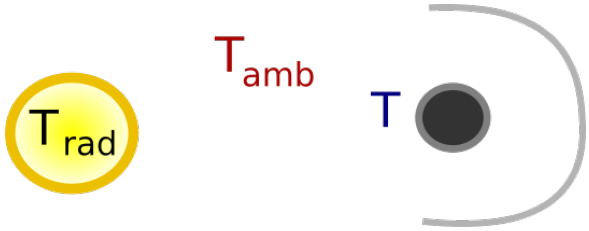
Simulación y linealización de un modelo de horno con radiación

Este vídeo ejecutó sin errores en Matlab **2022a**

Presentación en vídeo en https://youtu.be/8f5lJnR-c-0

Modelo teórico

Este código simula un calentador por radiación descrito por la figura:



de modelo:

$$\frac{dT}{dt} = \frac{1}{MC_e}\left(h\cdot(T_{amb}-T) + k\cdot(T_{rad}^4 - T^4) - k_2\cdot T^4\right)$$

siendo $T(t)$ la temperatura de una pieza depositada en el horno, $T_{amb}(t)$ la temperatura del aire que rodea al objeto calentado (con transferencia por conducción/convección lineal, constante h) y $T_{rad}(t)$ la temperatura de un elemento radiador que, dado que $T_{rad}(t)$ es elevada, produce una transmisión de calor por radiación (constante k), teniendo la pieza calentada asimismo unas pérdidas por radiación a cuerpos de temperatura bastante más baja de constante k_2.

Las variables de entrada son T_{amb} y T_{rad}, por lo que el modelo con una ecuación permite obtener (o simular) $T(t)$.

*Considerar T_{rad} como entrada implica asumir que las subidas y bajadas de T no influyen en T_{rad}, porque T_{rad} tenga algún sistema de control interno de temperatura o porque para el objeto radiante, el término $k\cdot(T_{rad}^4 - T^4)$ sea despreciable. Por ejemplo, la temperatura del sol no se ve influenciada por los cambios de temperatura en la superficie de la tierra.

Codificación del modelo en repr. interna normalizada (no lineal)

Parámetros constantes del modelo

```
MCe=500; k=2e-8; h=10; k2=6e-9;
```

Ecuación del modelo normalizada

```
dT_dt=@(Trad,Tamb,T) ...
    1/MCe*(k*(Trad^4 - T^4) -k2*T^4+ h*(Tamb - T));
```

Definición de las entradas a simular:

Escogemos un **punto de funcionamiento** de entradas (en Kelvin), arbitrario:

