Instalación de Pyomo, solvers y módulo de (as)NMPC.

*Federico Lozano Santamaría.* ([f.lozano396@uniandes.edu.co](mailto:f.lozano396@uniandes.edu.co))

Grupo de Diseño de Productos y Procesos, Departamento de Ingeniería Química, Universidad de los Andes.

Carrera 1 No. 18A – 10, Bogotá, Colombia

# Resumen

En este documento se presentan los pasos y requerimientos necesarios para instalar Pyomo, un grupo de *solvers* para Pyomo y el módulo desarrollado para diseñar y evaluar controladores (as)NMPC. Adicionalmente, se encuentran los enlaces a las páginas correspondientes para la instalación de cada uno de los componentes y en las carpetas que se distribuyen con este documento se pueden encontrar programas e información adicional. Cabe aclarar que este no es un manual sobre como usar Pyomo o el módulo de (as)NMPC, solo presenta los requerimientos y la instalación de estos programas.

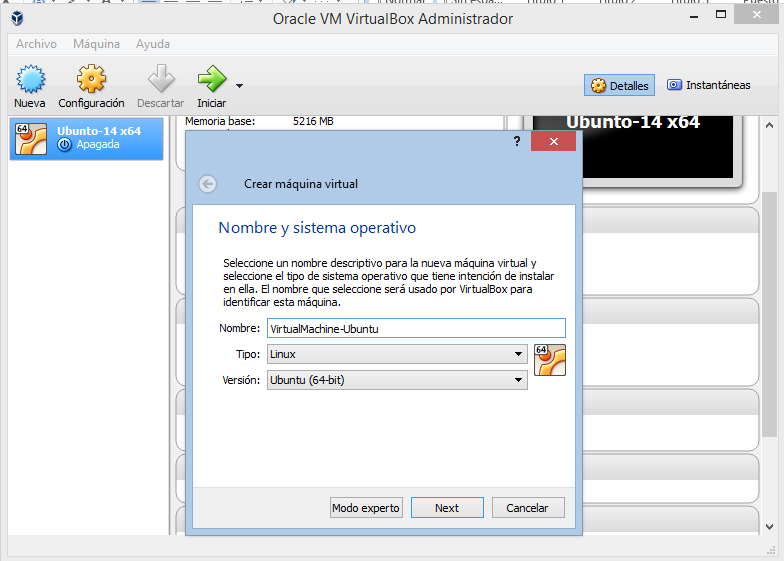
# 1. Máquina virtual

Pyomo es un programa de modelamiento que está desarrollado para Python, por lo tanto es necesario haberlo instalado previamente y disponer de una plataforma que lo ejecute. Aunque es posible trabajar todo desde Windows haciendo las instalaciones correspondientes aquí se presenta como trabajar todo desde Linux.

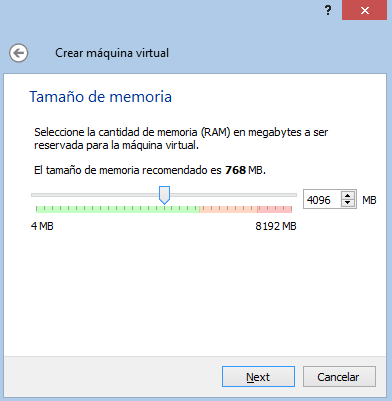
Si se trabaja en un equipo con Windows lo primero es instalar una máquina virtual o hacer una partición del disco para tener un sistema operativo con Linux. Se puede usar el programa Virtual Box (<https://www.virtualbox.org/>) que permite crear ambientes virtuales de sistemas operativos. En la página de este se puede encontrar fácilmente el enlace para descargarlo y la instalación del mismo es sencilla. Una vez instalado Virtual Box es necesario tener un instalador del sistema operativo de Linux, en este caso Ubuntu (<http://www.ubuntu.com/>).

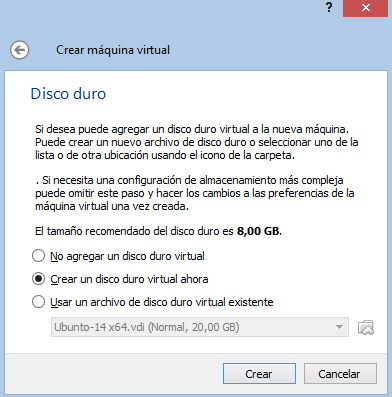
El instalador del sistema operativo de Ubuntu lo puede descargar desde la página de este (<http://www.ubuntu.com/download/desktop>). En la sección de descargas encuentra las versiones disponibles. Descargue la versión 14 - LST, aunque existen versiones más recientes disponibles solo se ha probado Pyomo y el módulo (as)NMPC en esta versión así que es posible que se presenten problemas de compatibilidad para nuevas versiones.

