Nonlinear Basis for Dynamics

Conor Rowan

March 2024

Contents

1 Introduction 1

2 Lagrangian Formulation 2

3 Generic Nonlinear Discretization 4

4 Finding a Stationary Point 4

5 Equivalence with Weak Form 8

6 Bad News 9

7 Collocation Method 9

8 Iterative Adaptivity 10

9 Last Resort: Finite Differences? 12

1 Introduction

Here we document a quest for a useful and novel method of using nonlinear basis expansions of solutions to partial differential equations for time dependent problems. The utility of a nonlinear basis is that the spatial shape functions are not fixed, and can be fit in the solution process to optimally capture the physics of the problem at hand. For many statics problems in solid mechanics, a minimum of a “total potential energy” governs the solution. For any spatial basis, such a minimum can be found, and is even guaranteed to be unique in some situations. This includes a basis which is a very poor approximation of the true solution–the minimum of the total potential simply says “this is the best you can do with what you got!” That being said, there is a straightforward correspondence between lower energy values and better solutions. So the minimum of the energy for a solution approximated with a good basis is lower than that of a bad basis. At the end of the day, the notions of “good” and “bad” relate to how well the boundary conditions and strong form of the governing equations are satisfied. When a nonlinear discretization is used, there are degrees of freedom which control the form of the spatial basis. When we minimize the total potential, these degrees of freedom can be updated in search of lower and lower energy values, thus taking full advantage of the nonlinearity to come up with a good solution. As is shown here, this is not the case in the dynamics–even for problems whose elasticity is variational, there is no energy-like functional whose minimum corresponds to a solution. Thus, we have to be careful to construct a method which adapts the basis to find good solutions. The twists and turns chronicled below indicate there may not be such an elegant formulation for dynamics as there is for variational statics problems.

2 Lagrangian Formulation

We seek to model the dynamics of a 1D bar undergoing small strains with a nonlinear stress-strain relation. Assume that there is a strain energy density Ψ which is a function of the displacement gradient (strain) that governs the elastic response of the material. Our starting point will be the Lagrangian for the dynamical system. When an elastic bar of length L is fixed at $x = 0$ and has a time dependent force applied at $x = L$, the Lagrangian is

$$\mathcal{L} = \int_0^T \int_0^L \frac{1}{2}\rho\left(\frac{\partial u}{\partial t}\right)^2 - \Psi\left(\frac{\partial u}{\partial x}(x,t)\right) + F(t)u(L,t)dxdt$$

The first term in the integral is the kinetic energy of the bar, and the second two terms are the strain energy and external work respectively. A stationary point of the Lagrangian governs the dynamics of the system. Note that the displacement solution in space and time we are after is not guaranteed to be a minimizer of the Lagrangian, as is the case with the total potential energy in statics. This is a subtle point which has important implications for how dynamics problems can be solved–we cannot discretize the displacement in space and time and directly minimize the Lagrangian to find a solution, as the solution may in fact be a saddle point. For a saddle point problem, a gradient descent algorithm could make the Lagrangian arbitrarily small. As it turns out, saddle point solutions are not limited to pathological cases. For example, consider a linearly elastic bar with $\rho = E = 1$ and no external forces. The Lagrangian for this problem is

$$\mathcal{L} = \int_0^T \int_0^L \frac{1}{2}\left(\frac{\partial u}{\partial t}\right)^2 - \frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2 dxdt$$

Consider some general space-time discretization $u(x,t) = \sum_i a_i N_i(x,t)$. Here, we treat the space and time variables equivalently, as we would in a problem of two spatial dimensions. Why not? It is not clear mathematically why space and time should always be distinguished in numerical solutions. There is the intuition that the time response of a system is “local,” in the sense that the future and distant past do not contribute to the system's response at the current time. This contrasts with statics problems, for which there is no direction of information propagation in the spatial coordinate, and the solution at one point has some influence on the solution everywhere in the domain. But mathematically, this simply says that a typical dynamics problem has two boundary conditions on one side of the time domain (the initial time), unlike boundary value problems in statics. Information propagates forward in time because an initial position and velocity uniquely determine the solution at later times. Perhaps apart from additional complication introduced by the Second Law of Thermodynamics, typical time variables differ from spatial ones in the distribution of boundary conditions. Either way, we plug in the space-time discretization to the Lagrangian for the linearly elastic bar and obtain

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}a_i a_j\left(\int_0^T\int_0^L \frac{\partial N_i}{\partial t}\frac{\partial N_j}{\partial t}dxdt\right) - \frac{1}{2}a_i a_j\left(\int_0^T\int_0^L \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}dxdt\right)\\ &= \frac{1}{2}a_i a_j\Big(T_{ij} - V_{ij}\Big)\end{aligned}$$

