Técnicas Avanzadas de Modelado y Simulación

5. Modelica y Modelado Orientado a Objetos

Ernesto Kofman

Departamento de Control, Escuela de Ingeniería Electrónica, Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario.
CIFASIS – CONICET

Organización de la Presentación

Modelado Orientado a Objetos

- Hasta aquí, planteamos los modelos como un conjunto de relaciones constitutivas y estructurales que conforman un sistema de DAEs.
- Si hay componentes repetidos, encontramos duplicadas las ecuaciones constitutivas correspondientes.
- En las relaciones estructurales, hay variables que se igualan y otras con suma cero al conectarse.

Ante esto:

- ¿es necesario plantear todas las relaciones constitutivas al agregar un componente, o alcanza con decir cuál es el componente agregado?
- ¿es necesario listar todas las relaciones estructurales o alcanza con decir cómo están conectados los componentes?.

Con el Modelado Orientado a Objetos alcanzará con decir qué componentes tiene el sistema y cómo están conectados.

Organización de la Presentación

Ejemplo Introductorio

En el siguiente circuito cada componente incluye los potenciales de sus conectores:

Relaciones constitutivas:

$$C\,\frac{\mathrm{d}v_C}{\mathrm{d}t}(t) - i_C(t) = 0 \quad (5.1a)$$

$$V_{C_p}(t) - V_{C_n}(t) - v_C(t) = 0 \quad (5.1b)$$

$$L\,\frac{\mathrm{d}i_L}{\mathrm{d}t}(t) - v_L(t) = 0 \quad (5.1c)$$

$$V_{L_p}(t) - V_{L_n}(t) - v_L(t) = 0 \quad (5.1d)$$

$$R\;i_R(t) - v_R(t) = 0 \quad (5.1e)$$

$$V_{R_p}(t) - V_{R_n}(t) - v_R(t) = 0 \quad (5.1f)$$

$$R_2\;i_{R_2}(t) - v_{R2}(t) = 0 \quad (5.1g)$$

$$V_{R2_p}(t) - V_{R2_n}(t) - v_{R2}(t) = 0 \quad (5.1h)$$

$$V_{G_p}(t) = 0 \quad (5.1i)$$

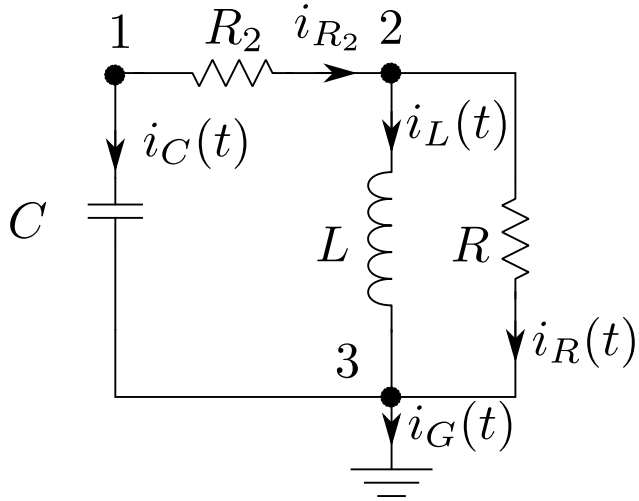


Figura 5.1: Circuito RLC

Ejemplo Introductorio

En el siguiente circuito cada componente incluye los potenciales de sus conectores:

Relaciones estructurales:

$$V_{C_p}(t) - V_{R2_p}(t) = 0 \quad (5.2a)$$
$$-i_C(t) - i_{R2}(t) = 0 \quad (5.2b)$$
$$V_{R2_n}(t) - V_{L_p}(t) = 0 \quad (5.2c)$$
$$V_{R2_n}(t) - V_{R_p}(t) = 0 \quad (5.2d)$$
$$i_{R2}(t) - i_L(t) - i_R(t) = 0 \quad (5.2e)$$
$$V_{C_n}(t) - V_{G_p}(t) = 0 \quad (5.2f)$$
$$V_{R_n}(t) - V_{G_p}(t) = 0 \quad (5.2g)$$
$$V_{L_n}(t) - V_{G_p}(t) = 0 \quad (5.2h)$$
$$i_R(t) + i_L(t) + i_C(t) - i_G(t) = 0 \quad (5.2i)$$