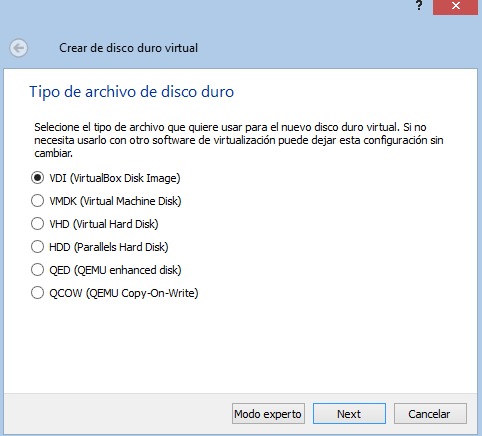
Para crear la máquina virtual abra el programa Virtual Box y seleccione *Nueva*, deberá abrirse una nueva ventana donde se asigna un nombre a la máquina virtual y se selecciona el sistema operativo y la versión.

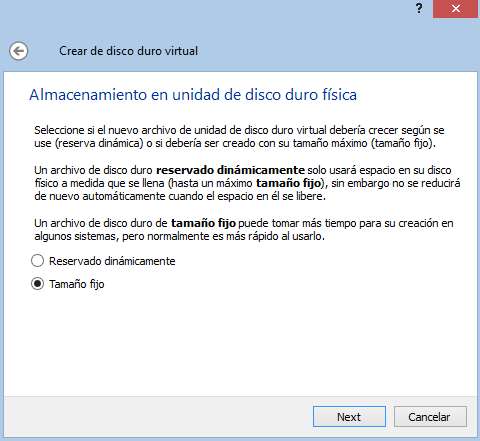


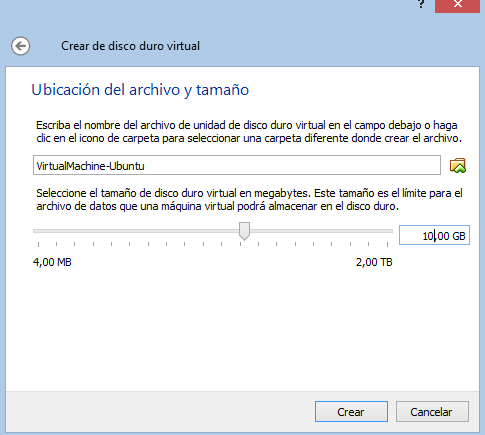
Ahora, oprima siguiente y siga las instrucciones en las ventanas que aparecen como asignar el tamaño de memoria RAM que utilizara la máquina virtual y crear un disco duro virtual (VDI) o usar uno ya existente. Si decide crear un disco duro virtual este puede ser de tamaño fijo o de tamaño variable, siendo el primero el más recomendable. Después de asignar el tamaño deseado al disco duro oprima crear, así podrá visualizar en Virtual box la nueva máquina virtual creada. Se recomienda que el tamaño mínimo asignado al disco duro virtual sea de 25 GB.

