Simulation of a high proton temperature plasma toroidal magnetic trap to be used in proton-^{11}B fusion

J. L. Lopez Segura[1], N. Urgoiti Moinot[1], E. Lazzaro[2]

[1] *Advanced Ignition SL* C/francisco Medina Y Mendoza, Parcela 1, Poligono 19171 - (Cabanillas Del Campo) - Guadalajara*, Spain*
[2] Istituto Fisica del Plasma CNR, Via R. Cozzi 53, 20125 Milano, Italy

(Received: 02. June 2019, Accepted: 19. Nov. 2019, Published online: 20. Nov. 2019)

Several tokamaks-like toroidal magnetic confinement structures have been simulated using 500keV protons to be used in P-B11 fusion. In order to find the optimal confinement configuration, the simulation was carried out by an evolutionary algorithm running 145,000 simulations whose results are presented in this document including the feasibility to reach ignition with some of them by accelerating and colliding ^{11}B and proton ions.

(DOI: 10.31281/jtsp.v1i1.6)

jlopez@advancedignition.eu

I. Introduction

After extensive testing of the Pulsotron-2A z-pinch device and several ion accelerators using the Pulsotron-2B device it was found that it is necessary to design fusion devices that reduce the electron temperature to decrease loses [14] [15] [16]. The Pulsotron II 2A device consists of a small Z-pinch device and the Pulsotron II 2B represents a modification of a capacitor bank to be used in several tests related to plasma confinement. The Z-pinch successfully achieved the required pressure but with a low plasma temperature. In order to design a boron-proton fusion reactor it was designed an electrostatic and a magnetic ion accelerator but the design of a confinement chamber is still needed. In the present work the results reached after simulating 145,000 different magnetic toroidal Tokamak-like design are presented. The chamber is designed to sustain boron-proton fusion. The Boron fuel state will be Ions with one or more positive charges B.

The Larmor radius of the protons is taken to be the major radius R instead of the minor one that would require much larger magnetic fields. Boron can be injected at low speed but must be confined within the higher concentration of protons. In this first simulation campaign, only protons are used in order to determine which reactor configuration is the best candidate. The losses are calculated from the ion acceleration and a Larmor radius equal to chamber major radius for a confinement time of 0.59 ms for a small 200 mm reactor, so losses are about 5.47 x 10^{-24} watts according to the excel table found in [17], formula 41, so less than one eV is lost by the protons during their confinement. Defining the Larmor radius as the major radius of the chamber the reactor chamber dimensions can be greatly reduced as, for example, a 500 keV proton whose mass is 1.66 x 10^{-27} kg has an approximate speed of 9.8222 x 10^{6} m/s, so when submitted to a 1.5 Tesla magnetic field (easily achievable in a small reactor), the Larmor radius decreases to: $R = \frac{mv}{qB} = 6.787x10^{-3}m$. Hence, a proton plasma can theoretically be confined in a small 135 mm diameter torus. In the simulation the particles leaving the torus are taken into account as losses. In real toroidal systems, coils would be installed to recover electricity from lost ions and to slow them down to non-harmful kinetic energy levels. The main coils would receive some proton flux, but this was not calculated in this round of simulations. In the simulation copper wires are used instead of superconductive coils that would suffer less from ion bombardment.

The chamber is designed for high temperature aneutronic fusion. To deal with the extremely high ion speed a new configuration design is used. Additionally, a new parameter A_{eff} gives the effective cross section of the reactor that can be used to obtain the percentage of reacting particles inside the reactor in a direct way. C++ source code used in the simulation is also provided which equations are described in more detail in [2, 3, 5]. The main objective was to find the feasibility of using a tokamak-like configuration to allow the fusion reactions inside. The configuration must allow the installation of direct electricity recovery coils to be used to extract currents from the generated alpha particles.

Particles energy was assumed to be 500±12 keV which is close to the maximum cross section [5] with angle distribution of ± 20° obtained from elastic scattering [9-11, 6].

II. Equations and algorithms used

The percentage of the particles that react can be obtained using the following formula:

$$R = N * (1 - e^{-N*t*\frac{\sigma}{Aeff}}) \qquad (1)$$

Where:
N is the number of particles (protons or borons)
σ=cross section of the P-^{11}B reaction, A_{eff}=Effective cross section of the reaction, t = average laps number of particles paths. All areas are taken in square meters.