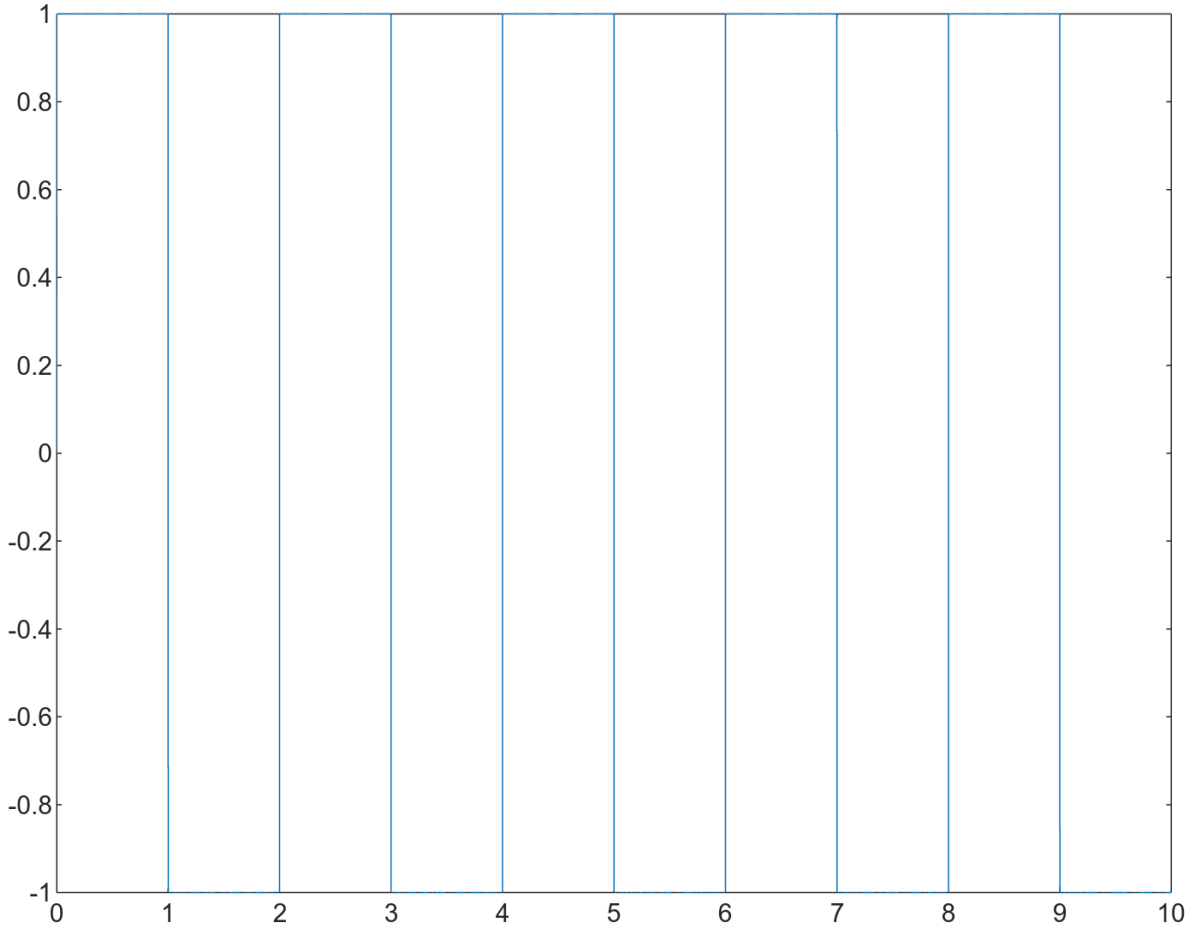
```
T_radpf = 273+600; T_ambpf = 273+110; %600 y 110 centigrados,
respectivamente.
```

Definimos funcion auxiliar para generar la entrada en función del tiempo, por ejemplo, un generador de ondas cuadradas de un período fijado

```
ondacuadrada=@(time,periodo) sign( sin(2*pi*time/periodo) );
```

Por ejemplo, este comando genera y dibuja una onda cuadrada de período 2 segundos desde el instante t=0 a t=10:

```
fplot( @(t) ondacuadrada(t,2) , [0,10] )
```



Especificación de funciones del tiempo Trad, Thor **arbitrarias**

Elegimos Tradiador como 75 s en pto. func. (600 ºC) y luego, onda cuadrada de

período 100 segundos y amplitud 50 grados alrededor de él, y luego de mayor amplitud

```
T_rad=@(time) T_radpf+ (time>75).*ondacuadrada(time-75,100)*50+
(time>280).*(30+ondacuadrada(time-280,120)*170);
```

