index

April 27, 2018

Este documento contiene una serie de ejemplos de aplicación de las recomendaciones del articulado de la ROM 1.1. Estos ejemplos están implementados en Jupyter, un entorno que permite combinar: celdas de texto con las explicaciones teóricas, celdas de código con la programación de la herramienta en Python y celdas de salida con las salidas gráficas de la herramienta.

Para que funcionen, los notebooks deben ser descargados en la misma carpeta donde se encuentran los códigos de los diferentes módulos (src.zip) y los ficheros de entrada (ejemplos.zip), que se pueden descargar desde los enlaces proporcionados en el correo.

Estos ejemplos han sido programados y redactados por los miembros del Grupo de Dinámica de Flujos Ambientales de la Universidad de Granada.

1 Índice

1.4 Verificacion de la alineacion principal del dique

- Concepción de la obra
- Lectura de estados característicos, comportamiento hidráulico y análisis de modos de fallo

1.5 Ejemplo para el cálculo de costes de construcción en el alcance de estudios previos

- Lectura, cálculo y resultados

1.6 Cálculo de costes de construccion en el alcance de estudios soluciones clase I (Estrategia protectora avance en paralelo)

- Parte 1 - Lectura de los datos de entrada de la forma en planta
- Parte 2 - Lectura de los datos de entrada asociados a la descripción del proceso constructivo
- Parte 3 - Verificación del proceso constructivo mediante simulación numérica

1.7 Cálculo de costes de reparación en el alcance de estudios soluciones clase I

- Parte 1 - Lectura de los datos de entrada
- Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil
- Parte 3 - Representación de resultados y cálculo de costes

1.8 Cálculo de costes de reparación en el grado de desarrollo de proyecto de inversión

- Parte 1 - Lectura de los datos de entrada
- Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil
- Parte 3 - Representación de resultados y cálculo de costes
- Postproceso, agrupación de resultados obtenidos en las simulaciones del cálculo de costes de reparacion en el grado de desarrollo de proyecto de inversión

cl_lectura_pretratamiento_01

April 27, 2018

0.0.1 Lectura de las series temporales de los agentes climáticos marítimos y atmosféricos

Este ejemplo presenta la lectura de una serie temporal de variables marítimas y atmosféricas y un análisis de la calidad de los datos. En este caso se realiza la lectura de las variables de oleaje del punto SIMAR 1052046 proporcionado por Puertos del Estado.

Las funciones que se utilizan son:

- entrada_datos.lectura.lectura_simar
- lectura.plot_series_temporales
- entrada_datos.pretratamiento.analisis_calidad_series
- entrada_datos.pretratamiento.establecer_cadencia_series

Asimismo, se proporcionan herramientas generales de visualizacion de los datos de entrada como rosa de oleaje, rosa de viento, funciones de densidad y distribucion, espectros de energia, graficos de dispersion, correlacion, etc. La funciones de visualización se incluyen en:

- clima_maritimo.graficas.plot_analisis

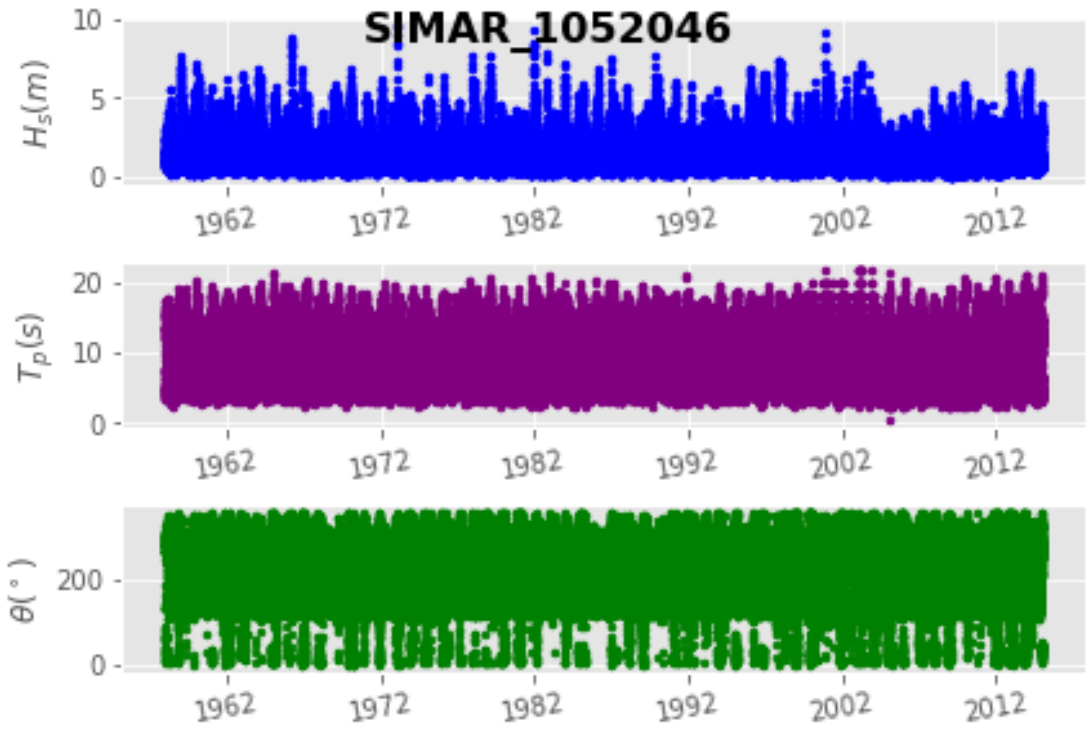
```
In [2]: # imports Anaconda
        from __future__ import division
        import os

        # imports ROM 1.1
        from entrada_datos import lectura, pretratamiento
        from clima_maritimo.graficas import plot_analisis

        cadiz_simar = 'SIMAR_1052046'
        dir_input = os.path.join(env.input_path, 'clima')
        dir_data = os.path.join(env.data_path, 'clima')
```

La función *lectura_simar* permite la obtención de cada una de las siguientes variables:

- Altura de ola significante H_s
- Periodo pico de oleaje T_p
- Dirección media de procedencia del olejae θ
- Velocidad media de viento V_w
- Dirección media de procedencia θ_w

Es posible representar las series temporales con la funcion *plot_series_temporales*.

```
In [3]: cadiz_hs_sim, cadiz_tp_sim, cadiz_dh_sim, cadiz_vv_sim, cadiz_dv_sim =
        lectura.lectura_simar(dir_input, cadiz_simar)

        df_oleaje = cadiz_hs_sim.join([cadiz_tp_sim, cadiz_dh_sim])
        lectura.plot_series_temporales(df_oleaje, cadiz_simar)
```



Es posible obtener un análisis de la calidad de los datos utilizando la función *analisis_calidad_series*. Esta información se guarda en un archivo de texto.

```
In [4]: inf_cadiz = open(os.path.join(dir_data, 'informe_cadiz.txt'), 'w')
        cadiz_hs_sim = pretratamiento.analisis_calidad_series(cadiz_hs_sim, inf_cadiz,
        cadiz_simar + '- hs')
```

```
La serie temporal tiene una longitud de 56.0 años y 359.0 días
La serie temporal tiene las siguientes cadencias:
157584 datos con una cadencia de 0 days 03:00:00
26422 datos con una cadencia de 0 days 01:00:00

La serie temporal tiene un total de 0.0 años y 340.0 días sin datos (huecos)
correspondiente a un 1.64% de la serie
La duración máxima de huecos para la variable es de:
30 days 18:00:00
```

Si la serie temporal presenta varias cadencias a lo largo de su longitud, es posible establecer la misma cadencia *cad_sim* la cual debe ser elegida por el usuario. La variable *nd_max_interp* define el número de datos máximos que se permite interpolar de huecos. En este caso, se elige la cadencia de 3 horas dado que es la presente en la mayor parte de datos de la serie y se define $nd\_max\_interp = 5$

```
In [5]: cad_sim  = 3*3600
        nd_max_interp = 5

        cadiz_hs_sim = pretratamiento.establecer_cadencia_series(cadiz_hs_sim, cad_sim,
        nd_max_interp, inf_cadiz)
```

```
Se ha interpolado a la cadencia de 0 days 03:00:00
```

Se guarda a archivo la serie obtenida:

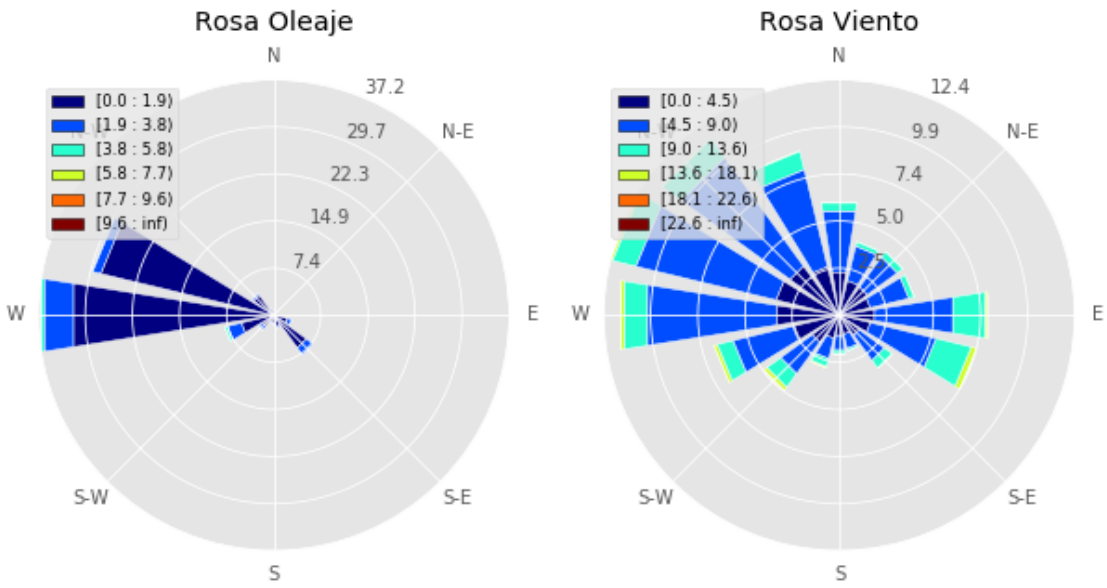
```
In [6]: cadiz_hs_sim.to_pickle(os.path.join(dir_data, 'cadiz_hs_simc.pkl'))
        inf_cadiz.close()
```

A continuación se presentan algunos ejemplos de las herramientas de visualización de datos presentes en el módulo *clima_maritimo.graficas.plot_analisis*

Rosa de oleaje y viento:

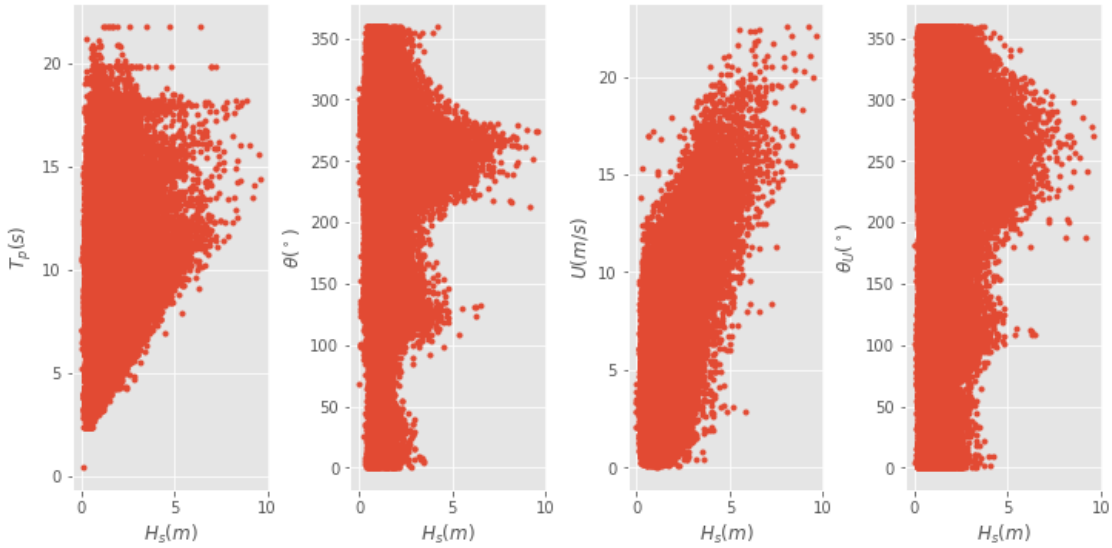
```
In [7]: df_sim = df_oleaje.join([cadiz_vv_sim, cadiz_dv_sim])
        plot_analisis.rosa_oleaje_viento(df_sim)
```

```
/Users/andrealira/anaconda/lib/python2.7/site-packages/numpy/lib/function_base.py:804:
RuntimeWarning: invalid value encountered in greater_equal
  not_smaller_than_edge = (sample[:, i] >= edges[i][-1])
```



Gráficos de dispersión:

```
In [8]:  plot_analisis.dispersion(df_sim, 'hs')
```



En el caso que existan muchos huecos de datos en la serie temporal, se proporcionan las herramientas para el relleno de los huecos mediante el método de los análogos. Esto se realiza en el apartado Pretratamiento de las series temporales de los agentes climáticos marítimos y atmosféricos.

cl_lectura_pretratamiento_02

April 27, 2018

0.0.1 Pretatamiento de las series temporales de los agentes climáticos marítimos y atmosféricos

Este ejemplo continuará a partir de los datos temporales guardados al final del apartado Lectura de las series temporales de los agentes climáticos marítimos y atmosféricos. En este ejemplo se presenta el procedimiento para el relleno de huecos de los datos del punto SIMAR 1052046, proporcionado por Puertos del Estado. Se utilizan datos de la Boya del Golfo de Cadiz para completar los datos del punto SIMAR 1052046.

Las funciones que se utilizan son:

- analogos.reconstruct_analogs.input_data_analogos_rellenoh
- analogos.reconstruct_analogs.reconstruct_analogs
- analogos.reconstruct_analogs.output_data_analogos
- analogos.reconstruct_analogs.plot_reconstruct_analogos

```
In [10]: # imports Anaconda
         from __future__ import division
         import pandas as pd
         import numpy as np
         import os

         # imports ROM 1.1
         from entrada_datos import lectura
         from analogos import reconstruct_analogs
         from clima_maritimo.clima_maritimo import pretratamiento_clima

         dir_data = os.path.join(env.data_path, 'clima')
         dir_input = os.path.join(env.input_path, 'clima')
         cadiz_hs_sim = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_simc.pkl'))
```

En primer lugar, es necesarion realizar la lectura de los datos de altura de ola significante correspondiente a la Boya del Golfo de Cádiz. Para ello, se utiliza la función *lectura_redcos* del módulo *lectura*.

```
In [11]: cadiz_boya = 'REDCOS_HIS_CostCadiz'
         cadiz_hs_boy, cadiz_tp_boy, cadiz_dh_boy = lectura.lectura_redcos(dir_input, cadiz_boya)
```

Los datos correspondientes a la serie temporal de altura de ola correspondientes al punto SIMAR 1052045, que debe ser reconstruida se denominan como "predictandos" (df_loc), mientras que los datos de la Boya de Cádiz corresponde al conjunto de datos "predictores" (df_pred).

```
In [12]: df_loc = cadiz_hs_sim
         df_pred = pd.merge(cadiz_hs_boy, cadiz_tp_boy, how='inner', left_index=True,
         right_index=True)
```

Las relaciones entre ambos conjuntos de datos durante el periodo de entrenamiento (*ttrain*) son empleadas para reconstruir la serie observada durante el periodo para el cual no existen observaciones (periodo a reconstruir *trec*). Con la función *input_data_analogos_rellenoh* se identifican los datos de ambas series temporales correspondientes a los periodos de entrenamiento y reconstrucción.

NOTA: En el caso que la serie de predictores no posea información en todos los instantes de tiempo de huecos de la serie de los predictandos, se muestra un aviso. El software permite continuar con el proceso aunque seguirán habiendo puntos en los que no sea posible realizar la reconstrucción.

```
In [13]: loc_ttrain, pred_ttrain, pred_trec =
         reconstruct_analogs.input_data_analogos_rellenoh(df_loc, df_pred)

El dataframe de datos de los predictores no contiene suficiente informacion. Revisar
los resultados de la reconstrucción
```

Se deben definir una serie de parámetros:

- *k*: Numero de analogos escogido para reconstruir cada instante.
- *atrend*: Tendencia temporal de los analogos (intervalo de comparacion en horas).
- *sw*: Ventana de busqueda para los candidatos a analogos en horas.
- *wp*: Vector de ponderacion de pesos para la metrica de analogos.

Tambien se debe indicar los siguientes campos: - *esc_pred*: Columnas de predictores correspondientes a variables escalares (empieza en 1). - *circ_pred*: Columnas de predictores correspondientes a variables circulares (empieza en 1).

```
In [14]: par = {'k': 25, 'a_trend': 12, 'sw': 0, 'wp': np.array([1, 1])}
         esc_pred = np.array([1, 2])
         circ_pred = np.array([])
```

La función *reconstruct_analogs* permite obtener los datos correspondiente a la serie de predictandos durante el periodo de reconstrucción. A continuación la función *output_data_analogs* permite la obtención de la serie final de predictandos.

NOTA: Dado que la función *reconstruct_analogs* conlleva una gran cantidad de tiempo de ejecución. Se han guardado los resultados de la serie reconstruida a archivo y no se ejecutan en este ejemplo.

```
In [ ]: # df_loc_trec = reconstruct_analogs.reconstruct_analogs(loc_ttrain, pred_ttrain,
        pred_trec, esc_pred, circ_pred, par)
        # cadiz_hs_sim_reconst = reconstruct_analogs.output_data_analogos(loc_ttrain,
        df_loc_trec)

        cadiz_hs_sim_reconst = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_reconst.pkl'))
        cadiz_hs_sim.update(cadiz_hs_sim_reconst)
```

Por último se incluye en esta función el uso de la función: - clima_maritimo.clima_maritimo.pretratamiento_clima.interp_distrib

Esta función permite distribuir los datos para evitar problemas en los ajustes por máxima verosimilitud usados en el análisis estadístico del clima. En esta función se debe definir el parámetro:

- umb_an: Umbral para descartar anos con menos porcentaje de numero de datos

```
In [ ]: umb_an = 0.8
        cadiz_hs_sim = pretratamiento_clima.interp_distrib(umb_an, cadiz_hs_sim)
        cadiz_hs_sim.to_pickle(os.path.join(dir_data, 'cadiz_hs_sim_pretrat.pkl'))
```

El proceso de transformación oceanográfica de estados de oleaje offshore al emplazamiento se realiza en el apartado Transformacion de los estados de las variables oceanograficas_al_emplazamiento.

cl_transformacion_emplazamiento_01

April 27, 2018

0.0.1 Transformacion de los estados de las variables oceanograficas al emplazamiento

Este ejemplo continuará a partir de los datos temporales guardados al final del apartado Pretratamiento de las series temporales de los agentes climáticos marítimos y atmosféricos para la variable altura de ola significante. Realizando el mismo procedimiento se obtienen las series temporales del periodo pico del oleaje y la dirección media de procedencia.

Se presenta un ejemplo de la metodologia general de transformacion oceanografica de estados de oleaje offshore del punto 1052046 de la Red SIMAR de Puertos del Estado ubicado en la costa de Cadiz.

Las funciones que se utilizan son:

- mda_rbf.normalizar.normalizar
- mda_rbf.mda.mda
- mda_rbf.mda.plot_data_casos
- entrada_datos.lectura.lectura_propagacion_oleaje
- mda_rbf.reconstruction_rbf.plot_minim_c
- mda_rbf.reconstruction_rbf.reconstruction_agente
- mda_rbf.mda.plot_series_reconstruidas

```
In [2]: # imports Anaconda
        from __future__ import division
        import matplotlib.dates as dates
        import pandas as pd
        import numpy as np
        import os

        # imports ROM 1.1
        from entrada_datos import lectura
        from mda_rbf import normalizar, mda, reconstruction_rbf
        from clima_maritimo.clima_maritimo import pretratamiento_clima

        dir_data = os.path.join(env.data_path, 'clima')

        cadiz_hs_sim = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim.pkl'))
        cadiz_tp_sim = pd.read_pickle(os.path.join(dir_data, 'cadiz_tp_sim.pkl'))
        cadiz_dh_sim = pd.read_pickle(os.path.join(dir_data, 'cadiz_dh_sim.pkl'))
        cadiz_vv_sim = pd.read_pickle(os.path.join(dir_data, 'cadiz_vv_sim.pkl'))
        cadiz_dv_sim = pd.read_pickle(os.path.join(dir_data, 'cadiz_dv_sim.pkl'))
```

NOTA: Se han leido los archivos originales (sin pretratamiento) dado que la propagación de los casos con modelo numérico se realizó con estas series.

La muestra original se divide en dos poblaciones: por un lado se considera la cola superior a partir del correspondiente umbral de altura de ola significante y, por otro, la cola inferior junto con el cuerpo central de la muestra. Se seleccionan los m casos representativos de la variabilidad del conjunto.

En este caso se define el umbral como el percentil 90 y $m = 500$.

```
In [3]: c_i = 0.0001
        c_f = 0.3
        n_c = 100

        U2 = cadiz_hs_sim.quantile(q=0.90).values[0]
        ncasos_tot = 500

        cadiz_sim = cadiz_hs_sim.join([cadiz_tp_sim, cadiz_dh_sim, cadiz_vv_sim, cadiz_dv_sim])
        cadiz_sim.dropna(inplace=True)

        n_sim = np.array(cadiz_sim.index.map(dates.date2num), dtype=np.float64)
```

Selección de casos representativos: Para el caso del cuerpo central y cola inferior, se asigna una proporción de 2/3 en relación a la cantidad de datos disponibles y el resto para la cola superior.

```
In [4]: cadiz_cen = cadiz_sim.loc[cadiz_sim['hs'] <= U2]
        ncasos_cen = int(np.floor(2 / 3 * ncasos_tot * (len(cadiz_cen) / len(cadiz_sim))))
```

En primer lugar se normalizan las variables de manera que la distancia entre dos puntos de la serie quede definida en el intervalo [0,1]. Esto se realiza mediante el uso de la función *normalizar* en la que se indica la naturaleza de la vairable (0 variable escalar, 1 variable circular).

```
In [5]: n_cen = np.array(cadiz_cen.index.map(dates.date2num), dtype=np.float64)
        hs_cen_norm = normalizar.normalizar(cadiz_cen.hs.values, 0)
        tp_cen_norm = normalizar.normalizar(cadiz_cen.tp.values, 0)
        dh_cen_norm = normalizar.normalizar(cadiz_cen.dh.values, 1)
        vv_cen_norm = normalizar.normalizar(cadiz_cen.vv.values, 0)
        dv_cen_norm = normalizar.normalizar(cadiz_cen.dv.values, 1)
        x_cen = np.vstack((hs_cen_norm, tp_cen_norm, dh_cen_norm, vv_cen_norm, dv_cen_norm)).T
```

La selección de los casos representativos se realiza mediante la función *mda*. Para el uso de esta función se deben especificar los siguientes parámetros:

- esc: Columnas de la matriz de datos correspondientes a variables escalares.
- circ: Columnas de la matriz de datos correspondientes a variables circulares.
- id_: Identificador de la variable que determina el primer punto del subconjunto.

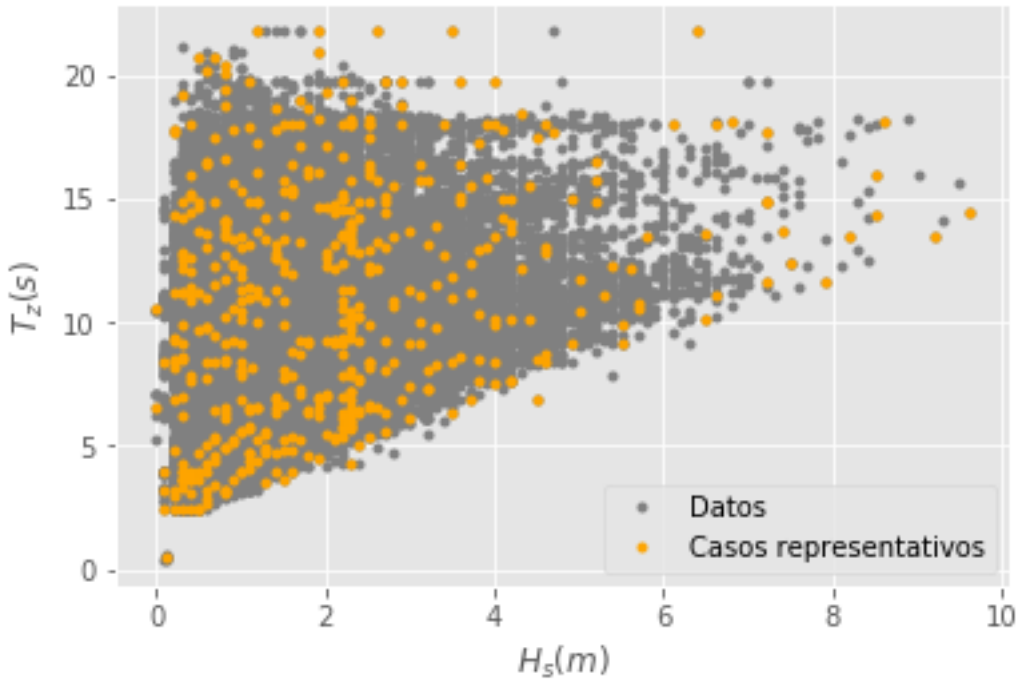
```
In [6]: esc = np.array([0, 1, 3])
        circ = np.array([2, 4])
        id_ = 0
        casos_cen = mda.mda(x_cen, esc, circ, ncasos_cen, id_)

        n_casos_cen = n_cen[casos_cen]
        var_casos_cen = x_cen[casos_cen, :]

        mask_n_cen = np.searchsorted(n_sim, n_casos_cen)
        cadiz_cen_casos = cadiz_sim.iloc[mask_n_cen, :]
```

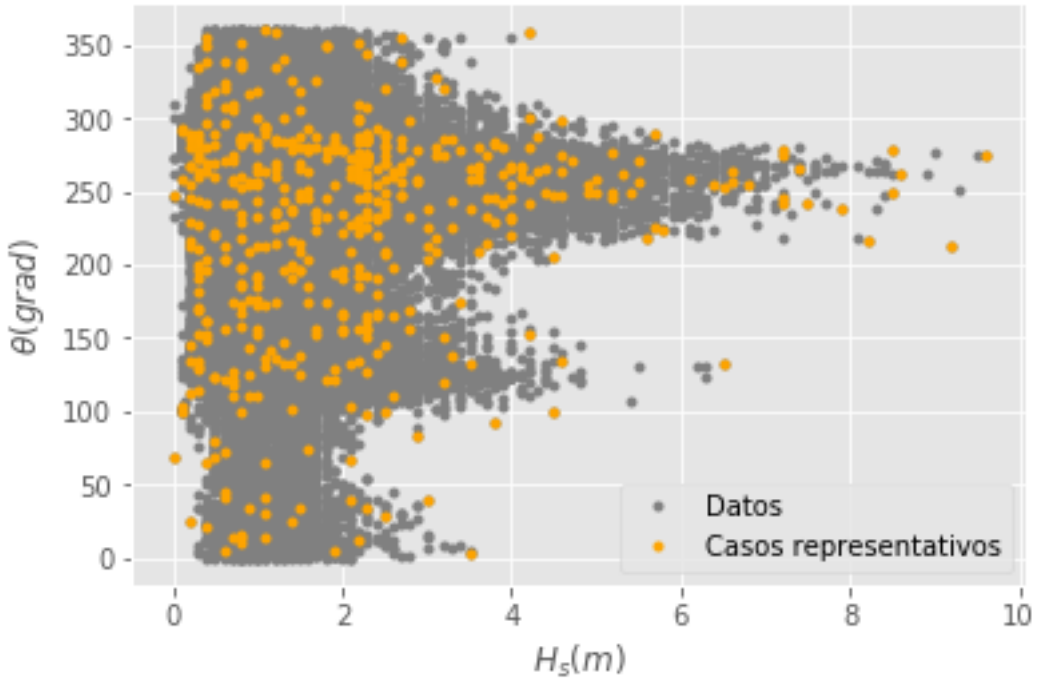
En el caso de la cola superior se realiza el mismo procedimiento:

```
In [7]: cadiz_sup = cadiz_sim.loc[cadiz_sim['hs'] > U2]
        ncasos_sup = ncasos_tot - ncasos_cen

        n_sup = np.array(cadiz_sup.index.map(dates.date2num), dtype=np.float64)
        hs_sup_norm = normalizar.normalizar(cadiz_sup.hs.values, 0)
        tp_sup_norm = normalizar.normalizar(cadiz_sup.tp.values, 0)
        dh_sup_norm = normalizar.normalizar(cadiz_sup.dh.values, 1)
        vv_sup_norm = normalizar.normalizar(cadiz_sup.vv.values, 0)
        dv_sup_norm = normalizar.normalizar(cadiz_sup.dv.values, 1)

        x_sup = np.vstack((hs_sup_norm, tp_sup_norm, dh_sup_norm, vv_sup_norm, dv_sup_norm)).T
        # variables normalizadas
        casos_sup = mda.mda(x_sup, esc, circ, ncasos_sup, id_)

        n_casos_sup = n_sup[casos_sup]
        var_casos_sup = x_sup[casos_sup, :]

        mask_n_sup = np.searchsorted(n_sim, n_casos_sup)
        cadiz_sup_casos = cadiz_sim.iloc[mask_n_sup, :]
```

La función *plot_data_casos* permite la visualización de los casos seleccionados

```
In [8]: cadiz_sim_casos = pd.concat([cadiz_cen_casos, cadiz_sup_casos]).sort_index()
        mda.plot_data_casos(cadiz_sim.hs.values, cadiz_sim.tp.values, cadiz_sim_casos.hs.values,
        cadiz_sim_casos.tp.values, '$H_s (m)$', '$T_z (s)$')
        mda.plot_data_casos(cadiz_sim.hs.values, cadiz_sim.dh.values, cadiz_sim_casos.hs.values,
        cadiz_sim_casos.dh.values, '$H_s (m)$', r'$\theta (grad)$')
```





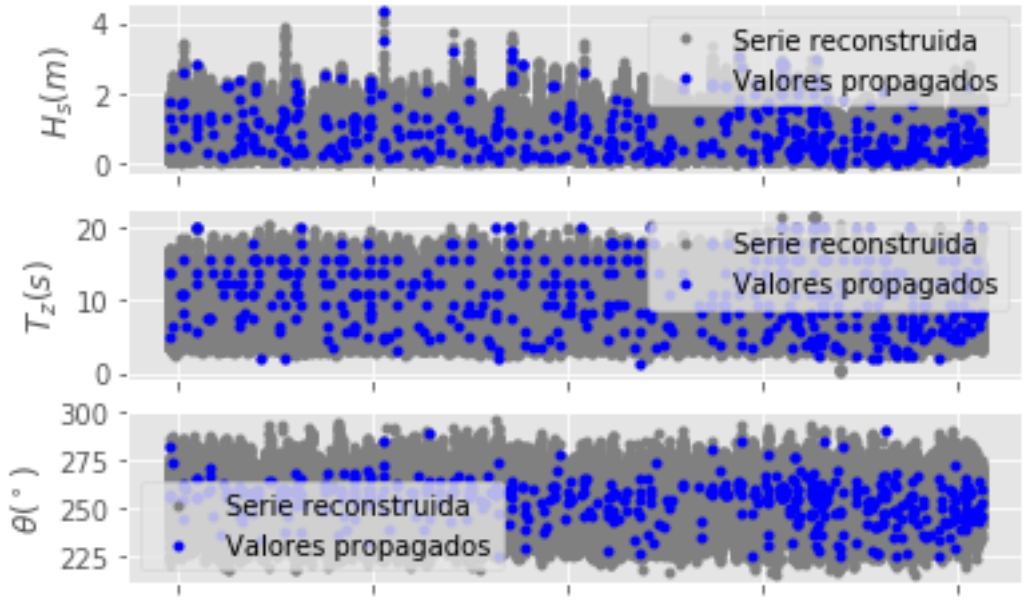
Lectura datos propagados: El subconjunto seleccionado ha sido propagado mediante un modelo numerico. Dado que esto debe ser un paso separado, aqui solamente se cargan los resultados de la propagación.

```
In [9]: # COLA INFERIOR Y CUERPO CENTRAL
        cadiz_sim_prop_cen, cadiz_hs_sim_prop_cen, cadiz_tp_sim_prop_cen, cadiz_dh_sim_prop_cen
        = lectura.lectura_propagacion_oleaje(dir_data, 'cadiz_prop_infcen.txt')

        # se eliminan los puntos con datos erroneos
        posincc = np.where((cadiz_sim_prop_cen == -999).any(1))[0]
        var_casos_cen = np.delete(var_casos_cen, posincc, 0)
        n_casos_cen = np.delete(n_casos_cen, posincc, 0)
        cadiz_sim_prop_cen = cadiz_sim_prop_cen.loc[(cadiz_sim_prop_cen != -999).all(1)]

        casos_cen_prop = cadiz_sim_prop_cen.values.T
        serie_casos_prop_cen = np.vstack((n_casos_cen, casos_cen_prop)).T

        # COLA SUPERIOR
        [cadiz_sim_prop_sup, cadiz_hs_sim_prop_sup, cadiz_tp_sim_prop_sup,
        cadiz_dh_sim_prop_sup] = lectura.lectura_propagacion_oleaje(dir_data,
        'cadiz_prop_sup.txt')

        posincs = np.where((cadiz_sim_prop_sup == -999).any(1))[0]
        var_casos_sup = np.delete(var_casos_sup, posincs, 0)
        n_casos_sup = np.delete(n_casos_sup, posincs, 0)
        cadiz_sim_prop_sup = cadiz_sim_prop_sup.loc[(cadiz_sim_prop_sup != -999).all(1)]

        casos_sup_prop = cadiz_sim_prop_sup.values.T
        serie_casos_prop_sup = np.vstack((n_casos_sup, casos_sup_prop)).T
```

Reconstrucción de serie en el emplazamiento La función *reconstruction_agente* permite la reconstrucción de la serie temporal en el emplazamiento. Se deben imponer los límites y el número de datos para la optimización del algoritmo de iterpolación (c_i, c_f, n_c).

NOTA: Dado que la función *reconstruction_agente* puede tardar un poco en ejecutar, se han guardado los resultados de la serie reconstruida a archivo y no se ejecutan en este ejemplo.

```
In [10]: nvars = 3

         c_i = 0.0001
         c_f = 0.3
         n_c = 100

         # COLA INFERIOR Y CUERPO CENTRAL
         # data_cen, vc_vars_cen, e_vars_cen, posmin_vars_cen =
         reconstruction_rbf.reconstruction_agente(x_cen, var_casos_cen, n_cen, casos_cen_prop,
         nvars, c_i, c_f, n_c, esc, circ)
         data_cen = np.loadtxt(os.path.join(dir_data, 'serie_recons_cen'))

         # COLA SUPERIOR
         # data_sup, vc_vars_sup, e_vars_sup, posmin_vars_sup =
         reconstruction_rbf.reconstruction_agente(x_sup, var_casos_sup, n_sup, casos_sup_prop,
         nvars, c_i, c_f, n_c, esc, circ)
         data_sup = np.loadtxt(os.path.join(dir_data, 'serie_recons_sup'))

         serie_casos_prop = np.vstack((serie_casos_prop_cen, serie_casos_prop_sup))
         serie_reconstruida = np.vstack((data_cen.T, data_sup.T))
         mda.plot_series_reconstruidas(serie_reconstruida, serie_casos_prop, 3)

         serie_reconstruida_ord = serie_reconstruida[serie_reconstruida[:,0].argsort()]
         cadiz_hs_sim_emp = pd.DataFrame(serie_reconstruida_ord[:, 1], index=cadiz_sim.index,
         columns=['hs'])
         cadiz_tp_sim_emp = pd.DataFrame(serie_reconstruida_ord[:, 2], index=cadiz_sim.index,
         columns=['tp'])
         cadiz_dh_sim_emp = pd.DataFrame(serie_reconstruida_ord[:, 3], index=cadiz_sim.index,
         columns=['dh'])
```



NOTA: Se aplica la función *interp_distrib* presentada en el apartado Pretratamiento de las series temporales de los agentes climáticos marítimos y atmosféricos:

```
In [11]: cadiz_hs_sim_emp = pretratamiento_clima.interp_distrib(0.8, cadiz_hs_sim_emp)
         cadiz_tp_sim_emp = pretratamiento_clima.interp_distrib(0.8, cadiz_tp_sim_emp)
         cadiz_dh_sim_emp = pretratamiento_clima.interp_distrib(0.8, cadiz_dh_sim_emp)
```

Se guardan los datos a archivo:

```
In [12]: cadiz_hs_sim_emp.to_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
         cadiz_tp_sim_emp.to_pickle(os.path.join(dir_data, 'cadiz_tp_sim_emp.pkl'))
         cadiz_dh_sim_emp.to_pickle(os.path.join(dir_data, 'cadiz_dh_sim_emp.pkl'))
```

Con las series temporales de las variables de interés en el emplazamiento, se puede proceder al cálculo del regimén extremal de la variable predominante que se presenta en el apartado Distribucion de extremos de la variable predominante.

cl_regimen_extremal_01

April 27, 2018

0.0.1 Distribucion de extremos de la variable predominante

Este ejemplo continuará a partir de los datos temporales guardados al final del apartado Transformacion de los estados de las variables oceanograficas al emplazamiento

En este ejemplo se presenta el analisis del regimen extremal de la altura de ola significante H_s utilizando los datos del punto SIMAR 1052046 (Cadiz, Espana) proporcionados por Puertos del Estado. En este caso se utizará el análisis de extremos mediante picos sobre umbral (POT).

Este ejemplo se dividirá en 4 partes: 1. Obtener la serie temporal de eventos extremos 2. Análisis del umbral para el analisis de picos sobre umbral (POT) 3. Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno T_R

Continuar con Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos.

cl_regimen_extremal_02

April 27, 2018

0.0.1 Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos

Este ejemplo continuará a partir de los datos temporales guardados al final del apartado Transformacion de los estados de las variables oceanograficas al emplazamiento. Se utiliza la serie temporal de altura de ola significante H_s. Se obtienen todos los eventos de extremos utilizando una ventana temporal móvil de longitud N_i días para garantizar su independencia. La longitud de dicha ventana es definida por el usuario.