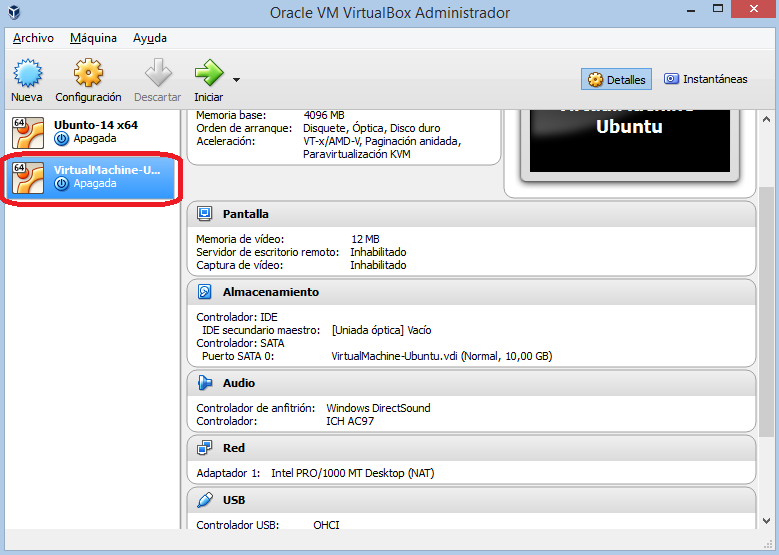




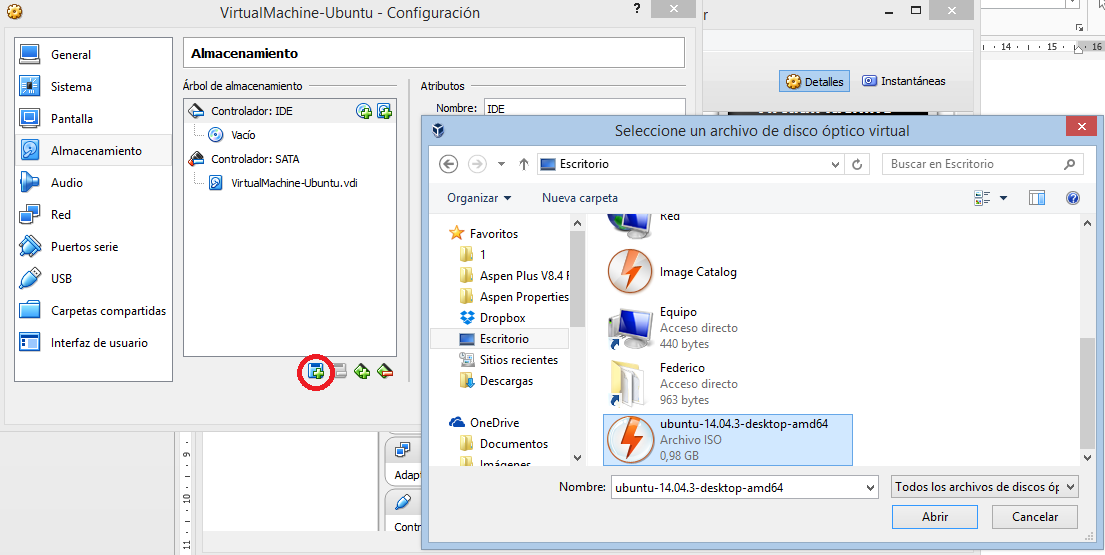




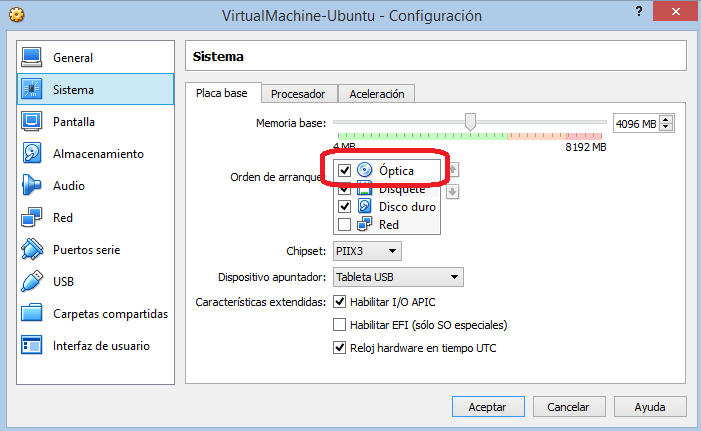




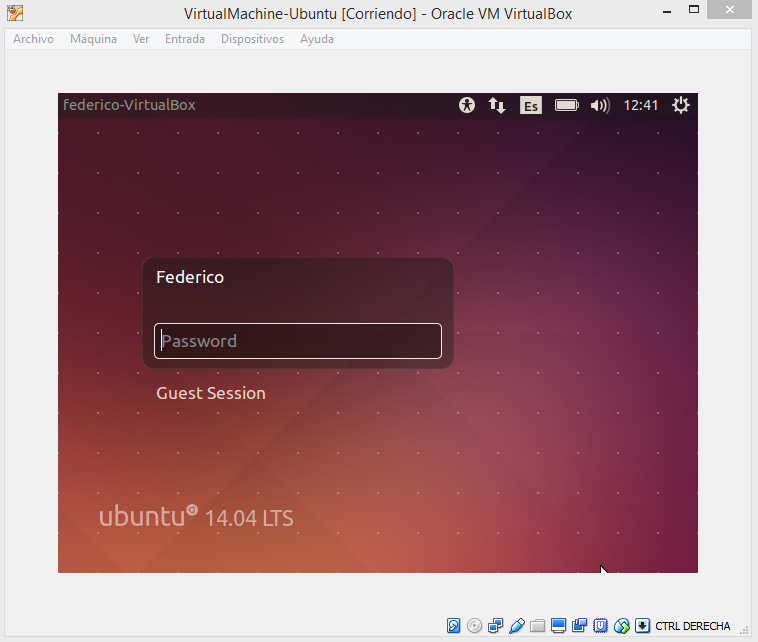
Con la máquina virtual creada es necesario configurarla para que inicie el instalador de Ubuntu. Selecciona la máquina virtual que creo y luego vaya a *Configuración*, *Almacenamiento,* *Controlador: IDE* y agregar unidad óptica. Aquí seleccione el archivo de instalación de Ubuntu que descargo previamente de tal forma que la máquina virtual puede cargar el CD de instalación.



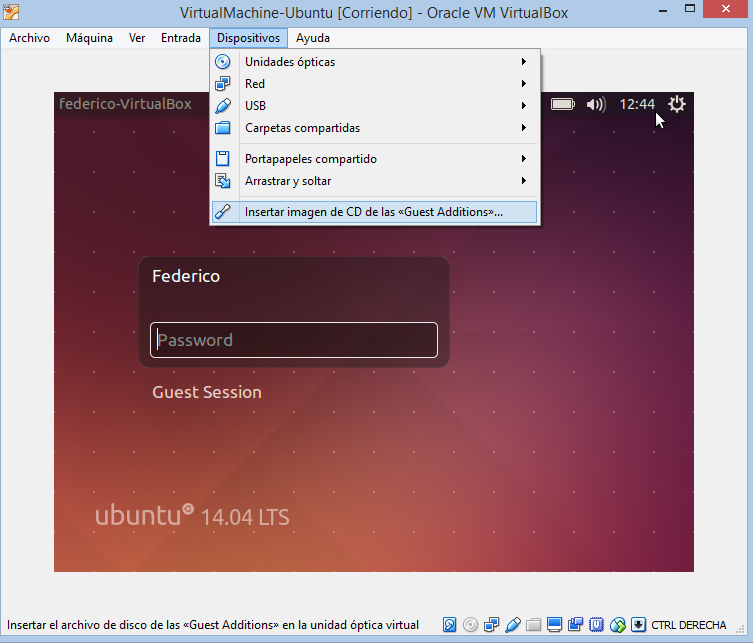
Luego, diríjase a *Sistema* y seleccione la unidad de CD/DVD como la primera unidad desde la que se inicia la máquina virtual.



