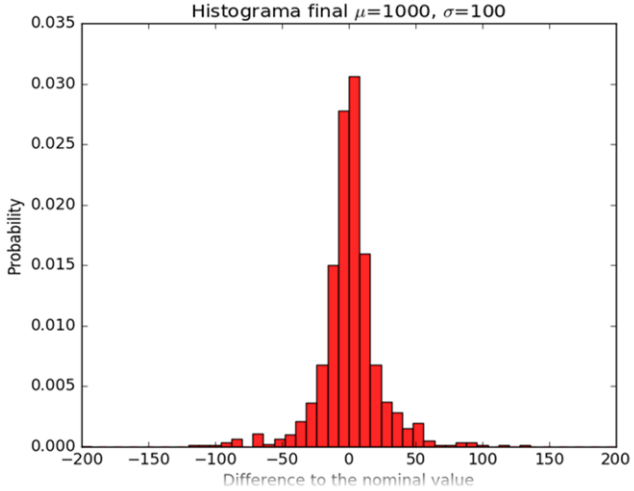
Flux discret. Política d'eliminació: "When the hoppers remain closed in x operations:



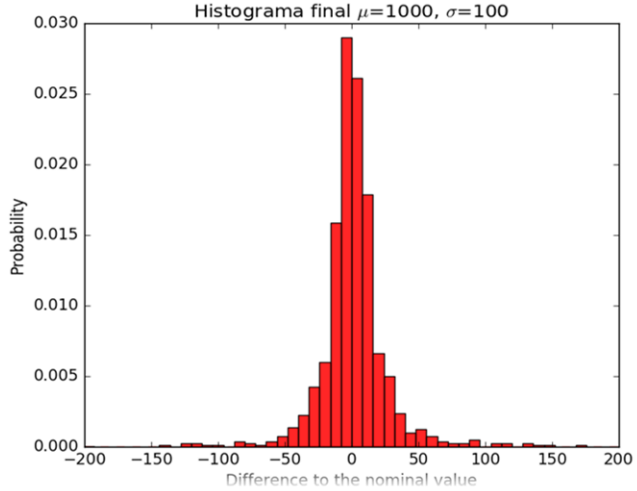
Tolves tancades 10 operacions



Tolves tancades 15 operacions

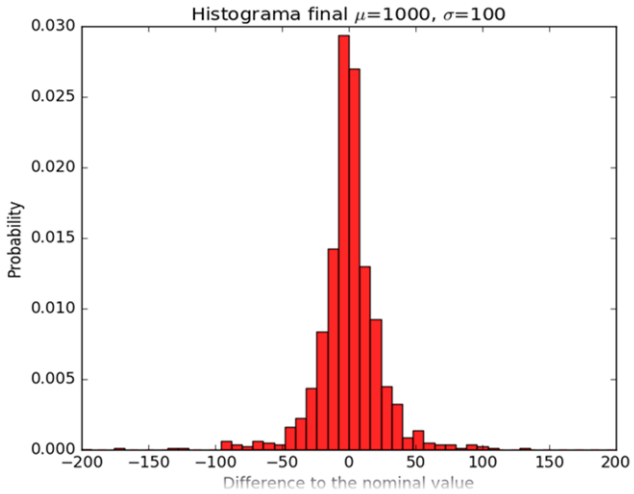


Tolves tancades 20 operacions

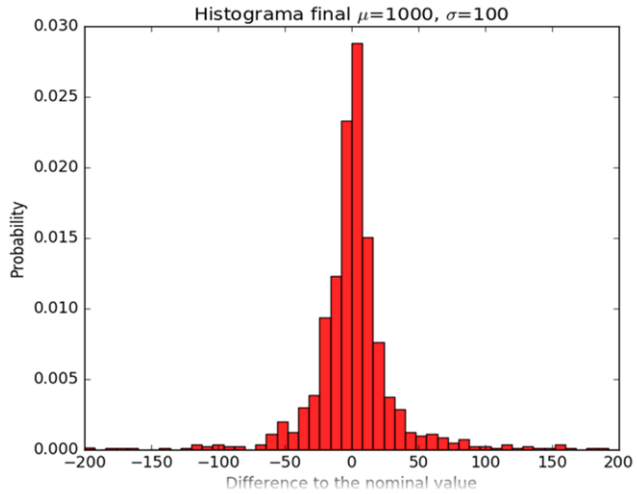


Tolves tancades 25 operacions

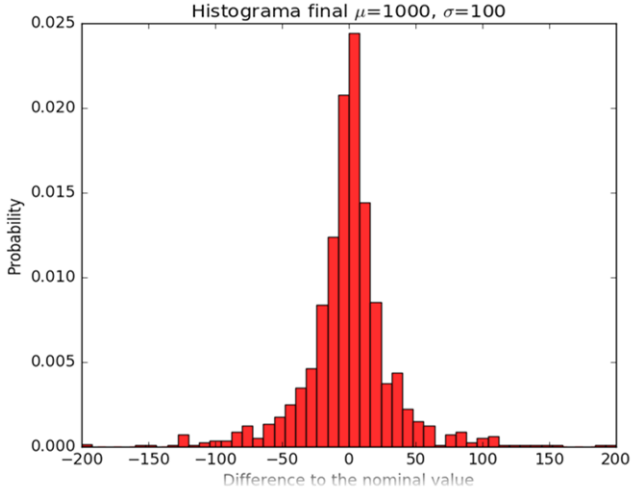
Flux discret. Política d'eliminació: "When the hoppers remain closed in x operations:



Tolves tancades 30 operacions

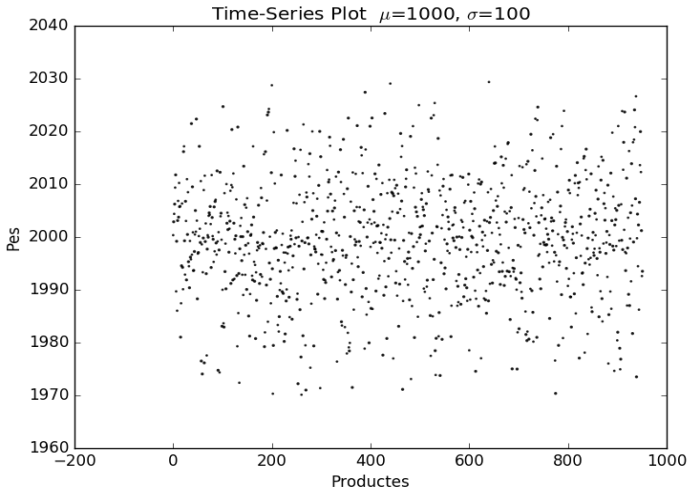


Tolves tancades 40 operacions

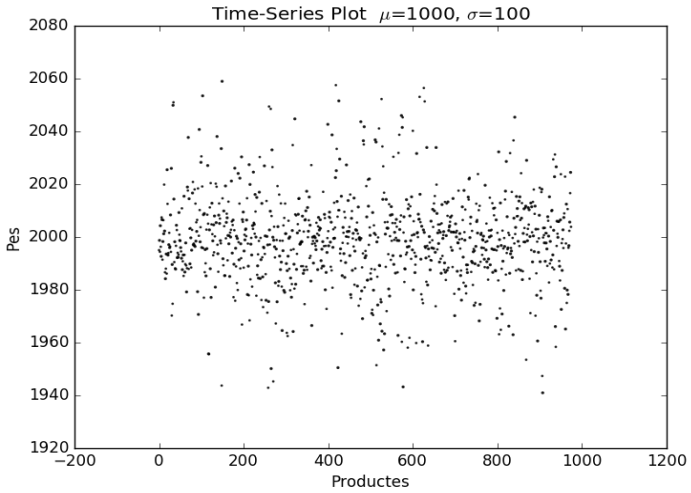


Tolves tancades 50 operacions

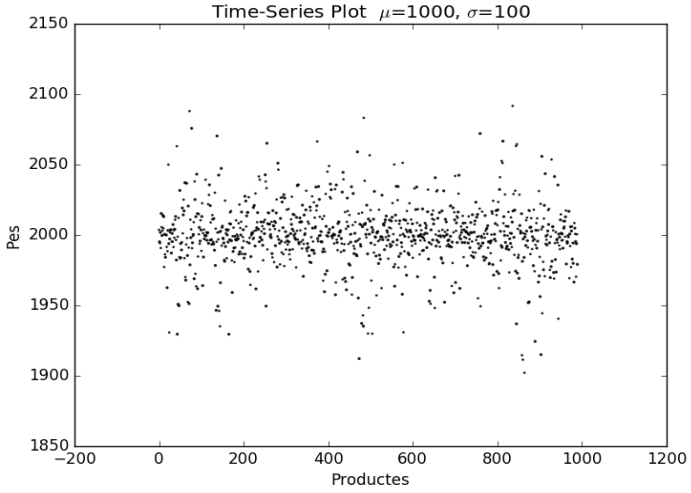
Flux discret. Política d'eliminació: "Be superior or inferior to NV in a % of:



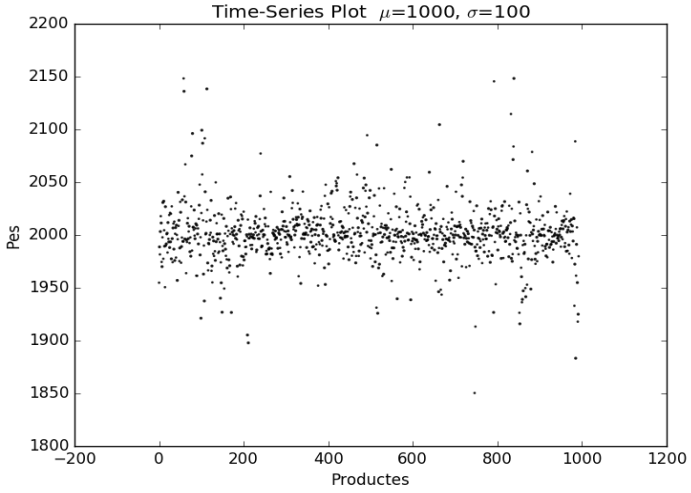
No difereixin en més d'un 1,5%



No difereixin en més d'un 3%

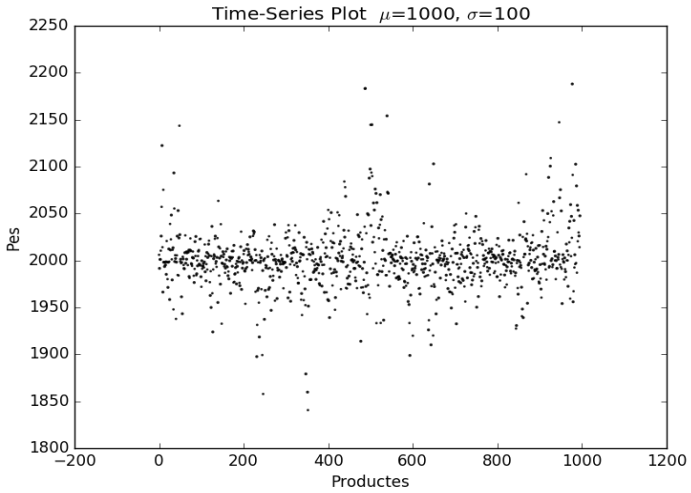


No difereixin en més d'un 5%

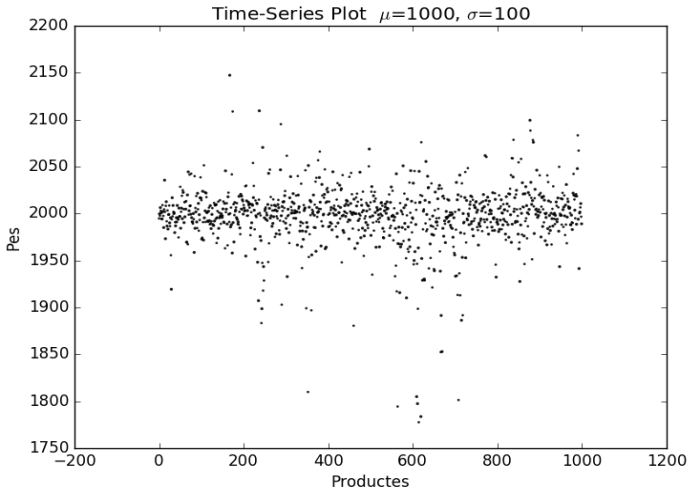


No difereixin en més d'un 7,5%

Flux discret. Política d'eliminació: "Be superior or inferior to NV in a % of:

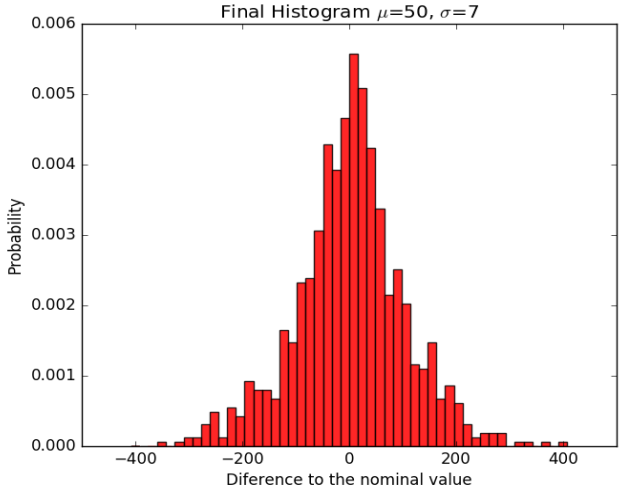


No difereixin en més d'un 10%

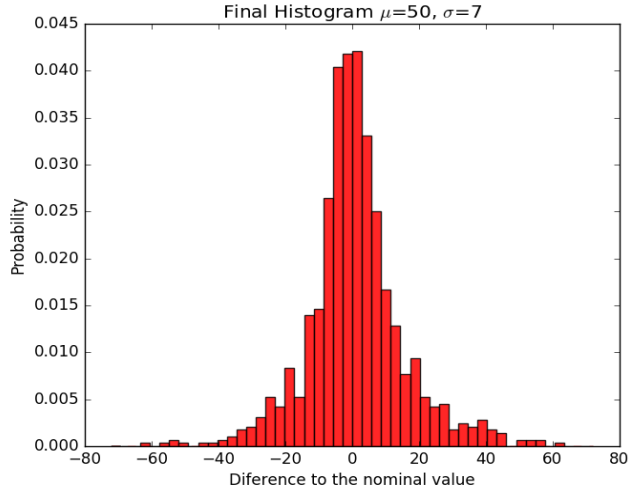


No difereixin en més d'un 15%

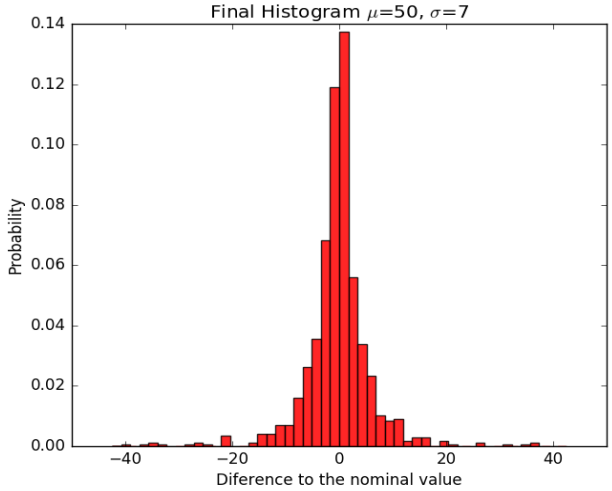
Flux varies unitats. Com afecta el nombre de tolves buidades per operació?



