Prácticas con wxMaxima: cálculo diferencial de funciones de una variable

BENITO J. GONZÁLEZ RODRÍGUEZ (bjglez@ull.es)
DOMINGO HERNÁNDEZ ABREU (dhabreu@ull.es)
MATEO M. JIMÉNEZ PAIZ (mjimenez@ull.es)
M. ISABEL MARRERO RODRÍGUEZ (imarrero@ull.es)
ALEJANDRO SANABRIA GARCÍA (asgarcia@ull.es)

Departamento de Análisis Matemático
Universidad de La Laguna

Índice

Universidad de La Laguna

universia red de universidades, red de oportunidades

OPEN COURSEWARE CONSORTIUM

CC BY NC SA

PRÁCTICA 2: CÁLCULO DIFERENCIAL DE FUNCIONES DE UNA VARIABLE.

Duración estimada: 4 horas.

Objetivo: trabajar algunos conceptos relacionados con el cálculo diferencial de funciones de una variable (cálculo de límites, derivadas, recta tangente, puntos críticos, etc.).

1 Dominio y recorrido de una función.

Estudiemos el dominio y recorrido de las siguientes funciones polinómicas.

Usaremos el paquete "draw" () para hacer las representaciones gráficas, pues permite más recursos que las órdenes plot. En particular, consideraremos instrucciones del tipo "draw2d(opciones, objeto gráfico,...)".

```
--> load(draw);

--> grafica:explicit(x^3-6*x^2-3*x-1,x,-2,6);

--> wxdraw2d(color="cyan", title="Polinomio de grado tres", key="y=x^3-6*x^2-3*x-1",grafica);

--> grafica:explicit(x^4+7*x^3-5*x^2-7*x+2,x,-8,3);

--> wxdraw2d(color="blue",title="Polinomio de grado cuatro", key="y=x^4+7*x^3-5*x^2-7*x+2",grafica);

--> grafica:explicit(x^6+3*x^5-41*x^4-87*x^3+400*x^2+444*x-720,x,-7,6);

--> wxdraw2d(color="red",title="Polinomio de grado seis", grafica);
```

Vemos que en general puede ser algo complicado determinar la imagen de una función. Para ello habrá que recurrir a sus máximos y mínimos relativos.

Estudiemos ahora el dominio y recorrido de las siguientes funciones exponenciales. Para ello, primero tendremos en cuenta que una función se declara en wxmaxima con la sentencia "f(x):=".

```
--> f(x):=2^x$ wxdraw2d(explicit(f(x),x,-10,10));

--> f(x):=(1/2)^x$ wxdraw2d(explicit(f(x),x,-10,10));

--> f(x):=(-2)^x; wxdraw2d(explicit(f(x),x,-1,1));

--> f(-1/2); f(1/4);
```

Observamos que la función f(x)=(-2)^x no es una función valuada real (f(x) toma valores complejos cuando x no es un número entero). Sin embargo, por defecto, el paquete draw2d representa su parte real, mientras que con la orden plot2d se reconoce el carácter no real de la función.

```
--> wxdraw2d(draw_realpart = false,explicit(f(x),x,-1,1));

--> wxplot2d(f(x),[x,-1,1]);
```

Determinar el dominio de la función racional f(x)=((2x+3)*(x-1))/((x^2-3x+2)*(x^2+1)).

```
--> solve((x^2-3*x+2)*(x^2+1));
```

```
--> grafica:explicit(((2*x+3)*(x-1))/((x^2-3*x+2)*(x^2+1)),x,-5,5);

--> wxplot2d(((2*x+3)*(x-1))/((x^2-3*x+2)*(x^2+1)),[x,-5,5],[y,-100,100],[nticks,1000]);

--> wxdraw2d(nticks=1000,yrange=[-100,100],grafica);
```

Si queremos representar la asintota vertical en x=2 podemos crear un objeto en paramétricas formado por puntos de la forma "(2,y)".

```
--> asintota:parametric(2,y,y,-100,100);

--> wxdraw2d(grid=true,nticks=1000,yrange=[-100,100],color="blue",grafica,color="red",asintota);
```

