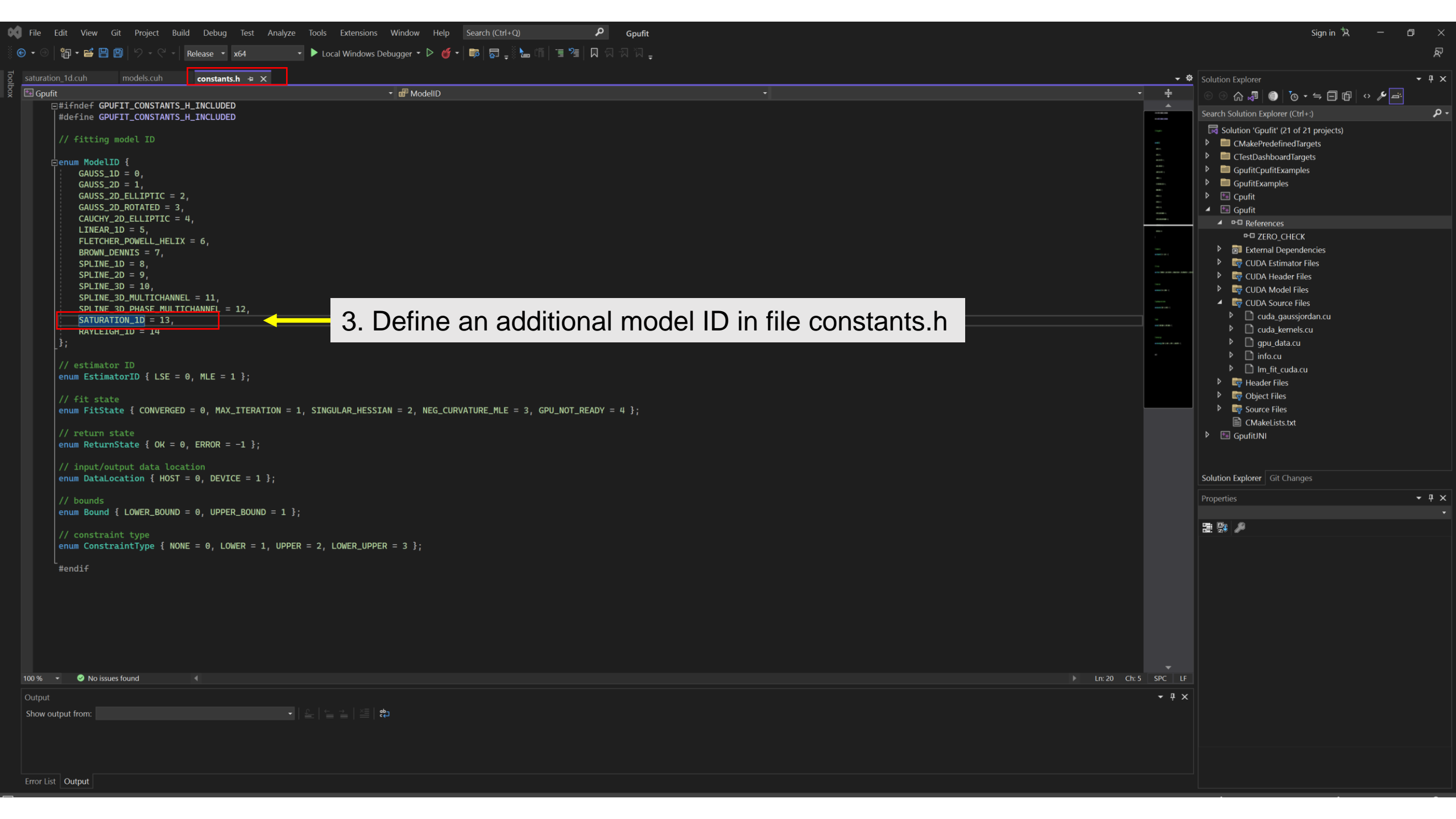


Step by Step guide to add a new fit model function into the GPUFIT

Gpufit GitHub repository: <https://github.com/gpufit/Gpufit>
Gpufit documentation: <https://gpufit.readthedocs.io/en/latest/index.html>