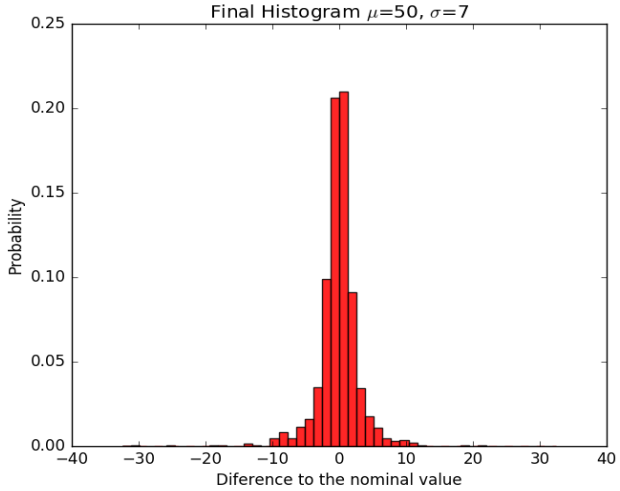
10%



20%

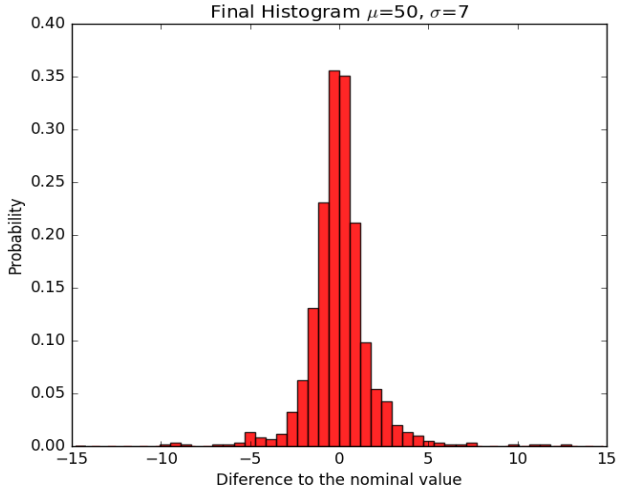


30%

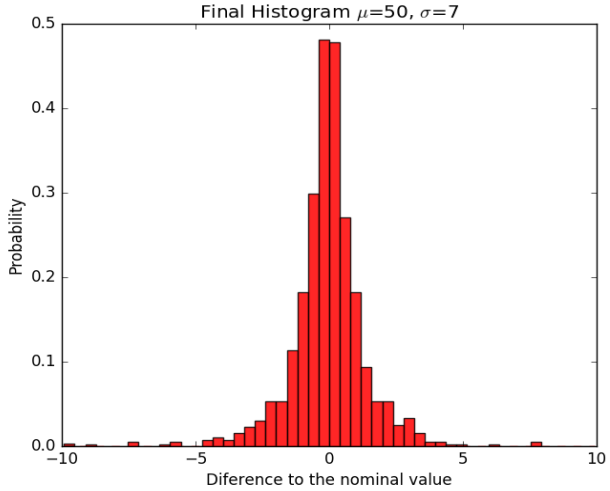


40%

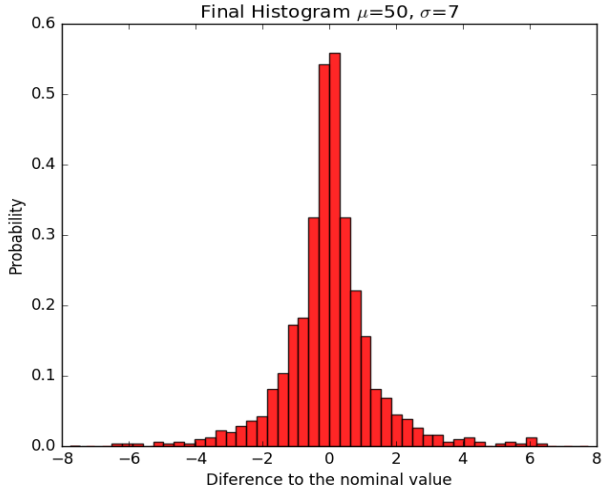
Flux varies unitats. Com afecta el nombre de tolves buidades per operació?