For a multi-dimensional function of this sort, we can identify a saddle point by showing that the Hessian, or matrix of second derivatives, has both positive and negative eigenvalues. The Hessian of the Lagrangian is

$$H_{ij} = \frac{\partial^2 \mathcal{L}}{\partial a_i \partial a_j} = T_{ij} - V_{ij}$$

We need to pick a discretization in order to compute the eigenvalues. Let's go with

$$N_i(x,t) = \sin\left(\frac{f(i)\pi x}{L}\right)\sin\left(\frac{g(i)\pi t}{T}\right)$$

The functions f and g are just to indicate that there is some re-shaping operation to turn what is mostly simply written as a matrix of indexed shape functions into a vector. This basis discretizes the problem with some boundary conditions we are not interested in. What is interesting is that when the Hessian is computed with this discretization, it has both positive and negative eigenvalues. This demonstrates that the simplest elastic dynamical system is a saddle point problem for which the usual minimization methods cannot be used. This makes the use of a nonlinear basis tricky because energy methods are the most straightforward to get the adaptivity properties we are looking for. The value of the energy can be further reduced by adapting the basis, but weak formulations of the dynamics problem can be satisfied for any given basis. Thus, it is not clear that a weak formulation will pick the "best" basis. We thus need to think carefully about how to force the method to choose an optimal basis.

3 Generic Nonlinear Discretization

With either spectral or finite element methods, the displacement is discretized with

$$u(x,t) = \sum_i a_i(t) N_i(x)$$

The spatial shape functions N_i are known, and the time-dependent coefficients a_i are to be determined with the physics of the problem. Our challenge is to explore using neural network or other nonlinear discretizations in the context of dynamics. In the literature, dynamics problems are solved with collocation methods by finding the parameters of a neural network such that the strong form error is minimized. Neural network bases are interesting in the case of dynamics because of their ability to capture localized behavior in the solution. This is especially the case when some localized behavior translates in time, as this response could not be captured by a single basis function in the set, even if the spatial structure up to a shift is constant in time. Intuitively, it is wasteful to use many basis functions to capture the translation of some coherent spatial phenomenon when a single shape function could be shifted in time. But by including parameters which determine the shape functions themselves, we cannot avoid nonlinear dependence of the solution on the parameters. Neural networks are simple and flexible ways to construct a discretization of this sort. We will explore approximating the displacement as a neural network with time-dependent parameters. The only fundamental difference between this and the typical discretization mentioned above is that the solution is a nonlinear function of the time-dependent parameters. This discretization takes the very general form

$$u(x,t) = N\Big(x, \underline{\theta}(t)\Big)$$

4 Finding a Stationary Point

The displacement is approximated with a neural network, so the "coordinates" of the problem are the parameters $\underline{\theta}(t)$. Governing equations for the dynamical system can be obtained by finding a stationary point of the Lagrangian. The calculus of variations says that a functional $\mathcal{L}$ is minimized when

$$\delta\mathcal{L} := \frac{\partial}{\partial\epsilon}\Big(\mathcal{L}(\underline{\theta}(t) + \epsilon\underline{\eta}(t))\Big)\Bigg|_{\epsilon=0} = 0 \quad \forall\underline{\eta}(t) \text{ s.t. } \underline{\eta}(0) = \underline{\eta}(T) = 0$$

This is the continuous equivalent of saying that a stationary point is obtained when the gradient of a function is zero. Zero gradient is recognized at some point by checking whether small nudges in any direction change the value

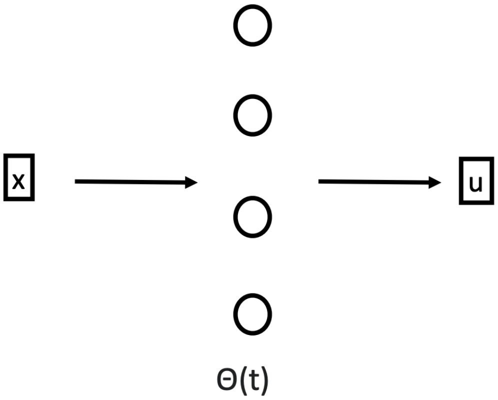

Figure 1: The spatial coordinate x is input into the network and is acted on by parameter $\underline{\theta}$ which vary in time to produce the displacement.