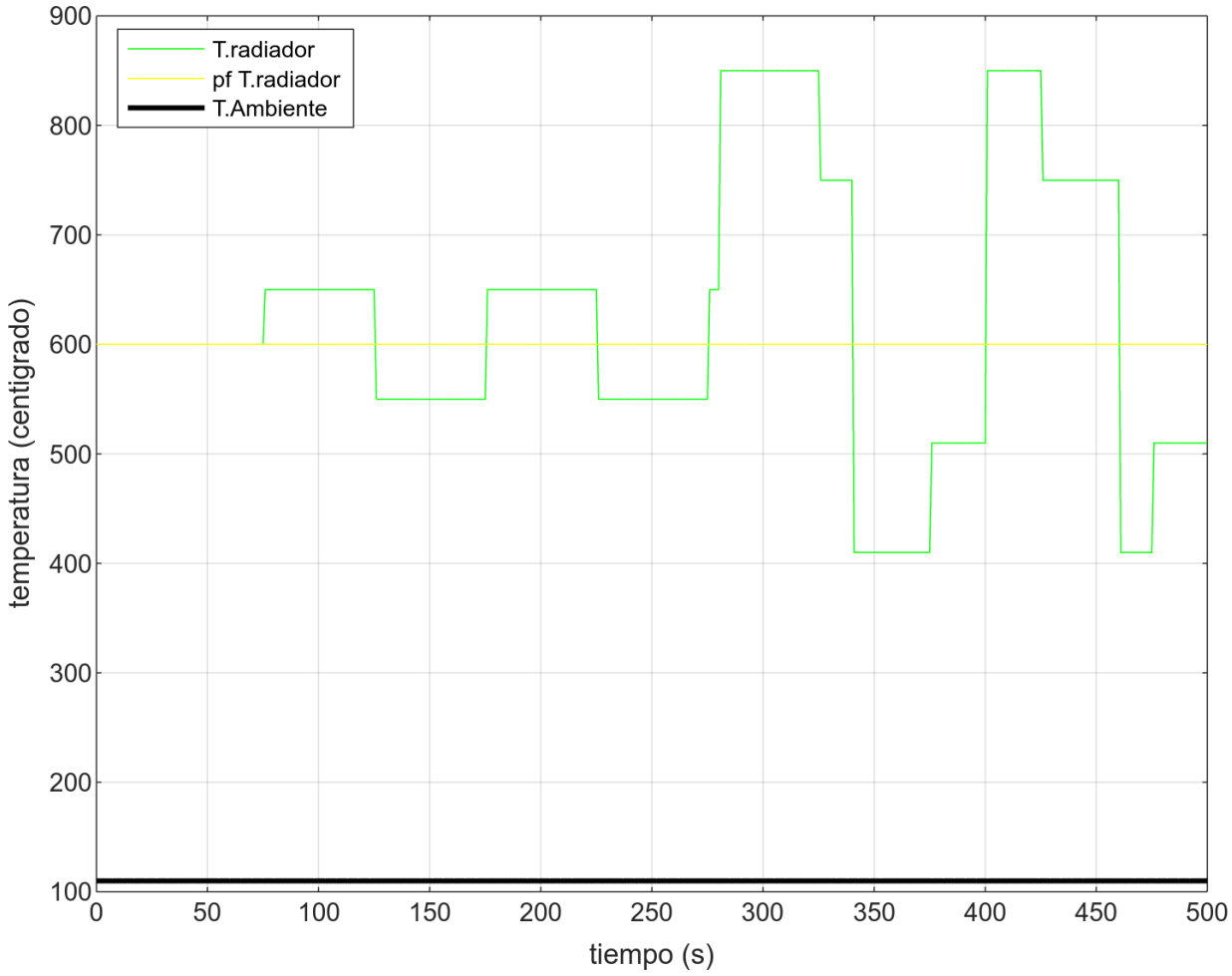
y la temperatura del aire ambiente, por simplicidad, la elegimos constante, igual al punto de funcionamiento

```
T_amb=@(time) T_ambpf * ones(size(time)); %función del tiempo que
devuelve, simplemente, una constante
```

Vamos a dibujarlas durante 500 segundos:

```
vectordetiempos=0:1:500;
plot(vectordetiempos,T_rad(vectordetiempos)-273,'g') %pasamos a
centigrado, T_rad en verde
hold on
```

```
plot([0 500],[T_radpf,T_radpf]-273,'y'); %pasamos a centigrado,
dibujamos el p.f. en amarillo
plot(vectordetiempos,T_amb(vectordetiempos)-273,'k','LineWidth',2);
legend('T.radiador','pf T.radiador','T.Ambiente','location','best')
ylabel('temperatura (centigrado)')
xlabel('tiempo (s)')
grid on
hold off
```



Opciones para la simulacion

tiempo final y opciones de tolerancia para simulador:

```
Tiempofinal=500;
opts=odeset('RelTol',1e-5,'AbsTol',0.01,'MaxStep',10,'InitialStep',4
);
```

Modelo para simulación (entradas sustituidas)

Debemos expresarla adecuadamente para ***ode45*** (será el argumento *odefun* en |doc ode45|)

donde los argumentos sean el tiempo y las variables a resolver.

