A continuación, se explicarán los scripts creados que hacen uso de ArrayCalc.... para poder llevar a cabo todas las simulaciones que se llevan a cabo en este proyecto.  
  
lanzador.m

Este es el script principal. Es una función a la que se le pueden pasar los siguientes parámetros: ejecucion, caso y clusters\_fijados. La variable ejecucion equivale a la realización del caso y su único fin es el de poder identificar dicha realización cuando se guarden las variables en un fichero, pues el salvado de estas variables se lleva a cabo en cada realización. clusters\_fijados servirá para establecer la cantidad de clusters que deseamos establecer en el escenario. En todo momento, la máxima cantidad de clusters es igual a la cantidad de sensores en el escenario; sin embargo, clusters\_fijados nos permitirá elegir cuántos clusters queremos disponer. Si clusters\_fijados == 0, le estaremos indicando a la simulación que optimice la cantidad de clusters, en lugar de que venga fijada. Dado el caso introducido, el script contiene gran cantidad de ifelse para poder identificar cada uno de estos casos. Las diferencias entre ellos serán las distintas características que diferencian los escenarios que se desean simular: máxima carga de batería para todos los sensores o carga aleatoria para todos ellos, sensores fijos a lo largo del tiempo o posiciones aleatorias, la dirección en la que se desea radiar (phi y theta), los pasos de integración tanto para poder calcular el campo del escenario actual como para poder hacer la comparación y poder calcular la eficiencia, ..... y el tiempo de simulación. El tiempo de simulación es un entero que representa la cantidad de tiempo durante el cual se simulará el escenario en caso de que los nodos sean móviles; en cambio, si los nodos adoptan posiciones fijas a lo largo del tiempo, es de esperar que la optimización siempre sea la misma a lo largo del tiempo (aunque no siempre, dado que los algoritmos genéticos son algoritmos pseudo-aleatorios). Por ello, en dicho caso, el tiempo de simulación representa la cantidad de realizaciones del caso en cuestión que se desea llevar a cabo.

Este script se compilará mediante la herramienta DEPLOYTOOL de Matlab, que lo convertirá en un archivo ejecutable, con  extensión .exe. Gracias a tener todo el código compilado de esta forma, conseguiremos que, mediante la introducción de parámetros, podamos ejecutar cientos o incluso miles de simulaciones en paralelo en múltiples instancias en distintos ordenadores. El principal motivo para llevar a cabo las simulaciones de esta manera es conseguir gran cantidad de realizaciones de un mismo experimento, en el mismo tiempo que tardaría en ejecutarse una sola realización.

Una vez identificado mediante el script el caso que se desea simular, se le pasarán las variables específicas de ese caso al script casos.

casos

Este script se encarga de "traducir" algunas de las variables del caso en variables que puedan utilizar las siguientes funciones. Por ejemplo,  si la dimensión del caso a tratar es 2, casos establecerá el tamaño del escenario, dando los valores máximos permitidos en el eje X e Y, estableciendo el valor máximo del eje Z a cero, pues se requiere que los nodos del escenario se sitúen en un plano (2D).

Tras esto, casos llamará al script TODOXXX, que se encargará del resto. Cuando TODOXXX termine, casos guardará las variables como caso\_X\_ejecucion\_Y\_clusters\_fijados\_[Z.mat](http://Z.mat).

TODOXXX

Este script será el encargado de disponer el escenario, llamar al algoritmo de optimización, calcular el tiempo de vida de los sensores del escenario y el tiempo de vida de los sensores que no usen beamforming. Con estos dos últimos datos, calculará la eficiencia dividiéndolos.

En primer lugar, se definen algunas variables necesarias para la correcta ejecución, como range\_config, la cual es una variable de ArrayCalc útil para indicar que se desea calcular el campo lejano.

XXX

En segundo lugar, se asignan las posiciones que tomarán los sensores en el escenario, dependiendo del tamaño del mismo, que vendrá dado por la dimensión del caso a simular y de los parámetros establecidos en el script casos.m. De este modo, para calcular la posición de los sensores, se tendrá en cuenta la dimensión del escenario, pero también se tendrá en cuenta si los sensores adquirirán posiciones fijas a lo largo del tiempo de simulación o posiciones móviles. En este segundo caso, se llamará al script movilidad.m, que devolverá las posiciones de unos nodos en movimiento en función del tiempo. En dicho script, se puede editar la velocidad de los sensores, la probabilidad de que estén pausados, etc.

En tercer lugar, se asignará el porcentaje de batería a cada uno de los sensores. A todos se les asignará el 100% de la batería en caso de que la variable potencia\_fija tenga asignado el valor 1 en el script casos; en cambio, la potencia de los sensores será aleatoria en caso de que esa variable tenga asignado el valor 0.

Tras estas asignaciones, comenzará el bucle principal de la simulación. Este bucle principal consiste en tres bucles for distintos: el primero, recorre la componente theta de las direcciones deseadas; el segundo, la componente phi de dichas direcciones; el tercero recorrerá el tiempo de simulación, en caso de que sea superior a 1.

XXXIntroducir dependencia de generico y estaciones

En el bucle principal, se inicializa la variable array\_config con las posiciones de los sensores en el escenario y su orientación. Tras esto, se le pasarán distintos parámetros a la función optimiza.m, devolviendo la variable array\_config rellena con los valores de amplitud y fase para cada uno de los nodos. En esa función optimiza.m, se llama a los algoritmos genéticos para poder aumentar la directividad en la dirección deseada, es decir, realiza la optimización que se pretende conseguir en este proyecto. De este modo, una vez devuelta la variable array\_config con los valores optimizados de fase y amplitud, se calculará el tiempo de vida de los sensores, calculando en primer lugar la potencia radiada y la potencia que se recibiría en la dirección deseada. Además de esos valores, la función optimiza devolverá la cantidad de clusters resultado de la optimización, qué sensores pertenecen a cada cluster y el centro del cluster de cada uno. De este modo, el bucle principal hará los cálculos de los tiempos de vida se casa sensor dependiendo de si dependen al cluster en cuestión.

Tras calcular el tiempo de vida de cada uno de los sensores, solo se tendrá en cuenta el mínimo para poder obtener el valor de la eficiencia al usar beamforming. Tras estos cálculos, se calculará el tiempo de vida de los sensores que no usan beamforming para poder realizar la comparación. Estos cálculos se llevarán a cabo sensor por sensor, y no en conjunto, como los cálculos del beamforming. Finalmente, se tomará para hacer la comparación el menor tiempo de vida.

Tras esto, se guardarán las variables. Ha de tenerse en cuenta que en caso de guardarse todas las variables (incluidos el campo E y P de cada cluster y para cada estación, además del campo de cada sensor de la comparación), el peso de cada archivo .mat puede llegar a pesar cientos de MB, dependiendo de la cantidad de sensores. Debido a que las variables del campo nos resultarán menos importancia a la hora de analizar los resultados, podemos prescindir de ellas y no guardarlas, pues son las que hacen que el peso de los archivos se eleve tanto. Con este borrado, las variables ocuparán menos de 50 KB. En caso de necesitar estas variables para poder dibujar el campo en polares, por ejemplo, pueden obtenerse fácilmente a partir de la variable array\_config guardada.