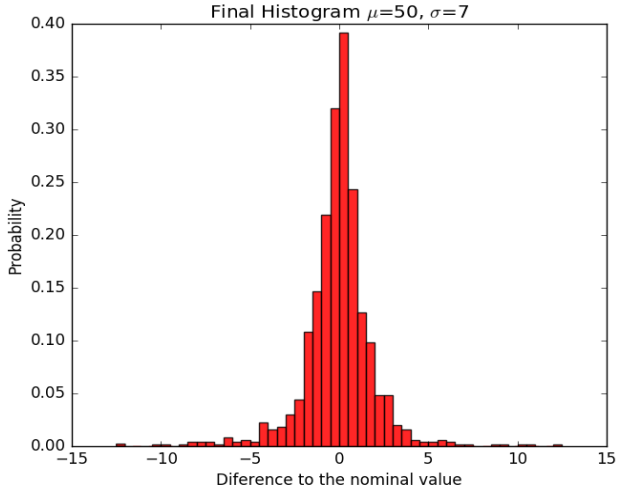
50%



60%

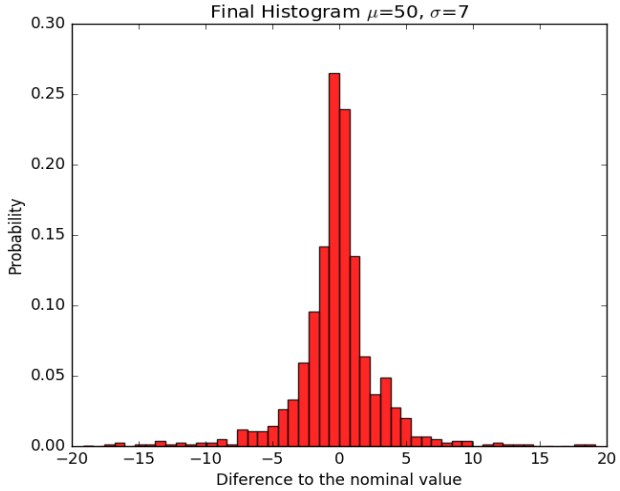


70 %

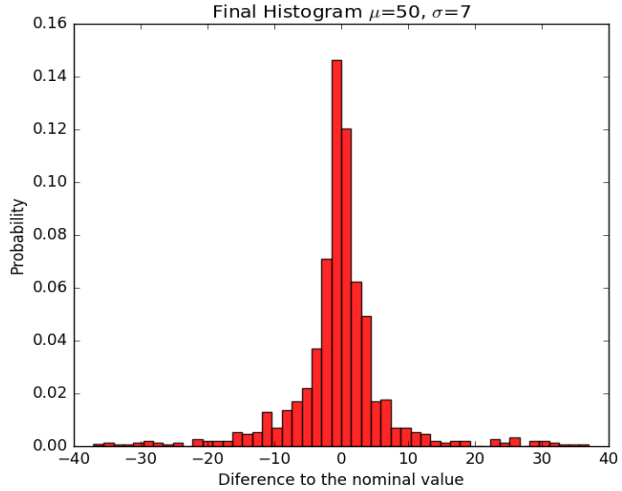


80%

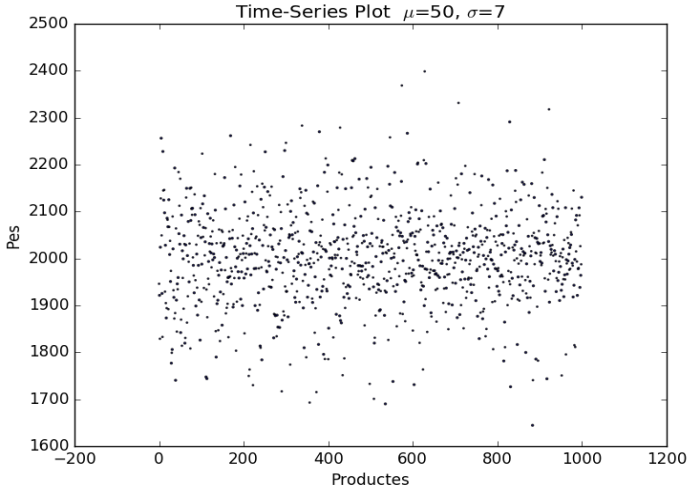
Flux varies unitats. Com afecta el nombre de tolves buidades per operació?



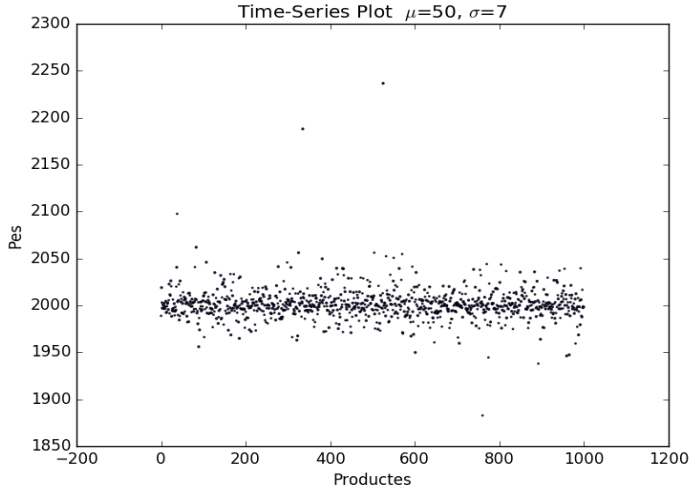
90%



100%

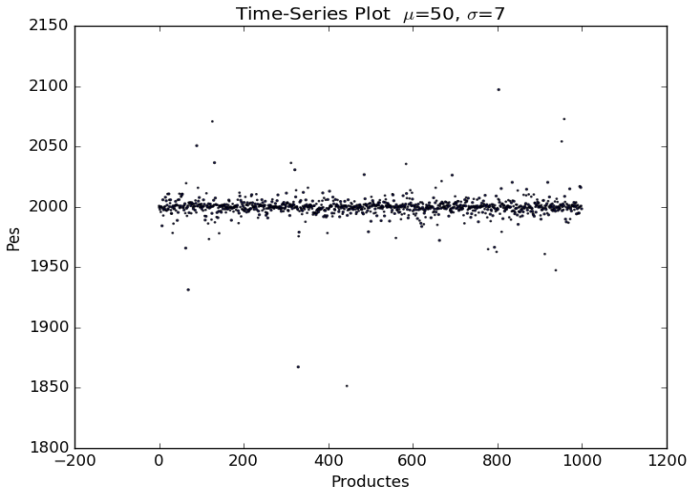


TS 10%

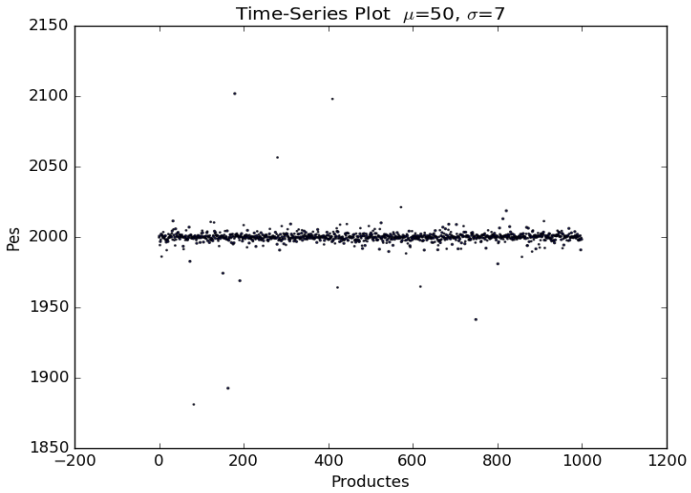


TS 20%

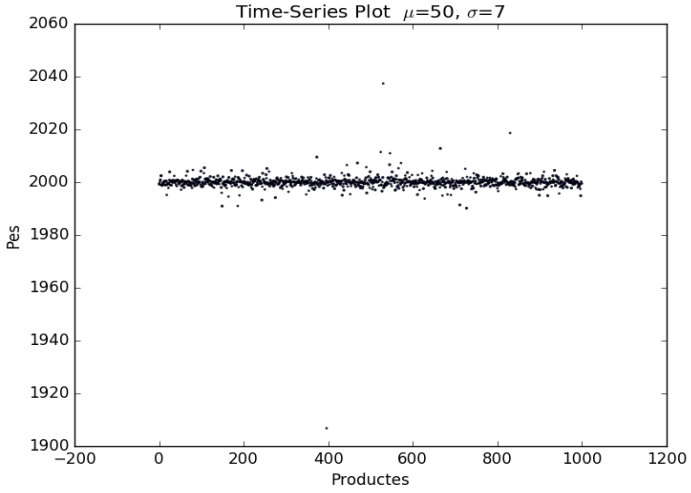
Flux varies unitats. Com afecta el nombre de tolves buidades per operació?



