



# Self Partial Inductance Calculations

## - Retaining Accuracy and Avoiding Numerical Instabilities

September, 2003  
Jonas.Ekman@sm.luth.se



## *Research Report*

---

### **Introduction**

This report details the use of the well known self partial inductance calculation routines presented in [1, 2]. The right calculation routines are extremely important since the PEEC method require:

- very high accuracy in the partial element values.
- very fast calculation routines.

to enable a correct analysis and simulation results.

To be more specific, the formulas considered in this report are:

- LpSelfRect. Part. self. ind. for general rectangular conductors from [1] (Eq. (15)).
- LpSelfZero. Part. self. ind. for infinitely thin conductors from [1] (Eq. (16)).
- LpSelfLong. Part. self. ind. for very long conductors from [2] (Eq. (8)).
- LpSelf. The decision algorithm from [2] utilizing a combination of the above calculation routines.

With the corresponding C++ implementations shown in Appendix A.

The report shows partial self inductance calculations for two different basic geometries. The first conductor has a quadratic cross section of 0.01 x 0.01 cm. The second conductor has a rectangular cross section of 1 x 0.001 cm. The length of the conductors are increased from 0.1 cm to 250 cm while monitoring the part. self inductance. The wrong usage of computation routines for self part. inductance can give extremely wrong results as shown in Fig. 1 where the LpSelfRect-routine from [1] are numerical unstable for inductive cell aspect ratios >1:500.



