

# Χρήση Αντικειμενοστραφούς Προγραμματισμού στον Υπολογιστικό Ηλεκτρομαγνητισμό

## *Εφαρμογή*

- Υλοποίηση βιβλιοθήκης για προσομοίωση Η/Μ διατάξεων με MAS

## *Εργαλεία*

- Γλώσσα Προγραμματισμού C++

## *Τεχνικές OOP*

- Κληρονομικότητα (Inheritance)
- Πολυμορφισμός (Polymorphism)
- Απόκρυψη Πληροφορίας (Information hiding)
- Ενθυλάκωση (Encapsulation)

# Αντικείμενα και κλάσεις

## Κλάσεις γεωμετρίας

- `point2d`, `point3d`, `coord_cart`
- `geom`, `geom_linear`, `geom_planar`,  
`geom_cyl_2d`, `geom_cyl_3d`

## Κλάσεις διανυσμάτων

`vector_cart3d`, `vector_sph`, `vector_cyl`

## Κλάσεις Η/Μ πηγών (AS)

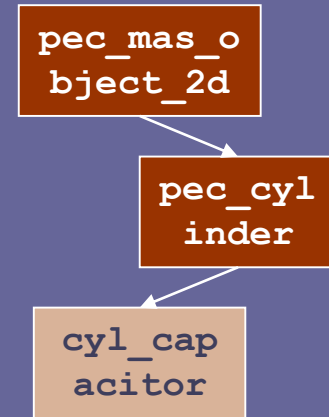
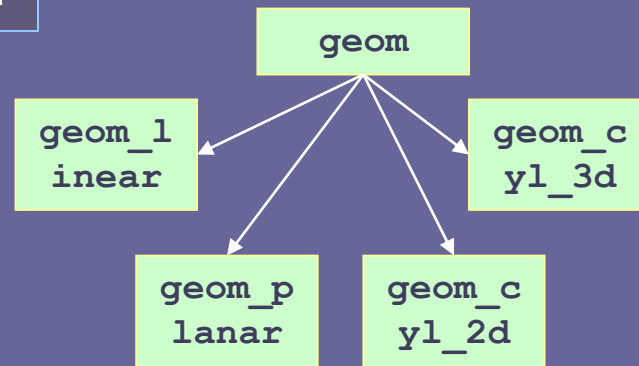
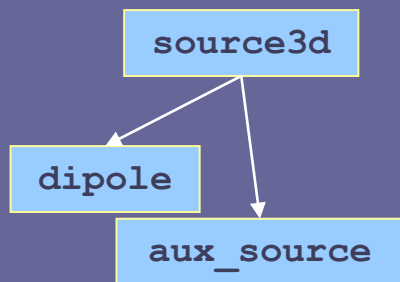
`wire`, `source3d`, `dipole`, `aux_source`

## Κλάσεις βασικών Η/Μ διατάξεων

`mas_object_2d`, `pec_mas_object_2d`,  
`mas_object_3d`, `pec_mas_object_3d`,  
`pec_cylinder`, `infinite_strip`, ...

# Κληρονομικότητα και πολυμορφισμός

## Κληρονομικότητα



## Πολυμορφισμός

Κατασκευαστές κλάσεων (class constructors)

```
[MAS_OBJECT_2D::GetEMField_in]
```

```
(positive k, POINT2D p2d)
```

```
(positive k, positive rho, double phi)
```

# Απόκρυψη πληροφορίας – Ενθυλάκωση

- `private` ιδιότητες (`protected`, `public`)
- αφαιρετικότητα (π. χ. `x`, `y`, `z` συντεταγμένες του `point3d`)
- διεπαφή μέσω συγκεκριμένων μεθόδων (π.χ. `wire::GetIO()`)
- τμήματα κώδικα σε *μαύρο κουτί* (βιβλιοθήκες)