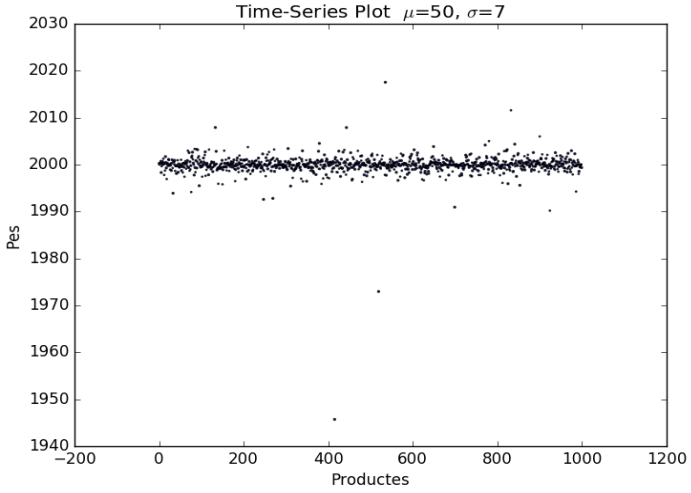
TS 30%



TS 40%

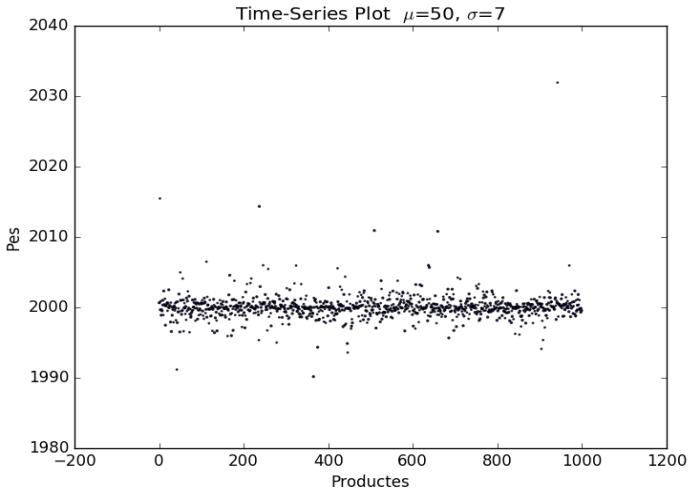


TS 50%

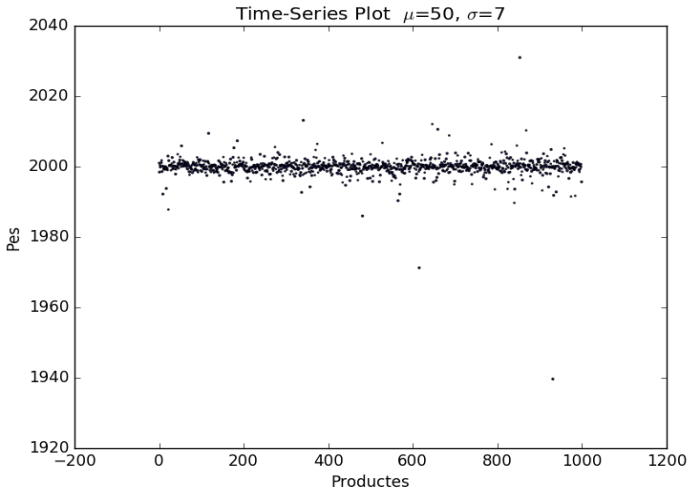


TS 60%

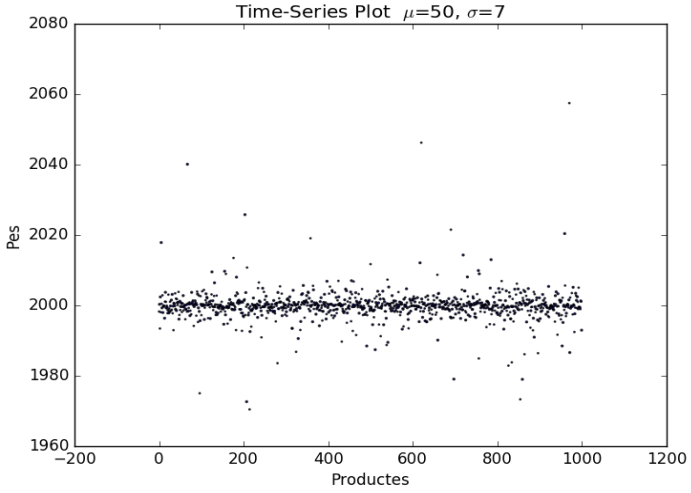
Flux varies unitats. Com afecta el nombre de tolves buidades per operació?



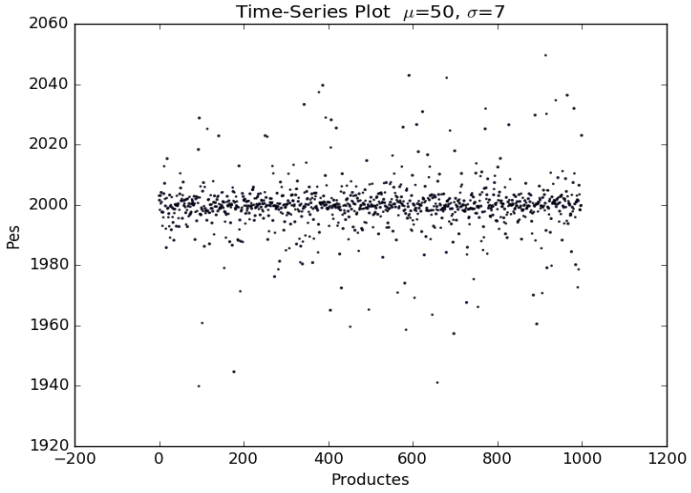
TS 70%



TS 80%

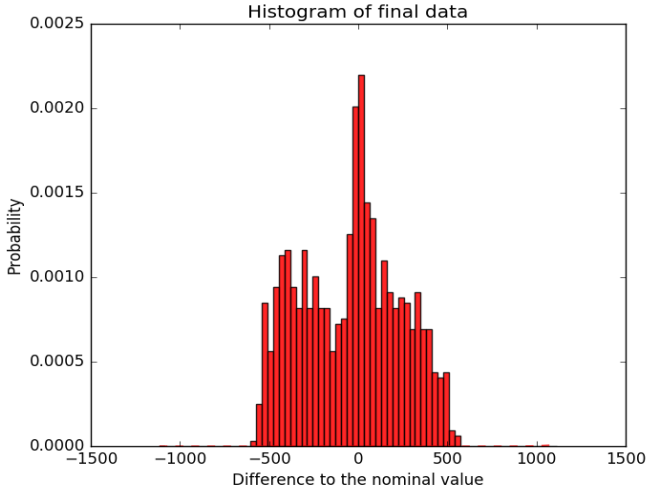


TS 90%

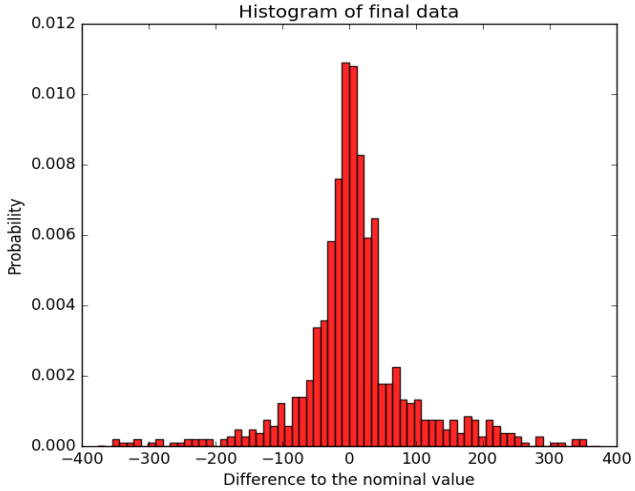


TS 100%

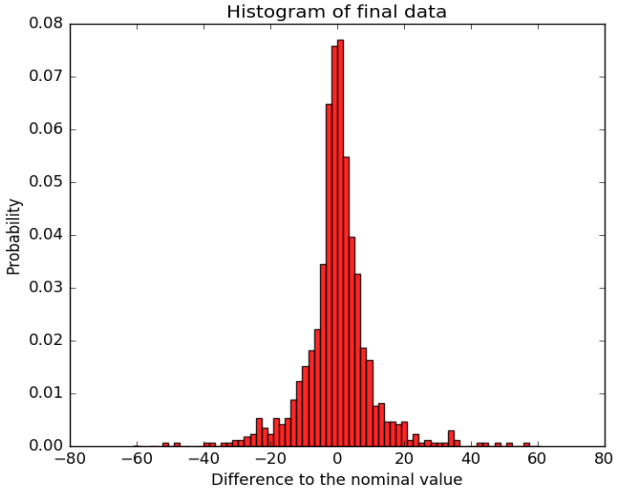
Flux continu. Com afecta el nombre de tolves buidades per operació?



