EZ-VORTEX DOCUMENTATION:
a Slender Vortex Filament solver

I. General

This package uses OpenGL for 3D rendering and has been optimized for use on SGI workstations. By using the Mesa library (public domain implementation of most OpenGL routines) it should be possible to run on virtually any machine supporting X. EZ-Vortex could be run on a PC with the Linux operating system. See http://mesa3d.sourceforge.net/ Note, you can run without graphics, but you must have OpenGL (or Mesa) header files and libraries to use the code.

This code is adapted from EZ-Scroll a Code for Simulating Scroll Waves developped by Dwight Barkley (http://www.maths.warwick.ac.uk/∼barkley/ez_software.html) with courtesy of Dwight Barkley (barkley@maths.warwick.ac.uk, Copyright (C) 1998). The EZ-Vortex package (in particular this document) is under development. There are aspects of the code which ones may not be happy with, but to my knowledge everything works correctly.

The philosophy of the original code and this one is to keep programs as simple as possible and to provide documentation by way of comments within the code itself. The user is expected to modify the programs according to his or her needs. The bulk of the package is devoted to graphics. Almost all of the execution time is spent in a loop in the routine `Step()` in *ezstep3d.c*.

The computational methods are described in more detail in the references at the end of this document. Ref. [1] describes the 3D implementation of the initial Code EZ-Scroll. Ref. [2] describes the asymptotic derivation of the Callegari and Ting equation of motion. Ref. [4,5] describes the equations and the computational methods. Ref. [3,6] describes the expansions of the Biot and Savart law. In Ref. [7] the non-simailar core equation are summarized. If you generate publications from using EZ-Vortex, I ask that you cite these papers [4,5].

II. Running EZ-Vortex

Files: You should have the following files:
ezvortex.c, ezstep3d.c, ezgraph3d.c, ezopengl.c, ezvortex.h, ezstep3d.h, ezgraph3d.h, ezopengl.h,

task.dat, ic.m, fc.m, history.m, and *Makefile.*
You will probably want to save copies of these files (in compressed tar format).

make: It is up to you to edit *Makefile* as necessary for your system. You can either compile using SGI C (cc) or else using GNU C (gcc). You may, if you wish, specify NP (the number of point on a filament) etc. at compile time. Then these will be ignored in the task file.

Note: On an SGI, using the SGI C compiler cc with -DNP etc gives the fastest execution. Using the GNU C compiler gcc with -DNP is slightly slower and gcc without -DNP is slightly slower still. Using cc without -DNP is terrible and should not be used, i.e. if you want to specify the number of grid points through the *task.dat* file, then you should to use GNU compiler and not the SGI C compiler. This is still being looked into.

Note: in our laboratory the LOADLIBES macro in the makefile is

- SGI machine (Berlioz):

  LOADLIBES = -lGL -lX11 -lXext -lm -I/usr/include

- linux machine (Liszt):

  LOADLIBES = -I/usr/X11R6/include -L/usr/X11R6/lib/ -lGL
  -L/usr/X11R6/lib/ -lX11 -lm

Make *ezvortex* by typing *make.* Then run by typing *ezvortex.* A window should open containing an initial condition for the vorticity field, i.e. one or several closed or open filaments. The centerlines of the filaments are plotted. Hitting the space bar in the EZ-Vortex window will start the simulation. This is a coarse resolution run showing the speed possible with EZ-Vortex simulations. With the pointer in the EZ-Vortex window, you can:

(1) Switch between curve display, worm display and no field by typing *c*, *w*, or *n* respectively. The worm display is a tube display with a thickness that allows to have a 3D view without rotating. Without rotating, no 3D view can be seen from the curve display. The tube radius is only geometrical: it is not linked to the physical radius which may change due to the global stretching or to diffusion.

(2) Pause the simulation by typing *p*, and resume by typing a *space.*

(3) Rotate the image by first pausing the simulation, then by holding down the left mouse button and moving the cursor.

(4) The key *r* resets the view to the initial (start up) view, and *z* sets the view to looking down the z-axis with the x- and y-axes in the usual orientation. This view is useful for moving the image.

(5) The arrow keys move the image in the x-y directions. The + key moves the image up the z-axis and the − moves it down the z-axis. Again, for moving the image it is best first to have set the view by typing *z*.

(6) Stop the simulation by typing:
q for soft termination with all files closed or
ESC for immediate termination without writing final conditions (equivalent to typing control-C from the shell).