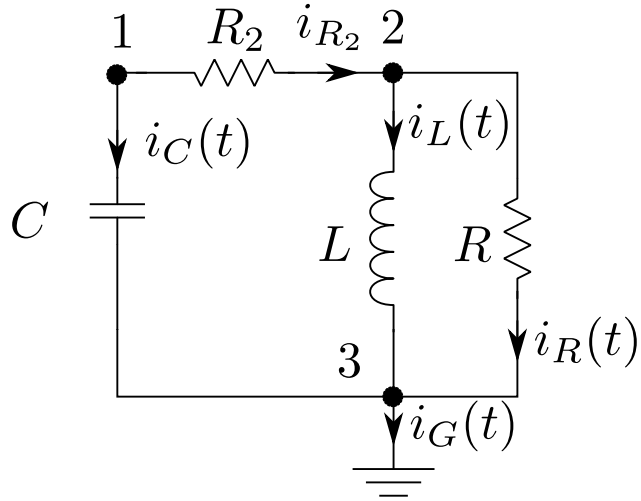


Figura 5.1: Circuito RLC

Ejemplo Introductorio I

El modelo, con 18 ecuaciones e incógnitas, puede representarse de manera trivial en Modelica:

```
model RLC
  Real vC(start=1), VCp, VCn, iC;
  Real vL, VLp, VLn, iL;
  Real vR, VRp, VRn, iR;
  Real vR2, VR2p, VR2n, iR2;
  Real VGp, iG;
  parameter Real C=1, L=1, R=1, R2=2;
equation
  //capacitor
  C * der(vC) - iC = 0;
  VCp - VCn - vC = 0;
  //inductor
  L * der(iL)- vL = 0;
  VLp - VLn - vL = 0;
  //resistor
  R * iR -vR = 0;
```

Ejemplo Introductorio II

```
  VRp - VRn - vR = 0;
  //resistor2
  R2 * iR2 - vR2= 0;
  VR2p - VR2n - vR2 = 0;
  // ground
  VGp = 0;
  // structure: Node 1
  VCp - VR2p = 0;
  -iC - iR2 = 0;
  // structure: Node 2
  VR2n - VLp = 0;
  VR2n - VRp = 0;
  iR2 - iL - iR = 0;
  // structure: Node 3
  VCn - VGp = 0;
  VRn - VGp = 0;
  VLn - VGp = 0;
  iR + iL + iC - iG = 0;
end RLC;
```

Ejemplo Introductorio III

Veremos a continuación estrategias para representar el mismo modelo de manera más simple y estructurada.

Organización de la Presentación

Modelado por Componentes

- En el modelo anterior las ecuaciones del resistor aparecen repetidas.
- El código es muy largo, con todas las declaraciones y ecuaciones en un mismo nivel jerárquico.
- Esto sería equivalente a escribir un programa largo en una única sección de código plano.

Lo primero que haremos entonces para simplificar y estructurar el modelo es dividirlo en distintos objetos pertenecientes a distintas clases.

Modelado por Componentes

```
model Capacitor
  Real v, i, Vp, Vn;
  parameter Real C=1;
equation
  C * der(v) - i = 0;
  Vp - Vn - v = 0;
end Capacitor;

model Inductor
  Real v, i, Vp, Vn;
  parameter Real L=1;
equation
  L * der(i) - v = 0;
  Vp - Vn - v = 0;
end Inductor;
```

```
model Resistor
  Real v, i, Vp, Vn;
  parameter Real R=1;
equation
  R * i - v = 0;
  Vp - Vn - v = 0;
end Resistor;

model Ground
  Real i, Vp;
equation
  Vp  = 0;
end Ground;
```

Aquí, cada modelo constituye una clase. Cada clase tiene como atributos ciertos parámetros (C, L, R) y ciertas variables (en este caso, de la clase `Real`).

