# #CHW061

Karl Traunmüller on Computable
http://computableapp.com

# About Me

- CompSci degree JKU Linz '93-98

- Embedded Systems '99-00

- C++/MFC '00-02

- Freelancer, Pocket PC Shareware '02-06

- .NET, Java EE (never, never, ever...) '06-10

- Some teaching FH Hagenberg '06-09
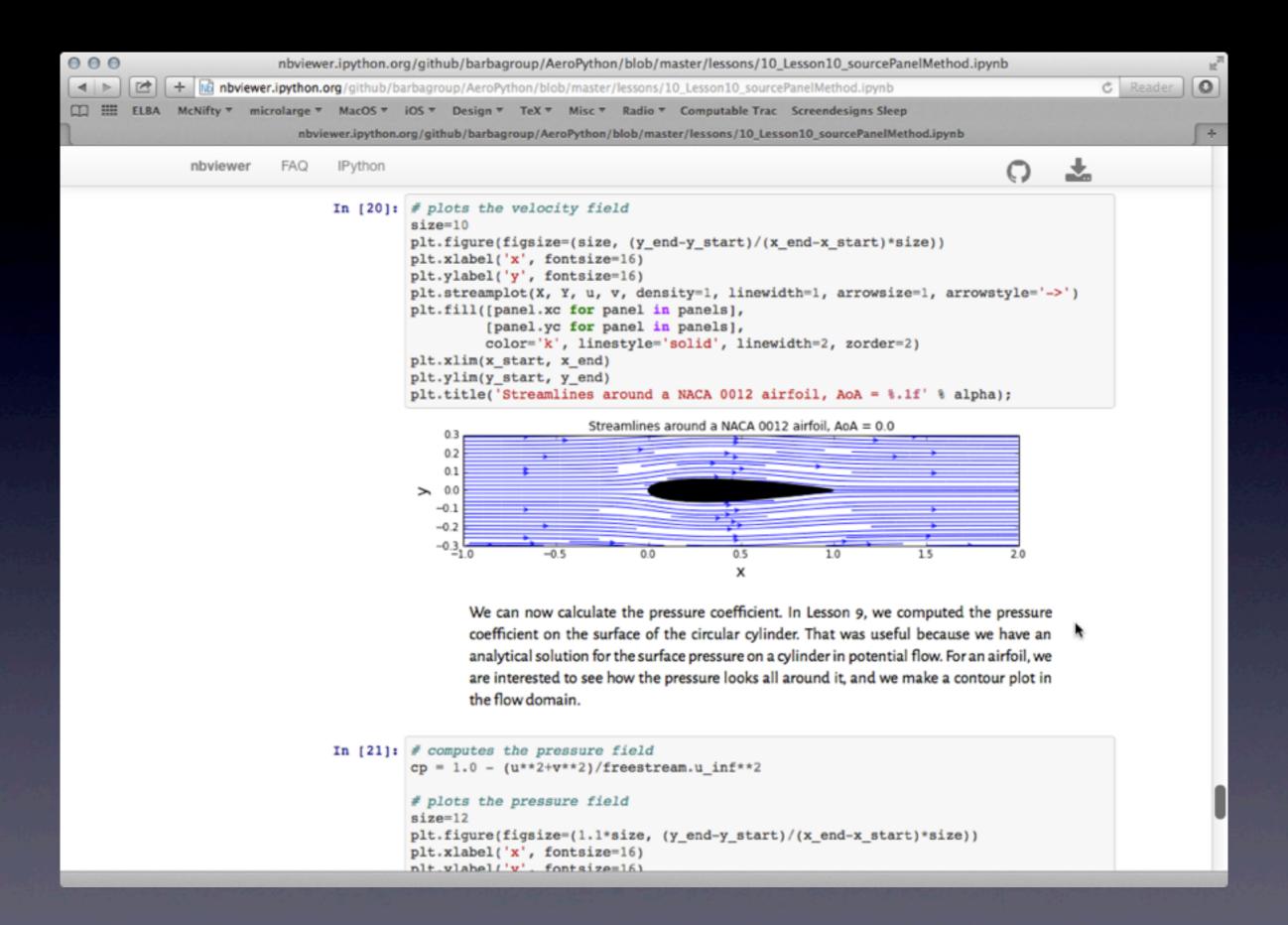
- Freelancer, Mac/iOS apps '10-now

# IPython

- "Open Source MATLAB"
- Would be cool to have on the iPad

ELBA   McNifty ▾   microlarge ▾   MacOS ▾   iOS ▾   Design ▾   TeX ▾   Misc ▾   Radio ▾   Computable Trac   Screendesigns Sleep