(7) Snapshot the window (on a SGI workstation) by typing *s*

After a successful run, you will have a file *fc.dat* in your directory which contains the final conditions of the run. If you copy this file to *ic.dat*, then the next time you run *ezvortex*, this file will be read and used as an initial condition. (The files *ic_dat.m* and *fc_dat.m* are matlab files that can also be used to see initial and final conditions.)

IV. Equations

You have the choice between four equations of vortex filament motion which are implemented in *ezvortex.h* :

(1) The Callegari and Ting equation [2] (not stiff)

$$\partial \mathbf{X}/\partial t = \mathbf{A} + \frac{\Gamma K(s,t)}{4\pi}\left[-\log\epsilon + \log(S) - 1 + C_v(t) + C_w(t)\right]\mathbf{b}(s,t), \qquad (1)$$

where

$$\mathbf{A}(s,t) = \frac{\Gamma}{4\pi}\int_{-\pi}^{+\pi} \sigma(s+s',t)\left[\frac{\mathbf{t}(s+s',t)\times(\mathbf{X}(s,t)-\mathbf{X}(s+s',t))}{|\mathbf{X}(s,t)-\mathbf{X}(s+s',t)|^3} - \frac{K(s,t)\mathbf{b}(s,t)}{2\,|\lambda(s,s',t)|}\right]\mathrm{d}s',$$

and $\lambda(s,s',t) = \int_{s}^{s+s'} \sigma(s^*,t)\mathrm{d}s^*$. Here, $\mathbf{X}$, is the centerline of the filament, Γ is its circulation, K is local curvature, ϵ is the reduced radius of its core, S the length of the closed filament, $\sigma(s,t) = |\partial\mathbf{X}/\partial s|$, $(\mathbf{t},\mathbf{n},\mathbf{b})$ is the Frenet frame of the curve $\mathbf{X}$, and $C_v(t)$ and $C_w(t)$ are known functions which describe the orthoradial and axial evolution of the inner velocity in the core. The core structure can be of different forms:

(11) *A similar vortex core*: The circonferential and axial components of the relative velocity field for a similar vortex are

$$v^{(0)} = \frac{\Gamma}{2\pi\bar{r}}\left[1 - e^{-\left(\bar{r}/\bar{\delta}\right)^2}\right], \qquad w^{(0)} = \frac{m_0}{\pi\bar{\delta}^2}\left(\frac{S_0}{S}\right)^2 e^{-\left(\bar{r}/\bar{\delta}\right)^2},$$

where $\bar{r} = r/\epsilon$ is the stretched radial distance to the filament, $\bar{\delta} = \delta/\epsilon$ is the stretched radius, and m_0 is the initial axial flux of the ring. The stretched radius $\bar{\delta}$ is given by

$$\begin{aligned} \bar{\delta}^2(t) &= \bar{\delta}_0^2\left(\frac{S_0}{S(t)}\right)1_{\bar{\nu}} \\ 1_{\bar{\nu}} &= 1 + \frac{\bar{\delta}_{\bar{\nu}}^2}{\bar{\delta}_0^2}, \\ \bar{\delta}_{\bar{\nu}}^2 &= 4\bar{\nu}\int_0^t \frac{S(t')}{S_0}\mathrm{d}t', \end{aligned}$$

where $\bar{\nu} = \nu/\epsilon^2$ is the stretched kinematic viscosity. For the similar core structure as it is implemented in *ezstep3d.h* the inner functions are given by

$$\begin{aligned} C_v(t) &= [1 + \gamma - \ln 2]/2 - \ln(\bar{\delta}), \\ C_w(t) &= -2(S_0/S)^4(m_0/(\Gamma\bar{\delta}))^2, \end{aligned}$$

where γ is the Euler number. The effect of the diffusion is easy seen in these expressions through the term τ_α in $\bar{\delta}$ and the one of the stretching through the ratio S_0/S.

