

Use case application of SU2 soft package

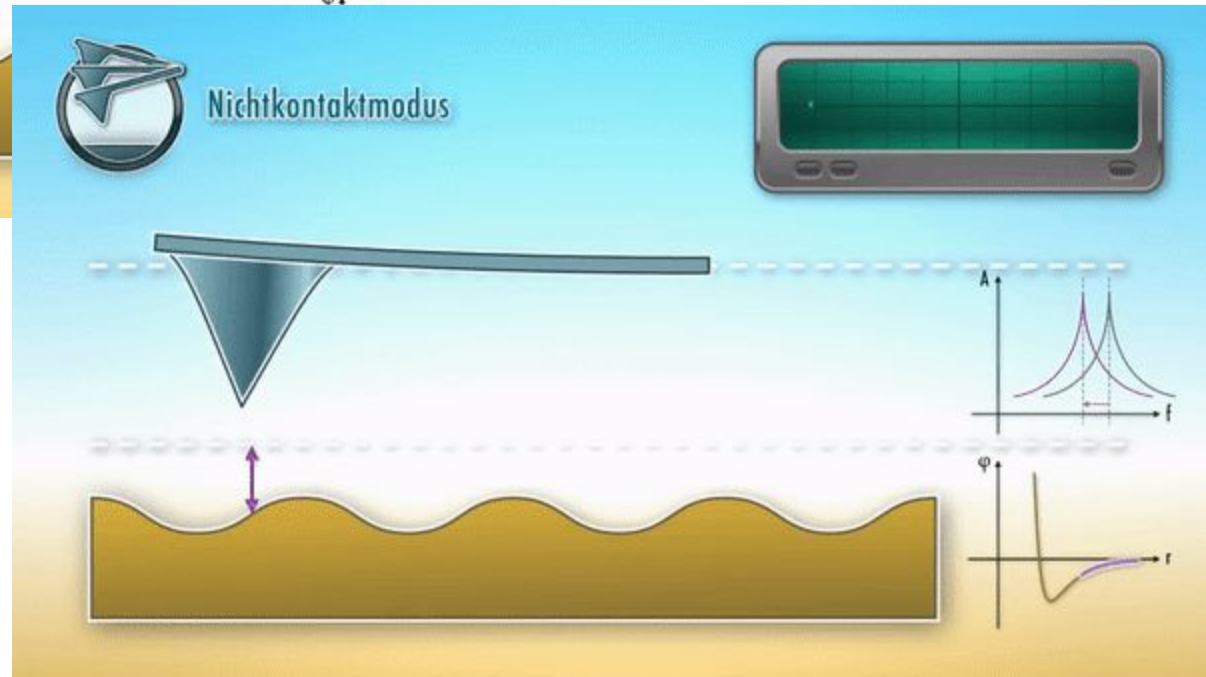
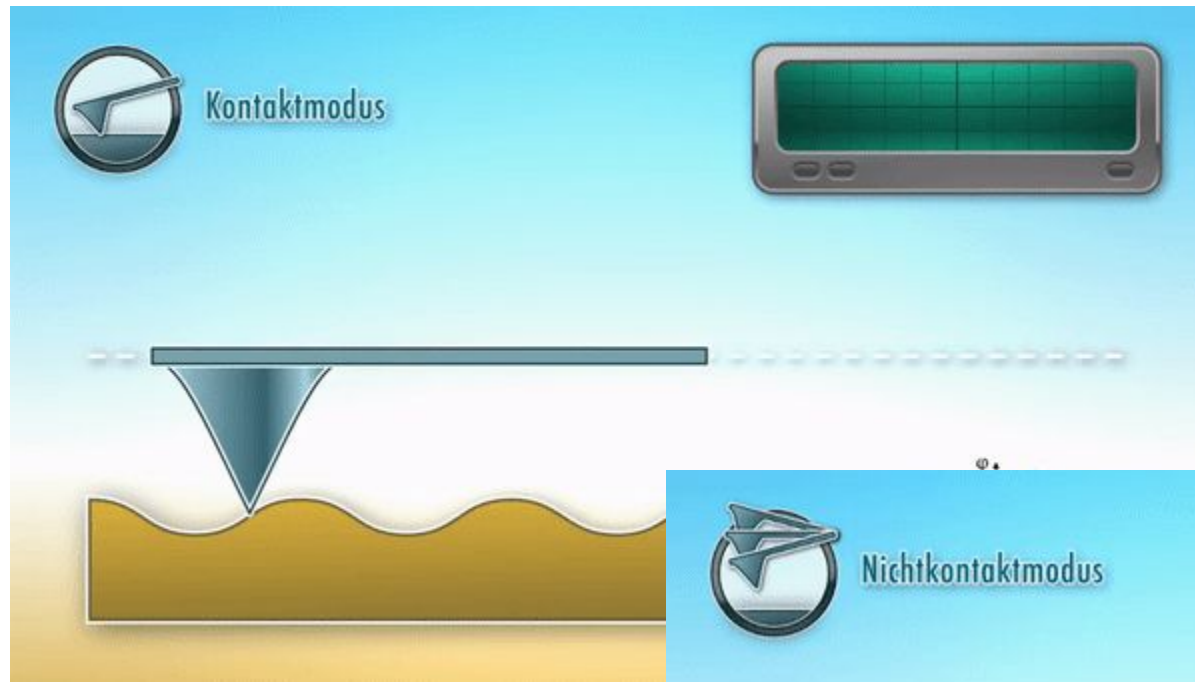
Advanced Computing