*Research Report*

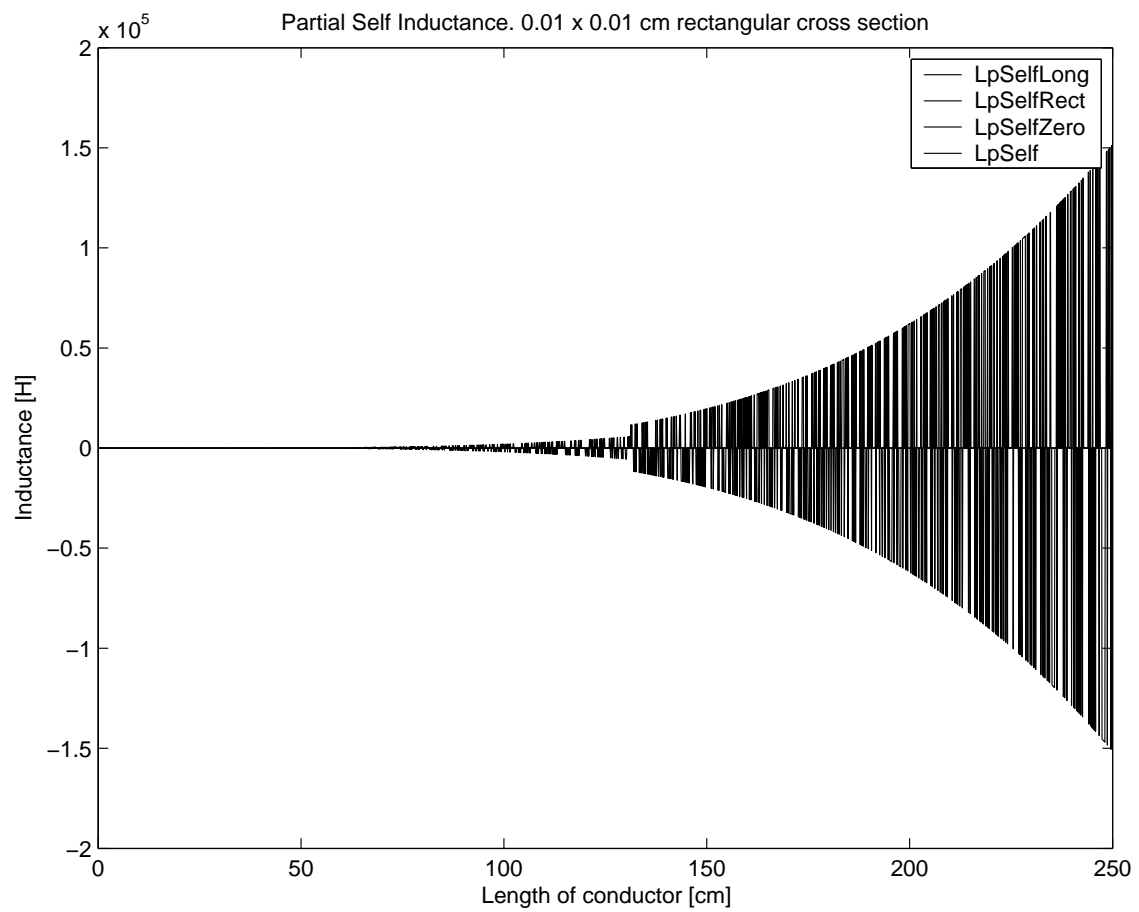


Figure 1: Numerical unstable part. self inductance calculation routine.



## Quadratic Cross Section

The 0.1 x 0.1 cm quadratic cross section can be encountered in (VFI)PEEC simulations when Skin-effects are considered important. Fig. 2 shows the difference in inductance for LpSelfZero and LpSelfLong calculation routines and the transition (LpSelf) from the first to the second as the length of the inductive cell increases.

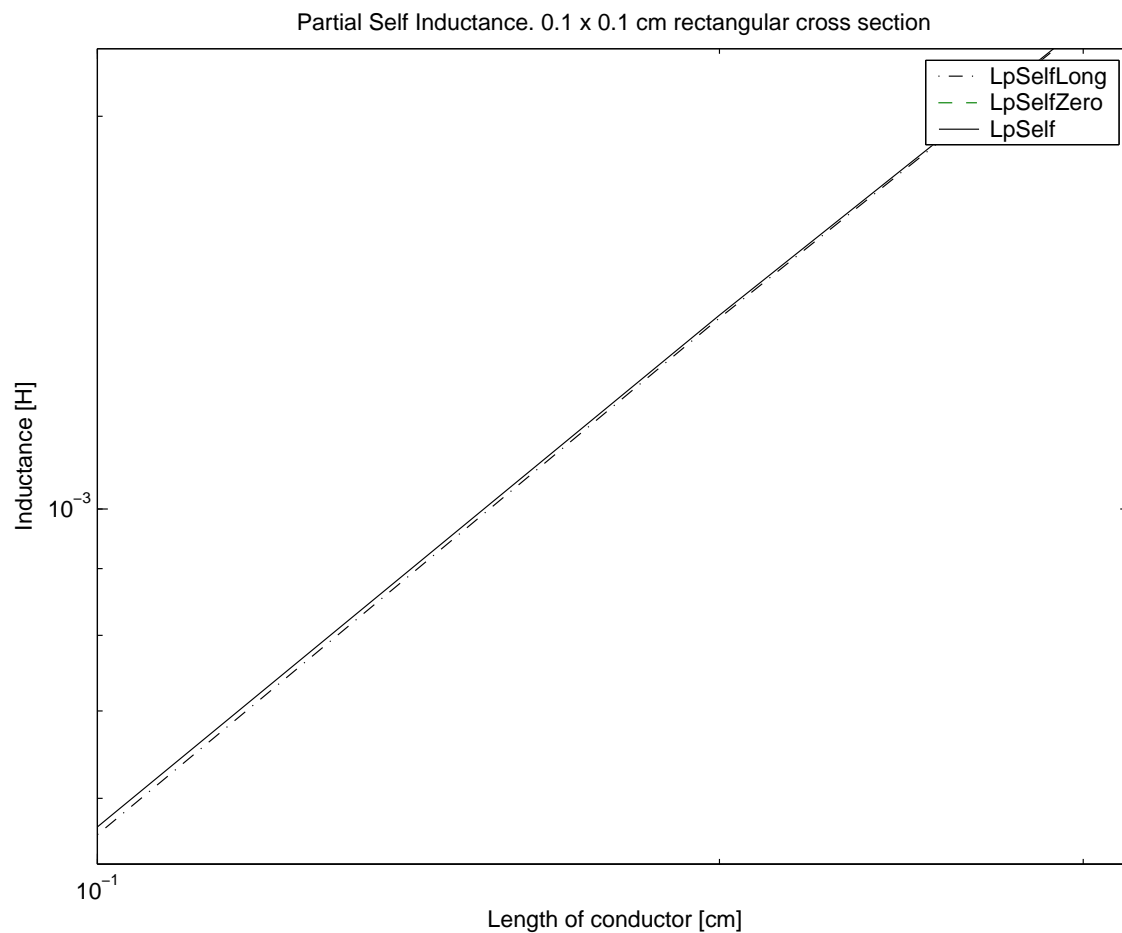


Figure 2: Difference in LpSelfZero and LpSelfLong calculation routines.