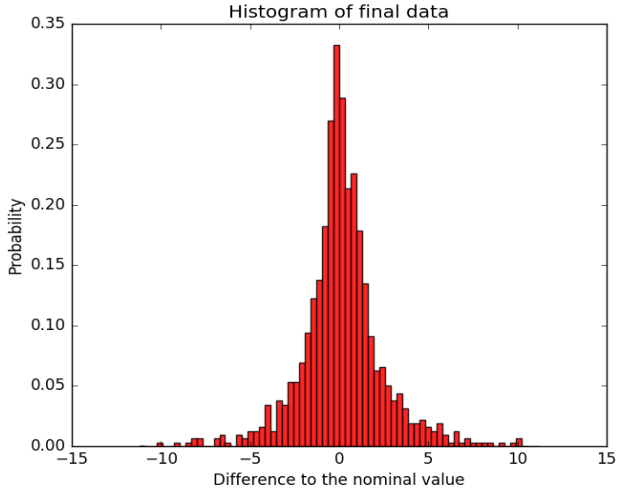
10%



20%

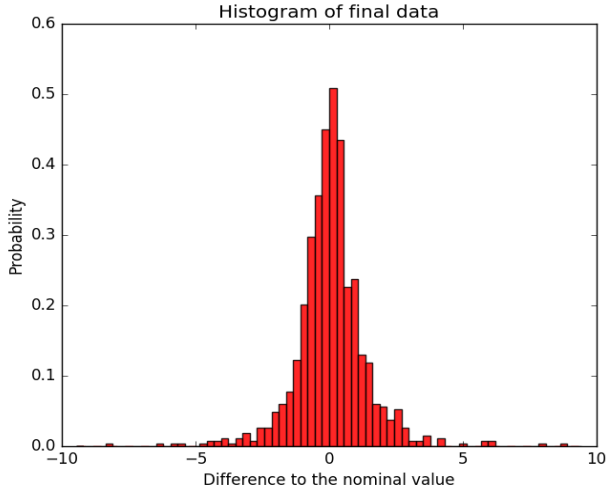


30%

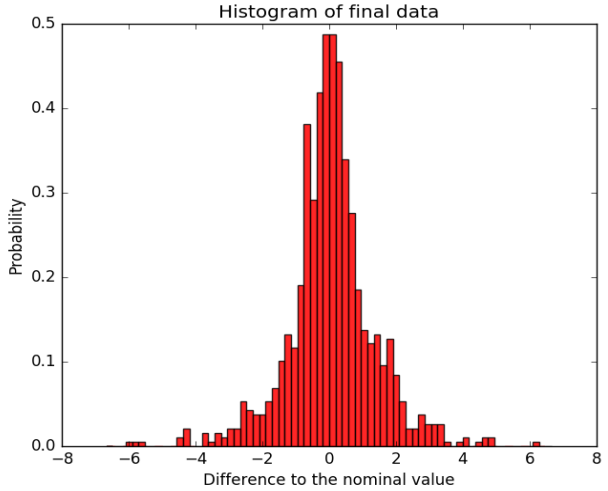


40%

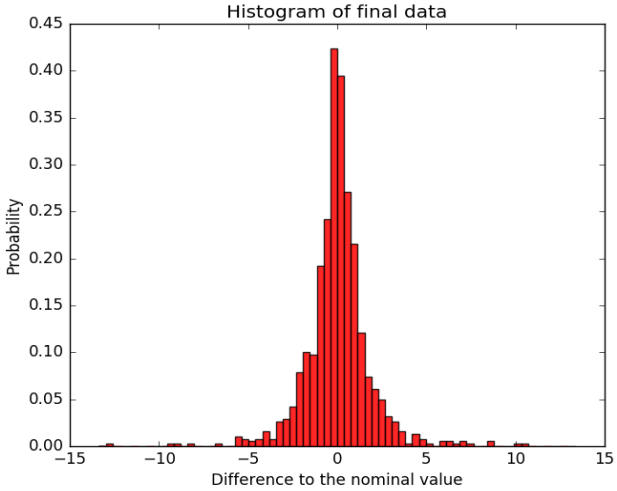
Flux continu. Com afecta el nombre de tolves buidades per operació?