C++ library NXV4

- 1 *Damped Oscillatory Motion in AFM*
- 2 *C++ packages SU2 / CPX*
- 3 *Elongation x determination*
- 4 *Appl 1 – Pulse and Step*
- 5 *Appl 2 – Harmonic Drive*

- 1 *Damped Oscillatory Motion in AFM*
- 2 *C++ packages SU2 / CPX*
- 3 *Elongation x determination*
- 4 *Appl 1 – Pulse and Step*
- 5 *Appl 2 – Harmonic Drive*

Damped Oscillatory Motion in AFM



AFM modes

- contact
- non-contact
- tapping



Modelling software

- *implement mathematical fields / vec's necessary*
- *in polymorphic form*
- *F77-like for the user*
- *independent of other frameworks*
- *CPU acceleration*



- 1 *Damped Oscillatory Motion in AFM*
- 2 *C++ packages SU2 / CPX*
- 3 *Elongation x determination*
- 4 *Appl 1 – Pulse and Step*
- 5 *Appl 2 – Harmonic Drive*

All polymorphic, ca. 1220 (CPX) op-instantiations, 2470 (CHI), 5680 (SU2)

```
cpx<int> + su2<double> = su2<cpx<double>>
```

CPX

- *complex numbers*
- *all algebraic op's*, exp, log, sqrt, a^b , $\sim z$
- fabs, phi, phi2, >, >=, ==, !=, *etc*

CHI

- *SU(2) vectors*
- *all algebraic op's*, $\sim v$, (psi | phi), $\mathbf{z} = a^b$, fabs, >, ==, !=, *etc*

SU2

- *SU(2) operators*
- *all algebraic op's*, exp, log, sqrt, $\sim \mathbf{z}$
- fabs, $\mathbf{A} | \text{phi}$, (psi | $\mathbf{A} | \text{phi}$), >, >=, ==, !=, *etc*

- 1 *Damped Oscillatory Motion in AFM*
- 2 *C++ packages SU2 / CPX*
- 3 *Elongation x determination*
- 4 *Appl 1 – Pulse and Step*
- 5 *Appl 2 – Harmonic Drive*

Newton

$$ma = -\eta v - kx + f(t)$$

$$\ddot{x} + \frac{1}{Q}\dot{x} + x = \frac{1}{k}f(\tau)$$

$$\omega_0 = \sqrt{\frac{k}{m}}$$

$$\tau = \omega_0 t \rightarrow \partial_t = \omega_0 \partial_\tau$$

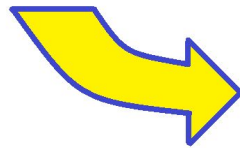
$$Q = \frac{1}{\eta} \sqrt{km}$$

$$\text{Notation : } \dot{z} = \partial_\tau z$$

$$\partial_\tau \begin{pmatrix} \dot{x} \\ x \end{pmatrix} + \underbrace{\begin{pmatrix} 1/Q & 1 \\ -1 & 0 \end{pmatrix}}_{\parallel \Omega} \underbrace{\begin{pmatrix} \dot{x} \\ x \end{pmatrix}}_{\parallel |\psi\rangle} = \frac{f}{k} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{f}{k} |\uparrow\rangle$$