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_dur
- clima_maritimo.graficas.plot_extremal.plot_serie_picos

```
In [8]: # imports Anaconda
        from __future__ import division
        import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        # imports ROM 1.1
        from clima_maritimo.clima_maritimo.fdist import utils
        from clima_maritimo.graficas import plot_extremal

        dir_data = os.path.join(env.data_path, 'clima')
        df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
```

Entradas de usuario:

- Número de dias para garantizar que eventos independientes N_i

```
In [9]: Ni = 5
```

NOTA: La duración de la ventana se hace según el número de datos que corresponden a N_i dias por lo que es necesario que la serie tenga una cadencia homogénea. En caso contrario, se eligirá la más representativa.

```
In [10]: df['td'] = df.index.to_series() - df.index.to_series().shift()
         cadencias = df['td'].value_counts()
         cadencias = cadencias[cadencias > 0.01 * len(df)]

         if len(cadencias)>1:
             print('La serie temporal no es homogénea. Presenta las siguientes cadencias:')
             str_4 = ''
             for j, cad in enumerate(cadencias):
```

```
            str_4 += str(cad) + ' datos con una cadencia de ' + str(cadencias.index[j]) +
'\n'
        print(str_4)
        print('Para el cálculo se utilizará la cadencia de ' + str(cadencias.argmax()))
        cad_df = cadencias.argmax().total_seconds()
    else:
        print('Para el cálculo se utilizará la cadencia de ' + str(cadencias.index[0]))
        cad_df = cadencias.index[0].total_seconds()

    df.drop('td', axis=1, inplace=True)
    no_datos_dia = (24 * 3600) / cad_df
    dur_picos = Ni * int(no_datos_dia)
    nanios = len(np.unique(df.index.year))
```

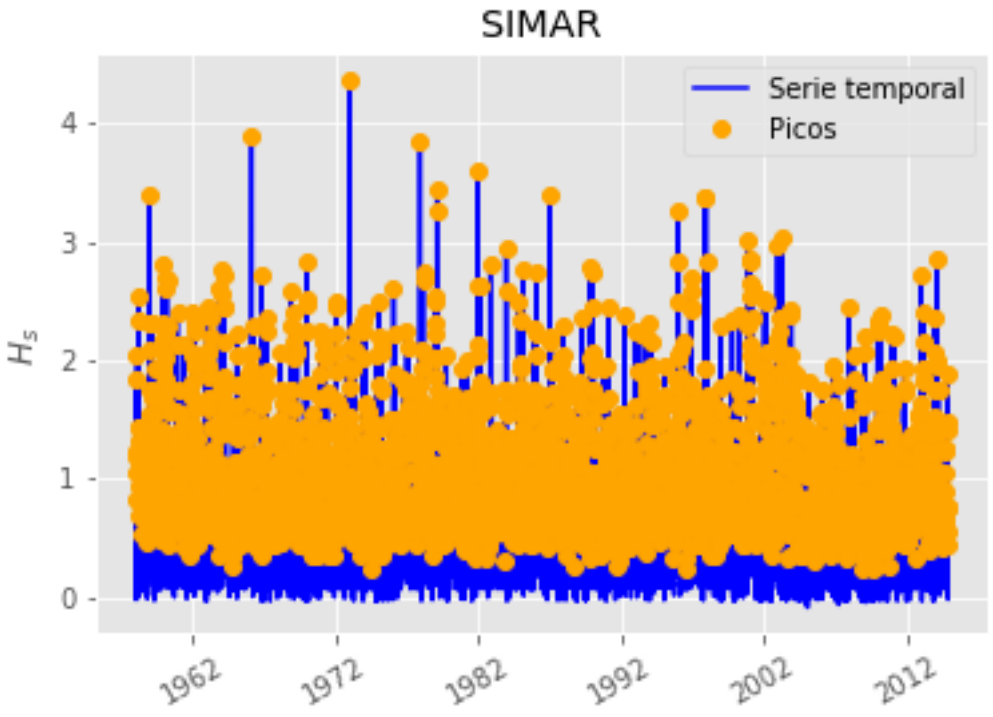
```
Para el cálculo se utilizará la cadencia de 0 days 03:00:00
```

Utilizando la funcion *picos_dur* se obtienen todos los eventos de extremos presentes en la serie temporal. Se guarda el dataframe de picos.

```
In [11]: pos_picos = utils.picos_dur(df, dur_picos)
         df_picos = df.iloc[pos_picos, :]
         df_picos.to_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
```

Se visualizan los resultados utilizando la función *plot_serie_picos*:

```
In [12]: plot_extremal.plot_serie_picos(df, df_picos, '$H_s$', 'SIMAR')
         plt.show()
```



Continuar con Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT).

cl_regimen_extremal_03

April 27, 2018

0.0.1 Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT)

Este ejemplo continuará a partir de los datos temporales de altura de ola H_s guardados al final del apartado Transformacion de los estados de las variables oceanograficas al emplazamiento. y la serie de picos independientes obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos. Se sigue la metodología de Solari et al. (2017) para la elección del umbral adecuado en el análisis del regimén extremal POT (picos sobre umbral).

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_umb
- clima_maritimo.graficas.plot_extremal.plot_serie_picos_umbral
- clima_maritimo.clima_maritimo.fdist.regimen_extremal.analisis_pot_lmon

```
In [2]: # imports Anaconda
        from __future__ import division
        import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        # imports ROM 1.1
        from clima_maritimo.clima_maritimo.fdist import utils, regimen_extremal
        from clima_maritimo.graficas import plot_extremal

        dir_data = os.path.join(env.data_path, 'clima')
        df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
        df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
```

Entradas de usuario:

- Umbral mínimo a considerar umb_ex. En este ejemplo se considera el valor correspondiente al percentil 90.
- Nivel de significancia para las bandas de confianza $alpha$. (0.1 para tener nivel significancia al 95%)

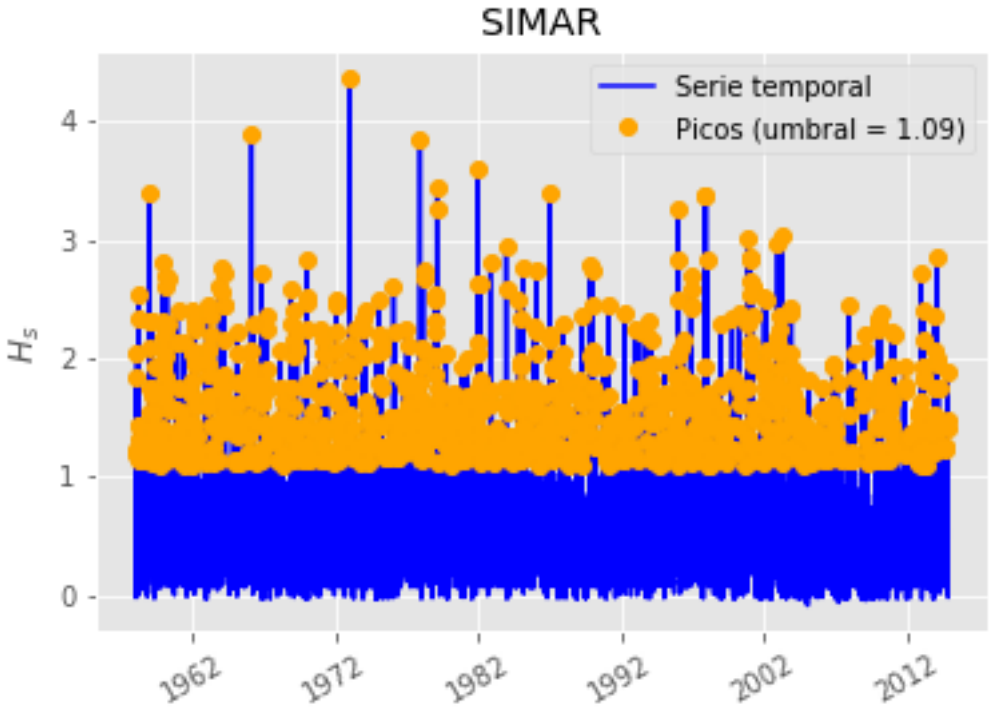
```
In [3]: umb_ex = [df.quantile(q=0.90).values[0]]
        alpha = 0.1
```

De todos los picos independientes en la serie temporal se consideran solo aquellos que superan el umbral definido y se guarda el dataframe obtenido.

```
In [4]: nanios = len(np.unique(df.index.year))
        pos_picos_umb = utils.picos_umb(df_picos.values, umb_ex)

        # Guardar a archivo
        df_picos_umb_i = df_picos.iloc[pos_picos_umb[0], :]
        df_picos_umb_i.to_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos_umbp90.pkl'))
```

Utilizando la funcion *plot_serie_picos_umbral* es posible visualizar los picos sobre umbral:

```
In [5]: plot_extremal.plot_serie_picos_umbral(df, df_picos_umb_i, '$H_s$', umb_ex[0], 'SIMAR')
        plt.show()
```



A partir de la serie de picos obstenida y una serie de umbrales, a partir del definido en el apartado anterior, se estiman los tres parametros que definen la distribucion generalizada de Pareto segun el valor del umbral y se estima la distribucion del estadistico A_2^R y el correspondiente $p-valor$. Asimismo, se calculan los valores medios e intervalos de confianza de H_s para distintos periodos de retorno T_R. Los resultado se visualizan con la función *plot_pot_lmom*. NOTA: Este paso puede tardar mucho tiempo ($O(h)$). Se muestra en pantalla un contador del avance.

```
In [6]: au2pv_lmom, au2_lmom_i, valmed_lmom_i, limsup_i, liminf_i, umb_i, treval_i, nanios_i = \
            regimen_extremal.analisis_pot_lmon(df_picos_umb_i, alpha, nanios)

        # Guardar a archivo
        out_pot_lmon_i = au2pv_lmom, au2_lmom_i, valmed_lmom_i, limsup_i, liminf_i, umb_i,
        treval_i, nanios_i
        np.save(os.path.join(dir_data, 'out_pot_ini_hs_sim_p90.npy'), out_pot_lmon_i)
        # au2pv_lmom, au2_lmom_i, valmed_lmom_i, limsup_i, liminf_i, umb_i, treval_i, nanios_i =
        np.load(os.path.join(dir_data, 'out_pot_ini_hs_sim_p90.npy'))
```

```
/Users/andrealira/CloudStation/ROM__1_1/Herramienta/clima_maritimo/clima_maritimo/fdis
t/regimen_extremal.py:218: VisibleDeprecationWarning: boolean index did not match
```

```
indexed array along dimension 0; dimension is 1019 but corresponding boolean dimension
is 1018
  umb = umb[np.diff(umb) > 0]


1 de 1018
2 de 1018
3 de 1018
4 de 1018
5 de 1018
6 de 1018
7 de 1018
8 de 1018
9 de 1018
10 de 1018
11 de 1018
12 de 1018
13 de 1018
14 de 1018
15 de 1018
16 de 1018
17 de 1018
18 de 1018
19 de 1018
20 de 1018
21 de 1018
22 de 1018
23 de 1018
24 de 1018
25 de 1018
26 de 1018
27 de 1018
28 de 1018
29 de 1018
30 de 1018
31 de 1018
32 de 1018
33 de 1018
34 de 1018
35 de 1018
36 de 1018
37 de 1018
38 de 1018
39 de 1018
40 de 1018
41 de 1018
42 de 1018
43 de 1018
44 de 1018
45 de 1018
46 de 1018
47 de 1018
48 de 1018
49 de 1018
50 de 1018
51 de 1018
52 de 1018
53 de 1018
54 de 1018
```

55 de 1018
56 de 1018
57 de 1018
58 de 1018
59 de 1018
60 de 1018
61 de 1018
62 de 1018
63 de 1018
64 de 1018
65 de 1018
66 de 1018
67 de 1018
68 de 1018
69 de 1018
70 de 1018
71 de 1018
72 de 1018
73 de 1018
74 de 1018
75 de 1018
76 de 1018
77 de 1018
78 de 1018
79 de 1018
80 de 1018
81 de 1018
82 de 1018
83 de 1018
84 de 1018
85 de 1018
86 de 1018
87 de 1018
88 de 1018
89 de 1018
90 de 1018
91 de 1018
92 de 1018
93 de 1018
94 de 1018
95 de 1018
96 de 1018
97 de 1018
98 de 1018
99 de 1018
100 de 1018
101 de 1018
102 de 1018
103 de 1018
104 de 1018
105 de 1018
106 de 1018
107 de 1018
108 de 1018
109 de 1018
110 de 1018
111 de 1018
112 de 1018
113 de 1018

114 de 1018
115 de 1018
116 de 1018
117 de 1018
118 de 1018
119 de 1018
120 de 1018
121 de 1018
122 de 1018
123 de 1018
124 de 1018
125 de 1018
126 de 1018
127 de 1018
128 de 1018
129 de 1018
130 de 1018
131 de 1018
132 de 1018
133 de 1018
134 de 1018
135 de 1018
136 de 1018
137 de 1018
138 de 1018
139 de 1018
140 de 1018
141 de 1018
142 de 1018
143 de 1018
144 de 1018
145 de 1018
146 de 1018
147 de 1018
148 de 1018
149 de 1018
150 de 1018
151 de 1018
152 de 1018
153 de 1018
154 de 1018
155 de 1018
156 de 1018
157 de 1018
158 de 1018
159 de 1018
160 de 1018
161 de 1018
162 de 1018
163 de 1018
164 de 1018
165 de 1018
166 de 1018
167 de 1018
168 de 1018
169 de 1018
170 de 1018
171 de 1018
172 de 1018

173 de 1018
174 de 1018
175 de 1018
176 de 1018
177 de 1018
178 de 1018
179 de 1018
180 de 1018
181 de 1018
182 de 1018
183 de 1018
184 de 1018
185 de 1018
186 de 1018
187 de 1018
188 de 1018
189 de 1018
190 de 1018
191 de 1018
192 de 1018
193 de 1018
194 de 1018
195 de 1018
196 de 1018
197 de 1018
198 de 1018
199 de 1018
200 de 1018
201 de 1018
202 de 1018
203 de 1018
204 de 1018
205 de 1018
206 de 1018
207 de 1018
208 de 1018
209 de 1018
210 de 1018
211 de 1018
212 de 1018
213 de 1018
214 de 1018
215 de 1018
216 de 1018
217 de 1018
218 de 1018
219 de 1018
220 de 1018
221 de 1018
222 de 1018
223 de 1018
224 de 1018
225 de 1018
226 de 1018
227 de 1018
228 de 1018
229 de 1018
230 de 1018
231 de 1018

232 de 1018
233 de 1018
234 de 1018
235 de 1018
236 de 1018
237 de 1018
238 de 1018
239 de 1018
240 de 1018
241 de 1018
242 de 1018
243 de 1018
244 de 1018
245 de 1018
246 de 1018
247 de 1018
248 de 1018
249 de 1018
250 de 1018
251 de 1018
252 de 1018
253 de 1018
254 de 1018
255 de 1018
256 de 1018
257 de 1018
258 de 1018
259 de 1018
260 de 1018
261 de 1018
262 de 1018
263 de 1018
264 de 1018
265 de 1018
266 de 1018
267 de 1018
268 de 1018
269 de 1018
270 de 1018
271 de 1018
272 de 1018
273 de 1018
274 de 1018
275 de 1018
276 de 1018
277 de 1018
278 de 1018
279 de 1018
280 de 1018
281 de 1018
282 de 1018
283 de 1018
284 de 1018
285 de 1018
286 de 1018
287 de 1018
288 de 1018
289 de 1018
290 de 1018

291 de 1018
292 de 1018
293 de 1018
294 de 1018
295 de 1018
296 de 1018
297 de 1018
298 de 1018
299 de 1018
300 de 1018
301 de 1018
302 de 1018
303 de 1018
304 de 1018
305 de 1018
306 de 1018
307 de 1018
308 de 1018
309 de 1018
310 de 1018
311 de 1018
312 de 1018
313 de 1018
314 de 1018
315 de 1018
316 de 1018
317 de 1018
318 de 1018
319 de 1018
320 de 1018
321 de 1018
322 de 1018
323 de 1018
324 de 1018
325 de 1018
326 de 1018
327 de 1018
328 de 1018
329 de 1018
330 de 1018
331 de 1018
332 de 1018
333 de 1018
334 de 1018
335 de 1018
336 de 1018
337 de 1018
338 de 1018
339 de 1018
340 de 1018
341 de 1018
342 de 1018
343 de 1018
344 de 1018
345 de 1018
346 de 1018
347 de 1018
348 de 1018
349 de 1018

350 de 1018
351 de 1018
352 de 1018
353 de 1018
354 de 1018
355 de 1018
356 de 1018
357 de 1018
358 de 1018
359 de 1018
360 de 1018
361 de 1018
362 de 1018
363 de 1018
364 de 1018
365 de 1018
366 de 1018
367 de 1018
368 de 1018
369 de 1018
370 de 1018
371 de 1018
372 de 1018
373 de 1018
374 de 1018
375 de 1018
376 de 1018
377 de 1018
378 de 1018
379 de 1018
380 de 1018
381 de 1018
382 de 1018
383 de 1018
384 de 1018
385 de 1018
386 de 1018
387 de 1018
388 de 1018
389 de 1018
390 de 1018
391 de 1018
392 de 1018
393 de 1018
394 de 1018
395 de 1018
396 de 1018
397 de 1018
398 de 1018
399 de 1018
400 de 1018
401 de 1018
402 de 1018
403 de 1018
404 de 1018
405 de 1018
406 de 1018
407 de 1018
408 de 1018

409 de 1018
410 de 1018
411 de 1018
412 de 1018
413 de 1018
414 de 1018
415 de 1018
416 de 1018
417 de 1018
418 de 1018
419 de 1018
420 de 1018
421 de 1018
422 de 1018
423 de 1018
424 de 1018
425 de 1018
426 de 1018
427 de 1018
428 de 1018
429 de 1018
430 de 1018
431 de 1018
432 de 1018
433 de 1018
434 de 1018
435 de 1018
436 de 1018
437 de 1018
438 de 1018
439 de 1018
440 de 1018
441 de 1018
442 de 1018
443 de 1018
444 de 1018
445 de 1018
446 de 1018
447 de 1018
448 de 1018
449 de 1018
450 de 1018
451 de 1018
452 de 1018
453 de 1018
454 de 1018
455 de 1018
456 de 1018
457 de 1018
458 de 1018
459 de 1018
460 de 1018
461 de 1018
462 de 1018
463 de 1018
464 de 1018
465 de 1018
466 de 1018
467 de 1018

468 de 1018
469 de 1018
470 de 1018
471 de 1018
472 de 1018
473 de 1018
474 de 1018
475 de 1018
476 de 1018
477 de 1018
478 de 1018
479 de 1018
480 de 1018
481 de 1018
482 de 1018
483 de 1018
484 de 1018
485 de 1018
486 de 1018
487 de 1018
488 de 1018
489 de 1018
490 de 1018
491 de 1018
492 de 1018
493 de 1018
494 de 1018
495 de 1018
496 de 1018
497 de 1018
498 de 1018
499 de 1018
500 de 1018
501 de 1018
502 de 1018
503 de 1018
504 de 1018
505 de 1018
506 de 1018
507 de 1018
508 de 1018
509 de 1018
510 de 1018
511 de 1018
512 de 1018
513 de 1018
514 de 1018
515 de 1018
516 de 1018
517 de 1018
518 de 1018
519 de 1018
520 de 1018
521 de 1018
522 de 1018
523 de 1018
524 de 1018
525 de 1018
526 de 1018

527 de 1018
528 de 1018
529 de 1018
530 de 1018
531 de 1018
532 de 1018
533 de 1018
534 de 1018
535 de 1018
536 de 1018
537 de 1018
538 de 1018
539 de 1018
540 de 1018
541 de 1018
542 de 1018
543 de 1018
544 de 1018
545 de 1018
546 de 1018
547 de 1018
548 de 1018
549 de 1018
550 de 1018
551 de 1018
552 de 1018
553 de 1018
554 de 1018
555 de 1018
556 de 1018
557 de 1018
558 de 1018
559 de 1018
560 de 1018
561 de 1018
562 de 1018
563 de 1018
564 de 1018
565 de 1018
566 de 1018
567 de 1018
568 de 1018
569 de 1018
570 de 1018
571 de 1018
572 de 1018
573 de 1018
574 de 1018
575 de 1018
576 de 1018
577 de 1018
578 de 1018
579 de 1018
580 de 1018
581 de 1018
582 de 1018
583 de 1018
584 de 1018
585 de 1018

586 de 1018
587 de 1018
588 de 1018
589 de 1018
590 de 1018
591 de 1018
592 de 1018
593 de 1018
594 de 1018
595 de 1018
596 de 1018
597 de 1018
598 de 1018
599 de 1018
600 de 1018
601 de 1018
602 de 1018
603 de 1018
604 de 1018
605 de 1018
606 de 1018
607 de 1018
608 de 1018
609 de 1018
610 de 1018
611 de 1018
612 de 1018
613 de 1018
614 de 1018
615 de 1018
616 de 1018
617 de 1018
618 de 1018
619 de 1018
620 de 1018
621 de 1018
622 de 1018
623 de 1018
624 de 1018
625 de 1018
626 de 1018
627 de 1018
628 de 1018
629 de 1018
630 de 1018
631 de 1018
632 de 1018
633 de 1018
634 de 1018
635 de 1018
636 de 1018
637 de 1018
638 de 1018
639 de 1018
640 de 1018
641 de 1018
642 de 1018
643 de 1018
644 de 1018

645 de 1018
646 de 1018
647 de 1018
648 de 1018
649 de 1018
650 de 1018
651 de 1018
652 de 1018
653 de 1018
654 de 1018
655 de 1018
656 de 1018
657 de 1018
658 de 1018
659 de 1018
660 de 1018
661 de 1018
662 de 1018
663 de 1018
664 de 1018
665 de 1018
666 de 1018
667 de 1018
668 de 1018
669 de 1018
670 de 1018
671 de 1018
672 de 1018
673 de 1018
674 de 1018
675 de 1018
676 de 1018
677 de 1018
678 de 1018
679 de 1018
680 de 1018
681 de 1018
682 de 1018
683 de 1018
684 de 1018
685 de 1018
686 de 1018
687 de 1018
688 de 1018
689 de 1018
690 de 1018
691 de 1018
692 de 1018
693 de 1018
694 de 1018
695 de 1018
696 de 1018
697 de 1018
698 de 1018
699 de 1018
700 de 1018
701 de 1018
702 de 1018
703 de 1018

704 de 1018
705 de 1018
706 de 1018
707 de 1018
708 de 1018
709 de 1018
710 de 1018
711 de 1018
712 de 1018
713 de 1018
714 de 1018
715 de 1018
716 de 1018
717 de 1018
718 de 1018
719 de 1018
720 de 1018
721 de 1018
722 de 1018
723 de 1018
724 de 1018
725 de 1018
726 de 1018
727 de 1018
728 de 1018
729 de 1018
730 de 1018
731 de 1018
732 de 1018
733 de 1018
734 de 1018
735 de 1018
736 de 1018
737 de 1018
738 de 1018
739 de 1018
740 de 1018
741 de 1018
742 de 1018
743 de 1018
744 de 1018
745 de 1018
746 de 1018
747 de 1018
748 de 1018
749 de 1018
750 de 1018
751 de 1018
752 de 1018
753 de 1018
754 de 1018
755 de 1018
756 de 1018
757 de 1018
758 de 1018
759 de 1018
760 de 1018
761 de 1018
762 de 1018

763 de 1018
764 de 1018
765 de 1018
766 de 1018
767 de 1018
768 de 1018
769 de 1018
770 de 1018
771 de 1018
772 de 1018
773 de 1018
774 de 1018
775 de 1018
776 de 1018
777 de 1018
778 de 1018
779 de 1018
780 de 1018
781 de 1018
782 de 1018
783 de 1018
784 de 1018
785 de 1018
786 de 1018
787 de 1018
788 de 1018
789 de 1018
790 de 1018
791 de 1018
792 de 1018
793 de 1018
794 de 1018
795 de 1018
796 de 1018
797 de 1018
798 de 1018
799 de 1018
800 de 1018
801 de 1018
802 de 1018
803 de 1018
804 de 1018
805 de 1018
806 de 1018
807 de 1018
808 de 1018
809 de 1018
810 de 1018
811 de 1018
812 de 1018
813 de 1018
814 de 1018
815 de 1018
816 de 1018
817 de 1018
818 de 1018
819 de 1018
820 de 1018
821 de 1018

822 de 1018
823 de 1018
824 de 1018
825 de 1018
826 de 1018
827 de 1018
828 de 1018
829 de 1018
830 de 1018
831 de 1018
832 de 1018
833 de 1018
834 de 1018
835 de 1018
836 de 1018
837 de 1018
838 de 1018
839 de 1018
840 de 1018
841 de 1018
842 de 1018
843 de 1018
844 de 1018
845 de 1018
846 de 1018
847 de 1018
848 de 1018
849 de 1018
850 de 1018
851 de 1018
852 de 1018
853 de 1018
854 de 1018
855 de 1018
856 de 1018
857 de 1018
858 de 1018
859 de 1018
860 de 1018
861 de 1018
862 de 1018
863 de 1018
864 de 1018
865 de 1018
866 de 1018
867 de 1018
868 de 1018
869 de 1018
870 de 1018
871 de 1018
872 de 1018
873 de 1018
874 de 1018
875 de 1018
876 de 1018
877 de 1018
878 de 1018
879 de 1018
880 de 1018

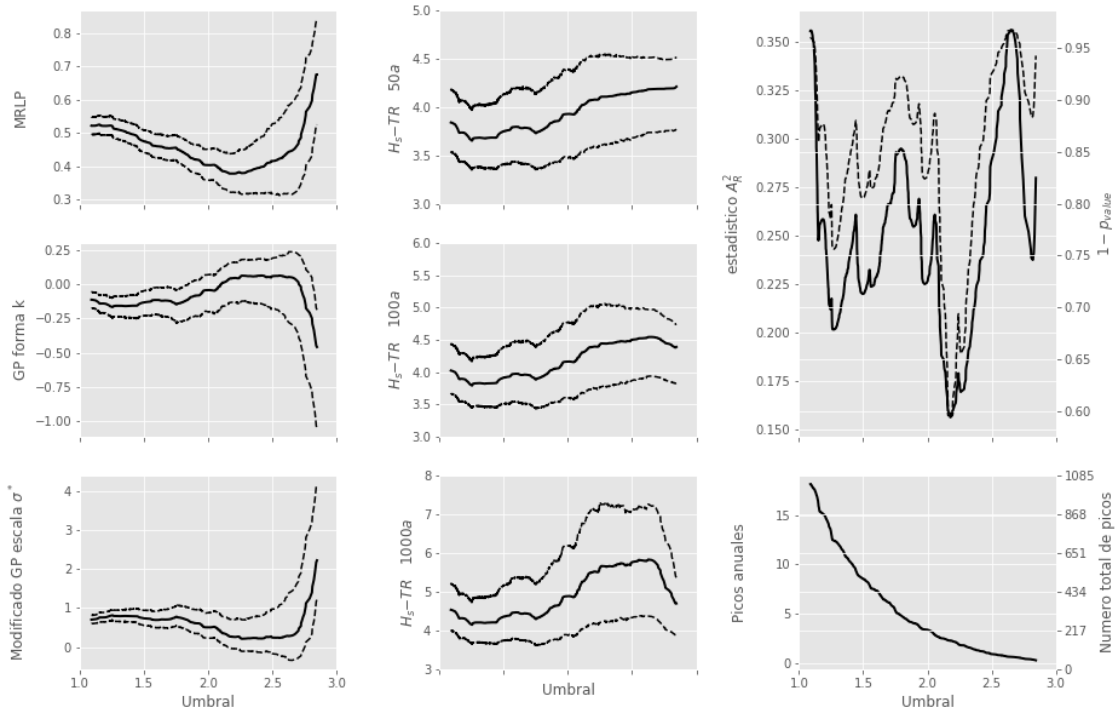
881 de 1018
882 de 1018
883 de 1018
884 de 1018
885 de 1018
886 de 1018
887 de 1018
888 de 1018
889 de 1018
890 de 1018
891 de 1018
892 de 1018
893 de 1018
894 de 1018
895 de 1018
896 de 1018
897 de 1018
898 de 1018
899 de 1018
900 de 1018
901 de 1018
902 de 1018
903 de 1018
904 de 1018
905 de 1018
906 de 1018
907 de 1018
908 de 1018
909 de 1018
910 de 1018
911 de 1018
912 de 1018
913 de 1018
914 de 1018
915 de 1018
916 de 1018
917 de 1018
918 de 1018
919 de 1018
920 de 1018
921 de 1018
922 de 1018
923 de 1018
924 de 1018
925 de 1018
926 de 1018
927 de 1018
928 de 1018
929 de 1018
930 de 1018
931 de 1018
932 de 1018
933 de 1018
934 de 1018
935 de 1018
936 de 1018
937 de 1018
938 de 1018
939 de 1018

940 de 1018
941 de 1018
942 de 1018
943 de 1018
944 de 1018
945 de 1018
946 de 1018
947 de 1018
948 de 1018
949 de 1018
950 de 1018
951 de 1018
952 de 1018
953 de 1018
954 de 1018
955 de 1018
956 de 1018
957 de 1018
958 de 1018
959 de 1018
960 de 1018
961 de 1018
962 de 1018
963 de 1018
964 de 1018
965 de 1018
966 de 1018
967 de 1018
968 de 1018
969 de 1018
970 de 1018
971 de 1018
972 de 1018
973 de 1018
974 de 1018
975 de 1018
976 de 1018
977 de 1018
978 de 1018
979 de 1018
980 de 1018
981 de 1018
982 de 1018
983 de 1018
984 de 1018
985 de 1018
986 de 1018
987 de 1018
988 de 1018
989 de 1018
990 de 1018
991 de 1018
992 de 1018
993 de 1018
994 de 1018
995 de 1018
996 de 1018
997 de 1018
998 de 1018

```
999 de 1018
1000 de 1018
1001 de 1018
1002 de 1018
1003 de 1018
1004 de 1018
1005 de 1018
1006 de 1018
1007 de 1018
1008 de 1018
1009 de 1018
1010 de 1018
1011 de 1018
1012 de 1018
1013 de 1018
1014 de 1018
1015 de 1018
1016 de 1018
1017 de 1018
1018 de 1018
```

```
In [7]: plot_extremal.plot_pot_lmom(umb_i, treval_i, valmed_lmom_i, liminf_i, limsup_i,
        au2_lmom_i, au2pv_lmom,
                                    nanios_i, '$H_s$')
        plt.show()
```

```
/Users/andrealira/CloudStation/ROM__1_1/Herramienta/clima_maritimo/graficas/plot_extre
mal.py:55: MatplotlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
  plt.hold(True)
/Users/andrealira/anaconda/lib/python2.7/site-packages/matplotlib/__init__.py:917:
UserWarning: axes.hold is deprecated. Please remove it from your matplotlibrc and/or
style files.
  warnings.warn(self.msg_depr_set % key)
/Users/andrealira/anaconda/lib/python2.7/site-packages/matplotlib/rcsetup.py:152:
UserWarning: axes.hold is deprecated, will be removed in 3.0
  warnings.warn("axes.hold is deprecated, will be removed in 3.0")
```



Se observa que existen varios mínimos en las curvas del complemento del $p - valor$ y el estadistico de Anderson-Darling modificado, los cuales seria necesario analizar y estudiar el ajuste a la funcion de distribucion correspondiente. En este caso se elige el umbral que minimiza el complemento del p-valor y del estadistico.

```
In [8]: umbral = umb_i[np.argmin(au2pv_lmom)]
        np.save(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'), umbral)

        print umbral
```

```
2.17862434623
```

Continuar con Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno.

Referencias:

Solari, S., Egüen, M., Polo, M. J., & Losada, M. A. (2017). Peaks over threshold (POT): A methodology for automatic threshold estimation using goodnessoffit pvalue. Water Resources Research. doi:10.1002/2016WR019426

cl_regimen_extremal_04

April 27, 2018

0.0.1 Distribución de extremos de la variable predominante y obtención de su valor para un determinado periodo de retorno

Este ejemplo continuará a partir de los datos temporales de altura de ola H_s guardados al final del apartado Pretratamiento de series temporales, la serie de picos independientes obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos y el umbral elegido para el anañisis POT en el apartado Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT).

Utiliando la serie de eventos extremos y el umbral elegido se ajustan los datos a una función de distribución Generalizada de Pareto y a una función Exponencial utilizando técnicas de remuestreo. Se obtiene la altura significante de ola H_s para una serie de periodos de retorno utilizando los ajustes obtenidos.

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.regimen_extremal.automatico_lmom_boot
- clima_maritimo.graficas.plot_extremal.plot_autom

```
In [2]: # imports Anaconda
        from __future__ import division
        import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from IPython.display import HTML

        # imports ROM 1.1
        from clima_maritimo.clima_maritimo.fdist import regimen_extremal
        from clima_maritimo.graficas import plot_extremal

        dir_data = os.path.join(env.data_path, 'clima')
        df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
        df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
        umbral = np.load(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'))

        tit_pp = 'Hs'
        lab_pp = '$H_s$ (m)'

        nanios = len(np.unique(df.index.year))
```

Entradas de usuario:

- Nivel de significancia para las bandas de confianza *alpha*. (0.1 para tener nivel significancia al 95%)
- Tipo de técnica de remuestreo *param*. Se pueden utilizar las opciones 'parametrico' o 'no_parametrico'.

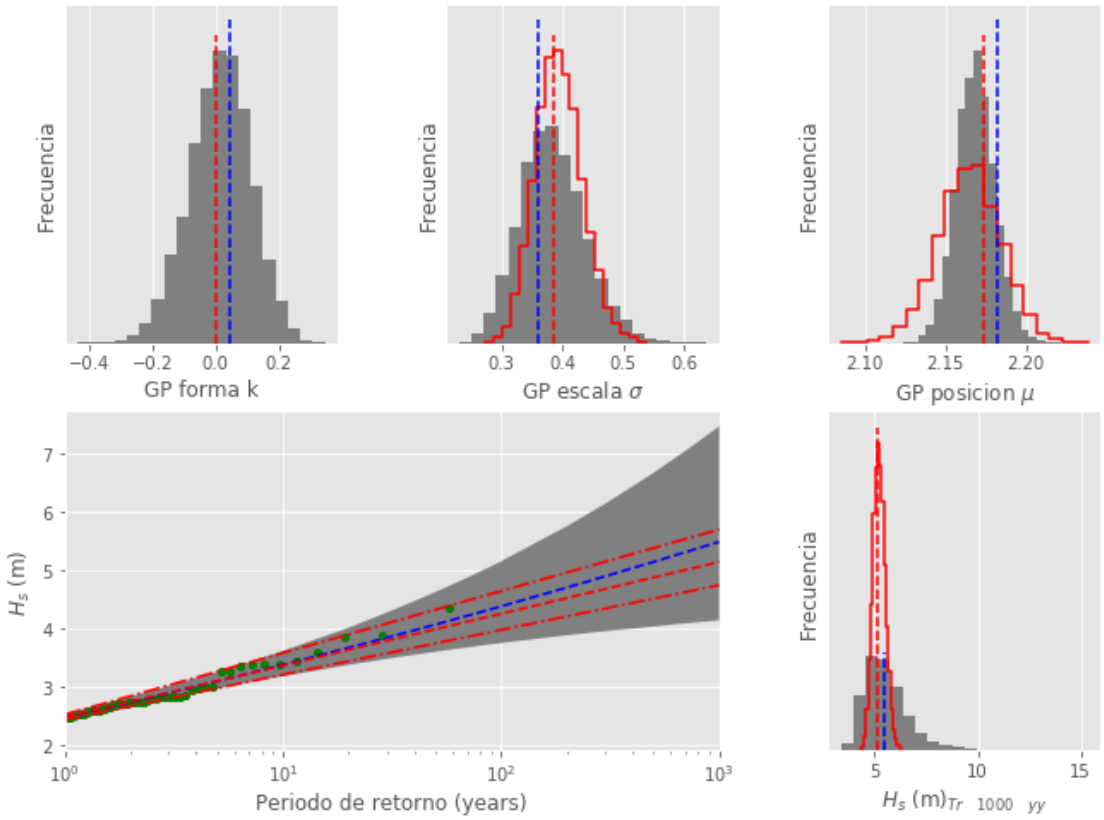
- Método de cálculo de los intervalos de confianza *bca*. Se pueden utilizar las opciones 'estandar' o 'bca'.
- Valores de periodos de retorno a estudiar *tr_calc*.

```
In [3]: alpha = 0.1
        param = 'parametrico'
        bca = 'estandar'
        tr_calc = [5, 200, 500, 750]
```

Utilizando la serie de eventos extremos y el umbral elegido es posible caracterizar el régimen extremal de la altura de ola significante. Utilizando técnicas de remuestreo se ajustan los datos a una función de distribución Generalizada de Pareto y a una función Exponencial.