Modelado por Componentes

Podemos entonces construir el modelo del circuito RLC instanciando objetos de estas clases y agregando las relaciones estructurales:

```
model RLC2
  Capacitor cap(v.start=1);
  Inductor ind;
  Resistor res, res2(R=2);
  Ground gr;
equation
 //Node 1
 cap.Vp - res2.Vp = 0;
 -cap.i - res2.i = 0;
 //Node 2
 res2.Vn-ind.Vp = 0;
 res2.Vn - res.Vp = 0;
 res2.i - ind.i - res.i = 0;
 //Node 3
 cap.Vn - gr.Vp = 0;
 res.Vn - gr.Vp = 0;
 ind.Vn - gr.Vp = 0;
 res.i+ind.i+cap.i-gr.i = 0;
end RLC2;
```

Este modelo es equivalente al anterior, pero con una descripción mucho más concisa.

Organización en Paquetes

- Para poder utilizar el modelo con componentes sin problemas, las clases deben estar definidas dentro de un mismo paquete.
- Para esto existe la clase especializada package que funciona como contenedor de otras clases (incluidos otros paquetes).

```
package Circuits1
  model Capacitor
    Real v, i, Vp, Vn;
    parameter Real C=1;
  equation
 ...
    package Examples
     model RLC2
       Capacitor cap(v.start=1);
       Inductor ind;
       Resistor res, res2(R=2);
       Ground gr;
     equation
       //Node 1
       cap.Vp - res2.Vp = 0;
       -cap.i - res2.i = 0;
       //Node 2
       res2.Vn-ind.Vp = 0;
       res2.Vn - res.Vp = 0;
       res2.i-ind.i-res.i = 0;
       //Node 3
       cap.Vn - gr.Vp = 0;
       res.Vn - gr.Vp = 0;
       ind.Vn - gr.Vp = 0;
       res.i+ind.i+cap.i-gr.i=0;
     end RLC2;
   end Examples;
end Circuits1;
```

Organización de la Presentación

Conectores y Conexiones

Como mencionamos antes, las relaciones estructurales pueden tener dos formas:

1. Igualación de variables (para magnitudes como potenciales, posiciones, etc.).
2. Suma de variables igual a cero (para magnitudes como corrientes, fuerzas, etc.)

En el contexto del lenguaje Modelica, las variables que se suman se denominan variables de flujo.

Para especificar conexiones Modelica provee la clase especializada `connector`, que permite describir qué variables se vinculan en una conexión y si las mismas son variables de flujo.

Conectores

Por ejemplo, para el caso de los circuitos eléctricos, podemos utilizar el siguiente conector:

```
connector Pin
  Real v;
  flow Real i;
end Pin;
```

Este código define la clase `Pin` que tiene dos atributos: el potencial `v` y la corriente `i` que es una variable de flujo.

Conectores y Conexiones

Usando conectores `Pin` podemos crear el modelo de un capacitor:

```
model Capacitor
  Pin p, n;
  Real v, i;
  parameter Real C=1;
equation
  C * der(v) - i = 0;
  v = p.v - n.v;
  p.i = i;
  n.i + p.i = 0;
end Capacitor;
```

Notar que estamos tomando positiva la corriente que entra por el conector `p` (positivo) y considerando positivas las corrientes entrantes a los conectores.

Conectores y Conexiones

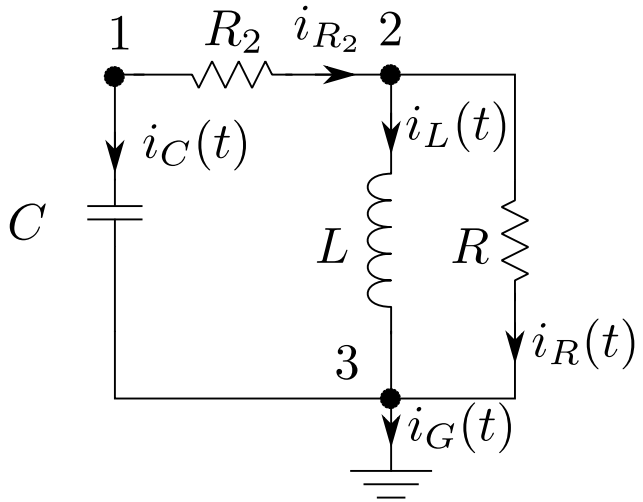
De manera similar podemos definir los restantes componentes:

```
model Inductor
  Pin p, n;
  Real v, i;
  parameter Real L=1;
equation
  L * der(i) - v = 0;
  v = p.v - n.v;
  p.i = i;
  n.i + p.i = 0;
end Inductor;

model Resistor
  Pin p, n;
  Real v, i;
  parameter Real R=1;
equation
  R * i - v = 0;
  v = p.v - n.v;
  p.i = i;
  n.i + p.i = 0;
end Resistor;

model Ground
  Pin p;
equation
  p.v = 0;
end Ground;
```