The maximum cross section for proton-^{11}B fusion is 0.8 x 10^{-28} m^2 (0.8 barns) accordingly EXFOR database [4]:

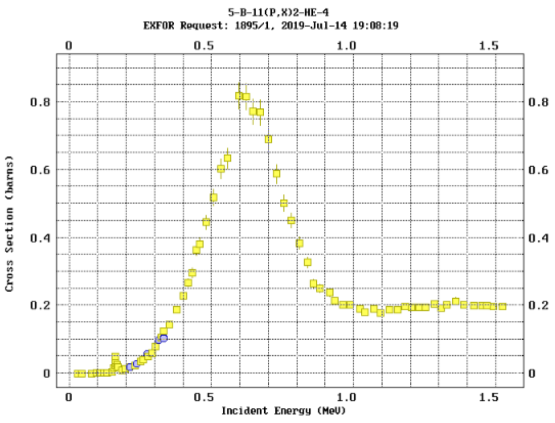

Figure1: Cross section of the Boron11-Proton fusion.

The exponential part of the reactor equation (1) must be about 1 in order to allow the reactions between most of the particles, so Eq. (1) can be used to calculate how many particles must be injected into the chamber. In order to have highly accurate and speedy simulations the C++ multithread technology was used to run high speed algorithms that apply elliptic integrals. As reference equations and codes collected from the book by J. D. Jackson [3] were used. The mutual inductance and inductance calculations uses Maxwell's Method [2] and the equations can be can be found at [13]. The obtained results of inductance and magnetic field were checked against the general equations of magnetic fields for standard devices, such as loops and wires. Furthermore, test devices were built and tested using standard magnetic field sensors, oscilloscopes and current sensors. With only one hundred simulations the achieved A_{eff} was too low, so an increase in the number of simulations to more than 10 thousands of tokamak-like configurations. To do that an initially low particle number of 16 was simulated and subsequently increased to 196 particles in those toroidal configurations that achieved longer confinement time and lower $A_{eff.}$ Also the genetic algorithm was used to generate vectors of data input in the more promising reactors.

II.a) Validity of Eq. (1)

The validity of Eq. (1) exponential part can be easily demonstrated using a small excel table using as example Nσ/A=0.02 where the remainder particles reacts each other every turn. As can be seen the exponential equation simplifies results:

N*σ/A	0.02		
Turns	remainder particles	Reacted particles	exp(-Nσ/A)
0	1	0.02	1
1	0.98	0.0196	0.9802
2	0.9604	0.019208	0.9608
3	0.9412	0.018824	0.9418
4	0.9224	0.018447	0.9231
5	0.9039	0.018078	0.9048
6	0.8858	0.017717	0.8869
7	0.8681	0.017363	0.8694
8	0.8508	0.017015	0.8521
9	0.8337	0.016675	0.8353
10	0.8171	0.016341	0.8187
11	0.8007	0.016015	0.8025
12	0.7847	0.015694	0.7866

Table. 1: Comparison of exponential part of Eq. (1) to obtain the remainder of particles.

III. Assumptions

The particles are injected when the magnetic field is saturated so there are only slow magnetic field variations. According to the simulations the maximum B-field is reached between 5 milliseconds and some seconds to values that ranges from 0.43 teslas for larger coils to 1.6 teslas for smaller ones.

The particles energy is 500±12keV according to the Miranda particles injector specifications. The particles angle distribution is ±20º, according to our simulations taking into account kinetics scattering data obtained from [6, 10, 11]. The magnetic field was simulated in different configurations by using Biot-Savart's law integrating along every coil. In order to calculate the inductance with high accuracy during short time elliptic integrals were applied along with the multithread technology. Using fixed time and length steps generates large errors in the coils proximity so the spatial increment in the proximity to the coils has to be narrowed down from one millimetre to less than 1 micrometre.

IV. Simulation Setup

In order to have high accurate simulations and speed the C++ multithread technology was used to run the algorithm, which works on the basis of elliptic integrals.

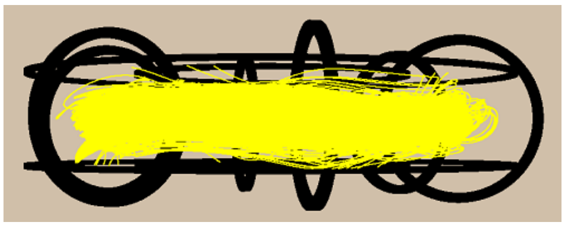
Figure 2: Plot of particles running inside a 280mm diameter Tokamak of 6 toroidal coil

A large amount of different configurations were simulated using only 16 particles at the beginning but the best results were obtained with 196 simulation particles.

IV.a) Input data

The main input data are the dimensions of coils, number of turns t, energy injected and proton energy (550 keV). Simulations were performed with 2 to 20 toroidal coils and up to 4 parallel coils (but more coils can be set).