## Research Report

The quadratic cross section was increased to a 1 x 1 cm cross section to show the difference in the calculation routines for 'thick' cells. Note the negative inductance given by the LpSelfLong-routine.

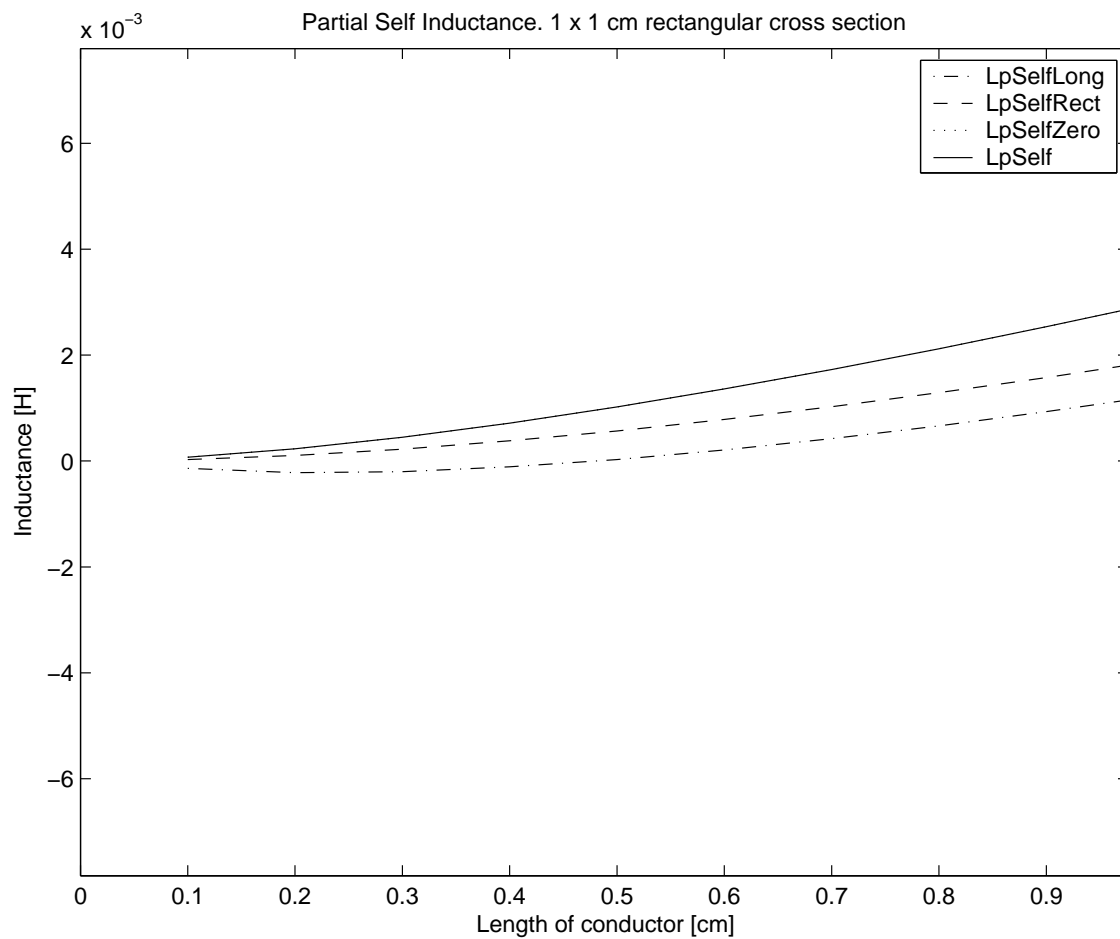


Figure 3: Negative inductance given by the LpSelfLong-routine.