Inicie la máquina virtual y debe observar que inicia el programa de instalación de Ubuntu. Siga los pasos de instalación, asegurándose de seleccionar la opción de descargar actualizaciones. Al final de la instalación se le pedirá que reinicie el sistema operativo, después de esto podrá acceder a Ubuntu.



Por último se debe realizar la instalación de aplicaciones huésped. Inicie Ubuntu y luego, en la barra de herramientas de Virtual Box, vaya a *Dispositivos*, *Insertar imágenes de CD…* Siga los pasos de instalación, la mayoría son automáticos, y al final reinicie la máquina virtual. Con este debe quedar Ubuntu funcionando completamente.

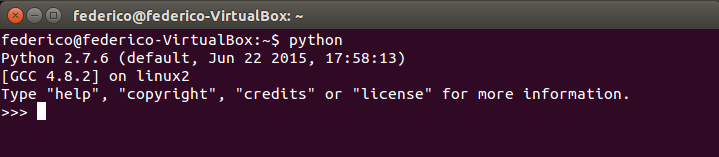


# 2. Pyomo

Antes de instalar Pyomo debe asegurarse que Python está instalado en el sistema. En el caso de Ubuntu este programa está por defecto y para verificarlo puede abrir la *Terminal* de Linux y ejecutar el comando “python” de tal forma que el *Command Line* de Python queda activo.

**$** python

Deberá poder ver algo como lo que se muestra en la siguiente imagen donde la segunda línea indica la versión de Python que se encuentra instalada.



Antes de proceder a instalar Pyomo es necesario instalar *Pip* que es un paquete alternativo que facilita la instalación de componentes y extensiones de Python. Este se puede instalar ejecutando la siguiente línea desde la *Terminal* de Linux.

**$** sudo apt-get install python-pip python-dev build-essential

La instalación de Pyomo también se puede realizar desde la *Terminal* ejecutando el comando:

**$** sudo pip install pyomo

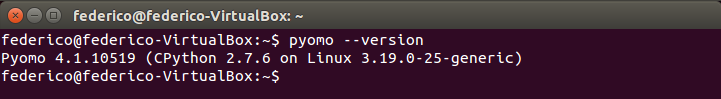
Sin embargo, esto instalará la última versión estable de Pyomo y el módulo (as)NMPC se desarrolló para la versión 4.1.10519 por lo que no funciona correctamente en versiones superiores. Específicamente, en versiones superiores se eliminó la función de la clase *Expression* que actualiza los índices de las expresiones después de discretizar modelos DAE y que se usa dentro de los módulos NMPC. Cabe aclarar que los módulos NMPC funcionan correctamente en versiones superiores siempre y cuando los modelos dinámicos no utilicen expresiones (*Expression*). Por lo tanto, para instalar la versión de Pyomo en la que los módulos de NMPC funcionan correctamente se debe ejecutar la siguiente línea de comando desde la *Terminal*:

**$** sudo pip install pyomo==4.1.10519

Para verificar que Pyomo se instaló de forma correcta ejecute el siguiente comando:

**$** pyomo --version

Deberá mostrarse la versión actual de Pyomo y la versión de Python con la que está corriendo.



Pyomo tiene una dependencia condicional de programas de terceros que no se distribuyen ni instalan junto a Pyomo. Muchos de ellos pueden ser instalados descargando y ejecutando el archivo *get\_pyomo\_extras.py* (<http://www.pyomo.org/installation/>). Una vez descargado el archivo diríjase a la carpeta donde se encuentra, usando el comando *cd* desde la terminal para cambiar de directorio, y ejecute el comando:

**$** sudo python get\_pyomo\_extras.py

Esto deberá instalar todos los extras de Pyomo. Nótese que Pyomo es solo el lenguaje usado para definir modelos de optimización en Python y no dispone de *solvers* para solucionar estos modelos, estos *solvers* deben ser instalados por separado. Básicamente cualquier *solver* que sea compatible con AMPL mediante la interfaz ASL es compatible con Pyomo.