(12) *An inviscid vortex core*: If the fluid is inviscid ($\bar{\nu} = 0$) the circonferential and axial components of the relative velocity field are are in the form

$$\begin{aligned} v^{(0)} &= v_0(\eta)\left(S_0/S(t)\right)^{-1/2}, \\ w^{(0)} &= w_0(\eta)\left(S_0/S(t)\right), \end{aligned}$$

where S_0 is the initial length of the filament, $\eta = \bar{r}/\bar{\delta}$, and $(v_0(\eta), w_0(\eta))$ are the initial velocity fields. The ϵ-stretched radius $\bar{\delta}$ is

$$\bar{\delta}^2(t) = \bar{\delta}_0^2\left(S_0^{(0)}/S^{(0)}(t)\right).$$

The inner functions are given by

$$\begin{aligned} C_v(t) &= C_v(0) - \log\bar{\delta}(t), \\ C_w(t) &= C_w(0)\left(S_0^{(0)}/S^{(0)}(t)\right)^3, \end{aligned}$$

where $C_v(0)$ and $C_w(0)$ are the associated initial core constant.

(13) *A non-similar vortex core*: If the flow is viscous ($\bar{\nu} \neq 0$) and the core is not similar, the circonferential and axial components of the relative velocity field are are in the form [7]

$$\begin{aligned} v^{(0)} &= \frac{1}{\bar{\delta}}\frac{1}{\eta}\left[\frac{\Gamma}{2\pi}\left(1-\mathrm{e}^{-\eta^2}\right)+\mathrm{e}^{-\eta^2}\sum_{n=1}^{\infty}\bar{\delta}_0^2 D_n L_n(\eta^2)P_n(\eta^2)1_{\bar{\nu}}^{-n}\right], \\ w^{(0)} &= \frac{2}{\bar{\delta}^2}\left(\frac{S_0}{S(t)}\right)^2\left[\frac{m_0}{2\pi}\mathrm{e}^{-\eta^2}+\mathrm{e}^{-\eta^2}\sum_{n=1}^{\infty}\bar{\delta}_0^2 C_n L_n(\eta^2)1_{\bar{\nu}}^{-n}\right]. \end{aligned}$$

The stretched radius $\bar{\delta}$ given by

$$\begin{aligned} \bar{\delta}^2(t) &= \bar{\delta}_0^2\left(\frac{S_0}{S(t)}\right)1_{\bar{\nu}} \\ 1_{\bar{\nu}} &= 1+\frac{\bar{\delta}_{\bar{\nu}}^2}{\bar{\delta}_0^2}, \\ \bar{\delta}_{\bar{\nu}}^2 &= 4\bar{\nu}\int_0^t \frac{S(t')}{S_0}\mathrm{d}t', \end{aligned}$$

where $\bar{\delta}_{\bar{\nu}}$ is the diffusion-added ϵ-stretched thickness of the core. L_n are the Laguerre polynomials, $P_n(\eta^2) = L_{n-1}(\eta^2) - L_n(\eta^2)$, γ is the Euler's constant, and (C_n, D_n) are the Fourier components of the initial axial velocity and tangential vorticity

$$\begin{aligned} C_n &= \int_0^{\infty} w_0(\eta)L_n(\eta^2)\eta\mathrm{d}\eta, \\ D_n &= \int_0^{\infty} \zeta_0(\eta)L_n(\eta^2)\eta\mathrm{d}\eta, \end{aligned}$$

$C_0 = m_0/2\pi\bar{\delta}_0^2$, $D_0 = \Gamma/2\pi\bar{\delta}_0^2$, $C_1 = D_1 = 0$. For the non-similar core structure as it is implemented in *ezstep3d.h* the inner functions are given by[7]

$$\begin{aligned} C_v(t) &= -\log\bar{\delta}+\frac{1}{2}(1+\gamma-\log 2)+\frac{4\pi^2}{\Gamma^2}\sum_{(n,m)\in\mathbb{N}^2\setminus(0,0)}^{\infty}\frac{\bar{\delta}_0^4 D_n D_m A_{nm}}{n+m}1_{\bar{\nu}}^{-(n+m)}, \\ C_w(t) &= -\frac{2}{\bar{\delta}^2}\left(\frac{S_0}{S(t)}\right)^4\left[\frac{m_0^2}{\Gamma^2}+\frac{8\pi^2}{\Gamma^2}\sum_{(n,m)\in\mathbb{N}^2\setminus(0,0)}^{\infty}\bar{\delta}_0^4 C_n C_m A_{nm}1_{\bar{\nu}}^{-(n+m)}\right], \end{aligned}$$

where

$$\begin{aligned} A_{nm} &= \int_0^{\infty}\mathrm{e}^{-2x}L_n(x)L_m(x)\mathrm{d}x \\ &= \frac{(n+m)!}{n!\ m!\ 2^{m+n+1}}. \end{aligned}$$