## Research Report

### Rectangular Cross Section

The calculation for the rectangular cross section shows a more stable behaviour, Fig. 4.

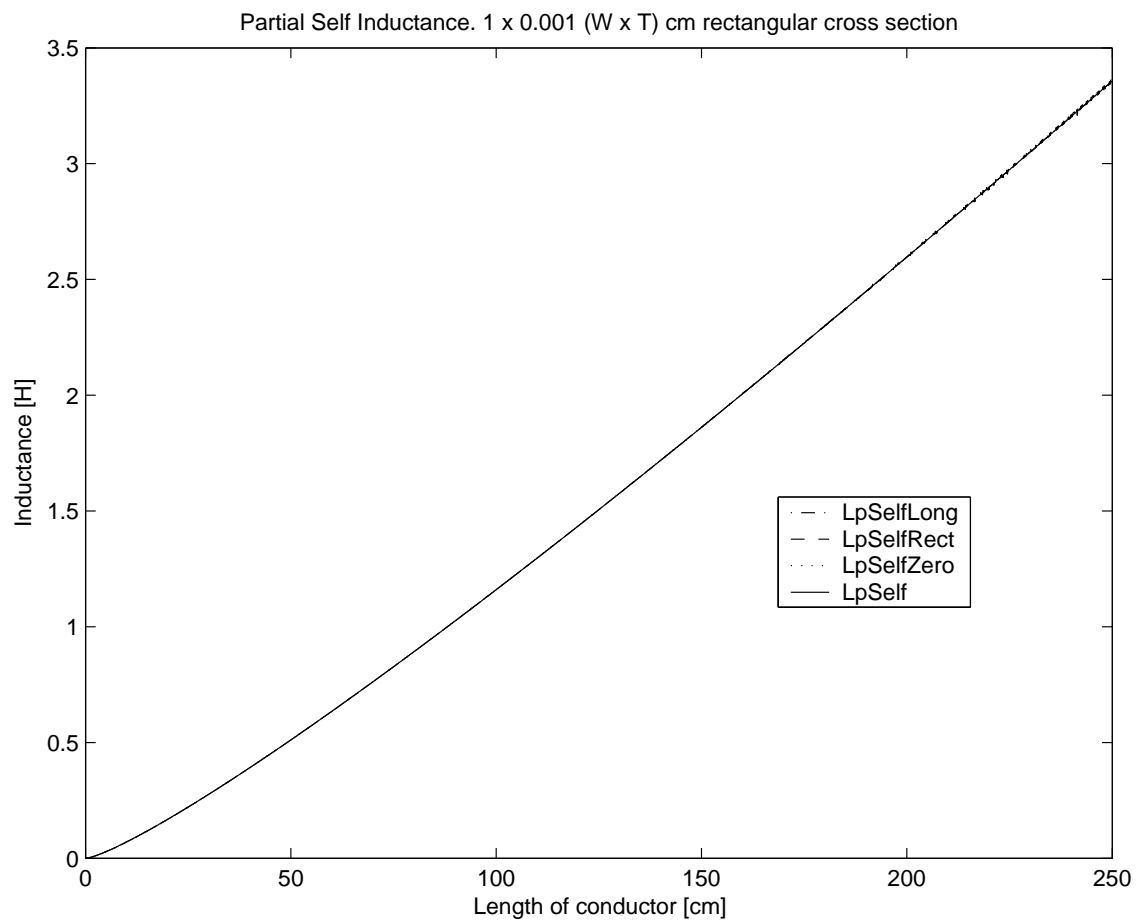


Figure 4: Part. self inductance for rectangular cross section conductor.



## Research Report

The LpSelfRect-routine shows even for this case an unstable behavior.

