# CAD practical – programming part 1: Lagrange polynomials and transforms

*Written by C. Leblanc, updated by A. Bolyn*

## *Requirements*

- Provided Ubuntu virtual machine working.
- First CMake generated project (see previous exercise)

## *Goals*

*Coding with C++*

- Edit and run C++ project in CodeBlocks

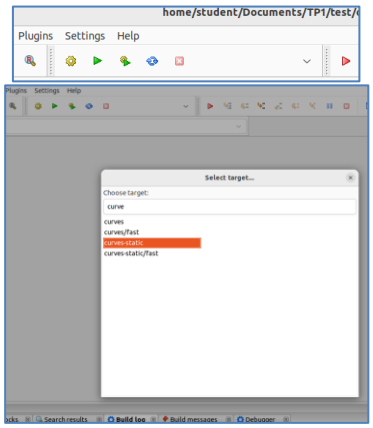
*Computational Geometry*

- Code Lagrangian curves and interpolation
- Application of Tchebychev abscissa for Lagrangian curves

---

## *CodeBlocks project*

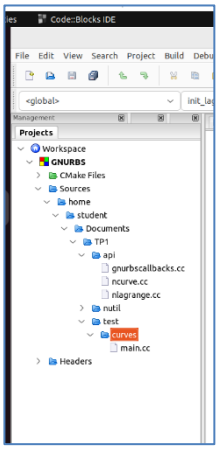
Open the CodeBlocks project generated by CMake (see previous exercise). It is called `GNURBS.cbp` and should be in your `build` directory (or other file selected in CMake for building). If CodeBlocks asks you to choose a compiler, choose the GNU GCC Compiler and set it as default.



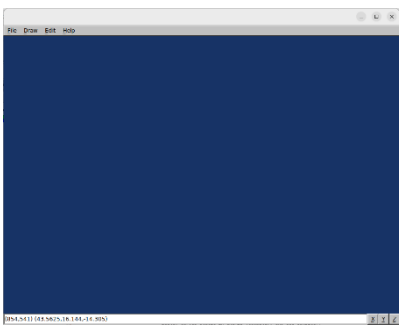

Before working, be sure to select "**curve-static**" as build target: click on the chevron on the right of the run button (as presented in the figure) and select it in the list. About these buttons in CodeBlocks (Tool buttons):

- The gear of the tool menu for compiling
- The green play button for running
- You can build and run your work with the shortcut button next to them.
- If you need to force the run to stop, use the red box with white cross inside (but normally simply exiting the windows of the executable should do it)

The outputs of the compilator are on the bottom panel. In the tab "Build log" are the log prompts: if there are warnings or errors in your code, it will be exposed there but after a compilation.