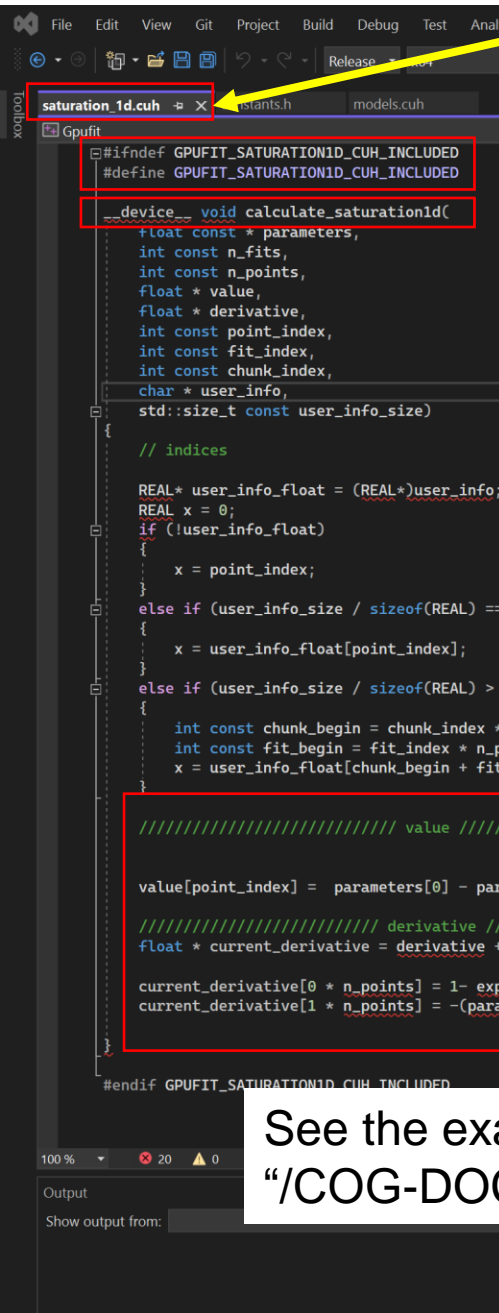
Steps:

1. Download CUDA
[CUDA](#) Toolkit 6.5 or later (tested with 6.5-11.4)
2. Download Visual Studio 2022



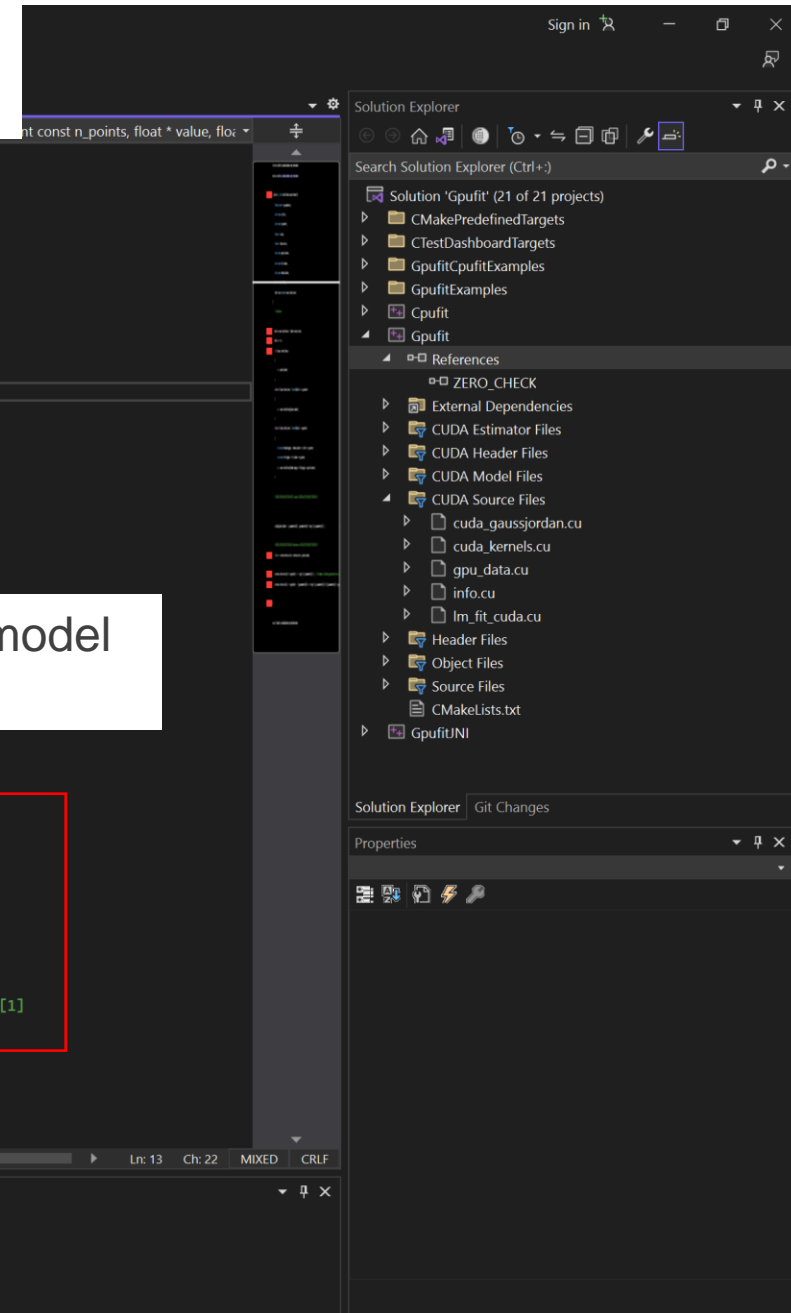
3. Define an additional model ID in file constants.h