Conectores y Conexiones

Utilizando estos componentes, podemos armar el modelo del circuito RLC como sigue:

```
model RLC3
  Capacitor cap(v.start=1);
  Inductor ind;
  Resistor res, res2(R=2);
  Ground gr;
equation
  connect(cap.p,res2.p);
  connect(res2.n,ind.p);
  connect(res2.n,res.p);
  connect(cap.n,gr.p);
  connect(ind.n,gr.p);
  connect(res.n,gr.p);
end RLC3;
```



Conectores y Conexiones

Al declarar `connect(cap.p,res2.p)` estamos diciendo que los conectores positivos del capacitor `cap` y del resistor `res2` están conectados. Esto implicará dos cosas:

1. que hay una ecuación implícita `cap.p.v=res2.p.v`.
2. que las variables de flujo `cap.p.i` y `res2.p.i` participarán de una suma igual a cero junto a las variables de flujo de los demás conectores que estén conectados al mismo punto.
   En este caso, como no hay otro conector conectado a dicho nodo, resultará directamente `cap.p.i+res2.p.i=0`.

De esta manera, podemos definir directamente las relaciones estructurales brindando las conexiones, sin necesidad de enumerar ninguna ecuación de manera explícita.

Organización de la Presentación

Herencia

Observando los componentes que definimos, hay atributos y ecuaciones repetidos:

- A excepción del modelo de la tierra, el resto de los componentes tienen los mismos atributos (salvo el parámetro)
- Tienen también las mismas ecuaciones (salvo la primera, que define que se trata de un capacitor, inductor o resistor).
- Estas coincidencias se deben a que son dipolos pasivos. Cualquier dipolo pasivo tendrá dos conectores, una diferencia de potencial entre ambos y una corriente definida como positiva cuando ingresa por el conector positivo.

Estas coincidencias pueden explotarse utilizando el concepto de herencia en los lenguajes orientados a objetos.

Herencia

Para esto, podemos definir una clase correspondiente al modelo de un dipolo pasivo como sigue:

```
partial model OnePort
  Pin p, n;
  Real v, i;
equation
  v = p.v - n.v;
  p.i = i;
  n.i + p.i = 0;
end OnePort;
```

Notar que utilizamos el modificador `partial` para definir la clase. Esto implica que la clase no se puede instanciar, sólo puede utilizarse para *heredar* sus atributos y ecuaciones.

Herencia

Aprovechando la herencia podemos reformular de manera más simple las clases correspondientes a los dipolos pasivos:

```
model Capacitor
  extends OnePort;
  parameter Real C=1;
equation
  C * der(v) - i = 0;
end Capacitor;

model Inductor
  extends OnePort;
  parameter Real L=1;
equation
  L * der(i) - v = 0;
end Inductor;

model Resistor
  extends OnePort;
  parameter Real R=1;
equation
  R * i - v = 0;
end Resistor;
```

El comando `extends` permite heredar todos los atributos y ecuaciones de una clase parcial al crear una nueva clase.

Organización de la Presentación

Modelado Jerárquico

La Figura 5.2 muestra el circuito equivalente de un panel solar que está conectado a un circuito elevador tipo *Boost*. El esquema a su vez alimenta una resistencia de carga R.