of the function. This is what the calculus of variations is saying, but in regards to functionals–a stationary point is obtained when the functional does not change its value from small nudges to the coordinates. The "variations" $\underline{\eta}$ are arbitrary except that they must not violate boundary/initial conditions. It is always a little bit strange why $\underline{\eta}(T) = 0$ when only an initial displacement is specified. The reason seems to be that the final value of the displacement is uniquely determined by the initial displacement and velocity. This implies that the variations should actually come from the space of functions such that $\underline{\dot{\eta}}(0) = 0$ so that the initial velocity is not varied. But nowhere in the following derivation do we make use of the values of the time derivatives of the variation at the boundaries, so neglecting this apparent restriction should not have any effect. It is necessary to proceed carefully through the calculus of variations here because the usual derivations do not straightforwardly carry through when the solution is a nonlinear function of the parameters. The Lagrangian for nonlinear bar and neural network discretization is

$$\mathcal{L} = \int_0^T \int_0^L \frac{1}{2}\rho\bigg(\frac{\partial}{\partial t}N(x, \underline{\theta}(t))\bigg)^2 - \Psi - F(t)N(x, \underline{\theta}(t))dxdt$$

We will compute the first variation in steps: first we plug in the varied parameters, then take the time derivative with respect to ϵ, then we will evaluate the resulting expression at $\epsilon = 0$. We will call the varied parameters $\underline{\tilde{\theta}} = \underline{\theta} + \epsilon\underline{\eta}$ to simplify some notation. The utility of this will be seen shortly. It is clear that $\underline{\tilde{\theta}}\Big|_{\epsilon=0} = \underline{\theta}$. Plugging in the varied parameters, we have

$$\mathcal{L}(\underline{\theta} + \epsilon\underline{\eta}) = \int_0^T \int_0^L \frac{1}{2}\rho\bigg(\frac{\partial}{\partial t}N(x, \underline{\theta} + \epsilon\underline{\eta})\bigg)^2 - \Psi(\underline{\theta} + \epsilon\underline{\eta}) - F(t)N(x, \underline{\theta} + \epsilon\underline{\eta})dxdt$$

The time derivative in the kinetic energy term must be computed with the chain rule. This expression reads

$$\mathcal{L}(\underline{\theta} + \epsilon\underline{\eta}) = \int_0^T \int_0^L \frac{1}{2}\rho\left(\frac{\partial N}{\partial \tilde{\theta}_i}\left(\frac{\partial \theta_i}{\partial t} + \epsilon\frac{\partial \eta_i}{\partial t}\right)\right)^2 - \Psi(\underline{\tilde{\theta}}) - F(t)N(x, \underline{\tilde{\theta}})dxdt$$

Now that we have written how the varied Lagrangian, we compute its derivative with respect to ϵ:

$$\frac{\partial}{\partial \epsilon}\mathcal{L}(\underline{\tilde{\theta}}) = \int_0^T \int_0^L \rho\left(\frac{\partial N}{\partial \tilde{\theta}_i}\left(\frac{\partial \theta_i}{\partial t} + \epsilon\frac{\partial \eta_i}{\partial t}\right)\right)\left(\frac{\partial N}{\partial \tilde{\theta}_j}\frac{\partial \eta_j}{\partial t} + \frac{\partial^2 N}{\partial \tilde{\theta}_j \partial \tilde{\theta}_k}\eta_j\left(\frac{\partial \theta_k}{\partial t} + \epsilon\frac{\partial \eta_k}{\partial t}\right)\right) \\ - \frac{\partial \Psi}{\partial \tilde{\theta}_i}\eta_i + F(t)\frac{\partial N}{\partial \tilde{\theta}_i}\eta_i dxdt$$

The next step is to evaluate this at $\epsilon = 0$ to consider only small variations around the stationary point. This simplifies the expression:

$$\frac{\partial}{\partial \epsilon}\mathcal{L}(\underline{\tilde{\theta}})\bigg|_{\epsilon=0} = \int_0^T \int_0^L \rho\left(\frac{\partial N}{\partial \theta_i}\frac{\partial \theta_i}{\partial t}\right)\left(\frac{\partial N}{\partial \theta_j}\frac{\partial \eta_j}{\partial t} + \frac{\partial^2 N}{\partial \theta_j \partial \theta_k}\eta_j\frac{\partial \theta_k}{\partial t}\right) - \frac{\partial \Psi}{\partial \theta_i}\eta_i + F\frac{\partial N}{\partial \theta_i}\eta_i dxdt$$

We have used the fact that evaluating the varied parameters at $\epsilon = 0$ simply returns the parameters. This is how we get rid of derivatives involving $\tilde{\theta}$. This is a condition for a stationary point of the Lagrangian in terms of the time-dependent neural parameters, but it is not useful in this form. We want to derive a “strong” version of this principle in the form of a system of ordinary differential equations in time. This can be accomplished in the following way. First, we transfer the time derivative off of the variation η with integration by parts. The boundary term does not contribute because $\eta(0) = \eta(T) = 0$ as noted previously. The equation becomes