Para ello, sustituimos los argumentos genéricos de entradas Trad y Tamb en [dT_dt] por las funciones del tiempo particulares ya definidas.

```
dT_dt_odefun=@(tiempo, T) ...
    dT_dt(T_rad(tiempo), T_amb(tiempo), T);
```

Cálculo de temperatura T en equilibrio en el punto de funcionamiento

Usamos *fsolve(f,valordepruebainicial)*, que obtiene numéricamente una solucion de **f(x)=0**.

Decimos que busque una solución *entre medio*, por ejemplo, próxima a 600 Kelvin.

```
% 600 K es "estimacion inicial de la solucion"
Tp_eq=fsolve(@(T_eq) dT_dt(T_radpf, T_ambpf, T_eq),600)
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
Tp_eq = 744.7630
```

Pasamos a centigrado el resultado en equilibrio:

```
T_eq_centigrado=Tp_eq-273
```

```
T_eq_centigrado = 471.7630
```

Simulación

Escogemos, arbitrariamente, un estado inicial de temperatura de la pieza, en Kelvin

```
Temp_inicial=273+100;
```

Llamamos al integrador numérico (cronometramos su ejecución, con tic-toc):

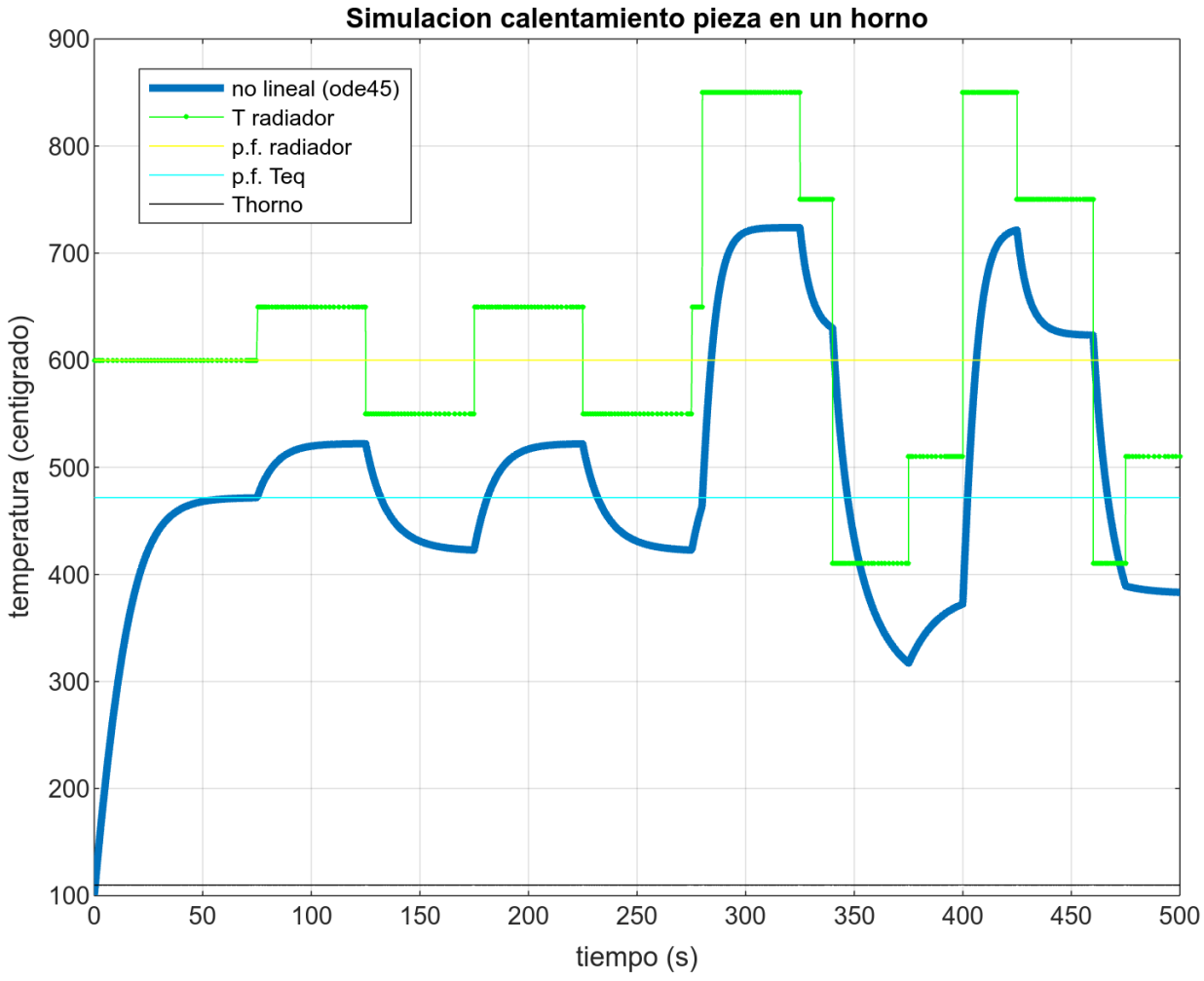
```
tic
[tiempos,T_simulada]=ode45(dT_dt_odefun, [0 Tiempofinal],
Temp_inicial, opts);
```

```
toc
```

```
Elapsed time is 0.037338 seconds.
```

