Peridynamic 3D models of nanofiber networks and carbon nanotube-reinforced composites

Florin Bobaru* and Stewart A. Silling†

*Department of Engineering Mechanics, University of Nebraska-Lincoln, Lincoln, NE 68588-0526
†Computational Physics Department, Sandia National Laboratories, Albuquerque, NM 87185-0820

Abstract. Here we employ a reformulation of the continuum mechanics theory, the *peridynamic formulation (PF)* in an integral form that, at the discretized level, resembles molecular dynamics (MD). The peridynamic theory is based on a continuum formulation and can capture nucleation and propagation of defects and discontinuities without ad-hoc assumptions or special treatments needed by classical continuum theory. We analyze nanofiber networks and CNT-reinforced polymer composites. We treat all crossovers contacts between fibers as perfect bonds. The use of repulsive short-range forces eliminates the need for complex contact detection algorithms. We generate the fibers as 3D curves with random orientation, with or without preferred directionality. We use an object-oriented code written in Fortran 90/95 to define the geometrical entities. The PF can capture the deformation and complex fracture behavior in fully 3D dynamic simulations. van der Waals forces are included in these calculations. The strength of the bonds between the polymer chains and the CNTs, as well as among the chains, is controllable.

INTRODUCTION

The discovery of single- and multi-walled carbon nanotubes (CNTs) has generated huge interest in modeling and mechanical simulations at the nanoscale. One question remains: how to account for the nano-effects while being able to solve problems posed for systems of larger size, where continuum theories are effective? Important progress has been made in understanding the deformation and fracture of a single- or multi-walled CNTs. The methods currently in use include, among others, molecular dynamics (MD) simulations with or without a localized atomistic calculation (see, e.g. [1]). The size of the nanomaterials and nanostructures that can be analyzed using these methods is rather limited, and atomistic or MD models are used only at specific, *predetermined* locations.

Here we employ a reformulation of the continuum mechanics theory in an integral form that, at the discretized level, resembles molecular dynamics calculations. The peridynamic theory (see [2]) is able to capture nucleation and propagation of defects and discontinuities without ad-hoc assumptions or special treatments needed in the classical continuum theory. The numerical solution method for the integral equations discretizes the region into nodes that interact with any other nodes within a finite distance called the horizon. This interaction is not limited just to nearest neighbors. The physical interaction between nodes takes place through bonds that may be thought of as springs. The bonds may be elastic or inelastic, linear or nonlinear, and may include rate dependence. Bonds may also break irreversibly, which provides a way to model damage. When large numbers of bonds break in a continuous body, they tend to organize themselves into two-dimensional patterns that grow at their edges; this is how fracture emerges spontaneously in the peridynamic model. Fracture occurs without introducing supplemental kinetic relations that in classical fracture mechanics approaches would be required to govern crack growth. The numerical method is meshfree in the sense that it does not use geometrical connections such as elements between the nodes, hence there is no need for a mesh generator to create elements.

The PF is similar to MD at the discrete level. The difference from the MD simulations is that the peridynamic model is a continuum formulation and, as a consequence, it does not have to deal with the enormous number of atoms needed by an MD simulation.

The PF replaces the spatial derivatives in the classical continuum formulation for the equations of motion with an integral formulation:

$$\rho \ddot{\mathbf{u}}(\mathbf{x},t) = \int_{V_{\mathbf{x}}} \mathbf{f}[\mathbf{u}(\mathbf{s},t) - \mathbf{u}(\mathbf{x},t), \mathbf{s}-\mathbf{x}]\, dV_{\mathbf{s}} + \mathbf{b}(\mathbf{x},t)$$

where $V_{\mathbf{x}}$ is a neighborhood of $\mathbf{x}$, $\mathbf{u}$ is the displacement field, $\mathbf{b}$ is the body force, ρ is the mass density, and $\mathbf{f}$ is a pairwise force vector per unit volume squared that the particle $\mathbf{s}$ exerts on the particle $\mathbf{x}$ (see [2], [3], [4]).

We consider two material systems: nanofiber networks and CNT-reinforced polymer composites. We treat all crossovers between fibers as perfect bonds. The option of decoupling some or all of these bonds exists. Using an object-oriented Fortran 90/95 code to define the geometrical models, we generate the nanofibers as 3D curves with random orientation, possibly with preferred directionality. We analyze the deformation and complex fracture behavior with the PF in fully 3D dynamic simulations. We note the differences in results when molecular scale forces (van der Waals) are omitted from calculations. The possibility of controlling the strengths of the bonds between the fibers and the CNTs, as well as among the fibers, exists.