4. Implement a CUDA device function within a newly created .cuh file in folder Gpufit/Gpufit/models according to the following template.



partial derivatives of the model, with respect to the model parameters

See the examples model functions in thin link:
"/COG-DOCT/Program/GPUFit/Gpufit-1.2.0/Gpufit/models"



5. Include the newly created .cuh file in models.cuh

Add a switch case in the CUDA device function `calculate_model()` in file `models.cuh` to allow calling the newly added model function.

See the next page (Continued)

```
#ifndef GPUFIT_MODELS_CUH_INCLUDED
#define GPUFIT_MODELS_CUH_INCLUDED

#include <assert.h>
#include "linear_1d.cuh"
#include "gauss_1d.cuh"
#include "gauss_2d.cuh"
#include "gauss_2d_elliptic.cuh"
#include "gauss_2d_rotated.cuh"
#include "cauchy_2d_elliptic.cuh"
#include "fletcher_powell_helix.cuh"
#include "brown_dennis.cuh"
#include "spline_1d.cuh"
#include "spline_2d.cuh"
#include "spline_3d.cuh"
#include "spline_3d_multichannel.cuh"
#include "spline_3d_phase_multichannel.cuh"
#include "saturation_1d.cuh"
#include "rayleigh_1d.cuh"

__device__ void calculate_model(
    ModelID const model_id,
    REAL const * parameters,
    int const n_fits,
    int const n_points,
    REAL * value,
    REAL * derivative,
    int const point_index,
    int const fit_index,
    int const chunk_index,
    char * user_info,
    int const user_info_size)
{
    switch (model_id)
    {
    case GAUSS_1D:
        calculate_gauss1d(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
        break;
    case GAUSS_2D:
        calculate_gauss2d(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
        break;
    case GAUSS_2D_ELLIPTIC:
        calculate_gauss2delliptic(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
        break;
    case GAUSS_2D_ROTATED:
        calculate_gauss2drotated(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
        break;
    case CAUCHY_2D_ELLIPTIC:
        calculate_cauchy2delliptic(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
        break;
    case INFAR_1D:
        calculate_infar1d(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
        break;
    }
}
```

Visual Studio Code interface showing the `models.cuh` file in the `Gpufit` project. The file contains a switch statement for `configure_model()` and a `case SATURATION_1D:` block.

The `case SATURATION_1D:` block is highlighted with a red box, and a yellow arrow points to it from the text box.