(2) The Local Induction Approximation (LIA) equation (not stiff)

$$\partial\mathbf{X}/\partial t = \frac{\Gamma K(s,t)}{4\pi}\left[-\log\epsilon + \log(S) - 1 + C_v(t) + C_w(t)\right]\mathbf{b}(s,t), \tag{2}$$

(3) A simple de-singularized method (stiff)

$$\partial\mathbf{X}/\partial t = \frac{\Gamma}{4\pi}\int\limits_{\mathcal{C}} \sigma(s',t)\frac{\mathbf{t}(s',t)\times(\mathbf{X}(s,t)-\mathbf{X}(s',t))}{\left[|\mathbf{X}(s,t)-\mathbf{X}(s',t)|^2 + s_c^2\right]^{3/2}}\mathrm{d}s', \tag{3}$$

with

$$s_c(s,t) = \epsilon\exp\left(-C_v(t) - C_w(t)\right). \tag{4}$$

(4) The M1 de-singularized method of Knio and Klein (not stiff)[3,5]

$$\partial\mathbf{X}/\partial t = \mathbf{v}_{\sigma_1} + (\mathbf{v}_{\sigma_1} - \mathbf{v}_{\sigma_2})\frac{\log(\sigma_1/\delta^{ttm})}{\log(\sigma_2/\sigma_1)} \tag{5}$$

where

$$\mathbf{v}_x = \frac{\Gamma}{4\pi}\int\limits_{\mathcal{C}} \sigma(s',t)\frac{\mathbf{t}(s',t)\times(\mathbf{X}(s,t)-\mathbf{X}(s',t))}{\left[|\mathbf{X}(s,t)-\mathbf{X}(s',t)|^2\right]^{3/2}}\kappa\left(\frac{|\mathbf{X}(s,t)-\mathbf{X}(s',t)|}{x}\right)\mathrm{d}s', \tag{6}$$

with $\kappa(r) = \tanh(r^3)$ and

$$\begin{aligned}
\delta^{ttm} &= \epsilon\exp\left(C^{ttm} + 1 - C_v(t) - C_w(t)\right), && (7)\\
\sigma_1 &= 3\sigma_{max}, && (8)\\
\sigma_2 &= 2\sigma_{max}, && (9)\\
\sigma_{max} &= \mathrm{d}s \max_{s\in[0,2\pi]}\sigma(s,t). && (10)
\end{aligned}$$

With the choice of $\kappa(r) = \tanh(r^3)$, the C^{ttm} constant is $C^{ttm} = -0.4202$.

In case of several filaments the mutual induction velocity of the filament $\mathbf{X}_j$

$$\frac{\Gamma_j}{4\pi}\int\limits_{\mathcal{C}_j} \sigma_j(s',t)\frac{\mathbf{t}_j(s',t)\times(\mathbf{X}(s,t)-\mathbf{X}_j(s',t))}{|\mathbf{X}(s,t)-\mathbf{X}_j(s',t)|^3}\mathrm{d}s', \tag{11}$$

is added to the previous velocity.

In case of open filament, the filament is supposed to be periodical in the x direction. A bits of filament corresponding to an integer numbers of periods is advanced in time. The local induction at point $\mathbf{X}(s,t)$ on this filament is evaluated with either of the formulas (1), (2),

(3) or (4) as for a closed filament by having the location of this point at the center of a bit of filament with the integer numbers of periods. The induction of the missing part of the open filament is then evaluated with the mutual induction velocity formula (11) as if it were coming from other filaments.

V. Compilation Macros

The main compilation parameters are in *ezvortex.h*:

- You can choose the equation of motion whether the macros LOCAL_INDUCTION, CALL_AND_TING, DE_SINGU, and M1_KNIO_KLEIN are set to 1 or 0.
- You can choose closed or open filament whether the macro CLOSED is set to 1 or 0.
- You can have graphics or not at run time whether the macro GRAPHICS is set to 1 or 0.
- You can snapshot different history steps if the macro MOVIE is set to 1 (only on a SGI work station).
- You can run a *history.dat* file instead of computing whether the macro COMPUTE is set to 1 or 0.
- You can have a non-similar part of the core or not whether the macro NON_SIMIL_PART is set to 1 or 0. The macro SIMIL_PART has to be set to 1 in any case.
- You can have an uniform core by setting the macro SIMIL_PART to 0 and the macro UNIFORM_CORE to 1 (to be coherent you have to choose nu_bar=0 in *task.dat*).
- You can choose to have spectral spatial derivative or not by setting the macro SPECTRAL to 1 or 0.
- You can choose to have an explicit (order 1) time stepping with the macro EXPLICIT, an implicite one with the macro NEWTON, and an explicit Adams-Bashforth with the macro ADAMS_BASHFORTH. The explicit methods (order 1 or Adams Bashforth) can be performed either with a Jacobi iteration or a Gauss-Seidel one whether the macro GAUSS_SEIDEL is set to 0 or 1. The Callegari and Ting method and the LIA method are numerically instable with an explicit (order 1 or Adams Bashforth) scheme either

a Jacobi or a Gauss-Seidel stepping is used. The M1 Knio and Klein method is stable with an explicit scheme.

- You can test the convergence of the Biot-Savart velocity computation at initial time if the macro CONV_ANALYSE is set to 1. A file *look* is then generated with three columns associated to the three components of the velocity on the filament. The convergence can be assess with the number of point and also with the number of periodic boxes for open filaments.

- You can have an automatic motion of the graphic window in the z direction or not whether the macro MOVE is set to 1 or 0.

VI. Parameters

The parameters are either set in the routine Generate_ic() at the end of *ezvortex.c* when these vortices are initially created or in the file *task.dat*. The parameters of *task.dat* are loaded at the beginning of a run and so you need not to re-compile *ezvortex* when these parameters are changed.

In the case of similar vortex filaments delta_0_bar_param is the initial stretched core radius $\bar{\delta}_0$, m_0_param is the initial axial flux m_0, gamma_param is the circulation Γ, and epsilon is the reduced thickness ϵ. They are parameters of the core structure. nu_bar is the stretched viscosity $\bar{\nu}$ of the fluid. Thus the "physical" parameters in the simulation are: delta_0_bar_param, m_0_param, gamma_param, epsilon, nu_bar and nf the number of filaments. The parameters delta_0_bar_param, m_0_param, gamma_param are selected for each filaments in the routine Generate_ic() at the end of *ezvortex.c* when these vortices are initially created. The parameters epsilon and nu_bar are set in the file *task.dat*.

The "numerical" parameters for the simulation are: np = number of spatial points in each filament, ts = time step, n_b = number of periodic boxes (for open filaments) and are also set in the file *task.dat*.

The other parameters set in *task.dat* are:

Number of time steps to take

Time steps per plot. Also set the number of time steps per filament computation.

error_stop. Error to stop the Newton iteration for the implicit method.

Time steps per write to history file and of snapshot if MOVIE is set to 1 in *ezvortex.h*

initial field display : curve or worm

initial condition type : from 0 to 12.

output type : ascii or binary

verbose

These are more or less self-explanatory.

If Time steps per write is non-zero then the filament data will be written to a file (*history.dat*) every Time steps per write (whether or not there is any graphics) which can be executed by *ezvortex* with the macro COMPUTE set to 1. (The filaments will also be saved every Time steps per write to a file (*history_dat.m*) which can be executed with matlab.)

I leave it to you to look at different initial condition types at the end of *task.dat* and *ezvortex.c*. The number of filaments nf set has to be coherent with the initial condition chosen. You can choose between :

(0) Oscillations of a Vortex Ring (Standing wave)

(1) Oscillations of an ellipse in a plan y-z

(2) Oscillations of a Vortex Ring (Travelling wave)

(3) Oscillations of a triangle Vortex Ring

(4) Motion of a Lissajous ring

(5) Motion of two side by side vortex rings

(6) Leap frogging of two vortex rings

(7) Motion of two face to face (and shifted) vortex rings

(8) Motion of two face to face vortex rings

(9) Motion of two linked vortex rings

(10) Motion of two vortex rings

(11) Oscillations of a straight filament

(12) Motion of a helical filament

(13) Oscillation of two trailling vortices

(14) Crow instability of two trailling vortices

(15) Crow instability of two trailling vortices (initialy at most instable angle)

The other place to look for "parameters" is in the header files. The main compilation parameters are in *ezvortex.h*. Many of the macro definitions in the other header files can be replaced with variables.

VII. Convergence and value of numerical parameters

In this section we give the values of the numerical parameter that give converged numerical results for the different configurations under consideration at the end of the previous section.