Como editor de texto para realizar los *scripts* con extensión de Python (.py) que son los ejecutables de Pyomo se recomiendo Jedit. Para instalar este editor de texto ejecute la siguiente línea de comando desde la *Terminal*:

**$** sudo apt-get install jedit

Para mayor información acerca de Pyomo puede visitar la página (<http://www.pyomo.org/>), el foro de usuarios (<https://groups.google.com/forum/#!forum/pyomo-forum>) y revisar la bibliografía del programa (<http://www.pyomo.org/documentation>). Entre la bibliografía se encuentra la guía de instalación, un manual de primeros pasos y un gran número de ejemplos. Como material introductorio para aprender a usar Pyomo se recomienda revisar en detalle el documento *Pyomo Online Documentation 4.1.*

# 3. Solvers

Pyomo provee interfaz a una gran variedad de *solvers*. Estos pueden ser usados de dos formas, mediante una transcripción directa del modelo al *sovler* por medio de clases dependientes (tightly coupled) o mediante una interfaz de archivos y comandos del sistema (loosly coupled). En esta sección se presenta la instalación de tres *solvers* y un *pseudo-solver*, todos de acceso libre, que pueden ser utilizados en Pyomo. Tenga en cuenta que Pyomo no está restringido a estos *solvers* y puede trabajar como muchos más si se dispone del ejecutable y de la licencia correspondiente si es el caso.

## 3.1. GLPK

El paquete GLPK (GNU Linear Programming Kit) permite resolver problemas lineales (LP) y problemas lineales con variables enteras (MIP). En la página del paquete se puede encontrar mucha más información acerca de este así como los programas e instrucciones para su instalación (<http://www.gnu.org/software/glpk/>). Diríjase a la sección de descargas y descargue la última versión de GLPK.

Descomprima el archivo que descargó y mueva esta carpeta al directorio en el que quiere instalar el programa. Ahora, desde la *Terminal* use *cd* para cambiar al directorio de esta carpeta y ejecute los siguientes comandos para realizar la instalación del *solver*:

**$** sudo ./configure

**$** sudo make

**$** sudo make install

**$** sudo make check

Si todo está bien la última línea debe ejecutar un ejemplo y en la *Terminal* debe leerse “OPTIMAL LP SOLUTION FOUND”. Si no observa este mensaje sino un error relacionado con una librería que no se encuentra debe adicionar las librerías de glpk a las variables de entorno. Para hacer esto abra el archivo *~/.bashrc* como se describe en los siguientes párrafos y agregue al final de este la siguiente línea.

export LD\_LIBRARY\_PATH=”usr/local/lib:$LD\_LIBRARY\_PATH”

Para poder utilizar este *solver* desde Pyomo el directorio donde se encuentra el ejecutable *glpsol* debe estar incluido en el PATH. Para hacer esto se debe agregar al archivo *~/.bashrc* el acceso al directorio en que se encuentra el ejecutable. Para abrir este archivo ejecute el siguiente comando desde la *Terminal*:

**$** sudo gedit ~/.bashrc

Una vez con el archivo abierto agregue la siguiente línea al final de este.

export PATH=”usr/local/bin:$PATH”

Por último, para probar que se puede ejecutar el *solver* con Pyomo use *cd* para cambiar el directorio de trabajo por el directorio de uno de los ejemplos de Pyomo que se llama *diet*. Desde este directorio ejecute la siguiente línea de comando que carga el modelo *diet1.py* y lo resuelve con el este *solver.* Si todo es correcto deberá ver un mensaje indicando que se encontró solución óptima y que el valor de la función objetivo es 2.81.

**$** pyomo solve ––solver=glpk diet1.py diet.dat

## 3.2. IPOPT

IPOPT (Interior point optimizer) es un *solver* o paquete usado para resolver problemas no lineales de optimización encontrando mínimos locales. Este paquete es de licencia libre y es mantenido por COIN-OR (<https://projects.coin-or.org/Ipopt>). Adicionalmente, en la página del programa se pueden encontrar las instrucciones de instalación y los requerimientos de este.

IPOPT depende de una gran cantidad de programas de terceros para su funcionamiento que no se instalan automáticamente con el *solver.* En esta página (<http://www.coin-or.org/Ipopt/documentation/node10.html>) se presentan cada uno de los requerimientos y los pasos de instalación de IPOPT en Linux. Primero que todo es necesario instalar los compiladores necesarios ejecutando el siguiente comando desde la *Terminal*:

**$** sudo apt-get install gcc g++ gfortran subversion patch wget

Proceda a descargar la última versión estable de IPOPT en <http://www.coin-or.org/download/source/Ipopt>. El archivo que descarga debe tener una extensión .tgz, un tipo de archivo comprimido, que debe descomprimir. Luego, cambie el nombre de la carpeta que extrajo por “CoinIpopt” y muévala al directorio donde quiere instalar el programa. Desde esta carpeta se puede hacer la instalación del *solver* pero antes es necesario descargar e instalar paquetes adicionales como la librería ASL, las subrutinas HSL y los *sovlers* lineales.