# 2D πρόβλημα σκέδασης από αγωγίμο κύλινδρο απείρου μήκους

## 1. Γεωμετρία του προβλήματος

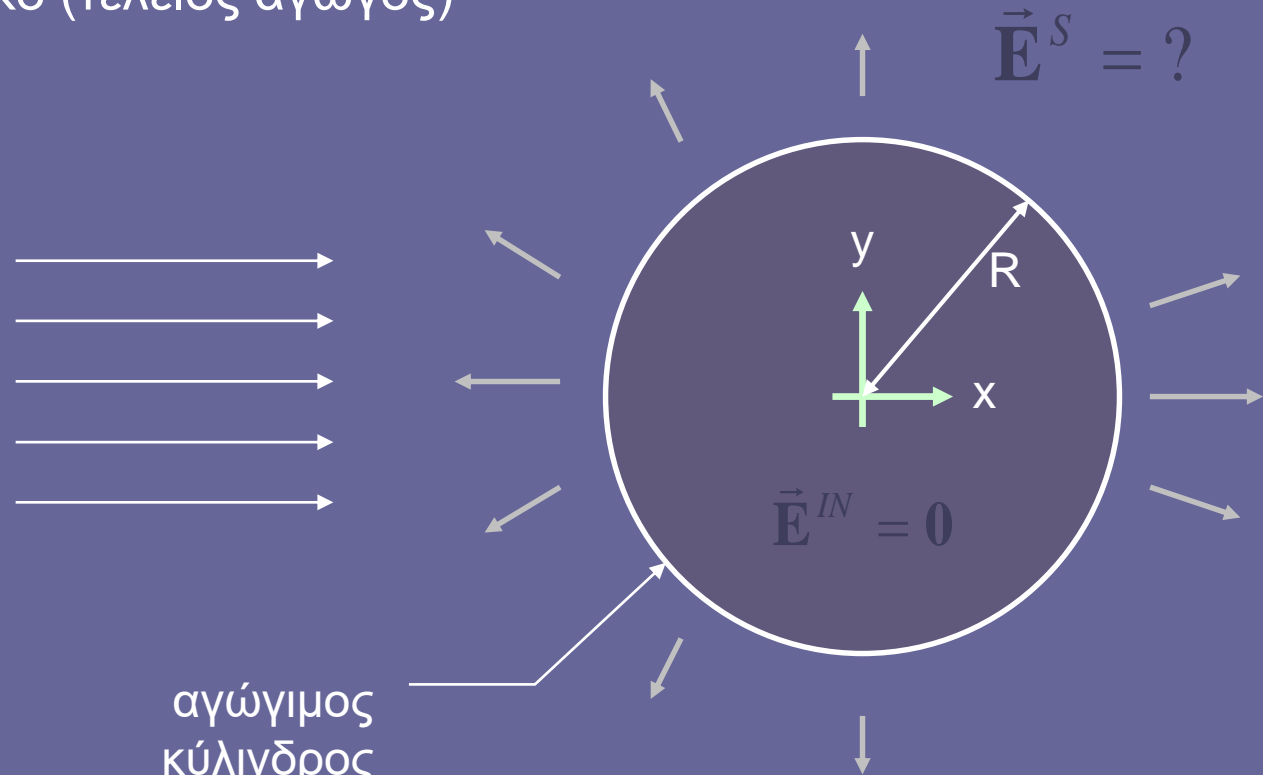
προσπίπτον πεδίο: επίπεδο κύμα (TM πόλωση)

εσωτερικό πεδίο: μηδενικό (τέλειος αγωγός)