Consideremos ahora algunas funciones logarítmicas.

```
--> f(x):=log(x)/log(10); g(x):=log(x)/log(1/10);

--> f(10); g(10); log(exp(1));

--> objeto1:explicit(f(x),x,0,4)$
    objeto2:explicit(g(x),x,0,4)$
    objeto3:explicit(log(x),x,0,4)$
    wxdraw2d(color="red",key="log10(x)",objeto1,color="green",key="ln(x)",objeto3);
    wxdraw2d(color="blue",key="log_{1/10}(x)",objeto2,color="green",key="ln(x)",objeto3);

--> wxdraw2d(explicit(log((x^4+1)/(x^2+1)),x,-2,2));

--> f(x):=log(x^6+3*x^5-41*x^4-87*x^3+400*x^2+444*x-720);

--> wxdraw2d(explicit(f(x),x,-15,15));

--> wxplot2d(f(x),[x,-15,15]);
```

A la hora de representar funciones logarítmicas con la orden draw2d debemos tener cuidado de evitar valores no reales.

```
--> f(0);
```

Estudiemos el dominio de la función logarítmica definida anteriormente. Para resolver inecuaciones, haremos uso del paquete "fourier_elim".

```
--> load(fourier_elim);

--> fourier_elim([x^6+3*x^5-41*x^4-87*x^3+400*x^2+444*x-720<=0],[x]);

--> wxdraw2d(draw_realpart=false,explicit(log(x^6+3*x^5-41*x^4-87*x^3+400*x^2+444*x-720),x,-15,15));
```

Continuamos con algunas funciones trigonométricas.

```
--> wxplot2d([sin(4*x),sin(2*x)],[x,-%pi,%pi],[color,red,green]);

--> wxplot2d(cos(2*x-%pi/2)^3+2*sin(x/2),[x,0,%pi]);
```

Podemos crear también gráficas animadas con la orden "with_slider" (asociada a "plot2d") o con "with_slider_draw" (asociada a "draw2d").

En las sentencias que presentamos a continuación "A" es un parámetro que toma valores en una cierta lista que creamos al efecto con la sentencia "makelist".

```
--> with_slider(A,makelist(i,i,1,10),sin(A*x),[x,-%pi,%pi],[y,-1.1,1.1]);
```

```
--> with_slider_draw(A,makelist(i,i,1,10),yrange=[-1.1,1.1],explicit(sin(A*x),x,-%pi,%pi));

--> with_slider(A,makelist(%pi/5*i,i,0,20),A*cos(A+x^2),[x,-%pi,%pi],[y,-4*%pi,4*%pi]);

--> with_slider_draw(A,makelist(%pi/5*i,i,0,20),yrange=[-4*%pi,4*%pi],explicit(A*cos(A+x^2),x,-%pi,%pi));
```

Observemos la diferencia que producen las dos siguientes sentencias, donde en la segunda se limita el rango de "y".

```
--> wxplot2d(tan(x),[x,-2*%pi,2*%pi]); wxplot2d(tan(x),[x,-2*%pi,2*%pi],[y,-4,4]);

--> with_slider_draw(A,makelist(%pi/10*i,i,0,10),yrange=[-10,10],explicit(tan(x+A)/(x+A),x,-2*%pi,2*%pi));
```

2 Obtención de la función inversa.

```
--> f(x):=(4-4*x)/(5*x+4);

--> solve(f(y)=x,y);

--> part(solve(f(y)=x,y),1);

--> invf(x):=rhs(part(solve(f(y)=x,y),1));

--> wxplot2d([x,f(x),invf(x)],[x,-10,10],[y,-10,10]);
```

