Proyecto Fin de Carrera

Algoritmo de simulación de Dymola en Matlab para investigación de Modelos de Usuario deterministas en su aplicación a edificios

Autor: Luis Jarque Catalán

Director: Marcus Fuchs
Codirectora: Sara Manzano Martínez
Ponente: Carlos Monné Bailo

Ingeniería Industrial
Escuela de Ingeniería y Arquitectura
Mayo 2013

Resumen

ALGORITMO DE SIMULACIÓN DE DYMOLA EN MATLAB PARA INVESTIGACIÓN DE MODELOS DE USUARIO DETERMINISTAS EN SU APLICACIÓN A EDIFICIOS

Los modelos de representación de edificios son una herramienta importante para simular su comportamiento dinámico. La creación de dicho modelo requiere una gran labor de esfuerzo y tiempo, pero el objetivo principal es el estudio de la influencia de los parámetros sobre el edificio para su control o mejora.

Hasta el momento se han utilizado los perfiles de usuario aplicando lo establecido en la norma DIN 18599 y SIA 2014 para estimar las cargas internas del edificio debidas a personas, máquinas y luz y con ellas se simula el comportamiento energético del edificio.

Este proyecto aborda la estimación de los parámetros debido a cargas internas a partir de medidas reales de energía consumida de calefacción. La optimización consiste en un proceso iterativo de calibración del resultado de energía de calefacción de la simulación del modelo con datos reales de consumo. Este proceso se realiza mediante tres métodos.

El primer método corresponde a una función implementada dentro del programa de simulación Dymola. El segundo método es un programa desarrollado con Matlab que permite acoplar Dymola para la optimización de parámetros. El tercero consiste en el desarrollo de un algoritmo de Matlab que implementa el método matemático de Gauss-Newton por pasos.

Tras el desarrollo y aplicación de los métodos correspondientes se estudia su comportamiento sobre el modelo del edificio de referencia. El estudio sobre uno o varios parámetros del modelo de usuario permite obtener los resultados del proyecto y compararlos. Mediante el análisis de sensibilidad se estudia la dependencia entre parámetros del modelo de usuario y gracias a los resultados se concluye que los dos programas desarrollados con Matlab convergen con la solución óptima en menos tiempo y con menos iteraciones que Dymola.

El proyecto se ha desarrollado en el Centro de Investigación E.ON para el Instituto de Eficiencia Energética en Edificios y Clima Interior de Aachen (Alemania) en el ámbito del desarrollo de una herramienta de planificación integral de energía para un campus de Alemania. El presente proyecto puede ayudar en la estimación de parámetros de los edificios estudiados.

Agradecimientos

Agradezco al centro de investigación E.ON por haberme dado la oportunidad
de desarrollar mi experiencia profesional en el ámbito de las energías.
Mencionar a Marcus Fuchs por su seguimiento constante a lo largo del proyecto.

Agradezco a mi codirectora Sara Manzano por su disponibilidad e interés en todo momento.

En especial a mi familia, Lola y Jacinto, vuestro esfuerzo diario
me ha ayudado en la lucha por mis objetivos.
A ti Nieves, por todo lo que me has demostrado a lo largo de estos dos años.
A mis amigos.

Muchas gracias.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

1.1. Motivación

Las tecnologías de eficiencia energética son la llave para reducir la demanda energética. en especial, el sector de las energías muestra un gran potencial en la reducción de energías de calefacción y climatización.

Las simulaciones dinámicas son la herramienta perfecta para esta investigación. Sin embargo es necesario simplificar el edificio para que pueda ser simulado a un nivel adecuado y cumplir los requisitos en cuanto a convergencia, estabilidad, tiempo de cálculo y esfuerzo en la parametrización.

Normalmente, se dedica mucho tiempo a la creación de los inputs para la simulación de un modelo, pero tras ese trabajo, el usuario no determina los valores de los parámetros reales que conducen a un rendimiento del sistema óptimo. Esto puede ser porque no existe suficiente tiempo que perder para hacer el tedioso proceso de cambiar los valores de entrada, simular, interpretar los nuevos resultados y acertar cómo cambiar los inputs para la siguiente prueba o porque el sistema que se analiza es tan complejo que el usuario es solamente incapaz de entender las interacciones no lineales de los parámetros. Sin embargo, utilizando programación matemática es posible hacer una simple o múltiple aproximación con técnicas de búsqueda.

1.2. Objetivo

Cuando un técnico crea el modelo de un edificio real, intenta representarlo tan específico como sea posible en función de los datos que dispone. Tras la creación del modelo algunos parámetros importantes pueden ser optimizados para reducir la complejidad del modelo y conseguir un modelo igualmente fiable. Para esta acción necesitamos datos reales de medidas del edificio, entonces se puede comparar la simulación del modelo con los datos medidos y obtener la precisión.

Para este proyecto se parte de un modelo de edificio ya creado donde todos los parámetros referidos a materiales son bien conocidos. Sin embargo, hay parámetros como la presencia de personas, número de máquinas (ordenadores) o el ratio de luz (W/m^2) en cada una de las zonas que tienen un comportamiento estocástico y no posible de determinar exactamente. Estos tres parámetros componen las cargas internas y contribuyen a disminuir el consumo de calefacción. A lo largo del proyecto, citamos el concepto ,Modelo de Usuario, referido a la influencia y conducta de las cargas internas en el edificio.

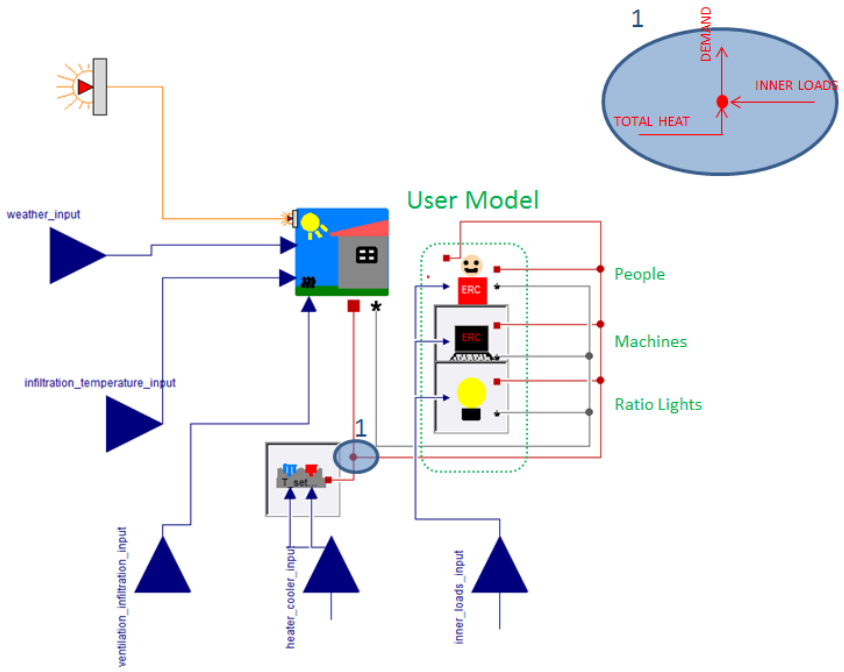


Figura 1.1: Modelo de usuario del edificio

Estos modelos de usuario se consideran determinísticos porque la evolución del consumo de calor está determinada por las condiciones iniciales que son las cargas internas, es decir, los inputs influyen en los outputs sin tener en cuenta la relación de ecuaciones. Ésto es porque no hay ecuaciones directas que relacionen cargas internas y consumo de energía de calefacción. De forma que la principal tarea consiste en calibrar resultados de simulación a los datos de demanda medidos.

El objetivo es desarrollar dos herramientas que permitan hacer la tarea anterior. Para ello se van a desarrollar dos programas en Matlab. El primero de ellos basado en la herramienta

de Optimización que incluye el programa y el segundo será un método matemático cuyas iteraciones también se realizarán con Matlab. Así pues ambos programas tienen que simular a través de Dymola el comportamiento del edificio a lo largo de un año para que después de las iteraciones necesarias se encuentre el valor de la variables independientes que se ajustan a unos criterios.

Estos dos métodos se compararán con otra herramienta de calibración implementada en Dymola. Esta herramienta tiene la misma función de calibración que las dos ya explicadas.

Los tres métodos deben estudiar la optimización de parámetros del modelo de usuario en distintas variantes y casos del modelo propuesto. Los criterios objetivo para la optimización son el tiempo necesario de cálculo, ajuste de parámetros, convergencia de los resultados y estabilidad de las optimizaciones.

Después de ésta evaluación, la comparación entre modelos informará del método que mejor se adapta a los modelos.

El programa de optimización a desarrollar en Matlab tiene que primero ajustar unos valores de parámetros, simular el modelo en Dymola y reducir la función objetivo, que es la suma de diferencias cuadradas entre simulación y datos medidos (2.1). Un programa óptimo debería priorizar también el tiempo de cálculo y ajuste adecuado así como el número de parámetros que pueden ser ajustados con precisión.

El método matemático también hecho en Matlab, tiene como objetivo linealizar la función objetivo con el método de mínimos cuadrados y determinar la dirección adecuada de los parámetros (aumento o disminución de valores) en cada iteración. Su metodología es paso por paso y eso nos permite un estudio más detallado del algoritmo.

Una vez desarrollados los tres métodos para todos los casos estudiados y a través de un análisis de sensibilidad se pueden garantizar las condiciones de operación para el cuál un método es estable. Una comparación entre los resultados obtenidos de cada método proporciona la información necesaria para ajustar un método en función del caso estudiado.

El primer caso incluye sólo un parámetro que representa las cargas internas como flujo de calor fijo (W). El segundo caso se refiere a los tres parámetros (personas, máquinas y ratio de luz). El último caso debería estimar el número de personas que hay en el edificio a lo largo de un día de oficina medido en horas (11 horas =11 parámetros) (1.1).

En el caso de que uno de los tres métodos fuera lo suficientemente eficaz, podría utilizarse como herramienta de optimización de parámetros para el centro de investigación de E.ON.

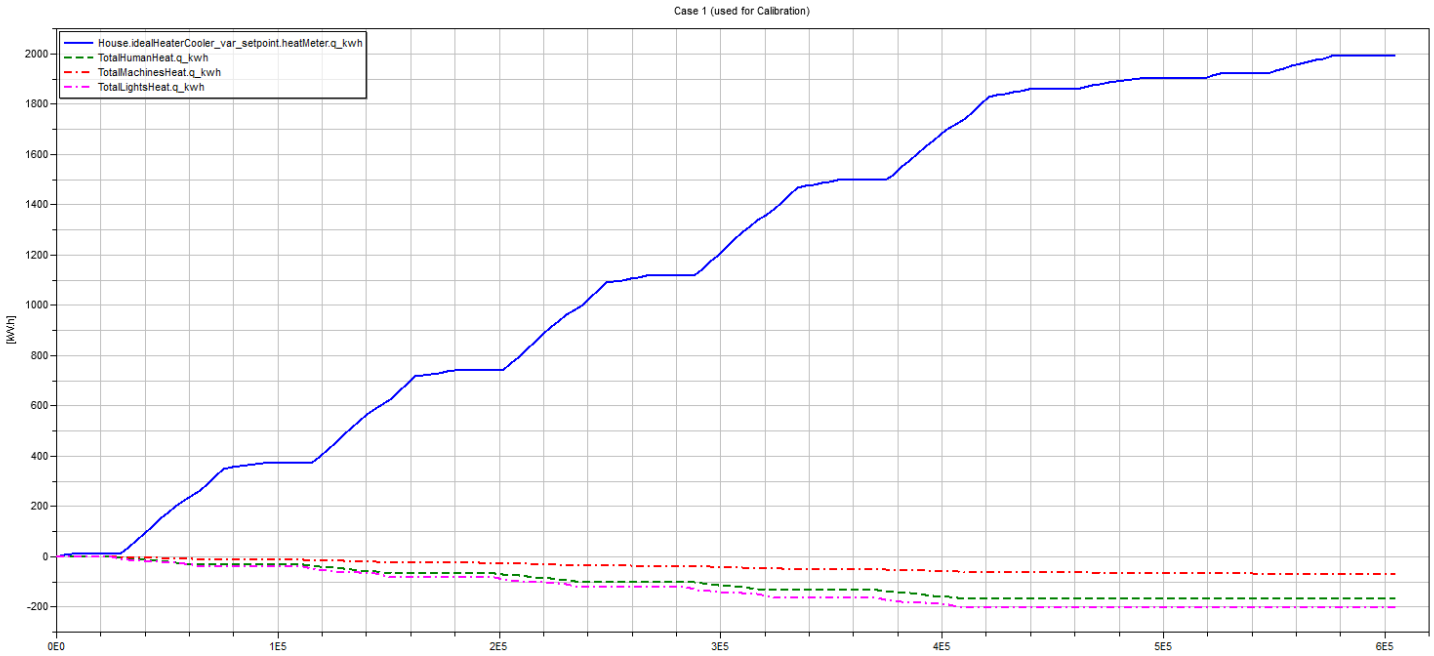


Figura 1.2: Contribucion de energia de las cargas internas durante una semana

1.3. Estructura de la tesis

La tesis se ha escrito en el mismo orden en el que se ha escrito. Primero se incluye una explicación de los aspectos más importantes para la optimización de parámetros en un modelo. El *capítulo 2* explica los métodos de optimización donde se puede ver una introducción a la optimización general y su importancia en ingeniería. Después, una explicación del edificio de referencia aclara todos los conceptos que se necesitan saber acerca del edificio concreto. Entonces se hace una valoración para la selección de los métodos a usar y en los siguientes apartados se detallan dichos métodos. El *capítulo 3* desarrolla los métodos explicados en el capítulo anterior. Se necesitan diferentes casos de estudios para determinar el comportamiento de los programas. Para ello se calibran uno, tres y once parámetros. El *capítulo 4* muestra un análisis de sensibilidad de los resultados y la influencia de los parámetros en los métodos. Sabiendo los resultados de simulación la comparación de métodos permite evaluar los programas implementados. El *capítulo 5* es la conclusión y consideraciones del proyecto. El *anexo A* incluye el código del método de optimización matemática con Matlab, mientras que el *anexo B* incluye el código de la función de matemática implementada en Matlab.

2 Métodos de optimización

2.1. Visión general de las optimizaciones

El uso de simulación de sistemas para problemas de análisis complejos de ingeniería envuelve muchas variables independientes y solo puede ser optimizado mediante optimización numérica.

Los estudios paramétricos pueden lograr un gran ajuste de tales sistemas, incluso cuando sólo aportan una mejora parcial en los modelos a pesar de que requieren una gran labor para su implementación. [Bernard P. Zeigler, 2000]

En estos estudios de parámetros se pretende optimizar una función objetivo con respecto a parámetros cuyos valores se desconocen y se han estimado para la creación del modelo. El proceso es iterativo y cada vez que se varía un parámetro otros dejan de ser óptimos y también necesitan ser ajustados. Es por ello que se descarta un proceso manual de dos o más variables independientes.

Los edificios y los sistemas de climatización diseñados pueden mejorar si se usa optimización numérica. Sin embargo, si una función objetivo que es suave se evalúa en los parámetros de diseño por un programa de simulación dinámica de edificios como es Modelica, dicha función se puede sustituir por una aproximación numérica que es discontinua en los parámetros de diseño.

Además, muchos programas de simulación de edificios no permiten obtener un límite de error para las aproximaciones numéricas a la función objetivo. Así pues, si una función es evaluada por tal programa, el algoritmo de optimización que depende de la suavidad de la función de objetivo puede fallar lejos de un mínimo. [Wetter u. Polak, 2003]

En la forma más general, los problemas de optimización se pueden indicar como: Sea X un conjunto de restricciones específicas ajustadas por el usuario y sea $f: x \rightarrow \Re$ la función objetivo definida por el usuario que está limitada desde abajo. La función objetivo $f(\bullet)$ mide el ajuste del sistema. El objetivo consiste en encontrar una solución al problema:

$$\min_{x \to X} f(x) \tag{2.1}$$

Este problema se resuelve por métodos iterativos, los cuales construyen infinitas secuencias de aproximación progresiva a la solución. [Wetter u. Wright, 2003]

Los parámetros de diseño en el estudio de nuestro problema de optimización son variables discretas sin restricciones de desigualdad.

2.2. Edificio de referencia

Para entender el caso de la tesis se necesita una descripción del edificio así como de las cargas internas que afectan al edificio. El edificio de referencia es un instituto de investigación de Alemania construido acorde a las normas estándares de Alemania (EnEV). El uso específico del edificio es el típico de oficinas. [M. Lauster, 2012]

El edificio consiste en dos plantas de oficinas. La planos en planta muestran seis zonas distintas con una área total de $1300m^2$ y un volumen de $3800m^3$ (fig: B.3). Las propiedades del edificio se conocen muy bien y están incluidas en hojas excel que e introducidas como inputs para determinar el modelo del edificio.

Los datos relacionados con medidas de energía del edificio también están disponibles. Éstos fueron suministrados por el promotor del proyecto del edificio de referencia y recogidos en una hoja Excel que contiene tiempo, potencia eléctrica, potencia de calefacción y energía de calefacción.

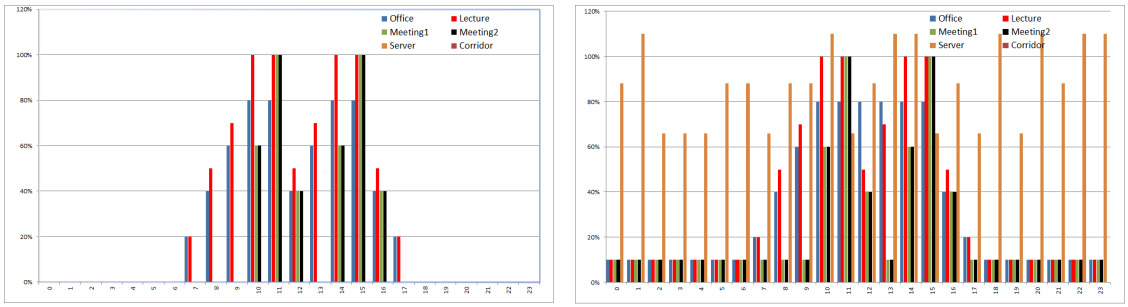
El fichero de medidas tuvo que ser adaptado a las necesidades del proyecto. Éste dispone de más de *40.000* medidas que fueron tomadas cada *15* minutos. En esta hoja había huecos de medidas que se resolvieron mediante interpolación de los datos correspondientes a energía acumulada de calefacción.

El modelo de usuario tiene una influencia en las cargas internas por la emisión de calor en el edificio. Así pues, el perfil de usuario muestra el comportamiento de las cargas internas (personas, maquinas y luces) a lo largo de un día medido en intervalos de una hora. El comportamiento diario se repite de lunes a viernes y es 0 el fin de semana porque nadie trabaja y las máquinas y luces se apagan (2.2(a), 2.2(b), 2.2(c)).

Los datos medidos usados para calibración corresponden a el consumo de energía de calefacción, columna 5 de la hoja Excel ,Heat_Counter, (fig: 2.1).

	A	B	C	D	E
1	Zeit (sec)	Elec_Power (W)	Elec_Counter (Wh)	Heat_Power_calculate (W)	Heat_Counter (Wh)
2	63504000	11184	39283881	20000	16539999
3	63507600	11061	39292791	0	16539999
4	63511200	10896	39301631	0	16539999
5	63514800	10853	39310441	0	16539999
6	63518400	10896	39319191	0	16539999
7	63522000	10926	39327951	0	16539999
8758	95025600	13238	122999682	0	38379998
8759	95029200	11868	123010372	10000	38389998
8760	95032800	9769	123019632	10000	38389998
8761	95036400	9764	123027482	10000	38399998
8762	95040000	9703	123035322	0	38409998

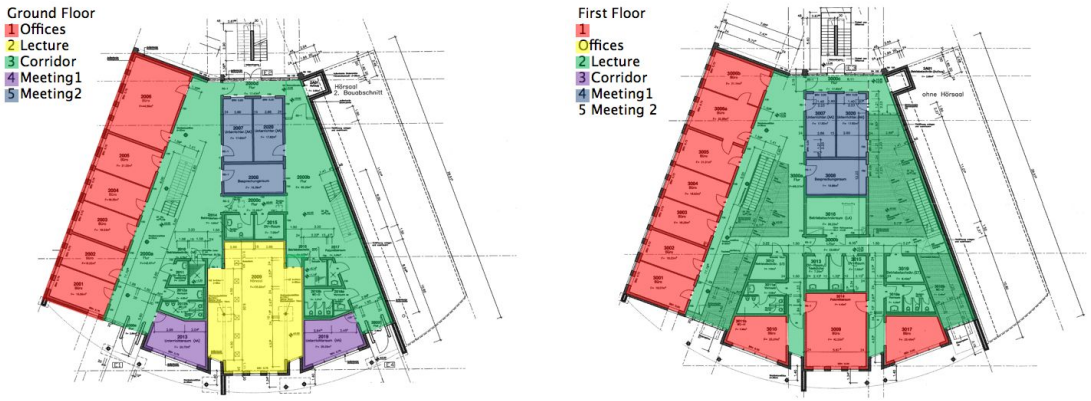
Figura 2.1: Datos medidos filtrados por horas



(a) Personas (b) Máquinas

(c) Luces

Figura 2.2: Cargas internas correspondientes al modelo complejo de seis zonas



(a) Planta baja (b) Primera planta

Figura 2.3: Diferentes zonas en las que se clasifica el edificio

La tabla (tab: 2.1) muestra la estimación de parámetros de personas, flujo de calor proporcionado por máquinas y el ratio de luz. Los valores correspondientes a al ratio se basan en la norma DIN 18599 y SIA 2014 y son los usados en el modelo creado como parámetros fijos. Sabiendo el consumo real del edificio, esos parámetros tienen que calibrarse en el sentido de reducir el cuadrado de la suma de doferencias entre consumo de energía para calefacción y consumo simulado por el modelo.

Cuadro 2.1: Estimación de parámetros de cargas internas

	Nr People (m^2/p)	**Machines** (W/m^2)	**Lights**(W/m^2)	**Area** (m^2)	**Persons**	**Machines**
Offices	14.0	7.0	15.9	371	26.5	26
Corridor	0.0	0.0	3.0	668.7	0.1	1
Meeting1	3.0	7.0	15.9	41	13.65	3
Meeting2	3.0	6.5	15.9	107.5	35.85	7
Lecture	3.2	4.7	13.0	105.8	33	5
Server	30.0	660.0	7.1	7.58	0.25	50

2.2.1. Métodos elegidos para la estimación de parámetros

El modelo de usuario estimado en 2.1 se usa como valores iniciales de nuestros parámetros antes de la estimación. Ésto lleva a incertidumbres en la simulación, sin embargo la comparación es una referencia de la habilidad de simulación para reflejar un edificio real con todas sus influencias.

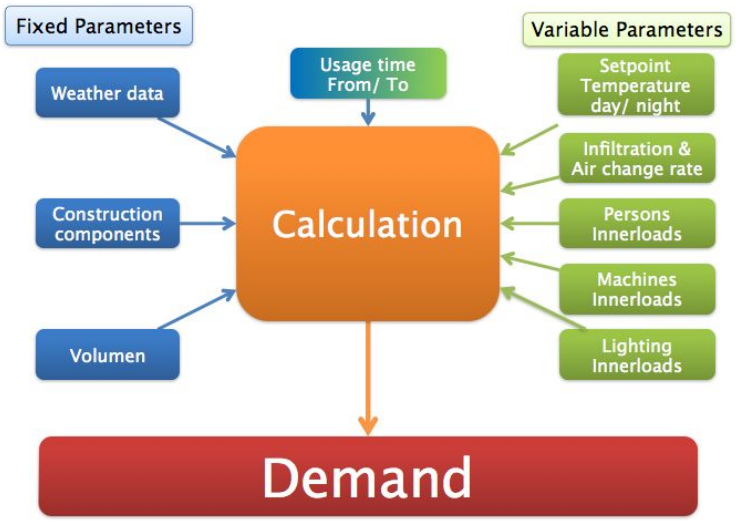


Figura 2.4: Interaction between parameters and demand

Los modelos estudiados contienen un gran número de parámetros cuyo comportamiento no se puede predecir. Los métodos de solución numérica en Modelica permiten reutilizar los modelos de simulación para optimización. Además los problemas de optimización dinámica no lineal se pueden tratar de forma eficiente como problemas discretos en el tiempo y resuelto numéricamente por aplicación de métodos no lineales de optimización a gran escala.[Bertsekas, 1999]

El objetivo es analizar diferentes métodos y comparar su ajuste para minimizar la diferencia anual demanda de calefacción en el edificio de referencia.

El modelo estudiado tiene todos los parámetros fijados y la función objetivo consiste en reducir un escalar calculado como la suma de diferencias al cuadrado. Además, los métodos tienen que considerar el consumo de energía horario sin ecuaciones diferenciales. Así pues, los métodos elegidos para estudiar son los siguientes por las ventajas que se muestran:

1. Función de calibración en Dymola
   - ▷ La función de calibración a está implementada en una librería de Dymola.
   - ▷ Rapidez y facilidad de preparación para la calibración.
   - ▷ Modelica programa el modelo y Dymola lo simula.
2. Herramienta de optimización de Matlab.

- ▷ El método Pattern Search representa una subclase de algoritmo de búsqueda directa en el cual se busca el minimizado de una función continua sin el uso de derivadas. [Audet u. J. E. Dennis, 2000]
- ▷ Numéricamente robusto abordando los criterios no suaves. [H. Elmqvist, 2005]
- ▷ Función de optimización implementada en Matlab.
- ▷ Ajuste de los limites de los parámetros.

3. Método matemático en Matlab (Levenberg-Marquardt method)
   - ▷ Minimiza la función objetivo de forma no diferenciable.
   - ▷ Aproximación del gradiente por diferencias finitas.
   - ▷ Método fiable y rígido para problemas de alto nivel.
   - ▷ Convierte el problema no lineal en uno lineal

2.3. Calibración del modelo con Dymola

La calibración del modelo consiste en la estimación de parámetros. Es este proceso los datos medidos de un dispositivo real se usan para ajustar parámetros de tal forma que los resultados de simulación tengan un buen acuerdo con los datos de medida. Los parámetros que ajustamos están referidos en esta herramienta como sintonizadores. Dymola varía los sintonizadores y simula para la búsqueda de soluciones satisfactorias. Matemáticamente, el proceso de ajuste de parámetros es un proceso de optimización para minimizar el error entre resultados de simulación y medidas a través de iteraciones.[AB, 2012]

2.3.1. Configuración de la calibración

Antes de sintonizar los parámetros de las medidas, debemos estudiar qué parámetros pueden ser estimados de las medidas disponibles. Cambiando un parámetro a estimar, debe influir en la salida. Sin embargo, dos o más parámetros pueden influir en el resultados de una forma similar (covarianza), los cuales no son posibles estimar individualmente. [H. Elmqvist, 2005]

2.3.2. Ajuste de parámetros

El proceso de ajuste de parámetros mediante la herramienta de Dymola se explica detalladamente en la **versión de la memoria en inglés** (B.3.1) ajustando los parámetros de optimiza-

ción para un modelo creado a partir del modelo de referencia, en el cual se han reducido las seis zonas a una con las mismas características.