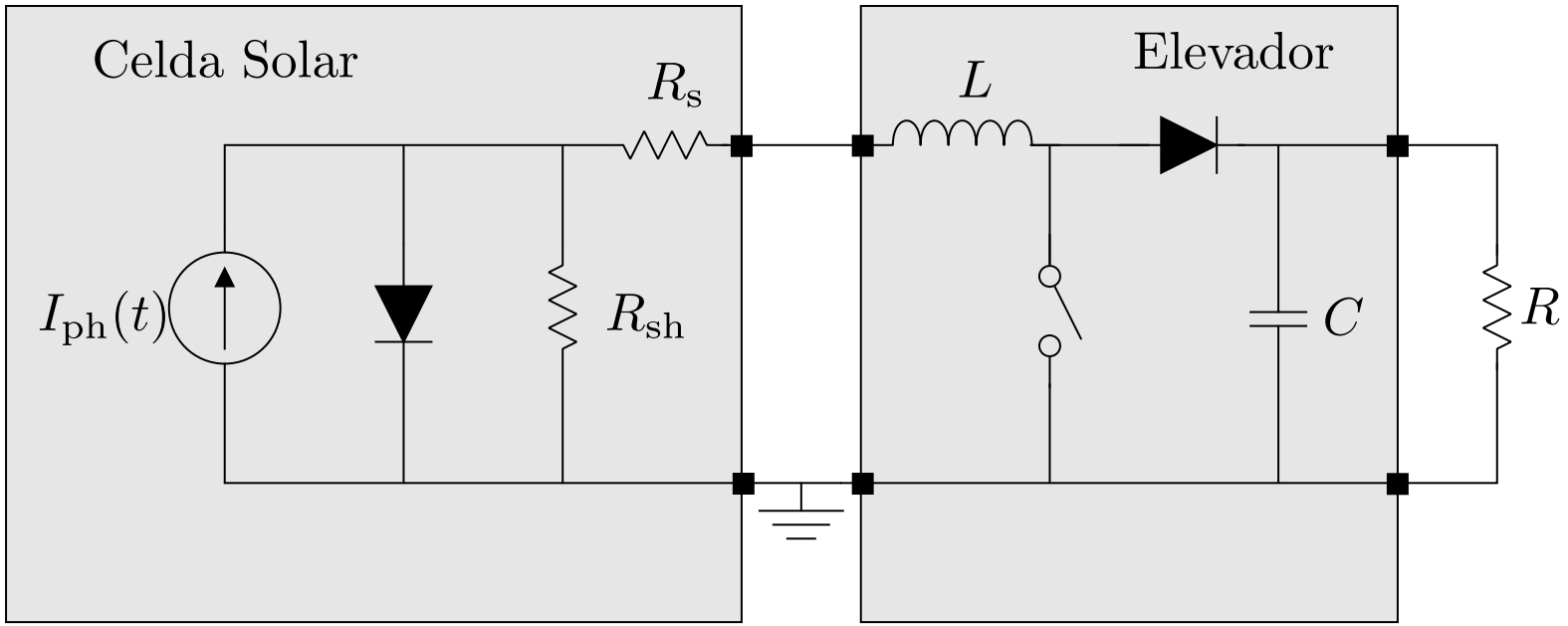


Figura 5.2: Circuito equivalente de un panel solar con un convertidor *Boost*.

Modelado Jerárquico

Para modelar este circuito resta definir los modelos de la fuente de corriente, de los diodos y de la llave (los restantes componentes ya los conocemos):

```
model Diode
  extends OnePort;
  parameter Real Ron=1e-5, Roff
     =1e5, Vknee=0.6;
equation
  i= if v>Vknee then (v-Vknee)/
     Ron+Vknee/Roff else v/Roff;
end Diode;

model Switch
  extends OnePort;
  parameter Real Ron=1e-5, Roff
     =1e5;
  discrete Real s;
equation
  v= if s>0.5 then i*Ron else i*
     Roff;
end Switch;

model ConstCurr
    extends OnePort;
  parameter Real I=1;
equation
  i= -I;
end ConstCurr;
```

Modelado Jerárquico

Luego, conectando los componentes, el modelo completo nos quedará:

```
model PanelBoost
  Capacitor cap(C=1e-4);
  Inductor ind(L=1e-4);
  Ground gr;
  Switch sw;
  Diode di;
  Resistor rs(R=0.35),rsh(R=200)
     ,rl(R=10);
  Diode dio;
  ConstCurr cc(I=2);
  parameter Real dc=0.75;
  parameter Real T=1e-4;
equation
  connect(cc.p,dio.p);
  connect(cc.p,rsh.p);
  connect(cc.p,rs.p);
  connect(cc.n,dio.n);
  connect(cc.n,rsh.n);
  connect(cc.n,gr.p);
  connect(rs.n,ind.p);
  connect(ind.n,sw.p);
  connect(ind.n,di.p);
  connect(di.n,cap.p);
  connect(cap.p,rl.p);
  connect(cap.n,rl.n);
  connect(cap.n,sw.n);
  connect(cap.n,gr.p);

algorithm
   when sample(0,T) then
     sw.s:=1;
   end when;
   when sample(dc*T,T) then
     sw.s:=0;
   end when;
end PanelBoost;
```