Todo lo que se muestra de aquí en adelante se debe ejecutar desde el directorio CoinIpopt. Descargar las librerías BLAS, LAPACK y ASL:

**$** cd /ThirdParty/Blas

**$** ./get.Blas

**$** cd ../Lapack

**$**.get.Lapack

**$** cd ../ASL

**$**./get.ASL

Para descargar las subrutinas HSL debe dirigirse a la página <http://www.hsl.rl.ac.uk/ipopt/> y en la parte derecha seleccione la versión de Linux de Coin-HSL Full (Stable), complete los datos y en aproximadamente un día hábil debe recibir un correo con el enlace de descarga de las subrutinas. Sin embargo, en los archivos que se distribuyen con este documento están las subrutinas. Dentro de estas subrutinas se incluyen algoritmos de solución de problemas lineales que explotan la dispersión y estructura de las matrices como los que se muestran a continuación:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Solver* | Libre para todos | Libre para academia | Tamaño del problema | Paralelo | Respuestas repetibles | Notas |
| MA27 | Si | Si | Pequeño | No | Si | Desactualizado, relativamente lento |
| MA57 |  | Si | Pequeño/ Mediano | Con BLAS | Si |  |
| HSL\_MA77 |  | Si | Gigante | Limitado | Si | Out-of-core |
| HSL\_MA86 |  | Si | Grande | Altamente | No | Diseñado para múltiples procesadores |
| HSL\_MA97 |  | Si | Pequeño/ Mediano/ Grande | Si | Si | Más lento que HSL\_MA86 para problemas grandes |

Para que las subrutinas HSL se compilen con IPOPT debe extraer el archivo que descargo (o copiarlo de la carpeta que se provee con el manual) y cambiarle el nombre por “coinhsl”, después debe copiar esta carpeta en /CoinIpopt/ThirdParty/HSL.

Para obtener el *solver* linear MUMPS, uno de los más comunes y de dominio público, debe ir al directorio CoinIpopt/ThirdParty/Mumps y ejecutar el siguiente *script*:

**$** ./get.Mumps

Además de MUMPS puede instalar otros *sovlers* lineales que se presentan en la página de instalación (<http://www.coin-or.org/Ipopt/documentation/node13.html>) como Pardiso y WSMP.

Por último, es necesario instalar la librería METIS que proporciona un algoritmo de ordenamiento de matrices para hacer más eficientes los *solvers* lineales MUMPS; MA57, HSL\_MA77, HSL\_MA86 y HSL\_MA97. Para obtener esta librería diríjase al directorio CoinIpopt/ThirdParty/Metis y ejecute el siguiente *script*:

**$** ./get.Metis

Ahora bien, con esto se completa la descarga de todos los paquetes adicionales que requiere IPOPT y se puede proceder con la compilación e instalación del programa. Primero debe crear una carpeta llamada “build” en el directorio de CoinIpopt. Luego, ejecute los siguientes comandos desde la *Terminal* estando en el directorio de CoinIpopt:

**$** ./configure

**$** sudo make

**$** sudo make test

**$** sudo make install

Si todas las librerías quedaron bien instaladas después de ejecutar los *test* debe leerse que se han pasado todos y en la siguiente línea se instalará IPOPT completamente. Sin embargo, para poder usar este *solver* desde Pyomo la carpeta donde está el ejecutable (“ipopt”) debe incluirse en el PATH. Para hacer esto se debe agregar al archivo *~/.bashrc* el acceso al directorio en que se encuentra el ejecutable. Para abrir este archivo ejecute el siguiente comando desde la *Terminal*:

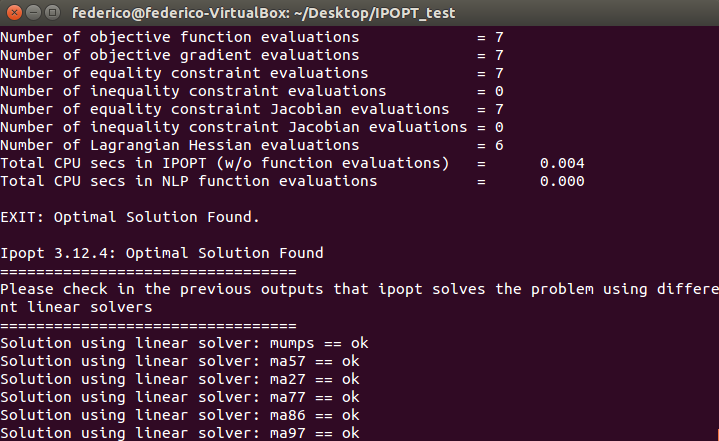
**$** sudo gedit ~/.bashrc

Una vez con el archivo abierto agregue la siguiente línea al final de este. Note que en esta línea puede cambiar el directorio dependiendo de en donde haya guardado la carpeta CoinIpopt.

export PATH=”home/user/Desktp/CoinIpopt/bin:$PATH”

Finalmente, puede probar que todo está funcionando correctamente corriendo el *script* *test.py* que se encuentra en la carpeta IPOPT\_test. En este *script* se resuelve un pequeño problema de optimización no lineal con IPOPT y todos los *solvers* lineales que se instalaron. Si todo funciona bien no debe observar ningún mensaje de error.

**$** python test.py



## 3.3. sIPOPT

sIPOPT (optimal sensitivity based on IPOPT) es una herramienta que usa la teoría de sensibilidad NLP para obtener soluciones rápidas y aproximadas a problemas de optimización cuando los parámetros cambian (<https://projects.coin-or.org/Ipopt/wiki/sIpopt>). En la página se puede encontrar más información y para revisar como funciona y la teoría que hay detrás puede revisar el manual.