Para ajustar los parámetros, primero se realiza una validación de la simulación. Ésto consiste en determinar si el modelo es una representación precisa de un sistema real y el grado en el que se asemeja sin tener que ajustar parámetros.

Tras la validación (fig: 2.5), se calcula un error del criterio de $4{,}7e6$ y una diferencia de energía total acumulada de *21.5 %*.

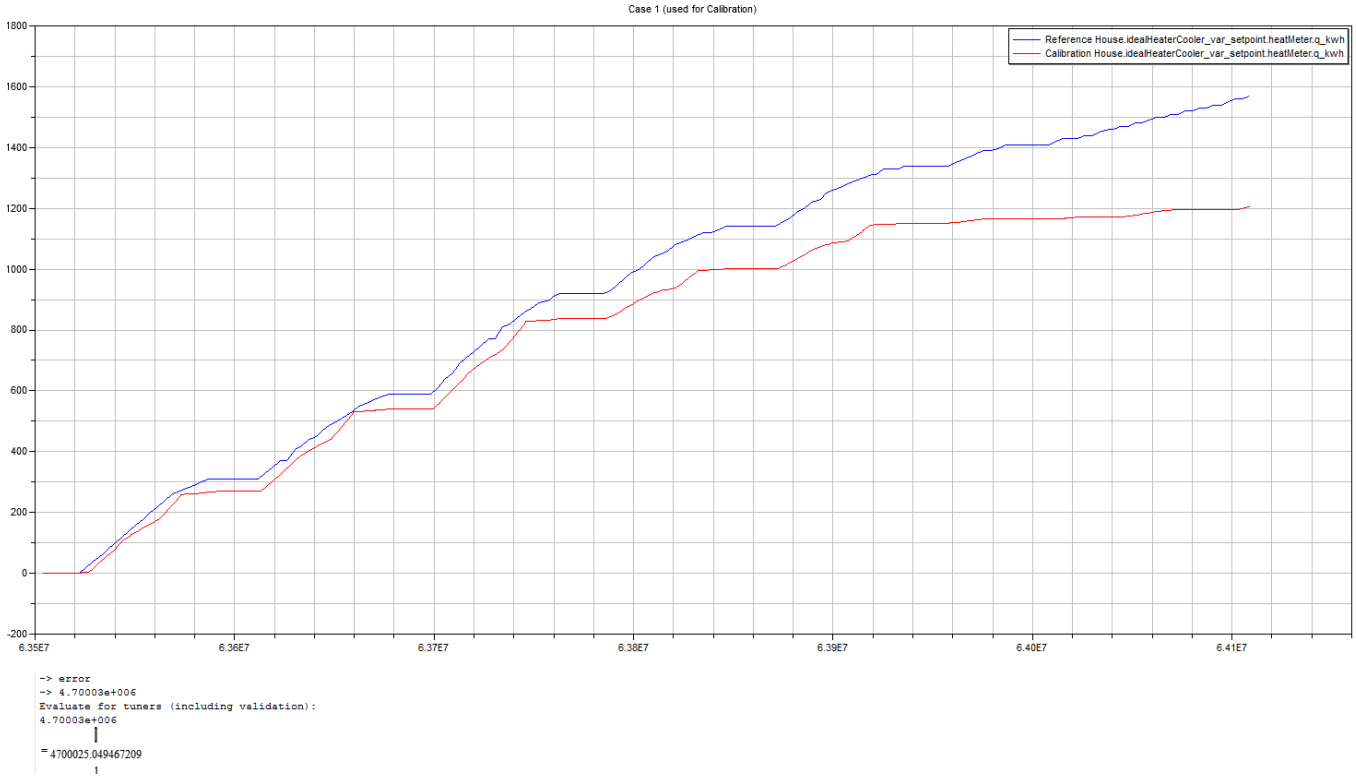
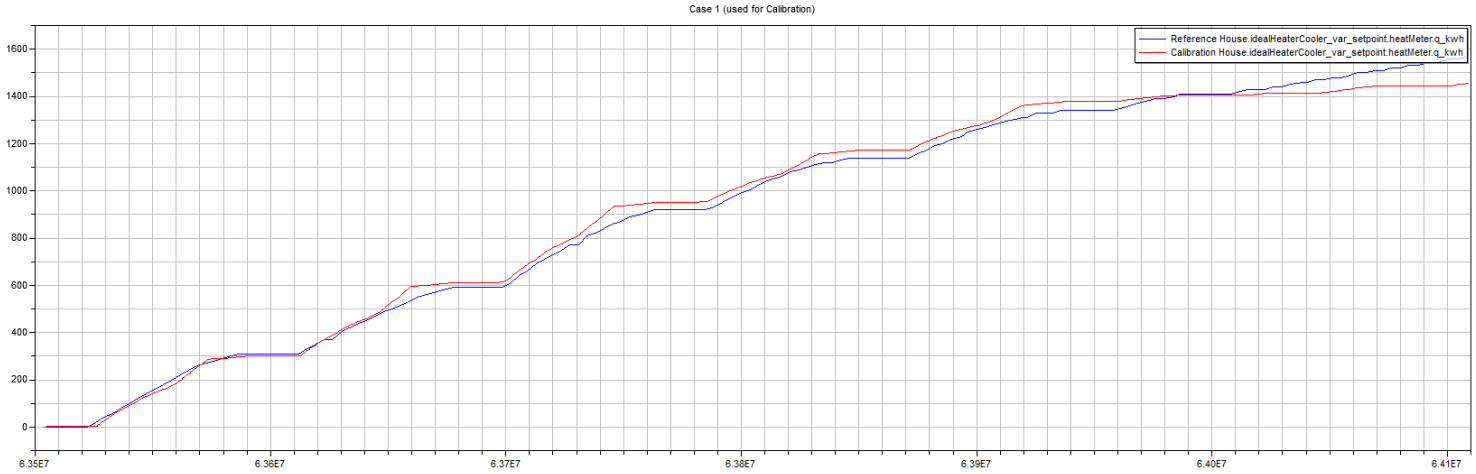


Figura 2.5: Resultados de validación para una semana

Una vez validado el modelo hay que seleccionar los parámetros para la calibración, que en este caso son los tres parámetros de las cargas internas estimadas con las normas DIN 18599 and SIA 2024 (tab: 2.1) donde el valor de los tres parámetros es (110, 41, 15.9) y fijamos los límites superior e inferior, se puede calibrar el modelo.

Tras 13 iteraciones Dymola devuelve una pantalla con los resultados. En el que se muestra el error del criterio ($2{,}44e5$). La diferencia es este caso de energía total acumulada se ha reducido hasta un *7 %* (fig: 2.6).

La sincronización afecta a los resultados disminuyendo el error del criterio en un 94,8 %.




```
House.zoneParam.NrPeople House.zoneParam.NrPeopleMachines House.zoneParam.LightingPower -> error
110                      41                               15.9                          -> 4.70003e+006
180                      20                               0                             -> 1.47845e+006
180                      20                               0                             -> 1.47845e+006
180                      58.4134                          0                             -> 258856
180                      52.8115                          0.102942                      -> 245642
180                      52.7421                          0.196558                      -> 244644
180                      52.7314                          0.21853                       -> 244579
180                      52.713                           0.223278                      -> 244574
180                      52.7101                          0.2274                        -> 244571
180                      52.7093                          0.229589                      -> 244570
180                      52.7091                          0.229839                      -> 244570
180                      52.7089                          0.230073                      -> 244570
180                      52.7088                          0.229993                      -> 244570
Evaluate for tuners (including validation):
244570
  {180.0, 52.70881834418674, 0.22999319434273063}
=
          |
  244569.89189935615
        105
```

Figura 2.6: Resultados de calibración para una semana

2.4. Herramienta de optimización de Matlab

La herramienta de optimización proporciona un amplio uso de algoritmos para optimización estándar y gran escala. Esos algoritmos resuelven problemas discretos continuos con y sin restricciones. La herramienta incluye funciones para programación lineal, cuadrática, optimización no lineal, mínimos cuadrados no lineales, sistemas de ecuaciones no lineales y optimización multiobjetivo.[The MathWorks, 2003]

El método Pattern Search se usa con la herramienta de optimización (fig: 2.7). Es un algoritmo basado en la búsqueda de el valor mínimo para la función objetivo. El algoritmo construye una malla que se evalúa de forma que actualiza el parámetro óptimo cada vez que una búsqueda se realiza con éxito. Poll ocurre cuando el paso buscado no está disponible para obtener un punto en la malla actual que disminuya el valor. Si no disminuye la función objetivo en los puntos de la malla alrededor de la iteración actual, la distancia entre puntos de la

malla se reducen a la mitad $\Delta x = \Delta x/2$, es decir, se refina la malla y el proceso se repite hasta encontrar el punto adecuado. [Audet u. J. E. Dennis, 2000]

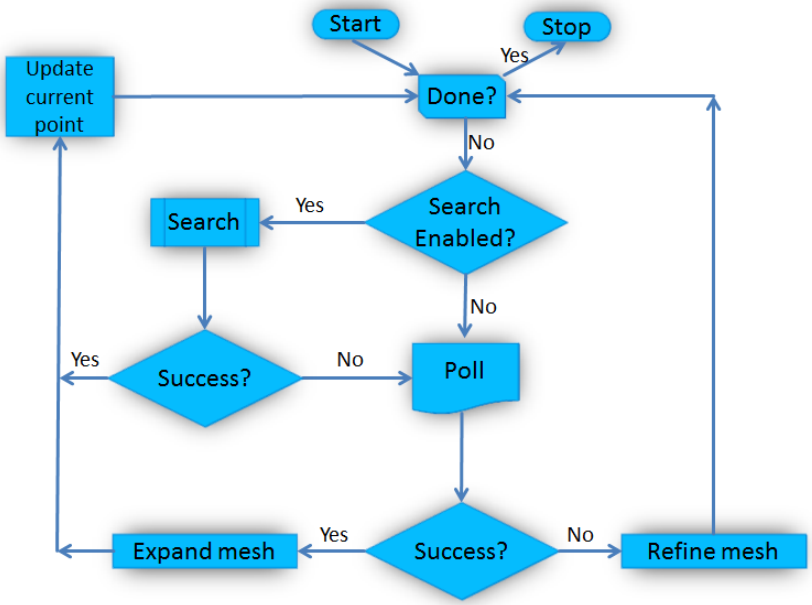


Figura 2.7: Basis of pattern search method

2.4.1. Conexión entre Dymola y Matlab

La conexión entre ambos programas es una parte importante del proyecto. El script de Matlab ,dymolaM, hace que desde Matlab se pueda ejecutar cualquier comando que pueda usarse en Dymola. De esta forma podemos mandar las siguientes ordenes para calibrar el modelo desde Matlab:

1. Transladar el modelo de Modelica a Dymola
2. Ajustar los parámetros estimados por Pattern Search
3. Simulación del modelo

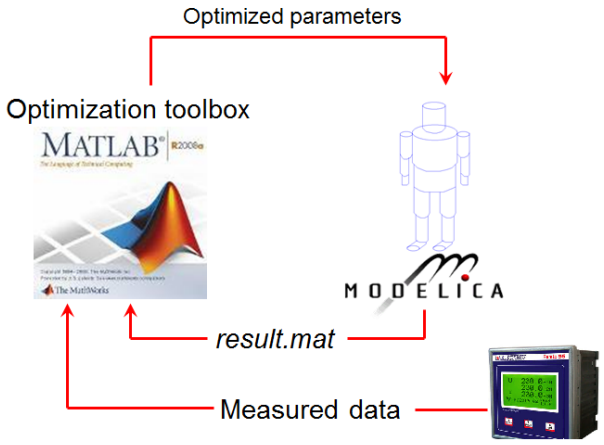


Figura 2.8: Conexión entre Dymola - Matlab

2.4.2. Algoritmo y optimización

El algoritmo de optimización desarrolado se ha desarrollado para ser más eficiente respecto a reducción en el tiempo de cálculo, precisión y estabilidad de resultados. Sin embargo, esta herramienta necesita un gran esfuerzo para ser implantada (Ver código de Matlab en 5. Añadir también que la única forma de conseguir un programa eficiente y eficaz es ajustar correctamente las opciones de la herramienta, este trabajo conlleva un trabajo duro de análisis del comportamiento del algoritmo en Matlab. (fig: 2.9).

	FixedHeatFlow (W)	error^2	Iterarions	function evaluations	time (sec)
Without options	16781	1.40126E+10	36	61	444
MeshAccelerator ,on, ; ScaleMesh ,off,	16781	1.40126E+10	57	94	716
Tol and ScaleMesh ,on,	16769	1.40127E+10	17	25	181
Cache and CacheTol (1e-10)	16769	1.40127E+10	17	19	136.8
Cache and CacheTol (eps)	16769	1.40127E+10	17	19	138.9
Plot	16769	1.40127E+10	12	15	117
ScaleMesh ,off,	slow				
MeshAccelerator ,off,	16769	1.40127E+10	14	19	140
TolBind , default,	16769	1.40127E+10	14	19	142
GSSPositiveBasisNp1	16769	1.40127E+10	12	15	113.8

Figura 2.9: Evaluación de las opciones para mejora del algoritmo

Por otra parte, una vez implementada la herramienta se puede adaptar de forma sencilla para diferentes casos mediante los siguientes pasos:

1. Modificación del script

- ▷ Dar los datos de medida real como fichero de matlab .mat (durante el proyecto no fue necesario cambiarlo).
- ▷ Ajustar las condiciones iniciales del vector de parámetros ($x0$).
- ▷ Ajustar los límites superior e interior (lb, ub) así como restricciones si fuera necesario (A, b, Aeq, beq).
- ▷ Ajustar la tolerancia de los resultados de la función objetivo ($Tolmesh : 1e-06$).

2. Modificación de la función
   - ▷ Dar el nombre de los parámetros cuyos parametros se cambiarán en Dymola.
   - ▷ Cambiar las características de simulación (tiempo a simular y tiempo entre intervalos, tolerancia y solver).
   - ▷ Cambiar el criterio de la función objetivo a minimizar ($|TotalHeat - ydata|^2$).

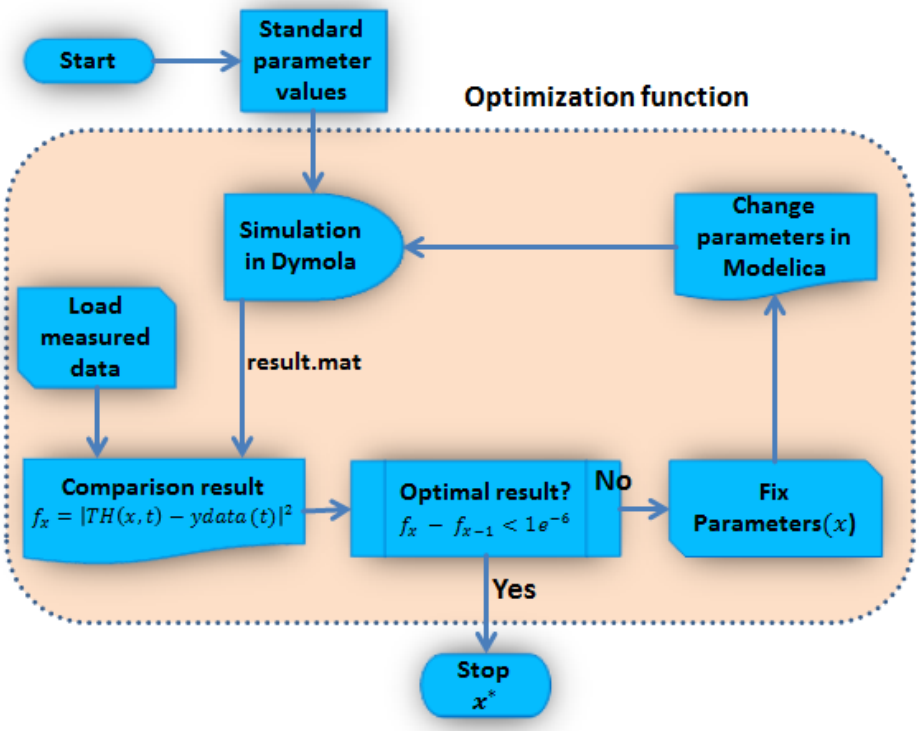


Figura 2.10: Programa desarrollado en Matlab para estimación de parámetros

En la función de optimización (fig: 2.10), los parámetros iniciales se cargan y simulan en Dymola y se comparan con los datos medidos para obtener la función objetivo. El programa

varía los parámetros y los cambia en Dymola para volver a simular y compara de nuevo con los datos medidos. A partir de ahí, el algoritmo elige el resultado de la función objetivo óptimo y sigue un proceso iterativo hasta que la resta de funciones objetivo es menor que *1e-6*.

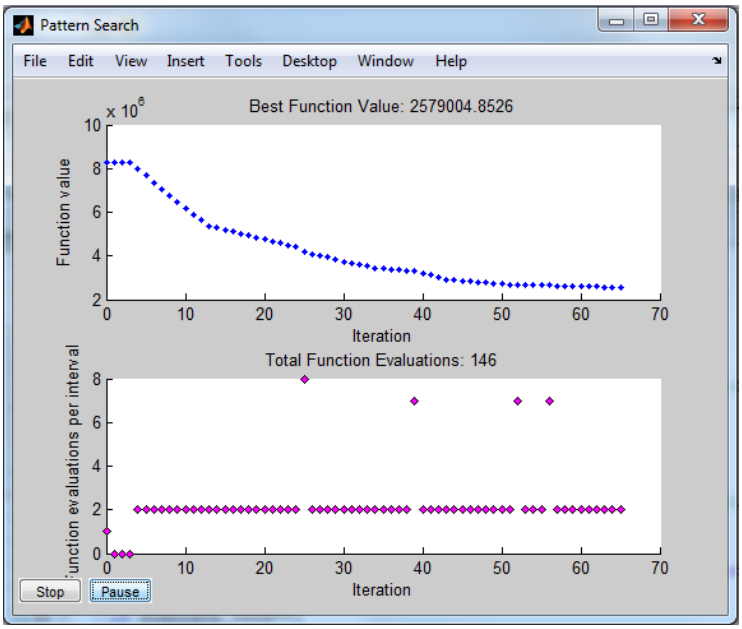


Figura 2.11: Valor de la función y evaluación en función de las iteraciones con Pattern Search

2.4.3. Precisión del algoritmo

El grado de optimización de parámetros se verifica mediante un proceso inverso. Es decir, con los parámetros por defecto del modelo se simula el modelo y la energía total de calefacción obtenida va a ser el input de datos medidos. Así pues, se quiere estimar 11 parámetros, correspondientes a las 11 horas que está abierto el edificio y que representan el porcentaje de la presencia de personas que hay en el edificio con un intervalo de una hora entre las 7 a.m. y las 18 a.m.. Se han estudiado dos casos, en el primero no se han fijado límites, es decir, la presencia de personas puede oscilar libremente entre 0 % y 100 % de la ocupación. La precisión de resultados respecto al modelo por defecto es del 96,8 % para una semana, en cambio de aprecia como el algoritmo a estimado las primeras horas de la mañana con mucha mayor presencia que las de mitad de día. Por ello se ha estudiado el segundo caso donde los límites se han fijado a ±20 %. Los resultados obtenidos reflejan una muy buena precisión del algoritmo del 99,1 %.

Cuadro 2.2: Presencia de personas en tanto por uno de forma horaria

Hour	7h	8h	9h	10h	11h	12h	13h	14h	15h	16h	17h
Model % ocup	0.2	0.4	0.6	0.8	0.8	0.4	0.6	0.8	0.8	0.4	0.2
Case 1 % ocup	0.93	0.76	0.49	0.49	0.51	0.49	0.49	0.49	0.49	0.49	0.49
Case2 % ocup	0.18	0.35	0.55	0.70	0.70	0.35	0.55	0.70	0.70	0.35	0.19

2.5. Método de optimización matemática

Los métodos anteriores utilizan procesos de optimización matemática para obtener la solución. Sin embargo, los métodos son tratados como una caja negra donde solo interesan los parámetros y los datos medidos como inputs para obtener la optimización de los parámetros como output.

El objeto del modelo se traslada normalmente a un sistema matemático de ecuaciones diferenciales y algebraico previo al tratamiento con solvers numéricos. Sin embargo el caso estudiado en este proyecto no puede ser escrito en ecuaciones diferenciables ya que sus parámetros no están directamente relacionados entre sí.

El método de mínimos cuadrados se ha elegido como método de resolución del problema, donde no se especifican límites en los parámetros. Así pues, el método de *Gauss-Newton* puede utilizarse gracias a su robustez en los resultados y fiabilidad en la convergencia.

La base teórica en la que se basa el método se referencia a la **versión de la memoria en inglés** (B.5).

El algoritmo ,Levenberg Marquardt, sigue pasos iterativos. Primero simula con los valores estándar de parámetros x y el incremento in x se fija en un 1 %. Lambda es una valor que se usa para determinar el tamaño del paso en cada parámetro. Aumentando lambda el tamaño del paso disminuye pero el tiempo de cálculo de la solución aumenta. El gradiente de la función ∇F se obtiene por diferencias finitas, en el que se necesitan tantas simulaciones como parámetros tiene, a pesar del tiempo que el método necesita para determinar en cada iteración el tamaño del paso y la dirección correcta. (fig: 2.12).

$$\frac{\partial f}{\partial x_1} = \frac{F(x + \Delta x_1) - F(x)}{\Delta x_1} \tag{2.2}$$

$$\nabla F = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

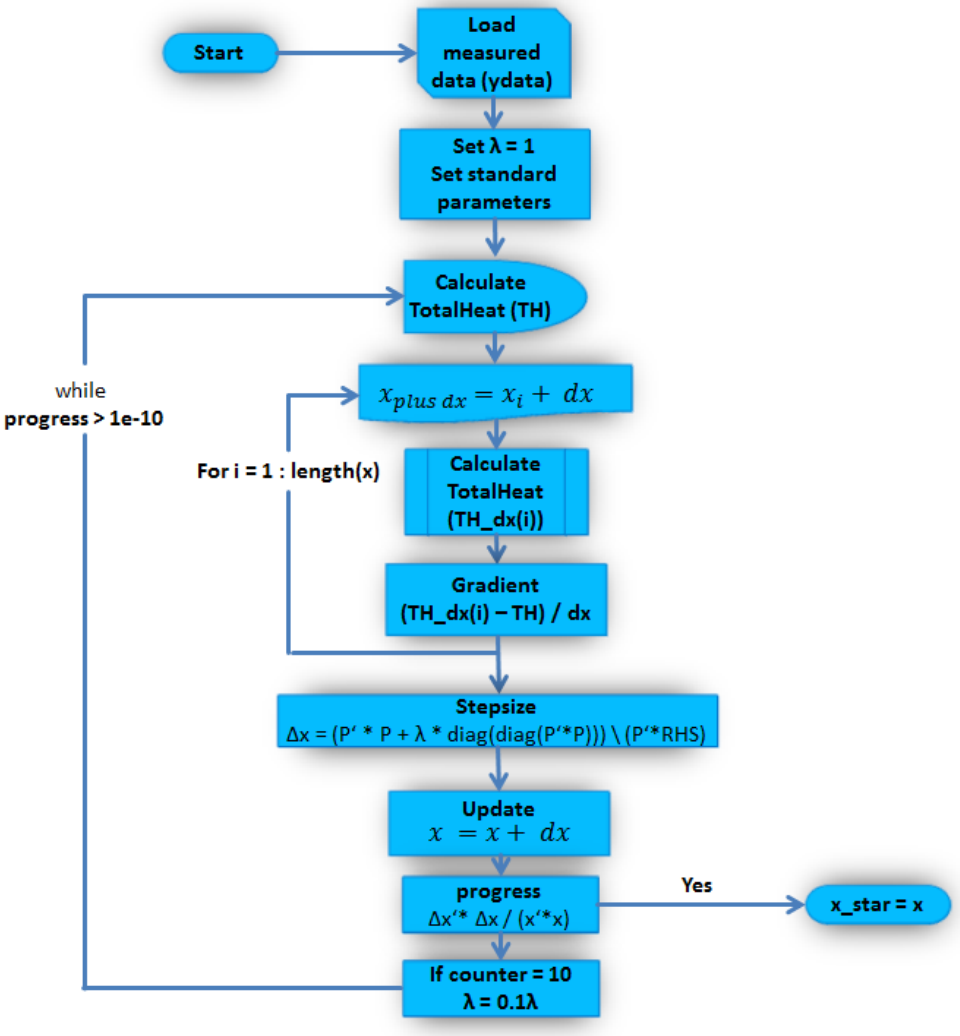


Figura 2.12: Método matemático desarrollado como una función de Matlab

3 Desarrollo de optimizaciones

Este capítulo muestra la aplicación de los tres métodos explicados en el capítulo anterior (2) sobre los casos necesarios para determinar el comportamiento de cada método, dependiendo del número de parámetros estudiados. También es importante conseguir un modelo tan simple como sea posible pero capaz de representar el edificio de referencia de forma estable.

3.1. Modelo de edificio simplificado: J1615 como una zona térmica

El modelo se ha simplificado de seis a una zona para investigar de una forma más sencilla pero menos precisa que el modelo complejo. Dicho modelo tiene las mismas características que el modelo de seis zonas (fig: 3.1). Esta modificación del modelo ha sido muy simple, ya que solo consiste en eliminar 5 de las 6 zonas del edificio de referencia y ajustar los parámetros de superficie total, volumen total y un input con un horario estándar que se ajusta al de las seis zonas. También se ha modificado el perfil de usuario total que se refleja en el correspondiente a máquinas. El servidor situado en una de las seis zonas es un equipo que genera una potencia térmica de unos $5kW_t$, este calor no se aprovecha en otras salas por ello el perfil de usuario referente a máquinas se reduce de *92* a *41* personas equivalentes. El de calor aportado por máquinas se expresa como personas equivalentes, con una cantidad de *100 Watts/pers eq* .