En la figura se presentan los histogramas (paneles superiores) de los parametros de la GP (en gris) y exponencial (en trazo rojo) obtenidos de la aplicacion de la tecnica de remuestreo para 10000 casos. En trazo discontinuo azul y rojo se ha representado el valor medio de los parametros de ambas funciones de distribucion. En el panel inferior izquierdo se ha representado la altura de ola significante obtenida, con el intervalo de confianza para periodos de retorno entre 1 y 1000 anos, para los ajustes con GP y Exponencial. Por ultimo, el panel inferior derecho, presenta el histograma de altura de ola significante para el periodo de retorno de 1000 anos.

```
In [4]: nanios = len(np.unique(df.index.year))

        boot_pot, orig_pot, ci_pot, tr_pot, picos_pot, npicos_pot, eventanu_pot = \
                            regimen_extremal.automatico_lmom_boot(df_picos, alpha, umbral,
        param, bca, nanios)

        plot_extremal.plot_autom(boot_pot, orig_pot, ci_pot, tr_pot, picos_pot, npicos_pot,
        eventanu_pot, 'GP', lab_pp)
        plt.show()

        # Guardar a archivo 'out_pot_ini_hs_sim_ddwms_p90.npy'
        out_pot_def = boot_pot, orig_pot, ci_pot, tr_pot, picos_pot, npicos_pot, eventanu_pot
        np.save(os.path.join(dir_data, 'out_pot_def_hs_sim.npy'), out_pot_def)
```



Con este ajuste es posible obtener el valor medio y los intervalos de confianza de la altura de ola signifcante H_s para un valor de periodo de retorno dado T_R.

```
In [5]: vmed_calc = np.interp(tr_calc, tr_pot, orig_pot[0][0][4:])
        ciinf_calc = np.interp(tr_calc, tr_pot, ci_pot[0][0][0, 4:])
        cisup_calc = np.interp(tr_calc, tr_pot, ci_pot[0][0][1, 4:])
```

```
In [6]: # definición del df con la salida
        df_out = pd.DataFrame({tit_pp: vmed_calc, tit_pp + '_ci_sup': cisup_calc, tit_pp +
        '_ci_inf': ciinf_calc, 'Tr': tr_calc})

        # Guardar html y df
        r_ds = os.path.join(env.output_path, 'clima', 'out_reg_extremal__' + tit_pp +
        '_tr.html')
        df_out.to_html(r_ds, sparsify=False)

        df_out.to_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp + '_tr.pkl'))

        # Mostrar en pantalla
        HTML(filename=r_ds)
```

```
Out[6]: <IPython.core.display.HTML object>
```

cl_regresion_Dh_02

April 27, 2018

0.0.1 Funciones de regresion para la caracterizacion de las variables no predominantes

NOTA: Este ejemplo es repetición de: Funciones de regresion para la caracterizacion de las variables no predominantes para la variable dirección media de procedencia del oleaje ste ejemplo continuará a partir de los datos temporales de altura de ola significante H_s Transformacion de los estados de las variables oceanograficas al emplazamiento, la serie de picos independientes de altura de ola significante H_s obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos, el umbral elegido para el anañisis POT en el apartado Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT) y los resultados del análisis extremal de H_S para los periodos de retorno estudiados obtienidos en el apartado Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno.

Se utilizan los valores concomitantes del registro de eventos extremos de la altura de ola significante H_s y se ajustan varios tipos de funciones de regresion. Es posible realizar ajustes utilizando funciones polinomiales de primer y segundo orden ('poli1', 'poli2'), potenciales de primer y segundo orden ('pote1, pote2'), hiperbolicas de primer y segundo orden ('pab1, pab2') y exponenciales de primer y segundo orden ('exp1, exp2').

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_umb
- clima_maritimo.clima_maritimo.fdist.regimen_conjunto.regresion
- clima_maritimo.graficas.plot_regresion.ajustes_sobre_umbral

```
In [4]: # imports Anaconda
        from __future__ import division
        import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from IPython.display import HTML

        # imports ROM 1.1
        from clima_maritimo.clima_maritimo.fdist import utils, regimen_conjunto
        from clima_maritimo.graficas import plot_regresion

        dir_data = os.path.join(env.data_path, 'clima')

        df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
        df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
        umbral = np.load(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'))
        tit_pp = 'Hs'
        lab_pp = '$H_s$ (m)'
        df_out_pp = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp +
        '_tr.pkl'))
```

```
        # Se carga de archivo el datframe con los datos de la variable no predominante
        df_sec = pd.read_pickle(os.path.join(dir_data, 'cadiz_dh_sim_emp.pkl'))
        nm_sec = df_sec.iloc[:, 0].name
        tit_sec = 'Dh'
        lab_sec = r'$\theta$ ($^\circ$)'
```

Entradas de usuario:

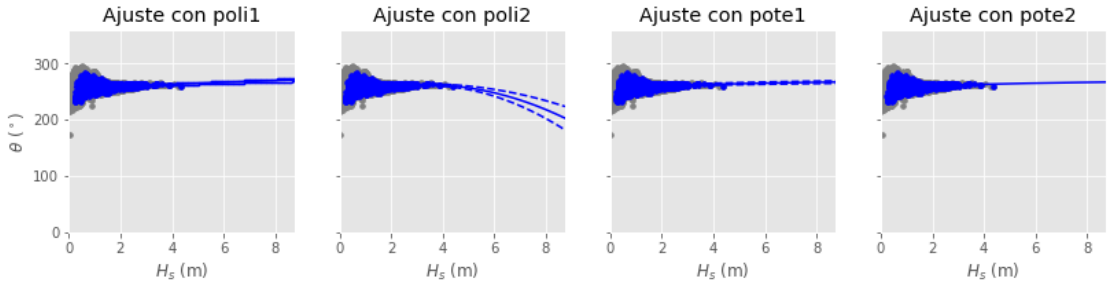
- Tipo de funciones de regresión que se desean analizar *fun_reg*.

```
In [5]: fun_reg =['poli1', 'poli2', 'pote1', 'pote2']
```

La figura muestra los resultados del análisis de regresión con las funciones elegidas. Las lineas continuas son el valor medio del ajuste y las discontinuas, el intervalo de confianza, elegido con un nivel de significancia del 0.05. Si la variable no predominante es circular (dirección de procedencia de oleaje o viento), el análisis se realiza utilizando los rangos de las direcciones principales. También se permite que el usuario elija las direcciones que desea estudiar agregando una entrada a la función *regresion*.

```
In [6]: pos_picos_umb_pot = utils.picos_umb(df_picos.values, [umbral])
        df_picos_umb_pot = df_picos.iloc[pos_picos_umb_pot[0], :]

        df_picos_merge = pd.merge(df_picos, df_sec, left_index=True, right_index=True,
        how='inner')
        df_merge = pd.merge(df, df_sec, left_index=True, right_index=True, how='inner')
        df_merge.dropna(inplace=True)

        reg = regimen_conjunto.regresion([df_picos_merge], df_picos, [umbral], [nm_sec],
        fun_reg, True)
        reg.to_pickle(os.path.join(dir_data, 'regres_' + tit_pp + '_' + tit_sec + '.npy'))

        plot_regresion.ajustes_sobre_umbral([df_picos_merge], df_merge, reg, [umbral], [nm_sec],
        fun_reg, [lab_pp, lab_sec])
```

```
/Users/andrealira/CloudStation/ROM__1_1/Herramienta/clima_maritimo/clima_maritimo/fdis
t/regimen_conjunto.py:304: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df_prov['d_bins'] = pd.cut(df_prov[comb[k - 1]], bins=dir_bins, labels=dir_lab)
/Users/andrealira/anaconda/lib/python2.7/site-packages/pandas/core/indexing.py:477:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```



El usuario debe elegir la función elegida fun_reg_def para obtener el valor medio y los intervalos de confianza del periodo pico T_p para los periodos de retorno estudiados.

```
In [7]: fun_reg_def = 'pote1'

In [8]: # Si la variable no predominante es circular, los resultados se dan para las direcciones
        principales
        if ((nm_sec == 'dh') | (nm_sec == 'dv')):
            ndirp = reg['ndirp' + nm_sec + fun_reg_def + str(umbral)][0]
            for kk in range(ndirp):
                out_x = reg['x' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
                out_y = reg['y' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
                out_y_inf = reg['yinf' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
                out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]

                if kk==0:
                    for row_df in df_out_pp.itertuples():
                        val_calc = getattr(row_df, tit_pp)
                        vmed_calc = np.interp(val_calc, out_x, out_y)
                        ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                        cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                        df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                        df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                        df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc
                else:
                    df_out_pp_prv = df_out_pp.copy()
                    for row_df in df_out_pp.itertuples():
                        val_calc = getattr(row_df, tit_pp)
                        vmed_calc = np.interp(val_calc, out_x, out_y)
                        ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                        cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                        df_out_pp_prv.loc[row_df.Index, tit_sec] = vmed_calc
                        df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                        df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

                    df_out_pp_sec = df_out_pp.append(df_out_pp_prv)

                if ndirp == 1:
                    df_out_pp_sec = df_out_pp

        else:
            out_x = reg['x' + nm_sec + fun_reg_def + str(umbral)]
            out_y = reg['y' + nm_sec + fun_reg_def + str(umbral)]
            out_y_inf = reg['yinf'+ nm_sec + fun_reg_def + str(umbral)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral)]

            for row_df in df_out_pp.itertuples():
                val_calc = getattr(row_df, tit_pp)
                vmed_calc = np.interp(val_calc, out_x, out_y)
```

```
            ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
            cisup_calc = np.interp(val_calc, out_x, out_y_sup)

            df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
            df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
            df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

        df_out_pp_sec = df_out_pp

    # Guardar html y df
    r_ds = os.path.join(env.output_path, 'clima', 'out_reg_extremal__' + tit_pp + '_' +
    tit_sec + '_tr.html')
    df_out_pp_sec.to_html(r_ds, sparsify=False)

    df_out_pp_sec.to_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp + '_' +
    tit_sec + '_tr.pkl'))

    # Mostrar en pantalla
    HTML(filename=r_ds)
```

Out[8]: <IPython.core.display.HTML object>

cl_regresion_Dv_04

April 27, 2018

0.0.1 Funciones de regresion para la caracterizacion de las variables no predominantes

NOTA: Este ejemplo es repetición de: Funciones de regresion para la caracterizacion de las variables no predominantes para la variable dirección media de procedencia de viento. En este ejemplo se utilizan los datos de viento con pretratamiento. ste ejemplo continuará a partir de los datos temporales de altura de ola significante H_s Transformacion de los estados de las variables oceanograficas al emplazamiento, la serie de picos independientes de altura de ola significante H_s obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos, el umbral elegido para el anañisis POT en el apartado Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT) y los resultados del análisis extremal de H_S para los periodos de retorno estudiados obtienidos en el apartado Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno.

Se utilizan los valores concomitantes del registro de eventos extremos de la altura de ola significante H_s y se ajustan varios tipos de funciones de regresion. Es posible realizar ajustes utilizando funciones polinomiales de primer y segundo orden ('poli1', 'poli2'), potenciales de primer y segundo orden ('pote1, pote2'), hiperbolicas de primer y segundo orden ('pab1, pab2') y exponenciales de primer y segundo orden ('exp1, exp2').

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_umb
- clima_maritimo.clima_maritimo.fdist.regimen_conjunto.regresion
- clima_maritimo.graficas.plot_regresion.ajustes_sobre_umbral

```
In [15]: # imports Anaconda
         from __future__ import division
         import os
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from IPython.display import HTML

         # imports ROM 1.1
         from clima_maritimo.clima_maritimo.fdist import utils, regimen_conjunto
         from clima_maritimo.graficas import plot_regresion

         dir_data = os.path.join(env.data_path, 'clima')

         df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
         df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
         umbral = np.load(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'))
         tit_pp = 'Hs'
         lab_pp = '$H_s$ (m)'
         df_out_pp = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp +
```

```
'_tr.pkl'))

# Se carga de archivo el datframe con los datos de la variable no predominante
df_sec = pd.read_pickle(os.path.join(dir_data, 'cadiz_dv_sim_pretrat.pkl'))
nm_sec = df_sec.iloc[:, 0].name
tit_sec = 'Du'
lab_sec = r'$\theta_U$ ($^\circ$)'
```

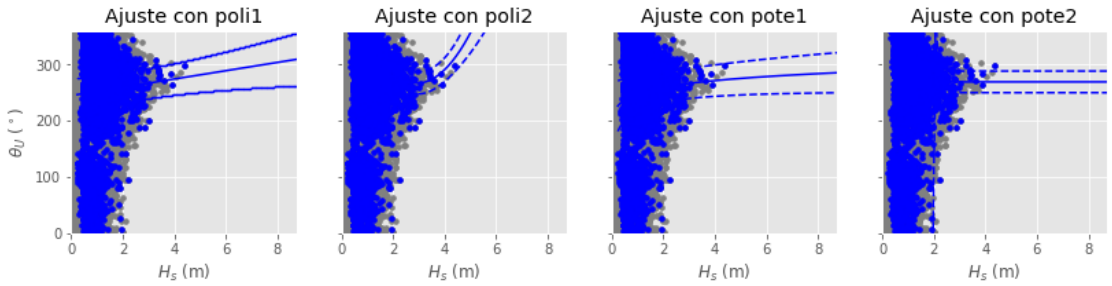
Entradas de usuario:

- Tipo de funciones de regresión que se desean analizar fun_reg.

```
In [16]: fun_reg =['poli1', 'poli2', 'pote1', 'pote2']
```

La figura muestra los resultados del análisis de regresión con las funciones elegidas. Las lineas continuas son el valor medio del ajuste y las discontinuas, el intervalo de confianza, elegido con un nivel de significancia del 0.05. Si la variable no predominante es circular (dirección de procedencia de oleaje o viento), el análisis se realiza utilizando los rangos de las direcciones principales. También se permite que el usuario elija las direcciones que desea estudiar agregando una entrada a la función *regresion*.

```
In [17]: pos_picos_umb_pot = utils.picos_umb(df_picos.values, [umbral])
         df_picos_umb_pot = df_picos.iloc[pos_picos_umb_pot[0], :]

         df_picos_merge = pd.merge(df_picos, df_sec, left_index=True, right_index=True,
         how='inner')
         df_merge = pd.merge(df, df_sec, left_index=True, right_index=True, how='inner')
         df_merge.dropna(inplace=True)

         reg = regimen_conjunto.regresion([df_picos_merge], df_picos, [umbral], [nm_sec],
         fun_reg, True)
         reg.to_pickle(os.path.join(dir_data, 'regres_' + tit_pp + '_' + tit_sec + '.npy'))

         plot_regresion.ajustes_sobre_umbral([df_picos_merge], df_merge, reg, [umbral], [nm_sec],
         fun_reg, [lab_pp, lab_sec])
```



El usuario debe elegir la función elegida fun_reg_def para obtener el valor medio y los intervalos de confianza del periodo pico T_p para los periodos de retorno estudiados.

```
In [18]: fun_reg_def = 'pote1'
```

```
In [19]: # Si la variable no predominante es circular, los resultados se dan para las direcciones
         principales
         if ((nm_sec == 'dh') | (nm_sec == 'dv')):
             ndirp = reg['ndirp' + nm_sec + fun_reg_def + str(umbral)][0]
             for kk in range(ndirp):
```

```
            out_x = reg['x' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y = reg['y' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y_inf = reg['yinf' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]

            if kk==0:
                for row_df in df_out_pp.itertuples():
                    val_calc = getattr(row_df, tit_pp)
                    vmed_calc = np.interp(val_calc, out_x, out_y)
                    ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                    cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                    df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                    df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                    df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc
            else:
                df_out_pp_prv = df_out_pp.copy()
                for row_df in df_out_pp.itertuples():
                    val_calc = getattr(row_df, tit_pp)
                    vmed_calc = np.interp(val_calc, out_x, out_y)
                    ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                    cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                    df_out_pp_prv.loc[row_df.Index, tit_sec] = vmed_calc
                    df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                    df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

                df_out_pp_sec = df_out_pp.append(df_out_pp_prv)

            if ndirp == 1:
                df_out_pp_sec = df_out_pp

        else:
            out_x = reg['x' + nm_sec + fun_reg_def + str(umbral)]
            out_y = reg['y' + nm_sec + fun_reg_def + str(umbral)]
            out_y_inf = reg['yinf'+ nm_sec + fun_reg_def + str(umbral)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral)]

            for row_df in df_out_pp.itertuples():
                val_calc = getattr(row_df, tit_pp)
                vmed_calc = np.interp(val_calc, out_x, out_y)
                ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

            df_out_pp_sec = df_out_pp

        # Guardar html y df
        r_ds = os.path.join(env.output_path, 'clima', 'out_reg_extremal__' + tit_pp + '_' +
        tit_sec + '_tr.html')
        df_out_pp_sec.to_html(r_ds, sparsify=False)

        df_out_pp_sec.to_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp + '_' +
        tit_sec + '_tr.pkl'))

        # Mostrar en pantalla
        HTML(filename=r_ds)
```

Out[19]: <IPython.core.display.HTML object>

cl_regresion_Mme_05

April 27, 2018

0.0.1 Funciones de regresion para la caracterizacion de las variables no predominantes

NOTA: Este ejemplo es repetición de: Funciones de regresion para la caracterizacion de las variables no predominantes para la variable nivel de mar debido a la marea meteorologica. En este ejemplo se utilizan los datos del pretratamiento. ste ejemplo continuará a partir de los datos temporales de altura de ola significante H_s Transformacion de los estados de las variables oceanograficas al emplazamiento, la serie de picos independientes de altura de ola significante H_s obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos, el umbral elegido para el anañisis POT en el apartado Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT) y los resultados del análisis extremal de H_S para los periodos de retorno estudiados obtenidos en el apartado Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno.

Se utilizan los valores concomitantes del registro de eventos extremos de la altura de ola significante H_s y se ajustan varios tipos de funciones de regresion. Es posible realizar ajustes utilizando funciones polinomiales de primer y segundo orden ('poli1', 'poli2'), potenciales de primer y segundo orden ('pote1, pote2'), hiperbolicas de primer y segundo orden ('pab1, pab2') y exponenciales de primer y segundo orden ('exp1, exp2').

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_umb
- clima_maritimo.clima_maritimo.fdist.regimen_conjunto.regresion
- clima_maritimo.graficas.plot_regresion.ajustes_sobre_umbral

```
In [16]: # imports Anaconda
         from __future__ import division
         import os
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from IPython.display import HTML

         # imports ROM 1.1
         from clima_maritimo.clima_maritimo.fdist import utils, regimen_conjunto
         from clima_maritimo.graficas import plot_regresion

         dir_data = os.path.join(env.data_path, 'clima')

         df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
         df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
         umbral = np.load(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'))
         tit_pp = 'Hs'
         lab_pp = '$H_s$ (m)'
         df_out_pp = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp +
```

```
'_tr.pkl'))

# Se carga de archivo el datframe con los datos de la variable no predominante
df_sec = pd.read_pickle(os.path.join(dir_data, 'cadiz_mme_pretrat.pkl'))
nm_sec = df_sec.iloc[:, 0].name
tit_sec = 'Mme'
lab_sec = r'$\eta_{Mme}$ ($m$)'
```

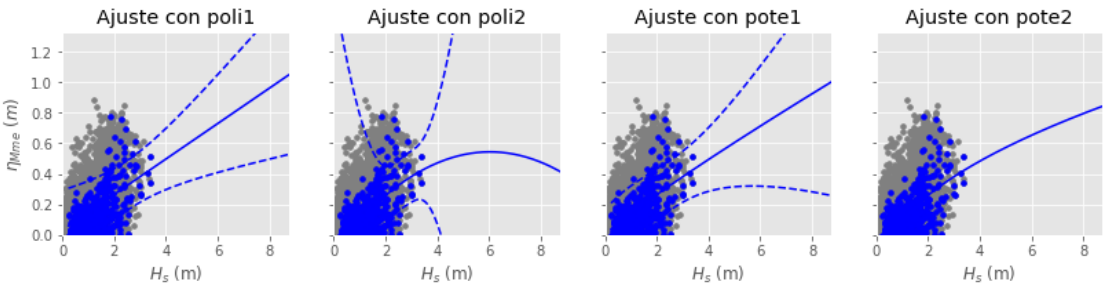
Entradas de usuario:

- Tipo de funciones de regresión que se desean analizar fun_reg.

```
In [17]: fun_reg =['poli1', 'poli2', 'pote1', 'pote2']
```

La figura muestra los resultados del análisis de regresión con las funciones elegidas. Las lineas continuas son el valor medio del ajuste y las discontinuas, el intervalo de confianza, elegido con un nivel de significancia del 0.05. Si la variable no predominante es circular (dirección de procedencia de oleaje o viento), el análisis se realiza utilizando los rangos de las direcciones principales. También se permite que el usuario elija las direcciones que desea estudiar agregando una entrada a la función *regresion*.

```
In [18]: pos_picos_umb_pot = utils.picos_umb(df_picos.values, [umbral])
         df_picos_umb_pot = df_picos.iloc[pos_picos_umb_pot[0], :]

         df_picos_merge = pd.merge(df_picos, df_sec, left_index=True, right_index=True,
         how='inner')
         df_merge = pd.merge(df, df_sec, left_index=True, right_index=True, how='inner')
         df_merge.dropna(inplace=True)

         reg = regimen_conjunto.regresion([df_picos_merge], df_picos, [umbral], [nm_sec],
         fun_reg, True)
         reg.to_pickle(os.path.join(dir_data, 'regres_' + tit_pp + '_' + tit_sec + '.npy'))

         plot_regresion.ajustes_sobre_umbral([df_picos_merge], df_merge, reg, [umbral], [nm_sec],
         fun_reg, [lab_pp, lab_sec])
```



El usuario debe elegir la función elegida fun_reg_def para obtener el valor medio y los intervalos de confianza del periodo pico T_p para los periodos de retorno estudiados.

```
In [19]: fun_reg_def = 'pote1'
```

```
In [20]: # Si la variable no predominante es circular, los resultados se dan para las direcciones
         principales
         if ((nm_sec == 'dh') | (nm_sec == 'dv')):
             ndirp = reg['ndirp' + nm_sec + fun_reg_def + str(umbral)][0]
             for kk in range(ndirp):
```

```
            out_x = reg['x' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y = reg['y' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y_inf = reg['yinf' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]

            if kk==0:
                for row_df in df_out_pp.itertuples():
                    val_calc = getattr(row_df, tit_pp)
                    vmed_calc = np.interp(val_calc, out_x, out_y)
                    ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                    cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                    df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                    df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                    df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc
            else:
                df_out_pp_prv = df_out_pp.copy()
                for row_df in df_out_pp.itertuples():
                    val_calc = getattr(row_df, tit_pp)
                    vmed_calc = np.interp(val_calc, out_x, out_y)
                    ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                    cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                    df_out_pp_prv.loc[row_df.Index, tit_sec] = vmed_calc
                    df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                    df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

                df_out_pp_sec = df_out_pp.append(df_out_pp_prv)

            if ndirp == 1:
                df_out_pp_sec = df_out_pp

        else:
            out_x = reg['x' + nm_sec + fun_reg_def + str(umbral)]
            out_y = reg['y' + nm_sec + fun_reg_def + str(umbral)]
            out_y_inf = reg['yinf'+ nm_sec + fun_reg_def + str(umbral)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral)]

            for row_df in df_out_pp.itertuples():
                val_calc = getattr(row_df, tit_pp)
                vmed_calc = np.interp(val_calc, out_x, out_y)
                ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

            df_out_pp_sec = df_out_pp

        # Guardar html y df
        r_ds = os.path.join(env.output_path, 'clima', 'out_reg_extremal__' + tit_pp + '_' +
        tit_sec + '_tr.html')
        df_out_pp_sec.to_html(r_ds, sparsify=False)

        df_out_pp_sec.to_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp + '_' +
        tit_sec + '_tr.pkl'))

        # Mostrar en pantalla
        HTML(filename=r_ds)
```

Out[20]: <IPython.core.display.HTML object>

cl_regresion_Tp_01

April 27, 2018

0.0.1 Funciones de regresion para la caracterizacion de las variables no predominantes

ste ejemplo continuará a partir de los datos temporales de altura de ola significante H_s y periodo pico del oleaje T_p guardados al final del apartado Transformacion de los estados de las variables oceanograficas al emplazamiento, la serie de picos independientes de altura de ola significante H_s obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos, el umbral elegido para el anañisis POT en el apartado Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT) y los resultados del análisis extremal de H_S para los periodos de retorno estudiados obtenidos en el apartado Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno.

Se utilizan los valores concomitantes del registro de eventos extremos de la altura de ola significante H_s y se ajustan varios tipos de funciones de regresion. Es posible realizar ajustes utilizando funciones polinomiales de primer y segundo orden ('poli1', 'poli2'), potenciales de primer y segundo orden ('pote1, pote2'), hiperbolicas de primer y segundo orden ('pab1, pab2') y exponenciales de primer y segundo orden ('exp1, exp2').

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_umb
- clima_maritimo.clima_maritimo.fdist.regimen_conjunto.regresion
- clima_maritimo.graficas.plot_regresion.ajustes_sobre_umbral

```
In [4]: # imports Anaconda
        from __future__ import division
        import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from IPython.display import HTML

        # imports ROM 1.1
        from clima_maritimo.clima_maritimo.fdist import utils, regimen_conjunto
        from clima_maritimo.graficas import plot_regresion

        dir_data = os.path.join(env.data_path, 'clima')

        df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
        df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
        umbral = np.load(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'))
        tit_pp = 'Hs'
        lab_pp = '$H_s$ (m)'
        df_out_pp = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp +
        '_tr.pkl'))
```

```
        # Se carga de archivo el datframe con los datos de la variable no predominante
        df_sec = pd.read_pickle(os.path.join(dir_data, 'cadiz_tp_sim_emp.pkl'))
        nm_sec = df_sec.iloc[:, 0].name
        tit_sec = 'Tp'
        lab_sec = '$T_p$ (s)'
```

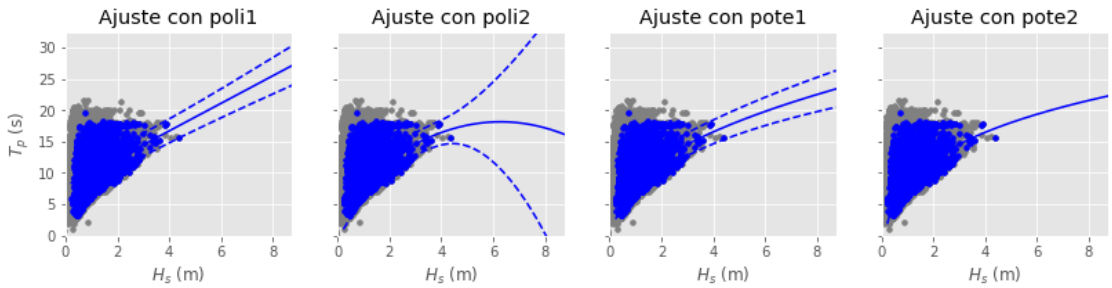
Entradas de usuario:

- Tipo de funciones de regresión que se desean analizar fun_reg.

```
In [5]: fun_reg =['poli1', 'poli2', 'pote1', 'pote2']
```

La figura muestra los resultados del análisis de regresión con las funciones elegidas. Las lineas continuas son el valor medio del ajuste y las discontinuas, el intervalo de confianza, elegido con un nivel de significancia del 0.05. Si la variable no predominante es circular (dirección de procedencia de oleaje o viento), el análisis se realiza utilizando los rangos de las direcciones principales. También se permite que el usuario elija las direcciones que desea estudiar agregando una entrada a la función *regresion*.

```
In [6]: pos_picos_umb_pot = utils.picos_umb(df_picos.values, [umbral])
        df_picos_umb_pot = df_picos.iloc[pos_picos_umb_pot[0], :]

        df_picos_merge = pd.merge(df_picos, df_sec, left_index=True, right_index=True,
        how='inner')
        df_merge = pd.merge(df, df_sec, left_index=True, right_index=True, how='inner')
        df_merge.dropna(inplace=True)

        reg = regimen_conjunto.regresion([df_picos_merge], df_picos, [umbral], [nm_sec],
        fun_reg, True)
        reg.to_pickle(os.path.join(dir_data, 'regres_hs_tp.npy'))

        plot_regresion.ajustes_sobre_umbral([df_picos_merge], df_merge, reg, [umbral], [nm_sec],
        fun_reg, [lab_pp, lab_sec])
```



El usuario debe elegir la función elegida fun_reg_def para obtener el valor medio y los intervalos de confianza del periodo pico T_p para los periodos de retorno estudiados.

```
In [7]: fun_reg_def = 'pote1'
```

```
In [8]: # Si la variable no predominante es circular, los resultados se dan para las direcciones
        principales
        if ((nm_sec == 'dh') | (nm_sec == 'dv')):
            ndirp = reg['ndirp' + nm_sec + fun_reg_def + str(umbral)][0]
            for kk in range(ndirp):
                out_x = reg['x' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
                out_y = reg['y' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
```

```
            out_y_inf = reg['yinf' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]

            if kk==0:
                for row_df in df_out_pp.itertuples():
                    val_calc = getattr(row_df, tit_pp)
                    vmed_calc = np.interp(val_calc, out_x, out_y)
                    ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                    cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                    df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                    df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                    df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc
            else:
                df_out_pp_prv = df_out_pp.copy()
                for row_df in df_out_pp.itertuples():
                    val_calc = getattr(row_df, tit_pp)
                    vmed_calc = np.interp(val_calc, out_x, out_y)
                    ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                    cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                    df_out_pp_prv.loc[row_df.Index, tit_sec] = vmed_calc
                    df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                    df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

                df_out_pp_sec = df_out_pp.append(df_out_pp_prv)

            if ndirp == 1:
                df_out_pp_sec = df_out_pp

        else:
            out_x = reg['x' + nm_sec + fun_reg_def + str(umbral)]
            out_y = reg['y' + nm_sec + fun_reg_def + str(umbral)]
            out_y_inf = reg['yinf'+ nm_sec + fun_reg_def + str(umbral)]
            out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral)]

            for row_df in df_out_pp.itertuples():
                val_calc = getattr(row_df, tit_pp)
                vmed_calc = np.interp(val_calc, out_x, out_y)
                ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

            df_out_pp_sec = df_out_pp

        # Guardar html y df
        r_ds = os.path.join(env.output_path, 'clima', 'out_reg_extremal__' + tit_pp + '_' +
        tit_sec + '_tr.html')
        df_out_pp_sec.to_html(r_ds, sparsify=False)

        df_out_pp_sec.to_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp + '_' +
        tit_sec + '_tr.pkl'))

        # Mostrar en pantalla
        HTML(filename=r_ds)
```

Out[8]: <IPython.core.display.HTML object>

cl_regresion_Vv_03

April 27, 2018

0.0.1 Funciones de regresion para la caracterizacion de las variables no predominantes

NOTA: Este ejemplo es repetición de: Funciones de regresion para la caracterizacion de las variables no predominantes para la variable velocidad media de viento. En este ejemplo se utilizan los datos de viento con pretratamiento. ste ejemplo continuará a partir de los datos temporales de altura de ola significante H_s Transformacion de los estados de las variables oceanograficas al emplazamiento, la serie de picos independientes de altura de ola significante H_s obtenida en el apartado Distribucion de extremos de la variable predominante: Serie temporal de eventos extremos, el umbral elegido para el anañisis POT en el apartado Distribucion de extremos de la variable predominante: Serie temporal de picos sobre umbral (POT) y los resultados del análisis extremal de H_S para los periodos de retorno estudiados obtienidos en el apartado Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno.

Se utilizan los valores concomitantes del registro de eventos extremos de la altura de ola significante H_s y se ajustan varios tipos de funciones de regresion. Es posible realizar ajustes utilizando funciones polinomiales de primer y segundo orden ('poli1', 'poli2'), potenciales de primer y segundo orden ('pote1, pote2'), hiperbolicas de primer y segundo orden ('pab1, pab2') y exponenciales de primer y segundo orden ('exp1, exp2').

Las funciones que se utilizan son:

- clima_maritimo.clima_maritimo.fdist.utils.picos_umb
- clima_maritimo.clima_maritimo.fdist.regimen_conjunto.regresion
- clima_maritimo.graficas.plot_regresion.ajustes_sobre_umbral

```
In [22]: # imports Anaconda
         from __future__ import division
         import os
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from IPython.display import HTML

         # imports ROM 1.1
         from clima_maritimo.clima_maritimo.fdist import utils, regimen_conjunto
         from clima_maritimo.graficas import plot_regresion

         dir_data = os.path.join(env.data_path, 'clima')

         df = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_emp.pkl'))
         df_picos = pd.read_pickle(os.path.join(dir_data, 'cadiz_hs_sim_picos.pkl'))
         umbral = np.load(os.path.join(dir_data, 'umbral_def_hs_sim_pot.npy'))
         tit_pp = 'Hs'
         lab_pp = '$H_s$ (m)'
         df_out_pp = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp +
         '_tr.pkl'))
```

```
# Se carga de archivo el datframe con los datos de la variable no predominante
df_sec = pd.read_pickle(os.path.join(dir_data, 'cadiz_vv_sim_pretrat.pkl'))
nm_sec = df_sec.iloc[:, 0].name
tit_sec = 'U'
lab_sec = r'$U$ ($m/s$)'
```

Entradas de usuario:

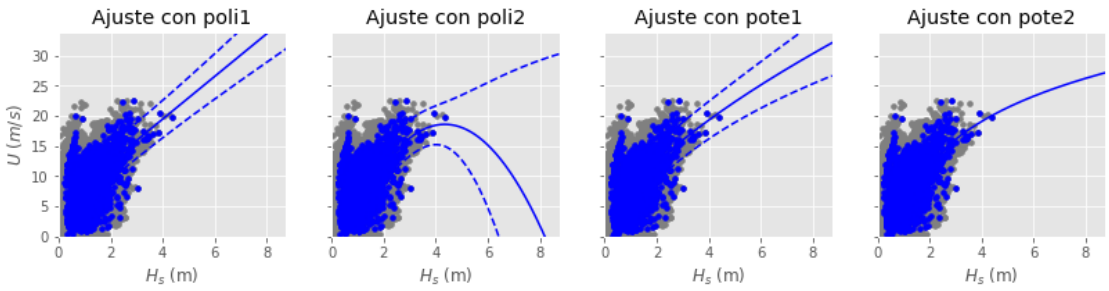
- Tipo de funciones de regresión que se desean analizar fun_reg.

```
In [23]: fun_reg =['poli1', 'poli2', 'pote1', 'pote2']
```

La figura muestra los resultados del análisis de regresión con las funciones elegidas. Las lineas continuas son el valor medio del ajuste y las discontinuas, el intervalo de confianza, elegido con un nivel de significancia del 0.05. Si la variable no predominante es circular (dirección de procedencia de oleaje o viento), el análisis se realiza utilizando los rangos de las direcciones principales. También se permite que el usuario elija las direcciones que desea estudiar agregando una entrada a la función *regresion*.

```
In [24]: pos_picos_umb_pot = utils.picos_umb(df_picos.values, [umbral])
         df_picos_umb_pot = df_picos.iloc[pos_picos_umb_pot[0], :]

         df_picos_merge = pd.merge(df_picos, df_sec, left_index=True, right_index=True,
         how='inner')
         df_merge = pd.merge(df, df_sec, left_index=True, right_index=True, how='inner')
         df_merge.dropna(inplace=True)

         reg = regimen_conjunto.regresion([df_picos_merge], df_picos, [umbral], [nm_sec],
         fun_reg, True)
         reg.to_pickle(os.path.join(dir_data, 'regres_' + tit_pp + '_' + tit_sec + '.npy'))

         plot_regresion.ajustes_sobre_umbral([df_picos_merge], df_merge, reg, [umbral], [nm_sec],
         fun_reg, [lab_pp, lab_sec])
```

```
/Users/andrealira/CloudStation/ROM__1_1/Herramienta/clima_maritimo/clima_maritimo/fdis
t/regimen_conjunto.py:389: RuntimeWarning: invalid value encountered in sqrt
  predstd = st.t.cdf(1-alpha, len(xaux)-len(par[0]))*np.sqrt(predvar)
```



El usuario debe elegir la función elegida fun_reg_def para obtener el valor medio y los intervalos de confianza del periodo pico T_p para los periodos de retorno estudiados.

```
In [25]: fun_reg_def = 'pote1'
```

In [26]:
```
# Si la variable no predominante es circular, los resultados se dan para las direcciones
principales
if ((nm_sec == 'dh') | (nm_sec == 'dv')):
    ndirp = reg['ndirp' + nm_sec + fun_reg_def + str(umbral)][0]
    for kk in range(ndirp):
        out_x = reg['x' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
        out_y = reg['y' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
        out_y_inf = reg['yinf' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]
        out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral) + 'dir' + str(kk)]

        if kk==0:
            for row_df in df_out_pp.itertuples():
                val_calc = getattr(row_df, tit_pp)
                vmed_calc = np.interp(val_calc, out_x, out_y)
                ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc
        else:
            df_out_pp_prv = df_out_pp.copy()
            for row_df in df_out_pp.itertuples():
                val_calc = getattr(row_df, tit_pp)
                vmed_calc = np.interp(val_calc, out_x, out_y)
                ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
                cisup_calc = np.interp(val_calc, out_x, out_y_sup)

                df_out_pp_prv.loc[row_df.Index, tit_sec] = vmed_calc
                df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
                df_out_pp_prv.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

            df_out_pp_sec = df_out_pp.append(df_out_pp_prv)

        if ndirp == 1:
            df_out_pp_sec = df_out_pp

else:
    out_x = reg['x' + nm_sec + fun_reg_def + str(umbral)]
    out_y = reg['y' + nm_sec + fun_reg_def + str(umbral)]
    out_y_inf = reg['yinf'+ nm_sec + fun_reg_def + str(umbral)]
    out_y_sup = reg['ysup' + nm_sec + fun_reg_def + str(umbral)]

    for row_df in df_out_pp.itertuples():
        val_calc = getattr(row_df, tit_pp)
        vmed_calc = np.interp(val_calc, out_x, out_y)
        ciinf_calc = np.interp(val_calc, out_x, out_y_inf)
        cisup_calc = np.interp(val_calc, out_x, out_y_sup)

        df_out_pp.loc[row_df.Index, tit_sec] = vmed_calc
        df_out_pp.loc[row_df.Index, tit_sec + '_ci_sup'] = cisup_calc
        df_out_pp.loc[row_df.Index, tit_sec + '_ci_inf'] = ciinf_calc

    df_out_pp_sec = df_out_pp

# Guardar html y df
r_ds = os.path.join(env.output_path, 'clima', 'out_reg_extremal__' + tit_pp + '_' +
tit_sec + '_tr.html')
df_out_pp_sec.to_html(r_ds, sparsify=False)

df_out_pp_sec.to_pickle(os.path.join(dir_data, 'out_reg_extremal__' + tit_pp + '_' +
tit_sec + '_tr.pkl'))

# Mostrar en pantalla
HTML(filename=r_ds)
```

Out[26]: <IPython.core.display.HTML object>

v_concepcion_obra_01_ep

April 27, 2018

1 Concepción de la obra

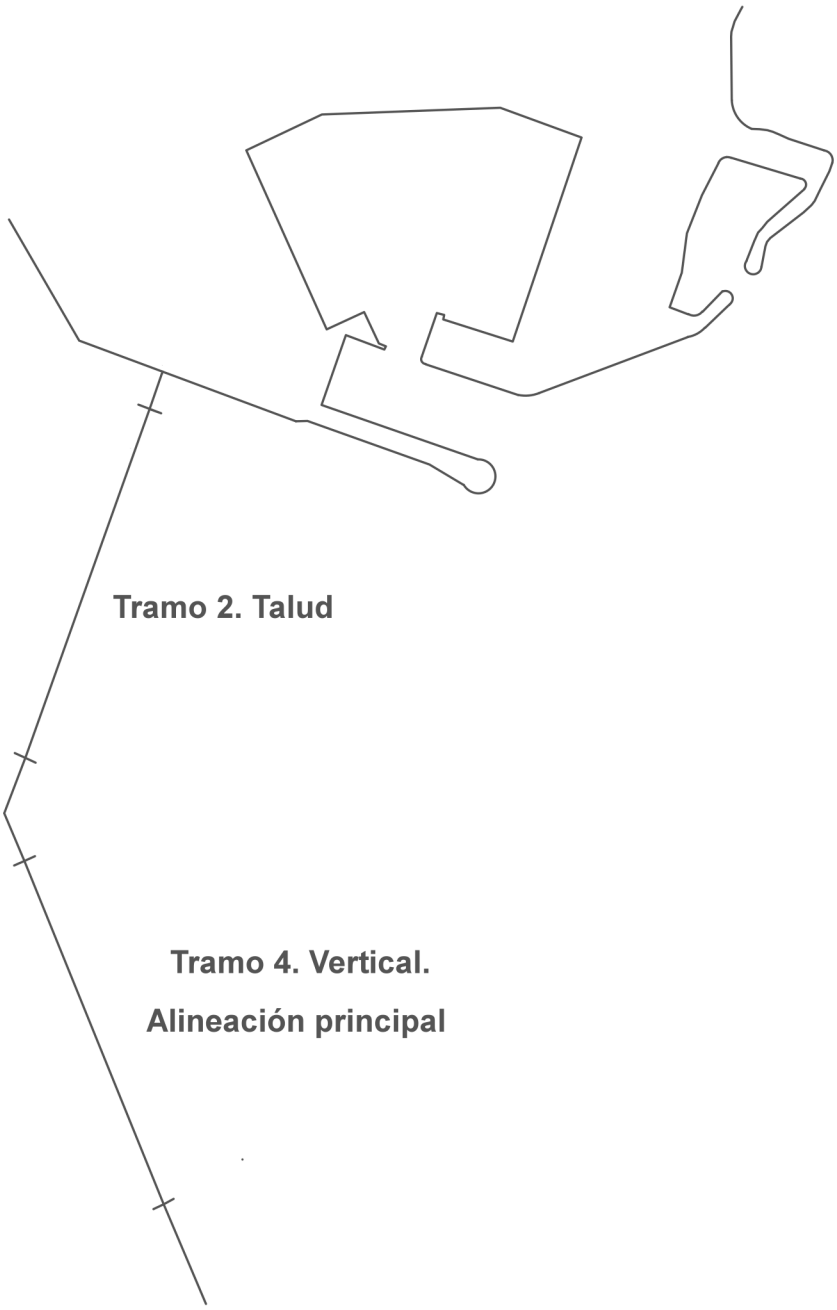
1.1 Importación de paquetes de IPython

```
In [2]: # Python 2/3 setup
        from __future__ import (absolute_import, division, print_function, unicode_literals)
        # from builtins import *

In [3]: # Juypyter setup

        %matplotlib inline
        import os
        import sys
        from IPython.display import HTML
```

1.2 Forma en planta

En este caso se plantea evaluar la viabilidad técnica y económica de la construcción de un dique de abrigo tras denir una forma en planta la cual no tiene referencia a ninguna obra existente o prevista.

