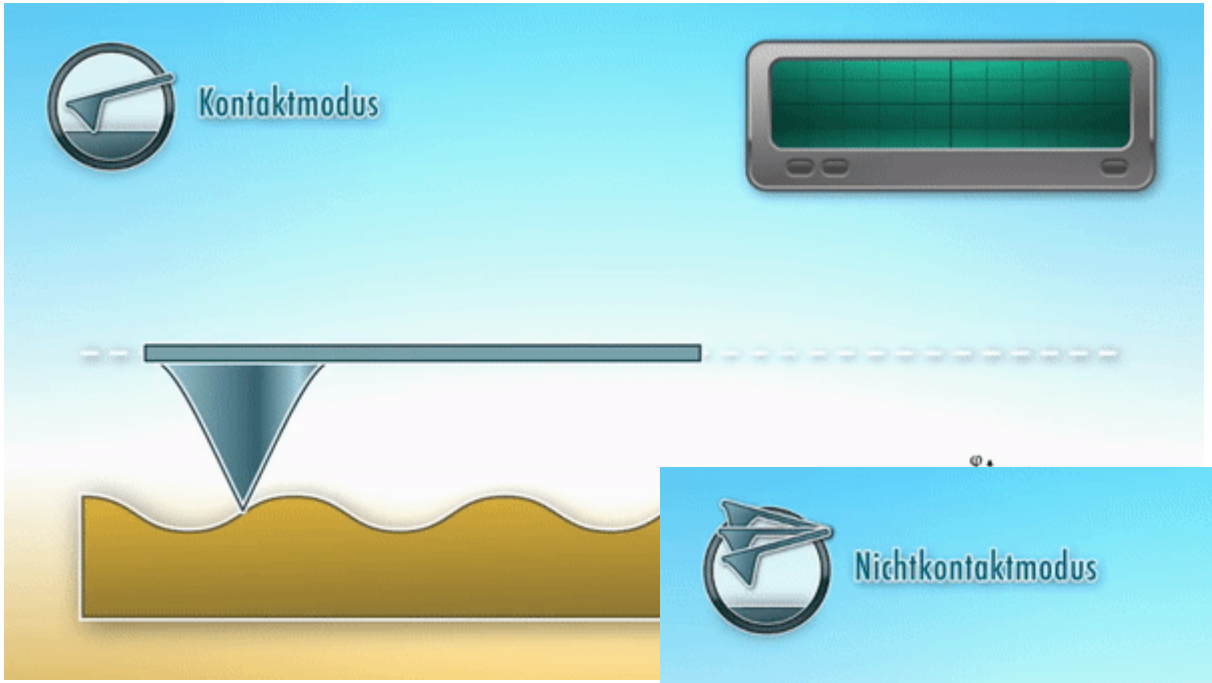
Use case application of SU2 soft package