$$\vec{\mathbf{E}}^{INC} = \hat{\mathbf{z}}E_0e^{-jk_0x}$$

αέρας  $\{\epsilon_0, \mu_0\}$

αγωγίμος  
κύλινδρος




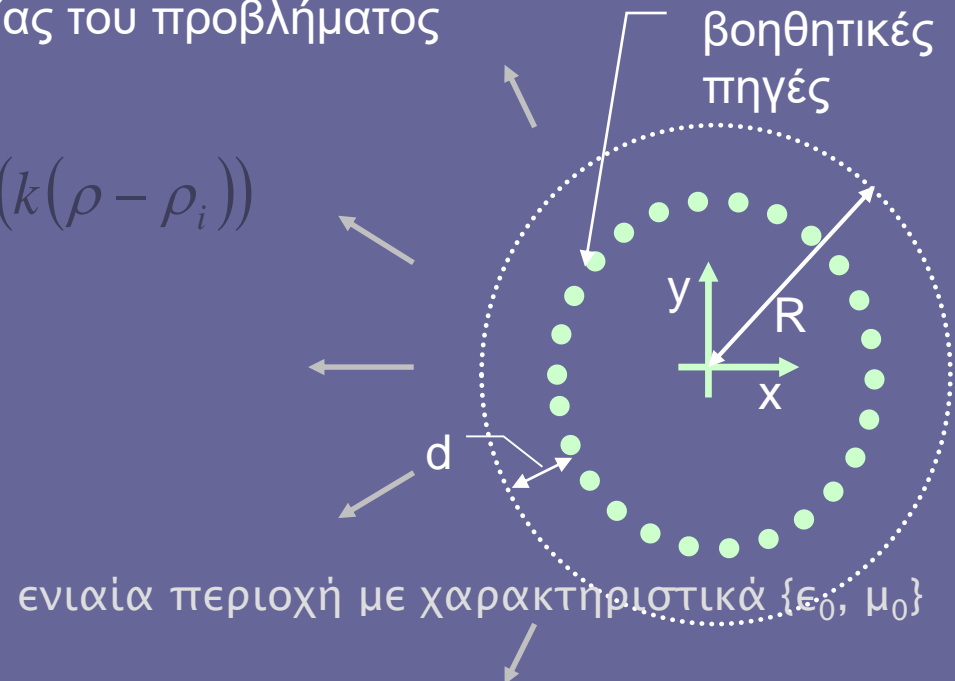
# 2D πρόβλημα σκέδασης από αγώγιμο κύλινδρο απείρου μήκους

## 2. Ισοδύναμο πρόβλημα MAS

- Επιλογή βοηθητικών πηγών:  $N$  ευθύγραμμες ρευματικές κατανομές απείρου μήκους κατά  $z$ , με άγνωστο πλάτος διέγερσης
- Τοποθέτηση βοηθητικών πηγών: σύμμορφη επιφάνεια
- Εκμετάλλευση της γεωμετρικής συμμετρίας του προβλήματος

$$\vec{\mathbf{E}}^S = \hat{\mathbf{z}}C \sum_{i=1}^N I_i H_0^{(2)}(k(\rho - \rho_i))$$


$$\vec{\mathbf{E}}^{INC} = \hat{\mathbf{z}}E_0 e^{-jk_0 x}$$



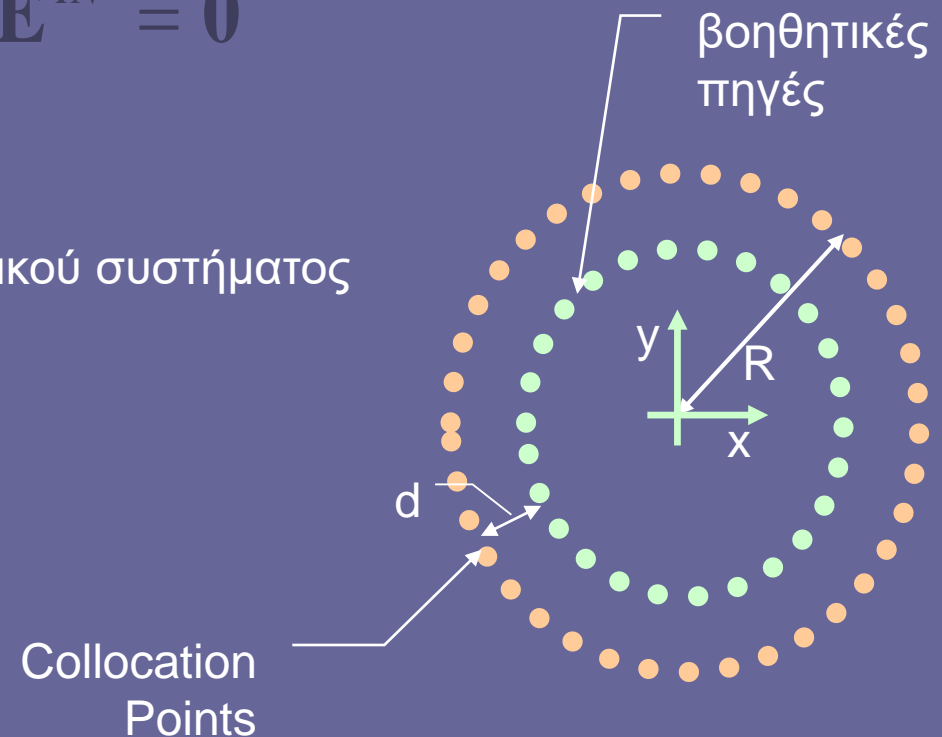
# 2D πρόβλημα σκέδασης από αγωγίμο κύλινδρο απείρου μήκους