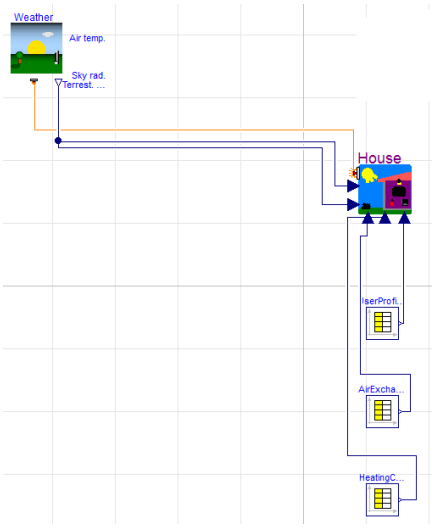


Figura 3.1: Edificio de referencia J1615 reducido a una zona

3.2. Estudio de parámetros fijos para el modelo de una zona

Se toma como base el modelo simplificado de una zona que previamente se ha obtenido del modelo de referencia, los siguientes casos proporcionan un primer estudio de los métodos donde dichos errores de calibración serán mayores que en el modelo complejo.

3.2.1. Un parámetro: flujo de calor fijo

El modelo más simplificado corresponde a sustituir todas las cargas internas por una potencia térmica equivalente a la contribución térmica sin eliminar el perfil de usuario. (fig: 3.2). Con estas modificaciones, el modelo estudiado tiene 827 parámetros en vez de 4581 que posee el modelo de referencia de seis zonas.

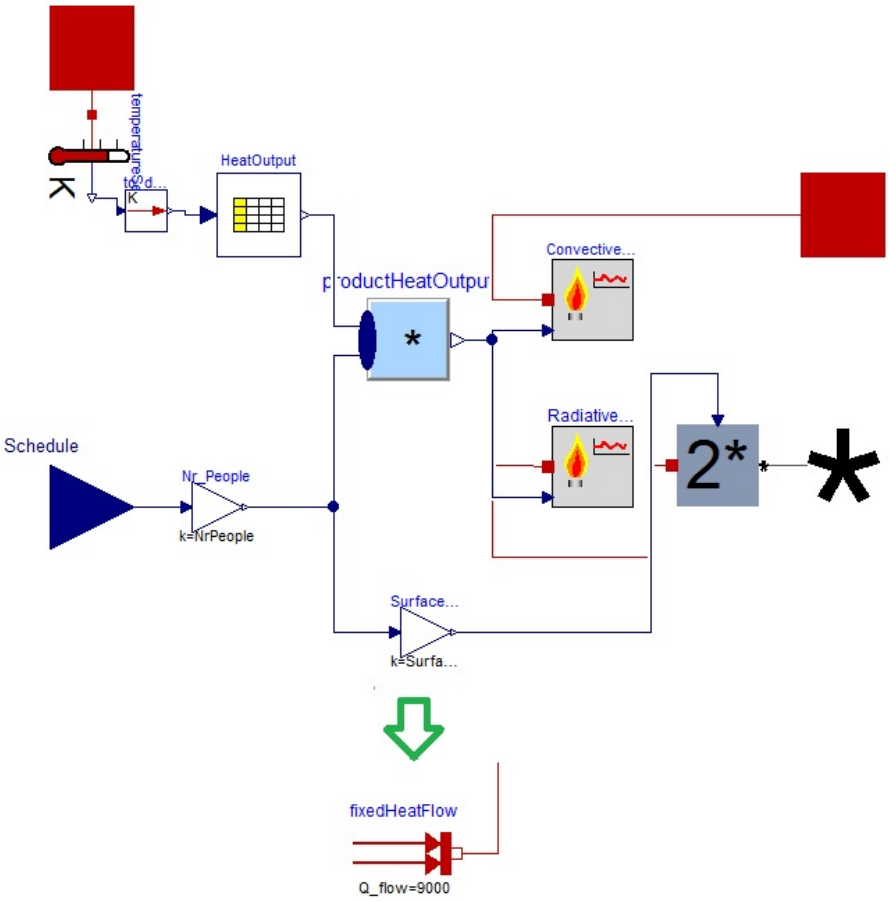


Figura 3.2: Simplificación de parámetros en el edificio de referencia

Tras la calibración para cuatro casos de una semana y un caso final de un año, el flujo de calor fijo y el error calculado por el método 2 de la herramienta de optimización de Matlab y el método matemático tienen muy buenos resultados del parámetro estimado y el error obtenido que concuerdan entre sí. Además el método matemático es el más rápido y necesita menos iteraciones para converger en la solución. (tab 3.1).

Cuadro 3.1: Resultados de calibración de una zona y un parámetro

	Time for simul.	**03.01-10.01**	**31.01-07.02**	**28.02-07.03**	**05.12-12.12**	**1 year**
FixedFlow (W)	Dymola	6299	9422	6169	6469	4346
	Toolbox Matlab	11720	7624	5576	5240	6392
	Math method	11698	7650	5553	5244	6396
$Error^2$	Dymola	5.79e5	9.29e5	5.92e5	1.45e6	6.69e9
	Toolbox Matlab	4.62e5	5.91e5	4.62e5	1.09e6	2.18e10
	Math method	4.69e5	5.91e5	4.62e5	1.09e6	2.18e10
Iterations	Dymola	25	29	17	13	25
	Toolbox Matlab	13	14	11	11	15
	Math method	5	8	4	4	5
Time (s)	Dymola	186	188	136	100	8312
	Toolbox Matlab	76	63	65	81	560
	Math method	16	30	18	14	336

3.2.2. Tres parámetros: personas, máquinas y ratio de luz

Para este caso se estudia la influencia de los tres parámetros sobre el consumo de calefacción. En realidad, estos parámetros estimados se multiplican por el porcentaje de presencia de las cargas internas en el edificio dado como input en un archivo como ,UserProfileOffice.txt, (fig: 3.4). El perfil se repite cada día de lunes a viernes y es *0* el fin de semana.

Lógicamente en el perfil de usuario cuanta más presencia de personas haya mayor número de máquinas estarán funcionando y más probabilidad de que las luces estén encendidas. Sin embargo, las medidas de calefacción son datos fijos que sirven para calibrar nuestros parámetros, de forma que un aumento en el número de personas implica un aumento en la potencia aportada por personas y una reducción en la correspondiente a máquinas y / o luz tab: 3.2).

Cuadro 3.2: Resultados de calibración de un año en el modelo de una zona y tres parámetros

	Dymola	Toolbox Matlab	Math method
Nr People	104	113	109
Machines	18	33	40
Light Ratio (W/m^2)	8.0	14.2	15.8
$Error^2$ (kWh)	6.57e9	2.07e10	2.07e10
Iterations	31	14	5
Time (s)	6520	1498	776

A pesar del bajo error obtenido por la herramienta de calibración de Dymola, requiere mucho

más tiempo e iteraciones. Los resultados de Dymola son interesantes porque los tres parámetros son inferiores a los obtenidos con los métodos de Matlab, esto es ilógico por una razón, el criterio para los tres métodos es la reducción de la suma de errores al cuadrado. En éste caso el método matemático es también tan preciso como el método de optimización de Matlab. Sin embargo, se a pesar de seguir siendo el más rápido, se necesita más tiempo para la estimación de los parámetros.

La siguiente gráfica es el resultado de aplicar los tres parámetros estimados con el programa de optimización durante una semana (fig: 3.3). La primera gráfica representa la potencia instantánea del equipo de calefacción; mientras que la segunda representa el número de personas y máquinas de forma dinámica; y la última gráfica muestra la potencia generada por las personas, máquinas y luces.

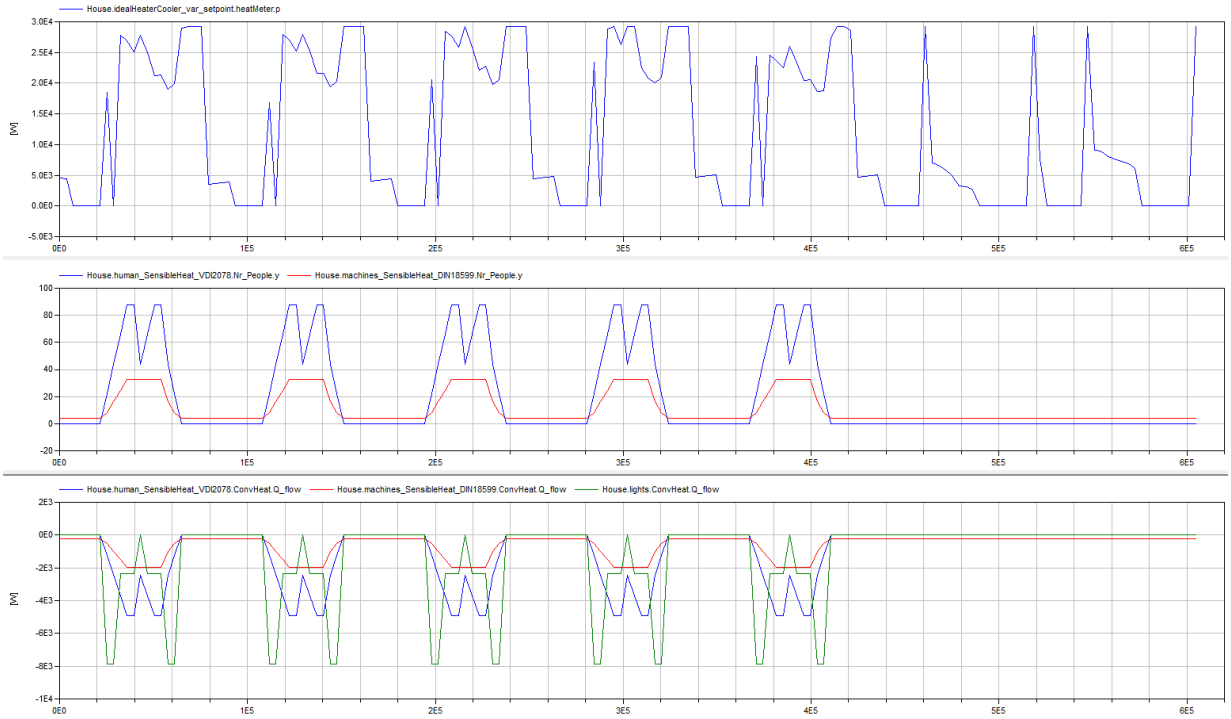


Figura 3.3: Resultados dinámicos de potencia térmica, valores de los parámetros y potencia de las cargas internas

3.2.3. Once parámetros: perfil de usuario de personas

El edificio de oficinas está abierto entre las 07 a.m. y las 6 p.m. y la ocupación a lo largo del día cambia constantemente, por ello el estudio horario de cargas internas debidas a personas

puede presentar una importante disminución en la función objetivo. Estas cargas las pueden modificar los programas creados con Matlab pero no la función de Dymola por tratarse de datos externos al programa de Dymola (fig: 3.4).

```
#1
double UserProfilesOffice(336, 5)
0       0       0.1     0       0
3540    0       0.1     0       0
3600    0       0.1     0       0
7140    0       0.1     0       0
7200    0       0.1     0       0
10740   0       0.1     0       0
10800   0       0.1     0       0
14340   0       0.1     0       0
14400   0       0.1     0       0
17940   0       0.1     0       0
18000   0       0.1     0       0
21540   0       0.1     0       0
21600   0       0.1     0       0
25140   0       0.1     0       0
25200   0.2     0.2     1       0.3
28740   0.2     0.2     1       0.3
28800   0.4     0.4     1       0.3
32340   0.4     0.4     1       0.3
32400   0.6     0.6     0.3     0.3
35940   0.6     0.6     0.3     0.3
36000   0.8     0.8     0.3     0.3
39540   0.8     0.8     0.3     0.3
39600   0.8     0.8     0.3     0.3
43140   0.8     0.8     0.3     0.3
43200   0.4     0.8     0       0
46740   0.4     0.8     0       0
46800   0.6     0.8     0.3     0.3
50340   0.6     0.8     0.3     0.3
50400   0.8     0.8     0.3     0.3
53940   0.8     0.8     0.3     0.3
54000   0.8     0.8     0.3     0.3
57540   0.8     0.8     0.3     0.3
57600   0.4     0.4     1       0.3
61140   0.4     0.4     1       0.3
61200   0.2     0.2     1       0.3
64740   0.2     0.2     1       0.3
64800   0       0.1     0       0
68340   0       0.1     0       0
68400   0       0.1     0       0
71940   0       0.1     0       0
72000   0       0.1     0       0
75540   0       0.1     0       0
75600   0       0.1     0       0
79140   0       0.1     0       0
79200   0       0.1     0       0
82740   0       0.1     0       0
82800   0       0.1     0       0
86340   0       0.1     0       0
86400   0       0.1     0       0
```

Figura 3.4: User profile for one zone

Los resultados para este caso se representan en el cuadro 3.3, donde la herramienta de optimización de Matlab gasta cinco veces más tiempo que el método matemático a pesar de que el error se reduce hasta prácticamente el mismo valor. El valor de los parámetros estimado por el método 2 es muy similar al del modelo por defecto debido a su restricción en los límites superior e inferior, mientras que el método 3 por no tener los límites establecidos tiene una estimación de parámetros muy diferida.

Cuadro 3.3: Calibration for eleven parameters

Hour	7h	8h	9h	10h	11h	12h	13h	14h	15h	16h	17h
Default % ocup	0.2	0.4	0.6	0.8	0.8	0.4	0.6	0.8	0.8	0.4	0.2
Toolbox Matlab	0.278	0.398	0.590	0.780	0.780	0.390	0.59	0.827	0.782	0.398	0.205
math method	0.421	0.426	0.410	0.450	0.435	0.418	0.427	0.401	0.423	0.460	0.428

	$Error^2$ (kWh)	Iterations	Time (sec)
Toolbox Matlab	2.06e10	65	13750
math method	2.07e10	6	2821

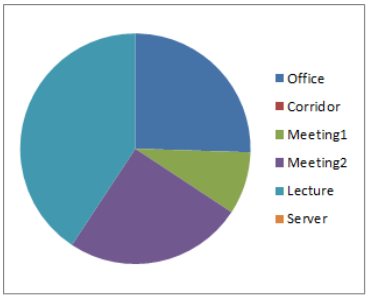
3.3. Modelo de edificio: J1615 como seis zonas térmicas

El modelo del edificio de referencia con seis zonas es el más complejo que se ha estudiado. Cada zona se comporta de forma diferente y posee su horario para el modelo de usuario. De esta forma, la energía total de calefacción se calcula como la suma de necesidades de cada

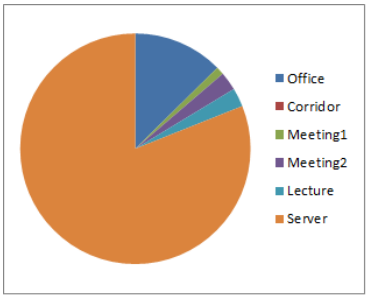
una de las zonas que dependerá a su vez y entre otros parámetros del perfil de usuario de esa zona (fig: 3.6).

Para la estimación de parámetros en este modelo se han fijado los porcentajes de cargas internas, de forma que solo es necesario estimar el número total de personas, máquinas y ratio de luz sobre el edificio y como ya se conoce la contribución de energía de cada zona y cada parámetro, los métodos desarrollados reparten el correspondiente del total a cada zona (fig: 3.5).

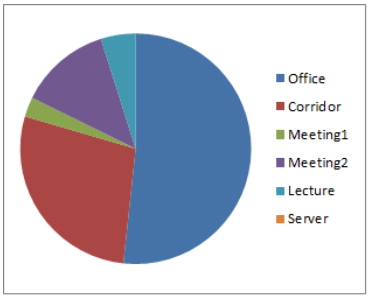
El modelo complejo implica un aumento en el número de parámetros de la estructura hasta *4581* parámetros. Esto significa cinco veces más ecuaciones que el modelo simple de una zona, es decir, aproximadamente el doble de parámetros por cada zona de más que tiene el modelo.



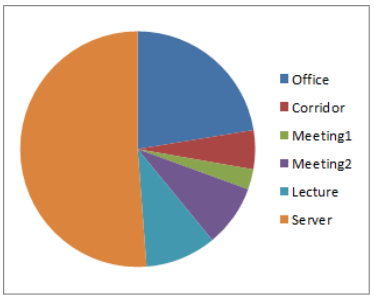
(a) Contribución debido a personas



(b) Contribución debido a máquinas



(c) Contribución debido a luces



(d) Contribución de todas las cargas internas

Figura 3.5: Reparto de energía total aportada por las cargas internas

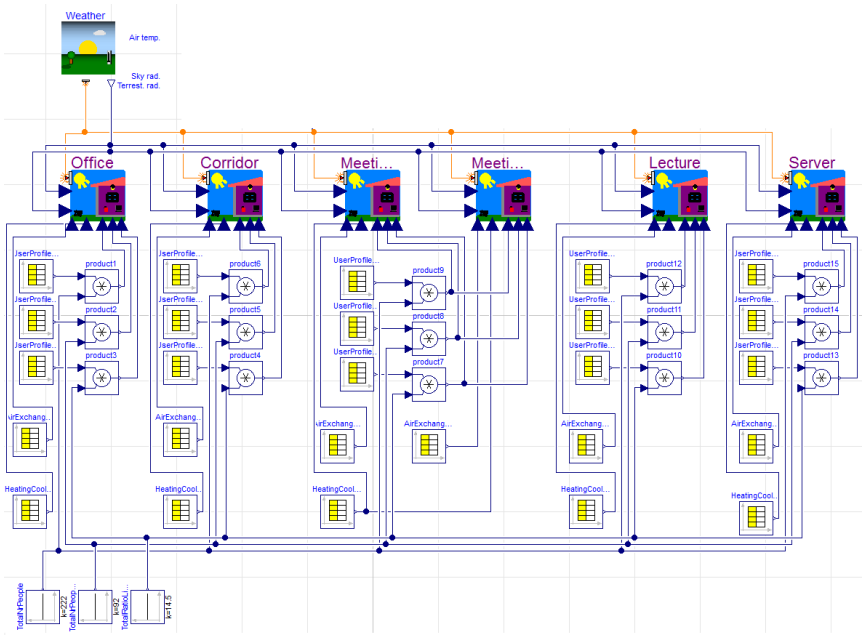


Figura 3.6: Edificio de referencia J1615 con seis zonas

3.4. Estudio de parámetros fijos para el modelo de seis zonas

3.4.1. Tres parámetros: personas, máquinas y ratio de luz

La estimación de parámetros en cada unos de los métodos aplicados es muy diferente. Ésto muestra la gran posibilidad de combinación ente parámetros que convergen en una solución mínima.

EL método matemático no es tan rápido comparado con el programa que utiliza el algoritmo Pattern Search y mucho más lento que los casos anteriores por la complejidad del modelo actual. La función de Dymola tiene el error más pequeño de los tres pero requiere 4 veces más tiempo y 9 iteraciones más que los dos programas de Matlab.

Cuadro 3.4: Calibración tres parámetros en el modelo de seis zonas para un año

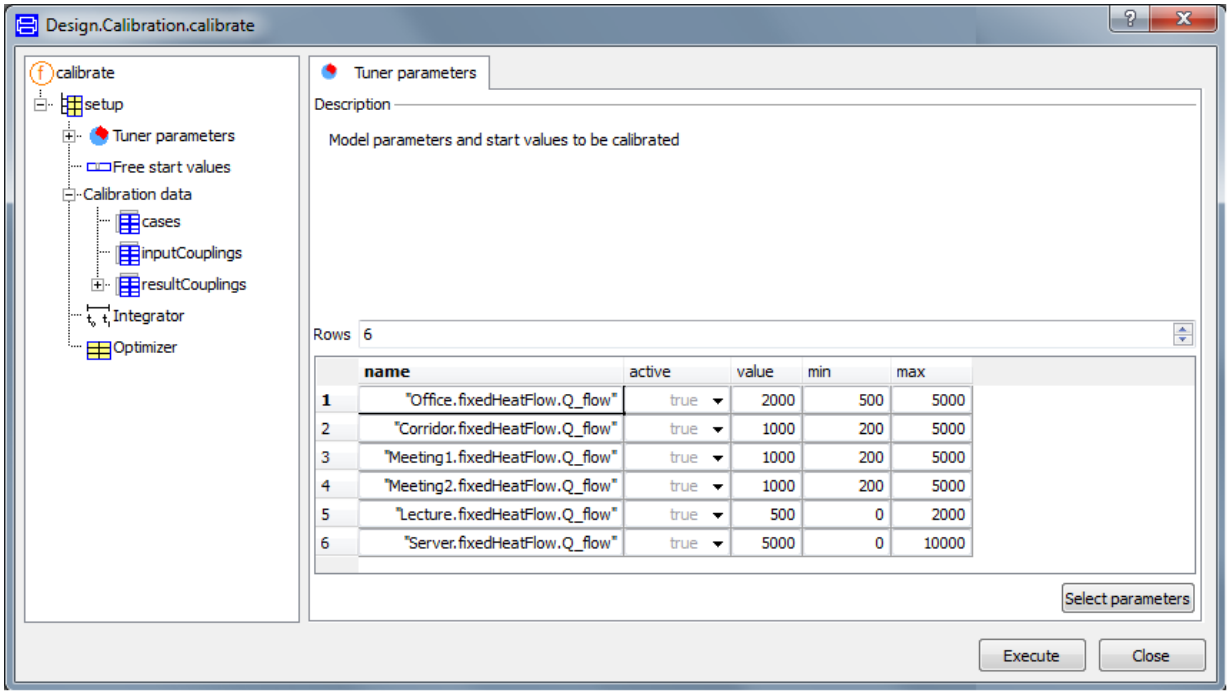
	Dymola	Toolbox Matlab	Math method
Nr People	26	222	117
Machines	102	92	268
Light Ratio (W/m^2)	25.0	14.5	13.7
$Error^2$ (kWh)	1.77e9	1.37e10	1.15e10
Iterations	23	14	14
Time (s)	11402	3420	3520

3.4.2. Seis zonas con flujo de potencia fijo

Se trata del mismo caso que el estudiado para el modelo simple de edificio 3.2.1 con la diferencia de las seis zonas y que hay un parámetro de potencia fija para cada una de las zonas del edificio. El principal objetivo del caso es la estabilidad de los métodos, especialmente la convergencia de la función de calibración con Dymola.

La calibración con Dymola fue imposible debido al fallo en la estimación de los parámetros (fig: 3.7). Ésto indica que el número máximo de parámetros que Dymola es capaz de estimar para nuestro modelo es 3.

Para este caso ambos métodos presentan estabilidad durante la simulación y son capaces de obtener resultados, pero no es necesario estimar los parámetros, ya que el objetivo es la comparación de los tres métodos.




```
Failed to expand Design.Calibration.calibrate(
  Design.Internal.Records.ModelCalibrationSetup(
    "Optimization2.J1615_FixedHeatFlow",
    {Design.Internal.Records.TunerParameter("Office.fixedHeatFlow.Q_flow", true,       2000, 500, 5000), Design.Internal.
    fill(Design.Internal.Records.FreeStartValues("", true, 0, -1E+100, 1E+100), 0),
    Design.Calibration.Internal.Dynamic_common(
      Design.Internal.Records.DynamicCommonCalibrationCases({"D:/mfu-lja/workspaces/onezone/optimierung/1615dinamicTable
      fill(Design.Internal.Records.DynamicCalibrationInputCoupling("", "", 1, 0),          0),
      {Design.Internal.Records.DynamicCalibrationResultCoupling("tEnergyMeterTotalHeat.q_kwh",          "Heat_Counter",
    Design.Internal.Records.CalibrationIntegrator(63504000, 95040000, 3600, 0,       "Dassl", 0.001, 0),
    Design.Internal.Records.CalibrationOptimizer(
      tolerance = 0.001,
      listOn = true,
      plotOn = true
    ))).
```

Figura 3.7: Fallo de calibracion con la función de Dymola

4 Resultados

4.1. Análisis de sensibilidad y peso de los parámetros

La dependencia entre parámetros y energía de calefacción no es linear, solo los modelos matemáticos consiguen hacer a los modelos lineales. Cambiando el número de personas a estimar en un edificio influye en la salida, sin embargo el número de máquinas y ratio de luz también influye en el resultado de forma que no se pueden analizar individualmente.

Cuando un conjunto de parámetros se sincroniza, el modelo se valida primero y los parámetros se ajustan de nuevo para que haya buena relación entre medidas y resultados de simulación.

Para unos datos medidos dados es posible conseguir buen ajuste incrementando la complejidad del modelo y el número de parámetros ajustados. Sin embargo, ésto no garantiza que el resultado también sea bueno para otras condiciones de operación.

El modelo de una zona tiene unos valores iniciales de los parámetros de *110* personas, *41* máquinas and *15.9* W/m^2 para el ratio de luz.

Tras el proceso que desarrolla el método de optimización (2.10) y teniendo en cuenta los datos reales de calefacción, los valores adecuados son respectivamente *113.5* personas, *33* máquinas y *14.25* W/m^2.

Por otro lado, mediante una función de Dymola que mide la sensibilidad de los parámetros se ha obtenido la siguiente dependencia entre parámetros:
$NrPeopleMachines + 0{,}8258 \cdot NrPeople + 8{,}5919 \cdot LightingPower$

Esta ecuación relaciona los tres parámetros entre sí. De forma que no existe solución única que minimice la función objetivo sino una línea a lo largo del valle con posibles combinaciones entre número de personas y máquinas y fijando el ratio de luz a *14.25* (fig: 4.1).

El peso de los parámetros indica como afecta el aumento en una unidad de cada uno de ellos sobre en las cargas internas. El peso de los tres parámetros son los siguientes: 7.9 % para personas, 9.6 % para máquinas y 82.5 % para el ratio de luz.

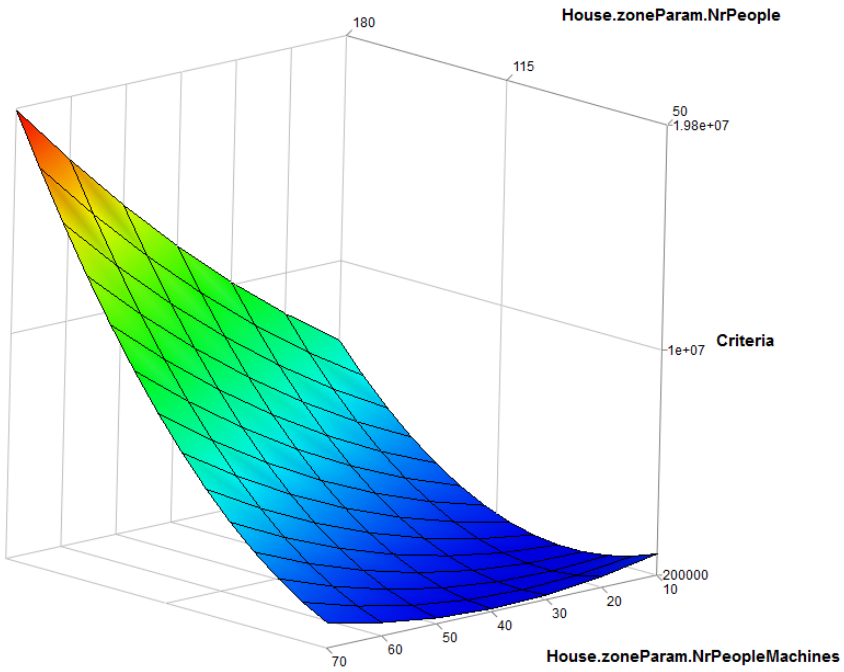


Figura 4.1: Solution after sweep two parameter on one zone building

En el caso estudiado para este análisis, la estimación de cargas internas para un edificio de oficinas es más efectiva que para un laboratorio, porque hay valores estándar conocidos. En cambio existen factores estocásticos cruciales que son imposibles de prever; los dos más importantes son la ventilación manual y el ajuste del termostato de temperatura (fig: 2.4).