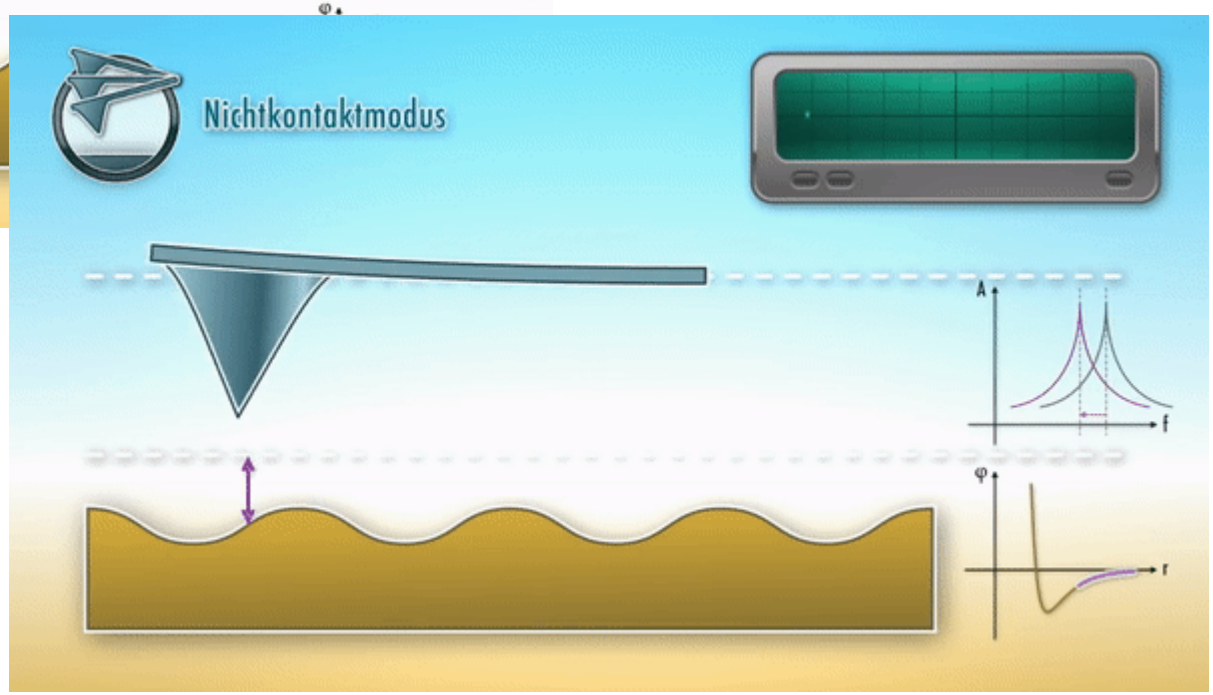
Advanced Computing *C++ library NXV4*

Contents

Contents

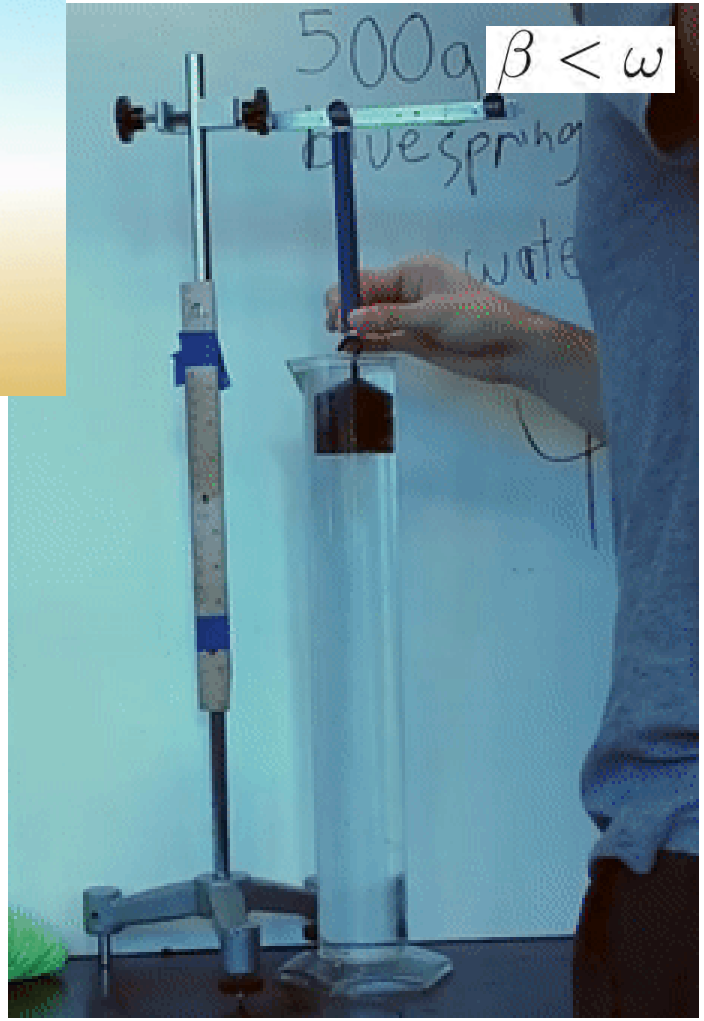
Damped Oscillatory Motion in AFM





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*

Contents

C++ packages SU2 / CPX

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`, `~z`
- `fabs`, `phi`, `phi2`, `>` , `>=`, `==`, `=!`, *etc*

CHI

- *SU(2) vectors*
- *all algebraic op's* , `~v`, `(psi|phi)`, **`Z`** = `a^b` , `fabs`, `>` , `==`, `=!`, *etc*