Add a switch case in function `configure_model()` in file `models.cuh`.

```
case SATURATION_1D:
    calculate_saturation1d(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
    break;

case SATURATION_1D:
    calculate_saturation1d(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
    break;

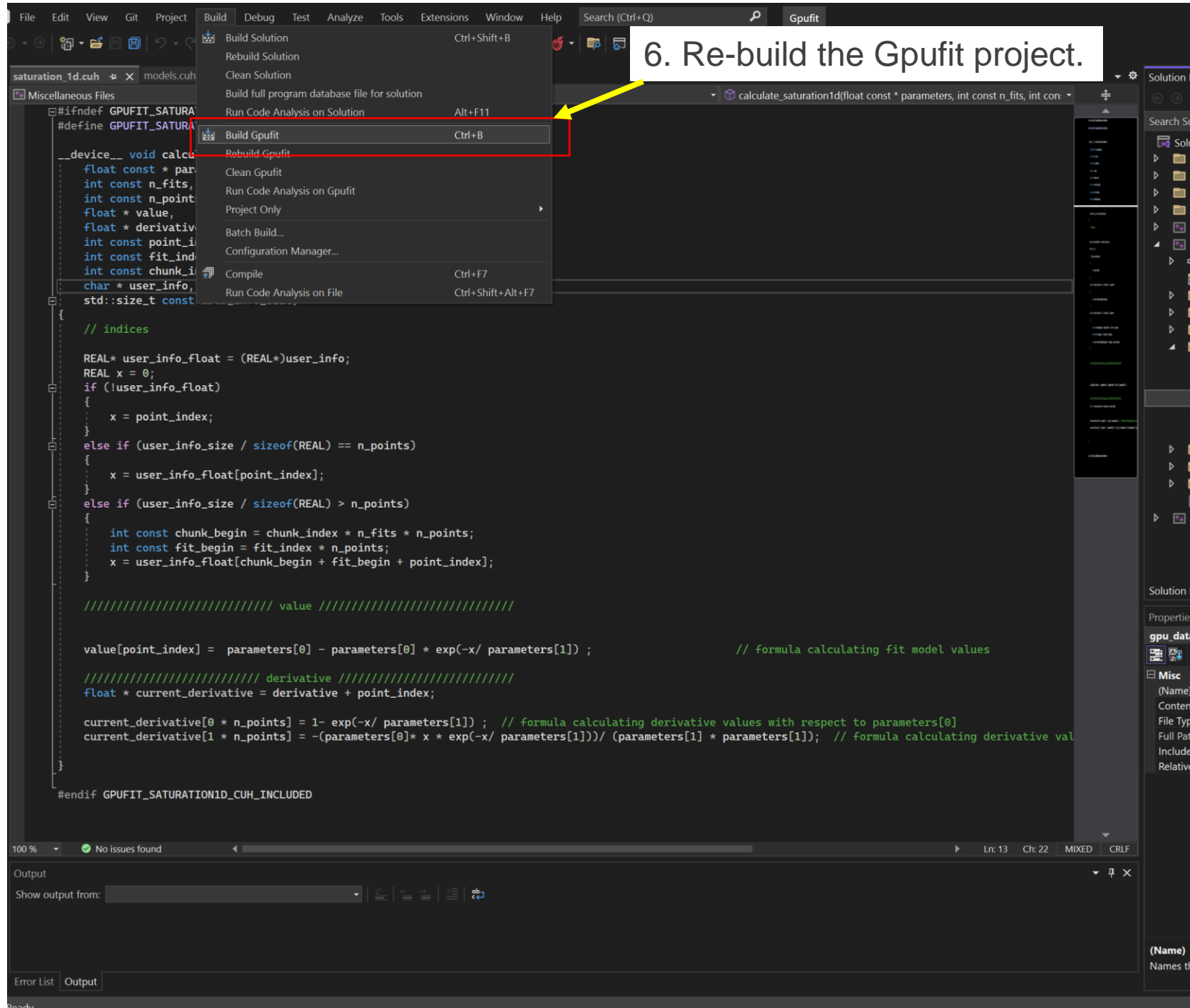
case RAYLEIGH_1D:
    calculate_rayleigh1d(parameters, n_fits, n_points, value, derivative, point_index, fit_index, chunk_index, user_info, user_info_size);
    break;

default:
    assert(0); // unknown model ID

void configure_model(ModelID const model_id, int & n_parameters, int & n_dimensions)
{
    switch (model_id)
    {
        case GAUSS_1D: n_parameters = 4; n_dimensions = 1; break;
        case GAUSS_2D: n_parameters = 5; n_dimensions = 2; break;
        case GAUSS_2D_ELLIPTIC: n_parameters = 6; n_dimensions = 2; break;
        case GAUSS_2D_ROTATED: n_parameters = 7; n_dimensions = 2; break;
        case CAUCHY_2D_ELLIPTIC: n_parameters = 6; n_dimensions = 2; break;
        case LINEAR_1D: n_parameters = 2; n_dimensions = 1; break;
        case FLETCHER_POWELL_HELIX: n_parameters = 3; n_dimensions = 1; break;
        case BROWN_DENNIS: n_parameters = 4; n_dimensions = 1; break;
        case SPLINE_1D: n_parameters = 3; n_dimensions = 1; break;
        case SPLINE_2D: n_parameters = 4; n_dimensions = 2; break;
        case SPLINE_3D: n_parameters = 5; n_dimensions = 3; break;
        case SPLINE_3D_MULTICHANNEL: n_parameters = 5; n_dimensions = 4; break;
        case SPLINE_3D_PHASE_MULTICHANNEL: n_parameters = 6; n_dimensions = 4; break;
        case SATURATION_1D: n_parameters = 2; n_dimensions = 1; break;
        case RAYLEIGH_1D: n_parameters = 1; n_dimensions = 1; break;
        default: throw std::runtime_error("unknown model ID");
    }
}
```