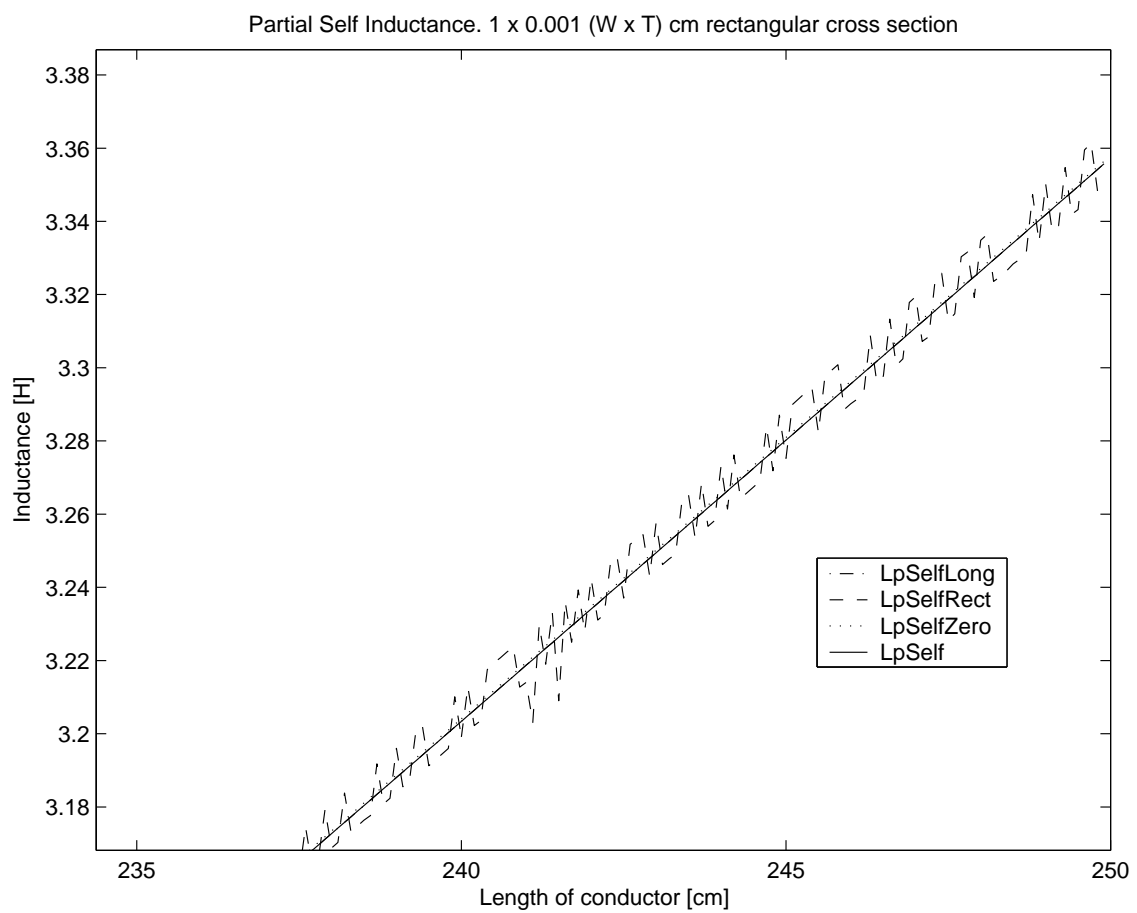


Figure 5: Instabilities in the LpSelfRect-routine.



## Research Report

### Contour Formulation

The contour formulation presented in [3] was also compared to the analytical formulas since the formulation is important for the evaluation of the nonorthogonal PEEC [4] elements.