## 3. Επίλυση ισοδύναμου προβλήματος

- Collocation Points (CPs) και εφαρμογή των οριακών συνθηκών του προβλήματος

- Σε κάθε CP ισχύει:  $\vec{\mathbf{E}}^{INC} + \vec{\mathbf{E}}^S = \vec{\mathbf{E}}^{IN} = \mathbf{0}$

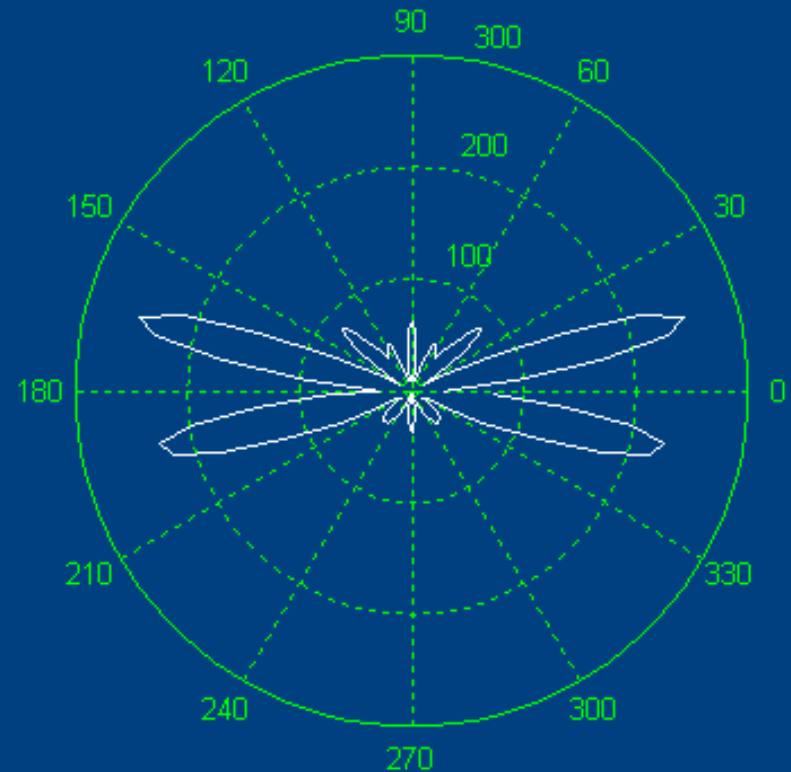
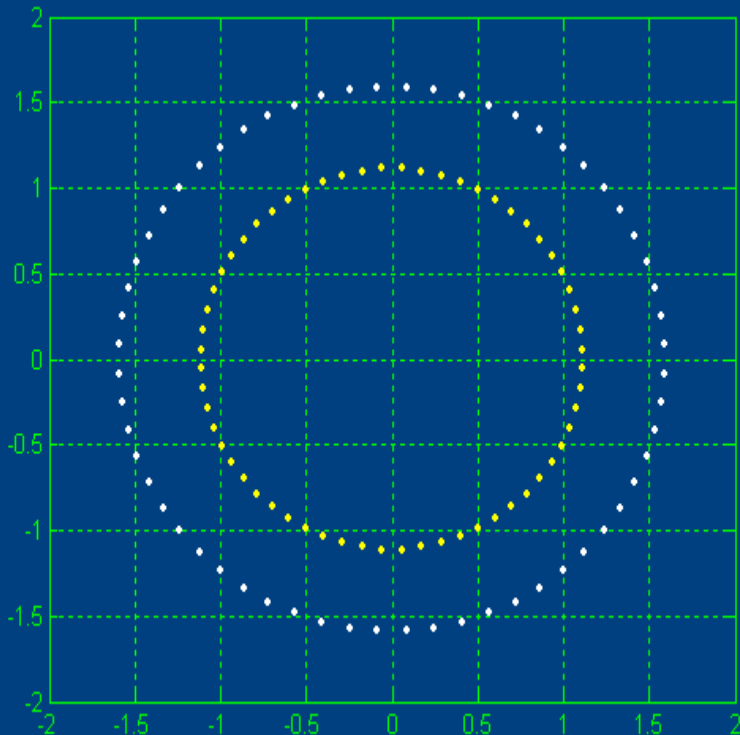
- Άγνωστα μεγέθη: τα ρευματικά πλάτη  $I_i$
- Κατάστρωση του  $N \times N$  πίνακα του γραμμικού συστήματος
- Επίλυση του γραμμικού συστήματος