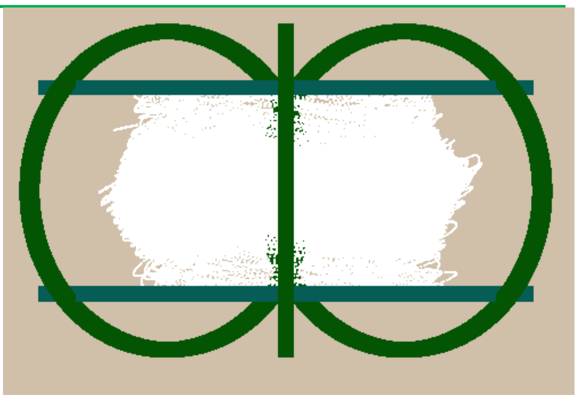
Figure 3: Plot of particles running inside a 280mm diameter Tokamak of 4 toroidal coil.

V. Performance of the simulator

Using low particle simulations allows a reduction in simulation time from some minutes per reactor to 10 tokamak configurations per second. If a particle passes close to a coil, the magnetic field gradient is higher so more calculation points are taken and the calculus is slower. One software thread per particle was used, so when a particle escapes to the reactor wall its thread is terminated. Hence, a better confinement in the reactor increases the escape time for particles and the simulation runs slower. The time consumption for one simulation is directly proportional to the particle number and the confinement time of the particles. It turns out that with only 9, 16 or 25 particles one can have similar results as with 196 or more particles. Hence, such preliminary results can be used for comparison purposes to filter out the best reactors before using a larger number of particles as can be seen in following graphs:

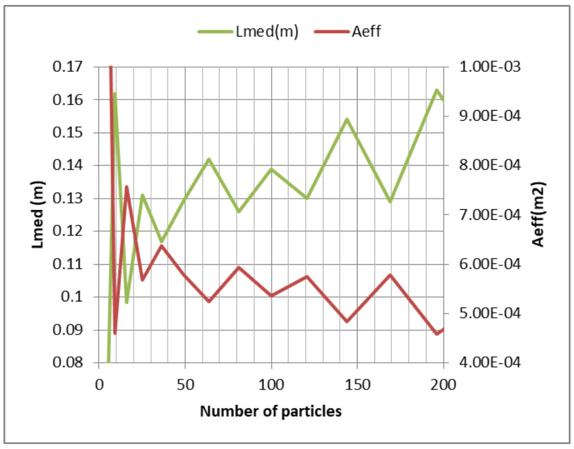

Figure 4: Simulation using up to 200 particles.

By using more particles the simulation result stabilizes:

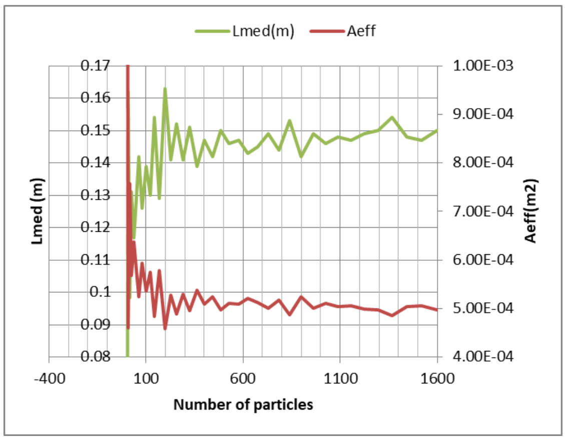

Figure 5: Simulation using up to 1600 particles.

Where L_{med} is the average particle path length of a proton before it escapes to the container walls. The best reactors have particles travelling up to 800 meters (see Fig 8).

VI. Simulation results

Initial results demonstrate that the A_{eff} was so high that very few particles could react within the confinement region. As can be seen from the following data in Table 2, A_{eff} was larger than 2 x 10^{-3} m² for tokamaks of 0.32 meters major radius, so the exponent part was so low that only one out of every 5 x 10^{8} particles reacts with another after injecting 100 kJ of input energy.