$$= \int_0^T \int_0^L -\rho\frac{\partial}{\partial t}\left(\frac{\partial N}{\partial \theta_i}\frac{\partial \theta_i}{\partial t}\frac{\partial N}{\partial \theta_j}\right)\eta_j + \rho\frac{\partial N}{\partial \theta_i}\frac{\partial \theta_i}{\partial t}\frac{\partial^2 N}{\partial \theta_j \partial \theta_k}\eta_j\frac{\partial \theta_k}{\partial t} - \frac{\partial \Psi}{\partial \theta_j}\eta_j + F\frac{\partial N}{\partial \theta_j}\eta_j dxdt$$

$$= \int_0^T \left(\int_0^L -\rho\frac{\partial}{\partial t}\left(\frac{\partial N}{\partial \theta_i}\frac{\partial \theta_i}{\partial t}\frac{\partial N}{\partial \theta_j}\right) + \rho\frac{\partial N}{\partial \theta_i}\frac{\partial \theta_i}{\partial t}\frac{\partial^2 N}{\partial \theta_j \partial \theta_k}\frac{\partial \theta_k}{\partial t} - \frac{\partial \Psi}{\partial \theta_j} + F\frac{\partial N}{\partial \theta_j}dx\right)\eta_j dt = 0$$

All terms in the time integral have a common factor of the variation η. A theorem from the calculus of variations says that because the variation is arbitrary, the term it multiplies in the integrand must be zero pointwise. This makes sense if we imagine η_j being a set of functions which are zero except for a small neighborhood around each time point t. This gives us the strong form of the governing equations:

$$\int_0^L -\rho\frac{\partial}{\partial t}\left(\frac{\partial N}{\partial\theta_i}\frac{\partial\theta_i}{\partial t}\frac{\partial N}{\partial\theta_j}\right)+\rho\frac{\partial N}{\partial\theta_i}\frac{\partial\theta_i}{\partial t}\frac{\partial^2 N}{\partial\theta_j\partial\theta_k}\frac{\partial\theta_k}{\partial t}-\frac{\partial\Psi}{\partial\theta_j}+F\frac{\partial N}{\partial\theta_j}dx=0$$

We can distribute the time derivative on the first term to obtain

$$\int_0^L -\rho\frac{\partial^2 N}{\partial\theta_i\partial\theta_k}\frac{\partial\theta_k}{\partial t}\frac{\partial\theta_i}{\partial t}\frac{\partial N}{\partial\theta_j}-\rho\frac{\partial N}{\partial\theta_i}\frac{\partial^2\theta_i}{\partial t^2}\frac{\partial N}{\partial\theta_j}-\rho\frac{\partial N}{\partial\theta_i}\frac{\partial\theta_i}{\partial t}\frac{\partial^2 N}{\partial\theta_j\partial\theta_k}\frac{\partial\theta_k}{\partial t}$$
$$+\rho\frac{\partial N}{\partial\theta_i}\frac{\partial\theta_i}{\partial t}\frac{\partial^2 N}{\partial\theta_j\partial\theta_k}\frac{\partial\theta_k}{\partial t}-\frac{\partial\Psi}{\partial\theta_j}+F(t)\frac{\partial N}{\partial\theta_j}dx=0$$

Mercifully, the third and fourth terms cancel out. We note that the parameters θ do not depend on space, so they can be factored out of the spatial integration. The index j is free defining a system of ODE's. Multiplying through by -1, the governing equation becomes

$$\ddot{\theta}_i\left(\int\rho\frac{\partial N}{\partial\theta_i}\frac{\partial N}{\partial\theta_j}dx\right)+\dot{\theta}_k\dot{\theta}_i\left(\int\rho\frac{\partial^2 N}{\partial\theta_i\partial\theta_k}\frac{\partial N}{\partial\theta_j}dx\right)+\int\frac{\partial\Psi}{\partial\theta_j}dx=\int F(t)\frac{\partial N}{\partial\theta_j}dx$$

This is a system of ODE's for the time-dependent parameters. As a sanity check, we can verify that this equation reduces to the usual equations of motion for a linear basis expansion and a simple constitutive model. Consider $N(x,t)=\sum_n\theta_n(t)w_n(x)$ and $\Psi=\frac{1}{2}\left(\frac{\partial N}{\partial x}\right)^2$. Then we have that

$$\int\frac{\partial\Psi}{\partial\theta_j}dx=\int\frac{\partial}{\partial\theta_j}\left(\frac{1}{2}\sum_n\sum_m\theta_n\theta_m\frac{\partial w_n}{\partial x}\frac{\partial w_m}{\partial x}\right)dx=\theta_n\left(\int\frac{\partial w_n}{\partial x}\frac{\partial w_j}{\partial x}dx\right)$$

This is the usual expression for the internal force of a linearly elastic system where the quantity in parentheses is the stiffness matrix. Now we turn to the term with second time derivative of the parameters. We can plug in the linear discretization and see that

$$\int\rho\frac{\partial N}{\partial\theta_i}\frac{\partial N}{\partial\theta_j}dx=\int\rho w_i w_j dx$$