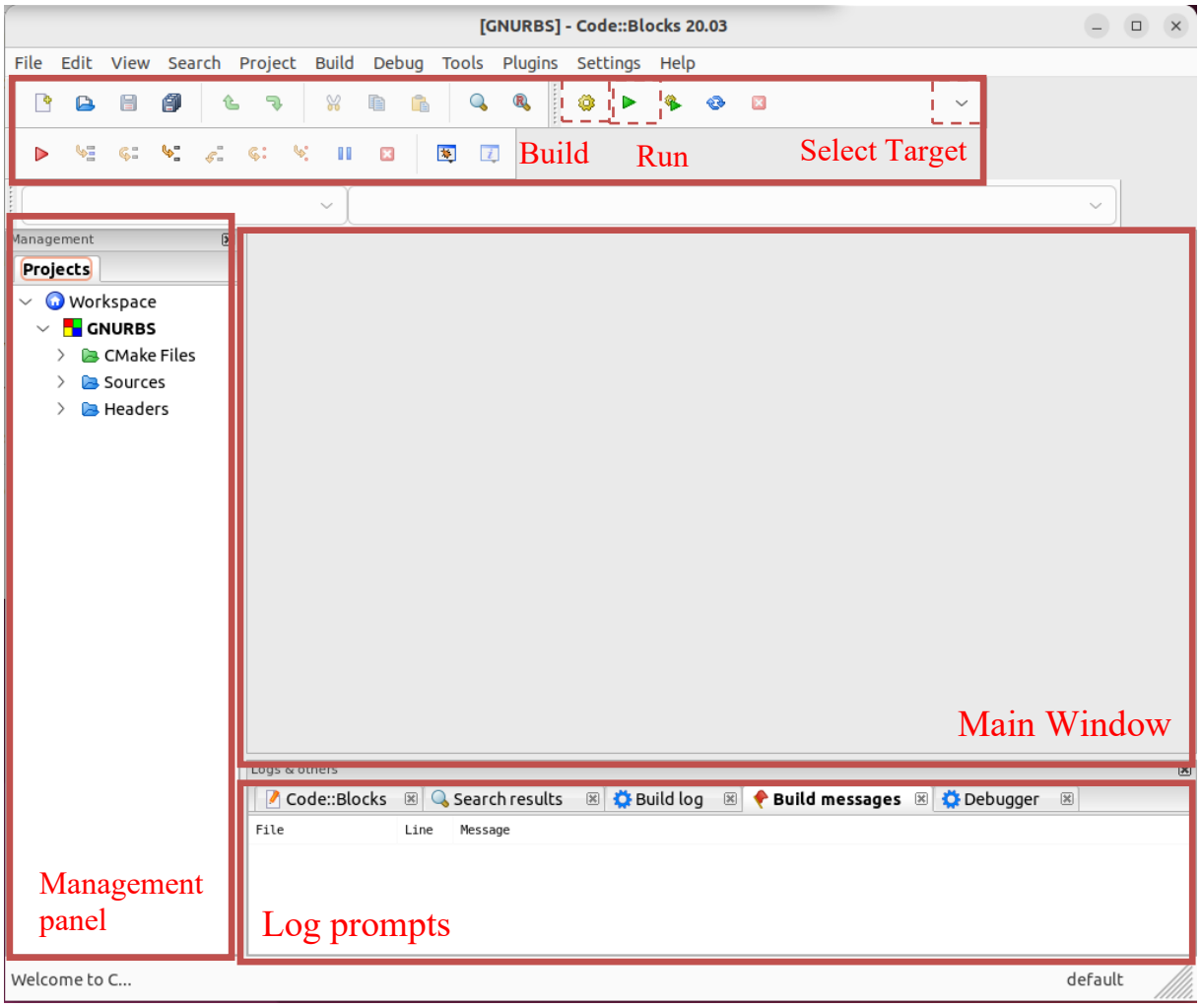



For this work, the C++ files "`api/nlagrange.cc`" and "`test/curves/main.cc`" will be edited. You can find them in the menu of the left panel (called "Management window") and double click on it to edit it in CodeBlocks.



You can already try to compile and run the code to see if the project and the compilator work. Since nothing has been coded so far you will have an empty blue window.

You can run you porogram outside of CodeBlocks. Since the executable was generated after the compilation, you can find it in your build folder and simply execute it with double clicking on it.



## *Lagrange polynomials*

For the first part of this lecture, we will implement functions to draw a semi-circle and a helix with Lagrange interpolation.

Reminder: $P(u) = \sum_{i=0}^{n-1} P_i \, l_i^{n-1}(u)$ with $l_i^{n-1}(u) = \prod_{j=0, i\neq j}^{n-1} \frac{u-u_j}{u_i-u_j}$

1. In the file "`api/nlagrange.cc`" implement the functions

   - `double nlagrange::Basis(int which, double u_) const`

     `Basis` returns the value of the ith Lagrange basis function evaluated at `u_` ($l_i^p(u)$).
   - `void nlagrange::P(double u_, npoint& ret) const`

     `P` returns by reference the point `ret` of parameter `u_` on the curve ($P(u)$).

2. In the file "`test/curves/main.cc`" (under `main()`) implement the following function to draw a semi-circle of radius 1 and centred at `(DX, DY)`.

   ```
   void init_lagrange(nlagrange &curve, double DX, double DY)
   ```

   Pay attention to:
   - Define a correct curve parameter `u`
   - Give to `curve` the control points $P_i$ and the associated parameter `u`.

3. Compile and run the code. If everything works correctly, you have a window with blue background showing your control points and the curve.

4. In the file "`test/curves/main.cc`" (under `main()`) implement the function to draw a helix of radius `r` and centred at (0,0).

   ```
   void init_lagrange_helix(nlagrange& curve, double r)
   ```

   Pay attention to:
   - Define a correct curve parameter `u`
   - Select a sufficient range of `u` to see the helix (2 periods are requested)
   - Space evenly the control points to have a good curve

   Reminder the equation of a helix of constant radius r in cartesian coordinates: $\begin{cases} x(\theta) = r\cos\theta \\ y(\theta) = r\sin\theta \\ z(\theta) = \theta \end{cases}$

5. Run the executable and check if a helix appears on the screen (be careful that the FLTK window - showing your curves and points- uses perspective).

## *Lagrange polynomials with Tchebychev knots*

As explained during the theoretical lecture, Lagrange interpolation can be subject to Runge phenomenon. However, using the Tchebychev abscissa for the location of the control point limits this phenomenon. We will compare helixes drawn with the two kinds of knots and the effects of perturbation on one knot.

1. In the file "`test/curves/main.cc`" an equivalent to `init_lagrange_helix()` must be coded with Tchebychev knots. Create this new function under the previous one but do not forget to also place its prototype before `main()`.

   ```
   void init_lagrange_tcheby(nlagrange& curve, double r)
   ```

   This function will do the same thing as the `init_lagrange_helix` function, except that the control points are located at the Tchebychev abscissa. These abscissas are given on a closed interval `[a, b]` by:

   $$t_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{2i+1}{2n}\pi\right), i = 0, \dots, n-1$$

   For our case, use the interval [0,1].

2. In the function `main()` create a new Lagrange curve with `nCP` control points by adding `nlagrange curve_tcheby(nCP)`. Also, pass this `curve` object to GNurbs to draw it by adding the line `CB.add_entity(&curve_tcheby)` next to the other similar lines.

3. Run the executable and see the similarity of the curves.

4. To see the effect of a perturbation on both curves, modify the two helix functions to move the middle control point vertically of a distance of 10% of the radius.

   For a better comparison of the two curves, translate the last curve (with Tchebychev knots) along the x axes by 30 units (`translate()`).

5. Run and analyse the two curves. Which one is the more stable?

## *Requests*

Send back the C++ files you edited (`nlagrange.cc` and `main.cc`) in a zip file by email to a.bolyn@uliege.be for the 10th of October 23:59 (with a clear object such as: “CADCG: work 1 files”). Do not forget to explain your code and your observations in the commentaries.

Please, all other files such as the builds and the executable are not requested. Only the two C++ files will be opened and tested.