First order differential equation – find ψ

$$|\psi\rangle' + \Omega|\psi\rangle = \frac{f}{k}|\uparrow\rangle$$



$$|\psi\rangle = e^{-\Omega\tau}|\chi\rangle$$

$$e^{-\Omega\tau}|\chi\rangle' = \frac{f}{k}|\uparrow\rangle$$

$$|\chi\rangle = \frac{1}{k} \int_{-\infty}^{\tau} e^{\Omega\tau_1} f(\tau_1) |\uparrow\rangle d\tau_1$$

$$|\psi\rangle = e^{-\Omega\tau} \frac{1}{k} \int_{-\infty}^{\tau} e^{\Omega\tau_1} f(\tau_1) |\uparrow\rangle d\tau_1$$

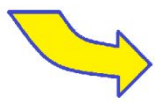
$$x = \langle \downarrow | \psi \rangle$$

$$x = \frac{\omega_0}{k} \int_{-\infty}^t \langle \downarrow | e^{-\Omega \omega_0 (t-t_1)} | \uparrow \rangle f(t_1) dt_1$$


 O_{II} eq. \Rightarrow prime integr. contained in here

Note:

$$\Omega = \frac{1}{2Q} (\mathbb{1} + \sigma_z) + i\sigma_y$$



$$\left(\Omega - \frac{1}{2Q} \right)^2 = -1 + \frac{1}{4Q^2} \simeq i^2 \cdot \mathbb{1}$$