```
In [10]: from IPython.display import Image
         img_name = os.path.join(env.input_path, 'obra', 'esquema_dique_planta.png')
         Image(filename=img_name, width=300)

Out[10]:
```



1.2.1 NOTA

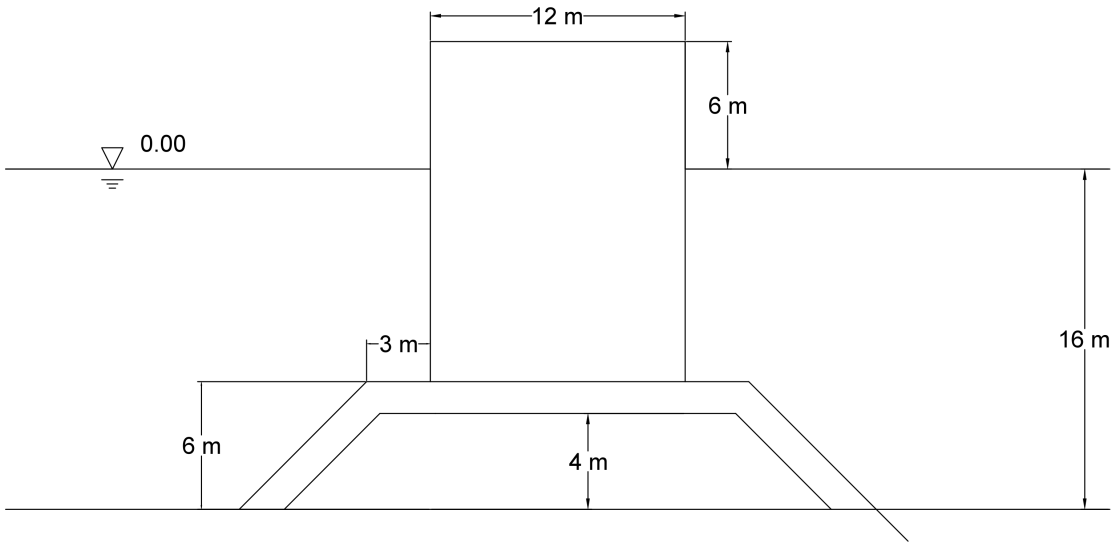
En este ejemplo se presenta el estudio de una forma en planta y una tipología de la alineación principal. De acuerdo al Articulado de la ROM 1.1 esta metodología debería realizarse para diversas formas en planta y tipologías de alineación principal y, en su caso, del morro y arranque, para seleccionar la más adecuada para iniciar el Estudio de Alternativas.

1.3 Tipología y prediseño

En este ejemplo se analiza únicamente el comportamiento de un dique de tipo mixto con berma baja de protección (Tipo C, LMB). Como punto de partida se consideran las siguientes dimensiones geométricas:

```
In [8]: img_name = os.path.join(env.input_path, 'obra', 'esquema_dique_c.png')
        Image(filename=img_name, width=500)
```

Out[8]:



- h_{BMVE}=16 m
- h_b=4 m
- B=12 m
- F_{MT}=6 m
- B_b=3 m
- F_c=6 m
- $cot\alpha_T$ =1.5
- D=0.65 m

v_verificacion_alineacion_principal_01_ep

April 27, 2018

0.0.1 Verificacion de la alineacion principal del dique

Este ejemplo continuará a partir de los estados caracerísticos de altura de ola significantes H_s guardados al final del apartado Distribución de extremos de la variable principal y obtención de su valor para un determinado periodo de retorno para un periodo de retorno $T_r = 200$ años y los valores correspondientes de periodo pico obtenidos en el apartado Funciones de regresion para la caracterizacion de las variables no predominantes.

Realizando el mismo procediemiento presentado en Funciones de regresion para la caracterizacion de las variables no predominantes para el periodo pico, se obtienen los valores correspondientes a la dirección media del oleaje θ , velocidad de viento U , dirección media del viento θ_U , nivel de mar η_{Mme}, η_{Mas} para un periodo de retorno $T_r = 200$ años y los intervalos de confianza.

En este ejemplo se realiza la verificacion de la fiabilidad de la alineacion principal de un dique de abrigo tipo mixto con berma baja de proteccion (Tipo C, LMB).

Las funciones que se utilizan son:

- verificacion.verificacion_mf.verificacion_simultanea

```
In [2]: # imports Anaconda
        from __future__ import division
        import os
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from IPython.display import HTML

        # imports ROM 1.1
        from verificacion.verificacion_mf import verificacion_simultanea

        dir_data = os.path.join(env.data_path, 'clima')
        dir_output_ver = os.path.join(env.output_path, 'verificacion')

        df_hs_tp = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__Hs_Tp_tr.pkl'))
        df_hs_dh = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__Hs_Dh_tr.pkl'))
        df_hs_vv = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__Hs_U_tr.pkl'))
        df_hs_dv = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__Hs_Du_tr.pkl'))
        df_hs_mme = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__Hs_Mme_tr.pkl'))
        df_hs_mas = pd.read_pickle(os.path.join(dir_data, 'out_reg_extremal__Hs_Mas_tr.pkl'))
```

Según el reparto de probabilidad de fallo considerado y una vida útil de la obra de $V = 25$ años, se obtienen parejas de balores (T_R, β). Seleccionado un periodo de retorno de 200 anos, el valor de β en las ecuaciones de verificacion debe ser mayor que 0.58.

Se seleccionan los valores de los estados caracteristicos para $T_r = 200$ años:

```
In [3]: tr = 200
        pos_tr = df_hs_tp['Tr']==tr
```

```
        hs, tp, dh = df_hs_tp.loc[pos_tr, 'Hs'].values[0], df_hs_tp.loc[pos_tr,
        'Tp'].values[0],\
                    df_hs_dh.loc[pos_tr, 'Dh'].values[0]
        vv, dv = df_hs_vv.loc[pos_tr, 'U'].values[0], df_hs_dv.loc[pos_tr, 'Du'].values[0]
        mme, mas = df_hs_mme.loc[pos_tr, 'Mme'].values[0], df_hs_mas.loc[pos_tr,
        'Mmas'].values[0]

        estados = pd.DataFrame({'hs' : np.repeat(hs, 3), 'tp' : np.repeat(tp, 3), 'dh' :
        np.repeat(dh, 3), \
                                'vv' : np.repeat(vv, 3),'dv' : np.repeat(dv, 3), 'marea': [-mas
        + mme, mme, mas + mme]})
```

```
In [4]: estados
```

Out[4]:

	dh	dv	hs	marea	tp	vv
0	263.808	274.238777	4.709996	-1.014432	17.770473	21.457686
1	263.808	274.238777	4.709996	0.572667	17.770473	21.457686
2	263.808	274.238777	4.709996	2.159766	17.770473	21.457686

Entradas de usuario - datos de partida y parámetros de diseño del dique:

```
In [5]: """ REQUISITOS DE PROYECTO """
        vida = 25
        pc_f = 0.1
        beta_min = 1.8
        nivel_averia = 'IA'

        """ DATOS DE PARTIDA """
        # Caracteristicas del emplazamiento
        profundidad = 17.75

        """ CARACTERISTICAS DEL DIQUE """
        tipo = 'C1'
        par_dique = {}
        par_dique['b'] = 12
        par_dique['d'] = 0.25
        par_dique['d_mean'] = 0.25
        par_dique['d_var'] = 0.05
        par_dique['dens_rel'] = 1.7
        par_dique['h_b'] = 4

        par_dique['f_mt'] = 6
        par_dique['b_b'] = 3
        par_dique['alpha_t'] = 1.5

        par_dique['d_b'] = 0.65
        par_dique['d_b_mean'] = 0.65
        par_dique['d_b_var'] = 0.05
        par_dique['pieza'] = True
        par_dique['ns'] = 0.4

        # Dique talud
        par_dique['tipo_pieza'] = 'E' # tipo pieza: E: escollera; C: cubo; T:tetrapodo

        # Rozamiento
        par_dique['mu_mean'] = 0.6
        par_dique['mu_var'] = 0.02

        lista_modos_fallo = ['deslizamiento', 'vuelco', 'estabilidad_berma']
```

El comportamiento hidraulico de la seccion en los estados analizados se caracteriza mediante el coeficiente de reflexion K_R, el coeficiente de transmision K_T, el coeficiente de disipacion D^* y la fase de la reflexion ϕ. Estos dependen de las caracteristicas de los agentes y del parametro

de dispersion A_{eq}/L^2 donde A_{eq} es el area equivalente de la seccion, correspondiente al area de material poroso por debajo del nivel del mar de referencia.

Para cada modo y estado se devuelve informacion sobre el valor medio del margen de seguridad (μ_S), su desviacion tipica (σ_S) y el coeficiente β que expresa el numero de desviaciones tipicas que separa el valor medio del margen de seguridad de la situacion de fallo $S = 0$.

```
In [6]: # Variables derivadas
        par_dique['a_eq_b'] = par_dique['b_b'] * par_dique['f_mt'] + 0.5 * par_dique['f_mt'] *
        par_dique['alpha_t'] * \
                                                                    par_dique['f_mt']
        par_dique['a_eq'] = par_dique['a_eq_b'] + par_dique['b'] * par_dique['h_b']

        par_dique['x4'] = par_dique['b_b'] + par_dique['alpha_t'] * par_dique['f_mt']
        par_dique['x2'] = par_dique['x4'] / 3
        par_dique['x3'] = par_dique['x4'] * 2 / 3


        modos_fallo = []
        info_mf = []

        list_df_comp, list_df_fallo = list(), list()
        for ei2, estado in estados.iterrows():
            par_dique['f_c'] = 6 - estado['marea']

            df_modos_fallo, d_info_mf = verificacion_simultanea(estado, lista_modos_fallo,
        profundidad, tipo, par_dique,
                                                            nivel_averia)
            # Comportamiento hidráulico
            df_out = pd.DataFrame({'h': d_info_mf['prof'], 'k_r':d_info_mf['k_r'],
        'k_t':d_info_mf['k_t'], 'dis': d_info_mf['dis'], 'phi': d_info_mf['phi']}, index=[ei2])
            list_df_comp.append(df_out)

            # Modos de fallo
            df_out_ver = pd.DataFrame({'h': d_info_mf['prof'], 'beta':df_modos_fallo['beta'],
        's_mean':df_modos_fallo['mean'], 's_dest': df_modos_fallo['dest'], 'estado':ei2})
            list_df_fallo.append(df_out_ver)

        # Guardar html y df
        df_outf_compo = pd.concat(list_df_comp)
        r_ds = os.path.join(env.output_path, 'verificacion', 'salida_comportamiento__' + tipo +
        '.html')
        df_outf_compo.to_html(r_ds, sparsify=False)
        df_outf_compo.to_pickle(os.path.join(env.output_path, 'verificacion',
        'salida_comportamiento__' + tipo + '.pkl'))

        # Guardar html y df
        df_outf_ver = pd.concat(list_df_fallo)
        r_ds = os.path.join(env.output_path, 'verificacion', 'salida_verificacion__' + tipo +
        '.html')
        df_outf_ver.to_html(r_ds, sparsify=False)
        df_outf_ver.to_pickle(os.path.join(env.output_path, 'verificacion',
        'salida_verificacion__' + tipo + '.pkl'))
```

Comportamiento hidráulico

```
In [8]: # Mostrar en pantalla
        r_ds = os.path.join(env.output_path, 'verificacion', 'salida_comportamiento__' + tipo +
        '.html')
        HTML(filename=r_ds)
```

```
Out[8]: <IPython.core.display.HTML object>
```

Verificación de modos de fallo

In [9]:
```
r_ds1 = os.path.join(env.output_path, 'verificacion', 'salida_verificacion__' + tipo +
'.html')
HTML(filename=r_ds1)
```

Out[9]: <IPython.core.display.HTML object>

co_costes_preliminares_01_ep

April 27, 2018

1 Ejemplo para el calculo de costes de construccion en el alcance de estudios previos

1.1 NOTA IMPORTANTE PARA LA EJECUCIÓN DE LOS EJEMPLOS

La carpeta nb contiene los ficheros notebooks con diferentes ejemplos de aplicación de la ROM 1.1, así como dos ficheros de configuración **`environment.py`** y **`modules_path.py`**. El fichero **`modules_path.py`** debe ser abierto por cada usuario mediante un editor de texto cualquiera para modificar la ruta a la carpera en la que se encuentren los códigos de los módulos de la ROM en nuestro ordenador.

A continuación se muestra un ejemplo: > Mac/Linux: > > > `modules_path = '/Users/user/Development/ROM_1_1/src'` > > > Windows (es importante separar cada carpeta mediante doble barra): > > "' > modules_path = 'C:\Development\ROM_1_1\src'

1.2 Importación de paquetes de IPython

```
In [2]: # Python 2/3 setup
        from __future__ import (absolute_import, division, print_function, unicode_literals)
        # from builtins import *

In [3]: # Juypyter setup

        %matplotlib inline
        import os
        import sys
        from IPython.display import HTML

In [4]: sys.path

Out[4]: ['',
         'D:\\REPOSITORIO GIT\\clima_maritimo',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\python27.zip',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\DLLs',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\plat-win',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\lib-tk',
         'C:\\Users\\GDFA-JUAN\\Anaconda2',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\site-packages',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\site-packages\\Babel-2.5.0-py2.7.egg',
         'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\site-packages\\win32',
```

```
'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\site-packages\\win32\\lib',
'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\site-packages\\Pythonwin',
'C:\\Users\\GDFA-JUAN\\Anaconda2\\lib\\site-packages\\IPython\\extensions',
'C:\\Users\\GDFA-JUAN\\.ipython',
'D:\\REPOSITORIO GIT\\modelo_construccion_2018\\construccion']
```

1.3 Importación de paquetes para el ejemplo

```
In [5]: import logging
        import os

        from construccion.simulacion_estudio_previo import simulacion_estudio_previo
        from construccion.utils_notebook import lectura_fichero_entrada
```

1.4 Datos de entrada

1.4.1 RUTA DE DIRECTORIOS DE DATOS DE ENTRADA Y SALIDA

```
In [6]: # Ruta con los datos de entrada
        ruta_de = os.path.join(env.input_path, 'construccion', 'estudios_previos')
        # Ruta con los datos de salida
        ruta_ds = os.path.join(env.output_path, 'construccion', 'estudios_previos')

        alcance = 'EP'
```

1.4.2 INTRODUCCIÓN

De acuerdo con el documento del articulado de la ROM 1.1., en el alcance de estudio previos se debe realizar una descripción preliminar del proceso constructivo, los medios necesarios y la disponibilidad de materiales. En lo relativo al cálculo de costes se deben evaluar al menos, los costes de construcción en función de la tipología y dimensiones principales de la alineación principal. Según el articulado, los costes totales del dique se podrán obtener de manera simplificada para este alcance mayorando los costes de inversión inicial por dos coeficientes: uno para ponderar la rigurosidad del clima y otro coeficiente mayor que la unidadad para incluir los costes de reparación, desmantelamiento, etc.

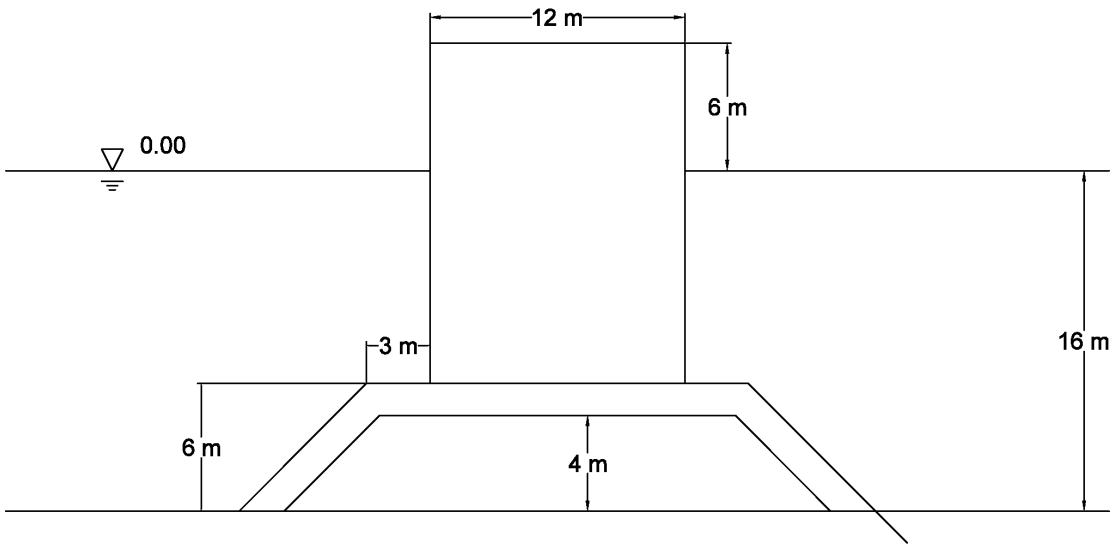
En este ejemplo se presenta el cálculo de los costes de construcción y su ponderación para la consideración de los costes totales únicamente para la alineación principal de una tipología de dique. Para el cálculo de costes en el resto de tipologías y de tramos se deberá proceder del mismo modo.

1.4.3 DEFINICIÓN DE LA TIPOLOGÍA Y FORMA EN PLANTA

En este ejemplo se propone analizar el coste de un dique de tipo mixto con berma baja de protección (Tipo C, LMB). La sección tipo del dique a analizar se muestra en la figura. En este ejemplo, se considera para la forma en planta un dique compuesto por una única alineación principal de 1000 m de longitud. El calado de la alineación principal es de 16 m respecto al nivel medio del mar, la altura de la berma es de 4 m y los cajones tienen una altura de 18 m por 12 m de ancho.

```
In [7]: from IPython.display import Image
        img_name = os.path.join(ruta_de, 'imagenes', 'esquema_dique_c.png')
        Image(filename=img_name, width=800)
```

Out[7]:



Fichero con los datos de entrada de tipología y forma en planta (Para modificar los datos utilizados en el ejemplo se deben modificar los ficheros de la carpeta de inputs)

```
In [8]: r_de = os.path.join(ruta_de, 'planta', 'datos_entrada_planta.txt')
        content = lectura_fichero_entrada(r_de, 2)
        HTML(content)
```

```
Out[8]: <IPython.core.display.HTML object>
```

1.4.4 BASE DE DATOS DE PRECIOS DE UNIDADES DE OBRA EMPLEADAS EN CONSTRUCCIÓN

Para el alcance de Estudios Previos, el calculo de los costes de construccion de la alineación principal del dique se realiza a partir de la base de datos contenida en el documento Bases de datos extraída a partir de El Observatorio de Obras Portuarias de Puertos del Estado con fecha de 2013. La utilidad de esta base de datos estriba en posibilitar la estimación del coste de una obra marítima a efectos de su planificación inversora de una forma simple. Este observatorio se compone de la siguiente base de datos: una base de datos de los precios unitarios de las unidades de obra más habituales en obras marítimas obtenidos a partir de proyectos redactados entre 2006 y 2013.

1.4.5 DESCRIPCIÓN DEL PROCESO CONSTRUCTIVO. UNIDADES DE OBRA Y MEDICIONES

Para la alineación principal del dique es necesario definir de forma preliminar la descripción del proceso constructivo a través de: (i) las unidades de obra a emplear durante la construcción del dique y (ii) las áreas equivalentes de las distintas partes del dique para la obtención de las mediciones de cada unidad de obra.

Se enumeran a continuación las unidades de obra, su descripción, área equivalente, medición y unidades a emplear en la construcción de la alineación principal del dique: * **Dragado** * 01-001: Dragado general en terreno suelto, carga, transporte y vertido. Medición a emplear 189000 m^3. *

Cimentación y berma baja de protección * 03-004: Todo uno de cantera, colocado en formación de núcleo de banqueta de cimentación, medido en báscula y ejecutado por medios marítimos. Área equivalente: 81.6 m^2. Medición: 81600 m^3. * 02-007: Escollera clasificada de peso entre 1000-2000 kg, suministro, transporte y colocación en manto y/o filtros por medios marítimos o terrestres. Área equivalente: 80.4 m^2. Medición: 80400 m^3. * 06-001: Enrase con grava en banqueta de cimentación de cajones o muros de muelle por medios marítimos. Medición 25600 m^3 * **Cajones** * 17-001: Botadura, transporte y fondeo de cajones, incluso fondeos intermedios provisionales. Medición 40 cajones de 25 m de eslora * 04-006: Relleno granular en celdas de cajones, incluso suministros, transporte y vertido, totalmente acabado procedente de dragado. Medición: 233000 m^3 * 04-007: Relleno granular en celdas de cajones, incluso suministros, transporte y vertido, totalmente acabado de procedencia terrestre. Medición: 233000 m^3 * **Superestructura** * 04-005: Relleno seleccionado en coronación de cajones, extendido y compactado de procedencia terrestre. Medición: 11250 m^3 * 13-002: Hormigón para armar HA-35 colocado en cajones, incluso encofrado, desencofrado, vibrado y curado. Medición: 83350 m^3 * 13-004: Hormigón para armar HA-30 colocado en superestructura, incluso encofrado, desencofrado, vibrado y curado. Medición: 4000 m^3 * 13-006: Hormigón para armar HA-30 colocado en alzados o espaldones, incluso encofrado, desencofrado, vibrado y curado.Medición: 39000 m^3 * 16-002: Aceros en redondos B500S para armado de cajones de hormigón, incluso cortado, doblado, p.p. de recortes y ataduras. Medición: 8335000 kg * 16-003: Aceros en redondos B500S para armado en superestructura, incluso cortado, doblado, p.p. de recortes y ataduras. Medición: 360000 kg

Fichero con los datos de entrada de la descripción del proceso constructivo. Unidades de obra y mediciones (Para modificar los datos utilizados en el ejemplo se deben modificar los ficheros de la carpeta de datos de entrada)

```
In [9]: r_de = os.path.join(ruta_de, 'tramos', 'T_0', 'unidades_obra', 'unidades_obra.txt')
        content = lectura_fichero_entrada(r_de, 20)
        HTML(content)
```

```
Out[9]: <IPython.core.display.HTML object>
```

1.4.6 COEFICIENTES DE MAYORACION DE CLIMA Y COSTES TOTALES

Para el calculo de los costes totales de la alineación principal del dique en el alcance de estudios previos se consideran dos coeficientes para la mayoración de los costes de construcción obtenidos: * Coeficiente de mayoración en función de la rigurosidad del clima en la zona. * Coeficiente de mayoración de los costes de construcción a fin de incluir de forma preliminar el resto de costes (reparación, desmantelamiento, etc.).

En este ejemplo se considera un coeficiente de mayoración por el clima de 1.4 y un coeficiente de mayoración para la obtención del coste total de 1.6.

Fichero con los datos de entrada de los coeficientes de mayoración (Para modificar los datos utilizados en el ejemplo se deben modificar los ficheros de la carpeta de datos de entrada)

```
In [10]: r_de = os.path.join(ruta_de, 'coeficientes', 'coeficientes.txt')
         content = lectura_fichero_entrada(r_de, 2)
         HTML(content)
```

```
Out[10]: <IPython.core.display.HTML object>
```

1.5 Cálculo del coste preliminar de construcción de la alineación principal (Estudios Previos)

Para la obtención del coste preliminar de construcción de la alineación principal del dique, la herramienta obtiene los costes por unidad de medición de la base de datos de construcción y multiplica el coste por unidad de medición de las unidades de obra seleccionadas por la medición introducida.

```
In [11]: (datos_salida) = simulacion_estudio_previo(ruta_de, alcance, ruta_ds)
```

1.6 Datos de salida

```
In [12]: r_ds = os.path.join(ruta_ds, 'Coste_total_dique.html')
         HTML(filename=r_ds)
```

```
Out[12]: <IPython.core.display.HTML object>
```

co_costes_construccion_01_es_c1

April 27, 2018

1 Ejemplo para el cálculo de costes de construccion en el alcance de estudios soluciones clase I (Estrategia protectora avance en paralelo): Parte 1 - Lectura de los datos de entrada de la forma en planta.

1.1 Importación de paquetes de IPython

```
In [2]: # Python 2/3 setup
        from __future__ import (absolute_import, division, print_function, unicode_literals)
        # from builtins import *

In [3]: # Juypyter setup

        %matplotlib inline
        import os
        import sys
        from IPython.display import HTML
```

1.2 Importación de paquetes para el ejemplo

```
In [4]: import logging
        import os
        import pickle

        from construccion.datos_entrada import datos_entrada_planta

        from construccion.main_tramos import simulacion_proceso_constructivo_tramo

        from construccion.representacion_v2 import representacion_resultados_tiempo
        from construccion.representacion import representacion_resultados_costes

        from construccion.calculos import calculo_costes
        from construccion.calculos import calculo_longitudes_volumenes_acumulados
        from construccion.calculos import extraccion_resultados

        from construccion.simulacion_estudio_previo import simulacion_estudio_previo

        from construccion.utils_notebook import lectura_fichero_entrada
```

1.3 Datos de entrada de la forma en planta

1.3.1 RUTA DE DIRECTORIOS DE DATOS DE ENTRADA Y SALIDA

```
In [5]: # Ruta con los datos de entrada
        ruta_de = os.path.join(env.input_path, 'construccion', 'estudio_soluciones_clase1')
        # Ruta con los datos de salida
        ruta_ds = os.path.join(env.output_path, 'construccion', 'estudio_soluciones_clase1')
```

1.3.2 INTRODUCCION

De acuerdo con el documento del articulado de la ROM 1.1., en el alcance de estudio de soluciones se deben analizar los procesos y medios construcctivos, sus condicionantes y estimar los plazos de ejecución. Además se deben evaluar los costes más probables de construcción considerando estrategias simples de construcción. Para ella será necesario analizar la interacción de la obra con las condiciones climáticas propagadas a cada uno de los tramos del dique.

En este alcance (estudio de soluciones clase I) esta herramienta permite verificar el proceso construcctivo del dique mediante simulación numérica. Durante esta simulación la herramienta permite obtener los tiempos y costes de construcción en función de la tipología, dimensiones, rendimiento de los trabajos, medios constructivos y estrategia de avance

1.3.3 DEFINICIÓN DEL ALCANCE Y ESTRATEGIA

Se define en el ejemplo que el alcance es el de estudio de soluciones de clase I. En este alcance, la herramienta considera que los daños sufridos en cada estado son reparados de manera inmediata en el estado siguiente. Y finalmente se considera una estrategia de avance en paralelo en la que todas las subfases avanzan al mismo tiempo protegiendose unas a otra con sus avances. Sin embargo, se obliga a que el avance de una subfase nunca pueda ser superior al de las subfases siguientes.

En este alcance se pueden verificar tres tipos de estrategias: - Avance en serie: En dónde cada subafse comienza en el momento en el que finaliza la subfase anterior. Esta estrategia es la que maximiza el tiempo de ejecución y las pérdidas puesto que las subfases están expuestas ante las inclemencias climáticas. - Avance en paralelo: En dónde todas las subfases comienzan a la vez y por lo tanto se produce un avance en paralelo. Esta estrategia es la que minimiza el tiempo de ejecución y las pérdidas puesto que cada subfase se protege con el avance de las siguientes. - Cronograma de los trabajos: En dónde se define mediante un cronograma de los trabajos el inicio y el final teórico de cada una de las subfases constructivas. Este cronograma se introduce en la herramienta mediante una matriz en la que el eje X representa cada una de las subfases del tramo y el eje Y las horas desde la hora 0 de inicio de la obra

```
In [6]: alcance = 'EA'
        estrategia = 'avance_paralelo'
        rep_inmediata = 'si'
```

1.3.4 RUTA PARA LA CREACIÓN DEL FICHERO DE LOG

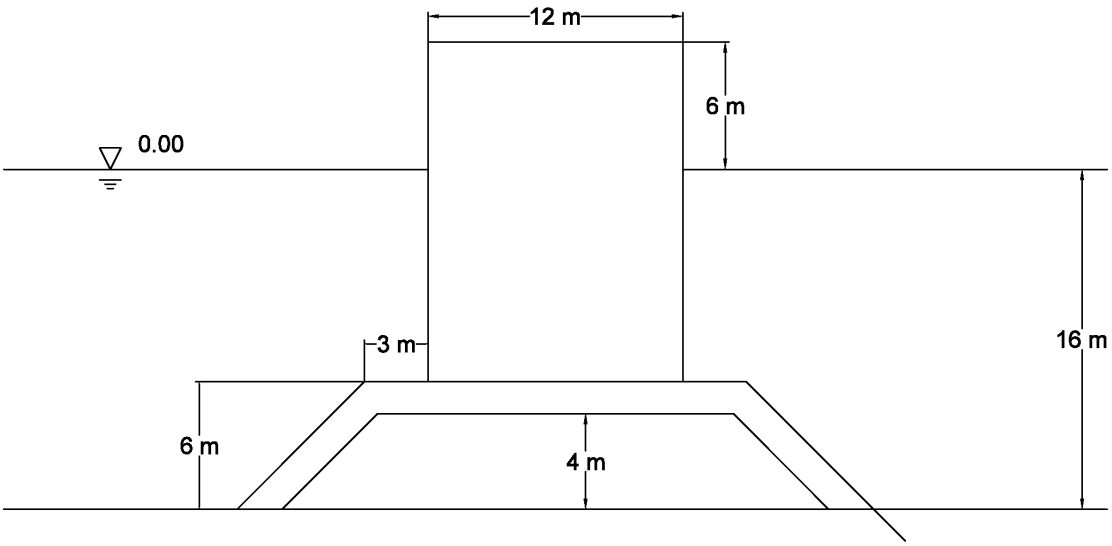
```
In [7]: direct = os.path.join(ruta_ds, 'debug_info.log')
        logging.basicConfig(filename=direct, level=logging.INFO)
```

1.3.5 DEFINICIÓN DE LA TIPOLOGÍA Y FORMA EN PLANTA

En este ejemplo se propone analizar el coste de un dique de tipo mixto con berma baja de protección (Tipo C, LMB). La sección tipo del dique a analizar se muestra en la figura. En este ejemplo, se considera para la forma en planta un dique compuesto por una única alineación principal de 500 m de longitud. El calado de la alineación principal es de 16 m respecto al nivel medio del mar, la altura de la berma es de 4 m y los cajones tienen una altura de 18 m por 12 m de ancho.

```
In [8]: from IPython.display import Image
        img_name = os.path.join(ruta_de, 'imagenes', 'esquema_dique_c.png')
        Image(filename=img_name, width=800)
```

Out[8]:



Lectura de los ficheros con los datos de entrada de tipología y forma en planta (Para modificar los datos utilizados en el ejemplo se deben modificar los ficheros de la carpeta de inputs)

In [13]:
```
# Lectura de los datos de entrada de la planta
(de_planta, p_invernal) = datos_entrada_planta(ruta_de, alcance, estrategia)

display(de_planta)
```

Out[13]:

	longitud	tipologia	calado
T_0	500.0	'mixto'	16.0

Cálculo del numero de tramos del dique En función del fichero de forma en planta introducido

In [10]:
```
# Calculo del numero de tramos del dique
n_tramos = de_planta.shape[0]
n_tramos
```

Out[10]: 1

Inicialización de las horas de inicio para la verificación del proceso constructivo

In [11]:
```
# Inicializacion de la variable hora que avanza con cada iteracion
hora_acumulada = 0

# Inicializacion de la hora de inicio del tramo 0
hora_inicio_tramos = [0]
```

Guardado de variables

```
In [12]: # Saving the objects:
         dir_data = os.path.join(env.data_path, 'construccion', 'estudio_soluciones_clase1',
         'var_co_costes_construccion_01_es_c1.pkl')

         with open(dir_data, 'w') as f:  # Python 3: open(..., 'wb')
             pickle.dump([n_tramos, de_planta, hora_acumulada, hora_inicio_tramos,
                          p_invernal, alcance, estrategia, ruta_de, ruta_ds, rep_inmediata], f)
```

Continuar con Parte 2 - Lectura de los datos de entrada asociados a la descripción del proceso constructivo..

co_costes_construccion_02_es_c1

April 27, 2018

1 Ejemplo para el cálculo de costes de construccion en el alcance de estudios soluciones clase I (Estrategia protectora avance en paralelo): Parte 2 - Lectura de los datos de entrada asociados a la descripción del proceso constructivo.

Este ejemplo continuará a partir de los datos temporales guardados al final del apartado Parte 1 - Lectura de los datos de entrada de la forma en planta.

De acuerdo con el documento Articulado de la ROM 1.1 la descripción del proceso constructivo deberá incluir: la disponibilidad de medios para ejecutar los trabajos necesarios, la disponibilidad de materiales y una organización del proceso constructivo mediante un organigrama con la secuencia de los trabajos. Para la evaluación del proceso constructivo del dique se deberán enfrentar las diferentes estrategias a las condiciones climáticas propagadas a cada uno de los tramos del dique. Se deberá describir la organización del proceso constructivo en los niveles jerárquicos: dique, tramos, subfases constructivas y unidades de obra.

1.1 Importación de paquetes de IPython

```
In [2]: # Python 2/3 setup
        from __future__ import (absolute_import, division, print_function, unicode_literals)
        # from builtins import *
```

```
In [3]: # Juypyter setup

        %matplotlib inline
        import os
        import sys
        import pickle
        import ast
        import pandas as pd
        env.pandas_setup()
        from IPython.display import HTML
```

1.2 Carga de variables del apartado anterior

```
In [4]: # Getting back the objects:
        dir_data = os.path.join(env.data_path, 'construccion', 'estudio_soluciones_clase1',
        'var_co_costes_construccion_01_es_c1.pkl')

        with open(dir_data) as f:  # Python 3: open(..., 'rb')
            n_tramos, de_planta, hora_acumulada, hora_inicio_tramos, p_invernal, alcance,
        estrategia, ruta_de, ruta_ds, rep_inmediata = pickle.load(f)
```

1.3 Importación de paquetes para el ejemplo

```
In [5]: from construccion.utils_notebook import lectura_fichero_entrada

        from construccion.datos_entrada import datos_entrada_tramo
        from construccion.datos_entrada import datos_entrada_fase
        from construccion.datos_entrada import rendimientos_y_umbrales_datos_entrada
```

1.4 Iteración sobre cada tramo

Llegados a este punto la herramienta recorre uno a uno cada uno de los tramo del dique. En primer lugar se leen los datos de entrada asociados a cada tramo para posteriomente verificar el proceso constructivo del mismo. En este ejemplo se considerará un único tramo del dique.