Modelado Jerárquico

- Si bien este modelo funcionará, tiene muchos componentes en un mismo nivel jerárquico.
- Esto en general complica la construcción, la visualización y la verificación de los modelos.
- Es conveniente dividir el modelo en submodelos más simples

Para esto, podemos construir dos subsistemas, uno correspondiente al modelo del panel solar y el otro al del circuito Boost, tal cual aparecen en la Figura 5.2.

Modelado Jerárquico I

Estos subsistemas pueden definirse según el siguiente Código de Modelica:

```
model Panel
 Pin p,n;
 Resistor rs(R=0.35),rsh(R=200);
 Diode dio;
 ConstCurr cc(I=2);
equation
  connect(cc.p,dio.p);
  connect(cc.p,rsh.p);
  connect(cc.p,rs.p);
  connect(rs.n,p);
  connect(cc.n,dio.n);
  connect(cc.n,rsh.n);
  connect(cc.n,n);
end Panel;
```

Modelado Jerárquico II

```
model Boost
  Pin p1,p2,n;
  Capacitor cap(C=1e-4);
  Inductor ind(L=1e-4);
  Switch sw;
  Diode di;
  parameter Real dc=0.5;
  parameter Real T=1e-4;
equation
  connect(p1,ind.p);
  connect(ind.n,sw.p);
  connect(ind.n,di.p);
  connect(di.n,cap.p);
  connect(cap.p,p2);
  connect(cap.n,n);
  connect(cap.n,sw.n);
 algorithm
   when sample(0,T) then
     sw.s:=1;
```

Modelado Jerárquico III

```
    end when;
    when sample(dc*T,T) then
      sw.s:=0;
    end when;
end Boost;
```

- Conectando el modelo del Panel a un resistor de carga y a una tierra, podemos probar su funcionamiento por separado.
- conectando el modelo del Boost a una fuente de tensión, a un resistor de carga y a una tierra, podemos también simular su funcionamiento independiente.

Esto facilita enormemente la depuración de errores en los modelos.

Modelado Jerárquico

Usando estas nuevas clases podemos construir de manera mucho más simple el modelo del panel con el circuito elevador:

```
model PanelBoost
  Panel panel;
  Boost boost;
  Ground gr;
  Resistor rl(R=10);
equation
  connect(panel.p,boost.p1);
  connect(boost.p2,rl.p);
  connect(rl.n,boost.n);
  connect(boost.n,panel.n);
  connect(boost.n,gr.p);
end PanelBoost;
```

Modelado Jerárquico

- Las nuevas clases `Panel` y `Boost` pueden reutilizarse en otros modelos o para crear modelos complejos con varias instancias de los mismos.
- Por ejemplo, podemos formar un conjunto de paneles solares consistente en N filas, donde cada fila tiene M paneles conectados en paralelo y las filas están conectadas en serie entre sí (así es como se utilizan habitualmente los paneles).