# 2D πρόβλημα σκέδασης από αγώγιμο κύλινδρο απείρου μήκους

## 4. Παρουσίαση της λύσης του προβλήματος (RCS)

$$R = 1.6 \lambda, N = 60, d = 0.3 R$$

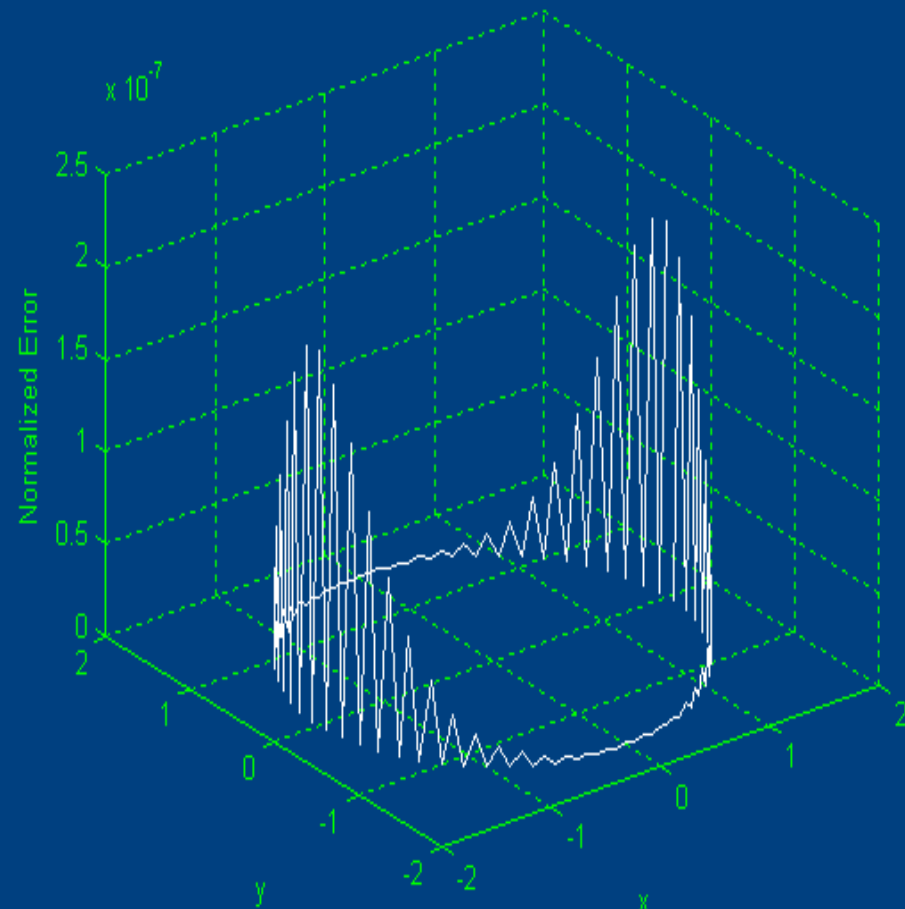
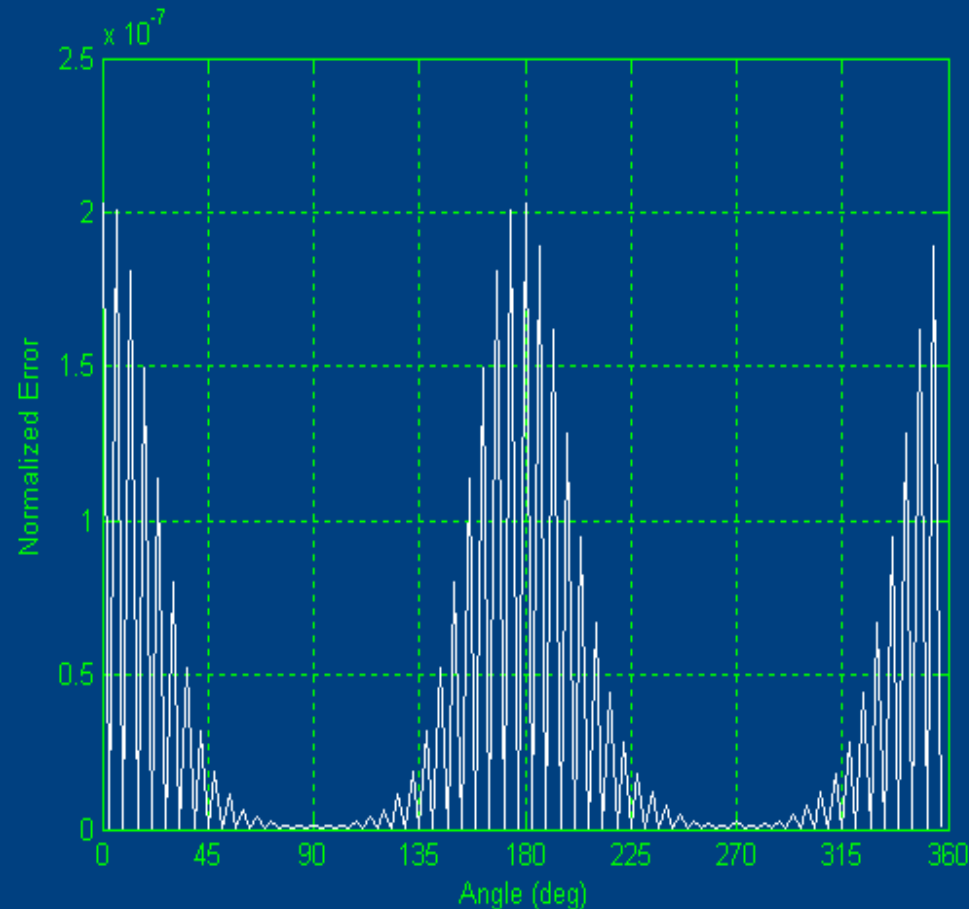


bistatic RCS plot



# 2D πρόβλημα σκέδασης από αγώγιμο κύλινδρο απείρου μήκους

## 5. Σφάλματα και Αιτιολόγηση