THE GEOMETRICAL MODEL

Controlling randomness in nanofiber-networks

In many practical applications, such as filtration problems, tissue engineering, smart sensors etc, nanofiber networks are manufactured in membrane-type forms. To create an equivalent computational model for such a network-membrane structure, we developed an algorithm that generates individual continuous fibers inside a bounding box. As the height of the bounding box becomes smaller and smaller, we approach the membrane-like construct of fibers. The following procedure is used to generate families of directional random strings:

1. generate seed nodes inside a small section at one of the ends of the box; use a normal distribution for the coordinates of these nodes;
2. for every seed node, calculate a new direction in which to move using
$$\mathbf{d}_{new} = \frac{\mathbf{d}}{\|\mathbf{d}\|} + \sigma\mathbf{r}; \tag{1}$$
where $\mathbf{d}$ is the preferential direction, σ is a given standard deviation, and $\mathbf{r}$ is a 3D vector whose elements are randomly generated numbers using a Gaussian distribution with zero mean and standard deviation one;
3. change the length of the new direction to desired length between nodes:
$$\mathbf{d}_{new} \leftarrow \frac{\mathbf{d}_{new}}{\|\mathbf{d}\|}\delta$$
with δ being the selected distance between two consecutive discretization nodes.
4. set the next node's coordinates as:
$$\mathbf{p}_{next} = \mathbf{p} + \mathbf{d}_{new};$$
5. check if next-point is inside the box: if true then continue to generate the following point, else reevaluate Eq. (1) with a different random vector $\mathbf{r}$ and a new $\sigma \leftarrow \sigma - \beta$ until the point falls inside the box. The small parameter β is used to reduce the number of iterations needed to find the next point inside the box.

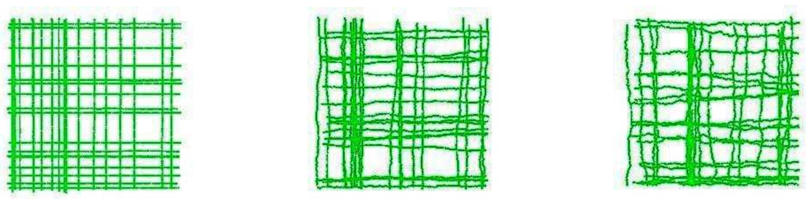
This algorithm will generate one family of directional fibers. Additional families can be generated along different directions such that a membrane-type structure is constructed. There exists the possibility of nodes overlapping each other or being to close to one another. This, physically, represents interpenetration of material. In the PF formulation, however, this possibility is not an issue since a very-short range repulsive force is built into the code to prevent non-physical interpenetration of material from taking place. Even when we start with a geometry that has nodes located at the same position, for instance, at the first iteration of the PF simulation these nodes will be separated by the action of the very-short range force. This process does not induce significant deformations since the range of action of these forces is extremely small. In our case the range of the repulsive forces is on the order of 1nm.

In figures 1a, 1b, and 1c, we plot three instances (for $\sigma = 0.01, 0.2, 0.3$ respectively) of fiber-networks generated in the fashion described above. We use two seed boxes of dimensions $5 \times 200 \times 5$ and $200 \times 5 \times 5$ to generate families with preferred directionality along the x and y direction respectively. By varying the parameter σ we can control the degree of "waviness" in the individual fibers. We note that the fiber networks so constructed are three-dimensional. We use different uniform distributions for every seed box.