Rmax	zmed	r	n1	n2	E0	Lout(m)	Lmax(m)	I (A)	Aeff
0.16	0.0425	0.0005	29	30	25728.5	0.61	2.75	3237	8.702E-03
0.16	0.0425	0.0005	30	30	24980.1	0.60	2.74	3129	8.905E-03
0.16	0.0425	0.0005	31	30	24302.4	0.59	2.74	3028	9.102E-03
0.16	0.0425	0.0005	32	30	23686.9	0.57	2.74	2934	9.309E-03
0.16	0.0425	0.0005	33	30	23126.2	0.57	2.74	2845	9.441E-03
0.16	0.0425	0.0005	34	30	22613.9	0.56	2.74	2761	9.594E-03
0.16	0.0425	0.0005	35	30	22144.6	0.55	2.74	2682	9.698E-03
0.16	0.0425	0.0005	36	30	21713.6	0.54	2.73	2608	9.841E-03
0.16	0.0425	0.0005	37	30	21316.9	0.53	2.73	2537	9.989E-03
0.16	0.04	0.0005	29	30	21328	0.55	2.92	2983	8.710E-03
0.16	0.04	0.0005	30	30	20734.1	0.54	2.92	2883	8.887E-03
0.16	0.04	0.0005	31	30	20196.4	0.53	2.92	2790	9.070E-03
0.16	0.04	0.0005	32	30	19708	0.53	2.91	2703	9.191E-03
0.16	0.04	0.0005	33	30	19263	0.52	2.91	2621	9.298E-03
0.16	0.04	0.0005	34	30	18856.5	0.97	92.23	2544	4.964E-03
0.16	0.04	0.0005	35	30	18484	0.91	82.28	2471	5.280E-03
0.16	0.04	0.0005	36	30	18142	0.99	98.03	2403	4.874E-03
0.16	0.04	0.0005	37	30	17827.1	0.72	44.83	2338	6.740E-03
0.16	0.0375	0.0005	29	30	17526	1.03	105.85	2737	4.203E-03
0.16	0.0375	0.0005	30	30	17060.4	0.94	89.92	2646	4.601E-03

Table 2: High A_{eff} reactor data.

Only after 6000 simulations the cross section went lower than 3 x 10^{-4} as shown in the following Fig. 6:

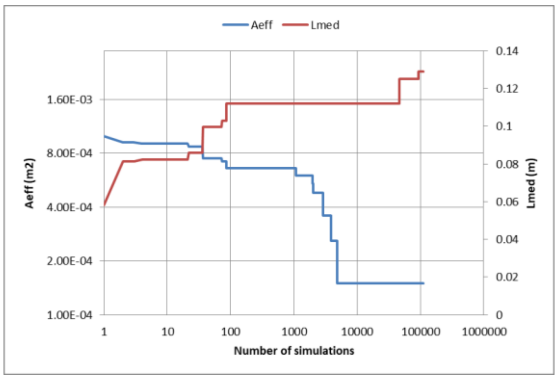

Figure 6: A_{eff} as a function of number of simulations before improvements.

In order to achieve better results algorithms that are used in Artificial Intelligence as genetic algorithms were applied. They give major improvements as the "23 fellows system" that consists basically of a selection of the 23 best matches and makes variations over them. The algorithm does not use random input data but lowers A_{eff} from a run to the next linearly. The results improved A_{eff} from 2 x 10^{-4} to 10^{-6} m^2, which was a 200-fold improvement but was still not good enough. The trajectories of the particles were simulated and it turns out that in some reactors they pass through very well confined areas as depicted in the following Fig. 7:

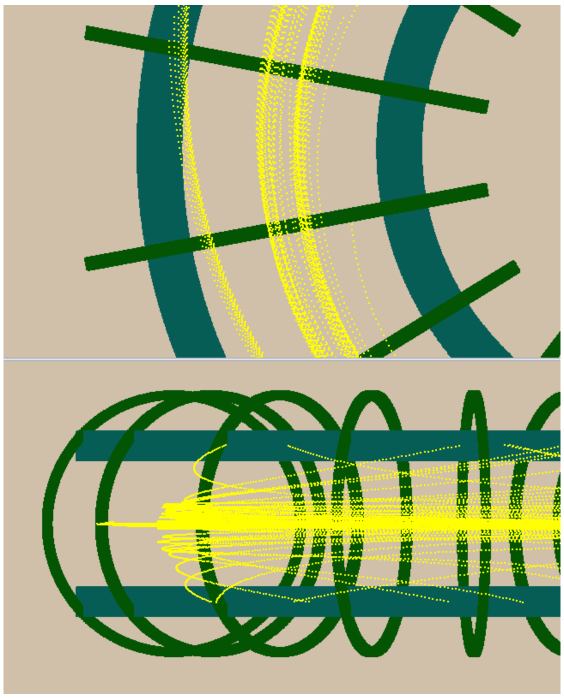

Figure 7: Plot of particles trajectory inside a reactor chamber obtained from a simulation run.

Thus, the cross section of the particles in 3 dimensions was included into the simulations. This increases the computational time but giving very good results, even when using only 16 particles. Afterwards 196 particles were used to obtain a highly accurate results improving from cross sections between 10^{-7} to 2 x 10^{-3} to cross sections between 4.32 x 10^{-12} to 10^{-24} m². With those simulations of 1660 reactors were performed. Out of these 1660 reactor configurations 1466 obtained a net energy gain of more than 4 times the input power. 784 reactor configurations obtained almost the maximum energy gain of 12.64. The reactor dimension was 0.13 to 0.47 meters in diameter and the energy range for confinement from 75 J to 360 kJ. The output power depends on how many times per minutes the reactor is fired. In the following plot (Fig. 8) the improvement of the simulations with respect to the number of simulations using the real cross section and also using the "23 fellow" genetic algorithm is summarized:

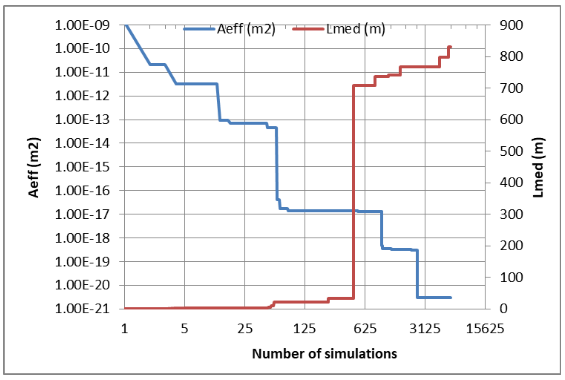

Figure 8: A_{eff} as a function of number of simulations after the improvements in simulations.

These results can be used for other types of aneutronic fusion reactions but a very accurate design must be done.

VI.a) Simulation results as a function of the toroidal coil number

In the following Fig. 9 A_{eff} is depicted as a function of the number of toroidal coils:

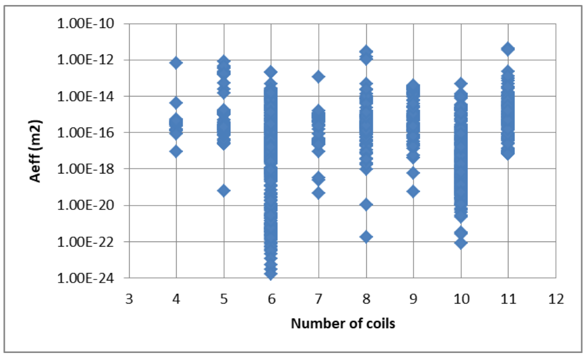

Figure 9: A_{eff} (m²) as a function of toroidal coils number. It was tried from 3 to 19 coils/reactor but only between 5 and 11 coils reach ignition conditions

VI.b) Reactor cross section as a function of chamber dimensions

The following Fig. 10 shows A_{eff} as a function of the external radius of the reactor. The dimensions of the reactors selected by the genetic algorithm showed that the best reactor have specific dimensions. It is not necessary to build large reactors:

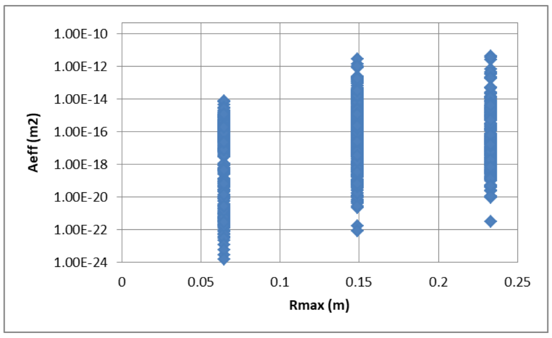

Figure 10: A_{eff} as a function of the external chamber radius. In the vertical axis there is A^{eff} (m^2) and in the horizontal axis the radius (m). The algorithm was commanded to try between 35 and 250mm radius tokamaks, finding the best of them at three discrete chamber radius.

VI.c) Reactor cross section as a function of injected energy

The performance of A_{eff} as a function of the optimal energy injected. As it can be seen from Fig. 11 some reactors need very low energy levels. It is not shown in Table 2, but the simulations prove that low energy reactors have a very low path length with very thin particle trajectory cross sections so a very accurate reactor design must be implemented.

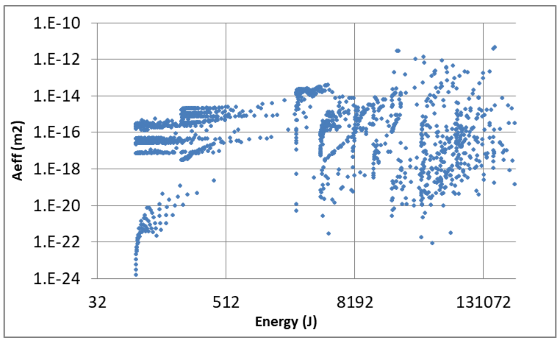

Figure. 11: A_{eff} as a function of the injected energy

VI.d) Travelled particle distance as a function of injected energy

In the following two Figs. 12 (linear scales) and 13 (logarithmic scale) the simulation results for the average distance travelled by the particles as a function of the injected energy are plotted. As can be seen the low energy systems confine the particles for a shorter distance (and also shorter time):