Graficar resultados de simulación

```
plot(tiempos,T_simulada-273,'LineWidth',3) % cambiamos de Kelvin a
Centigrado
hold on
plot(tiempos,T_rad(tiempos)-273,'g.-') %verde
plot([0 Tiempofinal],[T_radpf,T_radpf]-273,'y'); %amarillo
plot([0 Tiempofinal],[T_eq_centigrado,T_eq_centigrado],'c'); %azul
celeste
plot(tiempos,T_amb(tiempos)-273,'k'); %negro
legend('no lineal (ode45)','T radiador','p.f. radiador' ,'p.f.
Teq','Thorno','location','best')
ylabel('temperatura (centigrado)')
xlabel('tiempo (s)')
title('Simulacion calentamiento pieza en un horno')
grid on
```



Comprobamos que cuando Trad es constante, llega al equilibrio calculado. (t=75 s)

Linealizar alrededor de ese punto de equilibrio:

Variables pasan a incrementales. Modelado normalizado linealizado, derivadas parciales:

$$\frac{d\Delta T}{dt} = \frac{1}{MC_e}\left(h * (\Delta T_{amb} - \Delta T) + k * (4T^3_{rad,pf})\Delta T_{rad} - (k + k_2) * (4T^3_{pf})\Delta T\right)$$

```
dincTpieza_dt=@(incTrad,incTamb,incT) ...
     1/MCe*(k*((4*T_radpf^3)*incTrad - (4*Tp_eq^3)*incT)-
k2*(4*Tp_eq^3)*incT + h*(incTamb-incT));
```

Simulación del modelo linealizado aproximado

Sustituimos los valores adecuados de los incrementos de entrada para tener el modelo a simular, como funcion del tiempo y de la incognita `incTpieza`

```
incT_rad=@(t) T_rad(t)-T_radpf;
incT_amb=@(t) T_amb(t)-T_ambpf;
dincT_dt_odefun=@( t, incT )  dincTpieza_dt( incT_rad(t),
incT_amb(t), incT );
```

y tambien expresamos la condición inicial en incrementos:

```
incT_inicial = Temp_inicial - Tp_eq;
```

Llamamos al integrador numérico, que obtiene incTpieza(t):