# Κατάστρωση γεωμετρίας (1)

## Κλάσεις:

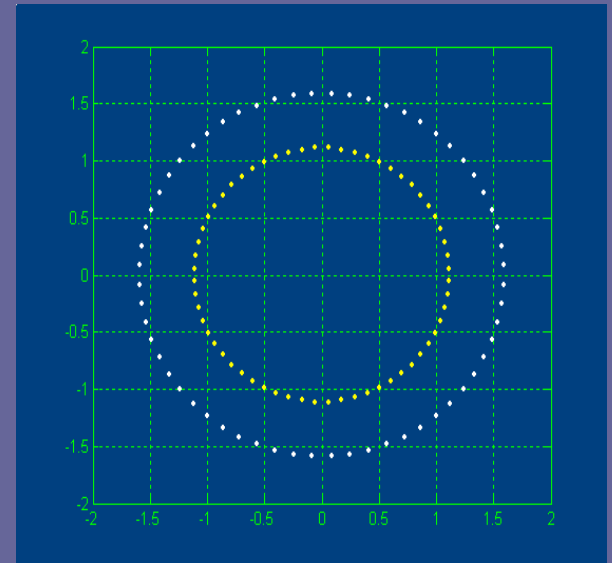
- `point2d`, `pec_object_2d`

## Διακριτοποίηση:

- `grid = (stopPhi - startPhi) / NOS`

## Παραμετρικές εξισώσεις:

- `phi = startPhi + grid * (-0.5 + i * phiDir)`
- `x = x0 + r * cos(phi)` // Κανονικοποιημένα σε  $\lambda$
- `y = y0 + r * sin(phi)`