For all following simulations (M1 method with Adams Bashforth), we took:

delta_0_bar_param= 1, m_0_param = 0, nu_bar =0.

(1) Oscillations of a Vortex Ring (Standing waves)
gamma_param=1, epsilon=0.1,
ρ= 0.01, mode 3
np= 101, ts = 0.0016, nsteps = 7000
computational time (SGI R10000 225MHz)=

(2) Oscillations of a straight filament
gamma_param=$\pm$1, epsilon=0.1,
ρ= 0.01, wave_length= 1.25
np= 101, ts = 0.00065, n_b = 4, nsteps = 7000
computational time =

(3) Oscillation of two trailling vortices
gamma_param=$\pm$1, epsilon=0.1,
ρ= 0.01, wave_length= 1.25
np= 101, ts = 0.00065, n_b = 4, nsteps = 8000
computational time =

(4) Crow instability of two trailling vortices
gamma_param=$\pm$1, epsilon=0.02,
ρ= 0.01, wave_length= 10.21, theta= 47.40 degree
np= 257, ts = 0.00019, n_b =4, nsteps = 795
computational time =

VIII. Indices and implementing variables

Coordinates of nodes i on the filament j are stored in the pointer u and are managed by the three macos $Ux(i,j), Uy(i,j), Uz(i,j)$. The same kind of pointer (u_s and u_{ss}) and macros are used for the first and second derivatives or for σ and the velocity components. The index i range from 0 to $np+1$ and the index j from 0 to $nf-1$.

(1) Closed filament:

On a closed filament, the point of index np is at the same location as the point of index 1. Points 0 and $np+1$ are added fictious points which may be of use. To advance all the points, the velocity has to be cumputed from $i=0$ to $i=np-1$.

(2) Open filament:

On an open filament, the point of index np is the periodic point associated to the point of index 1. Points 0 and $np+1$ are added fictious points which may be of use. To advance all the points, the velocity has to be cumputed from $i=0$ to $i=np$.

The number of points np has to be an odd number. The point of index $(np+1)/2$ is the middle point. In the spectral methods the only points of use are the non periodic points, i.e. the points from $i=1$ to $i=np-1$. The index $np-1$ is an even number and has to be 256.

To find the velocity at any point i, the temporary pointers $(ux_{tmp}, uy_{tmp}, uz_{tmp})$ are used to locate this point at the central index $(np+1)/2$ of these pointers and remaining points of these pointers are filled with related points. By using these pointers, the same procedure is then used to compute the velocity whatever point is under consideration. The same trick is used for open filament. The procedure to fill the $(ux_{tmp}, uy_{tmp}, uz_{tmp})$ pointers is different whether the filament is closed or open because indices have to be managed differently. For open filaments the induced velocity of translating points of the $(ux_{tmp}, uy_{tmp}, uz_{tmp})$ positions (n_b copies on the left and on the right) is added.

References

[1] M. Dowle, R.M. Mantel, and D. Barkley, *Fast simulations of waves in three-dimensional excitable media*, Int. J. Bif. Chaos, 7(11) 2529–2546 (1997).

[2] A.J. Callegari and L. Ting, *Motion of a curved vortex filament with decaying vortical core and axial velocity*, SIAM J. Appl. Math. 35 (1) 1978 pp. 148-175

[3] R. Klein and O.M. Knio : *Asymptotic Vorticity Structure and Numerical Simulation of Slender Vortex Filaments*, J. Fluid Mech.,284, pp.257-321,1995

[4] D. Margerit, P. Brancher, A. Giovannini, *Implementation and validation of a slender vortex filament code: Its application to the study of a four-vortex wake model*, International Journal of Numerical Methods in Fluids, Vol 44, N 2, pp. 175-196, 2004

[5] O.M. Knio and R. Klein, *Improved thin tube models for slender vortex simulations*, J. Comput. Phys., p. 68-82, 2000

[6] D. Margerit and J-P. Brancher : *Asymptotic Expansions of the Biot-Savart law for a slender vortex with core variation*, to appear in Journal of Engineering Mathematics

[7] D. Margerit: *Axial core-variation of axisymmetric shape on a curved slender vortex filament with a similar, Rankine, or bubble core*, Phys. of Fluids vol 14, n 12, pp.4406-4428, 2002

Please send comments to daniel.margerit@free.fr