The `case SATURATION_1D: n_parameters = 2; n_dimensions = 1; break;` line is highlighted with a red box.

6. Re-build the Gpufit project.



The screenshot shows the Visual Studio Code interface with the 'Build' menu open. The 'Rebuild Gpufit' option is highlighted with a red box and a yellow arrow pointing to it. The background shows a C++ source file named 'saturation_1d.cuh' with CUDA code. The bottom status bar indicates 'No issues found' and the output window is visible.

```
#ifndef GPUFIT_SATURATION1D_CUH_INCLUDED
#define GPUFIT_SATURATION1D_CUH_INCLUDED

__device__ void calculate_saturation1d(float const * parameters, int const n_fits, int const n_points, float * value, float * derivative, int const point_index, int const fit_index, int const chunk_index, char * user_info, std::size_t const user_info_size)
{
    // indices
    REAL* user_info_float = (REAL*)user_info;
    REAL x = 0;
    if (!user_info_float)
    {
        x = point_index;
    }
    else if (user_info_size / sizeof(REAL) == n_points)
    {
        x = user_info_float[point_index];
    }
    else if (user_info_size / sizeof(REAL) > n_points)
    {
        int const chunk_begin = chunk_index * n_fits * n_points;
        int const fit_begin = fit_index * n_points;
        x = user_info_float[chunk_begin + fit_begin + point_index];
    }

    // value
    value[point_index] = parameters[0] - parameters[0] * exp(-x / parameters[1]); // formula calculating fit model values

    // derivative
    float * current_derivative = derivative + point_index;
    current_derivative[0 * n_points] = 1 - exp(-x / parameters[1]); // formula calculating derivative values with respect to parameters[0]
    current_derivative[1 * n_points] = -(parameters[0] * x * exp(-x / parameters[1])) / (parameters[1] * parameters[1]); // formula calculating derivative values with respect to parameters[1]
}

#endif GPUFIT_SATURATION1D_CUH_INCLUDED
```

Code

Blame

323 lines (262 loc) · 13.9 KB

Raw



```
39 cuda_available_func = lib.gpufit_cuda_available
40 cuda_available_func.restype = c_int
41 cuda_available_func.argtypes = None
42
43 # gpufit_get_cuda_version function in the dll
44 get_cuda_version_func = lib.gpufit_get_cuda_version
45 get_cuda_version_func.restype = c_int
46 get_cuda_version_func.argtypes = [POINTER(c_int), POINTER(c_int)]
47
48
49 class ModelID:
50     GAUSS_1D = 0
51     GAUSS_2D = 1
52     GAUSS_2D_ELLIPTIC = 2
53     GAUSS_2D_ROTATED = 3
54     CAUCHY_2D_ELLIPTIC = 4
55     LINEAR_1D = 5
56     FLETCHER_POWELL = 6
57     BROWN_DENNIS = 7
58     SPLINE_1D = 8
59     SPLINE_2D = 9
60     SPLINE_3D = 10
61     SPLINE_3D_MULTICHANNEL = 11
62     SPLINE_3D_PHASE_MULTICHANNEL = 12
63     SATURATION_1D = 13
64
65
66 class EstimatorID:
67     LSE = 0
68     MLE = 1
69
70
```

Add the model ID in file gpufit.py