En este caso, f(x) y su inversa coinciden.

```
--> ratsimp(f(x)-invf(x));

--> f(x):=1+log(x+2); invf(x):=rhs(part(solve(f(y)=x,y),1));

--> invf(x);

--> wxplot2d([x,f(x),invf(x)],[x,-2,10],[y,-5,10],[nticks,1000],[gnuplot_preamble, "set size ratio 1"]);

--> f(x):=cos(x); invf(x):=rhs(part(solve(f(y)=x,y),1));

--> invf(x);

--> wxplot2d([x,f(x),invf(x)],[x,-%pi,%pi],[y,-%pi,%pi],[gnuplot_preamble, "set size ratio 1"]);

--> f(x):=csc(x); invf(x):=rhs(part(solve(f(y)=x,y),1));

--> trigsimp(f(x)-1/sin(x));

--> invf(x);

--> wxplot2d([x,f(x),invf(x)],[x,-40,40],[y,-40,40],[nticks,1000],[gnuplot_preamble, "set size ratio 1"]);

--> wxplot2d([x,f(x),invf(x)],[x,-%pi,%pi],[y,-10,10],[nticks,1000],[gnuplot_preamble, "set size ratio 1"]);
```

3 Límite de una función en un punto.

Para el cálculo de límites basta usar la sentencia "limit". Podemos calcular límites laterales introduciendo las opciones "minus" y "plus", para indicar límite por la izquierda o derecha, respectivamente.

```
--> f(x):=(x^3-8)/(x-2);
```

```
--> limit(f(x),x,2);

--> limit(f(x),x,2,minus);limit(f(x),x,2,plus);
```

Veamos cómo representar la discontinuidad evitable en x=2.

```
--> punto:[[2,12]]$ /* Creamos una lista con un único punto */
    wxplot2d([[discrete,punto],f(x)],[x,-5,3],[style, points, lines], [color, red, green],[point_type, circle],[legend,"(2,12)","(x^3-8)/(x-2)"]);

--> ratsimp(f(x));
```

Podemos además calcular límites en el infinito con los valores "inf" (+infinito) o "minf" (-infinito).

```
--> f(x):=(x^2+1)/(x^2+x+1);

--> limit(f(x),x,inf); limit(f(x),x,minf);

--> wxplot2d(f(x),[x,0,20]);
```

Escribiendo una comilla simple antes de la instrucción "limit" logramos representar el límite en modo simbólico.

```
--> f(x):=1/(1-x)-3/(1-x)^3;

--> 'limit(1/(1-x)-3/(1-x)^3,x,1)=limit(1/(1-x)-3/(1-x)^3,x,1);

--> 'limit(1/(1-x)-3/(1-x)^3,x,1,minus)=limit(1/(1-x)-3/(1-x)^3,x,1,minus);

--> 'limit(1/(1-x)-3/(1-x)^3,x,1,plus)=limit(1/(1-x)-3/(1-x)^3,x,1,plus);

--> ratsimp(f(x));

--> wxplot2d(f(x),[x,-10,10],[y,-20,20]);
```

Podemos representar además la asíntota vertical en x=1 del modo siguiente.

```
--> wxplot2d([[parametric, 1,y,[y,-20,20]],f(x)],[x,-10,10],[y,-20,20],[style, lines, lines], [color, red, blue],[legend,"x=1","f(x)"]);

--> f(x):=sqrt(x^2-5*x+6)-x$
    limit(f(x),x,inf);
    wxplot2d(f(x),[x,0,10],[y,-5,10]);

--> fourier_elim([x^2-5*x+6<0],[x]);

--> f(x):=sqrt(x+16)-sqrt(x)$
    limit(f(x),x,inf);
    wxplot2d(f(x),[x,-100,1000]);

--> limit(f(x),x,0,plus);

--> limit(f(x),x,0,minus);
```

Observar que el límite anterior se obtiene considerando aritmética compleja, dado que el dominio de la función es x>=0.

```
--> limit(sqrt(x),x,0,minus);

--> limit(sqrt(x-2),x,2);

--> limit(log(x-2),x,1);
```

```
--> f(x):=(x^3+2)/x$ limit(f(x),x,0,minus); limit(f(x),x,0,plus);

--> wxdraw2d(explicit(f(x),x,-5,5),yrange=[-100,100],line_width=5,grid=true);
```