- 1 *Damped Oscillatory Motion in AFM*
- 2 *C++ packages SU2 / CPX*
- 3 *Elongation x determination*
- 4 *Appl 1 – Pulse and Step*
- 5 *Appl 2 – Harmonic Drive*

$$f(t) = f_0 \delta(t - t_0)$$



Dirac function

$$x = \frac{f_0}{k} \langle \downarrow | e^{-\Omega \omega_0 (t - t_0)} | \uparrow \rangle \Theta(\tau - \tau_0)$$



Integral of a Dirac function is Heaviside function

Appl 1 – Pulse and Step

```
#include <cmath>
#include <iostream>

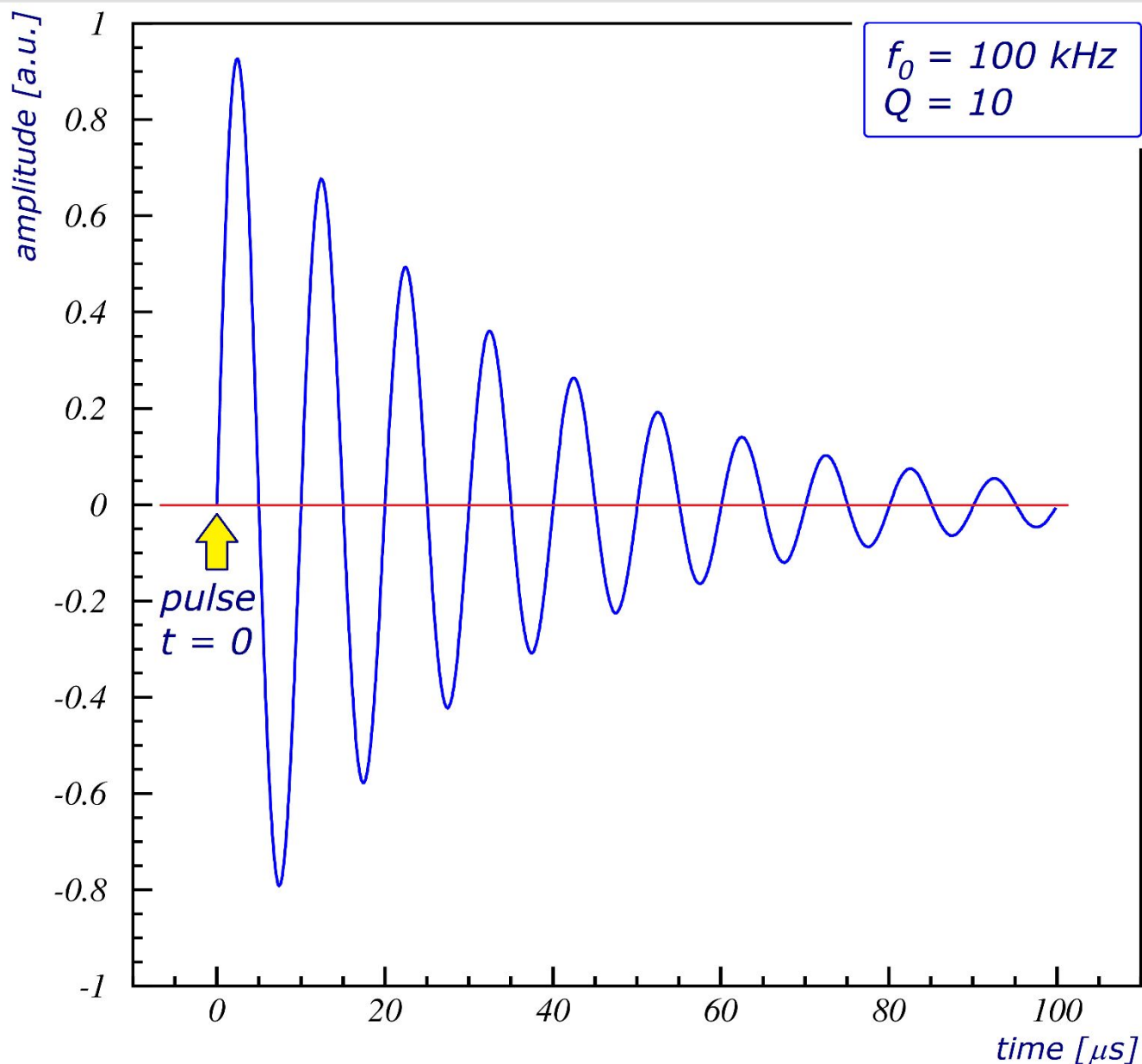
#include "cpx.hh"
#include "chi.hh"
#include "su2.hh"

using namespace std
using namespace cpx4
using namespace chi4
using namespace su24

int main()
{
    using fltx = float
    using dblx = double
    using real = long double

    double PI = 3.14159265

    auto i = cpx<real>(0,
    auto sx = su2<real>(0,
    auto sy = su2<real>(0,
    auto sz = su2<real>(1,
    auto up = chi<real>(1,
    auto down = chi<real>(0,
```