El flujo de potencia de calefacción dependiente de los parámetros que varían con el tiempo se representa en la figura (4.2), y se puede observar que la influencia del intercambio de aire y las infiltraciones en el edificio hacen disminuir la temperatura en el sensor de temperatura por debajo de la fijada a $20{,}5^oC$ durante las horas de trabajo. Como consecuencia de esta disminución el equipo de calefacción se enciende hasta que la temperatura supera de nuevo los $20{,}5^oC$. Mientras que en el edificio no se trabaja, la temperatura de control se disminuye a 16^oC, a pesar de que nunca baja por debajo de $19{,}5^oC$.

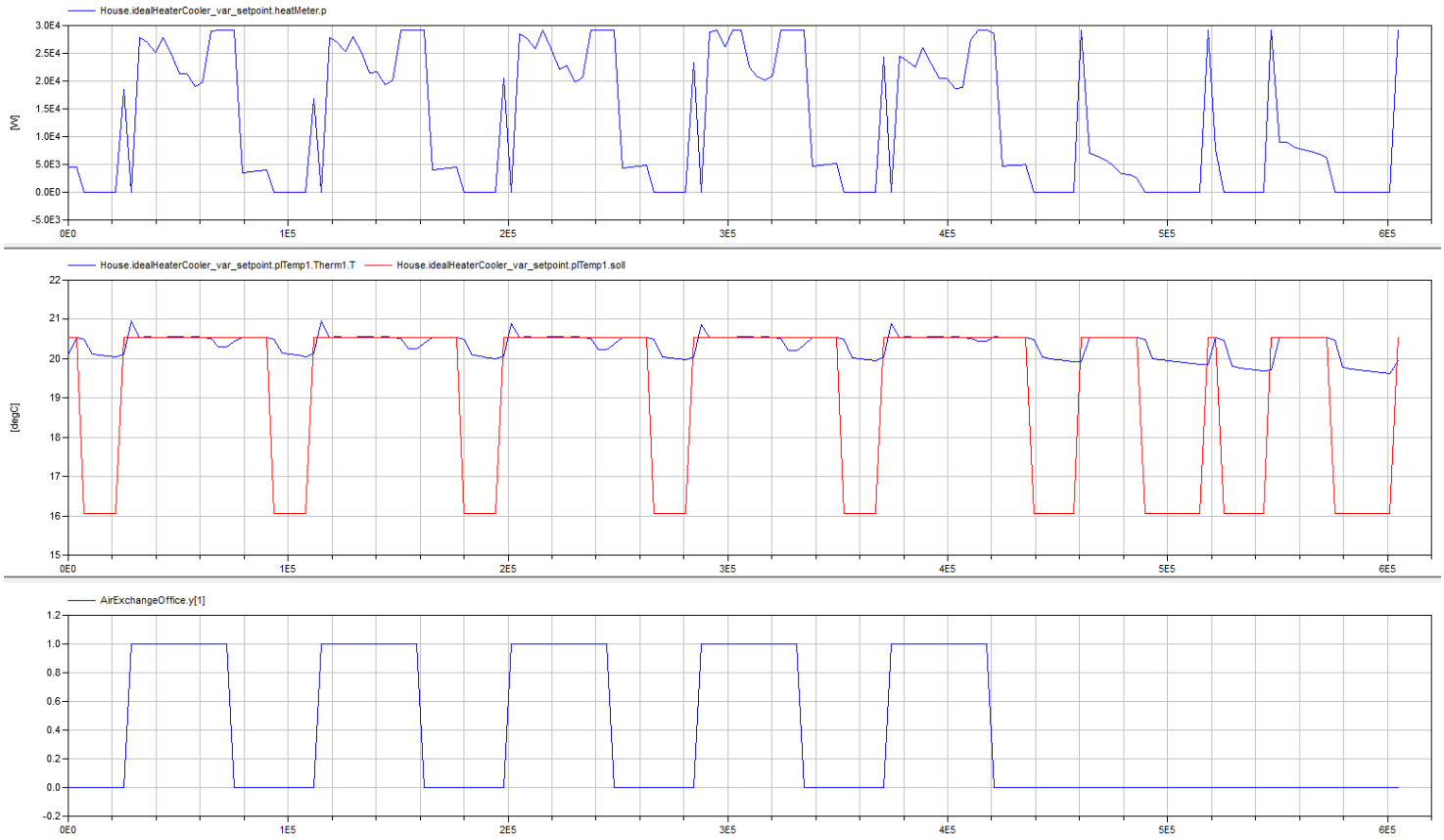


Figura 4.2: Variable parameters

4.2. Comparación de métodos y simulaciones

Gracias a las simulaciones de diferentes casos a lo largo del capítulo (3), se va podido analizar el comportamiento de tres métodos empleados. El análisis de los resultados de simulación pueden ayudar a determinar el mejor método usado para cada caso de estudio en función de las necesidades del usuario de programa.

1. Caso 1 (1 parámetro en una zona) (3.1).
   - ▷ La estimación de parámetros con el método matemático es muy rápida y converge rápido con la solución óptima para estudios de tiempo de una semana; además con una precisión mejor que la obtenida por el método de Dymola.
   - ▷ la precisión no es tan buena cuando el tiempo de estudio es de un año, a pesar de conseguirlo en escasos 5 minutos frente a las más de dos horas que necesita Dymola.

2. Caso 2 (3 parámetros en una zona) (3.2).
   - ▷ Para este caso los algoritmos creados en Matlab convergen en una solución donde los parámetros son diferentes, en cambio es un resultado muy lógico teniendo en cuenta que el ratio de luz tiene un peso ocho veces superior al de personas o máquinas.
   - ▷ El resultado de los parámetros obtenido con Dymola no es comparable porque es menor en los tres parámetros que los óptimos con los otros dos métodos.
   - ▷ La reducción en el número de parámetros fijos en el modelo de una zona permite que el método matemático sea el más rápido, exactamente nueves veces más que Dymola.
3. Caso 3 (11 parámetros en una zona) (3.3).
   - ▷ El error se reduce en ambos métodos hasta valores similares pero el método matemático necesita nueve veces menos tiempo y 59 iteraciones menos.
   - ▷ La estimación de parámetros para el método 2 de la herramienta de optimización con Matlab es muy similar a los parámetros por defecto del programa por su tolerancia del ±20 %.
4. Caso 4 (3 parámetros en seis zonas) (3.4).
   - ▷ El tiempo necesario para el método matemático aumenta ligeramente por encima del tiempo necesario por el método 2.
   - ▷ El cuadrado del error se reduce con el método matemático en un 16.06 %.
   - ▷ Cada método estima los parámetros de forma distinta.
5. Caso 5 (6 parámetros en seis zonas) (3.4.2).
   - ▷ El máximo número de parámetros que Dymola puede resolver es 3. Por eso Dymola devuelve un error con este estudio.

La simulación de un modelo simplificado necesita 3 veces menos tiempo y 5 veces menos número de parámetros, a pesar de ésto, la diferencia en los resultados no es muy notable (D.3). Comparando los resultados de calibración del modelo, la línea azul es la simulación de referencia y las otras tres líneas son casos estudiados, donde el ajuste de energía total de la línea roja (caso 1) es del *88.92 %*. El caso 2 representado por la línea verde tiene una adaptación a la de referencia del *90.43 %*. Mientras que el caso 4 es el que mejor se ajusta a la curva de referencia con un *91.11 %*.

Por otra parte, utilizando el método de optimización con Matlab para los cuatro primeros casos se muestra también diferencia en el comportamiento de los modelos. La suma de diferencias cuadradas medidas en kWh son respectivamente *2.18e10, 2.07e10, 2.06e10, 1.37e10*.

El caso con 3 parámetros en seis zonas es el modelo que genera mínimo error. Respecto a las iteraciones se necesitan respectivamente *15, 14, 65, 14*, mientras que el tiempo de simulación de cada caso es *560 seg, 1498 seg, 13750 seg, 3420 seg*.

Obviamente, usando el mismo método el error disminuye con el tiempo y el número de iteraciones, pero el usuario debe elegir el modelo correcto dependiendo de las prioridades tiempo o precisión.

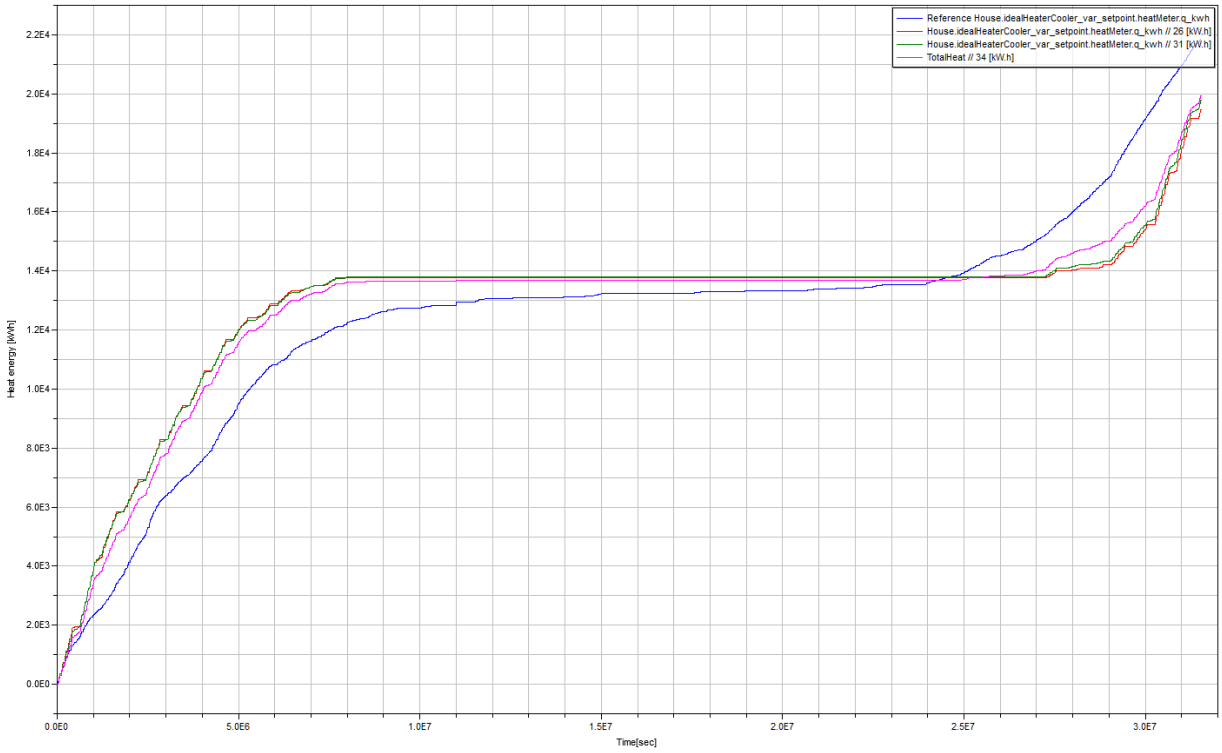


Figura 4.3: Simulación de la energía acumulada para 3 casos durante un año

5 Conclusión

Los programas de simulación térmica de edificios construyen una aproximación discontinua a una función de coste continuo diferenciable. Por esta razón, los algoritmos de optimización pueden fallar lejos de una solución óptima. En tales casos, usando aproximaciones de alta precisión podría requerir un gran tiempo de cálculo inviable si se usa para todas las iteraciones.

Los problemas de optimización dinámica no lineares se pueden tratar eficientemente como problemas discretos en el tiempo para control optimo y resolución numérica por aplicación de métodos de optimización no linear a gran escala.

Dymola es una herramienta que permite de una forma relativamente sencilla el ajuste de parámetros de calibración a medidas estáticas a pesar de su elevada necesidad de tiempo.

Cuando se resuelve sistemas no lineares de alto orden, el proceso puede divergir, porque la convergencia no está asegurada y la estimación inicial de parámetros se vuelve crítica para la convergencia.

En ambos programas desarrollados con Matlab, cuanto menor es la tolerancia entre simulaciones, mayor es el proceso de iteración para aumentar la precisión de la solución. Además, a pesar de que el algoritmo converja en un mínimo local o global, éste se puede alcanzar por la combinación adecuada de los parámetros (fig: 4.1).

Los dos algoritmos creados en Matlab son buenos candidatos para resolver los modelos examinados:

- El método ,Pattern Search, de la herramienta de optimización de Matlab utiliza un coste bajo y aproximaciones de baja precisión de la función objetivo cuando ésta se encuentra lejos de una solución. Sin embargo la precisión aumenta progresivamente conforme se acerca a la solución. Esto permite a la función objetivo la convergencia a un punto óptimo de primer orden.
- El método matemático ha demostrado ser el método más robusto y eficaz de los tres estudiados. El gradiente que se aproxima por diferencias finitas necesita tantas simu-

laciones como parámetros estima. Además el tiempo de calibración aumenta con el número de parámetros.

- ▷ Respecto a la reducción en el criterio por iteración el modelo matemático es claramente el más preciso.

Dymola permite estimar solo parámetros dados como inputs dentro del programa, pero no permite el ajuste de parámetros dados en ficheros externos. Ésto supone una gran desventaja a pesar de que se puede resolver creando una ÇombiTable.[en] Dymola cuyo inconveniente sería únicamente el tiempo de implementación.

Un aspecto muy importante a lo largo del proyecto es compara cuántos parámetros son capaces de calibrar cada método. En el caso de Dymola y para nuestros modelos solo se han podido calibrar hasta 3 parámetros mientras que los programas creados con Matlab pueden estimar más de 33 parámetros (11 horas por cada parámetro de estudio de cargas internas).

Trabajos futuros:
Algunas de las líneas futuras a estudiar se centran en:

- Aplicación de los algoritmos sobre otros parámetros determinantes como son el intercambio de aire y la temperatura sobre las zonas del edificio.
- Implementación de los mismos en paralelo para reducir el tiempo en la estimación de parámetros.
- Incorporar el método matemático desarrollado en la aplicación sobre otros edificios.

Anexo A: Código del script de optimización

```
%%%%%%%%SCRIPT FOR OPTIMIZATION ALGORITHM %%%%%%%%%
%Path for Dymola m files.
DymolaPath = 'C:/Program Files/Dymola 2013 FD01/Mfiles';

%Path for workspace.
MatlabPath = 'D:/mfu-lja/workspaces';
addpath(genpath(DymolaPath));
addpath(MatlabPath);
addpath([MatlabPath,'/onezone']);
addpath(genpath([MatlabPath,'/onezone/optimierung']));

%Change directory in Dymola.
onezone = [MatlabPath,'/onezone'];
dymolaM(['cd("',onezone,'/optimierung")']);

tic;
%Initial parameter values.
x0=[110 ; 41 ; 15.9];
%Lower and upper bounds.
lb=[90 10 0];
ub=[250 250 16];

format short e
%Optimization algorithm options.
opts = psoptimset('MeshAccelerator','on','ScaleMesh','on');
opts = psoptimset(opts,'TolMesh',1e-6);
opts = psoptimset(opts,'TolBind',1e-6);
opts = psoptimset(opts,'Cache','on','CacheTol',1e-10);
opts = psoptimset(opts,'PollMethod','GSSPositiveBasisNp1');
opts = psoptimset(opts,'SearchMethod',@positivebasisnp1, ...
    'PlotFcns',{@psplotbestf, @psplotfuncount});
%Optimization function Patternsearch.
[x,Fval,ExitFlag,Output] = patternsearch(
    @J1615_onezone3parameters_function_patternsearchH7,x0,[],[],[],[],lb,ub,[],opts);
```

```
fprintf('The parameters variated value was : %d\n', x);
fprintf('The number of iterations was : %d\n', Output.iterations);
fprintf('The number of function evaluations was : %d\n', Output.funccount);
fprintf('The best function value found was : %g\n', Fval);
tiempo = toc;
fprintf('The process took %d seconds', tiempo);\\
%%%%%%%%FUNCTION FOR OPTIMIZATION ALGORITHM %%%%%%%%%
function F= J1615_onezone3parameters_function_patternsearchH7(x)

MatlabPath = 'D:/mfu-lja/workspaces';
onezone = [MatlabPath,'/onezone'];

% load 1615dinamicTable.csv data.
load([MatlabPath,'/dataJ.mat']);

%If Simulate_week: 1= 03.01-10.01,  2= 31.01-07.02, 3= 28.02-07.03,
%4=05.12-12.12, 52=01.01-31.12

simulate_week=1;

if simulate_week==1
   %Week: 03.01.2011 - 10.01.2011
     ydata=(dataJ(49:217,5)/1000)-16540;

else if simulate_week==2
       %Week: 31.01.2011 - 07.02.2011
         ydata=(dataJ(721:889,5)/1000)-21570;

    else if simulate_week==3
           %Week: 28.02.2011 - 07.03.2011
            ydata=(dataJ(1393:1561,5)/1000)-25640;

         else if simulate_week==4
              %Week: 05.12.2011 - 12.12.2011
               ydata=(dataJ(8113:8281,5)/1000)-33740;

              else if simulate_week==52
                      %Year: 01.01.2011 - 01.01.2012
                       ydata=(dataJ(49:8809,5)/1000)-16540;

                  else num2str('The simulate week can not be optimized')
```

```
                end
            end
        end
    end
end

%dymolaM-command can execute every command that can be used in Dymola.
% Translate model J1615_onezone3Parameters.
res=dymolaM('translateModel("Optimization2.J1615_onezone3Parameters")');

% Parameters to set for optimization.
TotalNrPeople=x(1)
NrPeopleMachines=x(2)
RatioLights=x(3)

% Perform of parameter values in Dymola.
dymolaM(['House.zoneParam.NrPeople=',num2str(TotalNrPeople)]);
dymolaM(['House.zoneParam.NrPeopleMachines=',num2str(NrPeopleMachines)]);
dymolaM(['House.zoneParam.LightingPower=',num2str(RatioLights)]);

% Simulation of building.
if simulate_week==1
dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime=0, stopTime
    =604800, numberOfIntervals=0, outputInterval=3600, tolerance=1, resultFile="
    J1615_onezone3Parameters")');   %Simulation in Dymola with the value .k =x(1)
else if simulate_week==2
        dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime
            =2419200, stopTime=3024000, numberOfIntervals=0, outputInterval=3600,
            resultFile="J1615_onezone3Parameters")');
    else if simulate_week==3
            dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime
                =4838400, stopTime=5443200, numberOfIntervals=0, outputInterval=3600,
                tolerance=1, resultFile="J1615_onezone3Parameters")');
        else if simulate_week==4
                dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters",
                    startTime=29030400, stopTime=29635200, numberOfIntervals=0,
                    outputInterval=3600, tolerance=1, resultFile="
                    J1615_onezone3Parameters")');
            else if simulate_week==52
                    dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters",
                        startTime=0, stopTime=31536000, numberOfIntervals=0,
```

```
                        outputInterval=3600, tolerance=1, resultFile="
                        J1615_onezone3Parameters")');
                end
            end
        end
    end
end

%Loads data from result file .mat in a structure 'd'.
d=dymload([onezone,'/optimierung/J1615_onezone3Parameters.mat']);

%Extract from structure 'd' values of simulation.
TotalHeat= dymget(d,'House.idealHeaterCooler_var_setpoint.heatMeter.q_kwh');
TotalHeat= TotalHeat(1:(length(TotalHeat)-1));

F=sum((TotalHeat-ydata).^2)
```

Anexo B: Código de la función del algoritmo matemático

```
%%%%%%%FUNCTION FOR MATHEMATICAL ALGORITHM %%%%%%%%%
function [x_star] = levenberg_marquardt11
tic;

%Path of Dymola m files.
DymolaPath = 'C:/Program Files/Dymola 2013 FD01/Mfiles';

%Path of workspace.
MatlabPath = 'D:/mfu-lja/workspaces';
addpath(genpath(DymolaPath));
addpath(MatlabPath);
addpath([MatlabPath,'/onezone']);
addpath(genpath([MatlabPath,'/onezone/optimierung']));

%Change directory in Dymola.
onezone = [MatlabPath,'/onezone'];
dymolaM(['cd("',onezone,'/optimierung")']);

load([MatlabPath,'/dataJ.mat']);   %load 1615dinamicTable.csv data.
ydata=(dataJ(49:8809,5)/1000)-16540;   %measurement data for one year.

%solve nonlinear least squares problem using Levenberg-Marquardt
%algorithm.
%initial guess.
x = [0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5];

%increment in x for finite difference approximation.
delta_x = [0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];

%great lambda means short stepsize.
lambda = 1;

%counter
```

```
34 counter = 0;
35 progress = 1;
36
37 while progress > 1e-6
38   %counter
39   counter = counter+1;
40
41   % Simulation of Total Heat at vector of parameters 'x'.
42   % Call to dymola here with value of x
43   userprofil; % SCRIPT for editting the column corresponding to UserProfilesOffice.txt.
44
45   % Model translation
46   res=dymolaM('translateModel("Optimization2.J1615_onezone3Parameters")');
47
48   % Model simulation
49   dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime=0,
         stopTime=31536000, numberOfIntervals=0, outputInterval=3600, tolerance=1,
         resultFile="J1615_onezone3Parameters")');
50
51   d=dymload([onezone,'/optimierung/J1615_onezone3Parameters.mat']);
52   TotalHeat= dymget(d,'House.idealHeaterCooler_var_setpoint.heatMeter.q_kwh');
53   TH= TotalHeat(1:(length(TotalHeat)-1));   %[8761x1]
54
55   % Simulation of Total Heat at x+dx for each parameter.
56   for k = 1:length(x)
57
58       % x plus dx for parameter k.
59       x_plus_dx = x(k) + delta_x(k);
60
61       % Simulation of Total Heat at x plus dx.
62       x(k)=x_plus_dx;
63       userprofil;
64       x(k)= x_plus_dx - delta_x(k);
65       res=dymolaM('translateModel("Optimization2.J1615_onezone3Parameters")');
66       dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime=0,
             stopTime=31536000, numberOfIntervals=0, outputInterval=3600, tolerance=1,
             resultFile="J1615_onezone3Parameters")');
67
68       d=dymload([onezone,'/optimierung/J1615_onezone3Parameters.mat']);
69       TotalHeat= dymget(d,'House.idealHeaterCooler_var_setpoint.heatMeter.q_kwh');
70       TH_dx(:,k)= TotalHeat(1:(length(TotalHeat)-1)); % TH_dx [8761x11]
```

```
71
72          F=sum(abs(TH_dx(:,k)-ydata).^2)  %criterion
73      end
74
75      %Right hand side
76      RHS = ydata - TH;     %[8761x1]
77
78      %gradient
79      for k = 1:length(x)
80
81          %Partial derivative in k-direction.
82          P(:,k) = (TH_dx(:,k) - TH)./delta_x(k);   %P [8761x11] each column is divided by
                the column delta_x
83      end
84
85      %stepsize
86      delta_x = (P'*P + lambda*diag(diag(P'*P)))\(P'*RHS); %[11x1] = {([11x8761][8760x11])
            + [11x11]} \ [11x1]
87
88      %update
89      x = x + delta_x;
90
91      %progress
92      progress = (delta_x'*delta_x)/(x'*x); %[1x11][11x1] / [1x11][11x1]
93
94      if mod(counter,10) == 0
95          disp(['Counter : ',num2str(counter)]);
96          lambda = 0.1*lambda;
97      end
98  end
99
100 %optimal solution
101 x_star = x
102
103 %number of iterations
104 disp(['Number of iterations : ',num2str(counter)]);
105
106 tiempo = toc;
107 fprintf('The process took %d seconds', tiempo);
108
109
```

```
%%%%%%%%SCRIPT USERPROFIL %%%%%%%%%
file = [MatlabPath,'/onezone/optimierung/Tables/J1615'];
filename = [file,'/UserProfilesOffice.txt'];

Time=[0;3540;3600;7140;7200; ... ;597600;601140;601200;604740;];
People= [0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);x(2);x(3);x(3);x(4);x(4);x(5);x(5);
    x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x(10);x(11);x(11)
    ;0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);x(2);x(3);
    x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x(10);x(11);
    x(11);0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);x(2);
    x(3);x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x(10);x
    (11);x(11);0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);
    x(2);x(3);x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x
    (10);x(11);x(11);0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1)
    ;x(2);x(2);x(3);x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x
    (10);x(10);x(11);x(11);0;0;0;0;0; ... ;0;0;0;0;0;0;];
Machines=[0.1;0.1;0.1;0.1; ... ;0.1;0.1;0.1;0.1;];
Light1=[0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;1;1;0.3; ... ;0;0;0;0;];
Light2=[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0.3;0.3;0.3; ... ;0;0;0;0;];
A=[Time,People,Machines,Light1,Light2];


fileID = fopen([file,'/UserProfilesOffice.txt'],'wt');
fprintf(fileID,'#1\n');
fprintf(fileID,'double␣UserProfilesOffice(336,␣5)\n');


for i=1:length(A)
    fprintf(fileID, '%f\t%f\t%f\t%f\t%f\n',A(i,:));
end

fclose(fileID)
```

Bibliografía

[AB 2012] AB, Dassault S.: *Dymola: Dynamic Modelling Laboratory User Manual. Volume 2*, 2012

[Audet u. J. E. Dennis 2000] AUDET, Charles ; J. E. DENNIS, JR: Pattern Search algorithms for mixed variable programming. (2000), S. 573–594

[Bernard P. Zeigler 2000] BERNARD P. ZEIGLER, Tag Gon K. Herbert Praehofer P. Herbert Praehofer: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic System. (2000)

[Bertsekas 1999] BERTSEKAS, Dimitri P.: *Nonlinear Programming*. 1999

[H. Elmqvist 2005] H. ELMQVIST, S.E. Mattsson D. Brueck C. Schweiger D. Joos M. O. H. Ollson O. H. Ollson: Optimization for Design and Parameter Estimation. (2005), S. 255–266

[M. Lauster 2012] M. LAUSTER, M. Fuchs R. Streblow D. M. J. Teichmann T. J. Teichmann: Dynamic building model for city quartier simulation. (2012)

[Stefan Finsterle 2010] STEFAN FINSTERLE, Michael B. K.: A truncated Levenberg-Marquardt algorithm for the calibration of highly parameterized nonlinear models. (2010), S. 731–738

[The MathWorks 2003] THE MATHWORKS, Inc.: *Optimization Toolbox Users Guide of Matlab*, 2003

[Wetter u. Polak 2003] WETTER, Michael ; POLAK, Elijah: A convergence optimization method using pattern search algorithms with adaptative precision simulation. (2003), S. 1393–1400

[Wetter u. Wright 2003] WETTER, Michael ; WRIGHT, Jonathan: Comparison of a generalized pattern search and a genetic algorithm optimization method. (2003), S. 1400–1408

English version

E.ON Energy Research Center
EBC | Institute for Energy Efficient Buildings and Indoor Climate

Diploma Thesis

Optimization Procedures for Parameter Estimations in dynamic Simulations of Buildings

Aachen, May 2013

Luis Jarque Catalán

matriculation number: 317373

Supervisors:

Marcus Fuchs

Prof. Dr.-Ing. Dirk Müller

RWTH Aachen University

E.ON Energy Research Center | ERC

Institut for Energy Efficient Buildings and Indoor Climate | EBC

Mathieustraße 6, D-52074 Aachen

A Introduction

A.1. Motivation

Energy efficiency technologies are the key to reducing energy demand. In special, building sector displays a great potential in the reduction of heating and cooling energy.