4 Derivada de una función de una variable.

Calculemos en primer lugar algunas derivadas elementales por medio de la definición a través de un paso al límite.

```
--> kill(all);

--> f(x):=x^3$ limit((f(x+h)-f(x))/h,h,0);

--> f(x):=cos(x)$ limit((f(x+h)-f(x))/h,h,0);

--> f(x):=sin(x)$ limit((f(x+h)-f(x))/h,h,0);

--> f(x):=log(x)$ limit((f(x+h)-f(x))/h,h,0);
```

En este caso, wxMaxima tiene problemas para calcular la derivada, si no se le fuerza a operar los logaritmos.

```
--> limit(logcontract((f(x+h)-f(x))/h),h,0);

--> f(x):=sin(x)/cos(x)$ limit((f(x+h)-f(x))/h,h,0);

--> trigsimp(%);
```

Para derivar directamente una función real de variable real podemos usar directamente la orden "diff". Esta orden actúa tanto con funciones como con expresiones.

```
--> f(x):=x^4-3*x^3+2$ diff(f(x),x);

--> expr:x^4-3*x^3+2$ diff(expr,x);

--> 'diff(tan(x),x)=diff(tan(x),x);

--> trigsimp(%);

--> f(x):=atan(sqrt((1-x)/(1+x)))$ diff(f(x),x); ratsimp(%); radcan(%);

--> f(x):=(sin(x)+cos(x))/(sin(x)-cos(x))$ trigsimp(diff(f(x),x));

--> diff((1+1/x)^x,x); radcan(%);
```

Véase que la orden "diff" también permite trabajar con derivadas en modo simbólico.

```
--> 'diff(a(x)*b(x),x)=diff(a(x)*b(x),x);

--> 'diff(a(x)/b(x),x)=ratsimp(diff(a(x)/b(x),x));
```

Si además queremos hallar el valor de la derivada en un punto concreto, basta combinar la orden "diff" con la orden "ev" (o con la orden "subst").

```
--> expr: x^3+cos(x); diff(expr,x); ev(%,x=2);

--> f(x):=(sin(x)+cos(x))/(sin(x)-cos(x)); trigsimp(diff(f(x),x)); subst(x=3*%pi/4,%);
```

No obstante lo anterior, a efectos de trabajar con la función derivada, puede ser más recomendable definirla como función, para luego

evaluarla o representarla gráficamente como tal. Observemos que esto no debe hacerse como una definición directa, sino a través de la orden "define".

```
--> f(x):= x^3+cos(x); df(x):=diff(expr,x);

--> df(2);

--> define(df(x),diff(expr,x));

--> df(2); wxplot2d(df(x),[x,-%pi,%pi]);
```

Notamos además que la orden "diff" también se puede emplear directamente para obtener derivadas de orden superior, sin más que añadir a la sentencia el orden de derivación.

```
--> f(x):=sqrt(x); diff(f(x),x,4);

--> define(df(x),diff(f(x),x,4));

--> limit(df(x),x,0); limit(df(x),x,0,plus);

--> expr:x^2*%e^x; diff(expr,x,n); diff(expr,x,20); subst(x=0,%);

--> define(df(x),diff(expr,x,20)); df(0);
```

La orden "diff" también permite derivar funciones definidas implícitamente. Sin embargo, para ello habremos de combinar la instrucción "diff" con la orden "depends".

```
--> depends(y,x); diff(y^2+x^2=1,x);
```

Si queremos despejar la expresión de y'(x) usaremos entonces la orden "solve".

```
--> solve(%,diff(y,x));

--> dependencies; remove(y,dependency);

--> depends(x,y); diff(y^2+x^2=1,y); solve(%,diff(x,y)); remove(x,dependency);
```

5 Rectas tangente y normal a una curva en un punto.

```
--> kill(all);

--> f(x):=x^3+2*x^2-4*x-3;

--> define(df(x),diff(f(x),x));

--> tangente(x,a):=f(a)+df(a)*(x-a);

--> wxplot2d([f(x),tangente(x,2)],[x,1.5,3.5],[nticks,1000]);

--> f(x):=cos(x)$
    define(df(x),diff(f(x),x))$
    wxplot2d([f(x),tangente(x,%pi/2)],[x,0,%pi],[nticks,1000],[gnuplot_preamble, "set size ratio 1"]);
```