Para compilar e instalar sIPOPT se parte de que ya tiene instalada una versión del *solver* IPOPT. Si es así diríjase a la carpeta de este *solver* (CoinIpopt) y ejecute las siguientes acciones desde la *Terminal*:

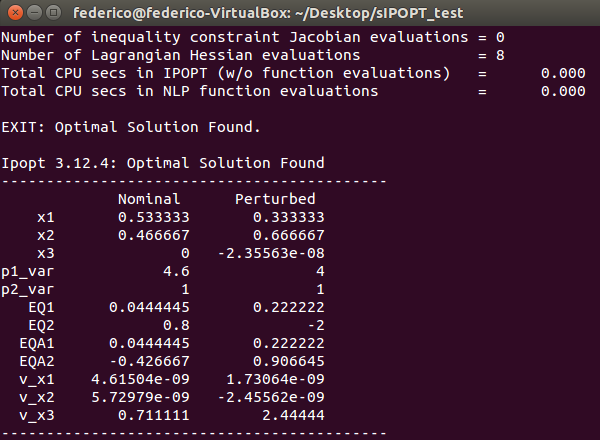
**$** cd CoinIpopt/Ipopt/contrib/sIPOPT

**$** sudo make

**$** sudo make install

Por último, para probar que la instalación fue correcta ejecute el *script* *test.py* que se encuentra en la carpeta *sIPOPT\_test*. Debe observar que en la consola se muestra que el problema llego a solución óptima y dos columnas con las variables del problema, donde la primera muestra los valores nominales y la segunda los valores calculados con sensibilidad NLP cuando cambian los parámetros del problema.

**$** python test.py



## 3.4. GJH y otros solvers

GJH (Gradient, Jacobiann, Hessian) es un *pseudo-solver* que permite acceder a la información de gradientes, Jacobiano y Hesiana de un problema de optimización dados los valores de las variables primales y duales (<http://ampl.com/resources/hooking-your-solver-to-ampl/>). El resultado de ‘solucionar’ un problema de optimización con este *sovler* es un archivo de texto con extensión .gjh que contiene la información de gradientes, Jacobiano y Hesiana del problema como si fueran parámetros de un modelo creado en AMPL. Dado que este *pseudo-solver* utiliza componentes de AMPL es necesario tener una versión de este programa. En la página <http://ampl.com/try-ampl/download-a-demo-version/> puede descargar una versión demo de AMPL para Linux.

Después de descargar la versión demo de AMPL descomprima la carpeta y muévala a la ubicación que desee. Usando la *Terminal* diríjase al directorio *ampl* que se encuentra en la carpeta de AMPL que acaba de descargar

**$** cd amplide-demo/ampl

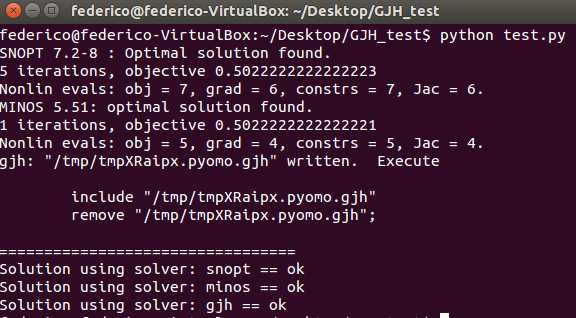
En este directorio ejecute los siguientes comandos desde *Terminal*:

**$** sudo chmod +x ampl gjh minos snopt

Por último, agregue el directorio …/amplide-demo/ampl al PATH de Linux como se ha hecho previamente. Como una consecuencia adicional a la instalación del *pseudo-solver* gjh el demo de AMPL también provee los *solvers:* cplex, gurobix, minos, snopt. Sin embargo, para los dos primeros no es claro que hace falta para ejecutarlos con Pyomo.

Puede que la versión que descargue de AMPL tenga otros directorios así que debe asegurarse de ejecutar los comandos anteriores en la carpeta donde se encuentran los ejecutables ampl, gjh, minos y snopt. Los puede reconocer como iconos morados en forma de rombo. Del mismo modo la carpeta donde se encuentran estos ejecutables es la que debe agregar al PATH.

Para probar que la instalación fue correcta ejecute el *script* *test.py* que se encuentra en la carpeta *GJH\_test*. Debe observar que en la consola se muestra que el problema llego a solución óptima y que se han creado archivos con extensión gjh.



# 4. Módulo (as)NMPC

Este módulo es una extensión de Pyomo que se desarrolló con el fin de facilitar el diseño y la evaluación de controladores NMPC ideales y de asNMPC (advanced step NMPC). El módulo funciona de forma similar a otros módulos de Pyomo donde primero se importa el paquete para acceder a sus elementos que ayudan a definir el modelo de optimización. Para conocer como se utiliza el módulo puede consultar el documento *Framework in Pyomo for the assesstment and implementation of (as)NMPC controllers*. Aunque esta es la primera versión del documento y se deben hacer algunas correcciones, este sirve de guía para conocer las funciones y el uso del módulo.