CNT's embedded in a polymer matrix

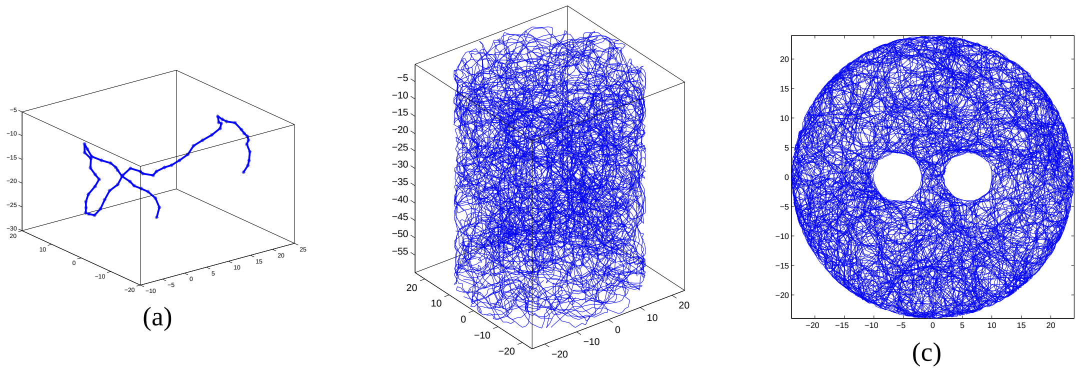
Using an algorithm similar to the one described above, we can generate fibers along arbitrary directions. To study the pull-out of a carbon nanotube embedded in a polymer matrix, we model individual molecular chains as three-dimensional strings with random orientation. We are also interested in evaluating the complex interaction between the nanotubes and the polymer chains as well as interactions of nanotubes with each other. In figure 2a we plot a typical single fiber.

We generate about 150 polymer strings (with $\sigma = 0.4$) inside a cylinder of radius 24nm with two interior cylindrical holes (see figures 2b and 2c). Assuming a diameter of 1nm for the fibers, this translates into a 60% volume fraction being occupied by the fibers. The seed nodes follow a uniform distribution in the x, y, z directions. When a string node gets close to the outer



(a) (b) (c)

FIGURE 1. Two “orthogonal” families of 3D fibers generated with various degrees of randomness determined by the parameter σ in Eq. (1). In (a) $\sigma = 0.01$, in (b) $\sigma = 0.2$, in (c) $\sigma = 0.3$.



(a) (b) (c)

FIGURE 2. Geometrical models of polymer chains. A single chain obtained with $\sigma = 0.4$ is shown in (a). Strings generated inside a cylindrical region with two tubular holes (b). Top view is shown in (c).

boundary of the cylinder, we notice that the following nodes are “crawling” along the boundary. This leads to agglomeration of fibers close to the outer boundary. To avoid that from happening, we select the seeds in a cylinder with a smaller external radius of 20 nm. Using this approach we can maintain a more uniform fiber density. Inside the two cylindrical holes we place the carbon nanotubes of diameter 6nm. There is a distance of 1nm between the CNTs and the boundary of the cylindrical holes in the matrix.

An OOP in Fortran 90/95

We implement the generation of the 3D strings using an object-oriented programming (OOP) approach in Fortran 90/95 (see, e.g. [5], [6]). The advantage for using an OOP style for the geometrical model rests in the ability of reusing code, and in the ease of managing and expanding it. The OOP implementation is also closer to the natural language representation. Most of the important characteristics of OOP (data encapsulation, polymorphism, inheritance) can be naturally expressed in F90/95. To implement run-time polymorphism (or dynamic dispatching) in F90/95 one has to write more code than is needed in C++ or Java, for example.

We describe below an OOP implementation for generating seed nodes inside a generic box. The 3D strings start at these randomly generated seed nodes. We use the code to construct the fiber network inside a parallelepiped box or cylindrical region with two tubular holes (see figure 2c). We combine the conditions for the string to form inside a box with an opposite condition for the strings not to enter a box, in order to generate fibers in the cylinder with two holes as in figure 2.

In distinct F90 modules (`classRBox`, `classCylinder`, etc.) we define new data types to describe objects such as a parallelepiped or a cylinder:

```
type cylinder
  real :: radius
  real :: height
  real(3) :: center ! center of the top face
end type cylinder
```

Each of these classes contains the procedures for defining new objects of the corresponding types: `newRBox`, `newCylinder`, etc. We can associate these different procedures with the same name `new`. This is static polymorphism. To model a hollow cylinder