Podemos visualizar la recta tangente como límite de rectas secantes creando una animación.

```
--> f(x):=x^3+2*x^2-4*x-3$ define(df(x),diff(f(x),x))$

--> secantes(x,a,b):=f(a)+((f(b)-f(a))/(b-a))*(x-a);
```

```
--> with_slider(b,makelist(3-i/20,i,0,19),[f(x),tangente(x,2),secantes(x,2,b),[discrete,[[2,f(2)],[b,f(b)]]]],[x,1.5,3.5],[y,-10,60],
[style,lines,lines,lines,points],[point_type, circle],[legend,false]);

--> f(x):=x*(x^2-1); define(df(x),diff(f(x),x));

--> wxplot2d([f(x),tangente(x,0)],[x,-1,1],[nticks,1000],[gnuplot_preamble, "set size ratio 1"]);

--> with_slider(b,makelist(1-i/20,i,0,19),[f(x),tangente(x,0),secantes(x,0,b),[discrete,[[0,f(0)],[b,f(b)]]]],[x,-1,1],[y,-1,1],
[style,lines,lines,lines,points],[point_type, circle],[legend,false],[gnuplot_preamble, "set size ratio 1"]);
```

Veamos la evolución de las rectas secantes trazadas a derecha e izquierda de x=0.

```
--> with_slider(b,makelist(1-i/20,i,0,19),[f(x),tangente(x,0),secantes(x,0,b),secantes(x,-b,0),[discrete,[[-b,f(-b)],[0,f(0)],[b,f(b)]]]],[x,-1,1],
[y,-1,1],[style,lines,lines,lines,lines,points],[point_type, circle],[legend,false],[gnuplot_preamble, "set size ratio 1"]);
```

En este ejemplo anterior, las tangentes trazadas simétricamente a izquierda y derecha coinciden por la simetría de la función.

```
--> f(x):=15*x*(x-1)*(x-1/2)*(x+2/3)*(x+1); ratsimp(define(df(x),diff(f(x),x)));

--> wxplot2d([f(x),tangente(x,0)],[x,-1,1],[nticks,1000]);

--> with_slider(b,makelist(1-i/20,i,0,19),[f(x),tangente(x,0),secantes(x,0,b),secantes(x,-b,0),[discrete,[[-b,f(-b)],[0,f(0)],[b,f(b)]]]],[x,-1,1],
[y,-6,6],[style,lines,lines,lines,lines,points],[point_type, circle],[legend,false]);

--> f(x):=1/2+15*x^2*(x-1)*(x-2/5)*(x+2/3)*(x+1); ratsimp(define(df(x),diff(f(x),x)));

--> wxplot2d([f(x),tangente(x,0)],[x,-1,1],[nticks,1000]);

--> with_slider(b,makelist(1-i/20,i,0,19),[f(x),tangente(x,0),secantes(x,0,b),secantes(x,-b,0),[discrete,[[-b,f(-b)],[0,f(0)],[b,f(b)]]]],[x,-1,1],
[y,-2,2],[style,lines,lines,lines,lines,points],[point_type, circle],[legend,false]);
```

Podemos, además, introducir rectas normales de modo análogo.

```
--> normaltang(x,a):=f(a)-(1/df(a))*(x-a);

--> normalsec(x,a,b):=f(a)-((b-a)/(f(b)-f(a)))*(x-a);

--> f(x):=x*(x^2-1); define(df(x),diff(f(x),x));
```

