CALCULUS - PRACTICAL II - ELEMENTARY CALCULUS

PEDRO FORTUNY AYUSO

The students will have already received the lessons about limits, continuity and derivation although these concepts should not be new for them at all.

Each section is intended to be performed in a single file (a `.m` file) and saved by the student for later reference or consultation.

1. Computing limits

Recall that in order to compute limits, one needs to declare the “variable” in the expression as symbolic, otherwise an error will take place.

In order to get some help, one can use either the `help` or the `doc` command.

Finally, the command for computing limits in Matlab has the following syntax:

```
  > limit(expr, var, point, 'left'/'right')
```

where `expr` is a symbolic expression, `var` is the variable with respect to which the limit is to be computed, `point` is the point at which the limit is computed and either `'left'` or `'right'` (which may be omitted if the limit is in both directions) indicates the direction of the limit.

1.1. **Drill.** Compute the following limits:

$$1)\ \lim_{x\to\infty}\frac{x^3-8}{\sqrt{6x^6-x^3+1}}$$

$$2)\ \lim_{x\to 0}(e^x-1)\log(x+1)$$

$$3)\ \lim_{x\to\pi}\sin(x)^{x-\pi}$$

$$4)\ \lim_{x\to 0^-}|x|^{|x|}$$

$$5)\ \lim_{x\to 0^+}x^{\sin(x)}$$

$$6)\ \lim_{n\to\infty}(\frac{n}{n+1}-\frac{2}{n-1})^n$$

2. Function definition and plotting

We shall use anonymous functions for plotting graphs. For example, in order to plot the function $f(u)=\sin(u)-\frac{1}{1+u^2}$ with green lines, from -4 to 2 using 500 points, we might use the following sequence of commands:

```
  > f = @(t) sin(t) - 1./(1+t.^2); % att!:  parentheses and dots (‘.’)
  > x = linspace(-4, 2, 500); % prepare the points
  > plot(x, f(x), '-g');
```

In order to plot *on the same figure* the function $\cos(t)$, one needs to tell Matlab “not to clear” the graphical window:

Date: October 9, 2013.

```
> hold on % do not clear the window
> plot(x, cos(x), '*r'); % use red stars
```

3. Derivatives, tangent line, etc...

The equation of the tangent line to the graph of $f(x)$, derivable at a is given by:

$$y = f'(a)(x-a) + f(a),$$

and the equation of the *normal line* at the same place is

$$y = -\frac{1}{f'(a)}(x-a) + f(a).$$

Given the anonymous function `P=@(x) x.^2-2*x+1`, for example, one can compute the symbolic derivative using `syms`. First of all, one needs a symbolic variable:

```
> syms u
```

and now one can compute the derivative, evaluating `P` on the symbolic variable and differentiating:

```
> diff(P(u), u)
```

The value returned by this operation is a symbolic expresssion *which cannot be evaluated*. If one wishes to use the result (to plot it, for example), one needs to transform it into an anonymous function:

```
> P = @(x) x.^2 - 2*x + 1
> syms u
> Q = diff(P(u), u)
> Q(3) % error:  symbolic expression cannot be evaluated
> Q_n = matlabFunction(Q) % define Q_n as anonymous
> Q_n(3) % can be evaluated
```

3.1. **Examples.**

In order to determine if the following function

$$f(x) = \begin{cases} x\sin(1/x) & \text{if } x > 0 \\ 0 \ \text{ if } x \leq 0 \end{cases}$$

is continuous at 0, one needs to compute the left and right limits at 0 and compare both with the value of f at 0.

```
> clear all; syms x;
> l_r = limit(0, x, 0, 'left');
> l_l = limit(x*sin(x), x, 0, 'right');
> l_r == l_l % do they match?
```

The three of them match in this case, so f is continuous at 0. Is it differentiable? Let us compute

$$\lim_{h\to 0+} \frac{f(0+h)-f(0)}{h} \quad \text{and} \quad \lim_{h\to 0+} \frac{f(0+h)-f(0)}{-h}$$

and compare them. Notice that both limits are *on the right* because $h > 0$ and on one limit it is added, on the other, it is subtracted.

```
> d_l = limit(0/-h,h,0,'right');
> d_r = limit((h*sin(1/h)-0)/h,h,0,'right');
> d_r == d_l % do they match?
```

The second limit does not exist (`d_r`), so that f is not differentiable at 0.