$$f(t) = f_0 \Theta(t - t_0)$$



Heaviside function

$$x = \frac{f_0}{k} \langle \downarrow | \frac{1}{\Omega} (\mathbf{1} - e^{-\Omega \omega_0 (t - t_0)}) | \uparrow \rangle \Theta(t - t_0)$$

Appl 1 – Pulse and Step

```
#include <cmath>
#include <iostream>

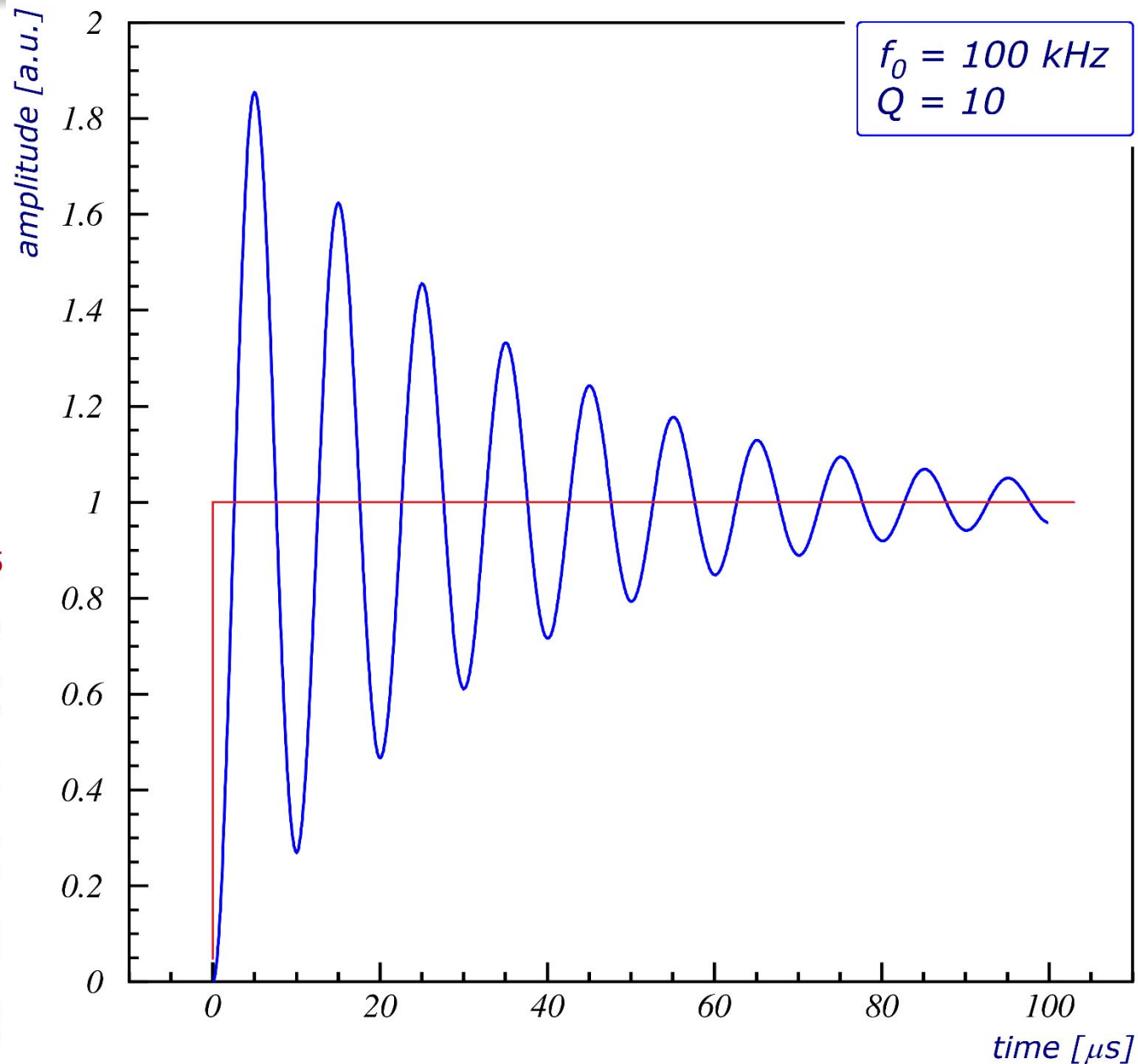
#include "cpx.hh"
#include "chi.hh"
#include "su2.hh"

using namespace std
using namespace cpx4
using namespace chi4
using namespace su24

int main()
{
    using fltx = float
    using dblx = double
    using real = long double

    double PI = 3.14159265

    auto i = cpx<real>(0,
    auto sx = su2<real>(0,
    auto sy = su2<real>(0,
    auto sz = su2<real>(1,
    auto up = chi<real>(1,
    auto down = chi<real>(0,
```



- 1 *Damped Oscillatory Motion in AFM*
- 2 *C++ packages SU2 / CPX*
- 3 *Elongation x determination*
- 4 *Appl 1 – Sudden Pulse*
- 5 *Appl 2 – Harmonic Drive*

$$x = \frac{f_0}{2k} \langle \downarrow | (\Omega + i\lambda)^{-1} (e^{i\lambda\omega_0 t} - e^{-\Omega\omega_0 t}) | \uparrow \rangle + \frac{f_0}{2k} \langle \downarrow | (\Omega - i\lambda)^{-1} (e^{-i\lambda\omega_0 t} - e^{-\Omega\omega_0 t}) | \uparrow \rangle \quad | \quad t \geq 0$$



$$(\Omega \pm i\lambda)^{-1} = \frac{\frac{1}{Q} \pm i\lambda - \Omega}{1 - \lambda^2 \pm \frac{i\lambda}{Q}}$$