This is the usual expression for the mass matrix which multiplies the second time derivative of the parameters. It is simple to see that the forcing term returns what we expect from the linear expansion. Finally, we turn our attention to the second term in the governing equation:

$$\dot{\theta}_k\dot{\theta}_i\left(\int\rho\frac{\partial^2 N}{\partial\theta_i\partial\theta_k}\frac{\partial N}{\partial\theta_j}dx\right)=0$$

for the linear expansion. This is because we take two derivatives with respect to the parameters. For the neural network, however, this term is nonzero. We expect that this term does not appear for the governing equations with the usual linear discretization, not least because it has a quadratic nonlinearity in time. Equations which are nonlinear in time are quite rare, but in this case, it is the price we pay for a nonlinear basis expansion of the displacement. Note that the matrix/tensor quantities in parentheses are functions of time implicitly through the parameters. The following definitions will help simplify the governing equation:

$$M_{ji}\Big(\underline{\theta}(t)\Big) := \int \rho \frac{\partial N}{\partial \theta_j}\frac{\partial N}{\partial \theta_i} dx$$

$$A_{ikj}\Big(\underline{\theta}(t)\Big) := \int \rho \frac{\partial^2 N}{\partial \theta_i \partial \theta_k}\frac{\partial N}{\partial \theta_j} dx$$

$$F_j^{int}\Big(\underline{\theta}(t)\Big) := \int \frac{\partial \Psi}{\partial \theta_j} dx$$

$$F_j^{ext}\Big(t, \underline{\theta}(t)\Big) := \int F(t)\frac{\partial N}{\partial \theta_j} dx$$

The governing equation is both nonlinear in the parameters and nonlinear time. Finally, we arrive at a reasonable form of the governing equation for the parameters of the network. This reads

$$M_{ji}\Big(\underline{\theta}(t)\Big)\ddot{\theta}_i + A_{ikj}\Big(\underline{\theta}(t)\Big)\dot{\theta}_i\dot{\theta}_k + F_j^{int}\Big(\underline{\theta}(t)\Big) = F_j^{ext}\Big(\underline{\theta}(t)\Big)$$

5 Equivalence with Weak Form

The strong form of the governing equation for the 1D inelastic bar is

$$\rho\frac{\partial^2}{\partial t^2}u = \frac{\partial}{\partial x}\left(\frac{\partial \Psi}{\partial \epsilon}\right) + F(t)$$

When we discretize the displacement with $u(x,t) = N(x, \underline{\theta}(t))$, this becomes

$$\rho\frac{\partial}{\partial t}\left(\frac{\partial N}{\partial \theta_i}\frac{\partial \theta_i}{\partial t}\right) = \rho\frac{\partial^2 N}{\partial \theta_i \partial \theta_k}\frac{\partial \theta_i}{\partial t}\frac{\partial \theta_k}{\partial t} + \rho\frac{\partial N}{\partial \theta_i}\frac{\partial^2 \theta_i}{\partial t^2} = \frac{\partial}{\partial x}\left(\frac{\partial \Psi}{\partial \epsilon}\right) + F(x,t)$$

We can use Galerkin projection in the case of a solution defined on a manifold, which states that residual (error) of the strong form error is orthogonal to the local tangent of the approximation. The tangents to the space in which the displacements are approximated are $\partial N/\partial \theta_j$. The weak form of the governing equation is

$$\int \rho\frac{\partial^2 N}{\partial \theta_i \partial \theta_k}\frac{\partial \theta_i}{\partial t}\frac{\partial \theta_k}{\partial t}\frac{\partial N}{\partial \theta_j} dx + \int \rho\frac{\partial N}{\partial \theta_i}\frac{\partial^2 \theta_i}{\partial t^2}\frac{\partial N}{\partial \theta_j} dx = \int \frac{\partial}{\partial x}\left(\frac{\partial \Psi}{\partial \epsilon}\right)\frac{\partial N}{\partial \theta_j} + F(x,t)\frac{\partial N}{\partial \theta_j} dx$$

Factoring out the displacement coordinates which do not depend on space and using integration by parts,

$$\frac{\partial\theta_i}{\partial t}\frac{\partial\theta_k}{\partial t}\int\rho\frac{\partial^2 N}{\partial\theta_i\partial\theta_k}\frac{\partial N}{\partial\theta_j}dx+\frac{\partial^2\theta_i}{\partial t^2}\int\rho\frac{\partial N}{\partial\theta_i}\frac{\partial N}{\partial\theta_j}dx=-\int\frac{\partial\Psi}{\partial\left(\frac{\partial N}{\partial x}\right)}\frac{\partial^2 N}{\partial x\partial\theta_j}+F(x,t)\frac{\partial N}{\partial\theta_j}dx$$