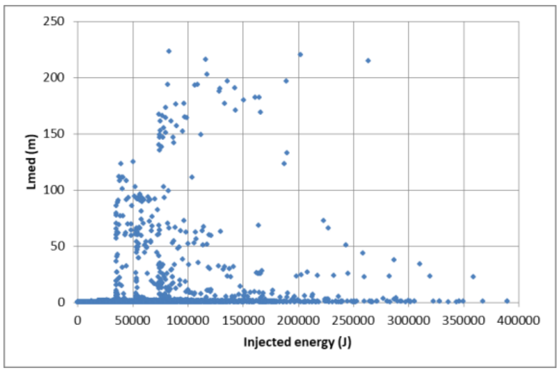

Figure 12: Average travelled distance in meters as a function of injected energy

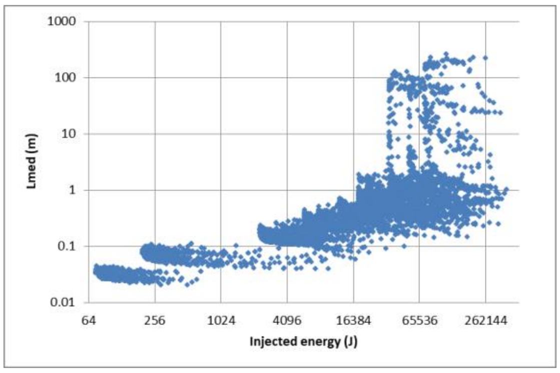

Figure 13: Average travelled distance in meters as a function of injected energy (logarithmic scale).

VI.e) Reactor magnetic field as a function of the reactor dimensions

The magnetic field is inverse proportional to the chamber radius and not very high for this kind of plasmas and some of them could be achievable without superconductors:

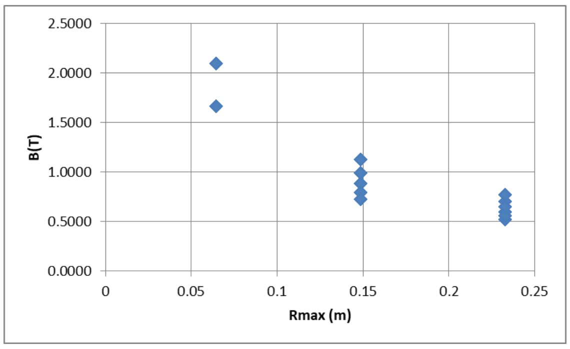

Figure 14: Confinement magnetic field as a function of reactor external dimension.

VI.f) Reactor energy as a function of the reactor dimensions

Obviously, the injected energy must be proportional to the reactor dimensions, but it can be seen that a large range of energies are allowed for every reactor size as shown in Fig. 14:

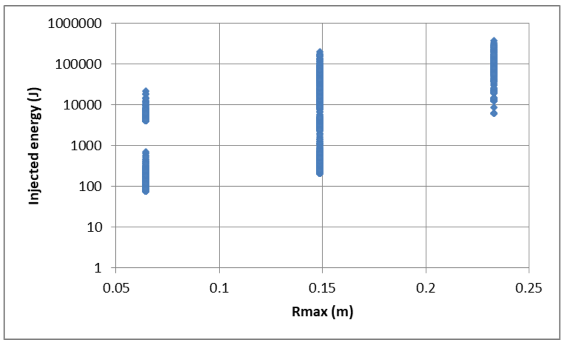

Figure 15: Injected energy as a function of reactor external dimension.

VII. Conclusion

We were able to show in this paper with extensive computer simulations that our Pulsotron tokamak design for aneutronic proton-boron fusion can reach energetic break-even. Furthermore, our numerical simulations reveal that it is also possible to construct small scale fusion devices by using hot protons with kinetic energies around 500 keV and comparably small electron temperatures.

VIII. References

[1] J. Marshall. "Acceleration of plasma into vacuum." *J. Nucl. Energy (1954)* 7.3-4 (1958): 276-276.

[2] A. C. M. de Queiroz Mutual Inductance and Inductance Calculations by Maxwell's Method, Homepage of C. M. de Queiroz, https://deanostoybox.com/hot-streamer/TeslaCoils/OtherPapers/Antonio/maxwell.pdf (2003), Version of July 2019.