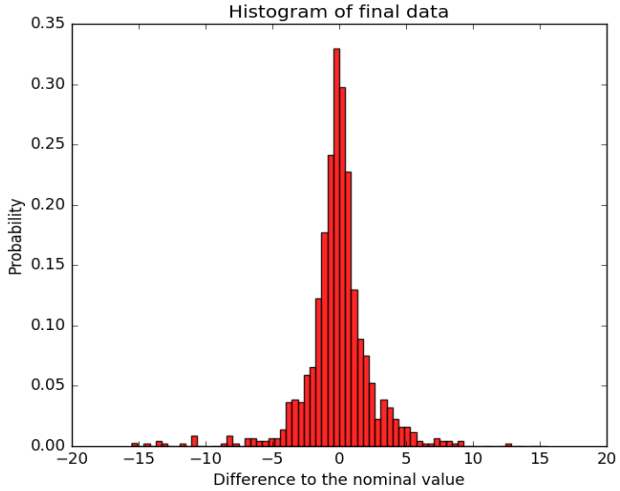
50%



60 %

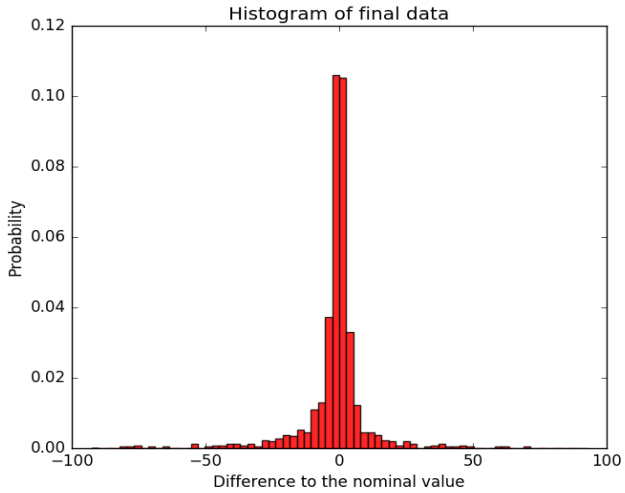


65%

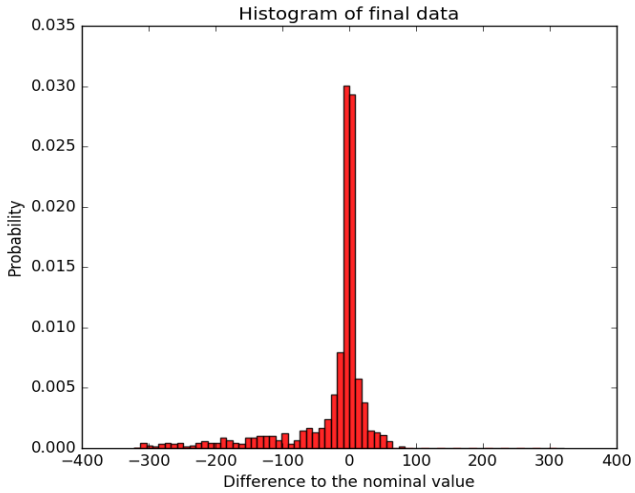


70%

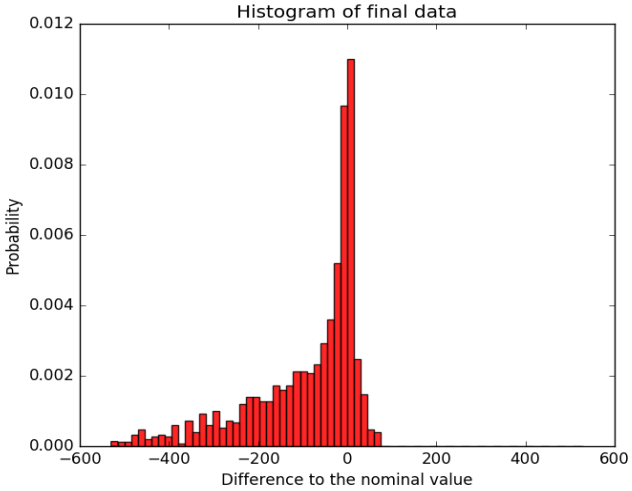
Flux continu. Com afecta el nombre de tolves buidades per operació?



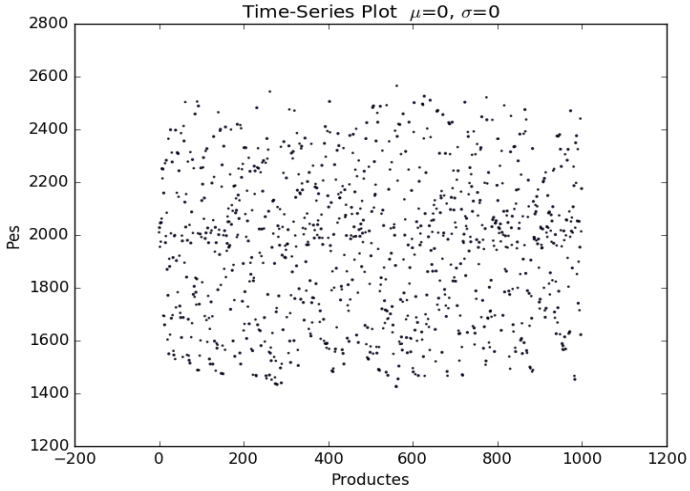
80 %



90%

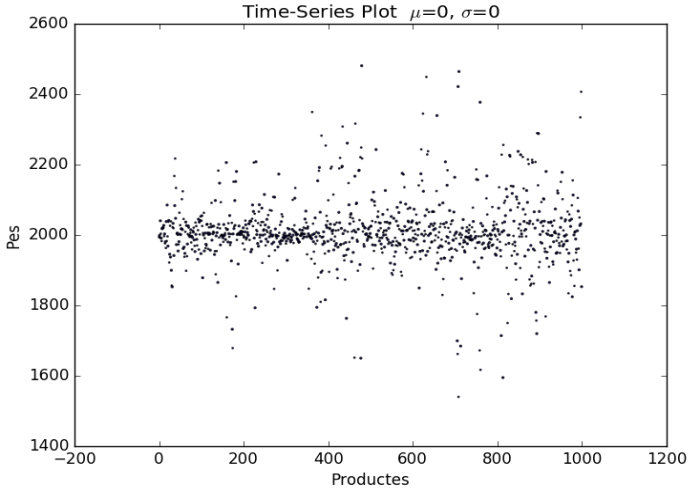


100%

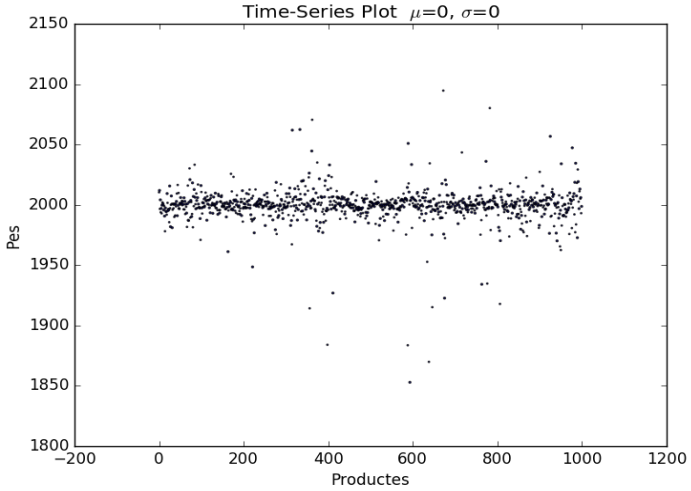


TS 10%

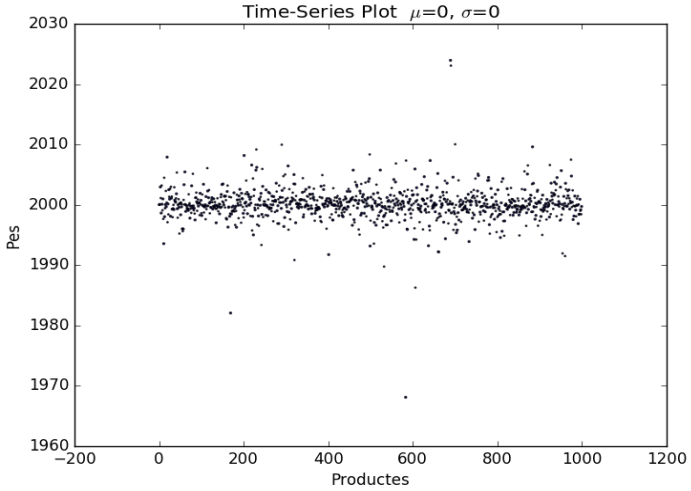
Flux continu. Com afecta el nombre de tolves buidades per operació?