Dynamic simulations are the properly tool for this investigations. However, it is necessary to simplify building for simulation in a suitable level to achieve requirements of convergence, stability, calculation time and effort of parametrization.

Usually, a lot of time is spent in creating the input for a simulation model, but once this is done, the user usually does not determine the parameter values that lead to optimal system performance.

This can be because there is no time left to do the tedious process of changing input values, running the simulation, interpreting the new results and guessing how to change the input for the next trial, or because the system being analysed is so complex that the user is just not capable of understanding the non linear interactions of the various parameters. However, using mathematical programming, it is possible to do automatic single- or multi-parameter approximation with search techniques that require only little effort.

A.2. Objective

When a technician create a model of a real building, he tries to represent it as specific as possible. After the model is made some parameter can be optimized in order to reduce the complexity of model. For this action, we need real measured data. Then, comparison of model simulation and measured data can be evaluated.

As studied model in this project is already built, all the parameters referring to materials are well know. However, presence of people, number of machines like computers and lights in each room have a stochastic behaviour and it is no possible to determine exactly. This three

parameters compose inner loads and they contribute decreasing heating consumption. Along the project, we cite the concept Üser modelrefers to the influence and conduct of inner loads on building (A.1).

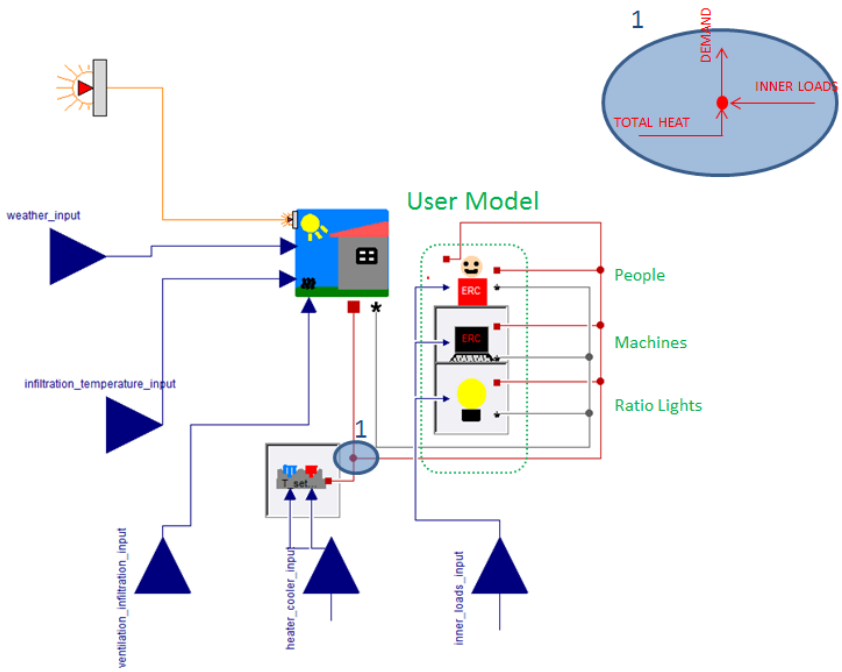


Figura A.1: User model in building

That User Models are considered deterministic because heating consumption evolution is determined by inner loads, that means, inputs influence outputs without taking into account of relations of equations. That is because there are no direct equations that relate inner loads and heating energy consumption. In such a way main task consists of tune simulation results and measured data of heat energy.

The goal is to develop an optimization program and a mathematical method with Matlab to simulate through Dymola building behaviour along one year. After that, finding with less labour time the independent variables that yield better performance of such systems. This two methods are compare with another calibration tool implemented in Dymola.

This tool provides the means for estimate parameters in the model to minimize error between simulation results and measurements through iterations. Also, for parameter optimization of

user model, we have to evaluate some criteria as calculation time, performance of parameter and stability of optimization. After this evaluations, a comparison between models reports the method to choose.

The optimization program to develop in Matlab has to adjust parameter values, simulate the model of Dymola and reduce the squared difference between simulation and measured data (B.2). The function into a script should be programmed for making this iterative process until the maximal reduction of the sum squared difference. An optimal program should be concern about calculation time and proper performance as well as the number of parameters that can be fixed with accuracy.

Mathematical method is also an program makes with Matlab, the goal is to linearise the objective function with least squares method and determine the proper direction of parameters (increase or decrease values). Its methodology is by steps, that permits us a further study of algorithm.

After develop the three methods in several cases, through a sensitivity analysis we have to guarantee the operating conditions for which a method is stable and can be used. A comparison between results of each method provides the necessary information to set the adequate method depending the studied case. Then, first study includes only one parameter that represents inner loads as a fixed heat flow *(W)*: Second study is referred to three parameter, people, machines and light. With last study should estimates number of persons hourly during one office day (11 hours - 11 parameters).

In case optimization or mathematical method was better than calibration function of Dymola, a standard procedure for user model optimization could be created. Applying the procedure to more than *200* building, that are currently being examined in E.ON Energy Research Center, could save time on parameter estimation on buildings.

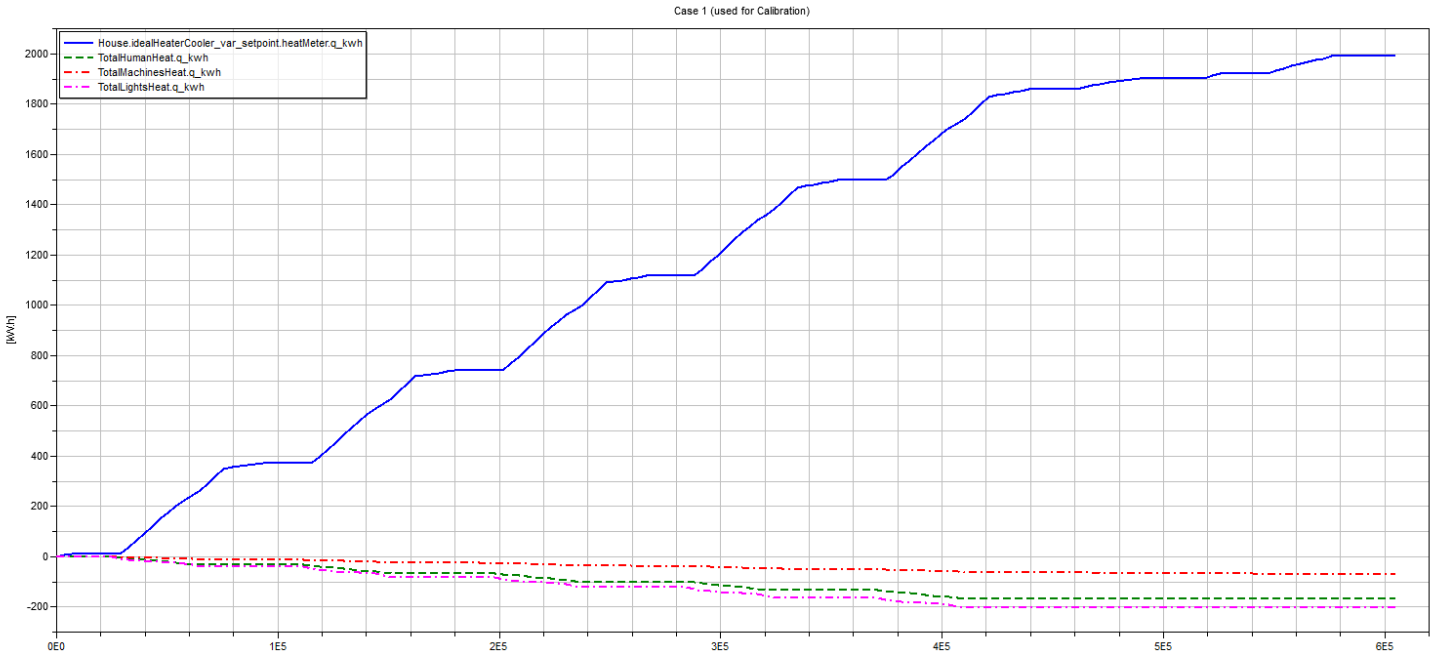


Figura A.2: Energy contribution of inner loads

A.3. Structure of the thesis

The thesis is written in the same order it was developed. First,an explanation of the most important aspect for a model optimization. *Chapter 2* explains optimization methods, where we can see an introduction to general optimization and its importance in engineering. After that, an explanation of the building case of studio gives important concepts of concrete building. Once the building is well know three methods are explained. Furthermore, each method is independent of other. *Chapter 3* develops methods explained in the previous chapter. Different studied cases are necessary to determine algorithm behaviour. Then, one, three and eleven parameters are calibrated. *Chapter 4* shows a sensibility analysis and the influence of parameters on methods. Knowing results of simulation and a comparison of methods allows to evaluate the implemented programs. *Chapter 5* is the conclusion and considerations of the project. *Annex A* includes the code of the optimization method with Matlab and *Annex B* code of the mathematical function of Matlab with the edition of a file.

B Optimization methods

B.1. General optimization view

The use of system simulation for analysing complex engineering problems involves many independent variables, and can only be optimized by means of numerical optimization. Parametric studies achieve good performance of such systems, even though such studies typically yield only partial improvement while requiring high labour time. [Bernard P. Zeigler, 2000] In such parametric studies, one usually fixes all but one variable and tries to optimize a cost function with respect to the non-fixed variable. The procedure is repeated iteratively by varying another variable. However, every time a variable is varied, all other variables typically become non-optimal and hence need also to be adjusted. It is clear that such a manual procedure is very time-consuming and often impractical for more than two or three independent variables.

Building and HVAC system design can significantly improve if numerical optimization is used. However, if a cost function that is smooth in the design parameter is evaluated by a building energy simulation program, it can be replaced with a numerical approximation that is discontinuous in the design parameter.

Moreover, many building simulation programs do not allow obtaining an error bound for the numerical approximations to the cost function. Thus, if a cost function is evaluated by such a program, optimization algorithms that depend on smoothness of the cost function can fail far from a minimum. [Wetter u. Polak, 2003]

In the most general form, the optimization problems can be stated as follows: Let X be a user-specified constraint set, and let $f: x \rightarrow \Re$ be a user-defined cost function that is bounded from below. The constraint set X consists of all possible design options, and the cost function $f(\bullet)$ measures the system performance. The objective consists to find a solution to the problem:

$$\min_{x \rightarrow X} f(x) \tag{B.1}$$

This problem is solved by iterative methods, which construct infinite sequences, of progressively better approximations to a solution, i.e. a point that satisfies an optimality condition.

Consequently, optimization algorithms can fail, possibly far from a solution, if $f(\bullet)$ is not differentiable in the continuous independent variables.

The design parameters in the studied optimization problem are discrete variables since they correspond to internal loads due to people, machines and lights. In addition, the problem has not inequality constraints on the dependent variables. [Wetter u. Wright, 2003]

B.2. Reference building

A description of the building is necessary for understanding the problem in hand and how inner loads affects on building. The reference building is a research institute built in recent years according to the German EnEV-standard. The specific uses of building is as a typical office. [M. Lauster, 2012]

The building consists of a two floors of offices. The floor plans of building show six distinct zones (fig: B.3). It has a total floor area of $1300m^2$ and a total volume of $3800m^3$. Building properties are well known. They are included in Excel sheets. Also, they were introduced in the model we use during the project.

Data related with energy measures on building are available. They has been supplied by the project partner of reference building and collected in an Excel sheet which contains time, electric power, electric energy, heating power and heating energy.

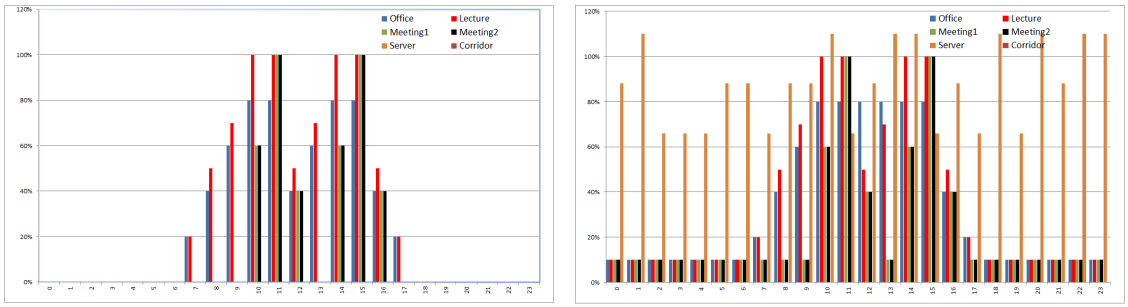
Measuring file was adapted to the needs of the project. It was an excel sheet with more than *40000* measures taking each 15 minutes. On this sheet there was gaps without measures solved with interpolation for accumulated heating energy. The adapted sheet was filtered hourly from 01.01.2011 until 03.01.2012. The reason of taking 2 days in 2012 is because we use for calibration dates at Monday 03.01.2011 until 03.01.2012 since we adjusted weather data and user schedules in the model on Dymola starting both on Monday 01.01.2011 at 00:00 a.m.

User model has a influence in inner loads since emit heat into the building. Then, User Profile shows the behaviour of people during a day measured in intervals of one hour. The daily behaviour is repeated until Friday and it becomes null at weekend because nobody works and machines and lights are turned off (B.2(a), B.2(b), B.2(c)).

Measured data used for calibration correspond to heat energy consumption, column 5 "Heat_Counter"(fig: B.1).

	A	B	C	D	E
1	Zeit (sec)	Elec_Power (W)	Elec_Counter (Wh)	Heat_Power_calculate (W)	Heat_Counter (Wh)
2	63504000	11184	39283881	20000	16539999
3	63507600	11061	39292791	0	16539999
4	63511200	10896	39301631	0	16539999
5	63514800	10853	39310441	0	16539999
6	63518400	10896	39319191	0	16539999
7	63522000	10926	39327951	0	16539999
8758	95025600	13238	122999682	0	38379998
8759	95029200	11868	123010372	10000	38389998
8760	95032800	9769	123019632	10000	38389998
8761	95036400	9764	123027482	10000	38399998
8762	95040000	9703	123035322	0	38409998

Figura B.1: Measurement data collected in hours

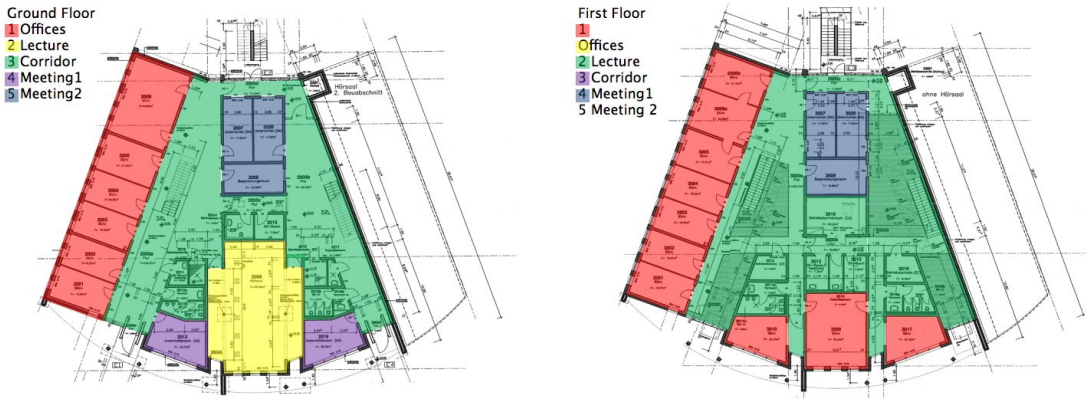


(a) People

(b) Machines

(c) Light

Figura B.2: Inner loads corresponding to complex model of six zones



(a) Ground floor (b) First floor

Figura B.3: Zones of building

Next table (tab: B.1) shows estimated parameters for number of persons, heat flow provided by machines and light ratio. Ratio values are based on norms DIN 18599 and SIA 2024 and they were used as fixed parameters for the model design. Knowing the real consumption of the building, these parameters have to be calibrated in sense to reduce the difference between measured heat energy consumption and the consumption of the model simulation.

Cuadro B.1: Building zones

	Nr People (m^2/p)	**Machines** (W/m^2)	**Lights**(W/m^2)	**Area** (m^2)	**Persons**	**Machines**
Offices	14.0	7.0	15.9	371	26.5	26
Corridor	0.0	0.0	3.0	668.7	0.1	1
Meeting1	3.0	7.0	15.9	41	13.65	3
Meeting2	3.0	6.5	15.9	107.5	35.85	7
Lecture	3.2	4.7	13.0	105.8	33	5
Server	30.0	660.0	7.1	7.58	0.25	50

B.2.1. Methods chosen for parameter estimation

User model estimated in table B.1 is used as initial values of our parameter before estimation. This leads to uncertainties in the simulation, nevertheless the comparison is a benchmark of the simulation's ability to reflect a real building with all its influences.

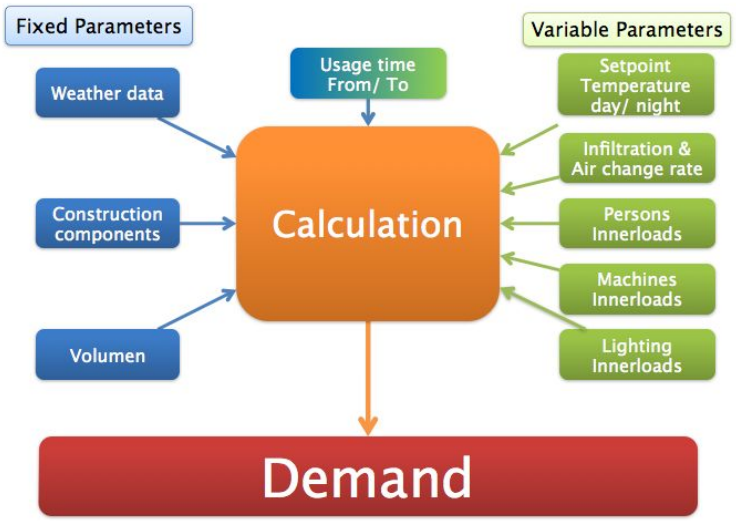


Figura B.4: Interaction between parameters and demand

Models studied contains a huge number of parameter whose behaviour can not be predicted. Numerical solution methods in Modelica allows the reuse of simulation models for optimization. Also nonlinear dynamic optimization problems can be treated efficiently as discrete-time optimal problem and solved numerically by applying largescale nonlinear optimization methods.

The goal is to analyse different methods and compare their performance in minimizing the annual difference of heating energy consumption on reference building. The studied model has all parameters fixed and the objective function consists on a scalar calculated as the sum of squared difference. Furthermore, the methods have to consider energy consumption hourly without differential equations. The method chosen to study are:

1. Dymola calibration function
   - ▷ Calibration function already implemented in a library of Dymola.
   - ▷ Fast method for preparation of calibration.
   - ▷ Modelica for model creation and Dymola for simulations.
2. Matlab optimization toolbox
   - ▷ Pattern search methods represents a subclass of direct search algorithms, in which the minimizer of a continuous function is sought without the use of derivatives. [Audet u. J. E. Dennis, 2000]

- ▷ Numerically robust in tackling with non-smoth criteria. [H. Elmqvist, 2005]
- ▷ Optimization function implements in Matlab.
- ▷ Adjust of parameter bounds.

3. Mathematical method in Matlab (Levenberg-Marquardt method)
   - ▷ Minimize objective functions non differentiable.
   - ▷ Gradient approximation by finite difference.
   - ▷ Stable method for high level problems.
   - ▷ Convert a non linear problem to lineal.

B.3. Model Calibration with Dymola

Model calibration consists in a parameter estimation. In this process measured data from a real device is used to tune parameters such that the simulation results are in good agreement with the measured data. The parameters that we tune are often referred to as tuners. Dymola varies the tuners and simulates when it searches for satisfactory solutions. Mathematically, the tuning procedure is an optimization procedure to minimize the error between the simulation results and the measurements via iterations.

B.3.1. Calibration setting

Before tuning parameters from measurements, we have to study which parameters can be estimated from the available measurements. Changing a parameter to be estimated must of course influence the output. However, two or several parameters may influence the result in a similar way (co-variance), such that it is not possible to estimate them individually. When a set of parameters have been tuned, the model is validated and the tuned parameters against other measured data to check that there is a good agreement between the simulation result and the new measurements. [AB, 2012]

B.3.2. Tune the parameters

Previous to calibration of parameters, we have to validate the model. Validation is utilized to determine that a model is an accurate representation of the real system. Validation is usually

achieved through the calibration of the model, an iterative process of comparing the model to actual system behavior and using the discrepancies between the two, and the insights gained, to improve the model. This process is repeated until model accuracy is judged to be acceptable.

By way of example for tuning parameters, a simplified model with three parameter (C.1) has been chosen for validation and calibration of model. Let us first check how the model with nominal parameters compares with measured data.Validation is set up very similar to calibration. A basic difference is that no tunable parameters need to be specified for the validation.

First, we specify the model to calibrate. Then, the model is translated in order to gather information needed to build browsers and selectors to support the remaining setting up of the calibration task (fig: B.5).

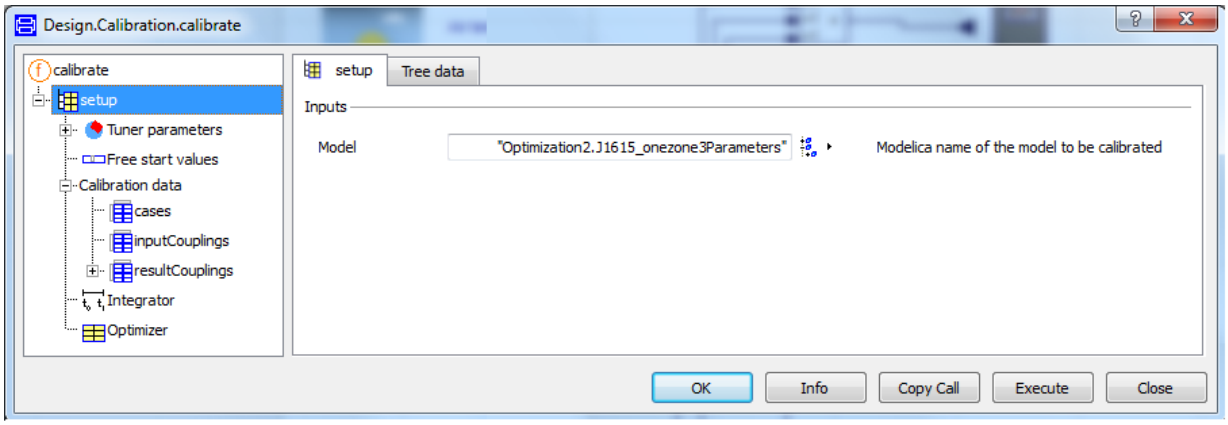


Figura B.5: Model specification

The next task is to specify the measurements and how they are stored (fig: B.6). This measurement file has to be the same as we will use with for algoriths (B.4 and B.5).

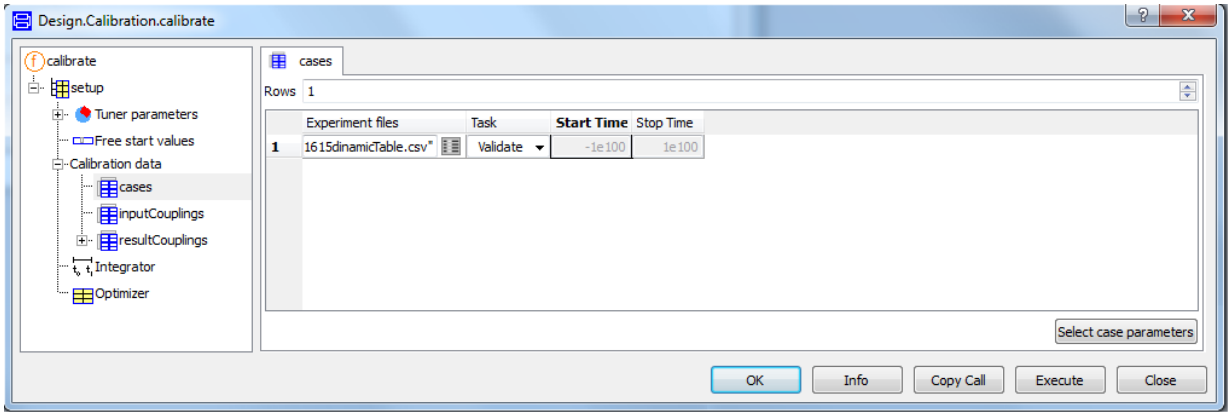


Figura B.6: Case of studio

If measured data are given in some unit different than that used in the model (kWh and kW), the scale column allows scaling of the measurements: $variable = data * scale + offset$. In case the deviations of several variables shall be used to specify the criterion, the weight column allows the user to give them different weights (fig: B.7).