Finally, we can run the chain rule in reverse to rid of the strain derivative of the energy density, and move this term to the other side of the equation. In doing the integration by parts, we assumed that there is no end traction applied at $x = L$. We see that we get back the governing equation derived from the calculus of variations:

$$\frac{\partial\theta_i}{\partial t}\frac{\partial\theta_k}{\partial t}\int\rho\frac{\partial^2 N}{\partial\theta_i\partial\theta_k}\frac{\partial N}{\partial\theta_j}dx+\frac{\partial^2\theta_i}{\partial t^2}\int\rho\frac{\partial N}{\partial\theta_i}\frac{\partial N}{\partial\theta_j}dx+\int\frac{\partial\Psi}{\partial\theta_j}dx=\int F(x,t)\frac{\partial N}{\partial\theta_j}dx$$

6 Bad News

Unfortunately, the above appears to amount to nothing more than an elaborate exercise in the calculus of variations and weak solutions to PDE's. Curiosity got the best of the author–it should have been clear from the onset that finding a stationary point to the Lagrangian does not necessarily find an optimal basis expansion. The Lagrangian can be made stationary for any time-independent basis, no matter how poor the approximation is. It seems that there is no sense in which the governing equation we derived will make the Lagrangian "more stationary" in the sense of adapting the basis to obtain good approximations. Although this method has not been implemented, there is no notion of a "good" solution built into this method. We should expect that a solution is obtained as some random local extremum, and we have no reason to think this is the one which globally minimizes the energy or strong form error. So far we still do not have a good way to make use of the nonlinear expansion of the displacement.

7 Collocation Method

Another alternative is to directly minimize the error of the strong form of the governing equations. This is definitely not a saddle point problem, so a minimization procedure should be able to be used. Minimizing the error with a collocation method should lead to the basis adaptivity we are looking for. It does not, however, lead to a system of ordinary differential equations, as a gradient descent procedure would be used. This method is logical, but has already been established as a technique for solving dynamics problems. Though it will not be pursued here, we can briefly illustrate the method for completeness. The collocation error and its gradient are

$$\mathcal{E}=\int_0^T\int_0^L\frac{1}{2}\left(\rho\frac{\partial^2}{\partial t^2}N-\frac{\partial}{\partial x}\frac{\partial\Psi}{\partial\epsilon}-F\right)^2dxdt$$

$$\frac{\partial \mathcal{E}}{\partial \theta_m} = \int_0^T \int_0^L \left(\frac{\partial^2}{\partial t^2} N - \frac{\partial}{\partial x} \frac{\partial \Psi}{\partial \epsilon} - F \right) \left(\rho \frac{\partial^3 N}{\partial t^2 \partial \theta_m} - \frac{\partial^3 \Psi}{\partial \epsilon \partial x \partial \theta_m} \right) dxdt$$

Note that a hybrid optimization problem could also be introduced, whereby we simultaneously minimize the norm of the residual equations and the strong form error. The residual equations are defined by

$$\frac{\partial \mathcal{L}}{\partial \theta_m} = 0$$

This is the condition for a stationary point of the Lagrangian discretized in terms of the parameters of the nonlinear basis. This optimization problem is called hybrid because it incorporates the weak and strong form of the governing equations. The objective and its gradient are

$$\Pi^h = \frac{1}{2} \sum_m \left(\frac{\partial \mathcal{L}}{\partial \theta_m} \right)^2 + \mathcal{E}$$

$$\frac{\partial \Pi^h}{\partial \theta_n} = \sum_m \frac{\partial L}{\partial \theta_m} \frac{\partial^2 \mathcal{L}}{\partial \theta_m \partial \theta_n} + \frac{\partial \mathcal{E}}{\partial \theta_n}$$

Here we have adaptivity, and potentially reap some of the rewards of weak formulations of the problem, but at the cost of having to form a Jacobian matrix (which we do not write out explicitly). Forming this matrix can be quite expensive. It would be interesting to study whether the hybrid formulation outperforms a pure colllocation method.

8 Iterative Adaptivity

Our situation is becoming dire. We have yet to come up with a satisfactory solution to our problem. Dear reader, I implore you, please do not lose hope. In the absence of an energy functional to minimize, we are forced to use the strong form error in some fashion to drive adaptivity of the basis. Minimizing the collocation error is one way to do this, but this method is fruitless from the standpoint of novelty. It is just a little too obvious...so we will cook up something bizarre instead. Let's call this method "iterative adaptivity[1]" The idea is as follows: we perform a staggered optimization problem where at each time step, we fix the basis and find coefficients at the next time step, then with these fixed coefficients, we adapt the basis to minimize the strong form error. The nonlinear basis expansion of the displacement will be written as

$$u(x,t) = \sum_i \alpha_i(t) h_i(x, \beta(t))$$

[1] This is probably in the literature somewhere under another name. I have not read up on adaptive methods in computational mechanics.