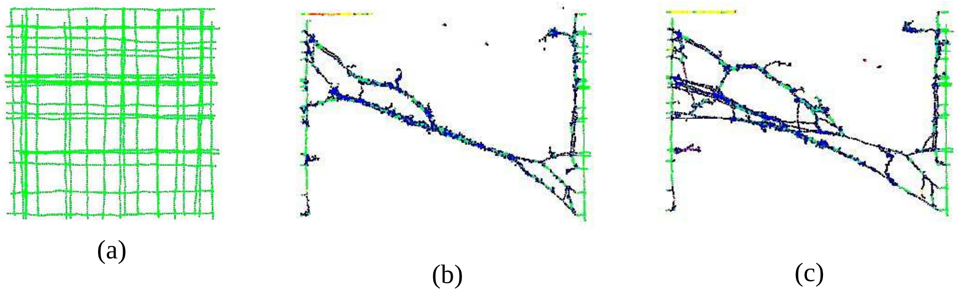


FIGURE 3. Two “orthogonal” families of fibers generated with a degree of randomness $\sigma = 0.15$ in Eq. (1). The left-end of the network moves with constant velocity $v_0 = 20$nm/ns to the left. From the initial state (a), when van der Waals forces are present, the deformed network is shown at 4.5ns in (b). At the same instant, without van der Waals forces, the fiber network is shown in (c). In both cases, the initial velocities vary linearly with the *x*-coordinate.

we can use inheritance by defining a new type (class) `hollowCyl` as follows:

```
type hollowCyl  ! concentric hollow cylinder
    real :: innerRadius
    type(cylinder) :: c
end type hollowCyl
```

The methods (subroutines) of the class cylinder are not directly inherited by the new class. This requires writing some more code compared to C++ for example, but, as noted in [5], it may be safer not to inherit everything by default.

The aim of OOP is to make the code easier to expand and maintain, as well as safer. Run-time polymorphism can help with some of these aspects. We write a piece of code that refers to a generic type of box. At run-time, the generic box is associated with the desired input (cylinder or parallelepiped) and the program will execute the corresponding routines. The framework for this procedure is exemplified below. We define a new type that points to our original classes `rbox` and `cylinder`. To generate seed nodes inside a generic box, we check the association of the components of the generic box and call the corresponding subroutine.

```
module classBox
 use classCylinder, classRBox
 implicit none

 type BoxType
  type(rBox), pointer      :: ptToRBox
  type(cylinder), pointer  :: ptToCyl
 end type BoxType

 interface new ! give a generic name
   module procedure newRBox, newCylinder
 end interface new
 interface assignBox
   module procedure assignRBox, assignCyl
 end interface assignBox
 interface seedsInBox
   module procedure seedsInRBox, seedsInCyl
   module procedure seedsInBoxType
end interface seedsInBox

contains

 ! assign a rectangular box to a generic box
 subroutine assignRBox (rBox, box)
   type(rBox), target, intent(in) :: rBox
   type(BoxType), intent(out)     :: box

   call nullifyBoxType(box)
   box%ptToRBox => rBox
 end subroutine assignRBox

 ! generate n seed points in a box-type object
 subroutine seedsInBoxType (n, box, pts)
   integer, intent(in) :: n
   type(boxType), intent(in) :: box
   real, intent(out) :: pts(n,3)

   if ( associated (box%ptToCyl) ) then
     call seedsInCyl(n, box%ptToCyl, pts)
   elseif ( associated (box%ptToRBox) ) then
     call seedsInRBox(n, box%ptToRBox, points)
   end if
  end subroutine seedsInBoxType
end module classBox
```

We use the `classBox` module in the main program, and after initializing an object of type, say, `cylinder`, we employ run-time polymorphism as shown below. New classes can be easily added (such as a hollow cylinder) using the existing classes (cylinder). The generic part of the code does not have to be changed. Of course, one has to add a couple of lines in the `seedsInBoxType` procedure above.

```
type (boxType) :: box
type(cylinder), target :: cBox

call new(radius, height, center, cBox)
call assignBox(cBox, box)
! use the generic code
call seedsInBox(n, box, points)
```