Antes de proceder con la instalación del módulo se deben asegurar de tener instalado *numpy, scipy*, *matplolib* y el *pseudo-solver* gjh los cuales son esenciales para el funcionamiento del módulo. Si no tiene las librerías de Python anteriores las puede instalar ejecutando la siguiente acción desde la *Terminal*:

**$** sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose

Adicional a estas librerías de Python el módulo NMPC requiere que se modifiquen algunos de los archivos del *core* de Pyomo, bien sea por que requiere de funcionalidades adicionales o por que se encontraron *bugs* en algunas funciones. La lista de modificaciones se resume en la siguiente tabla y en el archivo *PyomoModifications\_V4.1.10519* puede encontrar en detalle las modificaciones que se deben hacer. Adicionalmente en la carpeta *Modificaciones* puede encontrar los archivos que se deben modificar con los cambios ya realizados, solo debe copiarlos y pegarlos en el directorio correspondiente. Si no puede copiarlos en la carpeta de Pyomo por problemas de permiso revise los siguientes párrafos para habilitar los permisos de esta carpeta y así tener completo acceso a sus archivos.

|  |  |  |
| --- | --- | --- |
| Directorio | Archivo | Modificación |
| /usr/local/lib/python2.7/dist-packages/pyomo/dae | misc.py | \*En la función *update\_conset\_indexed\_component*  Agregar la opción de actualizar expresiones  \*Agregar la función *update\_expression*  \*Agregar la función *get\_index\_information*  \*Agregar la función *\_get\_idx* |
| /usr/local/lib/python2.7/dist-packages/pyomo/dae/plugins | colloc.py | \*Agregar la función *reduce\_collocation\_points*  \*Agregar la función *\_interpolation\_coeffs*  \*Al inicio del *script* importar la función *get\_index\_information* (from pyomo.dae.misc import get\_index\_information) |
| /usr/local/lib/python2.7/dist-packages/pyomo/environ | \_\_init\_\_.py | \*Agregar los nombres *iNMPC, iNMPC\_nl\_interface,* y *asNMPC* de tal forma que los módulos correspondientes puedan ser cargados |

Una vez hechas estas modificaciones copie las tres carpetas de los programas (iNMPC, iNMPC\_nl\_interface, asNMPC) en el directorio de Pyomo (/usr/local/lib/python2.7/dist-packages/pyomo). Tenga en cuenta que la carpeta de Pyomo puede tener restricciones para su modificación por lo que antes debería activar los permisos de edición de este directorio.

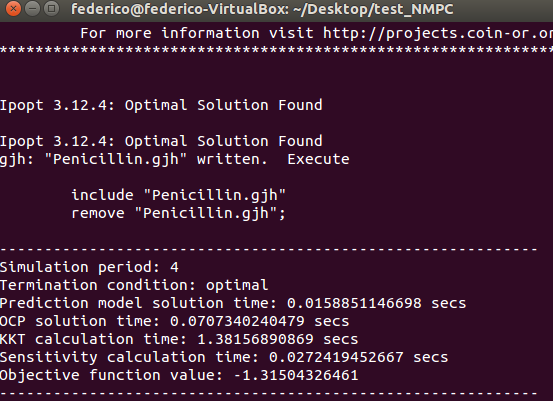
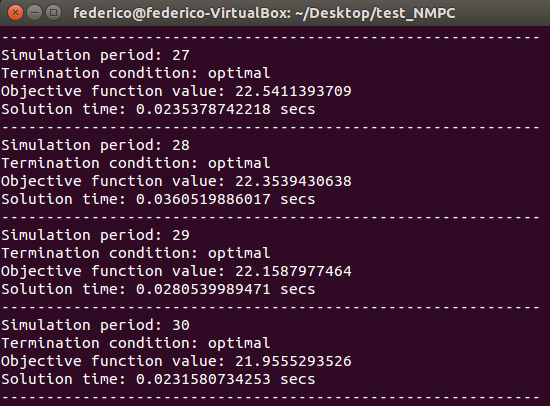
**$** sudo chmod 777 –R pyomo

Este debería ser suficiente para que pueda acceder a los módulos y usarlos desde un *script* de Python. Ahora, para probar las tres herramientas del módulo ejecute los tres *test* que encuentra en la carpeta *NMPC\_test* y deberá observar como se generan las gráficas de solución al problema a medida que el sistema avanza en un tiempo de muestreo.

**$** python test\_iNMPC

**$** python test\_iNMPC\_nl

**$** python test\_asNMPC



**NOTA:**

El módulo de controladores NMPC fue desarrollado como parte de un proyecto especial de maestría en la universidad y se encuentra en proceso de publicación. Si se utiliza este módulo en futuros trabajos por favor referencial el artículo *Framework in Pyomo for the assesstment and implementation of (as)NMPC controllers* como un artículo sometido a publicación, si para entonces este artículo ya se encuentra publicado por favor citar el documento de la revista.