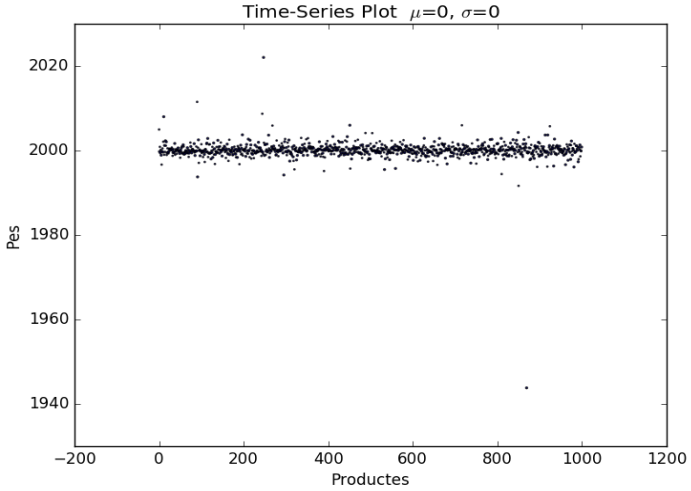
TS 20 %



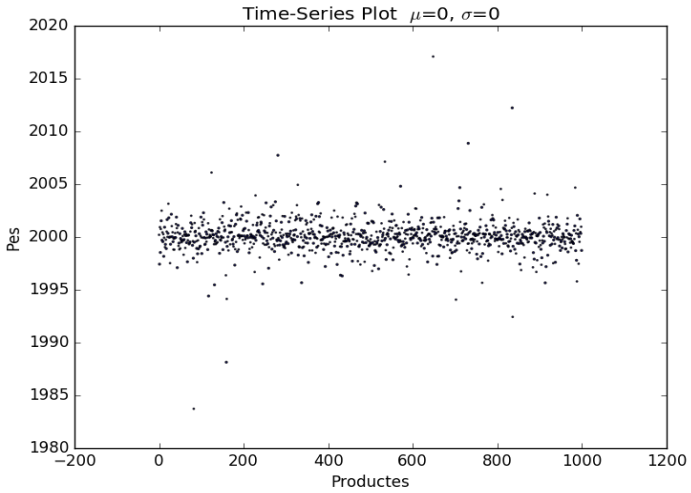
TS 30%



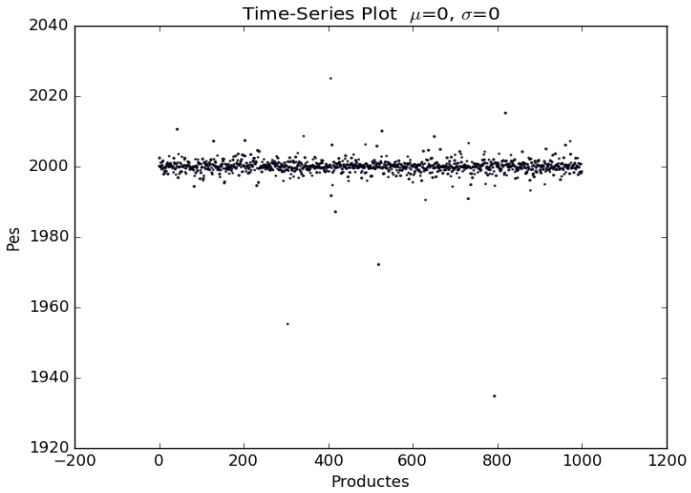
TS 40%



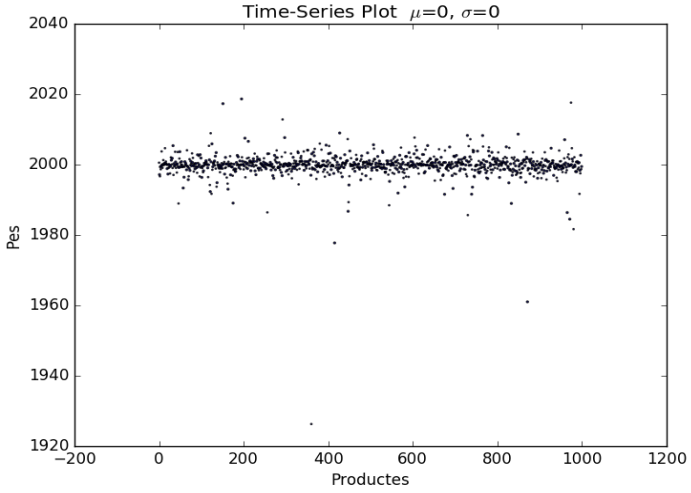
TS 50 %



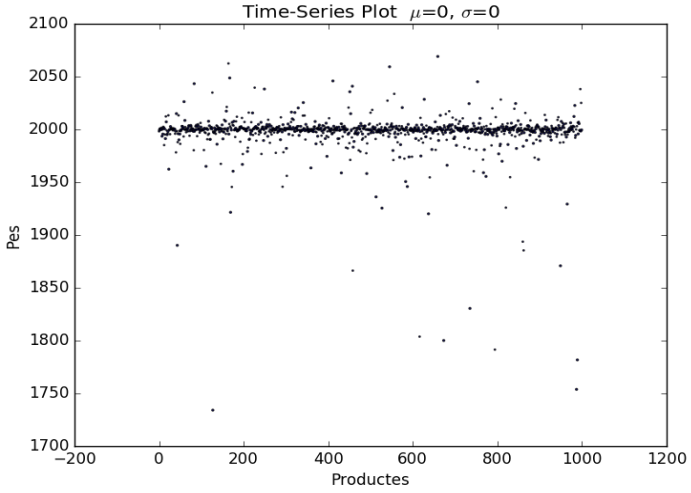
TS 60%



TS 65%

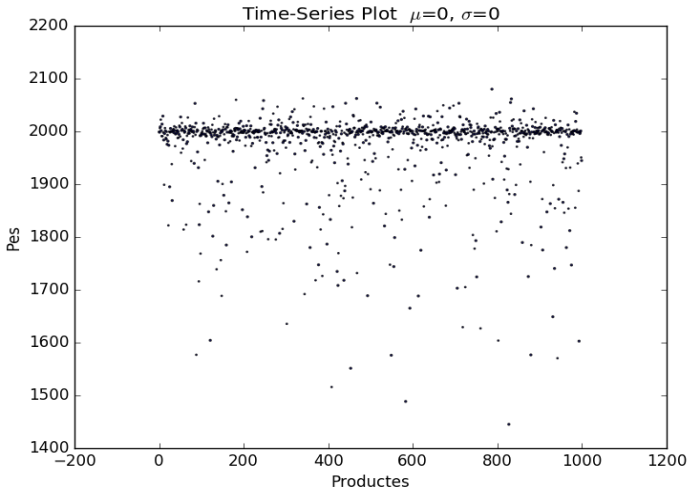


TS 70%

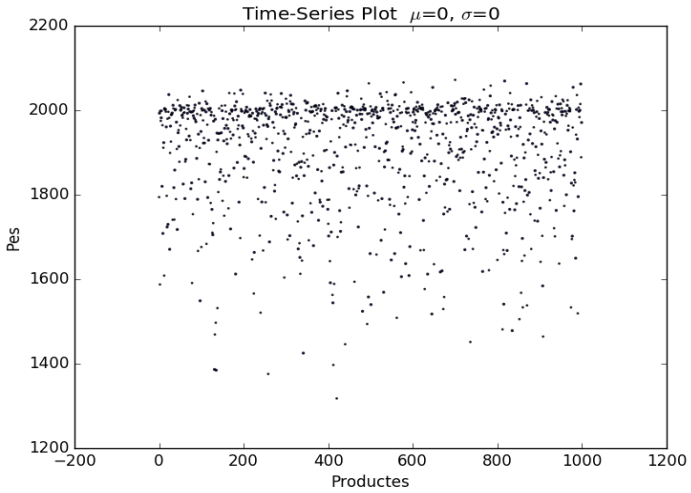


TS 80%

Flux continu. Com afecta el nombre de tolves buidades per operació?



TS 90 %



TS 100%

ANNEX C: Com executar el simulador.

El simulador creat es troba en el cd adjunt en el treball, juntament amb tot el treball en format pdf. Per a utilitzar-lo cal seguir els següents passos:

- Introdueixi el CD-ROM al ordenador.
- Obri el seu contingut i copiï el fitxer dist.rar al escriptori.
- Descomprimeix-hi el arxiu en la carpeta desitjada. (En cas de que no disposi de descompressor instalat, el CD-ROM inclou el programa descomprimit en una carpeta anomenada dist)
- Obri la carpeta descomprimida dist i executi l'aplicació "*Weighting simulator.exe"* que està al final de la carpeta.
- Per qualsevol dubte no dubtin en contactar-me, a pmignasi@gmail.com, o a ignasi.pons@upc.edu

A més en el CD-ROM hi podreu trobar el codi de les funcions principals del programa i del cos del programa, amb una breu descripció de que fa cadascuna d'elles.