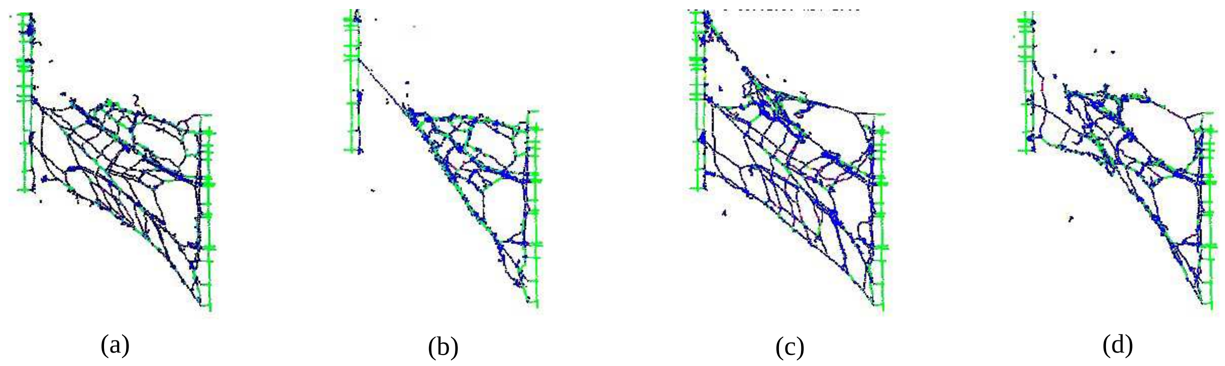


FIGURE 4. The nanofiber network under shear. Only the left side moves in the positive y-direction with 50nm/ns. The initial velocities vary linearly with the x-coordinate along the network. Elastic energy and the deformed shape when van der Waals forces are considered: after 2.35ns (a), and after 3.1ns (b). Results for the case when van der Waals forces are excluded from the computation: after 2.35ns (c), and after 3.1ns (d).

NUMERICAL RESULTS

Some of our simulations and results are described here. The material model we use corresponds to a microelastic brittle material (see [2]). Here, "brittle" means that no plastic deformation occurs prior to fracture, even though the total deformations may be substantial. We assume that polymer fibers have a critical bond stretch (see [4]) $s_0 = 0.3$. This means that peridynamic bonds break irreversibly when their deformed length equals 130% of their initial length. For the fibers, we use a micromodulus corresponding to a Young's modulus of 1.5 GPa, and for the CNTs a Young's modulus of 40.5 GPa.

Test 1: membrane stretching and influence of van der Waals forces

We generate a nanofiber network as described in the previous section. We select a parallelepiped with a length and width of 200 nm and height of 5 nm. In this parallelepiped we generate fibers as 3D curves of 1nm diameter and length 200 nm. Each 3D string is made up of straight segments of random orientations and has 100 nodes. We use $\sigma = 0.15$. We set zero displacements for the right-end region of the network defined by $x \in [85, 100]$ nm. At the symmetrical left-end we impose a constant velocity ($v_= 20$ nm/ns) displacement in the negative x direction. At time $t = 0$ the network nodes have velocities corresponding to a linear profile along the x-coordinate. From the initial stage seen in figure 1a, at 4.5 ns, the deformed and damaged network looks as in figure 1b when van der Waals forces are included. When the van der Waals forces are left out from the computation, the deformed configuration after 4.5 ns is given in figure 1c. The importance of including the molecular forces is demonstrated. The logarithm of the strain energy is used to distinguish the high strain regions (darker) from the low strain (lighter) zones.

Test 2: membrane shearing and influence of explicit modeling

Using the same network model as in test 1 (see figure 3a), we perform two shear tests: with and without van der Waals forces. We move only the left end with a constant positive velocity of 50nm/ns in the y-direction. We select an initial linear profile for the velocities: zero at the right-end and increasing proportionally with the distance from the right-end to reach 50nm/ns at the left-end. In figures 4a and 4b we show the damage and fracture patterns for the case that includes the van der Waals forces, after 2.35ns and 3.1ns respectively. When these forces are left out from the computation, the results, at the same time steps, are given in figures 4c and 4d. The logarithm of the strain energy is used to distinguish the high strain regions (darker) from the low strain (lighter) zones. Again, the dynamics, fracture patterns, and damage are drastically influenced by the inclusion of the nanoscale effects via van der Waals forces.