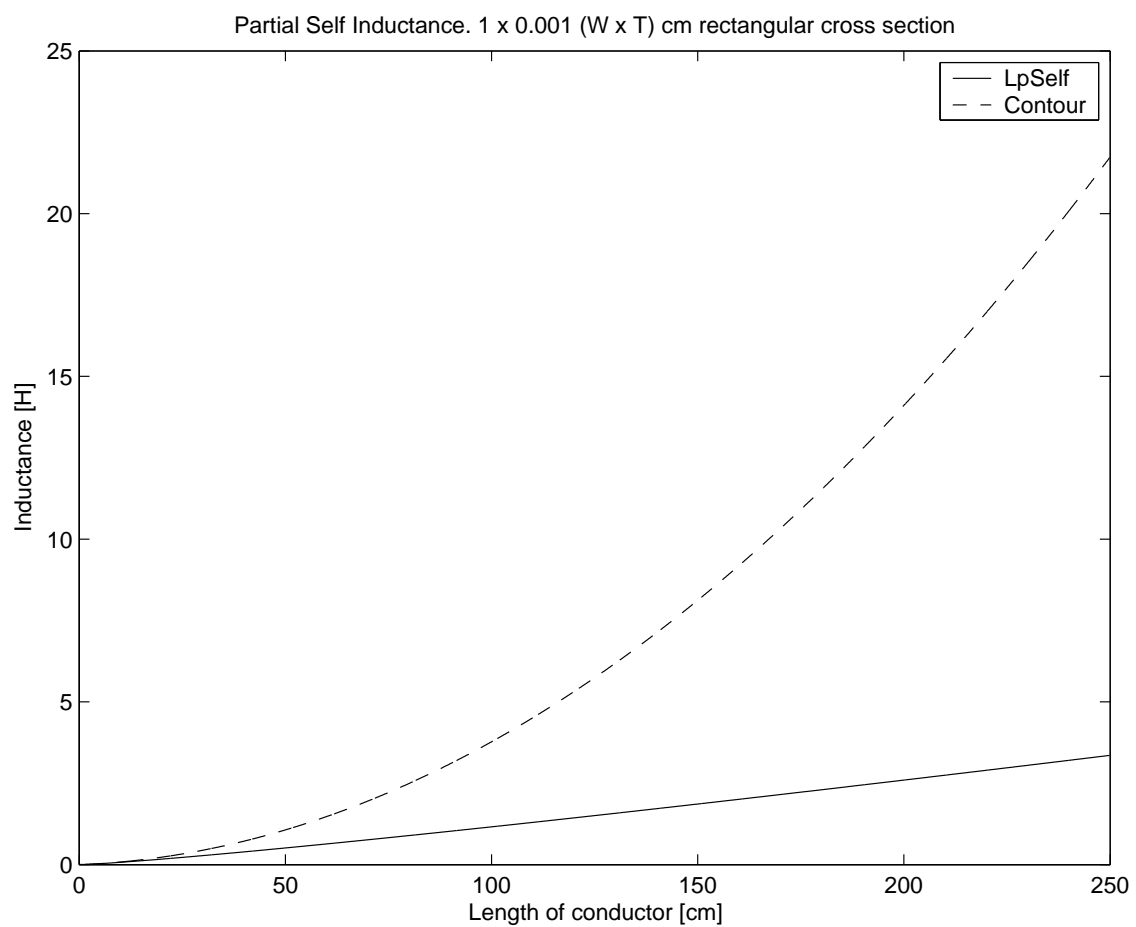


Figure 6: Contour formulation utilizing Gauss Legendre numerical integration compared with the decision algorithm.

The figure shows a large decrease in accuracy for moderately large cell aspect ratios,  $\approx 1:20$ . However, the formulation is important for the nonorthogonal PEEC method.





## *Research Report*

---

### **Conclusions**

- The decision rule presented in [2] selects the most appropriate calculation routine for the self partial inductance calculation....use it!
- The LpSelfRect-routine shows an unstable behavior for large cell aspect ratios. Be careful!
- The contour-formulation shows an reduced accuracy for moderate cell aspect ratios, use for low-aspect ratio cells!



## *Research Report*

# Appendix A

This appendix details the C++ implementation of the analytical formulas investigated in this report.

```
double LpSelf(double l, double w, double t) {
    double M=t/w;
    double U=l/w;
    double lp=0;

    if (M>1)
        M=1.0/M;

    if ((U>0.1) & (M<0.0003))
        lp=LpSelfZero(l,w);

    else if ((U>=80) & (M>0.0003))
        lp=LpSelfLong(l,w,t);

    else if ((U<0.1) & (M<0.0003) & (M>2e-5))
        lp=LpSelfRect(l,w,t);

    else if ((U<80) & (M>0.0003))
        lp=LpSelfRect(l,w,t);

    else {
        cout<<"\n\nUndefined case in LpSelf\n";
        cout<<"\n\nConductor: \n  L="<<l<<"\n  W="<<w<<"\n  T="<<t<<"\n";
        cout<<"\n\nNormalizations: \n  M="<<M<<"\n  U="<<U<<"\n";
        getch();
    }
}
```