3.2. **Solving equations.** Solving symbolic equations (for example, to compute the zeroes of the derivative) is performed using `solve`. As in all symbolic operations, a symbolic variable (declared with `syms`) is needed. The command `solve` returns a *matrix* of values, on each row a solution (either exact, if possible, or approximate) of the equation. For example,

```
> clear all; syms x
> roots = solve(x.^2 -2*x +1,x) % there are two 1's
> more_r = solve(x.^6-2*x^2-1,x) % exact values
> another = solve(log(x) - exp(x), x) % approx.  values
```

4. Exercises

Each exercise will be solved in a single file. If an exercise has several sections, separate them with a comment (`%`).

Exercise 1. Compute the following limits:

$$\lim_{x\to 0}\frac{\log(\tan(7x))}{\log(\tan(2x))}$$

$$\lim_{x\to 1}\left(\frac{x}{x-1}-\frac{1}{\log(x)}\right)$$

$$\lim_{x\to 1} x^{\frac{1}{1-x}}$$

$$\lim_{x\to a}\frac{\sin(x)-\sin(a)}{x-a}$$

Exercise 2. Derivatives, tangents and normals. Recall `matlabFunction` for the second part.

(1) Define $f(x)=\sin(\frac{20}{1+x^2})$. Plot it from -20 to 20 using 500 points.
(2) Compute the first derivative $f'(x)$ and plot it (on the same graph as $f(x)$).
(3) Always on the same graph, plot the tangent line to $f(x)$ at $x=1$ and the normal line to $f(x)$ at $x=-1$.

Exercise 3. Compute all the local maxima and minima and the inflection points of

$$f(x)=\frac{x^3-3x^2+3x-1}{1+x^2}$$

and plot the tangent line to $f(x)$ at the inflection points in order to verify that $f(x)$ is convex on one side and concave on the other.

5. Solutions to the Exercises

Exercise 1. We shall define anonymous functions first and then compute the limits using the `limit` command. The code in Listing 1 shows a way to do this exercise.

```
% Exercise 1, practical 2.

% First of all, define the appropriate anonymous functions.
% Notice the parenthesis (this is very hard for the students)
% We shall use dotted operators even though they are not
% necessary for this exercise.

f1 = @(x) log(tan(7*x))./log(tan(2*x));

% There is no need to use 'x' to define the functions, this
% is why they are called 'anonymous' because the name of the
% variable is irrelevant.

f2 = @(t) t./(t-1) - 1./log(t);

f3 = @(t) t.^(1./(1-t));

% for the last one, we need to define 'a' as a symbol!
syms a
f4 = @(x) (sin(x) - sin(a))./(x-a);

% Computing limits is a symbolic operation, so we need a symbol
% to be used as variable.
% We might use 'x' but let us use 'z' instead.
syms z;

% Now we proceed to compute the limits.
% 1) Always assign the results
%    to variables, otherwise they are 'lost' in the run.
% 2) We do not use semicolons in the next lines because we want to
%    see the results.
l1 = limit(f1(z), z, 0)
l2 = limit(f2(z), z, 1)
l3 = limit(f3(z), z, 1)
l4 = limit(f4(z), z, a)
```

Listing 1. Code for exercise 1.