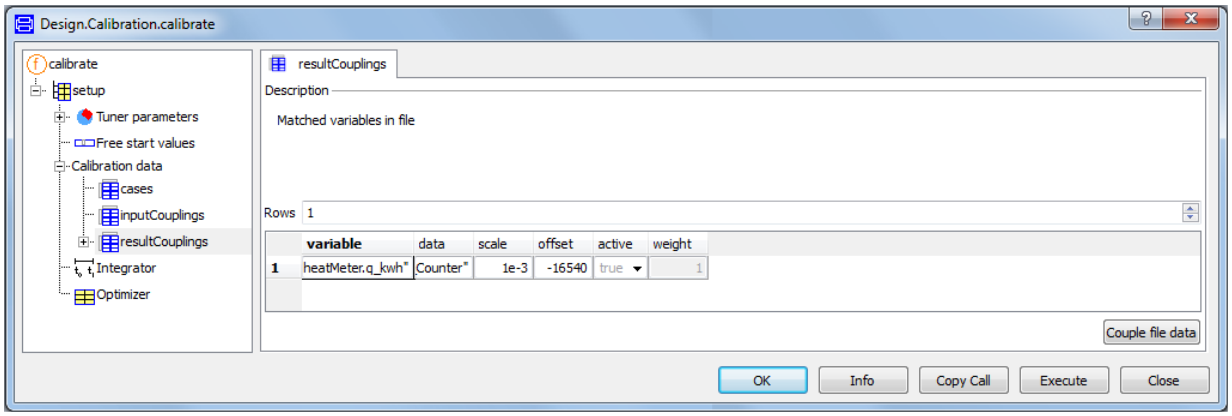


Figura B.7: Details of measurements

Integrator element allows the specification of a global simulation interval, this is important on load file when time is different to 0 as this case. Here it is studied the building during one year (fig: B.8).

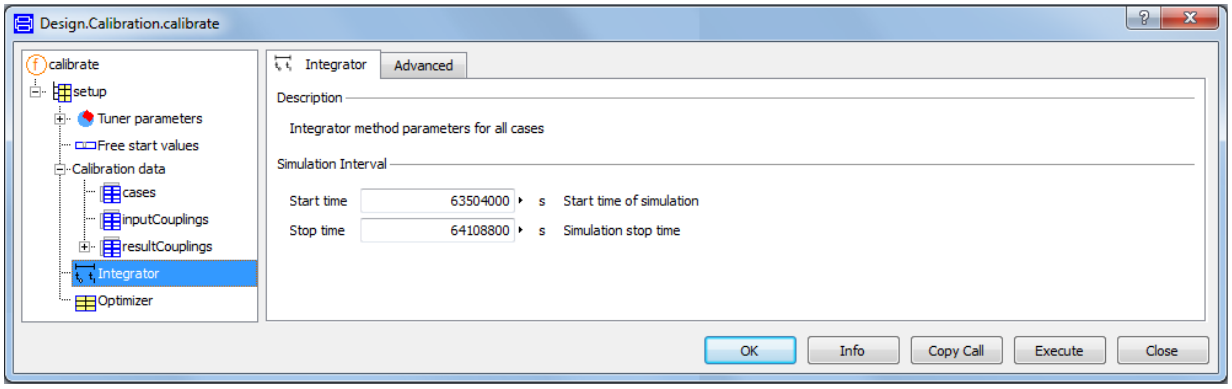


Figura B.8: Time interval for simulation

After validation (fig: B.9), an error of criterion of $4{,}7e6$ has been detected and total energy difference for one week respect reference is *21.5 %*. Now, we change the task in **Cases >Calibrate** and **Select parameters** in **Tuner parameters** (fig: B.10).

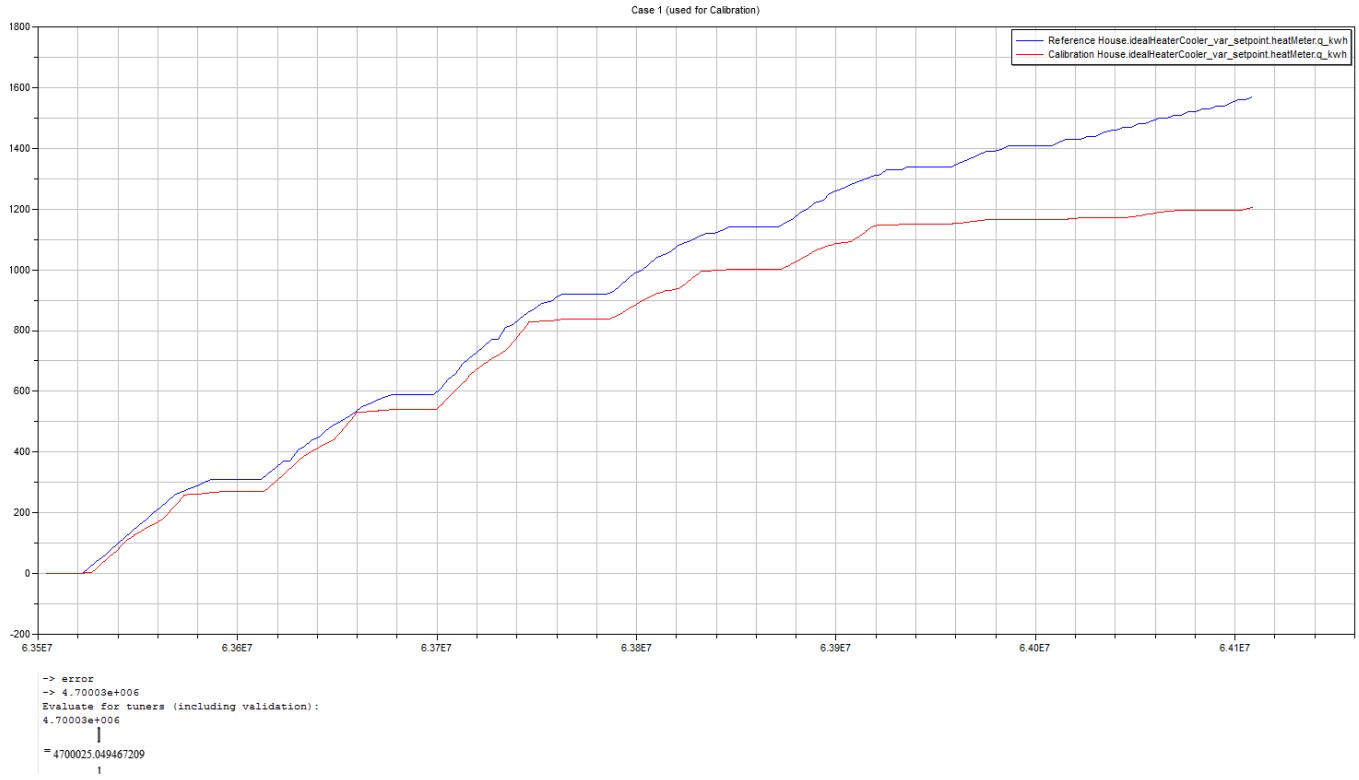


Figura B.9: Results of Validation for a week

Parameter selected for the calibration are the three inner loads This three parameters corres-

pond to estimated with norms DIN 18599 and SIA 2024 (tab: B.1) for this reason initial values are (110, 41, 15.9). Also, lower and upper bounds are from 0 until 250 for the two parameters people and machines, but bounds for ratio of lights is from 0 until 25. Bounds should permit to find optimal solution.

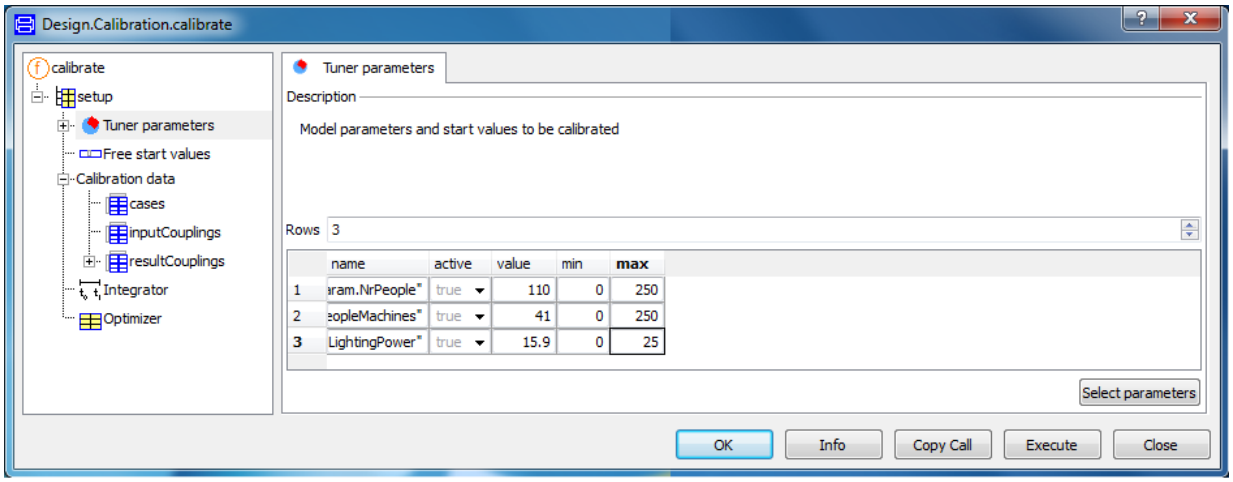
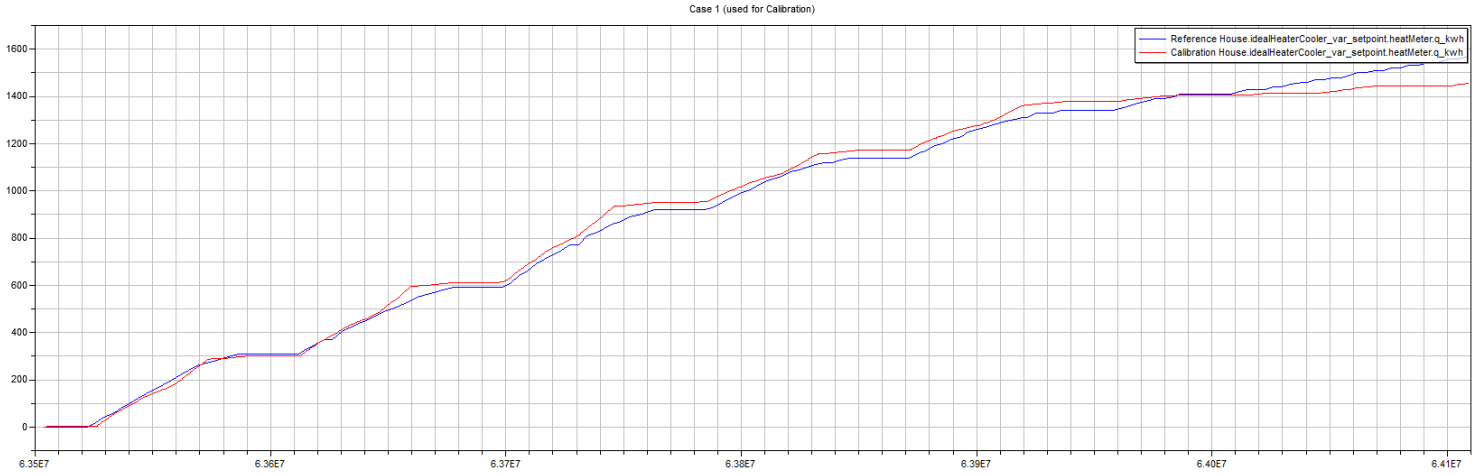


Figura B.10: Tuner parameters

After iterations Dymola returns a screen with results. Number of iterations (105), best result of parameter (180, 52.7, 0.2)and result of criterion (2,44e5). Which entails a total energy difference of *7%* respected measure.(fig: B.11). Mainly Tuning affects the results decreasing the error by 94,8 % as we can see comparing figures B.9 and B.11.




```
House.zoneParam.NrPeople House.zoneParam.NrPeopleMachines House.zoneParam.LightingPower -> error
110                      41                               15.9                          -> 4.70003e+006
180                      20                               0                             -> 1.47845e+006
180                      20                               0                             -> 1.47845e+006
180                      58.4134                          0                             -> 258856
180                      52.8115                          0.102942                      -> 245642
180                      52.7421                          0.196558                      -> 244644
180                      52.7314                          0.21853                       -> 244579
180                      52.713                           0.223278                      -> 244574
180                      52.7101                          0.2274                        -> 244571
180                      52.7093                          0.229589                      -> 244570
180                      52.7091                          0.229839                      -> 244570
180                      52.7089                          0.230073                      -> 244570
180                      52.7088                          0.229993                      -> 244570
Evaluate for tuners (including validation):
244570
  {180.0, 52.708818344186 74, 0.22999319434273063}
=
          |
  244569.89189935615
        105
```

Figura B.11: Results of calibration for a week

B.4. Matlab Optimization toolbox

Optimization Toolbox provides widely used algorithms for standard and large-scale optimization. These algorithms solve constrained and unconstrained continuous and discrete problems. The toolbox includes functions for linear programming, quadratic programming, binary integer programming, nonlinear optimization, nonlinear least squares, systems of nonlinear equations, and multiobjective optimization [The MathWorks, 2003]. They can be used to find optimal solutions, perform tradeoff analyses, balance multiple design alternatives, and incorporate optimization methods into algorithms and models.

Pattern search method is used with optimization toolbox (fig: B.12).The method is based on a search of the minimum value for objective function. Flow chart shows the method used by Pattern Search. It constructs a mesh which is then explored. Poll occurs when the search step was unable to obtain a point on the current mesh that decreased the incumbent value. If no decrease in cost is obtained on mesh points around the current iterate, then the distance between the mesh points is reduced (refine mesh), and the process is repeated. [Audet u.

J. E. Dennis, 2000]

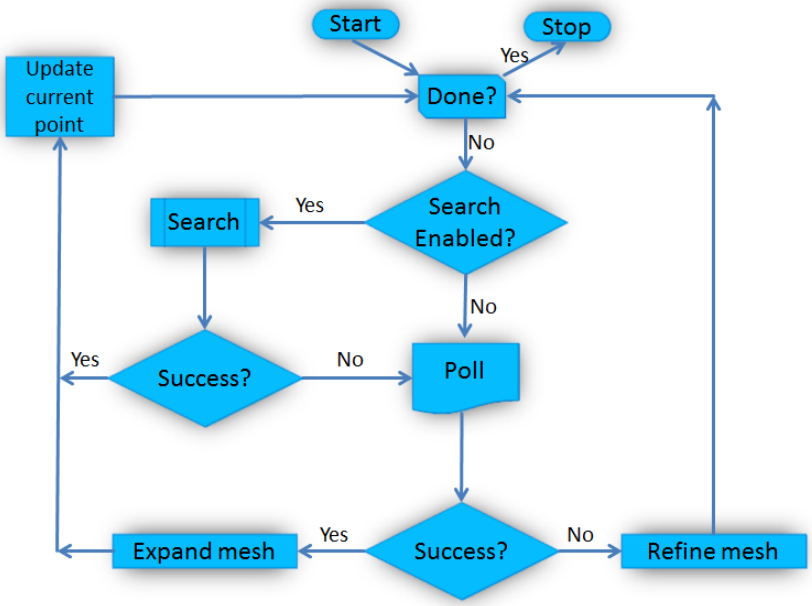


Figura B.12: Basis of pattern search method

B.4.1. Dymola to Matlab connection

Connection between both programs is an important part of the project. The script "dymolaMçan execute every command that can be used in Dymola. Thus, the achieved actions for model calibration are:

1. Translate model into Dymola.
2. Set estimated parameters with Pattern Search.
3. Simulate the model.

Also, dymload and dymget scripts load a result file from Dymola to Matlab and get the values of the searched parameter respectively.

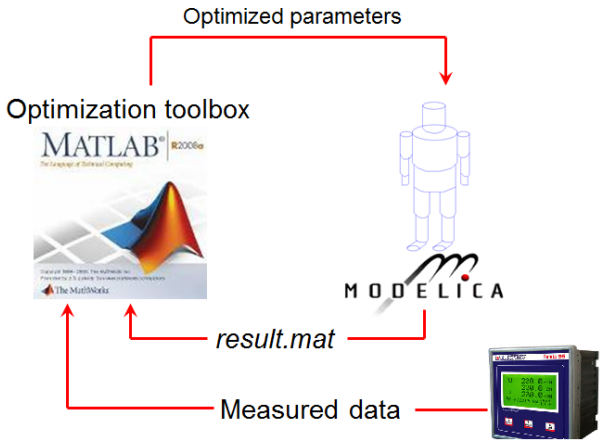


Figura B.13: Dymola - Matlab conecction

B.4.2. Algorithm and solver

The optimization algorithm is developed to be more efficiency with regard time effort, accuracy and stability of results. However, this tool needs great effort to be implemented (See Matlab code in 8). Also, the only way to get efficiency on program is to set optimization toolbox options in Matlab with simulation tests (fig: B.14).

	FixedHeatFlow (W)	error^2	Iterarions	function evaluations	time (sec)
Without options	16781	1.40126E+10	36	61	444
MeshAccelerator ,on, ; ScaleMesh ,off,	16781	1.40126E+10	57	94	716
Tol and ScaleMesh ,on,	16769	1.40127E+10	17	25	181
Cache and CacheTol (1e-10)	16769	1.40127E+10	17	19	136.8
Cache and CacheTol (eps)	16769	1.40127E+10	17	19	138.9
Plot	16769	1.40127E+10	12	15	117
ScaleMesh ,off,	slow				
MeshAccelerator ,off,	16769	1.40127E+10	14	19	140
TolBind , default,	16769	1.40127E+10	14	19	142
GSSPositiveBasisNp1	16769	1.40127E+10	12	15	113.8

Figura B.14: Options evaluation for algorithm in a simple one week study

Furthermore, all specific cases studied in project can be adapted in an easy way with next steps:

1. Script modification
   - ▷ Give your measured data as a .mat file (for the project was not necessary to change).

- ▷ Set initial condition vector ($x0$) of parameters.
- ▷ Set lower and upper bounds (lb, ub) as well as constrains in such case (A, b, Aeq, beq).
- ▷ Adjust tolerance of objective function results ($Tolmesh: 1e-06$).

2. Function modification
   - ▷ Give the parameter names to change their values in Dymola .
   - ▷ Change simulation characteristics (time, number of intervals, tolerance or solver).
   - ▷ Change criterion if it was necessary ($|TotalHeat - ydata|^2$).

In the optimization function (fig: B.15), at each iteration either a comparison shows improvement (simple decrease) in the function value and the improving point becomes the new iterate or no decrease in value is found at any point in the pattern (fig: B.16).

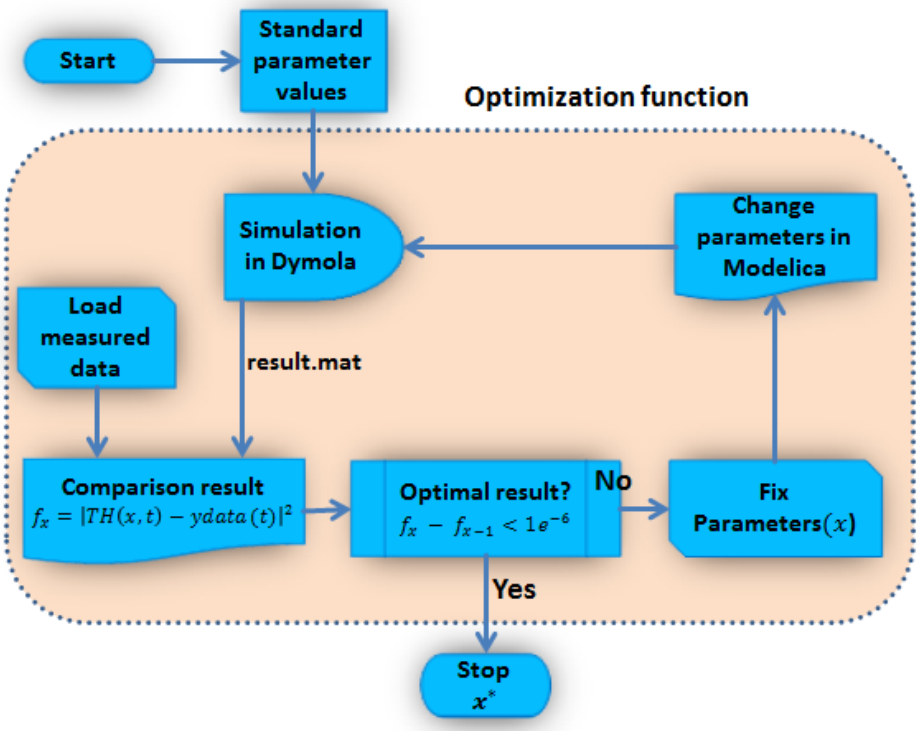


Figura B.15: Algorithm running with Optimization toolbox

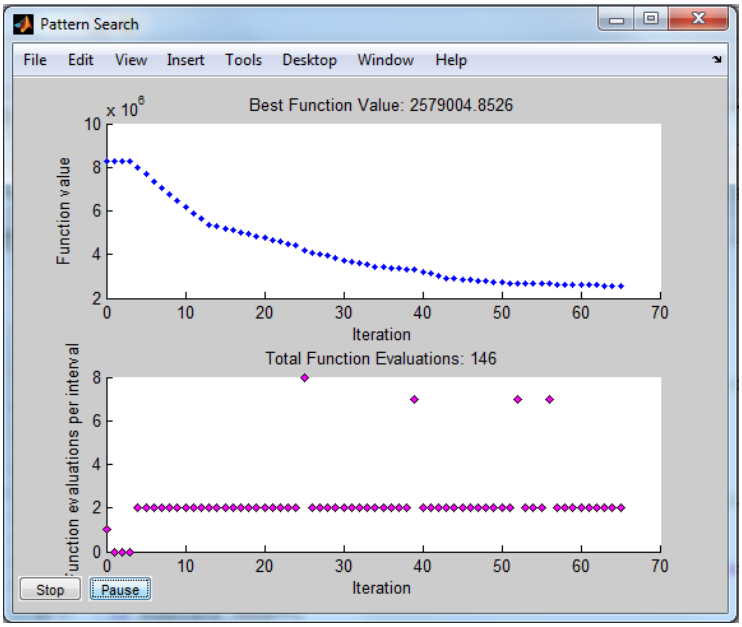


Figura B.16: Function value and evaluation with pattern search

B.4.3. Algorithm precision

Degree precision for parameters is verified with an inverse process. Taking total heating energy after simulation as measured data input and default building parameters of the model, the result of parameter after simulation for people user profile is compared with the default. We have studied two cases, where there are 11 parameters to estimate. These parameters are the presence of people on building each hour from 7 a.m. until 18 p.m..For case 1, take into account lower and upper bounds are *0%* and *100%* of occupation, the accuracy of results is 96,8% for one week,. In case 2, user profile has a tolerance respect to the standard ones of ±10%, this case is more realistic, because a user profile should be well know. Then, parameters calculated with the used method has a good approximation to standard (99,1%).

Cuadro B.2: Percentage of people hourly on one zone building

Hour	7h	8h	9h	10h	11h	12h	13h	14h	15h	16h	17h
Model % ocup	0.2	0.4	0.6	0.8	0.8	0.4	0.6	0.8	0.8	0.4	0.2
Case 1 % ocup	0.93	0.76	0.49	0.49	0.51	0.49	0.49	0.49	0.49	0.49	0.49
Case2 % ocup	0.18	0.35	0.55	0.70	0.70	0.35	0.55	0.70	0.70	0.35	0.19

B.5. Mathematical optimization method

Previous methods use mathematical processes to obtain solution. However, we provide the output in order to estimate parameters in sense of improve effort, time, robustness and in general to obtain the most efficient method for parameter estimation.

In this part of the chapter a direct mathematical application is developed. Thus, reasoning of how the algorithm estimates the parameter have to be explained.

The object oriented model is typically translated to a mathematical system of differential and algebraic equations prior to its treatment with numerical solvers. However, the studied case can not be written as differentiable equations, such its parameters are not directly related. That is why a method for minimizing the least squares cost is needed. Then, we start with the *Gauss-Newton method*. Least squares (LS) is the problem of finding a vector x that is a local minimizer to a function that is a sum of squares, subject to some constraints:

$$\min_x ||F(x)||_2^2 = \min_x \sum_i F_i^2(x) \tag{B.2}$$

such that $Ax \leq b, \quad A_{eq}x = b_{eq}, \quad lb \leq x \leq ub.$

In the least-squares problem the function $f(x)$ is minimized that is the sum of squares. They are also prevalent in control where you want the output, $y(x,t)$, to follow some continuous model trajectory provides by total heat consumption, $\varphi(t)$, for vector x and scalar t. This problem can be expressed as:

$$\min_{x\in\Re^n} \int_{t_1}^{t_2} (y(x,t) - \varphi(t))^2 dt \tag{B.3}$$

where $y(x,t)$ and $\varphi(t)$are scalar functions.

When the integral is discretized using a suitable quadrature formula, the above can be formulated as a least-squares problem:

$$\min_{x\in\Re^n} f(x) = \sum_{i=1}^{m} (\bar{y}(x,t_i) - \bar{\varphi}(t_i))^2, \tag{B.4}$$

where $\bar{y}$ and $\bar{\varphi}$ include the weights of the quadrature scheme. Then, in this problem the vector

$F(x)$ is:

$$F(x) = \begin{pmatrix} \bar{y}(x,t_1) - \bar{\varphi}(t_1) \\ \bar{y}(x,t_2) - \bar{\varphi}(t_2) \\ \cdots \\ \bar{y}(x,t) - \bar{\varphi}(t_1) \end{pmatrix}$$

In problems of this kind, the residual $F(x)$ is likely to be small at the optimun since it is general practice to set realistically achievable target trajectories. Although the function in Least Squares can be minimized using a general unconstrained minimization technique, certain characteristics of the problem can often be exploited to improve the iterative efficiency of the solution procedure. The gradient and Hessian matrix of LS (eq. B.3) have a special structure.

B.5.1. Gauss-Newton method (Levenberg-Marquardt)

In the Gauss-Newton method, a search direction, dk , is obtained at each major iteration, k, that is a solution of the linear least-squares problem.

$$\min_{x \in \Re^n} ||J(x_k)d_k - F(x_k)||_2^2 \tag{B.5}$$

The direction derived from this method is equivalent to the Newton direction. The search direction can be used as part of a line search strategy to ensure that at each iteration the function $f(x)$ decreases.

Although the LM algorithm is an iterative solver for parameter estimation for function optimization, that performs calculations to improve the estimate at each step, the iterations continue until an adequate bound is reached.