[3] J. D. Jackson, "Classical electrodynamics." *American Institute of Physics* 15.11 (2009): 62-65. Chapter 5.5. Magnetic Induction of a magnetic loop of current

[4] J. M. Davidson et al. "Low Energy Cross Sections for ^{11}B (p,3α)."Nucl. Phys. (1979), Section A; Vol.315, p.253. DOI: 10.1016/0375-9474(79)90647-X

[5] J.L. Lopez Segura "Coil off axis magnetic field using elliptic integrals and Maxwell method with C++ code", Technical Report (2014)

[6] A. Gallmann et al. "B11(d,p)B12 Angular Distributions at Ed=5.5 MeV for the B12 2.62- and 2.72 MeV Levels" Phys. Rev. 138(3):560-568 (1965). DOI: 10.1103/PhysRev.138.B560

[7] S. Eliezer et al., "Avalanche proton-boron fusion based on elastic nuclear collisions." Phys. Plasmas 23, 050704 (2016). DOI: 10.1063/1.4950824

[8] X. Guo "Kinetic advantage of controlled intermediate nuclear fusion", AIP Conference Proceedings 1479, 2407 (2012). DOI: 10.1063/1.4756680

[9] P. M. Endt et al., "Angular distributions of four proton groups from the B10(d, p)B11 reaction" Physica Vol. **18**, 6–7, 423-428. (1952) DOI: 10.1016/S0031-8914(52)80075-8

[10] M. C. Spraker et al., "The 11B(p,α)8Be →α + α and the ^{11}B(α,α)^{11}B Reactions at Energies Below 5.4 MeV", J. Fusion Energy, Vol. **31**, Issue 4, (2012), pp 357–367. DOI: 10.1007/s10894-011-9473-5

[11] G. W. Tautfest and S. Rubin, "Elastic Scattering of Protons from B^{11} and N^{14}" Phys. Rev. 103(1):196-199, (1956). DOI: 10.1103/PhysRev.103.196

[12] A. G. Ruggiero Nuclear Fusion of Protons with Boron. Prospects for Heavy Ion Inertial Fusion (1992).

[13] E. B. Rosa, “The Self and Mutual Inductances of Linear Conductors,” Bulletin of Bureau of Standards, Vol.4.No.2 (1908)

[14] 2014-April Pulsotron-2 test report, J. Lopez

[15] Pulsotron-2 ignition conditions verification. J. Lopez (2013)

[16] January 2014 Pulsotron-2 test report. J. Lopez

[17] Useful formula and excel tables for plasma physics V08, J. Lopez (2018)

IX. Annex: Code samples

The following represents the C++ code sample of the main functions used to calculate the magnetic field in the simulations. The code uses the formula obtained using the Law of Biot Savart, integrated over a circular current loop to obtain the magnetic field at any point in space. Its result was compared using the on axis formula result (as shown in Fig. 16).

$$B_x = B_0 \frac{1}{\pi\sqrt{Q}}\left[E(k)\frac{1-\alpha^2-\beta^2}{Q-4\alpha}+K(k)\right] \qquad (2)$$

$$B_r = B_0 \frac{\gamma}{\pi\sqrt{Q}}\left[E(k)\frac{1+\alpha^2+\beta^2}{Q-4\alpha}-K(k)\right] \qquad (3)$$

A=r/a β=x/a γ=z/r (4)

$$Q = [(1-\alpha)^2 + \beta^2] \qquad (5)$$

$$k = \sqrt{\frac{4\alpha}{Q}} \qquad (6)$$

Where B_0 is the magnetic field at the coil centre: $B_0=\mu_0 I/(2a)$
K(k) is the complete elliptic integral function, of the first kind
E(k) is the complete elliptic integral function, of the second kind.