$$\text{since } \left(\Omega - \frac{1}{2Q}\right)^2 = -1 + \frac{1}{4Q^2}$$

Appl 2 – Harmonic Drive

```
#include <cmath>
#include <iostream>

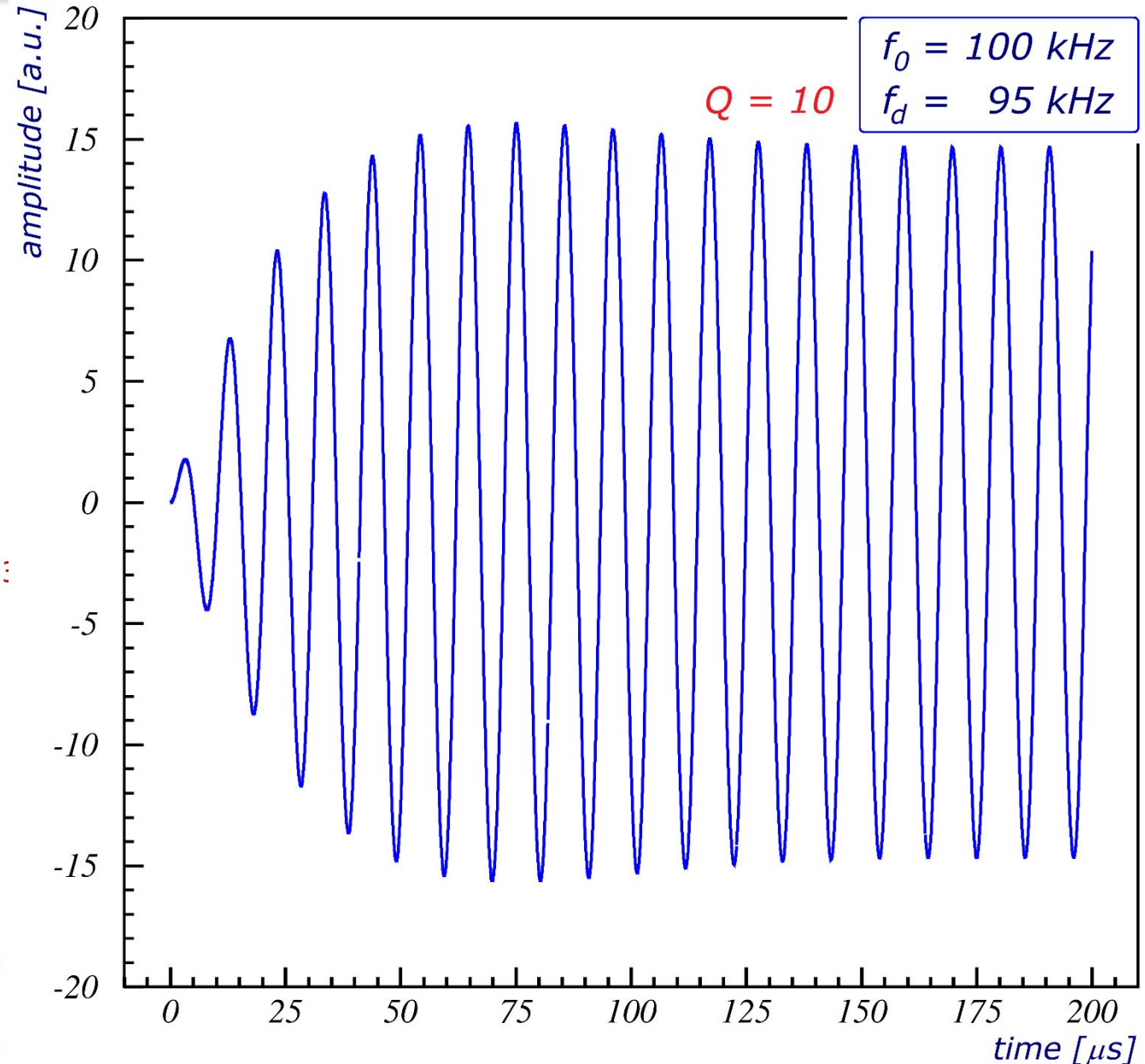
#include "cpx.hh"
#include "chi.hh"
#include "su2.hh"

using namespace std
using namespace cpx4
using namespace chi4
using namespace su24

int main()
{
    using fltx = float
    using dblx = double
    using real = long double

    double PI = 3.14159265358979323846264338327950288419716939937510582097494159572149612019999087614651406914561

    auto j = cpx<real>(0, 1)
    auto sx = su2<real>(0, 1)
    auto sy = su2<real>(0, 1)
    auto sz = su2<real>(1, 0)
    auto up = chi<real>(1, 0)
    auto down = chi<real>(0, 1)
}
```



NXV4 library

- *F77-like for the user*
- *independent of other frameworks*
- *CPU acceleration*

Polymorphic C++

- *quasi-polymorphism: extensive, multi-1000's operators*
tedious, but very rewarding

- ***cpx**, **chi**, **su2**: good CPU performance*

Спасибо !