Test 3: CNT pullout from an explicitly modeled polymer matrix

In this simulation we model a small volume of composite material containing two CNTs surrounded by polymer molecules. The polymer fibers are held in place at the outer radius (figure 5a) while a tensile load is applied to the free end of one of the CNTs. The boundary conditions for the model of molecular chains and carbon nanotubes are as follows: the left CNT has an imposed displacement with constant velocity of 25nm/ns on

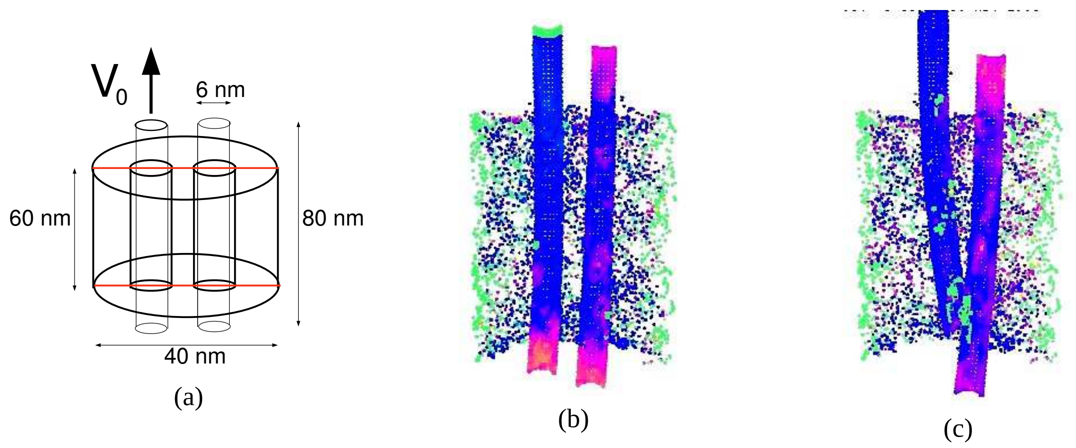


FIGURE 5. Pull-out test of a CNT from a polymer matrix. The polymer chains are explicitly modeled inside the cylindrical region shown in (a). The left CNT is pulled with constant velocity v_0=25nm/ns. The logarithm of the strain energy after 0.35ns is shown in (b), and after 0.67ns in (c). Only half of the system is shown in (b) and (c). The van der Waals forces are included. In this case the fibers have zero initial velocities.

its top part. The nodes in the bundle of polymer chains have zero displacements outside a radius of 20nm. We observe out-of-plane motion for the tubes. This complex non-symmetrical motion of the system is due to the non-symmetry of the explicitly modeled polymer chains. The van der Waals forces between the tubes, fibers and tubes, and among the fibers, influence the deformation of the entire structure. One can also notice a shielding effect of the CNT at rest that protects the right side of the matrix from damage propagation.

CONCLUSIONS

The primary advantages of the peridynamic approach for detailed modeling of the mechanics of nanoscale deformations include:

- Ability to model complex patterns of crack initiation and growth.
- Meshfree numerical implementation (the Emu code, see [7]) that avoids the need for finite element meshes.
- Contact between geometrically irregular bodies is modeled by short-range forces between nodes, without the need for complex contact algorithms.
- The underlying equations naturally allow for long-range forces, such as van der Waals forces, that are important at small size scales.
- Since the method is a continuum mechanics approach, it does not require a detailed knowledge of interatomic potentials or the mathematical difficulties of quantum mechanical calculations.

To exploit these capabilities we are developing algorithms to easily create detailed models of nanoscale structures such as the OOP generator for polymer molecules and nanotubes described above.

ACKNOWLEDGMENTS

F. Bobaru acknowledges the support from the Computer Science Research Institute at Sandia National Laboratory, Albuquerque, New Mexico, through a summer research fellowship, and from the University of Nebraska-Lincoln through a Fling Faculty Research Fellowship seed grant that facilitated this research.

REFERENCES

1. D. Qian, G.J. Wagner, R.S. Ruoff, M.F. Yu, and W.K. Liu, *Applied Mechanics Reviews* **55**(2), 495-533 (2002).
2. S.A. Silling, *Journal of the Mechanics and Physics of Solids* **48**, 175-209 (2002).
3. S.A. Silling, M. Zimmermann, and R. Abeyaratne, *Journal of elasticity* (in press).
4. S.A. Silling and E. Askari, under review (2004).
5. V.K. Decyk, C.D. Norton, and B.K. Szymanski, *Computer Physics Communications* **115**, 9-17 (1998).
6. Y-H. Kim, I-H. Lee, and R.M. Martin, *Computer Physics Communications* **131**, 10-25 (2000).
7. http://www.sandia.gov/emu/emu.htm