In [20]:
```python
# plots the velocity field
size=10
plt.figure(figsize=(size, (y_end-y_start)/(x_end-x_start)*size))
plt.xlabel('x', fontsize=16)
plt.ylabel('y', fontsize=16)
plt.streamplot(X, Y, u, v, density=1, linewidth=1, arrowsize=1, arrowstyle='->')
plt.fill([panel.xc for panel in panels],
         [panel.yc for panel in panels],
         color='k', linestyle='solid', linewidth=2, zorder=2)
plt.xlim(x_start, x_end)
plt.ylim(y_start, y_end)
plt.title('Streamlines around a NACA 0012 airfoil, AoA = %.1f' % alpha);
```



Streamlines around a NACA 0012 airfoil, AoA = 0.0

We can now calculate the pressure coefficient. In Lesson 9, we computed the pressure coefficient on the surface of the circular cylinder. That was useful because we have an analytical solution for the surface pressure on a cylinder in potential flow. For an airfoil, we are interested to see how the pressure looks all around it, and we make a contour plot in the flow domain.

In [21]:
```python
# computes the pressure field
cp = 1.0 - (u**2+v**2)/freestream.u_inf**2

# plots the pressure field
size=12
plt.figure(figsize=(1.1*size, (y_end-y_start)/(x_end-x_start)*size))
plt.xlabel('x', fontsize=16)
plt.ylabel('y', fontsize=16)
```

# IPython on iPad

- Doable?

  - Why has nobody done it before?

  - 1.5 mio apps in the store, but not this?

# Getting Started

- Let's run a feasibility study!

- Answer the doable question in 1-2 weeks

  - late Jan

# Up-front Questions

- We need an embeddable Python runtime

  - Should be no problem

- Will IPython work at all?

  - Architecture looking good

  - Kernel – Messaging – UI (ZeroMQ on iOS?)

  - Nice, simple messaging protocol

  - Notebooks are JSON (good)

# Up-front Questions

- We'll have to port the full SciPy stack

  - numpy

  - matplotlib

  - SciPy

  - SymPy

  - pandas

# Up-front Questions

- Can we get all core packages to compile?

  - Heavy use of native (C) extensions

  - Quite a bit of Fortran code in scipy

    - Translation to C required (f2c)

    - Extend numpy's distutils setup framework

- Plotting: Matplotlib

  - AGG software rendering (good)

# Cross Compiling Fun

```
export CC="$ARM_CC -I$BUILDROOT/include $ARM_CFLAGS -fno-builtin"
export CFLAGS="$ARM_CFLAGS"
export CFLAGS="$CFLAGS -I$BUILDROOT/python/include/python2.7"
export CFLAGS="$CFLAGS -I$TMPROOT/numpy-$NUMPY_VERSION/sources/src/private"
export CFLAGS="$CFLAGS -I$TMPROOT/numpy-$NUMPY_VERSION/sources/include"
export CFLAGS="$CFLAGS -iquote$TMPROOT/numpy-$NUMPY_VERSION/sources/numpy/core...
export LD="$ARM_LD $ARM_LDFLAGS"
export AR="$ARM_AR"
export LDFLAGS="$ARM_LDFLAGS"
export LDSHARED="$PRJROOT/tools/liblink"

...

echo "compiling sources..."
cp -f $PRJROOT/src/numpy_files/make_Makefile Makefile
make

echo "creating libnumpy.a..."
cp -f $PRJROOT/src/numpy_files/link_Makefile Makefile
make

cp -r $TMPROOT/numpy-$NUMPY_VERSION/numpy/core/include $BUILDROOT/include
cp libnumpy.a $BUILDROOT/lib
```

# Cross Compiling Fun

- Getting the static libraries to compile standalone is one thing

- Fixing all "unresolved symbol" linker errors in the app project is another

# Other Issues

- We're in a single-process, multi-threaded environment

    - Violates some core assumptions

    - Singletons, argh!

- We're in a static linking environment

    - Symbol name clashes

    - No dynamic loading of Python modules

# Other Issues

- Some Patching required

  - Duplicate/unresolved symbols

  - Non-matching prototype/impl. (Fortran)

- Python Module initialization

  - numpy and pandas require special setup

- Python runtime

  - Very easy to lock up (GIL)

# Other Issues

- Private API usage
  - BLAS clashes with Accelerate framework

# Doable?

- Yes

  - early Feb

  - large parts of SciPy stack ported by now

# App Architecture

- Python runtime

  - Static module initialization

- IPython application

  - kernel and notebook mgmt.

  - custom, based on HTML notebook app

- ZeroMQ backend/frontend messaging

- Frontend (sessions, notebooks, UI)

# Execution Model

- Frontend submits request (code to execute)

- Kernel receives request over ZMQ

- Kernel replies with several messages

  - status, execute reply, display data, ...