```
In [6]: # Se asocia la etiqueta del tramo
        n = 'T_0'
```

1.5 Lectura de los ficheros

En primer lugar se leen los ficheros con los datos de entrada necesarios para la descripción del proceso constructivo y posteriormente se pasa a explicar cada uno de ellos.

De acuerdo con el Documento del Articulado de la ROM 1.1 los datos de entrada necesarios relativos a cada tramo del dique deberán incluir: una descripción del proceso constructivo, la disponibilidad de medios para ejecutar los trabajos necesarios, la disponibilidad de materiales y una organización del proceso constructivo mediante un organigrama con la secuencia de los trabajos

```
In [7]: # Lectura de los datos de entrada del tramo a partir de ficheros (clima, organizacion
        proceso constructivo,
        # subfases constructivas etc)
        (de_tramo, plan_avance, clima, com_fin_teorico) = datos_entrada_tramo(de_planta, n,
        hora_inicio_tramos, alcance, estrategia, ruta_de, rep_inmediata)

        # Lectura de los datos de entrada de cada subfase: unds de obra, maquinaria, ecuaciones
        de danno, ecs de coste
        # y restricciones
        (maquinaria_fases, eq_danno_fases, eq_coste_fases, costes_fase, restricciones_fases) =
        datos_entrada_fase(de_planta, de_tramo, n, estrategia, alcance, ruta_de)
```

```
D:\REPOSITORIO
GIT\modelo_construccion_2018\construccion\construccion\ecuaciones_danno.py:38:
RuntimeWarning: invalid value encountered in double_scalars
  m = (d_max - d_0) / (x_max - x_0)
D:\REPOSITORIO
GIT\modelo_construccion_2018\construccion\construccion\ecuaciones_danno.py:77:
RuntimeWarning: invalid value encountered in double_scalars
  m = (c_max - c_min) / (d_max - 0)
```

1.6 Descripción de los datos de entrada

Los datos de entrada relacionados con la descripción del proceso constructivo se agrupan en: - Datos relativos a los parámetros constructivos de las diferentes subfases de la obra - Datos relativos a la organización del proceso constructivo de la obra mediante el cronograma con las fecha de inicio y finalización de los trabajos. - Un tercer grupo con los datos climáticos propagados al tramo del dique.

1.6.1 Datos climáticos

Dentro de la carpeta 'clima' es necesario que haya una carpeta por cada uno de los tramos que componen el dique con el nombre de la etiqueta del tramo (en este caso únicamente uno). Dentro de cada carpeta se debe encontrar un fichero denominado 'clima.txt' con las series temporales históricas propagadas al emplazamiento de cada uno de los agentes necesarios para realizar la verificación del dique. Las series temporales se estructurarán por columnas.

La matriz con los datos de clima tendrán un valor de nivel (respecto al NMM), velocidad de viento, altura de ola, periodo pico, dirección media de procedendia del oleaje y dirección media de procedencia del viento para cada hora.

Se muestra a continuación un ejemplo del fichero de clima:

```
In [8]: # Cambio el numero de filas y columnas que se muestran
        env.pandas_setup(10, 50)
        display(clima)
        # Cambio el numero de filas y columnas a sus valores por defecto
        env.pandas_setup()
```

	eta	vv	hs	calado
0	0.930918	1.041786	1.041786	16.930918
1	1.337872	1.060341	1.060341	17.337872
2	1.388591	1.078959	1.078959	17.388591
3	1.096256	1.097630	1.097630	17.096256
4	0.467977	1.113442	1.113442	16.467977
...	...	...	...	...
175316	0.138936	0.324749	0.324749	16.138936
175317	0.541487	0.324876	0.324876	16.541487
175318	0.882342	0.325003	0.325003	16.882342
175319	0.935548	0.325129	0.325129	16.935548
175320	0.739374	0.325256	0.325256	16.739374

[175321 rows x 4 columns]

1.6.2 Organización del proceso constructivo (Inicio y final teórico de los trabajos)

Se debe introducir el cronograma con los inicios y finales de cada una subfases constructivas. La estructura del fichero será la siguiente: Se debe introducir una matriz con tantas columnas como subfases constructivas se empleen para la construcción del tramo y se deseen verificar. Y tantas filas como horas dure la fase de construcción del tramo. Para cada subfase (columnas) se indicará con un 1 las horas en las que la subfase trabaja para ejecutar las obras del tramo y con un 0 las horas en las que no trabaja. Este fichero se introducirá dentro de la carpeta de organización del proceso constructivo con el nombre de plan_de_avance.txt

Se muestra a continuación un ejemplo del fichero:

```
In [9]: r_de = os.path.join(ruta_de, 'organizacion_proceso_constructivo', 'plan_avance.txt')
        data = pd.read_table(r_de, sep=',', quoting=2)
        data
```

```
Out[9]:        T_0\tT_0\tT_0\tT_0\tT_0\tT_0\tT_0\tT_0\tT_0\tT_0\tT_0
        0                          1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0
```

```
1                       1\t0\t0\t0\t0\t0\t0\t0\t0\t0
2                       1\t0\t0\t0\t0\t0\t0\t0\t0\t0
3                       1\t0\t0\t0\t0\t0\t0\t0\t0\t0
4                       1\t0\t0\t0\t0\t0\t0\t0\t0\t0
...                                               ...
21165                   0\t0\t0\t0\t0\t0\t0\t0\t0\t0
21166                   0\t0\t0\t0\t0\t0\t0\t0\t0\t0
21167                   0\t0\t0\t0\t0\t0\t0\t0\t0\t0
21168                   0\t0\t0\t0\t0\t0\t0\t0\t0\t0
21169                   0\t0\t0\t0\t0\t0\t0\t0\t0\t0

[21170 rows x 1 columns]
```

1.6.3 Parámetros constructivos de las subfases

De acuerdo con el documento del Articulado de la ROM 1.1 se asignará a cada subfase la cantidad de obra a realizar (medición a ejecutar), los equipos de producción de las unidades de obra y el conjunto de modos principales de fallo en cada una de ellas. Para la descripción de cada unidad de maquinaria se considerarán: los tiempos de arranque, transporte y colocación de la maquinaria, el número de horas laborables al día, los rendimientos y los umbrales de operatividad.

Los datos de entrada para este ejemplo se han estructurado del siguiente modo, dentro de la carpeta 'subfases' es necesario que haya una carpeta por cada uno de los tramos que componen el dique con el nombre de la etiqueta del tramo.

La carpeta correspondiente al tramo deberá contener a su vez cuatro carpetas más un fichero denominado 'subfases.txt'. En total la estructura es: - Fichero 'subfases.txt' - Carpeta 'parámetros constructivos' - Carpeta 'maquinaria' - Carpeta 'modos de fallo' - Carpeta 'costes'

FICHERO SUBFASES Este fichero le indica a la herramienta las subfases constructivas empleadas para la construcción del tramo así como el orden de ejecución. En este ejemplo se considera que el tramo T_0 se construye mediante las siguientes subfases:

- Subfase 0: Dragado
- Subfase 1: Vertido del núcleo de la banqueta
- Subfase 2: Vertido del manto de protección de la banqueta
- Subfase 3: Extracción de material para el enrase de la banqueta
- Subfase 4: Adición de material para el enrase de la banqueta
- Subfase 5: Fabricación de los cajones
- Subfase 6: Preparación de los cajones para el remolque
- Subfase 7: Transporte de los cajones
- Subfase 8: Acopio de los cajones
- Subfase 9: Fondeo de los cajones
- Subfase 10: Relleno de celdas

Se muestra a continuación ejemplo del fichero con las subfases constructivas:

```
In [10]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)

         r_de = os.path.join(ruta_de, 'sub_fases', n, 'sub_fases.txt')
         data = pd.read_table(r_de, sep=',', quoting=2, header=None)
```

```
data

# Cambio el numero de filas y columnas a sus valores por defecto
env.pandas_setup()
```

CARPETA PARAMETRO CONSTRUCTIVOS Cada fichero, contendrá la etiqueta de la subfase constructiva, el volumen a ejecutar por la subfase, si dicha subfase es protegida por el avance de alguna de las subfases siguientes (1-afirmativo o 0-negativo), en caso afirmativo la subfase que la protege y la forma de calcular el rendimiento de la subfase a partir del rendimiento de las unidades de obra que conforman la subfase.

Se muestra a continuación a modo de ejemplo el fichero con los parámetros constructivos asociados a la subfase del dragado. Para mostrar cualquier otra subfase basta con cambiar el nombre en las lineas de abajo

```
In [11]: # Codigo para mostrar el fichero en formato html
         r_de = os.path.join(ruta_de, 'sub_fases', n, 'parametros_constructivos',
         '0_dragado.txt')
         content = lectura_fichero_entrada(r_de, 2)
         data = pd.read_table(r_de, sep=',', quoting=2)
         display(data)
```

```
            vol_subfase  proteccion_por_subfases_siguientes  \
0_dragado     189000.0                                 1.0

            subfase_protectora rend_uds_obra
0_dragado                  1.0           min
```

Para este ejemplo, se considera que se van a ejecutar 189.000 m3 de dragado a lo largo de los 500 m de longitud del tramo. Esta subfase se protege a medida que avanza la subfase 1: vertido del núcleo de la banqueta. Y el rendimiento de la subfase se obtiene como el mínimo de los rendimientos de la unidades de obra de que conforman la subfase.

CARPETA DE MAQUINARIA Cada fichero contendrá para cada tipo de maquinaria dentro de cada unidad de obra: el número de máquinas a emplear durante la ejecución de la subfase, el rendimiento unitario de cada máquina, el tiempo de arranque necesario, el número de horas laborables al día, el número de días laborables a la semana, el coste de la maquinaria unitario, el coste de la mano de obra unitario, el coste adicional por retrasos y el umbral de operatividad de la máquina.

Se muestra a continuación a modo de ejemplo el fichero con la maquinaria asociada a la subfase del dragado. Para mostrar cualquier otra subfase basta con cambiar el nombre en las lineas de abajo

```
In [12]: # Codigo para la lectura del fichero
         r_de = os.path.join(ruta_de, 'sub_fases', n, 'maquinaria', '0_dragado.txt')
         data = pd.read_table(r_de, sep=',', quoting=2)
         # Extraccion del diccionario interior
         data = data.loc['Dragado_med_hidra', 'maquinaria']
         # Conversion de string a diccionario
         maq = ast.literal_eval(data)
         maq
```

```
Out[12]: {'draga_Volvox_Iberia_succion_marcha': {'coste_add_unit': 200,
             'coste_mano_obra_unit': 700,
```

```
            'coste_maq_unit': 700,
            'dia_lab_sem': 6,
            'h_lab_dia': 24,
            'num': 1,
            'rend_unit': 306,
            't_arranque': 48,
            'umb_operatividad': {'calado': 5, 'hs': 2.5, 'nivel': 100, 'vv': 100}}}
```

Para este ejemplo se considera que el dragado se realiza mediante una única unidad de obra con una única draga Volvox Iberia con un rendimiento de 306 $m^3\,h^{-1}$, un tiempo de arranque de 48 h, que trabaja durante 24 h al día de forma continua durante 6 días a la semana. El coste de la maquinaria unitario es de 700 $e\,h^{-1}$, el coste de la mano de obra unitario es de 700 $e\,h^{-1}$ y el coste adicional por retrasos es de 200 $e\,h^{-1}$. Los agentes que limitan operatividad de la máquina son el oleaje y el calado. Se considera que la máquina no puede operar con altura de ola superiores a 2.5 m ni con calados inferiores a los 5 m. El resto de agentes no afectan al funcionamiento de la máquina y por eso se les pone un valor de 100 para que no sean considerados por la herramienta.

CARPETA MODOS DE FALLO Cada fichero contendrá los datos relativos al modo de fallo principal de la subfase constructiva. Los modos de fallo deberán referirse siempre a fallo traducibles en un daño en la sección transversal de la subfase del tramo debidos a un agente principal. Los parámetros a introducir son: etiqueta del modo de fallo principal, etiqueta de la variable responsable de producir el daño, forma de la ecuación de daño (lineal, exponencial, archivo), valor de x_0 (valor de la variable correspondiente con el inicio de avería de la subfase), valor de x_{max} (valor de la variable responsable de la pérdida total de la sección trasversal de la subfase), d_0 (valor del daño correspondiente al inicio de avería y a un valor de la variable igual a x_0), d_max (valor del daño correspondiente a la pérdida total de la sección transversal de la subfase y a un valor de la variable igual a x_{max}), nombre del archivo de la ecuación de daño, forma de la ecuación de coste debido a las pérdidas (lineal, exponencial o archivo), valor de c_{min} (valor del coste de las pérdidas asociado con el inicio de avería en la sección trasversal de la subfase), valor de c_{max} (valor del coste asociado a la pérdida total de la sección trasversal de la subfase) y nombre del archivo de la ecuación de coste.

Se muestra a continuación a modo de ejemplo el fichero asociada a la subfase del dragado. Para mostrar cualquier otra subfase basta con cambiar el nombre en las lineas de abajo

```
In [13]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)

         # Codigo para mostrar el fichero en formato html
         r_de = os.path.join(ruta_de, 'sub_fases', n, 'modos_fallo', '0_dragado.txt')
         content = lectura_fichero_entrada(r_de, 2)
         data = pd.read_table(r_de, sep=',', quoting=2)
         display(data)

         # Devuelvo el numero de filas y columnas por defecto
         env.pandas_setup()
```

	modo_fallo_ppal	agente	ecuacion_danno	x_0	x_max	d_0	d_max	\
0	aterramiento	hs	lineal	1.0	4.0	0.0	378.0	

	archivo_dannos	ecuacion_coste	c_min	c_max	archivo_costes
0	NaN	lineal	0.0	2290.0	NaN

Para este ejemplo se considera que el modo de fallo principal para la subfase de dragado es el aterramiento de la subfase. La variable del agente principal responsable de producir daño es 'hs' (la altura de ola significante). La forma de la ecuación de daño es lineal, en donde el valor de x_0 es igual a 1 m y el valor de x_max es igual a 4 m y los valores de d_0 y d_max son 0 y 378 $m^3\,m^{-1}$ respectivamente. El valor de 378 se ha obtenido dividiendo el volumen total de la subfase 189.000 m^3 por la longitud total del tramo, 500 m. De modo que altura de ola iguales o inferiores a 1 m, el daño producido en la subfase es igual a 0, a partir de 1 m hasta los 4 m, el daño crece de forma lineal hasta los 378 $m^3\,m^{-1}$ y el coste crece hasta los 2290 € m^{-1}.

CARPETA COSTES Cada fichero contendrá los datos de: costes materiales unitarios, costes de mantenimiento unitarios y costes indirectos relacionados con la subfase constructiva.

Se muestra a continuación a modo de ejemplo el fichero asociada a la subfase del dragado. Para mostrar cualquier otra subfase basta con cambiar el nombre en las lineas de abajo

```
In [14]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)

         # Codigo para mostrar el fichero en formato html
         r_de = os.path.join(ruta_de, 'sub_fases', n, 'costes', '0_dragado.txt')
         content = lectura_fichero_entrada(r_de, 2)
         data = pd.read_table(r_de, sep=',', quoting=2)
         display(data)

         # Devuelvo el numero de filas y columnas por defecto
         env.pandas_setup()
```

```
   costes_materiales_unit  coste_mantenimiento  coste_proteccion  \
0                     0.0                  0.0               0.0

   costes_indirectos
0                0.1
```

1.6.4 Agrupación de los parámetros constructivos de las subfases

```
In [15]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)


         # Calculo de los valores de rendimiento, n. de horas laborables al dia, n. de dias
         laborables a la semana, tiempos
         # de arranque y umbrales de operatividad a partir de los datos de las maquinarias de las
         unidades de obra y
         # asignacion al dataframe de datos del tramo
         de_tramo = rendimientos_y_umbrales_datos_entrada(de_tramo, maquinaria_fases, alcance)
         de_tramo

         # Cambio el numero de filas y columnas que se muestran a su valor por defecto
         env.pandas_setup()
```

1.7 Guardado de variables

```
In [16]: # Saving the objects:
         dir_data = os.path.join(env.data_path, 'construccion', 'estudio_soluciones_clase1',
         'var_co_costes_construccion_02_es_c1.pkl')

         with open(dir_data, 'w') as f:  # Python 3: open(..., 'wb')
             pickle.dump([n, de_tramo, plan_avance, clima, com_fin_teorico,
```

```
                maquinaria_fases, eq_danno_fases, eq_coste_fases, costes_fase,
restricciones_fases], f)
```

Continuar con Parte 3 - Verificación del proceso constructivo mediante simulación numérica.

co_costes_construccion_03_es_c1

April 27, 2018

1 Ejemplo para el cálculo de costes de construccion en el alcance de estudios soluciones clase I (Estrategia protectora avance en paralelo): Parte 3 - Verificación del proceso constructivo mediante simulación numérica.

Este ejemplo continuará a partir de los datos temporales guardados al final del apartado Parte 2 - Lectura de los datos de entrada asociados a la descripción del proceso constructivo

1.1 Importación de paquetes de IPython

In [2]:
```
# Python 2/3 setup
from __future__ import (absolute_import, division, print_function, unicode_literals)
# from builtins import *
```

In [3]:
```
# Juypyter setup

%matplotlib inline
import os
import sys
import pickle
import ast
import math
import pandas as pd
env.pandas_setup()
from IPython.display import HTML

import warnings
warnings.filterwarnings('ignore')
```

1.2 Carga de variables del apartado anterior

In [4]:
```
# Getting back the objects:
dir_data = os.path.join(env.data_path, 'construccion', 'estudio_soluciones_clase1',
'var_co_costes_construccion_01_es_c1.pkl')

with open(dir_data) as f:  # Python 3: open(..., 'rb')
    n_tramos, de_planta, hora_acumulada, hora_inicio_tramos, p_invernal, alcance,
estrategia, ruta_de, ruta_ds, rep_inmediata = pickle.load(f)

# Getting back the objects:
dir_data = os.path.join(env.data_path, 'construccion', 'estudio_soluciones_clase1',
'var_co_costes_construccion_02_es_c1.pkl')

with open(dir_data) as f:  # Python 3: open(..., 'rb')
    n, de_tramo, plan_avance, clima, com_fin_teorico, maquinaria_fases, eq_danno_fases,
eq_coste_fases, costes_fase, restricciones_fases = pickle.load(f)
```

1.3 Importación de paquetes para el ejemplo

In [5]:
```
import numpy as np

from construccion.calculos import calcula_n_horas_proteger
from construccion.calculos import actualiza_longitudes_fase_siguiente
from construccion.calculos import recorte_matrices_resultantes
from construccion.calculos import genera_matriz_volumen
from construccion.calculos import genera_matriz_estado
from construccion.calculos import genera_matriz_contador_proteccion_horas_fijas
from construccion.calculos import actualiza_matrices_fases_finalizadas

from construccion.clasificacion_main_fase_I import clasificacion_main_fase_I

from construccion.fases_nivel_III import fase_parada_invernal

from construccion.comprobaciones import comprueba_fase_acabada
from construccion.comprobaciones import comprueba_fase_finalizada

from construccion.calculos import calculo_costes
from construccion.calculos import calculo_longitudes_volumenes_acumulados
from construccion.calculos import extraccion_resultados

from construccion.representacion_v2 import representacion_resultados_tiempo


import logging
```

1.4 Inicialización de las variables necesarias para la verificación del proceso constructivo

In [6]:
```
    # Inicializacion del dataframe de longitudes
    columnas = ['l_avanzada', 'l_protegida', 'l_desprotegida']
    indices = range(plan_avance.shape[0])
    longitudes = pd.DataFrame(0, index=indices, columns=columnas)

    # Inicializacion del dataframe de volumen ejecutado
    vol_ejecutado = genera_matriz_volumen(plan_avance)
    # Inicializacion del dataframe de volumen perdido a causa de los temporales
    vol_perdido = genera_matriz_volumen(plan_avance)
    # Inicializacion del dataframe de avance real (estado de la fase a Nivel 2)
    avance_real = genera_matriz_estado(plan_avance)
    # Inicializacion del dataframe de estado real (estado de la fase a Nivel 3)
    estado_real = genera_matriz_estado(plan_avance)
    # Inicializacion a 0 de los contadores de proteccion para las subfases que tengan
horas de proteccion fijas
    cont_pro_h_fijas = genera_matriz_contador_proteccion_horas_fijas(plan_avance)

    # Inicializacion a 0 de los contadores de tiempo de arranque, hora, hora laborable y
dia laborable
    columnas = ['t_arranque']
    cont_t_arranque = pd.DataFrame(0, index=indices, columns=columnas)
    hora = 0
    hora_labor = 0
    dia_labor = 0

    # Inicializacion a False del datafrae de las sub fases constructivas finalizadas
(acabdas + protegidas)
    fases_finalizadas = pd.Series(np.full(plan_avance.shape[0], False, dtype=np.bool))

    # Inicializacion de todas las subfases del tramo para la simulacion
    fase_pos = fases_finalizadas[fases_finalizadas == False]  # nopep8
```

1.5 Inicialización del fichero de logging

```
In [7]: direct = os.path.join(ruta_ds, 'debug_info.log')
        logging.basicConfig(filename=direct, level=logging.INFO)
```

1.6 Verificacion del proceso constructivo

De acuerdo con el Articulado de la ROM 1.1, la verificación de la fase de construcción del dique se realiza en cada uno de los estados de que componen la vida útil. Se debe verificar de forma simultánea el avance de cada una de las subfases encargadas de ejecutar los distintos tramos de la traza de la obra. El esquema seguido para la verificación se muestran en el documento del Manual de la ROM 1.1. La verificación en este alcance se realiza una vez a partir de las series históricas de los agentes propagadas a cada uno de los tramo del dique.

```
In [8]:     # Inicio del proceso constructivo. Simula hasta que todas las fases se encuentren
        finalizadas
            while not fases_finalizadas.all():
                logging.info('Hora: ' + str(hora) + 'hs: ' + str(clima.loc[hora, 'hs']) + 'vv: '
        + str(clima.loc[hora, 'vv']) + 'eta: ' + str(clima.loc[hora, 'eta']) + 'calado: ' +
        str(clima.loc[hora, 'calado']))

                actualiza_matrices_fases_finalizadas(fases_finalizadas, estado_real, hora)

                # Para cada hora (varia con un contador) se recorren cada una de las subfases
        del tramo
                for fase, _ in fase_pos.iteritems():

                    # Se comprueba si esta hora se encuentra dentro de una parada invernal
                    if hora_acumulada in p_invernal:
                        (vol_ejecutado, longitudes) = fase_parada_invernal(fase, hora, de_tramo,
        longitudes, estado_real,
                                                                           vol_ejecutado,
        cont_t_arranque, avance_real,
                                                                           eq_danno_fases)

                    else:

                        # Para cada subfase, antes de comenzar la verificacion, se actualiza la
        longitud protegida en base a la
                        # longitud avanzada por la fase siguiente. Limitando que la longitud
        protegida nunca sea mayor que la
                        # avanzada.
                        longitudes = actualiza_longitudes_fase_siguiente(fase, longitudes,
        plan_avance, de_tramo)

                        # Se calcula para cada iteracion el numero de horas necesario para
        proteger la longitud desprotegida
                        # de la subfase. Este dato se utilizara para clasificar la fase en
        funcion de que haya o no presencia
                        # de temporal en las siguientes 'n_horas_fase' horas
                        n_horas_fase = int(math.ceil(calcula_n_horas_proteger(de_tramo,
        longitudes, fase)))

                        # Se inicia la verificacion: Para cada subfase en cada hora se obtiene
        un valor del volumen ejecutado,
                        # el volumen perdido, la actualizacion de las longitudes avanzadaas,
        protegidas y desprotegidas y
                        # la actualizacion de las matrices de estado y avance real.
                        (vol_ejecutado, vol_perdido, longitudes,
                         cont_pro_h_fijas) = clasificacion_main_fase_I(fase, hora, clima,
        de_tramo, avance_real,
                                                                      vol_ejecutado,
        longitudes, plan_avance,
                                                                      n_horas_fase,
```

```
estado_real, hora_labor,
                                                                cont_t_arranque,
hora_acumulada,
                                                                com_fin_teorico,
eq_danno_fases, vol_perdido,
                                                                restricciones_fases,
dia_labor, cont_pro_h_fijas)

                # Tras la finalizacion de la simulacion de la subfase para el estado de
mar se añade +1 al contador
                # de proteccion de horas fijas en caso de que ya se haya iniciado la
proteccion, esto es que contador
                # para la subfase este activado (valor igual a 1)
                if cont_pro_h_fijas.ix[fase, 'activado'] == 1:
                    cont_pro_h_fijas.ix[fase, 'cont'] += 1

                # Se comprueba si la subfase ha acabado (volumen ejecutado >= volumen
total de la subfase)
                comprueba_fase_acabada(
                    vol_ejecutado, de_tramo, estado_real, fase, hora, cont_t_arranque)

                # Se comprueba si la fase esta finalizada (acabada mas protegida)
                fases_finalizadas = comprueba_fase_finalizada(
                    plan_avance, fases_finalizadas, vol_ejecutado, de_tramo, fase, hora,
cont_t_arranque)

        # Finaliza la verificacion para todas las subfases se avanza a la hora siguiente
        hora += 1
        # Se avanza la hora laborable
        hora_labor += 1
        # Se avanza la hora acumulada
        hora_acumulada += 1

        # Reinicio el contador cuando la hora laborable llega a 25 y avanzo 1 el dia
laborable
        if hora_labor == 25:
            logging.info('Hora: ' + str(hora) + ' Reinicio contador hora laborable')
            logging.info('Hora: ' + str(hora) + ' Avanzo el contador de dia laborable')
            dia_labor += 1
            hora_labor = 0
        # Reinicio el contador dias laborables cuando llega a 7
        if dia_labor == 7:
            logging.info('Dia: ' + str(dia_labor) + ' Reinicio contador dia laborable')
            dia_labor = 0

        # Se extran las fases aun no finalizadas (acabadas + protegidas) para repetir la
verificacion en la siguiente
        # hora
        fase_pos = fases_finalizadas[fases_finalizadas == False]  # nopep8

        #delete "\r" to append instead of overwrite
        sys.stdout.write("\r" + '   ---   ' +  str('LA: F0 ' +
str(np.round(longitudes.ix[0, 'l_avanzada']))) + str(' F1 ' +
str(np.round(longitudes.ix[1, 'l_avanzada']))) + str(' F2 ' +
str(np.round(longitudes.ix[2, 'l_avanzada']))) + str(' F3 ' +
str(np.round(longitudes.ix[3, 'l_avanzada']))) + str(' F4 ' +
str(np.round(longitudes.ix[4, 'l_avanzada']))) + str(' F5 ' +
str(np.round(longitudes.ix[5, 'l_avanzada']))) + str(' F6 ' +
str(np.round(longitudes.ix[6, 'l_avanzada']))) + str(' F7 ' +
str(np.round(longitudes.ix[7, 'l_avanzada']))) + str(' F8 ' +
str(np.round(longitudes.ix[8, 'l_avanzada']))) + str(' F9 ' +
str(np.round(longitudes.ix[9, 'l_avanzada']))) + str(' F10 ' +
str(np.round(longitudes.ix[10, 'l_avanzada']))) + '  ---  ' + str('LP  F0 ' +
str(np.round(longitudes.ix[0, 'l_protegida']))) + str(' F1 ' +
str(np.round(longitudes.ix[1, 'l_protegida']))) + str(' F2 ' +
str(np.round(longitudes.ix[2, 'l_protegida']))) + str(' F3 ' +
str(np.round(longitudes.ix[3, 'l_protegida']))) + str(' F4 ' +
str(np.round(longitudes.ix[4, 'l_protegida']))) + str(' F5 ' +
```

```
        str(np.round(longitudes.ix[5, 'l_protegida'])) + str(' F6 ' +
        str(np.round(longitudes.ix[6, 'l_protegida'])) + str(' F7 ' +
        str(np.round(longitudes.ix[7, 'l_protegida'])) + str(' F8 ' +
        str(np.round(longitudes.ix[8, 'l_protegida'])) + str(' F9 ' +
        str(np.round(longitudes.ix[9, 'l_protegida'])) + str(' F10 ' +
        str(np.round(longitudes.ix[10, 'l_protegida'])) )
               sys.stdout.flush()
```

```
   ---   LA: F0 500.0 F1 500.0 F2 500.0 F3 503.0 F4 503.0 F5 500.0 F6 500.0 F7 500.0
F8 500.0 F9 501.0 F10 500.0  ---  LP  F0 500.0 F1 500.0 F2 0.0 F3 501.0 F4 501.0 F5
0.0 F6 0.0 F7 500.0 F8 0.0 F9 0.0 F10 0.0
```

1.7 Adjudicación de costes del proceso constructivo

Tras la verificación del proceso constructivo, los descriptores de la obra obtenidos se utilizan como datos de entrada para la adjudicación de costes siguiendo el esquema descrito en el Manual.

```
In [9]: # Adjudicacion de costes a la simulacion del proceso constructivo
        (costes_ejecc_sf_unit, costes_directos_sf_unit, costes_ejecc_sf_total,
        costes_directos_sf_total,
            costes_sf_total) = calculo_costes(plan_avance, maquinaria_fases, estado_real,
        vol_perdido, eq_coste_fases,
                                              eq_danno_fases, clima, costes_fase, vol_ejecutado)
```

1.8 Almacenamiento de los resultados de la verificación

```
In [10]: # Finalizada la verificacion del tramo se annade la hora de finalizacion del tramo que
         es la misma que la de
         # inicio del tramo siguiente
         hora_inicio_tramos.append(hora_acumulada)

         # Recorte de matrices resultantes
         (clima, avance_real, estado_real, vol_ejecutado, vol_perdido) =
         recorte_matrices_resultantes(
             hora, clima, avance_real, estado_real, vol_ejecutado, vol_perdido)

         # Inicializacion de las variables de salida
         de_tramo_total = []
         com_fin_teorico_total = []
         avance_real_total = []
         estado_real_total = []
         vol_ejecutado_total = []
         vol_perdido_total = []
         avance_real_total = []
         longitudes_total = []
         costes_ejecc_sf_unit_total = []
         costes_directos_sf_unit_total = []
         costes_ejecc_sf_total_total = []
         costes_directos_sf_total_total = []
         costes_sf_total_total = []

         # Almaceno los resultados
         de_tramo_total.append(de_tramo)
         com_fin_teorico_total.append(com_fin_teorico)
         avance_real_total.append(avance_real)
         estado_real_total.append(estado_real)
         vol_ejecutado_total.append(vol_ejecutado)
         vol_perdido_total.append(vol_perdido)
         longitudes_total.append(longitudes)

         # if alcance != 'EA':
         costes_ejecc_sf_unit_total.append(costes_ejecc_sf_unit)
         costes_directos_sf_unit_total.append(costes_directos_sf_unit)
         costes_ejecc_sf_total_total.append(costes_ejecc_sf_total)
```

```
        costes_directos_sf_total_total.append(costes_directos_sf_total)
        costes_sf_total_total.append(costes_sf_total)

        (lon_ejecutada_total, lon_acumulada_total, vol_acumulado_total,
         lon_diferencia_total) = calculo_longitudes_volumenes_acumulados(estado_real_total,
        vol_ejecutado_total,
                                                                        de_tramo_total)
```

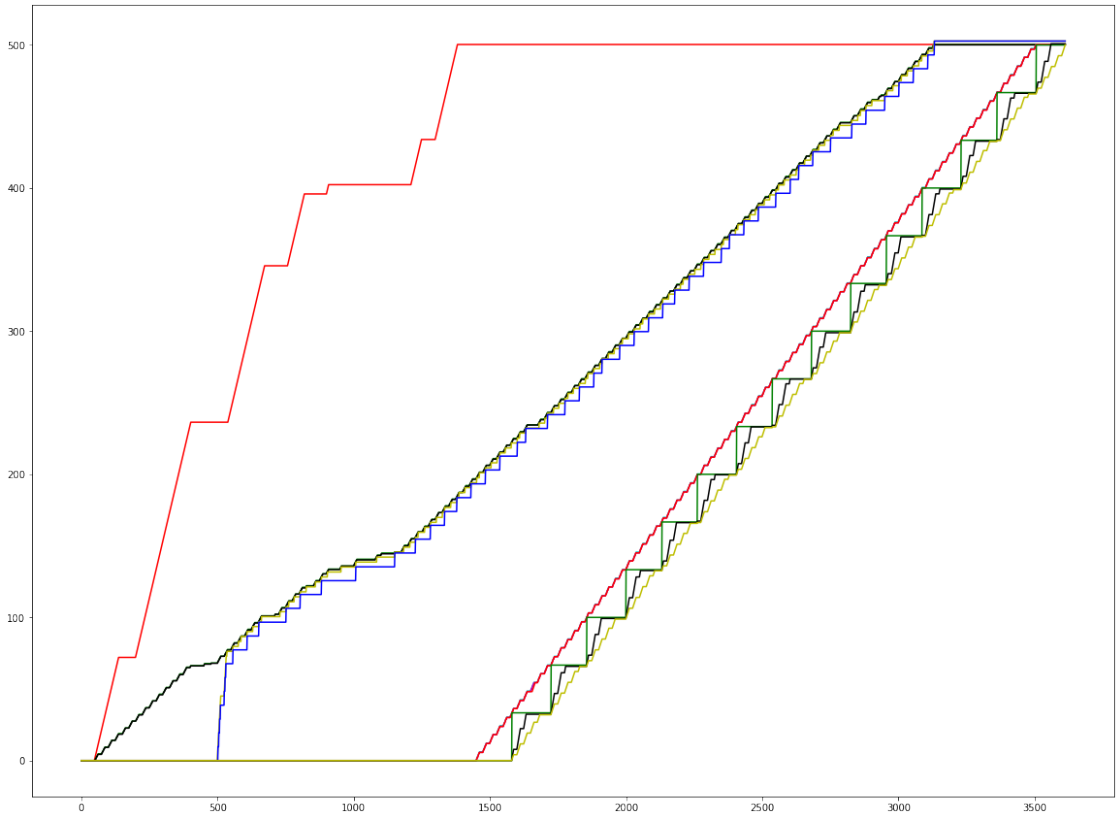
```
500.285714286
500.25
500.004500004
502.666666667
502.666666667
500.266666667
500.366666667
500.366666667
500.0
500.772547129
500.0
```

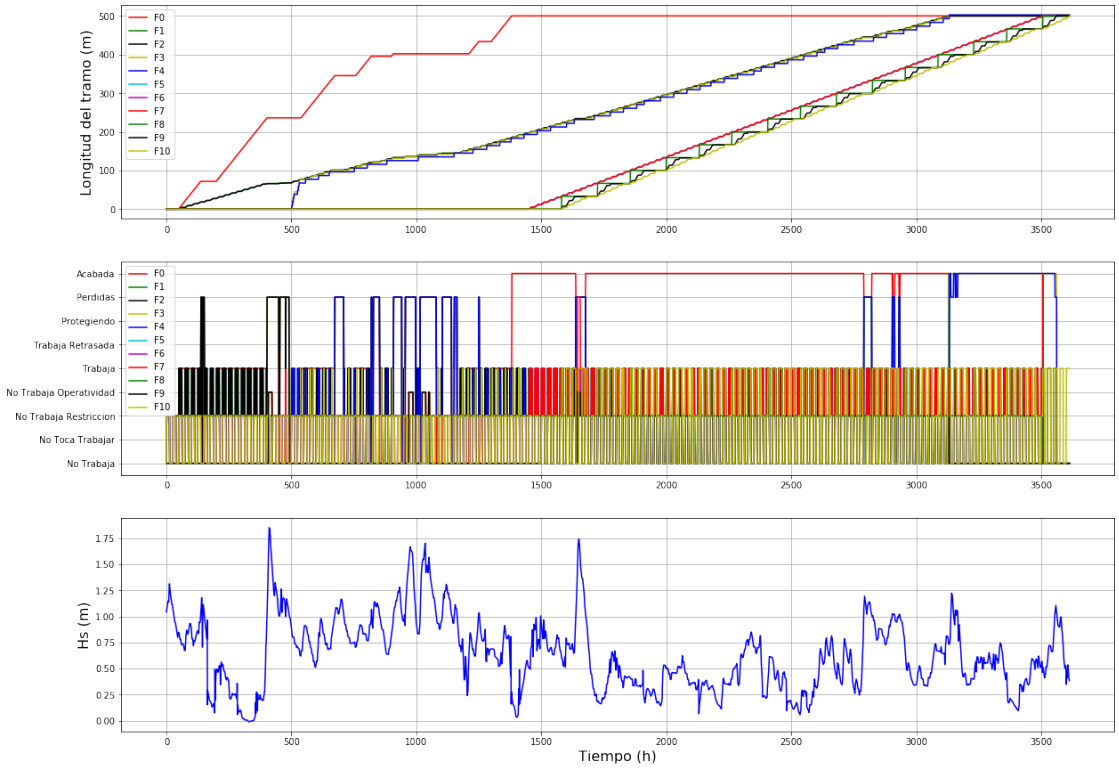
1.9 Extraccion de resultados

```
In [11]: # Extraccion de las variables de tiempos, probabilidades, volumenes y costes
         (df_duracion, df_horas, df_probabilidad, df_tiempos, df_volumen, df_costes_ejecucion,
         df_costes_directos,
          df_costes_totales) = extraccion_resultados(de_planta, n, estado_real_total,
         vol_acumulado_total,
          vol_ejecutado_total, de_tramo_total, hora_inicio_tramos, vol_perdido_total,
         costes_ejecc_sf_total_total,
          costes_directos_sf_total_total, costes_sf_total_total, com_fin_teorico_total, ruta_de,
         ruta_ds, alcance)
```

1.10 Resultados de la verificación del proceso constructivo y adjudicación de costes