SU2

- *SU(2) operators*
- *all algebraic op's,* `exp`, `log`, `sqrt`, `~`**`Z`**
- `fabs`, **`A`**`|phi`, `(psi|`**`A`**`|phi)`, `>` , `>=`, `==`, `=!`, *etc*

Contents

Newton

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

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

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

$$\tau = \omega_0 t \quad \longrightarrow \quad \partial_t = \omega_0 \partial_\tau$$

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

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

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

First order diferential ecuation – find ψ

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

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

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

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

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

Elongation x determination

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

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

O_{II} eq. => prime integr. contained in here

Note:

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

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

Contents

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

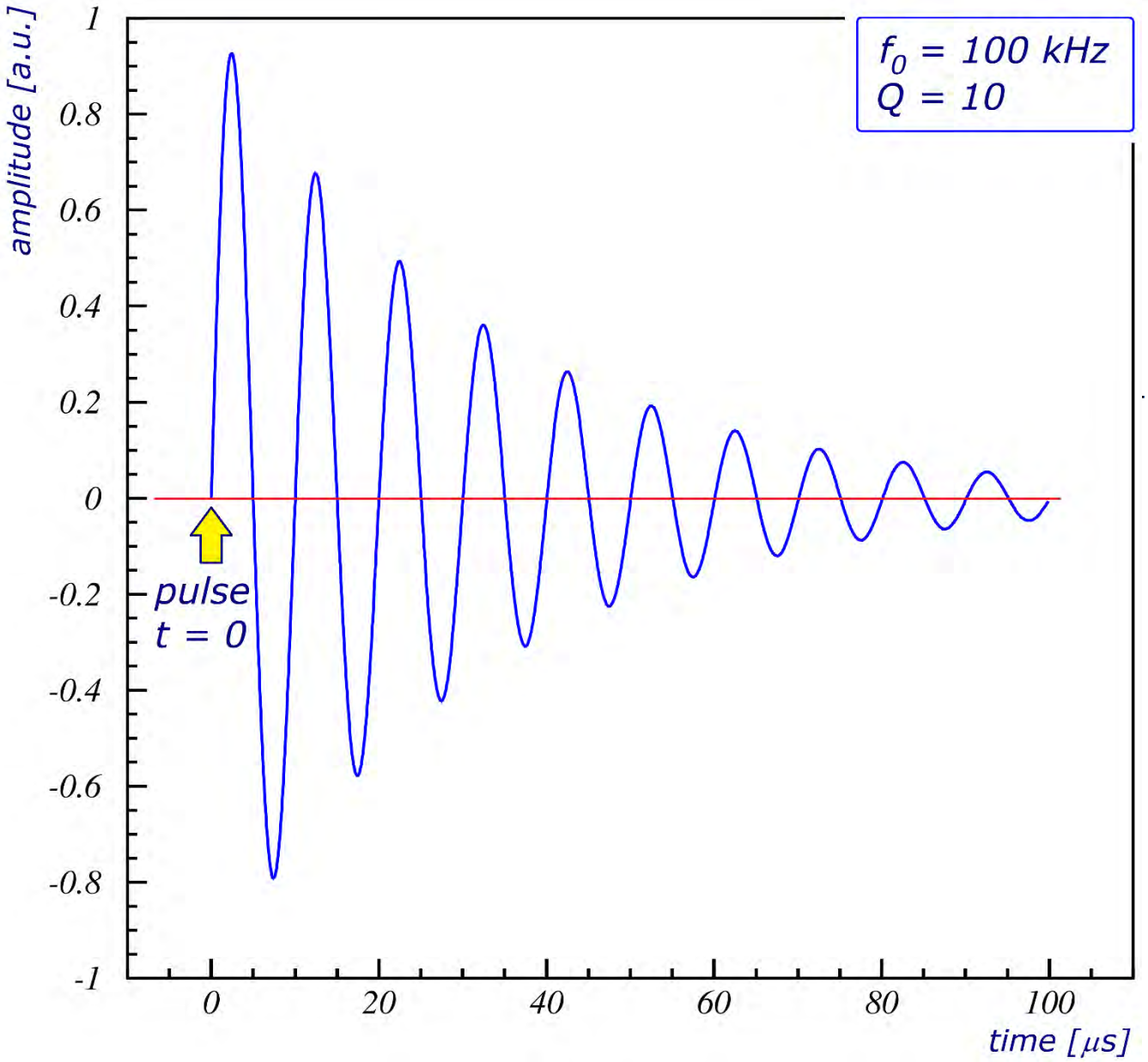
Dirac function

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

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,
                         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)
```



Appl 1 – Pulse and Step

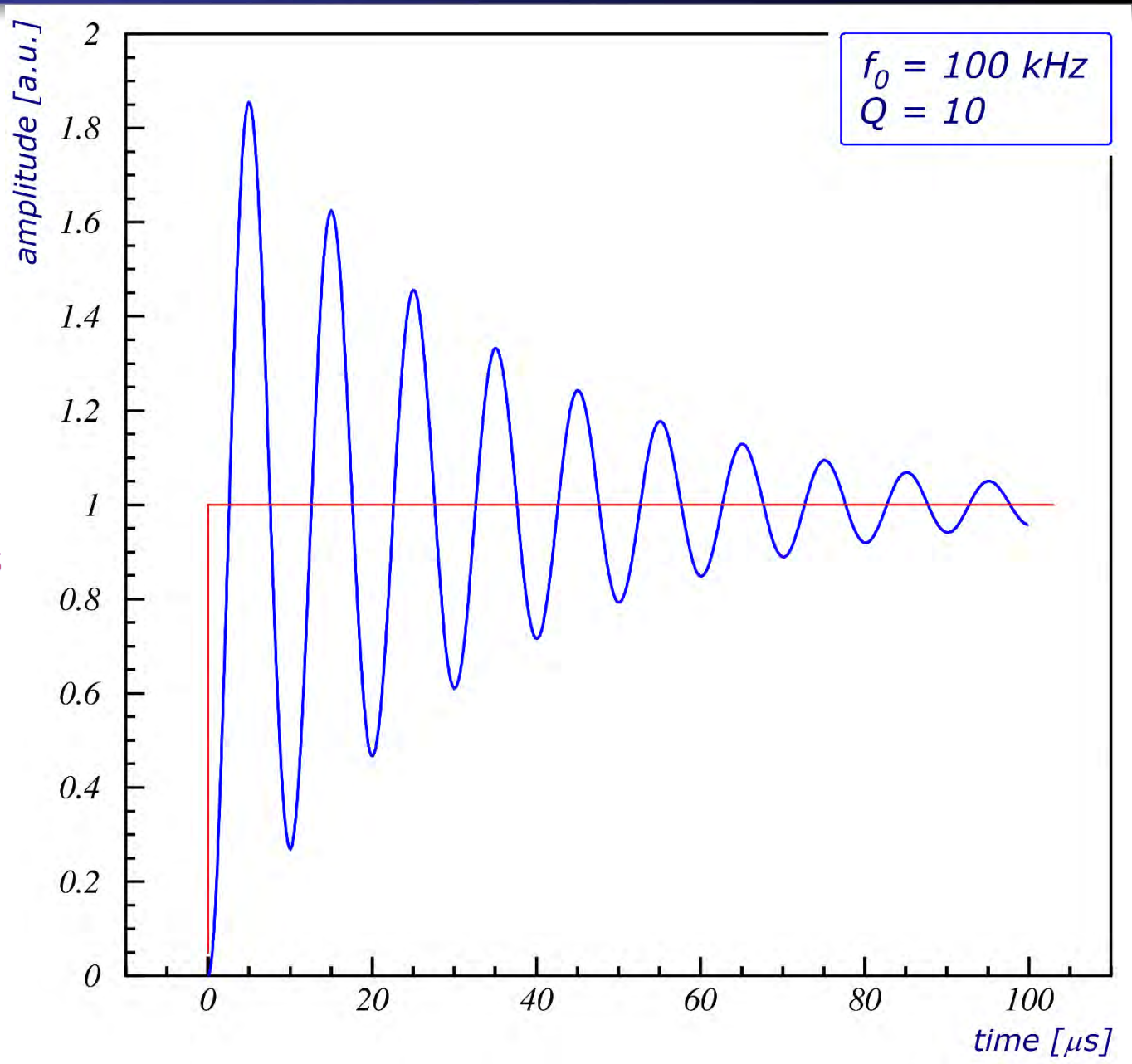
$$f(t) = f_0 \Theta(t - t_0)$$

Heaviside function

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

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,
                         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)
```