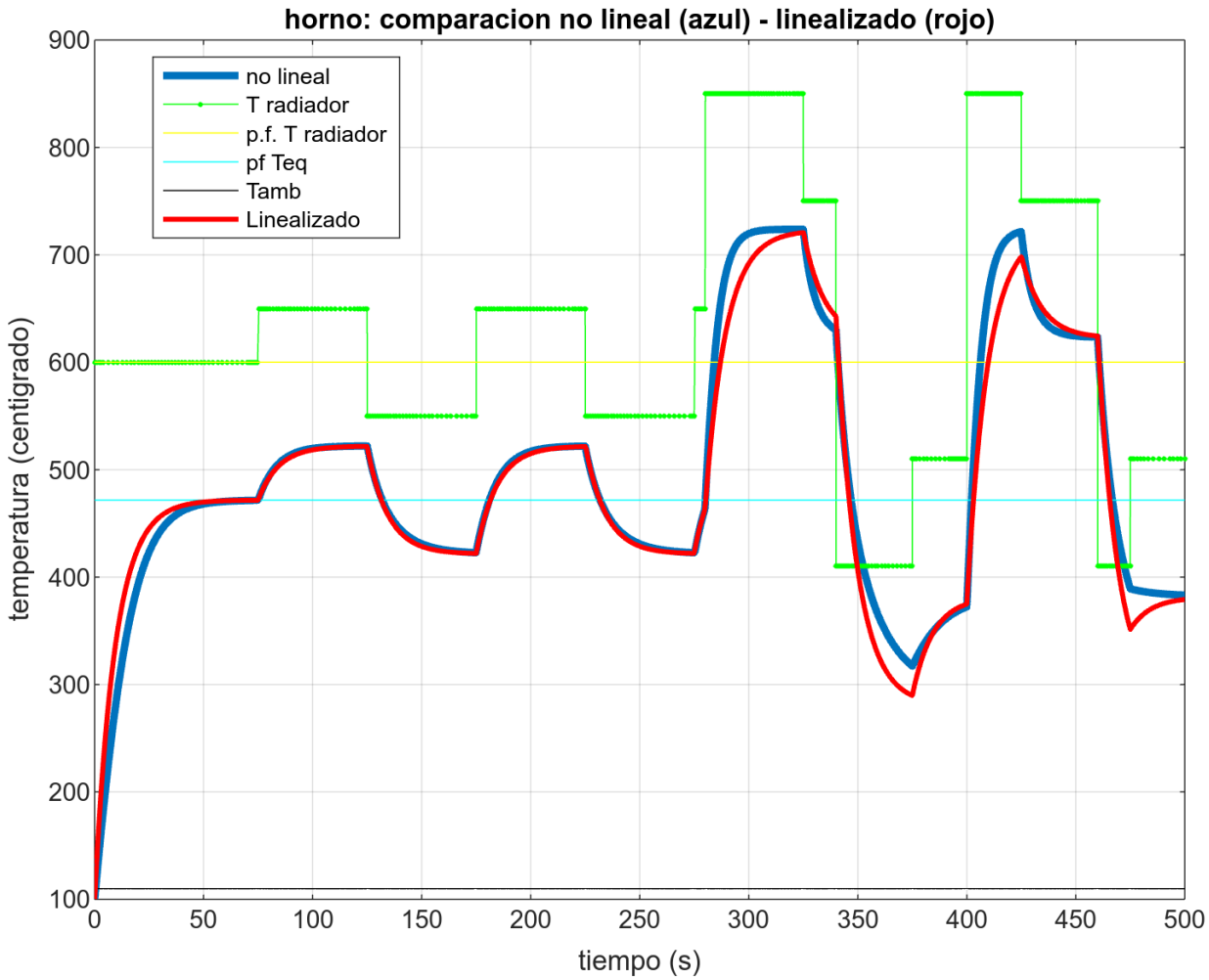
```
[tiempos_lineal,incT_simlineal] = ode45(dincT_dt_odefun, [0
Tiempofinal], incT_inicial, opts);
```

deshacer cambio de variable incremental para comparar soluciones en unidades absolutas:

```
T_lineal=incT_simlineal + Tp_eq;
```

ahora graficamos resultados (superponer simulación lineal a la gráfica de la simulación no lineal):

```
hold on
plot(tiempos_lineal,T_lineal-273,'r','LineWidth',2); %rojo
legend('no lineal','T radiador','p.f. T radiador','pf
Teq','Tamb','Linealizado','Location','best')
title('horno: comparacion no lineal (azul) - linealizado (rojo)')
hold off
```


horno: comparacion no lineal (azul) - linealizado (rojo)
no lineal
T radiador
p.f. T radiador
pf Teq
Tamb
Linealizado
temperatura (centigrado)
tiempo (s)
900
800
700
600
500
400
300
200
100
0
50
100
150
200
250
300
350
400
450
500