The full set of parameters is $\theta = \alpha + \beta$, but we have split them into weights α and shape function parameters β. The governing equations for fixed β are obtained by integrating the strong form against the spatial shape functions:

$$\int_0^L \rho \ddot{u} h_i(x;\beta(t))dx = \int_0^L \frac{\partial \sigma}{\partial x} h_i(x;\beta(t)) + F h_i(x;\beta(t))dx$$

With zero end traction, integration by parts yields

$$\int_0^L \rho \ddot{u} h_i(x;\beta(t))dx + \int_0^L \sigma \frac{\partial h_i}{\partial x}(x;\beta(t))dx = \int_0^L F h_i(x;\beta(t))dx$$

Plugging in this discretization of the displacement, we obtain

$$\sum_j \ddot{\alpha}_j \left(\int_0^L h_i(x;\beta(t)) h_j(x;\beta(t))dx \right) + \int_0^L \sigma \left(\sum_j \alpha_j \frac{\partial h_j}{\partial x} \right) h_i(x;\beta(t))dx = \int_0^L F h_i(x;\beta(t))dx$$

$$\implies M_{ij}(\beta(t))\ddot{\alpha}_j + F_i^{int}(\alpha(t), \beta(t)) = F_i^{ext}(t, \beta(t))$$

The simplest way to illustrate this method is to use central differencing to approximate the acceleration term. The acceleration at the next time point can be computed as

$$\underline{\alpha}^{n+1} = \underline{\underline{M}}^{-1}(\beta^n)\Big(\underline{F}^{ext}(t^n;\beta^n) - \underline{F}^{int}(\alpha^n, \beta^n)\Big) + 2\underline{\alpha}^n - \underline{\alpha}^{n-1}$$

For the evaluation of the next acceleration, the shape function parameters are held fixed. Having solved for the next acceleration in this way, the next set of shape function parameters are computed by holding the $\underline{\alpha}^{n+1}$ fixed and adjusting the shape function weights. The criteria for adjusting shape functions is the strong form residual. The residual at the next time step is

$$R^{n+1}(\underline{\beta}^{n+1}) = \int_0^L \left(\frac{\partial \sigma}{\partial x}(x;\underline{\alpha}^{n+1}, \underline{\beta}^{n+1}) + F(x, t^{n+1}) - \rho \ddot{u}^{n+1}(x;\underline{\alpha}^{n+1}, \underline{\beta}^{n+1}) \right)^2$$

We have a continuous representation of the displacement, and hence the stress, in terms of the spatial coordinate x. Thus, the first term in the strong form residual can be computed exactly. Similarly, we have an analytic representation of the distributed force, so its value at the next time step is computed exactly. We do not, however, have a continuous representation of the displacement in time, thus the acceleration needs to be approximated in terms of past time steps. Using the same central difference scheme, we see that

$$\ddot{u}^{n+1}(x) = \frac{1}{\Delta t^2}\left(\sum_i \alpha_i^{n+1} h_i(x;\beta^{n+1}) - 2\sum_i \alpha_i^n h_i(x;\beta^n) + \sum_i \alpha_i^{n-1} h_i(x, \beta^{n-1}) \right)$$

Note that the sums in parentheses cannot be combined because the spatial shape functions are different at each point in time through parameters β. This makes the resulting equations quite complicated. The "optimal" shape functions for the next time step are then computed by solving a minimization problem, governed by

$$\frac{\partial R^{n+1}}{\partial \beta_m^{n+1}} = 0$$

What is the rationale for holding the weights α^{n+1} fixed during this process? Well for one, we would be solving a collocation problem if we allowed them to vary, such that the time integration would not be needed. Intuitively, fixing the weights on the shape functions makes sense if they only need slight updates to find a solution at the next time step which minimizes the strong form error. The time integration finds appropriate weights on slightly suboptimal shape functions, then this error minimization step tweaks the shape functions as necessary. Note that the implementation of this method is fairly involved: it is not even simple to plot the solution as a function of time because the shape functions are constantly changing! Furthermore, an o ptimization problem needs to be solved at every time step, and the mass matrix recomputed according to the shape function updates. The changing β parameters also prohibit pre-computing any quantities in the internal force vector. We expect this method to be complex and expensive. Perhaps most importantly, it is not clear what advantage it offers over a pure collocation approach. Sure, there are certain theoretical guarantees on the performance/stability of Galerkin methods[2] which may not apply to a collocation solution, but if we see the strong form error as the gold standard of PDE solutions, it is tempting to think a collocation method is strictly better than this weird hybrid. Fixing the weights on the shape functions at each time step imposes a restriction on the discretization which collocation methods do not have. Plus, implementing the iterative adaptivity method is immensely difficult compared to a collocation method.