Exercise 2. The only complication in this exercise is the computation of the tangent and normal lines. The code in Listing 2 does everything.

```
% 1)
% We first define the anonymous function.
% Notice the dotted operators ./ and .^, which ARE NEEDED
% because we shall compute the value of f on a vector when
% plotting it.

f = @(x) sin(20./(1+x.^2));

% For plotting, one needs first the X-coordinates, which the
% statement specifies. Do not use 'x' or 'y' as variables because
% one gets easily confused. It is better to use 'x1', 'y1', etc.

x1 = linspace(-20, 20, 500);

% Why is it linspace?

% Plot. We use red color.
plot(x1, f(x1), 'r');

% 2)
% The derivative is a symbolic operation, we need a symbol
syms t
```

```
% Now we can compute df(t)/dt. Call it fp (for f-prime):
fp = diff(f(t), t);

% In order to plot fp, we MUST TRANSFORM IT INTO AN ANONYMOUS
% FUNCTION USING matlabFunction:
fp_n = matlabFunction(fp);

% And now we can plot it. As it is on the same graph, we have
% to tell Matlab to 'hold on'. We use 'green' color now.
hold on
plot(x1, fp_n(x1), 'g');

% 3)
% Recall that we have to use fp_n instead of fp everywhere.
%
% The Tangent Line at (a, f(a)) is
%       Y = fp_n(a).*(X-a)+f(a)
%
% While the Normal Line is
%       Y = -1./fp_n(a).*(X-a)+f(a)
%
% Both will be defined as anonymous functions. The tangent
% line is at a=1, the normal line at a=-1:

tangent = @(x) fp_n(1).*(x-1)+f(1);
normal  = @(x) -1./fp_n(-1).*(x+1)+f(-1);

% These are just functions, we can plot them.
% The tangent in black and the normal in magenta
% Notice that we do not need to 'hold on' because we have
% already done.
plot(x1, tangent(x1), 'k')
plot(x1, normal(x1), 'm')
% The magenta line is perpendicular at x=-1 to the green curve.
```

Listing 2. A solution to exercise 2.

Exercise 3. This exercise uses `solve` extensively. Local maxima and minima are the points at which the derivative is 0 (and which are not inflection points!). Inflection points are points at which the second derivative is 0 (well, not exactly but mostly). A detailed solution can be found in Listing 3.

```
% Exercise 3, practical 2.

% Define the anonymous function. Notice the dots.

f = @(x) (x.^3-3*x.^2+3*x-1)./(1+x.^2);

% We have to compute the derivative and the second derivative, this
% requires a symbol
syms x

fp  = diff(f(x), x);
fpp = diff(f(x), x, 2);

% Find the critical points (derivative = 0):
criticals = solve(fp)
% There are three 'critical points' but only one is real, x=1. So
% this is the only one. Is it a maximum, a minimum? We compute the
% value of the second derivative. To this end, we MUST TRANSFORM
% fpp INTO A NUMERICAL FUNCTION:
fpp_n = matlabFunction(fpp);
fpp_n(criticals(1))
% It turns out to be 0, which is non-informative...

% There are multiple ways to overcome this problem. For example,
% since f(x) is defined on all R (because the denominator never
% vanishes), it is either a global maximum or a global minimum or
% an inflection point and we can tell which of them it is just
% evaluating f at three points.
```

```
f(criticals(1)-1) % This is -1
f(criticals(1))   % This is 0
f(criticals(1)+1) % This is 1/5

% This means that the function is INCREASING, hence criticals(1)
% CAN ONLY BE AN INFLECTION POINT!

% Let us plot f to check this
x1 = linspace(-10, 10, 1000);
plot(x1, f(x1));
hold on
% Plot the critical point with a large red circle:
plot(criticals(1), f(criticals(1)), 'ro')
% (One sees clearly that it is an inflection point)

% Now the tangent line at the inflection point should be easy. Remember
% that the tangent line has equation
%   Y = fp(a)*(X-a)+f(a)
% We need to transform fp again into a numerical function
fp_n = matlabFunction(fp);
% There is just one critical point, we need just one equation:
tangent = @(x) fp_n(criticals(1)).*(x-criticals(1))+f(criticals(1));

% No need to hold on, already done above
plot(x1, tangent(x1), 'r')
```

Listing 3. A solution to exercise 3.