LM still operates in a deterministic fashion, where identical results are produced if the data and input parameters are not changed. This deterministic property will allow us to test and verify the distributed implementation since we expect the same results for the same configuration. [Stefan Finsterle, 2010]

The LM algorithm is based on the Gauss-Newton Method, where there are similar concepts in the frameworks, except the LM approach incorporates a dampening parameter that increases stability and therefore helps with convergence. Furthermore, LM Algorithm is also similar to other numerical solvers such as gradient descent or the conjugate gradient method, which are also iterative solvers that improve the estimate at each step by selecting the parameters that are in the direction of the basis opposite to the gradient, which is in the direction that will

reach the optimum point. However, these methods are data dependent, and require symmetric positive definite systems for convergence.

The Levenberg Marquardt algorithm performs a curve fitting on a given dataset, by finding the optimum function parameters x^* based on a user specified model, such that the final parameters can characterize the target function by minimizing the residual, which is the same as solving the least squares problem, where we want to minimize the sum of squares between the target values $\varphi(t)$ and the output of the user model $y(x,t)$.

The algorithm follows iterative steps. Firstly, we simulate with standard parameter values x and the increment in x for finite difference aproximation dx is fixed to 1 %. Lambda is a value used for determining step size. Increasing lambda step size decreases but time of calculation rises up. Such gradient needs so many simulations as parameters exist, the method needs long time to determine the step size and proper direction but the algorithm is efficient for a large number of parameter. (fig: B.17).

$$\frac{\partial f}{\partial x_1} = \frac{F(x+\Delta x_1) - F(x)}{\Delta x_1} \tag{B.6}$$

$$\nabla F = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

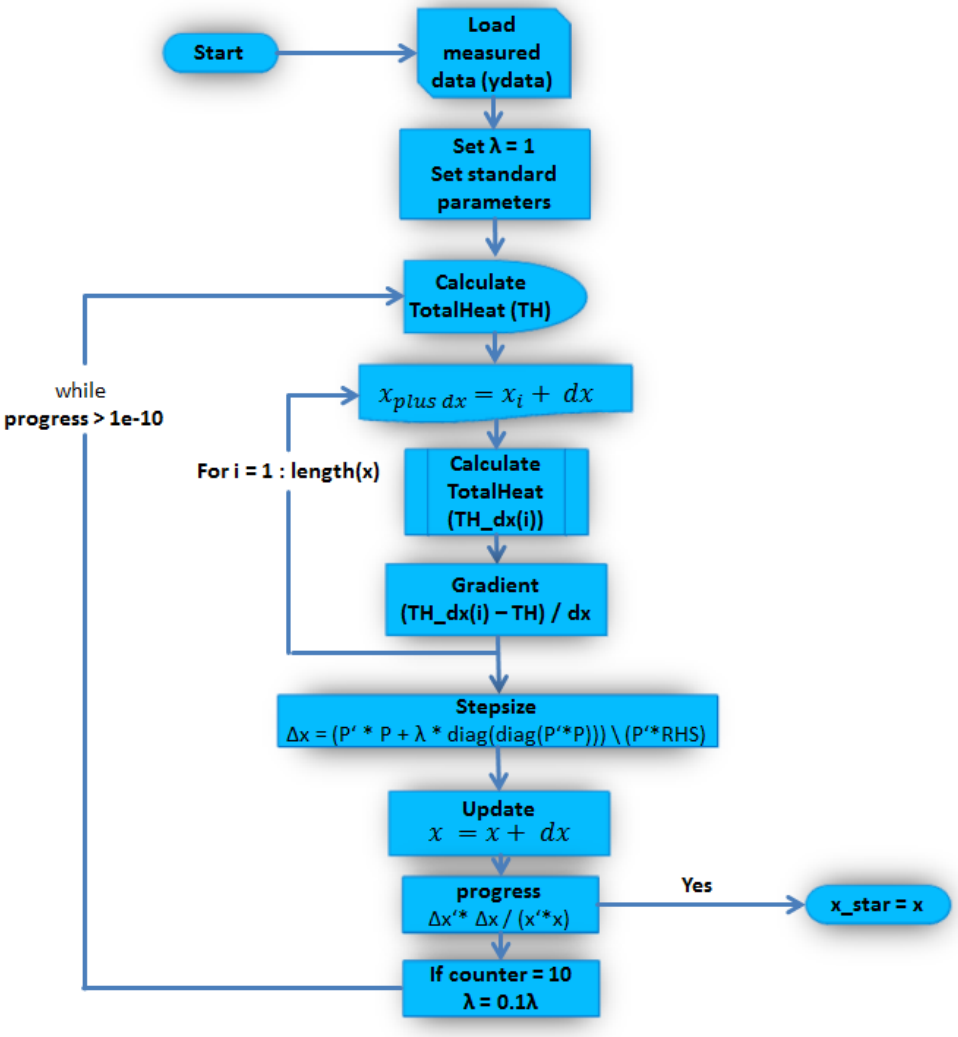


Figura B.17: Mathematical method as function in Matlab

C Development of optimizations

This chapter shows the application of the method explain in (chapter: B). Several cases are studied to determine the behaviour of the three methods depending of number of studied parameters. Also, it is important to get a model as simply as possible able to represent a real building in a stable manner.

C.1. Simplified building model: J1615 as one thermal zone

The model has been simplified for used methods from six zones to one zone in order to investigate in an easier way but less precise than the complex model. Such model has the same characteristics as the model with six zones (fig: C.1). Nevertheless, user profile corresponding to machines is reduced because of the zone where server is located. The server contributes with inner loads about *5kW* of power, but only in one of the six zones, that means, heat power because of machines is reduced from *92* to *41* equivalent persons. In the equivalent power of machines expressed as persons, one person amounts to *100 Watts*.

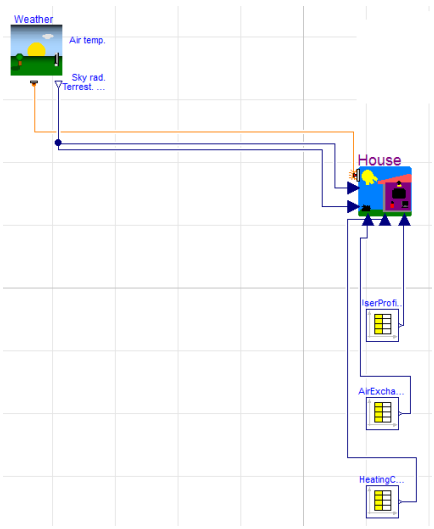


Figura C.1: J1615 one zone

C.2. Study of fixed parameters in one thermal zone model

On the basis of simplified model (C.1), next cases will provide a first study to the methods with such the error of calibration is bigger than in complex cases. This is because number of parameters is lower than six zones model.

C.2.1. One parameter: fixed heat flow

The most simplified model correspond to substitute all the user model for a heat power load (fig: C.2). One zone model has 827 parameters versus 4581 parameters that reference model with six zones has.

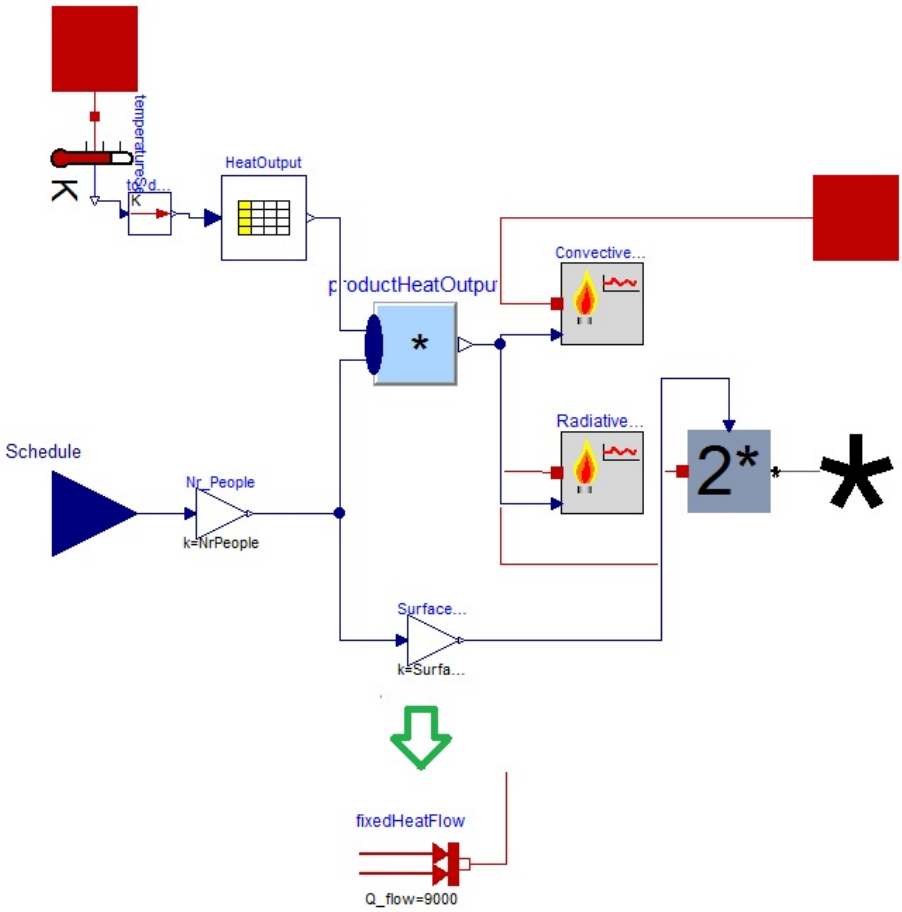


Figura C.2: J1615 people heat contribution

After calibration at easy model, fixed heat flow and error calculated by optimization toolbox and mathematical method have a good agreement between them. Also, mathematical method is the fastest and needs less iterations to converge the solution (tab C.1).

Cuadro C.1: Results of one zone calibration

	Time for simul.	**03.01-10.01**	**31.01-07.02**	**28.02-07.03**	**05.12-12.12**	**1 year**
FixedFlow (W)	Dymola	6299	9422	6169	6469	4346
	Toolbox Matlab	11720	7624	5576	5240	6392
	Math method	11698	7650	5553	5244	6396
$Error^2$	Dymola	5.79e5	9.29e5	5.92e5	1.45e6	6.69e9
	Toolbox Matlab	4.62e5	5.91e5	4.62e5	1.09e6	2.18e10
	Math method	4.69e5	5.91e5	4.62e5	1.09e6	2.18e10
Iterations	Dymola	25	29	17	13	25
	Toolbox Matlab	13	14	11	11	15
	Math method	5	8	4	4	5
Time (s)	Dymola	186	188	136	100	8312
	Toolbox Matlab	76	63	65	81	560
	Math method	16	30	18	14	336

C.2.2. Three parameters: people, machines and light ratio

In next case, the influence of three parameter on heating consumption is studied. In fact, these parameters are multiply by the percentage of building presence at each hour giving in UserProfileOffice.txt (fig: C.4). The profile is repeated every day from Monday until Friday and null at weekend.

Normally, more presence of people on building is related with more machines and light contribution. Nevertheless, we consider heating demand as fixed input and then increasing number of person reduces power due to machines and or lights (tab: C.2).

In spite of the lower error of Dymola calibration toolbox, it requires enough time and iterations. Dymola results are interesant because all three parameters are lower than obtained with Matlab, this is illogical for one reason, criterion for all method is the sum of squared difference. Now, mathematical method is also as accurate as optimization method. However, it is necessary more time for parameter estimation than before.

The next graphic is the result of apply the three estimate parameters with the toolbox of Matlab along a weeek (fig: C.3). First graphic represents a instantaneous heating power, whi-

Cuadro C.2: Calibration of one zone with three parameters for one year

	Dymola	Toolbox Matlab	Math method
Nr People	104	113	109
Machines	18	33	40
Light Ratio (W/m^2)	8.0	14.2	15.8
$Error^2$ (kWh)	6.57e9	2.07e10	2.07e10
Iterations	31	14	5
Time (s)	6520	1498	776

le second one represents dynamic number of person and machines estimate. Last graphic shows the power generates by people, machines and lights.

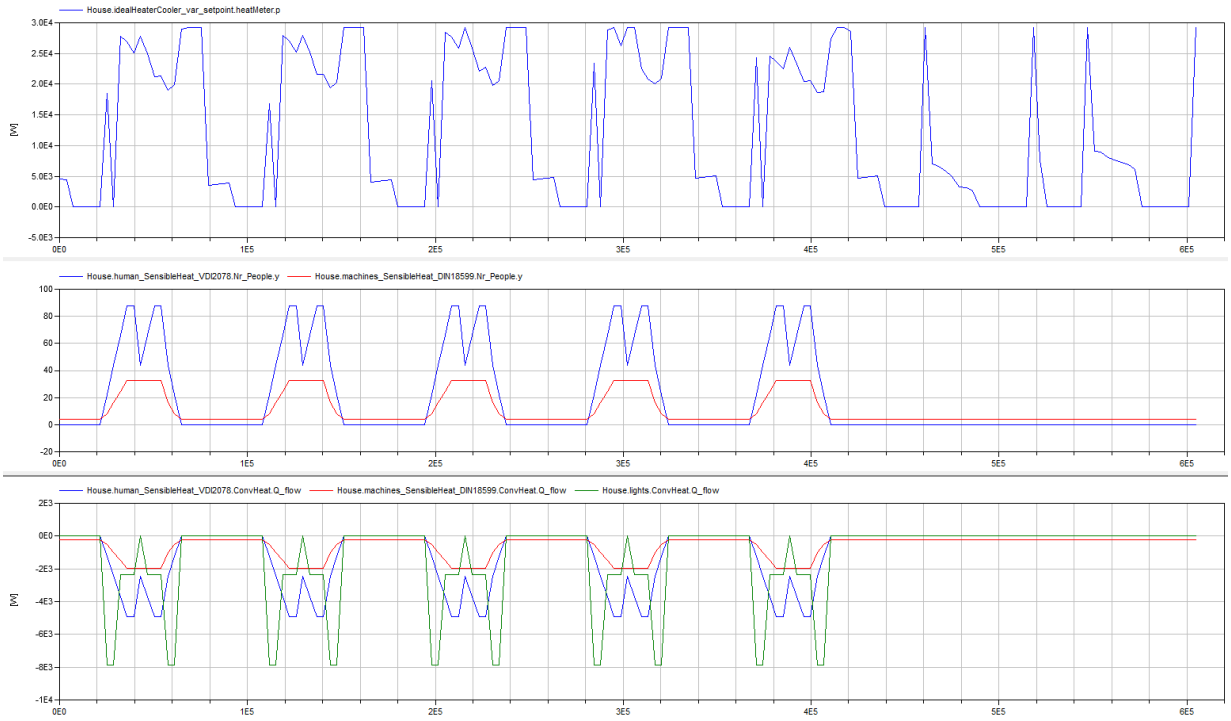


Figura C.3: Dynamic results of heating power, parameter values and inner loads heat power

C.2.3. Eleven parameters: user profile people

Office building is opened between 07 a.m. until 6 p.m. and occupation along the day change constantly, that is the reason why hourly presence of persons could indicate a significant decrease in objective function. This kind of parameters can be modified by both developed

programs of Matlab. Nevertheless, calibration function of Dymola is not allow to estimate them (fig: C.4).

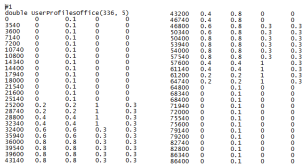


Figura C.4: User profile for one zone

Optimization toolbox of Matlab has an expensive waste of time calibrating all parameter. However, mathematical method fit elf parameters with a similar error but much faster.

Difference between setting of parameters depending of used method is for bound estimation. In case of toolbox Matlab, the function of optimization allows adjustment by upper and lower bounds. Nevertheless it is not the same for mathematical method where the step size of parameters depend on an equation that determine step and direction without bounds.

Cuadro C.3: Calibration for eleven parameters

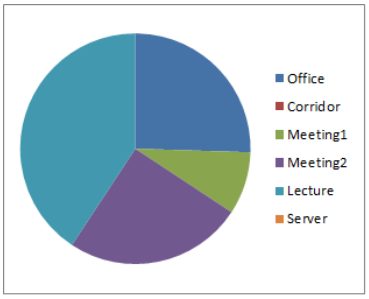
Hour	7h	8h	9h	10h	11h	12h	13h	14h	15h	16h	17h
Default % ocup	0.2	0.4	0.6	0.8	0.8	0.4	0.6	0.8	0.8	0.4	0.2
Toolbox Matlab	0.278	0.398	0.590	0.780	0.780	0.390	0.59	0.827	0.782	0.398	0.205
math method	0.421	0.426	0.410	0.450	0.435	0.418	0.427	0.401	0.423	0.460	0.428

	$Error^2$ (kWh)	Iterations	Time (sec)
Toolbox Matlab	2.06e10	65	13750
Math method	2.07e10	6	2821

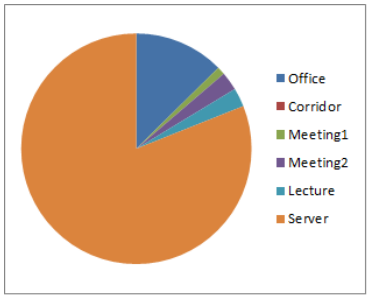
C.3. Building model: J1615 as six thermal zones

In this model are diferentiated six zones. Each zone behaves in many different ways. However, total heating energy is calculate as addition of individual demands. For creation of this model all contribution of inner loads were held (fig: C.5).

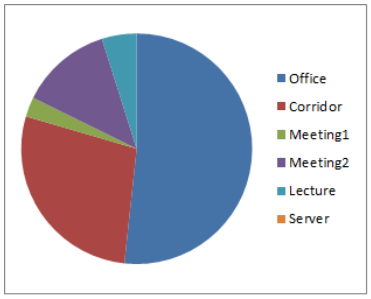
Complex model implies an increase in number of parameters making up the model; exactly to *4581* parameters. That means, a model with almost five times more equations due it has five rooms mores than model of one zone.



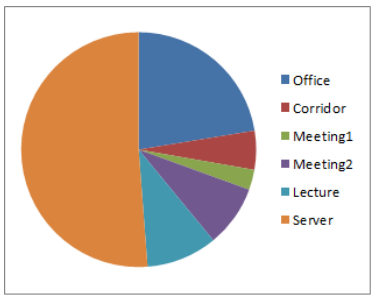
(a) People contribution



(b) Machines contribution



(c) Light contribution



(d) Inner loads together

Figura C.5: Total inner loads energy

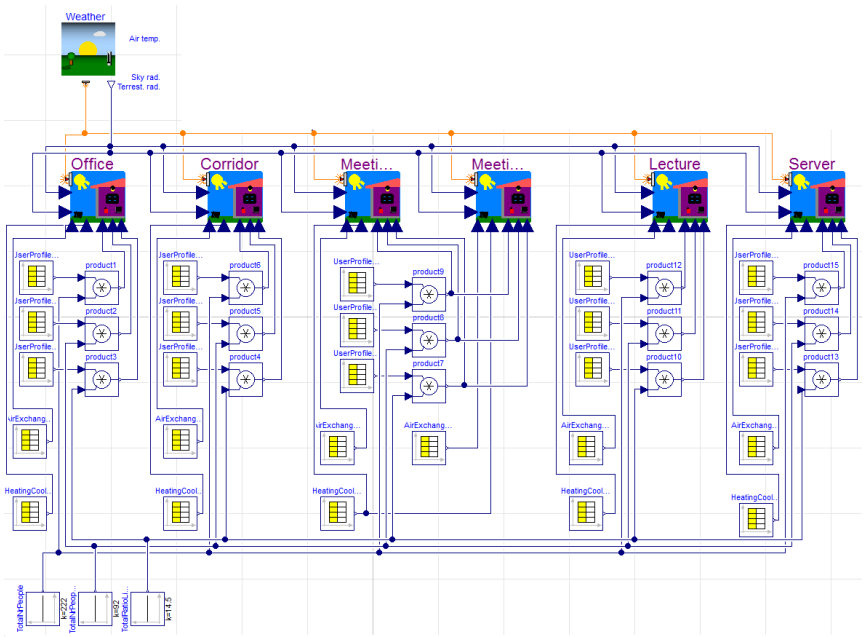


Figura C.6: J1615 model as six zones

C.4. Study of fixed parameters in six thermal zone model

C.4.1. Three parameters: people, machines and Light ratio

After applying the methods on this case, a varied parameter estimation is set for each method. Thus shows the huge possibility of parameter combinations that converge in a minimum solution.

Mathematical method is not so fast than before cases despite squared difference is reduced significantly respect to optimization algorithm of Matlab. However, Dymola has the lower error obtained with almost four times more requirements of time and 9 iterations more.

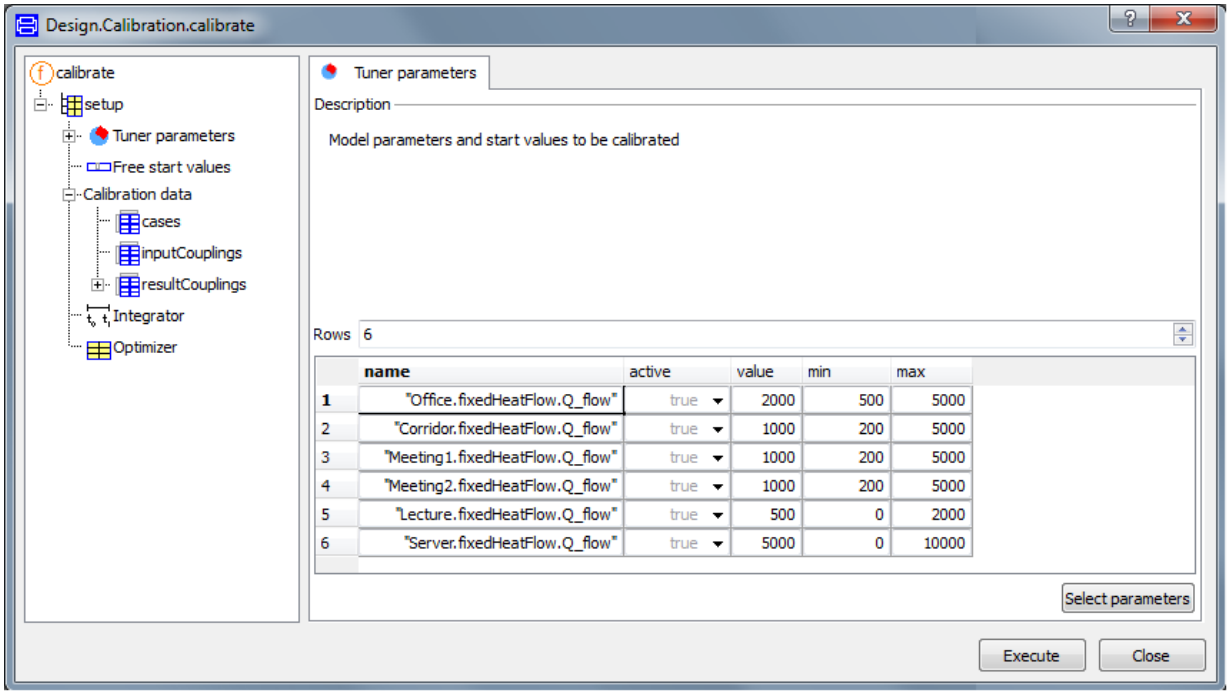
Cuadro C.4: Calibration of six zone with three parameters for one year

	Dymola	Toolbox Matlab	Math method
Nr People	26	222	117
Machines	102	92	268
Light Ratio (W/m^2)	25.0	14.5	13.7
$Error^2$ (kWh)	1.77e9	1.37e10	1.15e10
Iterations	23	14	14
Time (s)	11402	3420	3520

C.4.2. Six zones with fixed heat flow

Similar case is carried out as studied for one parameter in one zone. The difference is only six spaces instead one. Main propose of case is the stability of methods, specially the consistence of calibration function with Dymola.

Calibration with Dymola was impossible due to a fail in parameter estimation (fig: C.7). That means, the maximal number of parameter that Dymola is capable to perform in our model is 3.




```
Failed to expand Design.Calibration.calibrate(
  Design.Internal.Records.ModelCalibrationSetup(
    "Optimization2.J1615_FixedHeatFlow",
    {Design.Internal.Records.TunerParameter("Office.fixedHeatFlow.Q_flow", true,       2000, 500, 5000), Design.Internal.
    fill(Design.Internal.Records.FreeStartValues("", true, 0, -1E+100, 1E+100), 0),
    Design.Calibration.Internal.Dynamic_common(
      Design.Internal.Records.DynamicCommonCalibrationCases({"D:/mfu-lja/workspaces/onezone/optimierung/1615dinamicTable
      fill(Design.Internal.Records.DynamicCalibrationInputCoupling("", "", 1, 0),          0),
      {Design.Internal.Records.DynamicCalibrationResultCoupling("tEnergyMeterTotalHeat.q_kwh",          "Heat_Counter",
    Design.Internal.Records.CalibrationIntegrator(63504000, 95040000, 3600, 0,       "Dassl", 0.001, 0),
    Design.Internal.Records.CalibrationOptimizer(
      tolerance = 0.001,
      listOn = true,
      plotOn = true
    ))).
```

Figura C.7: Failure in calibration with Dymola for six parameters

Matlab algorithms submit stability in simulation. Thus, no parameter estimation is necessary because the goal is to compare the three methods.

D Results

D.1. Sensibility analysis and weight of parameters

Parameters on model have not a linear dependence between them, only mathematical model obtains linearise. Changing number of people in building to be estimated influences the output. However, number of machines and light ratio also influence the result in a similar way such that every parameter is not possible to be estimated individually. When a set of parameters have been tuned, the model is validated and the tuned parameters against other measured data to check that there is a good agreement between the simulation result and the new measurements.

For a specific series of measured data it is possible to get good fits by increasing the model complexity and the number of tuned parameters. However, this does not guarantee that the result is that good for other operating conditions.

The one zone model has a default values of *110* persons, *41* machines and *15.9* W/m^2. After the procedure in (B.4) having into account the real heating energy data, fitting values with optimization algorithm are respectively *113.5* persons, *33* machines and *14.25* W/m^2.

The weight of parameters has been calculated with Dymola, the linear parameter combination should be fixed in such a way to avoid one parameter take hold of the others. The next relation has been estimated for one week with a Dymola function that is able to calculate dependence between parameters:
$NrPeopleMachines + 0{,}8258 \cdot NrPeople + 8{,}5919 \cdot LightingPower$

This equation relates three parameters with each other. Then, no unique solution exists and all punts along the valley are possible solutions. (fig: D.1).

The weight of parameters refer is 7.9 % number of people, 9.6 % number of machines and 82.5 % ratio light. However, it is no possible to apply it for eleven parameter because this function is specific of Dymola that does not permit to estimate parameter of external file.