## Research Report

---

```
double LpSelfRect(double l1, double w1, double t1) {

    const double MU_OVER_4PI = 0.001;

    double u,w,uSq,wSq,invU,invW;
    double inv20,inv24,inv60,invA4;
    double a1,a2,a3,a4,a5,a6,a7;
    double lp;

    u=l1/w1;    w=t1/w1;
    uSq=u*u;    wSq=w*w;
    invU=1.0/u;
    invW=1.0/w;
    inv20=1.0/20;
    inv24=1.0/24;
    inv60=1.0/60;

    a1=sqrt(1.0+uSq);
    a2=sqrt(1.0+wSq);
    a3=sqrt(uSq+wSq);
    a4=sqrt(1.0+wSq+uSq);    invA4=1.0/a4;
    a5=log((1.0+a4)/a3);
    a6=log((w+a4)/a1);
    a7=log((u+a4)/a2);

    lp=(wSq*inv24*invU)*(log((1.0+a2)*invW)-a5);
    lp+=(inv24*invU*invW)*(log(w+a2)-a6)+(wSq*inv60*invU)*(a4-a3);
    lp+=wSq*inv24*(log((u+a3)*invW)-a7)+wSq*inv60*invU*(w-a2);
    lp+=inv20*invU*(a2-a4)+u*a5*0.25-uSq*10.0*inv60*invW*atan(w*invU*invA4);
    lp+=u*invW*0.25*a6-w*10.0*inv60*atan(u*invW*invA4)+0.25*a7;
    lp-=invW*10.0*inv60*atan(u*w*invA4);
    lp+=invW*invW*(inv24*(log(u+a1)-a7)+u*inv20*(a1-a4));
    lp+=inv60*invU*invW*invW*((1.0-a2)+(a4-a1));
    lp+=u*inv20*(a3-a4)+u*uSq*inv24*invW*invW*(log((1.0+a1)*invU)-a5);
    lp+=u*uSq*invW*(inv24*(log((w+a3)*invU)-a6) + inv60*invW*(a4-a1+u-a3));

    return(lp*MU_OVER_4PI*8.0*l1);
}
```



## *Research Report*

---

```
double LpSelfZero(double l1, double w1) {
    const double      MU_OVER_4PI= 0.001;

    double u,uSq;
    double one0vU;
    double lpRes;

    u = l1/w1;
    uSq = u * u;
    one0vU=1.0/u;

    lpRes=MU_OVER_4PI * 0.66666666666667 *
        // Line 1
        (3.0*log(u+techsoft::sqrt(uSq + 1.0)) + uSq + one0vU +
        // 1st part Line 2
        3.0 * u * log(one0vU + techsoft::sqrt(one0vU * one0vU + 1.0)) -
        // Last term
        pow(pow(u,1.333333333333)+ pow(one0vU,0.66666666666667),1.5));
    return(lpRes*l1);
}
```



## *Research Report*

---

```
double LpSelfLong(double l, double w, double t) {
    const double MU_OVER_4PI = 0.001;

    double lp=0;
    double M=t/w;
    double U=l/w;

    lp = log((2.0 * U)/(1 + M)) + 0.5 + ((2.0)/(9*U*(1+M)));
    return(lp*MU_OVER_4PI*2.0*l);
}
```



## *Research Report*

# Bibliography

- [1] A. E. Ruehli. Inductance Calculations in a Complex Integrated Circuit Environment. *IBM Journal of Research and Development*, 16(5):470–481, September 1972.
- [2] A. E. Ruehli and P. K. Wolff. Inductance Computations for Complex Three-Dimensional Geometries, In: *Proc. of the IEEE Int. Symposium on Circuits and Systems*, vol. 1, pages 16–19, New York, NY, 1981.
- [3] G. Antonini, A. Orlandi, and A. Ruehli. Analytical Integration of Quasi-Static Potential Integrals on Nonorthogonal Coplanar Quadrilaterals for the PEEC Method. *IEEE Transactions on EMC*, 44(2):399–403, May 2002.
- [4] A. E. Ruehli *et al.*. Nonorthogonal PEEC Formulation for Time- and Frequency-Domain EM and Circuit Modeling. *IEEE Transactions on EMC*, 45(2):167–176, May 2003.