- Frontend collects responses, updates UI

- Very nicely decoupled

# Notebook Rendering

- Markdown parser + obj model + renderer

- Parser

  - libsoldout (pluggable backend)

- Renderer

  - First attempt: NSAttributedString

  - (La)TeX: Oops! (no lib?!)

  - Rewrite for HTML / MathJax

# Notebook Rendering

- Markdown → "DOM" → HTML+Javascript

- Load into UIWebView

- Cache rendered output for performance

- Works, but very slow

## Step 4: Burgers' Equation

You can read about Burgers' Equation on its wikipedia page.

Burgers' equation in one spatial dimension looks like this:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

As you can see, it is a combination of non-linear convection and diffusion. It is surprising how much you learn from this neat little equation!

We can discretize it using the methods we've already detailed in Steps 1 to 3. Using forward difference for time, backward difference for space and our 2nd-order method for the second derivatives yields:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + u_i^n \frac{u_i^n - u_{i-1}^n}{\Delta x} = \nu \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$$

As before, once we have an initial condition, the only unknown is $u_i^{n+1}$. We will step in time as follows:

$$u_i^{n+1} = u_i^n - u_i^n \frac{\Delta t}{\Delta x}(u_i^n - u_{i-1}^n) + \nu \frac{\Delta t}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

### Initial and Boundary Conditions

To examine some interesting properties of Burgers' equation, it is helpful to use different initial and boundary conditions than we've been using for previous steps.

Our initial condition for this problem is going to be:

$$u = -\frac{2\nu}{\phi}\frac{\partial \phi}{\partial x} + 4$$

$$\phi = \exp\left(\frac{-x^2}{4\nu}\right) + \exp\left(\frac{-(x - 2\pi)^2}{4\nu}\right)$$

This has an analytical solution, given by:

$$u = -\frac{2\nu}{\phi}\frac{\partial \phi}{\partial x} + 4$$

$$\phi = \exp\left(\frac{-(x - 4t)^2}{4\nu(t + 1)}\right) + \exp\left(\frac{-(x - 4t - 2\pi)^2}{4\nu(t + 1)}\right)$$

Our boundary condition will be:

$$u(0) = u(2\pi)$$

# Some Git History

| | | | |
|---|---|---|---|
| kernels starting on iPad, although not with correct configuration file | 1eacc1e | Karl Traunmüller... | 04.03.2014 23:03 |
| fixed kernel poller (now using heartbeat channel instead of poll) | 91dfd5a | Karl Traunmüller... | 04.03.2014 18:48 |
| IPKernelApp running again! | c085671 | Karl Traunmüller... | 04.03.2014 01:49 |
| we need a custom kernel launcher... | 9fabb08 | Karl Traunmüller... | 04.03.2014 01:26 |
| multi-kernel app basically running! | 710f57c | Karl Traunmüller... | 04.03.2014 00:16 |
| notebook app refactoring partially working | e115c8a | Karl Traunmüller... | 03.03.2014 23:34 |
| new NotebookApp is launching | 67d2efd | Karl Traunmüller... | 02.03.2014 21:11 |
| before kernel manager refactoring | 8576046 | Karl Traunmüller... | 02.03.2014 16:44 |
| added kernel version info log output | a46a8fa | Karl Traunmüller... | 02.03.2014 15:05 |
| IPython upgrade to 1.2.1 | 7d3853d | Karl Traunmüller... | 02.03.2014 14:50 |

| | | | |
|---|---|---|---|
| second kernel doesn't launch... | 4dbd665 | Karl Traunmüller... | 12.02.2014 00:31 |
| 2 tests running | efac136 | Karl Traunmüller... | 11.02.2014 20:28 |
| fixed notebook startup & shutdown | f01554b | Karl Traunmüller... | 11.02.2014 14:23 |
| basic start sequence test runs through, but blocks main thread (todo: run interpreter in thread) | 17163a2 | Karl Traunmüller... | 11.02.2014 00:30 |
| unit tests | 27116f8 | Karl Traunmüller... | 09.02.2014 19:34 |
| python parser, unit test target | 4d87d75 | Karl Traunmüller... | 09.02.2014 19:11 |
| cell collection view, tab control | 551753b | Karl Traunmüller... | 09.02.2014 16:38 |
| basic view controller setup | 3776d3c | Karl Traunmüller... | 09.02.2014 14:24 |
| added session delegate, message parsing | 1a15431 | Karl Traunmüller... | 06.02.2014 23:30 |
| basic UI | 2681897 | Karl Traunmüller... | 06.02.2014 22:03 |
| initial commit | 6e05ada | Karl Traunmüller... | 06.02.2014 21:42 |