Contents

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

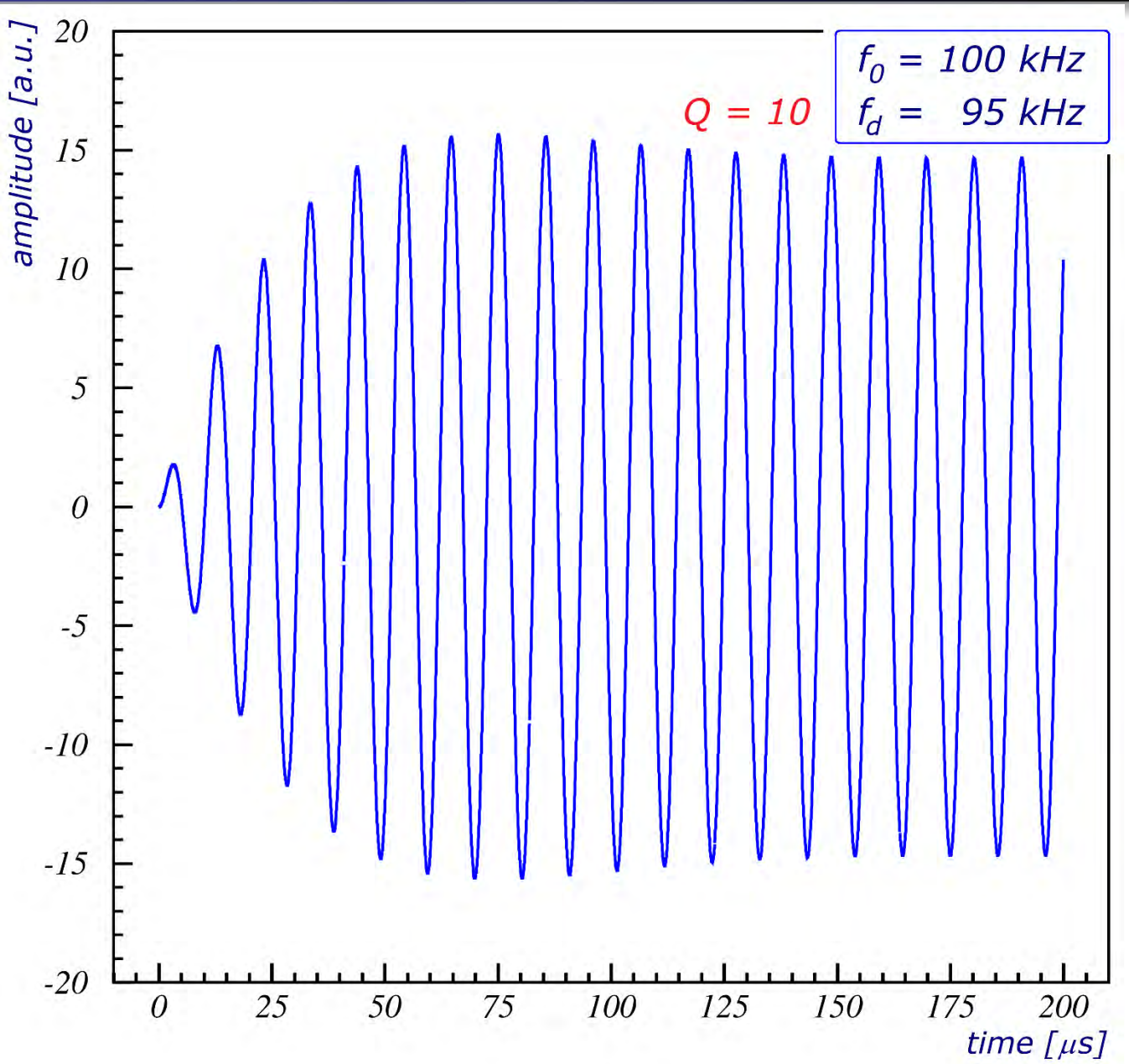
$$\frac{f_0}{2k}\langle\downarrow|\,(\boldsymbol{\Omega}-i\lambda)^{-1}(e^{-i\lambda\omega_0 t}-e^{-\boldsymbol{\Omega}\omega_0 t})\,|\uparrow\rangle \quad \Big|\; t\geq 0$$

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

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

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   =   3.14159265

 auto    j   = cpx<real>(0,

 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)
```



Conclusions

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 !