```
In [12]: # Representacion de resultados de tiempo de ejecucion
         representacion_resultados_tiempo(lon_acumulada_total, hora_inicio_tramos,
         estado_real_total, n, ruta_de, ruta_ds)
```


500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
3500


Longitud del tramo (m)
F0
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
Acabada
Perdidas
Protegiendo
Trabaja Retrasada
Trabaja
No Trabaja Operatividad
No Trabaja Restriccion
No Toca Trabajar
No Trabaja
Hs (m)
Tiempo (h)

1.10.1 Tiempos de ejecución

In [20]:
```
# Cambio el numero de filas y columnas que se muestran
env.pandas_setup(50, 50)
df_tiempos
```

Out[20]:

		t_ejecucion_tramo	\
T_0	0_dragado	3613.0	
	1_vertido_nucleo_banqueta	3613.0	
	2_vertido_manto_proteccion_banqueta	3613.0	
	3_enrase_banqueta_extraccion	3613.0	
	4_enrase_banqueta_adicion	3613.0	
	5_fabricacion_cajones	3613.0	
	6_preparacion_cajones_remolque	3613.0	
	7_transporte_cajones	3613.0	
	8_acopio_cajones	3613.0	
	9_fondeo_cajones	3613.0	
	10_relleno_celda	3613.0	

		t_ini_subfase_teorico	\
T_0	0_dragado	0.0	
	1_vertido_nucleo_banqueta	0.0	
	2_vertido_manto_proteccion_banqueta	0.0	
	3_enrase_banqueta_extraccion	0.0	
	4_enrase_banqueta_adicion	0.0	
	5_fabricacion_cajones	0.0	
	6_preparacion_cajones_remolque	0.0	
	7_transporte_cajones	0.0	
	8_acopio_cajones	0.0	
	9_fondeo_cajones	0.0	
	10_relleno_celda	0.0	

		t_ini_subfase_real	\
T_0	0_dragado	48.0	
	1_vertido_nucleo_banqueta	49.0	
	2_vertido_manto_proteccion_banqueta	50.0	
	3_enrase_banqueta_extraccion	499.0	
	4_enrase_banqueta_adicion	501.0	
	5_fabricacion_cajones	1449.0	
	6_preparacion_cajones_remolque	1450.0	
	7_transporte_cajones	1450.0	
	8_acopio_cajones	1580.0	
	9_fondeo_cajones	1580.0	
	10_relleno_celda	1580.0	

		t_fin_subfase_teorico	\

T_0	0_dragado	99999.0
	1_vertido_nucleo_banqueta	99999.0
	2_vertido_manto_proteccion_banqueta	99999.0
	3_enrase_banqueta_extraccion	99999.0
	4_enrase_banqueta_adicion	99999.0
	5_fabricacion_cajones	99999.0
	6_preparacion_cajones_remolque	99999.0
	7_transporte_cajones	99999.0
	8_acopio_cajones	99999.0
	9_fondeo_cajones	99999.0
	10_relleno_celda	99999.0

		t_fin_subfase_real \
T_0	0_dragado	3130.0
	1_vertido_nucleo_banqueta	3130.0
	2_vertido_manto_proteccion_banqueta	3130.0
	3_enrase_banqueta_extraccion	3560.0
	4_enrase_banqueta_adicion	3560.0
	5_fabricacion_cajones	3505.0
	6_preparacion_cajones_remolque	3506.0
	7_transporte_cajones	3506.0
	8_acopio_cajones	3506.0
	9_fondeo_cajones	3560.0
	10_relleno_celda	3613.0

		t_ejecucion_subfase \
T_0	0_dragado	3082.0
	1_vertido_nucleo_banqueta	3081.0
	2_vertido_manto_proteccion_banqueta	3080.0
	3_enrase_banqueta_extraccion	3061.0
	4_enrase_banqueta_adicion	3059.0
	5_fabricacion_cajones	2056.0
	6_preparacion_cajones_remolque	2056.0
	7_transporte_cajones	2056.0
	8_acopio_cajones	1926.0
	9_fondeo_cajones	1980.0
	10_relleno_celda	2033.0

		n_horas_subfase_trabaja \
T_0	0_dragado	618.0
	1_vertido_nucleo_banqueta	1380.0
	2_vertido_manto_proteccion_banqueta	817.0
	3_enrase_banqueta_extraccion	145.0
	4_enrase_banqueta_adicion	52.0
	5_fabricacion_cajones	1072.0
	6_preparacion_cajones_remolque	883.0
	7_transporte_cajones	883.0
	8_acopio_cajones	15.0

```
        9_fondeo_cajones                                    449.0
        10_relleno_celda                                    860.0

                                                  n_horas_subfase_trabaja_retrasada
    T_0 0_dragado                                                               0.0
        1_vertido_nucleo_banqueta                                               0.0
        2_vertido_manto_proteccion_banqueta                                     0.0
        3_enrase_banqueta_extraccion                                            0.0
        4_enrase_banqueta_adicion                                               0.0
        5_fabricacion_cajones                                                   0.0
        6_preparacion_cajones_remolque                                          0.0
        7_transporte_cajones                                                    0.0
        8_acopio_cajones                                                        0.0
        9_fondeo_cajones                                                        0.0
        10_relleno_celda                                                        0.0
```

1.10.2 Tiempos de parada

```
In [14]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)
         df_tiempos
```

```
Out[14]:                                          t_ejecucion_tramo  \
    T_0 0_dragado                                            3613.0
        1_vertido_nucleo_banqueta                            3613.0
        2_vertido_manto_proteccion_banqueta                  3613.0
        3_enrase_banqueta_extraccion                         3613.0
        4_enrase_banqueta_adicion                            3613.0
        5_fabricacion_cajones                                3613.0
        6_preparacion_cajones_remolque                       3613.0
        7_transporte_cajones                                 3613.0
        8_acopio_cajones                                     3613.0
        9_fondeo_cajones                                     3613.0
        10_relleno_celda                                     3613.0

                                                  t_ini_subfase_teorico  \
    T_0 0_dragado                                                   0.0
        1_vertido_nucleo_banqueta                                   0.0
        2_vertido_manto_proteccion_banqueta                         0.0
        3_enrase_banqueta_extraccion                                0.0
        4_enrase_banqueta_adicion                                   0.0
        5_fabricacion_cajones                                       0.0
        6_preparacion_cajones_remolque                              0.0
        7_transporte_cajones                                        0.0
        8_acopio_cajones                                            0.0
        9_fondeo_cajones                                            0.0
        10_relleno_celda                                            0.0

                                                  t_ini_subfase_real  \
```

		t_fin_subfase_teorico	\
T_0	0_dragado	99999.0	
	1_vertido_nucleo_banqueta	99999.0	
	2_vertido_manto_proteccion_banqueta	99999.0	
	3_enrase_banqueta_extraccion	99999.0	
	4_enrase_banqueta_adicion	99999.0	
	5_fabricacion_cajones	99999.0	
	6_preparacion_cajones_remolque	99999.0	
	7_transporte_cajones	99999.0	
	8_acopio_cajones	99999.0	
	9_fondeo_cajones	99999.0	
	10_relleno_celda	99999.0	

		t_fin_subfase_real	\
T_0	0_dragado	3130.0	
	1_vertido_nucleo_banqueta	3130.0	
	2_vertido_manto_proteccion_banqueta	3130.0	
	3_enrase_banqueta_extraccion	3560.0	
	4_enrase_banqueta_adicion	3560.0	
	5_fabricacion_cajones	3505.0	
	6_preparacion_cajones_remolque	3506.0	
	7_transporte_cajones	3506.0	
	8_acopio_cajones	3506.0	
	9_fondeo_cajones	3560.0	
	10_relleno_celda	3613.0	

		t_ejecucion_subfase	\
T_0	0_dragado	3082.0	
	1_vertido_nucleo_banqueta	3081.0	
	2_vertido_manto_proteccion_banqueta	3080.0	
	3_enrase_banqueta_extraccion	3061.0	
	4_enrase_banqueta_adicion	3059.0	
	5_fabricacion_cajones	2056.0	
	6_preparacion_cajones_remolque	2056.0	
	7_transporte_cajones	2056.0	
	8_acopio_cajones	1926.0	

```
                                    9_fondeo_cajones                                1980.0
                                    10_relleno_celda                                2033.0

                                                                    n_horas_subfase_trabaja  \
T_0 0_dragado                                                                        618.0
    1_vertido_nucleo_banqueta                                                       1380.0
    2_vertido_manto_proteccion_banqueta                                              817.0
    3_enrase_banqueta_extraccion                                                     145.0
    4_enrase_banqueta_adicion                                                         52.0
    5_fabricacion_cajones                                                           1072.0
    6_preparacion_cajones_remolque                                                   883.0
    7_transporte_cajones                                                             883.0
    8_acopio_cajones                                                                  15.0
    9_fondeo_cajones                                                                 449.0
    10_relleno_celda                                                                 860.0

                                                                    n_horas_subfase_trabaja_retrasada
T_0 0_dragado                                                                                     0.0
    1_vertido_nucleo_banqueta                                                                     0.0
    2_vertido_manto_proteccion_banqueta                                                           0.0
    3_enrase_banqueta_extraccion                                                                  0.0
    4_enrase_banqueta_adicion                                                                     0.0
    5_fabricacion_cajones                                                                         0.0
    6_preparacion_cajones_remolque                                                                0.0
    7_transporte_cajones                                                                          0.0
    8_acopio_cajones                                                                              0.0
    9_fondeo_cajones                                                                              0.0
    10_relleno_celda                                                                              0.0
```

1.10.3 Duración de las paradas

```
In [15]: df_horas
```

```
Out[15]:                                    n_horas_subfase_no_trabaja_por_restriccion  \
         T_0 0_dragado                                                             0.0
             1_vertido_nucleo_banqueta                                            25.0
             2_vertido_manto_proteccion_banqueta                                 588.0
             3_enrase_banqueta_extraccion                                       1193.0
             4_enrase_banqueta_adicion                                          1401.0
             5_fabricacion_cajones                                                 0.0
             6_preparacion_cajones_remolque                                      943.0
             7_transporte_cajones                                               2561.0
             8_acopio_cajones                                                   3491.0
             9_fondeo_cajones                                                    622.0
             10_relleno_celda                                                   1024.0

                                                                    n_horas_subfase_no_trabaja_por_operatividad
         T_0 0_dragado                                                                                     0.0
             1_vertido_nucleo_banqueta                                                                     0.0
```

	2_vertido_manto_proteccion_banqueta	0.0
	3_enrase_banqueta_extraccion	62.0
	4_enrase_banqueta_adicion	0.0
	5_fabricacion_cajones	0.0
	6_preparacion_cajones_remolque	0.0
	7_transporte_cajones	51.0
	8_acopio_cajones	0.0
	9_fondeo_cajones	62.0
	10_relleno_celda	0.0

		n_horas_subfase_esta_protegiendo
T_0	0_dragado	0.0
	1_vertido_nucleo_banqueta	0.0
	2_vertido_manto_proteccion_banqueta	0.0
	3_enrase_banqueta_extraccion	0.0
	4_enrase_banqueta_adicion	0.0
	5_fabricacion_cajones	0.0
	6_preparacion_cajones_remolque	0.0
	7_transporte_cajones	0.0
	8_acopio_cajones	0.0
	9_fondeo_cajones	0.0
	10_relleno_celda	0.0

		n_horas_subfase_esta_en_perdidas
T_0	0_dragado	431.0
	1_vertido_nucleo_banqueta	431.0
	2_vertido_manto_proteccion_banqueta	431.0
	3_enrase_banqueta_extraccion	0.0
	4_enrase_banqueta_adicion	363.0
	5_fabricacion_cajones	0.0
	6_preparacion_cajones_remolque	0.0
	7_transporte_cajones	11.0
	8_acopio_cajones	0.0
	9_fondeo_cajones	0.0
	10_relleno_celda	0.0

1.10.4 Volumen de pérdidas

In [16]: df_volumen

Out[16]:

		volumen_ejecutado_teorico
T_0	0_dragado	189000.0
	1_vertido_nucleo_banqueta	200000.0
	2_vertido_manto_proteccion_banqueta	111111.0
	3_enrase_banqueta_extraccion	15000.0
	4_enrase_banqueta_adicion	7500.0
	5_fabricacion_cajones	15.0
	6_preparacion_cajones_remolque	15.0
	7_transporte_cajones	15.0

	8_acopio_cajones	15.0
	9_fondeo_cajones	15.0
	10_relleno_celda	172000.0

		volumen_ejecutado_real
T_0	0_dragado	189108.000000
	1_vertido_nucleo_banqueta	200100.000000
	2_vertido_manto_proteccion_banqueta	111112.000000
	3_enrase_banqueta_extraccion	15080.000000
	4_enrase_banqueta_adicion	7540.000000
	5_fabricacion_cajones	15.008000
	6_preparacion_cajones_remolque	15.011000
	7_transporte_cajones	15.011000
	8_acopio_cajones	15.000000
	9_fondeo_cajones	15.023176
	10_relleno_celda	172000.000000

		volumen_minimo_perdidas_material
T_0	0_dragado	-148.234091
	1_vertido_nucleo_banqueta	-0.056721
	2_vertido_manto_proteccion_banqueta	-617.564254
	3_enrase_banqueta_extraccion	0.000000
	4_enrase_banqueta_adicion	-12.923093
	5_fabricacion_cajones	0.000000
	6_preparacion_cajones_remolque	0.000000
	7_transporte_cajones	-4.895000
	8_acopio_cajones	0.000000
	9_fondeo_cajones	0.000000
	10_relleno_celda	0.000000

		volumen_maximo_perdidas_material
T_0	0_dragado	-808316.027273
	1_vertido_nucleo_banqueta	-1640.422708
	2_vertido_manto_proteccion_banqueta	-260705.435978
	3_enrase_banqueta_extraccion	0.000000
	4_enrase_banqueta_adicion	-16588.585859
	5_fabricacion_cajones	0.000000
	6_preparacion_cajones_remolque	0.000000
	7_transporte_cajones	-4.895000
	8_acopio_cajones	0.000000
	9_fondeo_cajones	0.000000
	10_relleno_celda	0.000000

		volumen_medio_perdidas_material
T_0	0_dragado	-158446.238038
	1_vertido_nucleo_banqueta	-215.468903
	2_vertido_manto_proteccion_banqueta	-57887.872893
	3_enrase_banqueta_extraccion	0.000000

	4_enrase_banqueta_adicion	-3180.191660
	5_fabricacion_cajones	0.000000
	6_preparacion_cajones_remolque	0.000000
	7_transporte_cajones	-4.895000
	8_acopio_cajones	0.000000
	9_fondeo_cajones	0.000000
	10_relleno_celda	0.000000

		volumen_total_perdidas_material
T_0	0_dragado	-3.010479e+06
	1_vertido_nucleo_banqueta	-4.093909e+03
	2_vertido_manto_proteccion_banqueta	-1.099870e+06
	3_enrase_banqueta_extraccion	0.000000e+00
	4_enrase_banqueta_adicion	-5.724345e+04
	5_fabricacion_cajones	0.000000e+00
	6_preparacion_cajones_remolque	0.000000e+00
	7_transporte_cajones	-4.895000e+00
	8_acopio_cajones	0.000000e+00
	9_fondeo_cajones	0.000000e+00
	10_relleno_celda	0.000000e+00

1.10.5 Probabilidad de fallo y parada operativa

In [17]: df_probabilidad

Out[17]:

		probabilidad_de_fallo
T_0	0_dragado	0.139844
	1_vertido_nucleo_banqueta	0.139890
	2_vertido_manto_proteccion_banqueta	0.139935
	3_enrase_banqueta_extraccion	0.000000
	4_enrase_banqueta_adicion	0.118666
	5_fabricacion_cajones	0.000000
	6_preparacion_cajones_remolque	0.000000
	7_transporte_cajones	0.005350
	8_acopio_cajones	0.000000
	9_fondeo_cajones	0.000000
	10_relleno_celda	0.000000

		probabilidad_parada_operativa
T_0	0_dragado	0.000000
	1_vertido_nucleo_banqueta	0.000000
	2_vertido_manto_proteccion_banqueta	0.000000
	3_enrase_banqueta_extraccion	0.020255
	4_enrase_banqueta_adicion	0.000000
	5_fabricacion_cajones	0.000000
	6_preparacion_cajones_remolque	0.000000
	7_transporte_cajones	0.024805
	8_acopio_cajones	0.000000

	9_fondeo_cajones	0.031313
	10_relleno_celda	0.000000

In [18]: df_costes_totales

Out[18]:

		c_directos_total	c_indirectos_total	\
T_0	0_dragado	8.652000e+05	86520.000000	
	1_vertido_nucleo_banqueta	3.167100e+06	316710.000000	
	2_vertido_manto_proteccion_banqueta	1.346416e+06	134641.600000	
	3_enrase_banqueta_extraccion	1.450000e+05	14500.000000	
	4_enrase_banqueta_adicion	1.086800e+05	10868.000000	
	5_fabricacion_cajones	5.853120e+04	5853.120000	
	6_preparacion_cajones_remolque	5.854290e+04	5854.290000	
	7_transporte_cajones	6.030890e+04	6030.890000	
	8_acopio_cajones	0.000000e+00	0.000000	
	9_fondeo_cajones	5.859039e+04	5859.038801	
	10_relleno_celda	1.255600e+06	125560.000000	

		c_perdidas_total	c_total
T_0	0_dragado	1.823808e+07	1.918980e+07
	1_vertido_nucleo_banqueta	8.187818e+04	3.565688e+06
	2_vertido_manto_proteccion_banqueta	2.675358e+07	2.823464e+07
	3_enrase_banqueta_extraccion	0.000000e+00	1.595000e+05
	4_enrase_banqueta_adicion	2.595049e+06	2.714597e+06
	5_fabricacion_cajones	0.000000e+00	6.438432e+04
	6_preparacion_cajones_remolque	0.000000e+00	6.439719e+04
	7_transporte_cajones	1.909050e+04	8.543029e+04
	8_acopio_cajones	0.000000e+00	0.000000e+00
	9_fondeo_cajones	0.000000e+00	6.444943e+04
	10_relleno_celda	0.000000e+00	1.381160e+06

co_costes_reparacion_01_es_c1

April 27, 2018

1 Ejemplo para el cálculo de costes de reparacion en el alcance de estudios soluciones clase I: Parte 1 - Lectura de los datos de entrada.

1.1 Importación de paquetes de IPython

In [2]:
```
# Python 2/3 setup
from __future__ import (absolute_import, division, print_function, unicode_literals)
# from builtins import *
```

In [3]:
```
import os
import sys
```

In [4]:
```
# Juypyter setup

%matplotlib inline
import os
import sys
from IPython.display import HTML
```

1.2 Importación de paquetes para el ejemplo

In [5]:
```
import logging
import os
import pickle

from reparacion.datos_entrada import datos_entrada_planta
from reparacion.datos_entrada import datos_entrada_diagrama_modos
from reparacion.datos_entrada import datos_entrada_arbol_fallo
from reparacion.datos_entrada import datos_entrada_esquema_division_dique
from reparacion.datos_entrada import datos_entrada_clima_tramos
from reparacion.datos_entrada import datos_entrada_verificacion_dique


from reparacion.calculos import extraccion_resultados
from reparacion.calculos import calculo_costes

from reparacion.datos_entrada import datos_entrada_elementos_reparacion_necesarios
from reparacion.datos_entrada import datos_entrada_elementos_reparacion_disponibles
from reparacion.datos_entrada import datos_entrada_tipo_verificacion
```

1.3 Datos de entrada de la forma en planta

1.3.1 RUTA DE DIRECTORIOS DE DATOS DE ENTRADA Y SALIDA

In [6]:
```
# Ruta con los datos de entrada
ruta_de = os.path.join(env.input_path, 'reparacion', 'estudio_soluciones_clase1')
# Ruta con los datos de salida
ruta_ds = os.path.join(env.output_path, 'reparacion', 'estudio_soluciones_clase1')
```

1.3.2 INTRODUCCION

De acuerdo con el articulado de la ROM 1.1. en el alcance de estudio de soluciones de clase I se deben calcular los costes medios de reparación a partir de la información de los agentes en el emplazamiento. Los costes específicos de reparación se determinarán a partir de un diseño de ELU según la configuración del diagrama de modos de fallo del dique. Los costes anuales medios de reparación serán los de reconstrucción del tramo, y se contabilizarán siempre que se incumpla la ecuación de verificación de cualquiera de los modos principales.

El cálculo de los costes se realiza para tres niveles de aproximación o estrategias senciallas:

- **Estrategia de no reparación**: En esta estrategia o nivel de aproximación se podrá considerar que los modos de fallo no se reparan en ningún momento a lo largo de la vida útil de la obra de abrigo. Esta estratgia permitirá comprobar si el dique es capaz de cumplir los condicionantes y requisitos de proyecto sin reparaciones.
- **Estrategia de reparacion inmediata**: En esta estrategia o nivel de aproximación se podrá suponer que el tramo se repara de forma inmediata a la ocurrencia del fallo y recupera sus características estructurales y formales originales. Mediante esta aproximación se contabilizará a partir de las series históricas de los agentes propagadas al emplazamiento y las ecuaciones de verificación e cada uno de los modos del dique el número de veces que falla cada uno de ellos. A partir de esta información y del coste de reconstrucción de cada uno de los modos se obtiene un valor del coste idealizado de reparación del dique a lo largo de la vida útil
- **Estrategia reparacion en tiempo real**: En esta estrategia o nivel de aproximación y con el fin de evaluar la posibilidad de efectuar las operaciones de reparación del dique, se debe especificar el tiempo necesario que debe transcurrir entre que se da la orden de reparar y realmente se inicia la reparación del modo. Además se debe especificar el rendimiento con el que se realiza la reparación. Con esta nueva información se podrá contabilizar el número de veces que se incumplen los criterios de reparación considerados en función del intervalo de tiempo entre ciclos y de los datos de reparación considerados. En esta segunda aproximación, si se desea, se podrán considerar las curvas de acumulación de daño para cada modo y considerar una verificación a ELS.

Se muestra un ejemplo del cálculo de los costes de reparación de un subsistema de un tramo de dique compuesto por tres modos de fallo teóricos. Se describen a continuación los datos de entrada considerados.

1.3.3 DEFINICIÓN DEL ALCANCE Y ESTRATEGIA

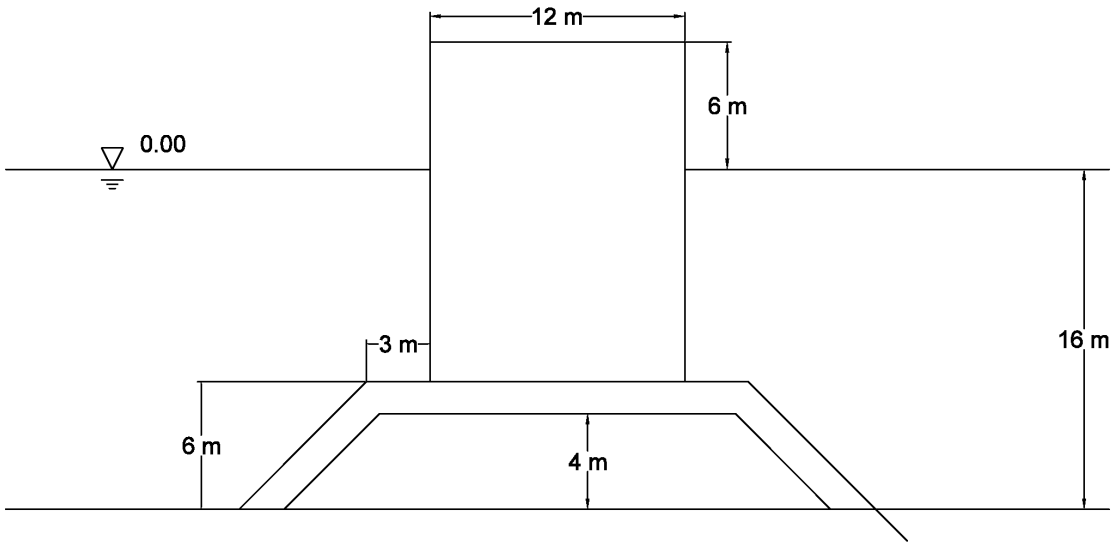
```
In [7]: alcance = 'EA'

        #estrategia = 'no_reparacion'
        #estrategia = 'reparacion_inmediata'
        estrategia = 'reparacion_tiempo_real'
```

1.3.4 DEFINICIÓN DE LOS DATOS DE ENTRADA

En este ejemplo se propone analizar el coste de un dique de tipo mixto con berma baja de protección (Tipo C, LMB). La sección tipo del dique a analizar se muestra en la figura. En este ejemplo, se considera para la forma en planta un dique compuesto por una única alineación principal de 500 m de longitud. El calado de la alineación principal es de 16 m respecto al nivel medio del mar, la altura de la berma es de 4 m y los cajones tienen una altura de 18 m por 12 m de ancho.

```
In [8]: from IPython.display import Image
        img_name = os.path.join(ruta_de, 'imagenes', 'esquema_dique_c.png')
        Image(filename=img_name, width=800)
```

Out[8]:



Lectura de los ficheros con los datos de entrada de tipología y forma en planta (Para modificar los datos utilizados en el ejemplo se deben modificar los ficheros de la carpeta de inputs)

```
In [9]: # Lectura de los datos de entrada en planta
        (de_planta) = datos_entrada_planta(ruta_de)
        de_planta
```

Out[9]:

	longitud	vida_util	calado
T_0	500.0	1.0	16.0

Lectura de los ficheros con los datos de entrada del esquema de componentes De acuerdo con el articulado la estructura jerárquica del dique se descompone en: tramos, subsistemas y modos de fallo. En este ejemplo teórico se muestra el cálculo de costes de un tramo compuesto por un subsistema con tres modos de fallo.

```
In [10]: # lectura del esquema de division del dique
         (de_esquema_division_dique) = datos_entrada_esquema_division_dique(ruta_de)
         de_esquema_division_dique
```

Out[10]:

	tramo	subsistema	modo_fallo
0	T_0	SS_0	MF_1
1	T_0	SS_0	MF_2
2	T_0	SS_0	MF_3

Lectura de los ficheros con los datos de entrada del diagrama de componentes De acuerdo con el articulado de la ROM 1.1. los diagramas de componentes expresan las posibles combinaciones de elementos del espacio de sucesos que suponen el incumplimiento de los requisitos del sistema. Los tipos de diagramas pueden ser: tipo serie (en el que el sistema falla si lo hace al menos uno de los componentes), tipo paralelo (el sistema falla si lo hacen todos los componentes) o tipo mixto (mezcla de ambos).

Para este ejemplo se ha definido un diagrama para cada estructura jerárquica del dique:

- Diagrama de modos del dique: En el que se indican las posibles combinaciones de tramos que deben fallar para considera el fallo del dique. En este ejemplo se considera que el dique falla si falla el tramo T_0.
- Diagrama de modos de los tramos: En el que se indican las posibles combinaciones de subsistemas que deben fallas para considerar el fallo del tramo. En este ejemplo se considera que el tramo falla si falla el subsistema SS_0.
- Diagrama de modos de los subsistemas: En el que se indican las posibles combinaciones de modos que deben fallar para considerar el fallo de un subsistema. Para este ejemplo se ha considerado un diagrama de modo tipo mixto en los que los modos MF_1 y MF_2 se encuentran en paralelo y éstos en serie respecto al MF_3

```
In [11]: # Lectura de los datos de entrada del diagrama de modos
         (de_diagrama_modos) = datos_entrada_diagrama_modos(ruta_de)
         de_diagrama_modos['de_diagrama_modos_dique']
```

```
Out[11]:            opciones_de_destruccion_total
         Dique  op1                        ['T_0']
```

```
In [12]: de_diagrama_modos['de_diagrama_modos_tramos']
```

```
Out[12]:         opciones_de_destruccion_total
         T_0 op1                       ['SS_0']
```

```
In [13]: de_diagrama_modos['de_diagrama_modos_subsistemas']
```

```
Out[13]:              opciones_de_destruccion_total
         T_0 SS_0 op1              ['MF_1', 'MF_2']
                  op2                       ['MF_3']
```

Lectura de los datos climáticos Dentro de la carpeta 'clima' es necesario que haya una carpeta por cada uno de los tramos que componen el dique con el nombre de la etiqueta del tramo (en este caso únicamente uno). Dentro de cada carpeta se debe encontrar un fichero denominado 'clima.txt' con las series temporales históricas propagadas al emplazamiento de cada uno de los agentes necesarios para realizar la verificación del dique. Las series temporales se estructurarán por columnas.

La matriz con los datos de clima tendrán un valor de nivel (respecto al NMM), velocidad de viento, altura de ola, periodo pico, dirección media de procedendia del oleaje y dirección media de procedencia del viento para cada hora.

Se muestra a continuación un ejemplo del fichero de clima:

```
In [14]: # Lectura de los datos de clima para cada tramo
         (clima_tramos, cadencia) = datos_entrada_clima_tramos(de_esquema_division_dique,
         de_planta, ruta_de)
```

```
        # Cambio el numero de filas y columnas que se muestran
        env.pandas_setup(10, 50)
        display(clima_tramos['T_0'])
        # Cambio el numero de filas y columnas a sus valores por defecto
        env.pandas_setup()
```

```
          nivel        vv        hs         tp          dh          dv  \
0     -0.197452  5.742884  0.412004   7.577019  254.247544  162.987077
1      0.468996  0.714908  0.821672   8.677967  257.565338  141.477188
2      0.118437  3.721660  0.199103  10.859557  253.640407  302.096175
3     -0.533291  2.043294  0.547271   7.138475  251.336751    2.889659
4     -0.267650  1.815516  0.750617   8.526316  255.215891  165.937557
...         ...       ...       ...        ...         ...         ...
29211 -0.030432  2.336437  0.438078   9.457665  237.048698  331.545860
29212 -0.566616  2.086231  0.446601   9.218670  241.627192  123.033967
29213 -0.160335  1.285514  0.398538  13.184551  245.040297  358.593331
29214  0.474897  2.019519  0.403624  11.975245  242.009375  354.965854
29215  0.042921  3.671009  0.472338  14.473563  247.397039  258.424375

            mas       mme     calado
0      1.595942 -0.067649  15.802548
1      2.258235 -0.063495  16.468996
2      1.912683 -0.068501  16.118437
3      1.257457 -0.065004  15.466709
4      1.521742 -0.063648  15.732350
...         ...       ...        ...
29211  1.758231 -0.062919  15.969568
29212  1.215802 -0.056673  15.433384
29213  1.630499 -0.065089  15.839665
29214  2.265278 -0.064636  16.474897
29215  1.833988 -0.065322  16.042921

[29216 rows x 9 columns]
```

Lectura de datos de entrada del tipo de verificación para cada modo de fallo De acuerdo con el Articulado de la ROM 1.1 en el alcance de estudio de soluciones de clase I se deben calcular los costes medios de reparación a partir de un diseño de ELU según la configuración del diagrama de modos de fallo del dique. Por lo tanto dentro de la carpeta 'tipo_verificacion_curva_acum_dano' el fichero 'tipo_verificacion_curva_acum_dano.txt' le indicará a la herramienta que cada modo de fallo se verifica a ELU y por lo tanto los parámetros de la curva de acumulación de daño se fijan a 0. Si se desease realizar una verificación a ELS bastarría con modificar el parámetro a ELS y añadir para cada modo los parámetros a, b y c de curva de acumulación de daño.

Se muestra a continuación un ejemplo con los ficheros de entrada para el tipo de verificación para cada modo de fallo:

```
In [15]: # Tipo de verificacion y curva de acumulacion de dano
         de_tipo_verificacion = datos_entrada_tipo_verificacion(ruta_de)

         de_tipo_verificacion
```

```
Out[15]:                tipo_diseno  par_a  par_b  par_c
         T_0 SS_0 MF_1         elu    0.0    0.0    0.0
                  MF_2         elu    0.0    0.0    0.0
                  MF_3         elu    0.0    0.0    0.0
```

Además se debe leer los resultados del módulo de verificación en el que se indica para cada estado de mar de la vida útil si cada uno de los modos de fallo fallan o no fallan

```
In [16]:     # Verificacion tramos del dique
             (de_verificacion_tramos, peralte) = datos_entrada_verificacion_dique(ruta_de,
         de_esquema_division_dique)
             de_verificacion_tramos.loc[:, ('T_0', 'SS_0', 'MF_1')]
```

```
Out[16]: 0        False
         1        False
         2        False
         3        False
         4        False
                  ...
         14595    False
         14596    False
         14597    False
         14598    False
         14599    False
         Name: (T_0, SS_0, MF_1), Length: 14600, dtype: object
```

Lectura de los datos de entrada con la estrategia de reparación Para cada modo de fallo es necesario especificar: el tiempo medio de espera que debe transcurrir desde que se inicia la orden de reparación hasta que se inician los trabajos de reparación; el rendimiento medio de reparación del modo en (% de avería reparada / hora) y los costes medios de reparación del modo en (€ / hora).

Es importante remarcar que la herramienta modificará algunos de los parámetros introducidos relacionados con la estrategia de reparación para adapatarlos a la estrategia definida:

- Si la estrategia es de **no reparación**: la herramienta no iniciará los trabajos de reparación en ningún momento independientemente de los parámetros introducidos.
- Si la estrategia es de **reparacion inmediata**: la herramienta considerará disponibilidad infinita de medios y un rendimiento de reparación infinito. De este modo, la herramienta reparará toda la avería en el estado posterior al que se produjo la avería independientemente de los parámetros introducidos.
- Si la estrategia es de **reparacion en tiempo real**: la herramienta considera los tiempos medios de inicio de reparación y rendimientos introducidos por el usuario.

NOTA IMPORTANTE Es importante remarcar que en función de la estrategia elegida, los costes introducidos serán tratados de forma diferentes por la herramienta: - Si la estrategia es de **no reparación**: No se utilizarán los costes de reparación. - Si la estrategia es de **reparacion inmediata**: El coste introducido deberá ser el de reconstrucción de todo el modo de fallo. Dicho coste será aplicado por la herramienta cada vez que se repare. - Si la estrategia es de **reparacion en tiempo real**: En este caso, el coste introducido deberá tener las unidades de $e/hora$. Es el coste total de reparación por cada hora de trabajo.

En este ejemplo se considerarán unos tiempos medios teóricos de inicio de reparación de 48, 12 y 24 h. Los rendimientos de reparación teóricos considerados son: 0.1, 0.001 y 0.01 (% / h) y el coste teórico de reparación de cada modo considerado es de 150 € / h. Para este alcance no se definen elementos de reparación disponibles en Puerto de forma permanente.

```
In [17]:     # Datos de entrada de reparacion necesario
             de_reparacion_necesarios = datos_entrada_elementos_reparacion_necesarios(ruta_de,
         cadencia, alcance,
                                                                                      estrategia)
             # Datos de entrada de reparacion disponibles
             de_reparacion_disponibles = datos_entrada_elementos_reparacion_disponibles(ruta_de,
         cadencia, alcance)

             de_reparacion_necesarios
```

Out[17]:

		t_espera_ini_reparacion	rend	costes_reparacion	\
T_0 SS_0	MF_1	48.0	0.001	150.0	
	MF_2	12.0	0.010	150.0	
	MF_3	24.0	0.100	150.0	

		na_umbral_ini_reparacion	na_umbral_fin_reparacion
T_0 SS_0	MF_1	0.01	0
	MF_2	0.01	0
	MF_3	0.01	0

Lectura de los datos de entrada de árbol de desencadenamiento y propagación del fallo En este alcance no es necesario especificar árboles de desencadenamiento y propagación del fallo por lo cual la herramienta automaticamente crea un árbol de desencadenamiento y propagación del fallo vacío.

```
In [18]: # Lectura de los datos de entrada de arbol de fallo
         (de_arbol_fallo) = datos_entrada_arbol_fallo(alcance, de_esquema_division_dique,
         ruta_de)
```

1.4 Guardado de las variables

```
In [19]: # Saving the objects:
         dir_data = os.path.join(env.data_path, 'reparacion', 'estudio_soluciones_clase1',
         'var_co_costes_reparacion_01_es_c1.pkl')

         with open(dir_data, 'w') as f:  # Python 3: open(..., 'wb')
             pickle.dump([ruta_de, ruta_ds, alcance, estrategia, de_planta,
         de_esquema_division_dique, de_diagrama_modos, clima_tramos, cadencia,
                         de_tipo_verificacion, de_verificacion_tramos, peralte,
         de_reparacion_necesarios, de_reparacion_disponibles, de_arbol_fallo], f)
```

Continuar con Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil..

co_costes_reparacion_01_pi

April 27, 2018

1 Ejemplo para el cálculo de costes de reparacion en el grado de desarrollo de proyecto de inversión: Parte 1 - Lectura de los datos de entrada.

1.1 Importación de paquetes de IPython

In [2]:
```
# Python 2/3 setup
from __future__ import (absolute_import, division, print_function, unicode_literals)
# from builtins import *
```

In [3]:
```
# Juypyter setup

%matplotlib inline
import os
import sys
from IPython.display import HTML
```

In [4]:
```
sim_path = os.path.join(env.modules_path ,'AP_PI_reparacion_est_conservadora', 'datos',
'sim_01')

if sim_path not in sys.path:
    sys.path.append(sim_path)
```

1.2 Importación de paquetes para el ejemplo

In [5]:
```
import logging
import os
import pickle

from reparacion.datos_entrada import datos_entrada_planta
from reparacion.datos_entrada import datos_entrada_diagrama_modos
from reparacion.datos_entrada import datos_entrada_arbol_fallo
from reparacion.datos_entrada import datos_entrada_esquema_division_dique
from reparacion.datos_entrada import datos_entrada_clima_tramos
from reparacion.datos_entrada import datos_entrada_verificacion_dique

from reparacion.calculos import extraccion_resultados
from reparacion.calculos import calculo_costes

from reparacion.datos_entrada import datos_entrada_elementos_reparacion_necesarios
from reparacion.datos_entrada import datos_entrada_elementos_reparacion_disponibles
from reparacion.datos_entrada import datos_entrada_tipo_verificacion
```

NOTA: MODIFICACIÓN MODULES PATH

1.3 Datos de entrada de la forma en planta

1.3.1 RUTA DE DIRECTORIOS DE DATOS DE ENTRADA Y SALIDA

```
In [6]: # Ruta con los datos de entrada
        ruta_de = os.path.join(env.input_path, 'reparacion', 'proyecto_inversion')
        # Ruta con los datos de salida
        ruta_ds = os.path.join(env.output_path, 'reparacion', 'proyecto_inversion')
```

1.3.2 INTRODUCCION

De acuerdo con el articulado de la ROM 1.1 en el grado de desarrollo de proyecto de inversión se deben calcular los costes completos de las fases de reparación del dique a lo largo de la vida útil. Para el cálculo de estos costes se deberán considerar lo siguiente: - El cálculo de costes deberá realizarse mediante la verificación simultánea de todos los modos de fallo de los distintos componentes del dique. - Se considearará un diseño a Estado Límite de Servicio en el que la avería de los distintos modos podrá progresar en el tiempo utilizando para ello una curva de acumulación de daño. - La verificación deberá realizarse en cada uno de los estados de las series climáticas debiendose considerar tanto los periodos de calma como los ciclos de solicitación - Se deberán definir las estrategias de reparación en su máximo nivel de precisión sin permitir ambigüedad. - El método para verificar el comportamiento de la obra en el este alcance es por métodos de Nivel III empleándose técnicas de simulación mediante técnicas de Monte Carlo para obtener nuevas series climáticas y repetir así la verificación un número elevado de veces que permita acotar la incertidumbre asociadas a los costes.

A modo de resumen, el cálculo de costes de reparación en este grado de desarrollo: - Considera un número elevado de series climáticas para obtner así los costes medios de reparación junto con su variabilidad asociada - No considera la reparación inmediata de los daños sufridos en cada modo, sino que el daño progresa en el tiempo hasta que es den las condiciones necesarias para efectuar las labores de reparación - Se podrán definir varias estrategias de reparación en función de los parámetros de entrada a definir - Se considera que el progreso de la avería en uno de los modos puede inducir el desencadenamiento del fallo en otros modos. - Se podrá diferencias entre elementos necesarios para la reparación y elementos disponibles en puerto de forma permanente para efectuar los trabajos de reparación. - En la verificación de los modos a ELS se considera la evolución del daño mediante las curvas de acumulación de daño. - Se considerará que le dique puede fallar en su conjunto mediante las diferentes configuraciones de fallo definidas en el diagrama de componentes.

A modo de ejemplo se muestra a continuación los datos de entrada, la verificación mediante simulación numérica y la obtención de los costes de reparación para una de las 25 simulaciones consideradas. Finalmente se mostrarán algunas gráficas con los resultados integrados de las 25 simulación

1.3.3 DEFINICIÓN DEL ALCANCE Y ESTRATEGIA

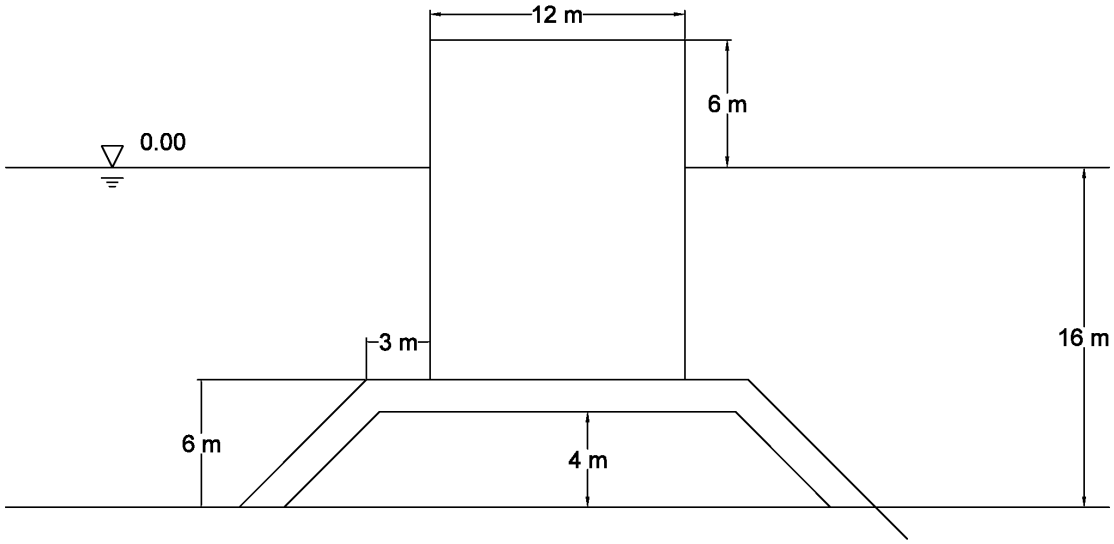
```
In [7]: alcance = 'PI'

        #estrategia = 'no_reparacion'
        #estrategia = 'reparacion_inmediata'
        estrategia = ''
```

1.3.4 DEFINICIÓN DE LOS DATOS DE ENTRADA

En este ejemplo se propone analizar el coste de un dique de tipo mixto con berma baja de protección (Tipo C, LMB). La sección tipo del dique a analizar se muestra en la figura. En este ejemplo, se considera para la forma en planta un dique compuesto por una única alineación principal de 500 m de longitud. El calado de la alineación principal es de 16 m respecto al nivel medio del mar, la altura de la berma es de 4 m y los cajones tienen una altura de 18 m por 12 m de ancho.