No obstante, en las siguientes instrucciones debemos evitar posibles divisiones por cero en el cálculo de las pendientes de las rectas normales. Además, hemos impuesto la restricción de igual medida en los ejes a efectos de apreciar la ortogonalidad de las rectas correspondientes.

```
--> wxplot2d([f(x),tangente(x,0),normaltang(x,0)],[x,-1,1],[nticks,1000],[gnuplot_preamble, "set size ratio 1"]);

--> with_slider(b,makelist(1-i/20,i,1,19),[f(x),tangente(x,0),normaltang(x,0),secantes(x,0,b),normalsec(x,0,b),[discrete,[[0,f(0)],[b,f(b)]]]],
[x,-1,1],[y,-1,1],[style,lines,lines,lines,lines,lines,points],[point_type, circle],[legend,false],[gnuplot_preamble, "set size ratio 1"]);
```

6 Puntos críticos, extremos y puntos de inflexión.

Para concluir esta segunda práctica, aplicamos algunas de las instrucciones arriba estudiadas a la clasificación de puntos críticos de funciones de una variable.

```
--> f(x):=x^3-7*x^2+5*x+1;

--> solve(f(x)=0,x) /* Ceros de la función */;

--> %,numer;
```

```
--> define(df(x),diff(f(x),x));

--> critpoints:solve(df(x)=0,x) /* Puntos críticos */;

--> %,numer;

--> define(d2f(x),diff(df(x),x));

--> solve(d2f(x)=0,x) /* Puntos de inflexión */;

--> %, numer;

--> subst(critpoints[1],d2f(x)),numer /* El primer punto crítico se trata de un máximo relativo */;

--> subst(critpoints[2],d2f(x)),numer /* El segundo punto crítico se trata de un mínimo relativo */;

--> wxplot2d([f(x),df(x),d2f(x)],[x,-1,7]);

--> f(x):=x^4*(4-x^2); solve(f(x)=0,x);

--> define(df(x),diff(f(x),x)); critpoints:solve(df(x)=0,x); %,numer;

--> define(d2f(x),diff(df(x),x)); solve(d2f(x)=0,x); %,numer;

--> subst(critpoints[1],d2f(x)); subst(critpoints[2],d2f(x)); subst(critpoints[3],d2f(x)) /* Los dos primeros puntos críticos resultan ser
    máximos relativos */;

--> subst(critpoints[3],diff(f(x),x,3));
    subst(critpoints[3],diff(f(x),x,4)) /* El punto crítico x=0 es mínimo relativo */;

--> wxplot2d(f(x),[x,-2,2]);

--> wxplot2d(df(x),[x,-2,2]);

--> wxplot2d(d2f(x),[x,-2,2]);

--> f(x):=sin(1/x);

--> limit(f(x),x,0) /* El resultado "ind" denota un límite indeterminado pero acotado */;
```

Cargamos el paquete "to_poly_solve" para determinar todas las raíces de f(x).

```
--> load(to_poly_solve)$

--> nicedummies(ratsimp(%solve(f(x)=0,x)));

--> wxplot2d(f(x),[x,1/(10*%pi),1],[y,-1.1,1.1],[nticks,1000]);

--> define(df(x),diff(f(x),x));

--> limit(df(x),x,0) /* El resultado "und" denota un límite indefinido */;

--> nicedummies(ratsimp(%solve(df(x)=0,x)));
```

wxMaxima tiene problemas incluso para resolver completamente la ecuación cos(x)=0, aunque podemos tener en cuenta que cos(x-%pi/2)=sin(x).

```
--> wxplot2d(df(x),[x,1/(10*%pi),1/4],[nticks,1000]);
```

```
--> define(d2f(x),ratsimp(diff(df(x),x)));
```

```
--> limit(d2f(x),x,0);
```

```
--> nicedummies(ratsimp(%solve(d2f(x)=0,x)));
```

En este caso, hallar los puntos de inflexión involucra resolver una ecuación trascendente. Podemos aproximar algunos puntos de inflexión con la orden "find_root".

```
--> find_root(d2f(x),x,0.5,4);
```

```
--> find_root(d2f(x),x,0.25,4);
```

```
--> wxplot2d(d2f(x),[x,0.75,10],[nticks,1000]);
```

```
--> wxplot2d(d2f(x),[x,1/(10*%pi),1/4],[nticks,1000]);
```

Created with wxMaxima.