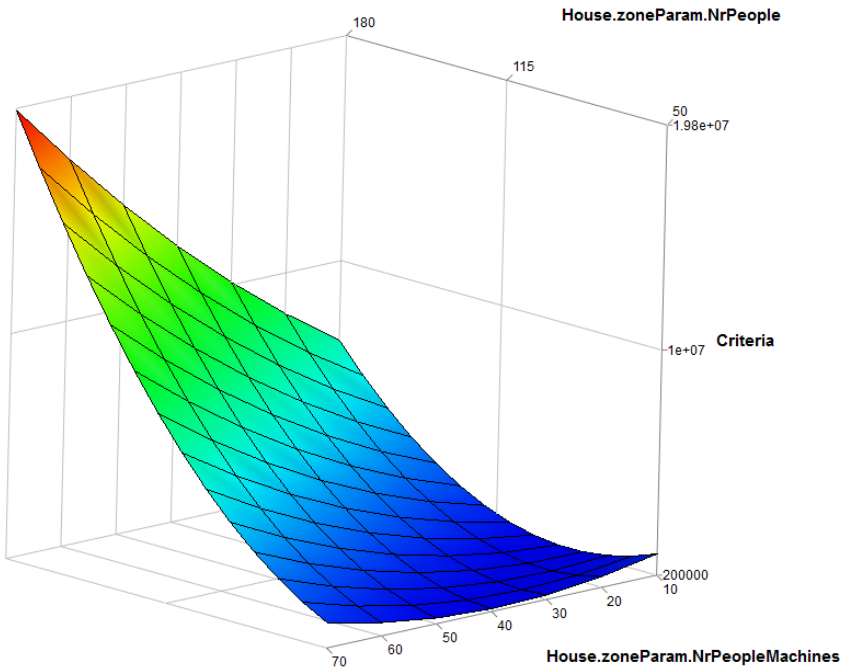


Figura D.1: Solution after sweep two parameter on one zone building

In the studied case, the estimation of internal loads for an office building is more practicable than for a laboratory, as standard values are available for offices but not for laboratories. The differences between the simulation and measurement of the heat consumption are clearly correlated to the accuracy of the internal loads estimation. The manual ventilation and heater set temperature are also crucial factors (fig: B.4).

After one week simulation, the main heat power flow depending of the parameters that variate with the time are represented in (fig: D.2), as we can see the influence of the air exchange and infiltration makes the sensor temperature on room drops bellow the temperature 20,5^oC during office hours. As consequence of this decreasing heating equipment turns on until temperature goes beyond 20,5^oC. During no office hours control temperature between 18 p.m. and 07 a.m. of next day is fixed to 16^oC, but never goes beyond 19,5^oC.

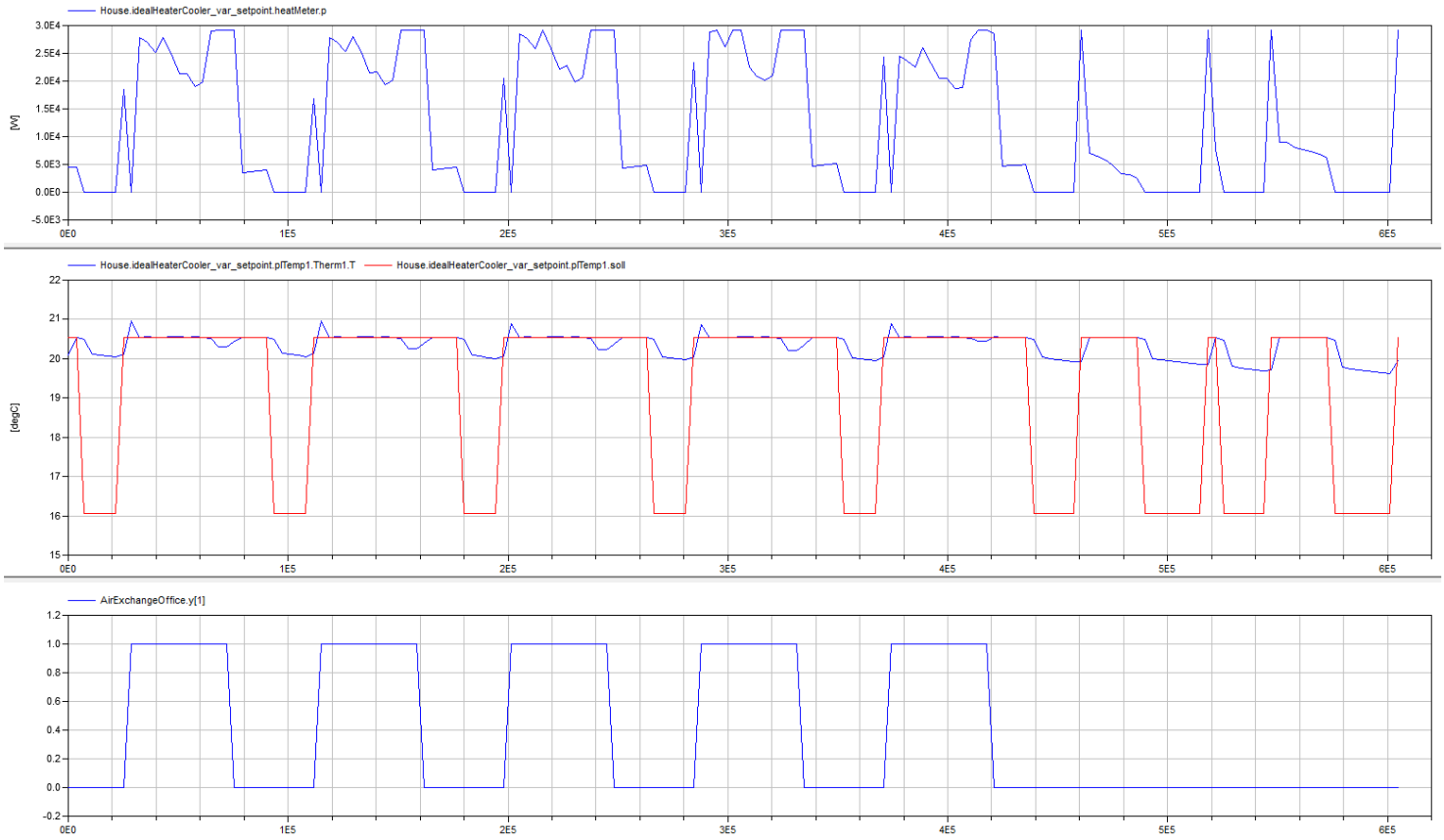


Figura D.2: Variable parameters

D.2. Comparison of methods and simulations

Through the simulation of different cases along the project we can estimate behaviour of three used methods depending of results in section C. The analysis of simulation results help to determine the best method in every case depending of the interest of program user.

1. Case 1 (1 parameter in onezone) (C.2.1).
   - ▷ Parameter estimation for both Matlab methods are almost the same for each time of simulation.
   - ▷ The major assessment about case 1 is referred to time, twelve times faster than Dymola method.
2. Case 2 (3 parameters in onezone) (C.2.2).

- ▷ Weight of heating contribution for light ratio is around eight times upper than people or machines.
- ▷ Reduction in number of fixed parameter on one zone model permits mathematical method be again the fastest one, exactly nine times faster than Dymola.

3. Case 3 (11 parameters in onezone) (C.2.3)
   - ▷ This is a notorious case, where error is reduced to similar values in seven times less time and 59 iterations less.
   - ▷ Parameter estimation for the toolbox of Matlab is quite similar to the default model due to the tolerance we gave ±20 %.
4. Case 4 (3 parameters in six zones) (C.4.1).
   - ▷ Time needed for mathematical method increases beyond the optimization algorithm ones.
   - ▷ Squared difference is reduced to 16.06 %.
   - ▷ Each method estimated parameters in a different way.
5. Case 5 (6 parameters in six zones) (C.4.2).
   - ▷ Most number of parameter that Dymola can solver is 3. In this case Dymola returns an error of simulation.

One zone model implies around 3 times less time for simulation and around 5 times less number of parameters despite the difference in results is not remarkable (D.3). Comparing results of model calibrations, blue line is the reference and the other three are the studied cases where the total adjustment of red line (case 1) is 88.92 %. Case 2 represented by green line has an adaptation to reference total heating energy of 90.43 %. While pink line shows case 4 that fitting better along the curve with 91.11 % as last energy point.

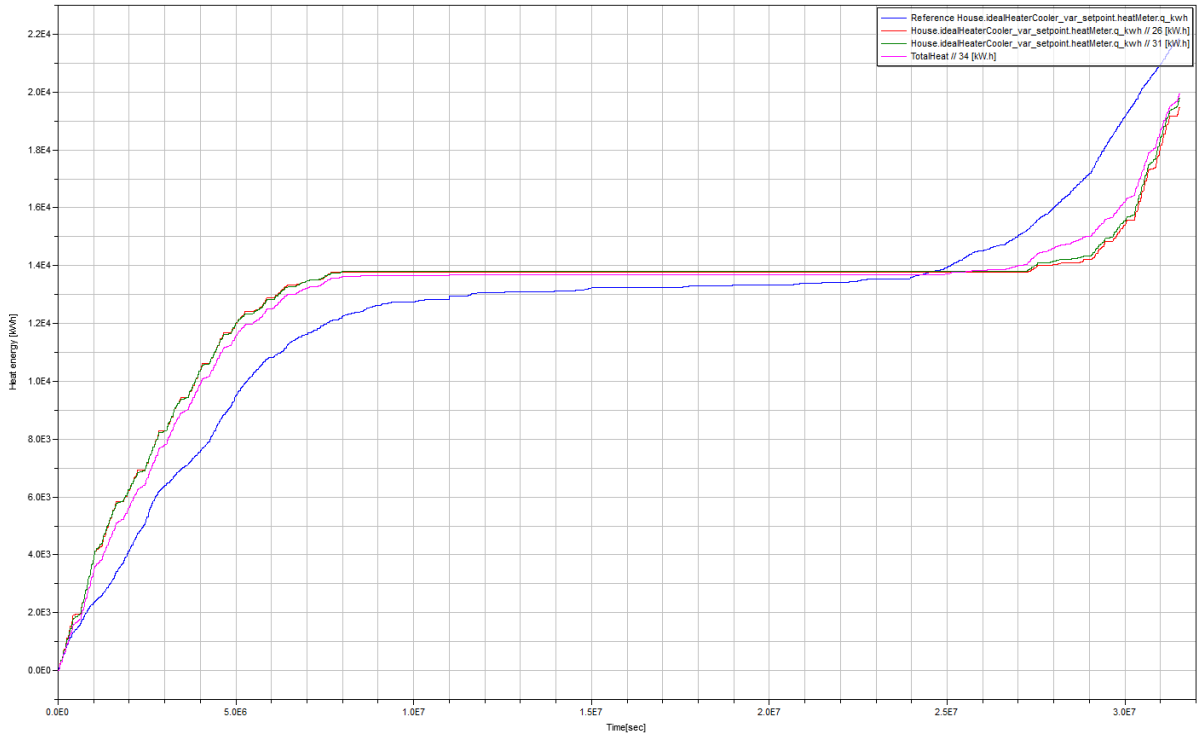


Figura D.3: Heating energy simulation for first four cases during one year

Using the same method (Optimization toolbox) for cases 1 to 4 it can differ behaviour of models in simulation. Sum of squared differences in kWh for one year are respectively *2.18e10*, *2.07e10*, *2.06e10*, *1.37e10*. Then, case with *3* parameters in six zones model generate minimal error of cases.

Respect to iterations, *15*, *14*, *65*, *14* are necessary.

For simulation time each case needs in seconds, *560*, *1498*, *13750*, *3420*.

Obviously, using the same method, error decreases over time and number of iterations. A consideration by user permits choice of correct model depending priorities or requirement.

E Conclusion

Thermal building simulation programs construct discontinuous approximations to a usually continuous differentiable cost function. For this reason, optimization algorithms can cause fail far from an optimal solution. In such cases, using high precision approximations may require a prohibitively long computation time if used for all iterations.

Numerical solution methods in Modelica allows the reuse of simulation models for optimization. Also nonlinear dynamic optimization problems can be treated efficiently as discrete-time optimal control problem and solved numerically by applying largescale nonlinear optimization methods.

Dymola is able to go from static measurements to calibration of parameters in an easy way for user despite the time needs.

When solving high-order non-linear systems, the process might diverge, since convergence is not guaranteed, and the initial parameter estimate becomes critical for convergence.

In both Matlab algorithms, the less the error tolerance is set the longer the algorithm is iterating to increase the solution accuracy. And despite, algorithms converge in global minimum, it can be reached by several combination of parameter (fig: D.1).

Both algorithms created in Matlab are good candidates for solving the examined models:

- ▷ The Pattern Search method of optimization toolbox uses low-cost, coarse precision approximations to the cost function when far from a solution, with the precision progressively increased as a solution is approached. This allows proving convergence to a first order optimal point of the cost function.
- ▷ Mathematical method has demonstrated to be the most robust and efficient of three. Gradient needs so many simulations as parameters exist, time for calibration increases exponentially over number of parameters.
- ▷ Respect to reduction in criterion per iteration, mathematical method is clearly the most accurate.

Dymola permits to estimate only parameters given as inputs into the program but not external files. That is a disadvantage that could be solve creating a ÇombiTableïn Dymola whose inconvenient is the implementation time.

In addition, Dymola permits to calibrate up to 3 parameters in our models, while Matlab algorithms can estimate more than 33 parameters (11 hours x 3).

Further works: Some of the futere lines that can be studied are:

- Application of developed algorithms over decisive parameters as air exchange and thermostat temperature.
- Implementation in parallel for decreasing estimation parameter time.
- Execute the mathematical developed method in building application.

Annex A: Code for optimization algorithm script

```
%%%%%%%%SCRIPT FOR OPTIMIZATION ALGORITHM %%%%%%%%%
%Path for Dymola m files.
DymolaPath = 'C:/Program Files/Dymola 2013 FD01/Mfiles';

%Path for workspace.
MatlabPath = 'D:/mfu-lja/workspaces';
addpath(genpath(DymolaPath));
addpath(MatlabPath);
addpath([MatlabPath,'/onezone']);
addpath(genpath([MatlabPath,'/onezone/optimierung']));

%Change directory in Dymola.
onezone = [MatlabPath,'/onezone'];
dymolaM(['cd("',onezone,'/optimierung")']);

tic;
%Initial parameter values.
x0=[110 ; 41 ; 15.9];
%Lower and upper bounds.
lb=[90 10 0];
ub=[250 250 16];

format short e
%Optimization algorithm options.
opts = psoptimset('MeshAccelerator','on','ScaleMesh','on');
opts = psoptimset(opts,'TolMesh',1e-6);
opts = psoptimset(opts,'TolBind',1e-6);
opts = psoptimset(opts,'Cache','on','CacheTol',1e-10);
opts = psoptimset(opts,'PollMethod','GSSPositiveBasisNp1');
opts = psoptimset(opts,'SearchMethod',@positivebasisnp1, ...
    'PlotFcns',{@psplotbestf, @psplotfuncount});
%Optimization function Patternsearch.
[x,Fval,ExitFlag,Output] = patternsearch(
    @J1615_onezone3parameters_function_patternsearchH7,x0,[],[],[],[],lb,ub,[],opts);
```

```
fprintf('The parameters variated value was : %d\n', x);
fprintf('The number of iterations was : %d\n', Output.iterations);
fprintf('The number of function evaluations was : %d\n', Output.funccount);
fprintf('The best function value found was : %g\n', Fval);
tiempo = toc;
fprintf('The process took %d seconds', tiempo);\\
%%%%%%%%FUNCTION FOR OPTIMIZATION ALGORITHM %%%%%%%%%
function F= J1615_onezone3parameters_function_patternsearchH7(x)

MatlabPath = 'D:/mfu-lja/workspaces';
onezone = [MatlabPath,'/onezone'];

% load 1615dinamicTable.csv data.
load([MatlabPath,'/dataJ.mat']);

%If Simulate_week: 1= 03.01-10.01,  2= 31.01-07.02, 3= 28.02-07.03,
%4=05.12-12.12, 52=01.01-31.12

simulate_week=1;

if simulate_week==1
   %Week: 03.01.2011 - 10.01.2011
     ydata=(dataJ(49:217,5)/1000)-16540;

else if simulate_week==2
       %Week: 31.01.2011 - 07.02.2011
         ydata=(dataJ(721:889,5)/1000)-21570;

    else if simulate_week==3
           %Week: 28.02.2011 - 07.03.2011
            ydata=(dataJ(1393:1561,5)/1000)-25640;

         else if simulate_week==4
              %Week: 05.12.2011 - 12.12.2011
               ydata=(dataJ(8113:8281,5)/1000)-33740;

              else if simulate_week==52
                      %Year: 01.01.2011 - 01.01.2012
                       ydata=(dataJ(49:8809,5)/1000)-16540;

                  else num2str('The simulate week can not be optimized')
```

```
75                  end
76              end
77          end
78      end
79  end
80
81  %dymolaM-command can execute every command that can be used in Dymola.
82  % Translate model J1615_onezone3Parameters.
83  res=dymolaM('translateModel("Optimization2.J1615_onezone3Parameters")');
84
85  % Parameters to set for optimization.
86  TotalNrPeople=x(1)
87  NrPeopleMachines=x(2)
88  RatioLights=x(3)
89
90  % Perform of parameter values in Dymola.
91  dymolaM(['House.zoneParam.NrPeople=',num2str(TotalNrPeople)]);
92  dymolaM(['House.zoneParam.NrPeopleMachines=',num2str(NrPeopleMachines)]);
93  dymolaM(['House.zoneParam.LightingPower=',num2str(RatioLights)]);
94
95  % Simulation of building.
96  if simulate_week==1
97  dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime=0, stopTime
        =604800, numberOfIntervals=0, outputInterval=3600, tolerance=1, resultFile="
        J1615_onezone3Parameters")');   %Simulation in Dymola with the value .k =x(1)
98  else if simulate_week==2
99          dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime
                =2419200, stopTime=3024000, numberOfIntervals=0, outputInterval=3600,
                resultFile="J1615_onezone3Parameters")');
100     else if simulate_week==3
101             dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime
                    =4838400, stopTime=5443200, numberOfIntervals=0, outputInterval=3600,
                    tolerance=1, resultFile="J1615_onezone3Parameters")');
102         else if simulate_week==4
103                 dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters",
                        startTime=29030400, stopTime=29635200, numberOfIntervals=0,
                        outputInterval=3600, tolerance=1, resultFile="
                        J1615_onezone3Parameters")');
104             else if simulate_week==52
105                     dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters",
                            startTime=0, stopTime=31536000, numberOfIntervals=0,
```

```
                        outputInterval=3600, tolerance=1, resultFile="
                        J1615_onezone3Parameters")');
                end
            end
        end
    end
end

%Loads data from result file .mat in a structure 'd'.
d=dymload([onezone,'/optimierung/J1615_onezone3Parameters.mat']);

%Extract from structure 'd' values of simulation.
TotalHeat= dymget(d,'House.idealHeaterCooler_var_setpoint.heatMeter.q_kwh');
TotalHeat= TotalHeat(1:(length(TotalHeat)-1));

F=sum((TotalHeat-ydata).^2)
```

Annex B: Code for mathematical algorithm function

```
%%%%%%%%FUNCTION FOR MATHEMATICAL ALGORITHM %%%%%%%%%
function [x_star] = levenberg_marquardt11
tic;

%Path of Dymola m files.
DymolaPath = 'C:/Program Files/Dymola 2013 FD01/Mfiles';

%Path of workspace.
MatlabPath = 'D:/mfu-lja/workspaces';
addpath(genpath(DymolaPath));
addpath(MatlabPath);
addpath([MatlabPath,'/onezone']);
addpath(genpath([MatlabPath,'/onezone/optimierung']));

%Change directory in Dymola.
onezone = [MatlabPath,'/onezone'];
dymolaM(['cd("',onezone,'/optimierung")']);

load([MatlabPath,'/dataJ.mat']);   %load 1615dinamicTable.csv data.
ydata=(dataJ(49:8809,5)/1000)-16540;   %measurement data for one year.

%solve nonlinear least squares problem using Levenberg-Marquardt
%algorithm.
%initial guess.
x = [0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5];

%increment in x for finite difference approximation.
delta_x = [0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01;0.01];

%great lambda means short stepsize.
lambda = 1;

%counter
counter = 0;
```

```
progress = 1;

while progress > 1e-6
  %counter
  counter = counter+1;

  % Simulation of Total Heat at vector of parameters 'x'.
  % Call to dymola here with value of x
  userprofil; % SCRIPT for editting the column corresponding to UserProfilesOffice.txt.

  % Model translation
  res=dymolaM('translateModel("Optimization2.J1615_onezone3Parameters")');

  % Model simulation
  dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime=0,
      stopTime=31536000, numberOfIntervals=0, outputInterval=3600, tolerance=1,
      resultFile="J1615_onezone3Parameters")');

  d=dymload([onezone,'/optimierung/J1615_onezone3Parameters.mat']);
  TotalHeat= dymget(d,'House.idealHeaterCooler_var_setpoint.heatMeter.q_kwh');
  TH= TotalHeat(1:(length(TotalHeat)-1));   %[8761x1]

  % Simulation of Total Heat at x+dx for each parameter.
  for k = 1:length(x)

      %x plus dx for parameter k.
      x_plus_dx = x(k) + delta_x(k);

      % Simulation of Total Heat at x plus dx.
      x(k)=x_plus_dx;
      userprofil;
      x(k)= x_plus_dx - delta_x(k);
      res=dymolaM('translateModel("Optimization2.J1615_onezone3Parameters")');
      dymolaM('simulateModel("Optimization2.J1615_onezone3Parameters", startTime=0,
          stopTime=31536000, numberOfIntervals=0, outputInterval=3600, tolerance=1,
          resultFile="J1615_onezone3Parameters")');

      d=dymload([onezone,'/optimierung/J1615_onezone3Parameters.mat']);
      TotalHeat= dymget(d,'House.idealHeaterCooler_var_setpoint.heatMeter.q_kwh');
      TH_dx(:,k)= TotalHeat(1:(length(TotalHeat)-1)); % TH_dx [8761x11]

```

```
72          F=sum(abs(TH_dx(:,k)-ydata).^2)  %criterion
73      end
74
75      %Right hand side
76      RHS = ydata - TH;     %[8761x1]
77
78      %gradient
79      for k = 1:length(x)
80
81          %Partial derivative in k-direction.
82          P(:,k) = (TH_dx(:,k) - TH)./delta_x(k);   %P [8761x11] each column is divided by
                the column delta_x
83      end
84
85      %stepsize
86      delta_x = (P'*P + lambda*diag(diag(P'*P)))\(P'*RHS); %[11x1] = {([11x8761][8760x11])
            + [11x11]} \ [11x1]
87
88      %update
89      x = x + delta_x;
90
91      %progress
92      progress = (delta_x'*delta_x)/(x'*x); %[1x11][11x1] / [1x11][11x1]
93
94      if mod(counter,10) == 0
95        disp(['Counter : ',num2str(counter)]);
96        lambda = 0.1*lambda;
97      end
98  end
99
100 %optimal solution
101 x_star = x
102
103 %number of iterations
104 disp(['Number of iterations : ',num2str(counter)]);
105
106 tiempo = toc;
107 fprintf('The process took %d seconds', tiempo);
108
109
110 %%%%%%%%SCRIPT USERPROFIL %%%%%%%%
```

```
file = [MatlabPath,'/onezone/optimierung/Tables/J1615'];
filename = [file,'/UserProfilesOffice.txt'];

Time=[0;3540;3600;7140;7200; ... ;597600;601140;601200;604740;];
People= [0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);x(2);x(3);x(3);x(4);x(4);x(5);x(5);
    x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x(10);x(11);x(11)
    ;0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);x(2);x(3);
    x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x(10);x(11);
    x(11);0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);x(2);
    x(3);x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x(10);x
    (11);x(11);0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1);x(2);
    x(2);x(3);x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x(10);x
    (10);x(11);x(11);0;0;0;0;0;0;0;0;0;0;0;0;      0;0;0;0;0;0;0;0;0;0;0;0;0;0;x(1);x(1)
    ;x(2);x(2);x(3);x(3);x(4);x(4);x(5);x(5);x(6);x(6);x(7);x(7);x(8);x(8);x(9);x(9);x
    (10);x(10);x(11);x(11);0;0;0;0;0; ... ;0;0;0;0;0;0;];
Machines=[0.1;0.1;0.1;0.1; ... ;0.1;0.1;0.1;0.1;];
Light1=[0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;1;1;0.3; ... ;0;0;0;0;];
Light2=[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0.3;0.3;0.3; ... ;0;0;0;0;];
A=[Time,People,Machines,Light1,Light2];


fileID = fopen([file,'/UserProfilesOffice.txt'],'wt');
fprintf(fileID,'#1\n');
fprintf(fileID,'double␣UserProfilesOffice(336,␣5)\n');


for i=1:length(A)
    fprintf(fileID, '%f\t%f\t%f\t%f\t%f\n',A(i,:));
end

fclose(fileID)
```

Bibliografía

[AB 2012] AB, Dassault S.: *Dymola: Dynamic Modelling Laboratory User Manual. Volume 2*, 2012

[Audet u. J. E. Dennis 2000] AUDET, Charles ; J. E. DENNIS, JR: Pattern Search algorithms for mixed variable programming. (2000), S. 573–594

[Bernard P. Zeigler 2000] BERNARD P. ZEIGLER, Tag Gon K. Herbert Praehofer P. Herbert Praehofer: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic System. (2000)

[Bertsekas 1999] BERTSEKAS, Dimitri P.: *Nonlinear Programming*. 1999

[H. Elmqvist 2005] H. ELMQVIST, S.E. Mattsson D. Brueck C. Schweiger D. Joos M. O. H. Ollson O. H. Ollson: Optimization for Design and Parameter Estimation. (2005), S. 255–266

[M. Lauster 2012] M. LAUSTER, M. Fuchs R. Streblow D. M. J. Teichmann T. J. Teichmann: Dynamic building model for city quartier simulation. (2012)

[Stefan Finsterle 2010] STEFAN FINSTERLE, Michael B. K.: A truncated Levenberg-Marquardt algorithm for the calibration of highly parameterized nonlinear models. (2010), S. 731–738

[The MathWorks 2003] THE MATHWORKS, Inc.: *Optimization Toolbox Users Guide of Matlab*, 2003

[Wetter u. Polak 2003] WETTER, Michael ; POLAK, Elijah: A convergence optimization method using pattern search algorithms with adaptative precision simulation. (2003), S. 1393–1400

[Wetter u. Wright 2003] WETTER, Michael ; WRIGHT, Jonathan: Comparison of a generalized pattern search and a genetic algorithm optimization method. (2003), S. 1400–1408