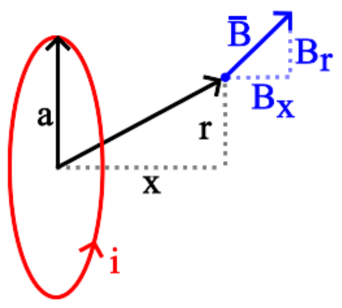

Figure 16: Schematics of the integration domain used in the simulations.

```
struct s_coil { vector<double> xyz, dxyz;
double R, r, current,num; double R2; };
```

```
//410. Calculates the field due to "coils" at
"point" due a currrent passing through them
void s_ccoil::halla_B_coils(vector <s_coil>
coils, vector <double> point, double I, vector
<double> &B)
{
        B = { 0,0,0 };
        vector <double> B1;
        for (auto const& icoil : coils)
        {
                halla_B_coil(icoil.xyz,
icoil.dxyz, icoil.R, point, B1);
                sumproductvector(B, B1,
icoil.num*I*icoil.current);
        }
}
//411. Caution: coil_dir module=1
//calculates the magnetic field B due a coil in
3D at point "point"
//Uses halla_hr() tocalculate h,r,vh,vr that
will be used by halla_BxBz(). (checked)
//Uses halla_BxBz() to calculate field radially
and axially and pass it to 3D
//B field must be multiplied by turns number
and current (with its sign)
void s_ccoil::halla_B_coil(vector <double>
coil_ori, vector <double> coil_dir, double
coil_R, vector <double> point, vector <double>
&B)
{
        double h, r, Br, Bh; vector <double>
vr, vh;
        halla_hr(coil_ori, coil_dir, point, h,
r, vr, vh);
        halla_BxBz(coil_R, r, h, Br, Bh); if (r
< 1e-7) Br = 0;
        B = { Br*vr[0] + Bh * vh[0],Br*vr[1] +
Bh * vh[1], Br*vr[2] + Bh * vh[2] };
}
//412. Calculates B field due a coil in two
directions: axially and radial
//R=loop radius, r=distance point-loop axis,
z=perpendicular distance of point to loop
surface
//Must be multiplied by current and relative
permeability and squared of number of wires
void s_ccoil::halla_BxBz(double R, double r,
double z, double &Br, double &Bz)
{
        double kr = 1.0;        if (r < 0.0) {
kr = -1.0; r = -r; }
        double al = r / R, be = z / R, ga = z /
r;
        double q = (1. + al)*(1. + al) + be *
be;
        double k = sqrt(4.*al / q);
        double Bz0 = 2.e-7 / (R*sqrt(q)+1e-
300);
#ifdef __BOOST
        using namespace boost::math;
        double Kk = ellint_1(k);
        double Ek1 = ellint_2(k) / (q -
4.*al+1e-300);
#else
        double Kk, Ek1;
        Complete_Elliptic_Integrals(k, &Kk,
&Ek1);
        Ek1=Ek1/ (q - 4.*al+1e-300);
#endif

        Bz = Bz0 * (Ek1*(1. - al * al - be *
be) + Kk);
        if (r > 1e-20)
                Br = kr * Bz0*ga*(Ek1*(1. + al
* al + be * be) - Kk);
        else
```

```
                Br = 0.0;
}
//413. Auxiliary function of hallaBxBz()
//Calculates height h and distance r from axis
of a point over a coil
//There is a vector beginning in a. Then it is
calculated point d in such line that is nearer
to it and pass through c
//Module of vector b must be = 1
void halla_hr(vector <double> a, vector
<double> b, vector <double> c, double &h,
double &r, vector <double> &vr, vector <double>
&vh)
{
        restavector(c, a);//calculates
difference between point a and coil center, so
now it is referenced from coordinate (0,0,0)
        h = (b[0] * c[0] + b[1] * c[1] + b[2] *
c[2]); vh = b; //vector vh is parallel to dir
and its module is 1 because b module is 1
        vr = { c[0] - h * b[0], c[1] - h * b[1]
, c[2] - h * b[2] };
        r = modulo(vr);
        if (r < 1e-7) vr = { 1,0,0 }; else
dividemodulo(vr);

}
void Complete_Elliptic_Integrals(double x,
double* Fk, double* Ek)
{
        const double PI_2 =
1.5707963267948966192313216916397514; // pi/2
        const double PI_4 =
0.7853981633974483096156608458198757; // pi/4
        double k;       // modulus
        double m;       // the parameter of the
elliptic function m = modulus^2
        double a;       // arithmetic mean
        double g;       // geometric mean
        double a_old;   // previous arithmetic
mean
        double g_old;   // previous geometric
mean
        double two_n;   // power of 2
        double sum;

        if (x == 0.0) { *Fk = M_PI_2;      *Ek
= M_PI_2;        return; }
        k = fabs(x);
        m = k * k;
        if (m == 1.0) { *Fk = DBL_MAX;      *Ek
= 1.0;       return; }

        a = 1.0;
        g = sqrt(1.0 - m);
        two_n = 1.0;
        sum = 2.0 - m;
        for (int i = 0; i < 27; i++)
        {
                g_old = g;
                a_old = a;
                a = 0.5 * (g_old + a_old);
                g = g_old * a_old;
                two_n += two_n;
                sum -= two_n * (a * a - g);
                if (fabs(a_old - g_old) <=
(a_old * DBL_EPSILON)) break;
                g = sqrt(g);
        }
        *Fk = (double)(PI_2 / a);
        *Ek = (double)((PI_4 / a) * sum);
        return;

}
```

```
        void restavector(vector <double> &a,
vector <double> b) { a[0] -= b[0]; a[1] -=
b[1]; a[2] -= b[2]; }
                double restamodulo(vector
<double> a, vector <double> b) { return
(modulo({ a[0] - b[0],a[1] - b[1], a[2] - b[2]
})); }
        //suma al vector a el vector b:

        void  sumproductvector(vector  <double>
&a, vector <double> b, double k) { a[0] += k *
b[0]; a[1] += k * b[1]; a[2] += k * b[2]; }


        double modulo(vector <double> xyz) {
return sqrt(xyz[0] * xyz[0] + xyz[1] * xyz[1] +
xyz[2] * xyz[2]+1e-30); }
        void dividemodulo(vector <double> &xyz)
{ double m = 1.0 / modulo(xyz); xyz[0] *= m;
xyz[1] *= m; xyz[2] *= m; }
```