# Κατάστροση γεωμετρίας (2)

Ορισμός συντεταγμένων:

- `cp[i].init(x[i], y[i]) // cp, as → point2D`
- `as[i].init(x[i], y[i])`

Ορισμός AS και CP:

- `wires[i].set(as)`: Μέθοδος της `wire` (`aux. source`)
- `setCP(cp)`, `setAS(wires)`: Μέθοδοι της `pec_object_2d`

$$\vec{\mathbf{E}}^{wire}(\rho) = \hat{\mathbf{z}} \cdot C \cdot I_i H_0^{(2)}\left(k(\rho - \rho_{wire})\right)$$

# Κατάστρωση επίλυσης (1)

Βασικές μεταβλητές:

- `double_complex** KernelArray`
- `double_complex* ConstantVector, currents;`

Δημιουργία γραμμικού συστήματος:

```
PEC_object_2D::BuildLinearSystem(&KernelArray,  
    &ConstantVector, Amplitude, k, direction)
```

$$\vec{\mathbf{E}}^S = \hat{\mathbf{z}}C \sum_{i=1}^N I_i H_0^{(2)}(k(\rho - \rho_i))$$

# Κατάστρωση επίλυσης (2)

Επίλυση συστήματος:

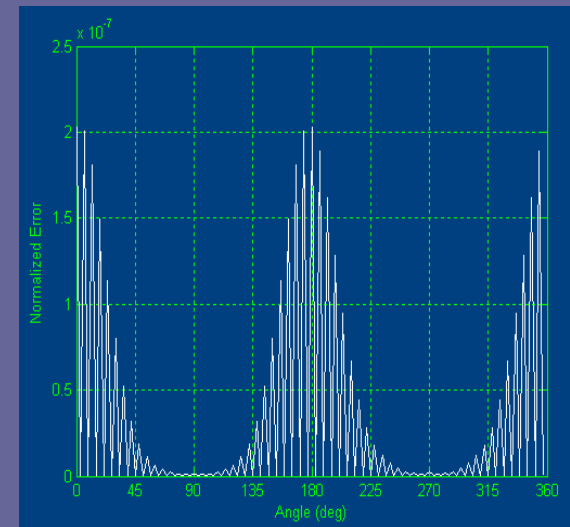
- `LinearSystemSolver_LU_Decomposition(KernelArray, ConstantVector, currents, size)`

Χρησιμοποίηση λύσεων για παράγωγα μεγέθη ( $\Delta E_{BC}$ , RCS)

- `PEC_object_2D::LoadCurrents(currents)`

$$\vec{\mathbf{E}}^{INC} = \hat{\mathbf{z}}E_0e^{-jk_0x}$$

$$\vec{\mathbf{E}}^{INC} + \vec{\mathbf{E}}^S = \vec{\mathbf{E}}^{IN} = \mathbf{0}$$



# Βοηθητικά Εργαλεία

## Προγραμματισμό με C/C++

- GNU Scientific Library (Bessel/Hankel)
- LU Decomposition
- nrutil.c και nrutil.h (για την LU decomposition)

## Προγραμματισμό με Fortran

- LU Decomposition
- Numerical Recipes in Fortran ([link](#))

## Προγραμματισμό με Java

- Μιγαδικοί αριθμοί και συναρτήσεις στη Java
- Παράδειγμα Κλήσης Συναρτήσεων Βιβλιοθήκης σε C με Java
- Υλοποίηση Συναρτήσεων Bessel
- Βοηθητικοί κώδικες σε Java (Hankel, LU, complex, κτλ.)