Modelado Jerárquico

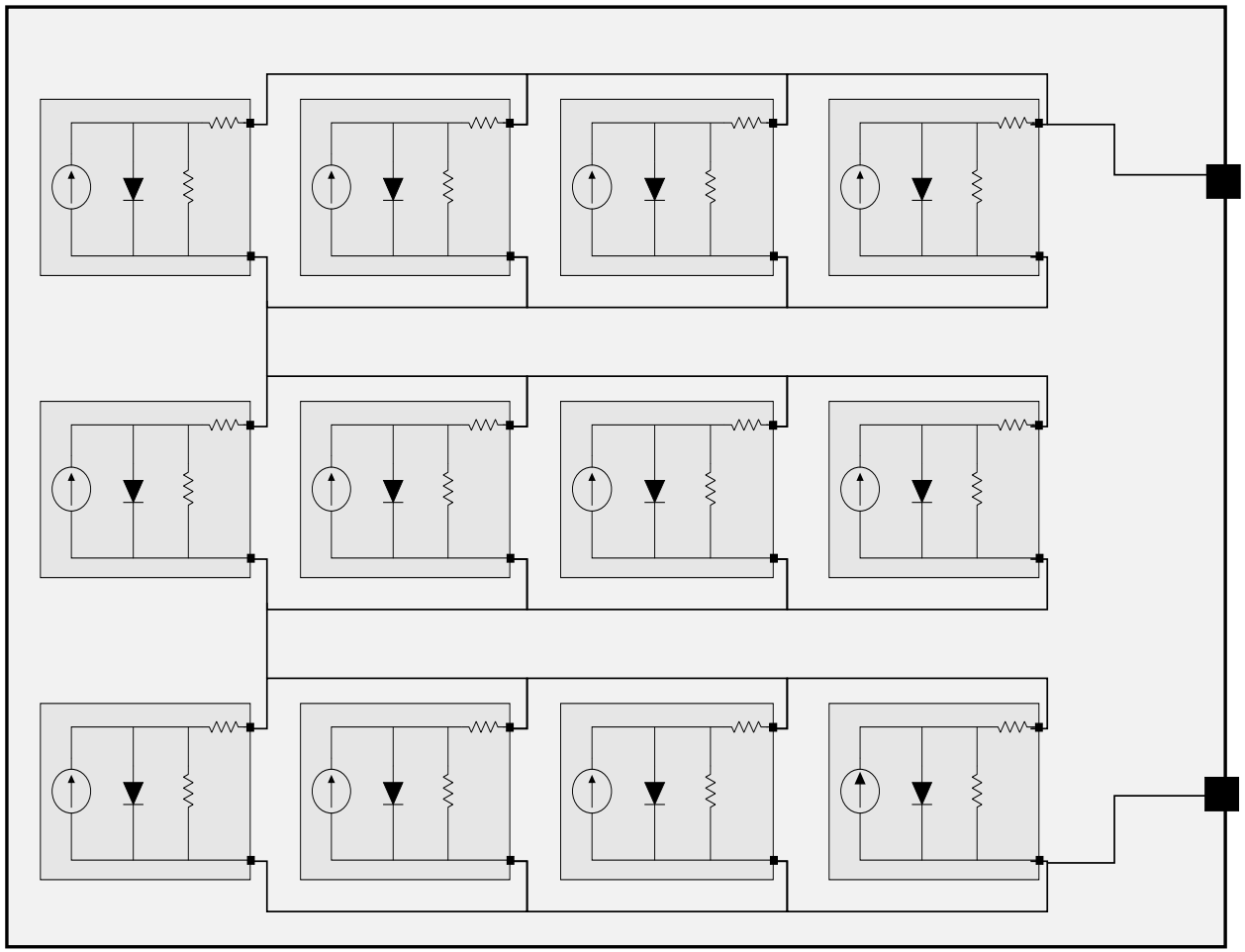


Figura 5.3: Conjunto de Paneles Solares

Modelado Jerárquico

El modelo para el conjunto de paneles solares puede plantearse como sigue:

```
model PanelArray
  parameter Integer N=3,M=4;
  Panel[N,M] panel;
  Pin p,n;
equation
  for i in 1:N loop
     for j in 1:M-1 loop
       connect(panel[i,j].p,panel[i,j+1].p);
       connect(panel[i,j].n,panel[i,j+1].n);
     end for;
  end for;
  for i in 1:N-1 loop
     connect(panel[i,1].n,panel[i+1,1].p);
  end for;
  connect(panel[1,1].p,p);
  connect(panel[N,1].n,n);
end PanelArray;
```

Modelado Jerárquico

Luego, podemos utilizar esta clase en lugar del panel en el modelo con el circuito Boost y la carga resistiva:

```
model PanelArrayBoost
  PanelArray panel(N=4,M=3);
  Boost boost;
  Ground gr;
  Resistor rl(R=10);
equation
  connect(panel.p,boost.p1);
  connect(boost.p2,rl.p);
  connect(rl.n,boost.n);
  connect(boost.n,panel.n);
  connect(boost.n,gr.p);
end PanelArrayBoost;
```

Organización de la Presentación

Modelado Orientado a Objetos

Organización de la Presentación

Modelado Orientado a Objetos

Organización de la Presentación

Modelado Orientado a Objetos