# Some Git History

| | | | |
|---|---|---|---|
| fixed Markdown rendering, added hyphenation | fb9e85d | Karl Traunmüller... | 14.04.2014 23:47 |
| HTML formatting | e225ded | Karl Traunmüller... | 13.04.2014 20:00 |
| added Computer Modern fonts | 6552c01 | Karl Traunmüller... | 13.04.2014 19:09 |
| basic Markdown HTML rendering with MathJax | 77f063d | Karl Traunmüller... | 13.04.2014 17:39 |
| before Markdown rendering refactoring | a053ef5 | Karl Traunmüller... | 13.04.2014 13:15 |
| fixed toolbar positioning when keyboard is visible | 10ce66c | Karl Traunmüller... | 28.04.2014 21:17 |
| added matplotlibrc setup | 59fb137 | Karl Traunmüller... | 28.04.2014 21:17 |
| added check to prevent exception on message dispatch | 51f40c0 | Karl Traunmüller... | 28.04.2014 20:54 |
| added pandas | 58f372d | Karl Traunmüller... | 27.04.2014 20:18 |
| added padding to output cells | a263616 | Karl Traunmüller... | 24.04.2014 00:14 |
| fixed a crash when image output is nil | 1c884c0 | Karl Traunmüller... | 24.04.2014 00:13 |
| toolbar positioning fixes | 627e0df | Karl Traunmüller... | 24.04.2014 00:13 |
| using matplotlib rcParams for Retina handling | 920eae7 | Karl Traunmüller... | 24.04.2014 00:13 |
| added sympy | ef1e7b9 | Karl Traunmüller... | 24.04.2014 00:12 |
| added basic notebook parsing | ecc82bb | Karl Traunmüller... | 21.03.2014 00:18 |
| we have image output!!! | 1fa6cf2 | Karl Traunmüller... | 14.03.2014 01:11 |
| we get display data! | f98ed08 | Karl Traunmüller... | 14.03.2014 00:34 |
| zmq message buffer now allocated on heap | 1333c8a | Karl Traunmüller... | 13.03.2014 01:24 |
| numpy and matplotlib seem to be ok now | b49d8a2 | Karl Traunmüller... | 13.03.2014 00:27 |
| added faulthandler | d531469 | Karl Traunmüller... | 10.03.2014 01:21 |
| fixed matplotlib dependencies | 4c12fe6 | Karl Traunmüller... | 10.03.2014 00:06 |
| added matplotlib, libbz2 | fbdcf51 | Karl Traunmüller... | 09.03.2014 23:11 |
| fixed libscipy | 549fbb7 | Karl Traunmüller... | 09.03.2014 22:36 |

# The Computable Icon

- We'll need that for submission

- Make it create its own icon :)

$$\frac{z}{cos(1 - z^{0.4})}$$

→

# App Store Submission

- It's end of June

  - 5 months of late-nighters

- You're exhausted when reaching 1.0

- Your build system shows last-minute flaws

- What do you do?

  - Submit a fully unlocked beta build!

  - http://bit.ly/1mOCunh

# Computable 1.1

- Python code completions

- Inline documentation

- Important step towards better usability

# Computable 1.2

- Get rid of HTML and MathJax

- Use (La)TeX for all Markdown rendering

- Status

  - Plain TeX + DVI rendering up & running

    - ~10x faster than previous approach

  - Currently working on LaTeX

- Paginated PDF export as nice byproduct

# Computable 1.2

```objc
@implementation DviRenderer

- (void)renderPage:(Page *)page intoContext:(CGContextRef)context
{
    CGContextSetAllowsAntialiasing(context, true);
    CGContextSetAllowsFontSmoothing(context, true);
    CGContextSetAllowsFontSubpixelPositioning(context, true);
    CGContextSetShouldSubpixelPositionFonts(context, true);
    CGContextSetAllowsFontSubpixelQuantization(context, true);
    CGContextSetShouldSubpixelQuantizeFonts(context, true);

    CGContextSetRGBFillColor(context, 0, 0, 0, 1);
    CGContextSetTextDrawingMode(context, kCGTextFill);
    CGContextSetTextMatrix(context, CGAffineTransformMakeScale(1, -1));

    for (Run *run in page.runs)
    {
        [self renderRun:run intoContext:context];
    }
}

- (void)renderRun:(Run *)run intoContext:(CGContextRef)context
{
    CGContextSetFont(context, run.cgFont);
    CGContextSetFontSize(context, CTFontGetSize(run.ctFont) * kFontScale);
    CGContextShowGlyphsAtPositions(context, run.glyphs, run.positions, run.length);
}

@end
```

# Computable 1.3

- More Python packages
  - scikit-learn, seaborn, plotly, ...

# Future

- Computable for R
  - would be great, but GPL
- Computable for Julia?
- LaTeX for iPad (and Mac)?

# Looking for a job?



- runtastic is hiring!

- Senior iOS developer
  https://www.runtastic.com/en/career

- Contact Stefan Damm std@runtastic.com