```
In [8]: from IPython.display import Image
        img_name = os.path.join(ruta_de, 'imagenes', 'esquema_dique_c.png')
        Image(filename=img_name, width=800)
```

Out[8]:



Lectura de los ficheros con los datos de entrada de tipología y forma en planta (Para modificar los datos utilizados en el ejemplo se deben modificar los ficheros de la carpeta de inputs)

A modo de ejemplo se simulará una vida útil de 5 años.

```
In [9]: # Lectura de los datos de entrada en planta
        (de_planta) = datos_entrada_planta(ruta_de)

        de_planta
```

Out[9]:

	longitud	vida_util	calado
T_0	500.0	5.0	15.0

Lectura de los ficheros con los datos de entrada del esquema de componentes De acuerdo con el articulado la estructura jerárquica del dique se descompone en: tramos, subsistemas y modos de fallo. En este ejemplo teórico se muestra el cálculo de costes de un tramo compuesto por un subsistema con un único modo de fallo.

```
In [10]:    # lectura del esquema de division del dique
            (de_esquema_division_dique) = datos_entrada_esquema_division_dique(ruta_de)
            de_esquema_division_dique
```

```
Out[10]:   tramo subsistema modo_fallo
         0   T_0       SS_0       MF_3
```

Lectura de los ficheros con los datos de entrada del diagrama de componentes De acuerdo con el articulado de la ROM 1.1. los diagramas de componentes expresan las posibles combinaciones de elementos del espacio de sucesos que suponen el incumplimiento de los requisitos del sistema. Los tipos de diagramas pueden ser: tipo serie (en el que el sistema falla si lo hace al menos uno de los componentes), tipo paralelo (el sistema falla si lo hacen todos los componentes) o tipo mixto (mezcla de ambos).

Para este ejemplo se ha definido un diagrama para cada estructura jerárquica del dique:

- Diagrama de modos del dique: En el que se indican las posibles combinaciones de tramos que deben fallar para considera el fallo del dique. En este ejemplo se considera que el dique falla si falla el tramo T_0.
- Diagrama de modos de los tramos: En el que se indican las posibles combinaciones de subsistemas que deben fallas para considerar el fallo del tramo. En este ejemplo se considera que el tramo falla si falla el subsistema SS_0.
- Diagrama de modos de los subsistemas: En el que se indican las posibles combinaciones de modos que deben fallar para considerar el fallo de un subsistema. Para este ejemplo se ha considerado un diagrama de modos en el que el fallo del subsistema se produce si falla el MF_3.

```
In [11]: # Lectura de los datos de entrada del diagrama de modos
         (de_diagrama_modos) = datos_entrada_diagrama_modos(ruta_de)
         de_diagrama_modos['de_diagrama_modos_dique']
```

```
Out[11]:            opciones_de_destruccion_total
           Dique  op1                        ['T_0']
```

```
In [12]: de_diagrama_modos['de_diagrama_modos_tramos']
```

```
Out[12]:         opciones_de_destruccion_total
           T_0 op1                       ['SS_0']
```

```
In [13]: de_diagrama_modos['de_diagrama_modos_subsistemas']
```

```
Out[13]:              opciones_de_destruccion_total
           T_0 SS_0 op2                        ['MF_3']
```

Lectura de los datos climáticos Dentro de la carpeta 'clima' es necesario que haya una carpeta por cada uno de los tramos que componen el dique con el nombre de la etiqueta del tramo (en este caso únicamente uno). Dentro de cada carpeta se debe encontrar un fichero denominado 'clima.txt' con las series temporales históricas propagadas al emplazamiento de cada uno de los agentes necesarios para realizar la verificación del dique. Las series temporales se estructurarán por columnas.

La matriz con los datos de clima tendrán un valor de nivel (respecto al NMM), velocidad de viento, altura de ola, periodo pico, dirección media de procedendia del oleaje y dirección media de procedencia del viento para cada hora.

Se muestra a continuación un ejemplo del fichero de clima:

```
In [14]: # Lectura de los datos de clima para cada tramo
         (clima_tramos, cadencia) = datos_entrada_clima_tramos(de_esquema_division_dique,
         de_planta, ruta_de)

         # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(10, 50)
         display(clima_tramos['T_0'])
         # Cambio el numero de filas y columnas a sus valores por defecto
         env.pandas_setup()
```

```
           nivel         vv        hs         tp          dh          dv  \
0      -0.187445   3.531673  0.572556   7.974521  260.433608  221.654369
1       0.464185   7.230433  0.772869   9.380499  264.472435  352.096121
2       0.125255   1.403388  0.596432   7.893324  256.241187  140.491183
3      -0.523136   4.437268  1.258365   9.176988  262.288329  171.496462
4      -0.261880   2.736062  0.552534  10.674567  254.761802   64.307393
...          ...        ...       ...        ...         ...         ...
29211  -0.036775   5.943159  0.046617  10.899674  238.902541   22.980948
29212  -0.573777  10.167237  0.036173   7.895515  250.807646  111.394604
29213  -0.162674   7.576738  0.146799   6.737357  253.179053   25.544986
29214   0.464816   4.531977  0.306809   5.750664  258.426859  325.263397
29215   0.039940   7.444838  0.284937   5.000000  255.795818  120.747631

            mas       mme     calado
0      1.595942 -0.057643  14.812555
1      2.258235 -0.068305  15.464185
2      1.912683 -0.061684  15.125255
3      1.257457 -0.054848  14.476864
4      1.521742 -0.057878  14.738120
...         ...       ...        ...
29211  1.758231 -0.069262  14.963225
29212  1.215802 -0.063835  14.426223
29213  1.630499 -0.067428  14.837326
29214  2.265278 -0.074718  15.464816
29215  1.833988 -0.068303  15.039940

[29216 rows x 9 columns]
```

Lectura de datos de entrada del tipo de verificación para cada modo de fallo De acuerdo con el Articulado de la ROM 1.1 en el alcance de proyecto de inversión se deben calcular los costes completos de reparación a partir de un diseño de ELS según la configuración del diagrama de modos de fallo del dique. Por lo tanto dentro de la carpeta 'tipo_verificacion_curva_acum_dano' el fichero 'tipo_verificacion_curva_acum_dano.txt' le indicará a la herramienta que cada modo de fallo se verifica a ELS y por lo tanto será necesario especificar los parámetros de la curva de acumulación de daño. Si se desease realizar una verificación a ELU bastaría con modificar el parámetro a ELU e igualar para cada modo los parámetros a, b y c de curva de acumulación de daño a 0.

Se muestra a continuación un ejemplo con los ficheros de entrada para el tipo de verificación para cada modo de fallo:

```
In [15]: # Tipo de verificacion y curva de acumulacion de dano
         de_tipo_verificacion = datos_entrada_tipo_verificacion(ruta_de)

         de_tipo_verificacion
```

Out[15]:

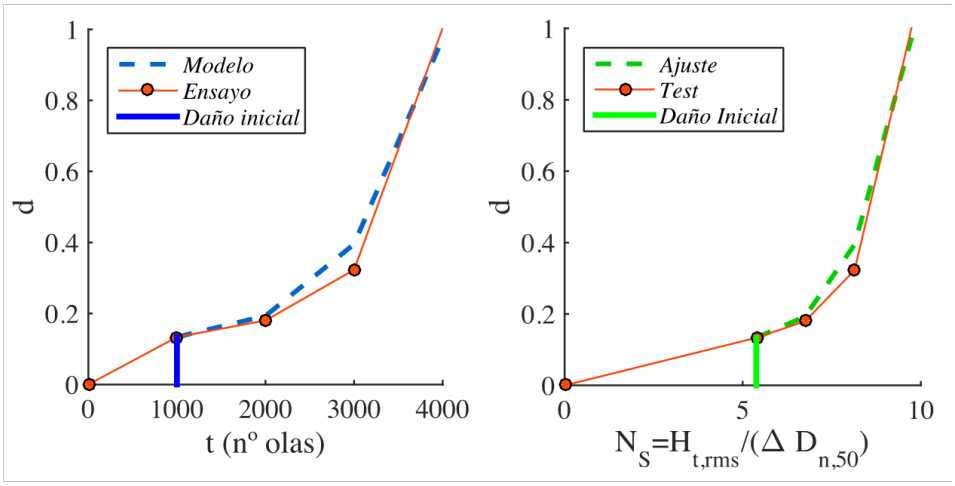
		tipo_diseno	par_a	par_b	par_c
T_0	SS_0	MF_3	els	0.00013	0.3935	3.1816

Se muestra a continuación la curva de acumulación de daño utilizada junto con sus parámetros de ajuste para este ejemplo.

```
In [16]: from IPython.display import Image
         img_name = os.path.join(ruta_de, 'imagenes', 'curva_acumulacion_daño.png')
         Image(filename=img_name, width=800)
```

Out[16]:

Curva de acumulación de daño para el modo de fallo



$$d(d_0, N_s, t) = \left[d_0^{1/b} + (aN_s^c)^{1/b}\, t \right]^b$$

Modo de fallo
Tipo de verificación: ELS
Parámetro a: 1.2955e-4
Parámetro b: 0.3935
Parámetro c: 3.1816

Además se debe leer los resultados del módulo de verificación en el que se indica para cada estado de mar de la vida útil si cada uno de los modos de fallo fallan o no fallan

```
In [17]:     # Verificacion tramos del dique
             (de_verificacion_tramos, peralte) = datos_entrada_verificacion_dique(ruta_de,
         de_esquema_division_dique)
             de_verificacion_tramos.loc[:, ('T_0', 'SS_0', 'MF_3')]
```

```
Out[17]: 0        False
         1        False
         2        False
         3        False
         4        False
                  ...
         14595    False
         14596    False
         14597    False
         14598    False
         14599    False
         Name: (T_0, SS_0, MF_3), Length: 14600, dtype: object
```

Lectura de los datos de entrada con la estrategia de reparación Para cada modo de fallo es necesario especificar los distintos parámetros de reparación que definen la estrategia, entre los que se incluyen los siguientes: - Nivel de avería umbral para iniciar la reparción: Se considera un valor de 0.5 - Nivel de avería umbral para finalizar la reparación: Se considera un valor de 0 - Maquinaria necesaria para llevar a cabo la reparación del modo - Tipo de maquinaria: Camión y gánguil - Tiempo necesario que debe transcurrir desde que se da la orden de reparar hasta que se consigue la maquinaria necesaria: 24 h - Umbral de operatividad de la maquinaria: Para el gánguil se considera que éste puede trabajar siempre que la altura de ola sea menor que 1.5 m y el calado superior a 3 m. - Costes unitario por hora de trabajo de cada máquina: 25 y 80 €$\,h^{-1}$ respectivamente. - Materiales necesarios para llevar a cabo la reparación del modo - Tipo de materiales: Todo uno de cantera - Cantidad total de materiales necesarios durante la reparación del modo: 300 m^3 - Tiempo necesario que debe transcurrir desde que se da la orden de reparar hasta que se consigue los materiales necesarios: 0 h - Coste total de los materiales durante un evento de reparación: 3000 € - Mano de obra necesaria para llevar a cabo la reparación del modo - Tipo de mano de obra: Operarios - Tiempo necesario que debe transcurrir desde que se da la orden de reparar hasta que se consigue la mano de obra necesaria: 24 h - Costes unitario por hora de trabajo de cada mano de obra: 10 €$\,h^{-1}$ respectivamente. - Rendimiento de reparación de la avería en base a los medios definidos: 0.001 (tanto por uno) de avería reparado en cada hora.

No se consideran elementos permanentes en puerto para los trabajos de reparación.

```
In [18]:     # Datos de entrada de reparacion necesario
             de_reparacion_necesarios = datos_entrada_elementos_reparacion_necesarios(ruta_de,
         cadencia, alcance,
                                                                                       estrategia)
             # Datos de entrada de reparacion disponibles
             de_reparacion_disponibles = datos_entrada_elementos_reparacion_disponibles(ruta_de,
         cadencia, alcance)

             # Cambio el numero de filas y columnas que se muestran
             env.pandas_setup(10, 50)
             display(de_reparacion_necesarios)
             # Cambio el numero de filas y columnas a sus valores por defecto
             env.pandas_setup()
```

```
                 na_umbral_ini_reparacion na_umbral_fin_reparacion  \
T_0 SS_0 MF_3                         0.3                         0

                           tipos_maq num_maq           tipos_mat cant_mat  \
T_0 SS_0 MF_3  [ganguil, camion]   [2, 5]  [todo_uno_cantera]    [300]
```

```
                  tipos_mo num_mo t_maq_ini_rep t_mat_ini_rep t_mo_ini_rep  \
T_0 SS_0 MF_3  [operarios]   [15]            24             0           24

                                                          umb_ope   rend  \
T_0 SS_0 MF_3  {'calado': 3, 'hs': 1.5, 'vv': 100, 'nivel': 100}  0.001

              coste_maq_rep coste_mat_rep coste_mo_rep
T_0 SS_0 MF_3           855          3000          450
```

Lectura de los datos de entrada de árbol de desencadenamiento y propagación del fallo En este alcance es necesario especificar árboles de desencadenamiento y propagación del fallo. Sin embargo, al tratarse de un ejemplo en el que sólo se considera un modo de fallo, el progreso de la avería en dicho modo no puede desencadenar el fallo en otros modos.

```
In [19]: # Lectura de los datos de entrada de arbol de fallo
         (de_arbol_fallo) = datos_entrada_arbol_fallo(alcance, de_esquema_division_dique,
         ruta_de)
         de_arbol_fallo
```

```
Out[19]:               modos nivel_averia
          T_0 SS_0 MF_3    []           []
```

1.4 Guardado de las variables

```
In [20]: # Saving the objects:
         dir_data = os.path.join(env.data_path, 'reparacion', 'proyecto_inversion',
         'var_co_costes_reparacion_01_pi.pkl')

         with open(dir_data, 'w') as f:  # Python 3: open(..., 'wb')
             pickle.dump([ruta_de, ruta_ds, alcance, estrategia, de_planta,
         de_esquema_division_dique, de_diagrama_modos, clima_tramos, cadencia,
                         de_tipo_verificacion, de_verificacion_tramos, peralte,
         de_reparacion_necesarios, de_reparacion_disponibles, de_arbol_fallo], f)
```

Continuar con Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil..

co_costes_reparacion_02_es_c1

April 27, 2018

1 Ejemplo para el cálculo de costes de reparacion en el alcance de estudios soluciones clase I: Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil

Este ejemplo continuará a partir de los datos de entrada guardados al final del apartado Parte 1 - Lectura de los datos de entrada.

De acuerdo con el Articulado de la ROM 1.1, la verificación de la fase de conservación - reparación del dique se realiza en cada uno de los estados de que componen la vida útil. Se debe verificar de forma simultánea la seguridad frente a la totalidad de los modos de fallo principales. El esquema seguido para la verificación se muestran en el documento del Manual de la ROM 1.1. La verificación en este alcance se realiza una vez a partir de las series históricas de los agentes propagadas a cada uno de los tramo del dique.

1.1 Importación de paquetes de IPython

```
In [2]: # Python 2/3 setup
        from __future__ import (absolute_import, division, print_function, unicode_literals)
        # from builtins import *
```

```
In [3]: # Juypyter setup

        %matplotlib inline
        import os
        import sys
        import pickle
        import ast
        import pandas as pd
        env.pandas_setup()
        from IPython.display import HTML
```

1.2 Carga de variables del apartado anterior

```
In [4]: # Getting back the objects:
        dir_data = os.path.join(env.data_path, 'reparacion', 'estudio_soluciones_clase1',
        'var_co_costes_reparacion_01_es_c1.pkl')

        with open(dir_data) as f:  # Python 3: open(..., 'rb')
            ruta_de, ruta_ds, alcance, estrategia, de_planta, de_esquema_division_dique,
        de_diagrama_modos, clima_tramos, cadencia, de_tipo_verificacion, de_verificacion_tramos,
        peralte, de_reparacion_necesarios, de_reparacion_disponibles, de_arbol_fallo =
        pickle.load(f)
```

1.3 Importación de paquetes para el ejemplo

In [5]:
```
import logging

from reparacion.calculos import calculo_final_simulacion
from reparacion.calculos import inicializacion_matrices_grado_averia

from reparacion.calculos import estado_de_mar_tramo
from reparacion.calculos import actuliza_nivel_de_averia
from reparacion.clasificacion_main_ciclos import clasificacion_main_ciclos

from tqdm import tqdm
```

1.3.1 RUTA PARA LA CREACIÓN DEL FICHERO DE LOG

In [6]:
```
direct = os.path.join(ruta_ds, 'debug_info.log')
logging.basicConfig(filename=direct, level=logging.INFO)
```

1.4 Verificación de la estrategia de reparación a lo largo de la vida útil

In [7]:
```
# Calculo de la hora final de simulacion
h_fin_sim = calculo_final_simulacion(de_planta, clima_tramos, cadencia)

# Inicializar las matrices de grado de averia para el dique, tramos, subsistemas y modos
de fallo
(ia_dique, ia_tramos, ia_subsistemas, ia_modos_fallo, averia_estado, averia_acum_estado,
 estado_modos_fallo, subsistemas, tramos, modos_fallo, estado_reparacion_modos,
 origen_maquinaria_reparacion_estado, origen_materiales_reparacion_estado,
 origen_mano_obra_reparacion_estado,
 datos_salida,
 datos_salida_prob_ini_averia) =
inicializacion_matrices_grado_averia(de_esquema_division_dique, h_fin_sim)

 # Inicializacion de las matrices de materiales de reparacion que quedan disponibles en
puerto y matrices de si
 # los elementos de reparacion se toman del puerto o se encargan para cada estado

# Recorrido por los estados de la vida util
for h in tqdm(range(1, h_fin_sim)):

    # Recorrido por los modos de fallo principales de cada subsistema de cada tramo
    for mst in de_esquema_division_dique.iterrows():
        # Obtencion del tramo, subsistema y modo de fallo
        mst = mst[1]
        tr = mst['tramo']
        ss = mst['subsistema']
        mf = mst['modo_fallo']
        # Se obtienen los datos del estado de mar para el tramo
        (hs, tp, u10, z, p90) = estado_de_mar_tramo(clima_tramos, tr, h)

        logging.info('Estado de mar. Hs: ' + str(hs))

#            if mf == 'MF_4':
#                print mf

        # Actualizacion del nivel de averia acumulado hasta este estado
        (averia_estado, averia_acum_estado) = actuliza_nivel_de_averia(averia_estado,
averia_acum_estado, tr, ss,
                                                                       mf, h)
        # En primer lugar se comprueba si el estado corresponde a ciclo de solicitacion
o no
        [datos_salida, averia_acum_estado, averia_estado, estado_modos_fallo,
origen_maquinaria_reparacion_estado,
         origen_materiales_reparacion_estado, origen_mano_obra_reparacion_estado] = \
               clasificacion_main_ciclos(hs, tp, u10, z, p90, h, tr, ss, mf,
averia_acum_estado, averia_estado, de_reparacion_necesarios,
```

```
                    de_reparacion_disponibles, estado_reparacion_modos,
        origen_maquinaria_reparacion_estado,
                    origen_materiales_reparacion_estado,
        origen_mano_obra_reparacion_estado, alcance,
                    estrategia, datos_salida, estado_modos_fallo, ia_modos_fallo,
        cadencia, de_verificacion_tramos,
                    peralte, de_esquema_division_dique, de_arbol_fallo,
        de_tipo_verificacion, clima_tramos, de_diagrama_modos,
                    ia_tramos, ia_subsistemas, subsistemas, ia_dique, tramos,
        datos_salida_prob_ini_averia)


            # Al final de cada iteracion actualizo los inicios de averia de los susbsistemas y
        tramos si alguno de los
            # elementos que lo conforman se encuentra en inicio de averia y si todos estan sin
        averia, el sussitema o
            # tramo deja de tener inicio de averia

                # Inicio de averia en subsistemas
                for s in subsistemas:
                    if all(ia_modos_fallo.loc[(slice(None), s), 'ini_averia'] == 0):
                        ia_subsistemas.loc[(slice(None), s), 'ini_averia'] = 0
                    elif any(ia_modos_fallo.loc[(slice(None), s), 'ini_averia'] == 1):
                        ia_subsistemas.loc[(slice(None), s), 'ini_averia'] = 1

                # Inicio de averia en tramos
                for t in tramos:
                    if all(ia_subsistemas.loc[t, 'ini_averia'] == 0):
                        ia_tramos.loc[t, 'ini_averia'] = 0
                    elif any(ia_subsistemas.loc[t, 'ini_averia'] == 1):
                        ia_tramos.loc[t, 'ini_averia'] = 1

            # Al final de cada iteracion se comprueba si el dique ha fallado, en caso afirmativo
        se corta la simulacion
            if ia_dique.ix[0, 'dest_total'] == 1:
                logging.info('Estado: ' + str(h) + ' Tramo: ' + str(tr) + ' Subsistema: ' + str(
                    ss) + ' Modo_fallo: ' + str(mf) + ' El dique sufre destruccion total')
                break
```

```
  0%|          | 0/2919 [00:00<?, ?it/s]C:\Users\GDFA-JUAN\Anaconda2\lib\site-
packages\ipykernel_launcher.py:66: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
100%|| 2919/2919 [01:04<00:00, 45.55it/s]
```

1.5 Guardado de variables

```
In [8]: # Saving the objects:
        dir_data = os.path.join(env.data_path, 'reparacion', 'estudio_soluciones_clase1',
        'var_co_costes_reparacion_02_es_c1.pkl')

        with open(dir_data, 'w') as f:  # Python 3: open(..., 'wb')
            pickle.dump([datos_salida, averia_acum_estado, averia_estado, estado_modos_fallo,
        origen_maquinaria_reparacion_estado,
                         origen_materiales_reparacion_estado,
        origen_mano_obra_reparacion_estado,
                         datos_salida_prob_ini_averia], f)
```

Continuar con Parte 3 - Representacion de resultados y cálculo de costes.

co_costes_reparacion_02_pi

April 27, 2018

1 Ejemplo para el cálculo de costes de reparacion en el grado de desarrollo de proyecto de inversión: Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil

Este ejemplo continuará a partir de los datos de entrada guardados al final del apartado Parte 1 - Lectura de los datos de entrada.

De acuerdo con el Articulado de la ROM 1.1, la verificación de la fase de conservación - reparación del dique se realiza en cada uno de los estados de que componen la vida útil. Se debe verificar de forma simultánea la seguridad frente a la totalidad de los modos de fallo principales. El esquema seguido para la verificación se muestran en el documento del Manual de la ROM 1.1. La verificación en este alcance se realiza un número elevado de veces, una por cada serie climática simulada y propagada al emplazamiento. En este ejemplo se muestra la verificación de una de las series climáticas simuladas. Del mismo modo se realizaría para el resto de series climáticas.

1.1 Importación de paquetes de IPython

```
In [10]: # Python 2/3 setup
         from __future__ import (absolute_import, division, print_function, unicode_literals)
         # from builtins import *

In [11]: # Juypyter setup

         %matplotlib inline
         import os
         import sys
         import pickle
         import ast
         import pandas as pd
         env.pandas_setup()
         from IPython.display import HTML

In [12]: sim_path = os.path.join(env.modules_path ,'AP_PI_reparacion_est_conservadora', 'datos',
         'sim_01')

         if sim_path not in sys.path:
             sys.path.append(sim_path)
```

1.2 Carga de variables del apartado anterior

```
In [13]: # Getting back the objects:
         dir_data = os.path.join(env.data_path, 'reparacion', 'proyecto_inversion',
         'var_co_costes_reparacion_01_pi.pkl')

         with open(dir_data) as f:  # Python 3: open(..., 'rb')
```

```
    ruta_de, ruta_ds, alcance, estrategia, de_planta, de_esquema_division_dique,
de_diagrama_modos, clima_tramos, cadencia, de_tipo_verificacion, de_verificacion_tramos,
peralte, de_reparacion_necesarios, de_reparacion_disponibles, de_arbol_fallo =
pickle.load(f)
```

1.3 Importación de paquetes para el ejemplo

```
In [14]: import logging

         from reparacion.calculos import calculo_final_simulacion
         from reparacion.calculos import inicializacion_matrices_grado_averia

         from reparacion.calculos import estado_de_mar_tramo
         from reparacion.calculos import actuliza_nivel_de_averia
         from reparacion.clasificacion_main_ciclos import clasificacion_main_ciclos

         from tqdm import tqdm
```

1.3.1 RUTA PARA LA CREACIÓN DEL FICHERO DE LOG

```
In [15]: direct = os.path.join(ruta_ds, 'debug_info.log')
         logging.basicConfig(filename=direct, level=logging.INFO)
```

1.4 Verificación de la estrategia de reparación a lo largo de la vida útil

```
In [16]: # Calculo de la hora final de simulacion
         h_fin_sim = calculo_final_simulacion(de_planta, clima_tramos, cadencia)

         # Inicializar las matrices de grado de averia para el dique, tramos, subsistemas y modos
         de fallo
         (ia_dique, ia_tramos, ia_subsistemas, ia_modos_fallo, averia_estado, averia_acum_estado,
          estado_modos_fallo, subsistemas, tramos, modos_fallo, estado_reparacion_modos,
          origen_maquinaria_reparacion_estado, origen_materiales_reparacion_estado,
          origen_mano_obra_reparacion_estado,
          datos_salida,
          datos_salida_prob_ini_averia) =
         inicializacion_matrices_grado_averia(de_esquema_division_dique, h_fin_sim)

          # Inicializacion de las matrices de materiales de reparacion que quedan disponibles en
         puerto y matrices de si
          # los elementos de reparacion se toman del puerto o se encargan para cada estado

         # Recorrido por los estados de la vida util
         for h in tqdm(range(1, h_fin_sim)):

             # Recorrido por los modos de fallo principales de cada subsistema de cada tramo
             for mst in de_esquema_division_dique.iterrows():
                 # Obtencion del tramo, subsistema y modo de fallo
                 mst = mst[1]
                 tr = mst['tramo']
                 ss = mst['subsistema']
                 mf = mst['modo_fallo']
                 # Se obtienen los datos del estado de mar para el tramo
                 (hs, tp, u10, z, p90) = estado_de_mar_tramo(clima_tramos, tr, h)

                 logging.info('Estado de mar. Hs: ' + str(hs))

         #            if mf == 'MF_4':
         #                print mf

                 # Actualizacion del nivel de averia acumulado hasta este estado
                 (averia_estado, averia_acum_estado) = actuliza_nivel_de_averia(averia_estado,
         averia_acum_estado, tr, ss,
                                                                                mf, h)
                 # En primer lugar se comprueba si el estado corresponde a ciclo de solicitacion
```

```
        o no
                [datos_salida, averia_acum_estado, averia_estado, estado_modos_fallo,
        origen_maquinaria_reparacion_estado,
                 origen_materiales_reparacion_estado, origen_mano_obra_reparacion_estado] = \
                      clasificacion_main_ciclos(hs, tp, u10, z, p90, h, tr, ss, mf,
        averia_acum_estado, averia_estado, de_reparacion_necesarios,
                           de_reparacion_disponibles, estado_reparacion_modos,
        origen_maquinaria_reparacion_estado,
                           origen_materiales_reparacion_estado,
        origen_mano_obra_reparacion_estado, alcance,
                           estrategia, datos_salida, estado_modos_fallo, ia_modos_fallo,
        cadencia, de_verificacion_tramos,
                           peralte, de_esquema_division_dique, de_arbol_fallo,
        de_tipo_verificacion, clima_tramos, de_diagrama_modos,
                           ia_tramos, ia_subsistemas, subsistemas, ia_dique, tramos,
        datos_salida_prob_ini_averia)


            # Al final de cada iteracion actualizo los inicios de averia de los susbsistemas y
        tramos si alguno de los
            # elementos que lo conforman se encuentra en inicio de averia y si todos estan sin
        averia, el sussitema o
            # tramo deja de tener inicio de averia

                # Inicio de averia en subsistemas
                for s in subsistemas:
                    if all(ia_modos_fallo.loc[(slice(None), s), 'ini_averia'] == 0):
                        ia_subsistemas.loc[(slice(None), s), 'ini_averia'] = 0
                    elif any(ia_modos_fallo.loc[(slice(None), s), 'ini_averia'] == 1):
                        ia_subsistemas.loc[(slice(None), s), 'ini_averia'] = 1

                # Inicio de averia en tramos
                for t in tramos:
                    if all(ia_subsistemas.loc[t, 'ini_averia'] == 0):
                        ia_tramos.loc[t, 'ini_averia'] = 0
                    elif any(ia_subsistemas.loc[t, 'ini_averia'] == 1):
                        ia_tramos.loc[t, 'ini_averia'] = 1

            # Al final de cada iteracion se comprueba si el dique ha fallado, en caso afirmativo
        se corta la simulacion
            if ia_dique.ix[0, 'dest_total'] == 1:
                logging.info('Estado: ' + str(h) + ' Tramo: ' + str(tr) + ' Subsistema: ' + str(
                    ss) + ' Modo_fallo: ' + str(mf) + ' El dique sufre destruccion total')
                break
```

```
  0%|          | 0/14599 [00:00<?, ?it/s]C:\Users\GDFA-JUAN\Anaconda2\lib\site-
packages\ipykernel_launcher.py:66: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  7%|         | 1059/14599 [00:41<08:52, 25.42it/s]
```

1.5 Guardado de variables

```
In [ ]: # Saving the objects:
        dir_data = os.path.join(env.data_path, 'reparacion', 'proyecto_inversion',
        'var_co_costes_reparacion_02_pi.pkl')

        with open(dir_data, 'w') as f:  # Python 3: open(..., 'wb')
            pickle.dump([datos_salida, averia_acum_estado, averia_estado, estado_modos_fallo,
        origen_maquinaria_reparacion_estado,
                         origen_materiales_reparacion_estado,
```

```
    origen_mano_obra_reparacion_estado,
                    datos_salida_prob_ini_averia], f)
```

Continuar con Parte 3 - Representacion de resultados y cálculo de costes.

co_costes_reparacion_03_es_c1

April 27, 2018

1 Ejemplo para el cálculo de costes de reparacion en el alcance de estudios soluciones clase I: Parte 3 - Representación de resultados y cálculo de costes

Este ejemplo continuará a partir de los datos guardados tras la verificación de la estrategia de reparación al final del apartado Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil.

1.1 Importación de paquetes de IPython

In [2]:
```
# Python 2/3 setup
from __future__ import (absolute_import, division, print_function, unicode_literals)
# from builtins import *
```

In [3]:
```
# Juypyter setup

%matplotlib inline
import os
import sys
import pickle
import ast
import pandas as pd
env.pandas_setup()
from IPython.display import HTML
```

1.2 Carga de variables del apartado anterior

In [4]:
```
# Getting back the objects:
dir_data = os.path.join(env.data_path, 'reparacion', 'estudio_soluciones_clase1',
'var_co_costes_reparacion_01_es_c1.pkl')

with open(dir_data) as f:  # Python 3: open(..., 'rb')
    ruta_de, ruta_ds, alcance, estrategia, de_planta, de_esquema_division_dique,
de_diagrama_modos, clima_tramos, cadencia, de_tipo_verificacion, de_verificacion_tramos,
peralte, de_reparacion_necesarios, de_reparacion_disponibles, de_arbol_fallo =
pickle.load(f)


# Getting back the objects:
dir_data = os.path.join(env.data_path, 'reparacion', 'estudio_soluciones_clase1',
'var_co_costes_reparacion_02_es_c1.pkl')

with open(dir_data) as f:  # Python 3: open(..., 'rb')
    datos_salida, averia_acum_estado, averia_estado, estado_modos_fallo,
origen_maquinaria_reparacion_estado, origen_materiales_reparacion_estado,
origen_mano_obra_reparacion_estado, datos_salida_prob_ini_averia = pickle.load(f)
```

1.3 Importación de paquetes para el ejemplo

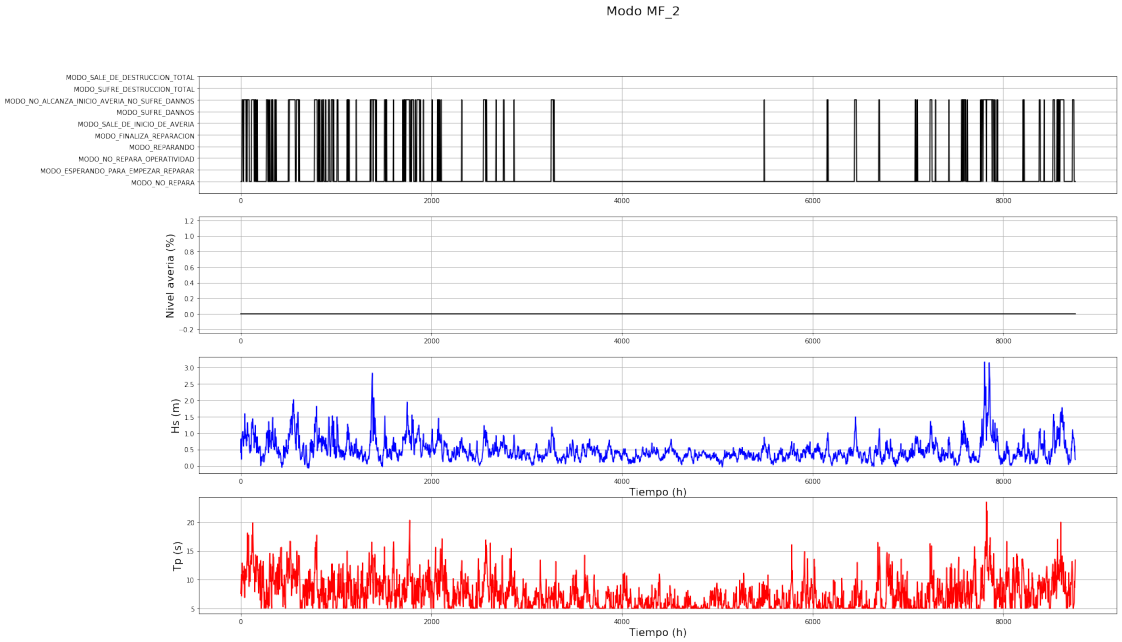
```
In [5]: import matplotlib.pyplot as plt
        from reparacion.calculos import calculo_final_simulacion
        from reparacion.calculos import extraccion_resultados
        from reparacion.calculos import calculo_costes
```

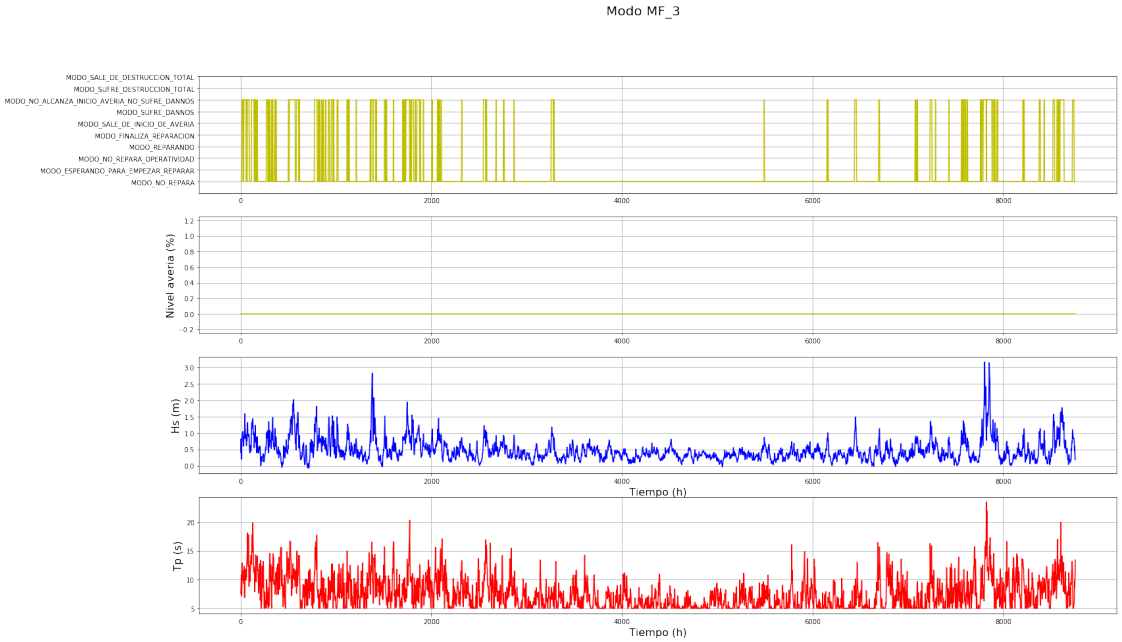
1.4 Representación de los resultados de la simulación numérica

```
In [6]: # Calculo de la hora final de simulacion
        h_fin_sim = calculo_final_simulacion(de_planta, clima_tramos, cadencia)

        # Representacion grafica
        col = ['r', 'g', 'k', 'y', 'b', 'c', 'm', 'r', 'g', 'k', 'y', 'b', 'c', 'm', 'r', 'g',
        'k', 'y', 'b', 'c']
        display = ['MF_0', 'MF_1', 'MF_2', 'MF_3', 'MF_4', 'MF_5', 'MF_6', 'MF_7', 'MF_8',
        'MF_9', 'MF_10', 'MF_11',
        'MF_12', 'MF_13', 'MF_14', 'MF_15', 'MF_16', 'MF_17', 'MF_18', 'MF_19', 'MF_20']

        cont = 1
        for mst in de_esquema_division_dique.iterrows():
            plt.figure(figsize=(25, 15))

            # Obtencion del tramo, subsistema y modo de fallo
            mst = mst[1]
            tr = mst['tramo']
            ss = mst['subsistema']
            mf = mst['modo_fallo']

            plt.suptitle('Modo ' + str(mf), fontsize=20)

            #ax1 = plt.subplot(de_esquema_division_dique.shape[0], 1, cont)
            ax1 = plt.subplot(4, 1, 1)
            x = range(0, cadencia * h_fin_sim, cadencia)
            ax1.plot(x, estado_modos_fallo.loc[(tr, ss, mf), :], color=col[cont],
        label=display[cont])
            plt.grid()
            plt.ylim(-1, 9)
            plt.yticks([0, 1, 2, 3, 4, 5, 6, 7, 8 ,9], [
                'MODO_NO_REPARA', 'MODO_ESPERANDO_PARA_EMPEZAR_REPARAR',
        'MODO_NO_REPARA_OPERATIVIDAD', 'MODO_REPARANDO',
                'MODO_FINALIZA_REPARACION', 'MODO_SALE_DE_INICIO_DE_AVERIA',
        'MODO_SUFRE_DANNOS',
                'MODO_NO_ALCANZA_INICIO_AVERIA_NO_SUFRE_DANNOS', 'MODO_SUFRE_DESTRUCCION_TOTAL',
        'MODO_SALE_DE_DESTRUCCION_TOTAL'])

            ax2 = plt.subplot(4, 1, 2, sharex=ax1)
            x = range(0, cadencia * h_fin_sim, cadencia)
            y = averia_acum_estado.loc[(tr, ss, mf), :]
            ax2.plot(x, y, color=col[cont], label=display[cont])
            plt.ylim(-0.25, 1.25)
            plt.grid()
            plt.ylabel('Nivel averia (%)', fontsize=16)

            ax3 = plt.subplot(4, 1, 3, sharex=ax1)
            x = range(0, cadencia * h_fin_sim, cadencia)
            hs = clima_tramos[tr].ix[0: h_fin_sim - 1, 'hs']
            ax3.plot(x, hs, color='b', label='Hs (m)')
            plt.grid()
            cont += 1
            plt.xlabel('Tiempo (h)', fontsize=16)
            plt.ylabel('Hs (m)', fontsize=16)

            ax4 = plt.subplot(4, 1, 4, sharex=ax1)
            x = range(0, cadencia * h_fin_sim, cadencia)
            tp = clima_tramos[tr].ix[0: h_fin_sim - 1, 'tp']
            ax4.plot(x, tp, color='r', label='Tp (s)')
```

```
        plt.grid()
        plt.xlabel('Tiempo (h)', fontsize=16)
        plt.ylabel('Tp (s)', fontsize=16)

        # Guardado de resultados
        name_fichero = ('Modo_' + str(mf) + '_estados.pdf')
        direct = os.path.join(ruta_ds, name_fichero)

        plt.savefig(direct)
        plt.show()
```

```
C:\Users\GDFA-JUAN\Anaconda2\lib\site-packages\ipykernel_launcher.py:42:
DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```





1.5 Extracción de resultados

```
In [7]: (datos_salida) = extraccion_resultados(datos_salida, datos_salida_prob_ini_averia,
        de_planta,
                                                estado_modos_fallo, h_fin_sim, ruta_ds, cadencia)
```

1.6 Cálculo de costes

Tras la verificación de las fases de reparación, los descriptores obtenidos se utilizan como datos de entrada para la adjudicación de costes siguiendo el esquema descrito en el Manual.

```
In [8]: (datos_salida_modos, datos_salida_total, datos_salida_ea_sencillo) =
        calculo_costes(estado_modos_fallo, de_reparacion_necesarios, de_reparacion_disponibles,
            datos_salida, de_planta, ruta_ds, alcance, estrategia, cadencia)
```

1.7 Presentación de resultados

1.7.1 Resultados de la verificación

Tras la verificación simultánea de los modos de fallo, la herramienta ofrece los siguientes resultados para cada componente de la obra de abrigo: - Vida útil simulada - Número de veces que se produce inicio de avería - Probabilidad de inicio de avería - Número de estados en los que se produce daño en el componente - Número de veces que se inicia la reparación de la componente - Probabilidad de que la componente sufra daños - Duraciones mínima, media y máxima de los eventos de daño en la componente - Duraciones mínima, media y máxima de espera para iniciar los trabajos de reparación una vez dada la orden - Duraciones mínima, media y máxima de los trabajos de reparación - Duraciones mínima, media y máxima de las paradas operativas

Se muestran a continuación los resultados ofrecidos por la herramienta para el modo de fallo MF_1

```
In [9]: # Cambio el numero de filas y columnas que se muestran
        env.pandas_setup(50, 50)
        datos_salida['datos_salida_modos'].loc['MF_1', :]
        # Cambio el numero de filas y columnas que se muestran a su valor por defecto
        env.pandas_setup()
```

1.7.2 Resultados de la adjudicación de costes de reparación

Tras la adjudicación de costes, en este alcance la herramienta ofrece el valor del coste total de reparación para cada componente en euros:

Se muestran a continuación los resultados ofrecidos por la herramienta:

```
In [10]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)
         datos_salida_modos
```

```
Out[10]:                coste_reparacion_total_maquinaria  \
          tr  ss   mf
          T_0 SS_0 MF_1                                0.0
                   MF_2                                0.0
                   MF_3                                0.0

                       coste_reparacion_total_materiales  \
          tr  ss   mf
          T_0 SS_0 MF_1                                  0
                   MF_2                                  0
                   MF_3                                  0

                       coste_reparacion_total_mano_obra  coste_reparacion_total
```

tr	ss	mf		
T_0	SS_0	MF_1	0.0	102150.0
		MF_2	0.0	0.0
		MF_3	0.0	0.0

co_costes_reparacion_03_pi

April 27, 2018

1 Ejemplo para el cálculo de costes de reparacion en el grado de desarrollo de proyecto de inversión: Parte 3 - Representación de resultados y cálculo de costes

Este ejemplo continuará a partir de los datos guardados tras la verificación de la estrategia de reparación al final del apartado Parte 2 - Verificación simultánea de todos los modos de fallo principales a lo largo de la vida útil.

1.1 Importación de paquetes de IPython

```
In [30]: # Python 2/3 setup
         from __future__ import (absolute_import, division, print_function, unicode_literals)
         # from builtins import *
```

```
In [31]: # Juypyter setup

         %matplotlib inline
         import os
         import sys
         import pickle
         import ast
         import pandas as pd
         env.pandas_setup()
         from IPython.display import HTML
```

```
In [32]: sim_path = os.path.join(env.modules_path ,'AP_PI_reparacion_est_conservadora', 'datos',
         'sim_01')

         if sim_path not in sys.path:
             sys.path.append(sim_path)
```

1.2 Carga de variables del apartado anterior

```
In [33]: # Getting back the objects:
         dir_data = os.path.join(env.data_path, 'reparacion', 'proyecto_inversion',
         'var_co_costes_reparacion_01_pi.pkl')

         with open(dir_data) as f:  # Python 3: open(..., 'rb')
             ruta_de, ruta_ds, alcance, estrategia, de_planta, de_esquema_division_dique,
         de_diagrama_modos, clima_tramos, cadencia, de_tipo_verificacion, de_verificacion_tramos,
         peralte, de_reparacion_necesarios, de_reparacion_disponibles, de_arbol_fallo =
         pickle.load(f)


         # Getting back the objects:
         dir_data = os.path.join(env.data_path, 'reparacion', 'proyecto_inversion',
```

```
        'var_co_costes_reparacion_02_pi.pkl')

        with open(dir_data) as f:  # Python 3: open(..., 'rb')
            datos_salida, averia_acum_estado, averia_estado, estado_modos_fallo,
        origen_maquinaria_reparacion_estado, origen_materiales_reparacion_estado,
        origen_mano_obra_reparacion_estado, datos_salida_prob_ini_averia = pickle.load(f)
```

1.3 Importación de paquetes para el ejemplo

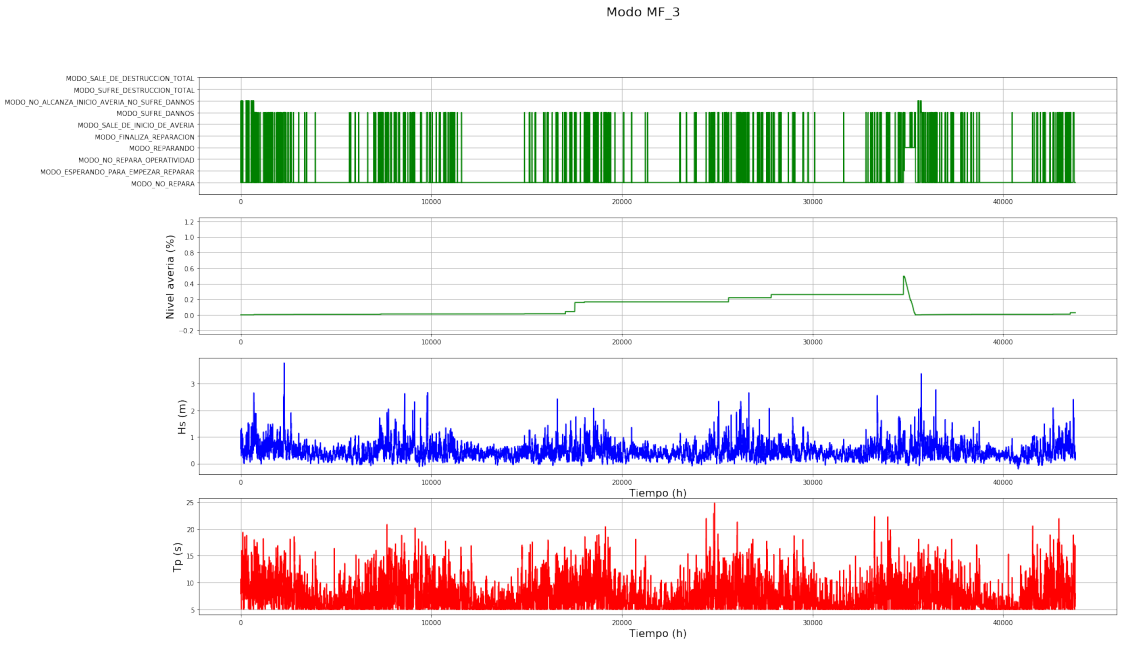
```
In [34]: import matplotlib.pyplot as plt
         from reparacion.calculos import calculo_final_simulacion
         from reparacion.calculos import extraccion_resultados
         from reparacion.calculos import calculo_costes
```

1.4 Representación de los resultados de la simulación numérica

```
In [35]: # Calculo de la hora final de simulacion
         h_fin_sim = calculo_final_simulacion(de_planta, clima_tramos, cadencia)

         # Representacion grafica
         col = ['r', 'g', 'k', 'y', 'b', 'c', 'm', 'r', 'g', 'k', 'y', 'b', 'c', 'm', 'r', 'g',
         'k', 'y', 'b', 'c']
         display = ['MF_0', 'MF_1', 'MF_2', 'MF_3', 'MF_4', 'MF_5', 'MF_6', 'MF_7', 'MF_8',
         'MF_9', 'MF_10', 'MF_11',
         'MF_12', 'MF_13', 'MF_14', 'MF_15', 'MF_16', 'MF_17', 'MF_18', 'MF_19', 'MF_20']

         cont = 1
         for mst in de_esquema_division_dique.iterrows():
             plt.figure(figsize=(25, 15))

             # Obtencion del tramo, subsistema y modo de fallo
             mst = mst[1]
             tr = mst['tramo']
             ss = mst['subsistema']
             mf = mst['modo_fallo']

             plt.suptitle('Modo ' + str(mf), fontsize=20)

             #ax1 = plt.subplot(de_esquema_division_dique.shape[0], 1, cont)
             ax1 = plt.subplot(4, 1, 1)
             x = range(0, cadencia * h_fin_sim, cadencia)
             ax1.plot(x, estado_modos_fallo.loc[(tr, ss, mf), :], color=col[cont],
         label=display[cont])
             plt.grid()
             plt.ylim(-1, 9)
             plt.yticks([0, 1, 2, 3, 4, 5, 6, 7, 8 ,9], [
                 'MODO_NO_REPARA', 'MODO_ESPERANDO_PARA_EMPEZAR_REPARAR',
         'MODO_NO_REPARA_OPERATIVIDAD', 'MODO_REPARANDO',
                 'MODO_FINALIZA_REPARACION', 'MODO_SALE_DE_INICIO_DE_AVERIA',
         'MODO_SUFRE_DANNOS',
                 'MODO_NO_ALCANZA_INICIO_AVERIA_NO_SUFRE_DANNOS', 'MODO_SUFRE_DESTRUCCION_TOTAL',
         'MODO_SALE_DE_DESTRUCCION_TOTAL'])

             ax2 = plt.subplot(4, 1, 2, sharex=ax1)
             x = range(0, cadencia * h_fin_sim, cadencia)
             y = averia_acum_estado.loc[(tr, ss, mf), :]
             ax2.plot(x, y, color=col[cont], label=display[cont])
             plt.ylim(-0.25, 1.25)
             plt.grid()
             plt.ylabel('Nivel averia (%)', fontsize=16)

             ax3 = plt.subplot(4, 1, 3, sharex=ax1)
             x = range(0, cadencia * h_fin_sim, cadencia)
             hs = clima_tramos[tr].ix[0: h_fin_sim - 1, 'hs']
             ax3.plot(x, hs, color='b', label='Hs (m)')
             plt.grid()
```

```
            cont += 1
            plt.xlabel('Tiempo (h)', fontsize=16)
            plt.ylabel('Hs (m)', fontsize=16)

            ax4 = plt.subplot(4, 1, 4, sharex=ax1)
            x = range(0, cadencia * h_fin_sim, cadencia)
            tp = clima_tramos[tr].ix[0: h_fin_sim - 1, 'tp']
            ax4.plot(x, tp, color='r', label='Tp (s)')
            plt.grid()
            plt.xlabel('Tiempo (h)', fontsize=16)
            plt.ylabel('Tp (s)', fontsize=16)

            # Guardado de resultados
            name_fichero = ('Modo_' + str(mf) + '_estados.pdf')
            direct = os.path.join(ruta_ds, name_fichero)

            plt.savefig(direct)
            plt.show()
```

```
C:\Users\GDFA-JUAN\Anaconda2\lib\site-packages\ipykernel_launcher.py:42:
DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```



En esta figura se muestra en la fila el estado por el que pasa el modo de fallo en cada estado a lo largo de los 5 años de simulación. La segunda fila muestra el grado de avería acumulado del fallo estado a estado y las filas tercera y cuarta muestran la alura de ola y periodo en cada estado.

Se muestran a continuación las salidas gráficas de otras simulaciones realizadas

Simulación 10

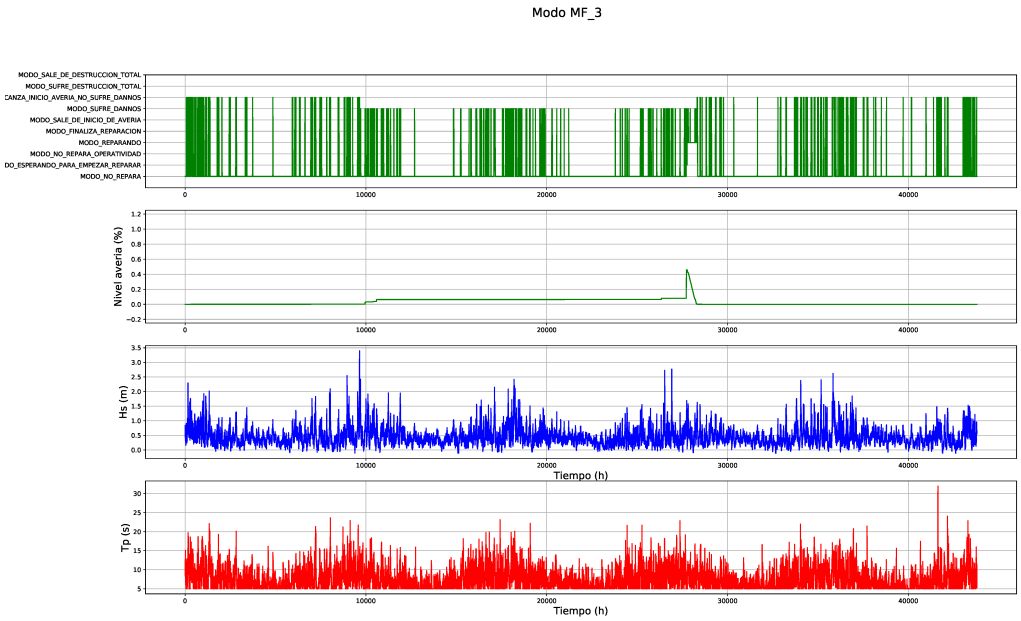
```
In [36]: from IPython.display import Image
         img_name = os.path.join(ruta_de, 'imagenes', 'Sim_10_Modo_MF_3_estados.png')
         Image(filename=img_name, width=800)
```

Out[36]:

Simulación 15

```
In [37]: from IPython.display import Image
         img_name = os.path.join(ruta_de, 'imagenes', 'Sim_15_Modo_MF_3_estados.png')
         Image(filename=img_name, width=800)
```

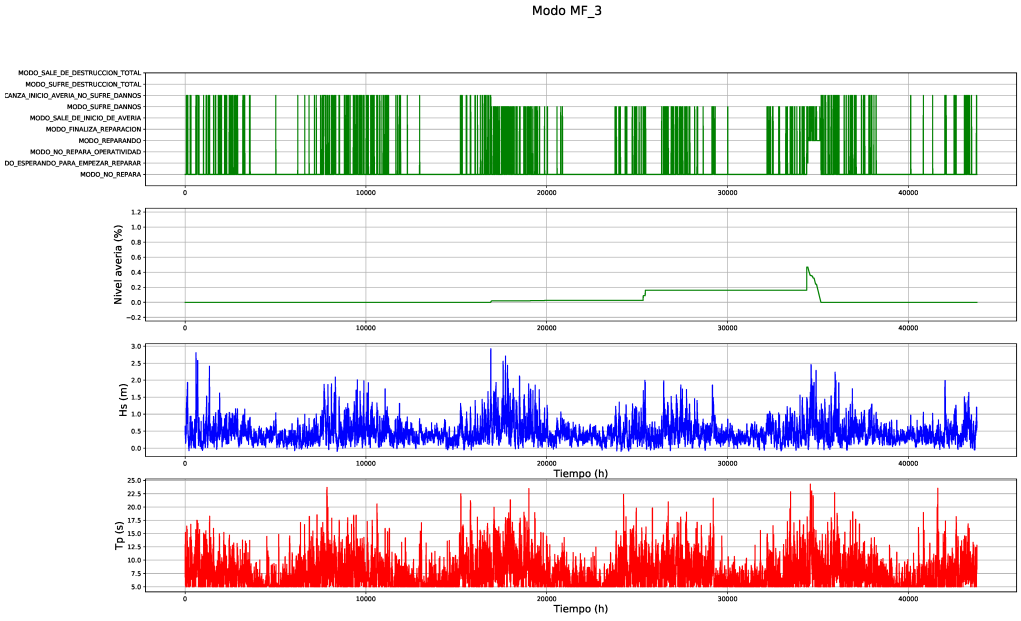
Out[37]:



Simulación 24

```
In [38]: from IPython.display import Image
         img_name = os.path.join(ruta_de, 'imagenes', 'Sim_24_Modo_MF_3_estados.png')
         Image(filename=img_name, width=800)
```

Out[38]:



1.5 Extracción de resultados

```
In [39]: (datos_salida) = extraccion_resultados(datos_salida, datos_salida_prob_ini_averia,
         de_planta,
                                                 estado_modos_fallo, h_fin_sim, ruta_ds, cadencia)
```

1.6 Cálculo de costes

```
In [40]: (datos_salida_modos, datos_salida_total, datos_salida_ea_sencillo) =
         calculo_costes(estado_modos_fallo, de_reparacion_necesarios, de_reparacion_disponibles,
             datos_salida, de_planta, ruta_ds, alcance, estrategia, cadencia)
```

1.7 Presentación de resultados

1.7.1 Resultados de la verificación

Tras la verificación simultánea de los modos de fallo, la herramienta ofrece los siguientes resultados para cada componente de la obra de abrigo: - Vida útil simulada - Número de veces que se produce inicio de avería - Probabilidad de inicio de avería - Número de estados en los que se produce daño en el componente - Número de veces que se inicia la reparación de la componente - Probabilidad de que la componente sufra daños - Duraciones mínima, media y máxima de los eventos de daño en la componente - Duraciones mínima, media y máxima de espera para iniciar los trabajos de reparación una vez dada la orden - Duraciones mínima, media y máxima de los trabajos de reparación - Duraciones mínima, media y máxima de las paradas operativas

Se muestran a continuación los resultados ofrecidos por la herramienta para el modo de fallo MF_3

```
In [41]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)
         datos_salida['datos_salida_modos']
```

Out[41]:

	vida_util	n_veces_ia	n_veces_dest_total	prob_ia	n_veces_danno	\
MF_3	5.0	2	0	0.000046	1423	

	n_veces_ini_rep	prob_sufrir_danno	dur_media_danno	dur_max_danno	\
MF_3	1	0.032489	9.614865	75.0	

	dur_min_danno	dur_media_espera_rep	dur_max_espera_rep	\
MF_3	3.0	12.0	15.0	

	dur_min_espera_rep	dur_media_rep	dur_max_rep	dur_min_rep	\
MF_3	9.0	35.571429	261.0	3.0	

	dur_media_par_ope	dur_max_par_ope	dur_min_par_ope
MF_3	0.0	0.0	0.0

1.7.2 Resultados de la adjudicación de costes de reparación

Tras la adjudicación de costes, en este alcance la herramienta ofrece el valor del coste total de reparación para cada componente en euros:

Se muestran a continuación los resultados ofrecidos por la herramienta:

```
In [42]: # Cambio el numero de filas y columnas que se muestran
         env.pandas_setup(50, 50)
         datos_salida_modos
```

```
Out[42]:                coste_reparacion_total_maquinaria  \
          tr  ss   mf
          T_0 SS_0 MF_3                          428355.0

                        coste_reparacion_total_materiales  \
          tr  ss   mf
          T_0 SS_0 MF_3                             3000

                        coste_reparacion_total_mano_obra  coste_reparacion_total
          tr  ss   mf
          T_0 SS_0 MF_3                         225450.0                656805.0
```

co_costes_reparacion_postproceso_simulaciones_01_pi

April 27, 2018

1 Ejemplo de postproceso, agrupación de resultados obtenidos en las simulaciones del cálculo de costes de reparacion en el grado de desarrollo de proyecto de inversión.

Este ejemplo continuará a partir de los datos obtenidos en cada una de las simulaciones numéricas de las fases de reparación a lo largo de la vida útil del dique.

1.1 Importación de paquetes de IPython

In [2]:
```
# Python 2/3 setup
from __future__ import (absolute_import, division, print_function, unicode_literals)
# from builtins import *
```

In [3]:
```
# Juypyter setup

%matplotlib inline
import os
import sys
import pickle
import ast
import pandas as pd
import matplotlib.pyplot as plt
env.pandas_setup()
from IPython.display import HTML
```

In [4]:
```
sim_path = os.path.join(env.modules_path ,'AP_PI_reparacion_est_conservadora')

if sim_path not in sys.path:
    sys.path.append(sim_path)
```

1.2 Importación de paquetes para el ejemplo

In [5]:
```
from posproceso import generar_tablas_costes_estado
from posproceso import coste_total_reparacion
from posproceso import figura_coste_total_reparacion
from posproceso import coste_anual_reparacion
from posproceso import figuras_coste_anual_reparacion
```

1.3 Datos de entrada

In [6]:
```
carpeta_resultados = os.path.join('resultados')
estrategias = ['estrategia_conservadora']
n_sim = 25
simulaciones = range(n_sim)
```

1.4 Generación de tablas de costes de reparación estado a estado

Se genera la tabla con los costes de reparación para cada modo de fallo estado a estado para cada simulación

```
In [7]: datos = generar_tablas_costes_estado(estrategias, simulaciones, carpeta_resultados,
        ruta_de=sim_path)

        # Cambio el numero de filas y columnas que se muestran
        env.pandas_setup(10, 10)
        datos
```

Out[7]:

	tr	ss	mf	0	1	...	14595	14596	14597	14598	14599
0	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
1	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
4	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
..	...	...	...	...	...	...	...	...	...	...	...
20	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
21	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
22	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
23	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
24	T_0	SS_0	MF_3	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

[25 rows x 14603 columns]

1.5 Generación de tablas de costes de reparación total

Se genera la tabla con los costes totales de reparación para cada modo de fallo para cada simulación a lo largo de la vida útil

```
In [8]: datos = coste_total_reparacion(estrategias, carpeta_resultados, ruta_de=sim_path)

        # Cambio el numero de filas y columnas que se muestran
        env.pandas_setup(24, 10)
        datos
```

Out[8]:

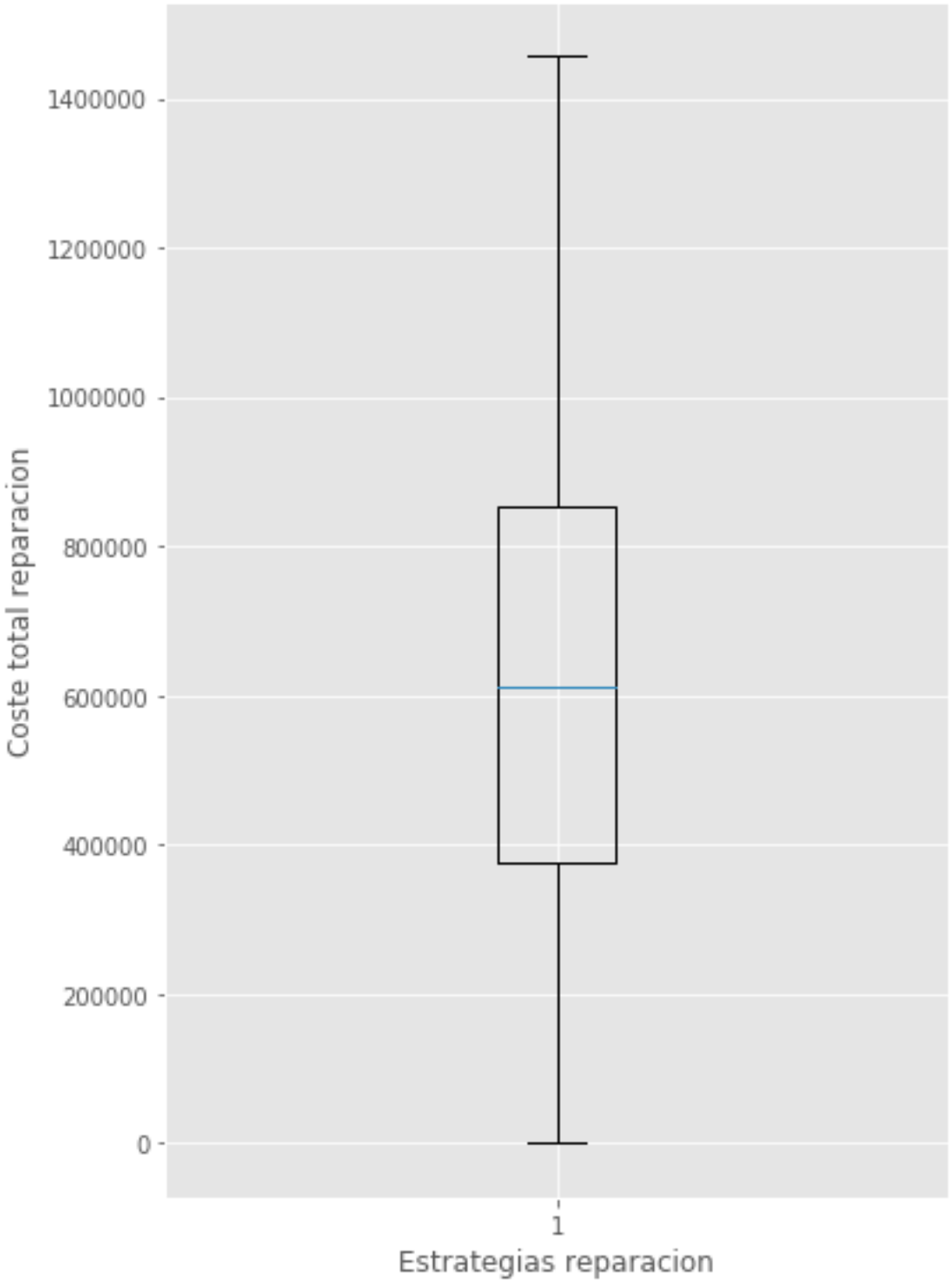
tr	ss	mf	
T_0	SS_0	MF_3	375840.0
		MF_3	207495.0
		MF_3	0.0
		MF_3	598995.0
		MF_3	0.0
		MF_3	853470.0
		MF_3	1331100.0
		MF_3	1017900.0
		MF_3	728190.0
		MF_3	728190.0
		MF_3	751680.0
		MF_3	598995.0
			...

```
MF_3          0.0
MF_3     481545.0
MF_3     602910.0
MF_3    1456380.0
MF_3     732105.0
MF_3    1381995.0
MF_3    1225395.0
MF_3     732105.0
MF_3     321030.0
MF_3     610740.0
MF_3     595080.0
MF_3     211410.0
Length: 25, dtype: float64
```

1.6 Boxplot coste total de reparación del modo de fallo en euros a lo largo de la vida útil

Boxplot con la variabilidad del coste total de reparación del modo de fallo en euros a lo largo de la vida útil. La línea azul indica el coste medio en las N simulaciones.

```
In [9]: plt.figure(figsize=(6, 8))
        figura_coste_total_reparacion(estrategias, carpeta_resultados, ruta_de=sim_path)
```


1400000
1200000
1000000
800000
600000
400000
200000
0
Coste total reparacion
1
Estrategias reparacion

1.7 Generación de tablas de costes anuales de reparación

Se genera la tabla con los costes anuales de reparación para cada modo de fallo para cada simulación a lo largo de la vida útil

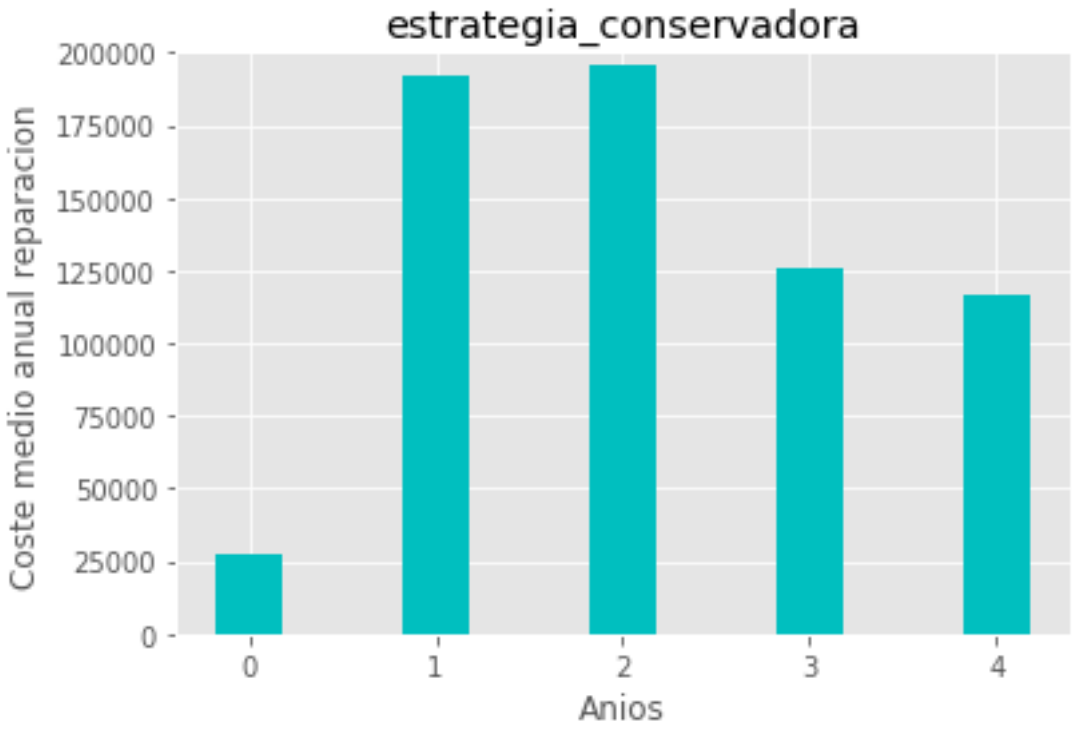
```
In [10]: muestra_coste_anual = coste_anual_reparacion(estrategias, carpeta_resultados,
         ruta_de=sim_path)
```

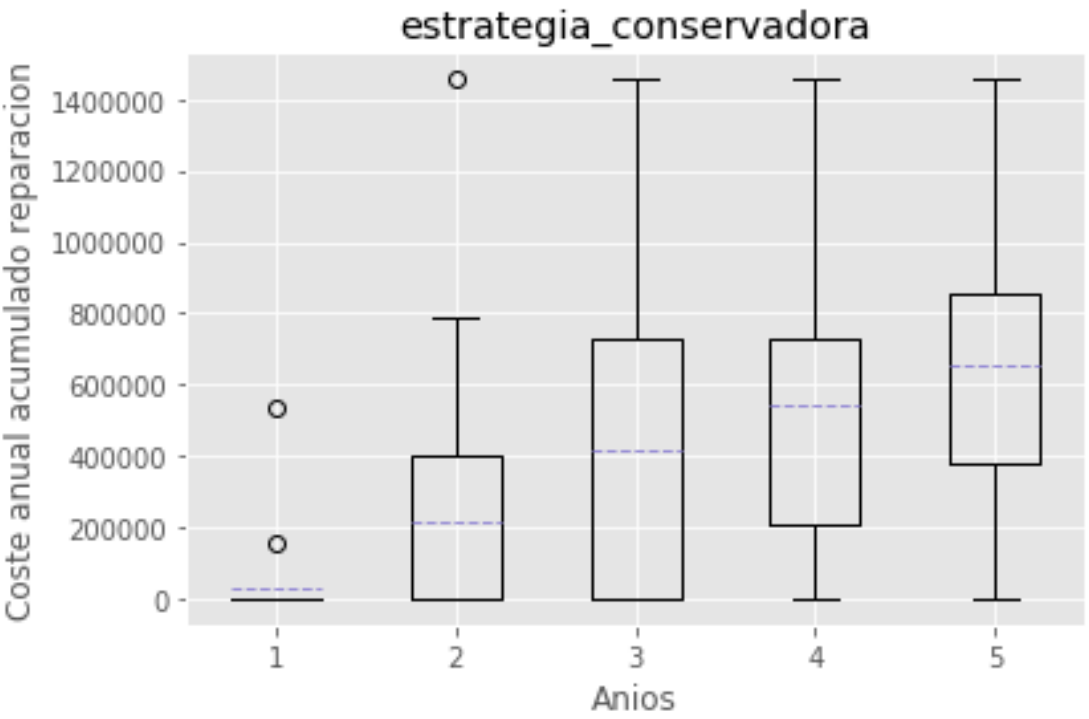
1.8 Generación de figuras de costes medio anual de reparación y coste acumulado año a año de reparación

En las figuras siguientes se muestra el histograma con los costes medios año a año de reparación del modo de fallo y un boxplot con los costes acumulados de simulación a lo largo de la vida útil junto con la variabilidad recogida en las N simulaciones

```
In [11]: plt.figure(figsize=(10, 15))
         figuras_coste_anual_reparacion(estrategias, carpeta_resultados, ruta_de=sim_path)
```

```
<matplotlib.figure.Figure at 0xd848c88>
```




estrategia_conservadora
Coste anual acumulado reparacion
1400000
1200000
1000000
800000
600000
400000
200000
0
1
2
3
4
5
Anios