9 Last Resort: Finite Differences?

We have shown that a lot of the machinery for solving dynamics problems is not well-suited for the adaptive basis discretization. One method that has not been touched on is the finite difference method. Is there any possibility of usefully applying the finite difference method (FDM) with neural network bases or other nonlinear discretizations? To the extent that finite difference methods rely on storing nodal values of the solution, the answer is no. If somehow nodal values were discretized with an underlying basis which had fewer degrees of freedom than there were nodal values at which to finite difference, then maybe there would be some logic to adaptivity. But this would be a very unusual situation. Plus, with the global basis, it would make sense to use a collocation method

[2] Don't ask me exactly what they are though.

with analytic computations of the derivatives instead of finite differencing. So from the author's humble point of view, there is little hope of using FDM with the nonlinear basis. But, since we have already surveyed a number of interesting topics in dynamics, we will continue this thread. We can prove that a traditional finite difference method is equivalent to a particular Petrov-Galerkin method. We will do this for the case of a wave equation, or the equation of motion for a linearly elastic bar. The continuous form of the governing equation is

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + F(x,t)$$

We assume the density and modulus are unity for simplicity. A spatial finite difference scheme can be written as

$$\ddot{u}(x_j,t) = \frac{1}{\Delta x^2}\Big(u(x_{j+1},t) - 2u(x_j,t) + u(x_{j-1},t)\Big) + F(x_j,t)$$

We leave the problem continuous on time, noting that the second time derivatives could be approximated in the same way as the second spatial derivatives. The 1D spatial domain is broken up into a grid, and the solution is stored at nodal points $(x_1, x_2, \ldots, x_N)$. The time update to the displacement at grid point x_j is driven by the distributed force at that point $F(x_j,t)$, and the second spatial derivative of the displacement (which should be interpreted as the gradient of the stress) evaluated using one point to the left and right. This is a central difference method for the second derivative. We want to show that this is equivalent to a particular weak form of the governing equations. Consider the linear finite element discretization shown in Figure 2. The shape function associated with nodal position x_j spans two elements (where an element is defined by the space between grid points). For simplicity, we assume that the nodal spacing (element size) is the constant Δx. Using the piecewise linear shape functions, the displacement can be written as

$$u(x,t) = \sum_i u_i(t) N_i(x)$$

The time dependent coefficients $u_i(t)$ are the nodal degrees of freedom evaluated at time t. In other words, $u_i(t) = u(x_i,t)$ by the definition of the finite element basis. This is our discretization of the solution field. We construct the weak form of the governing equations by substituting the discretization into the governing equations and integrating over the spatial domain against a set of test functions. When the space of test functions differs from that of the approximation space, the resulting weak form is called a Petrov-Galerkin method. In our case, this simply says that the test functions are not finite element basis functions we use to construct the solution. For simplicity, assume that we have zero displacement boundary conditions at all instants of time on both ends of the bar. This means that only the interior nodes of the "mesh" correspond to unknown degrees of freedom. We will test against Dirac delta functions centered at each interior node. When the $j-th$ test function is $\delta(x - x_j)$, the Petrov-Galerkin weak form reads

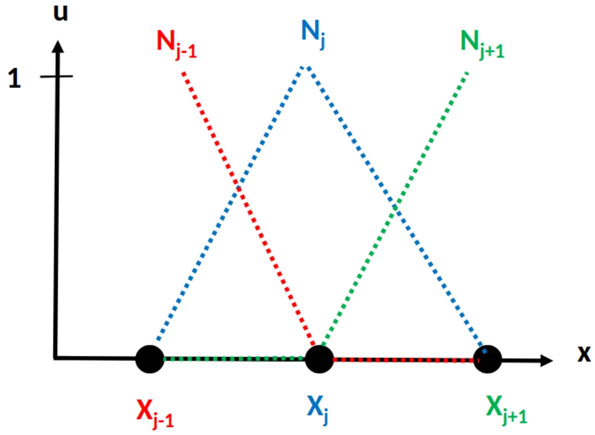

Figure 2: Piecewise linear finite element shape functions used to interpolate the displacement in space.

$$\int \left(\sum_i \ddot{u}_i N_i(x) \right) \delta(x - x_j) dx = \int \left(\sum_i u_i \frac{\partial^2 N_i}{\partial x^2} \right) \delta(x - x_j) dx + \int F(x,t) \delta(x - x_j) dx$$

We use the sifting property of the delta functions and the linearity of integration to simplify this to

$$\sum_i \ddot{u}_i N_i(x_j) = \sum_i u_i \frac{\partial^2 N_i}{\partial x^2}(x_j) + F(x_j, t)$$

By construction, the finite element shape functions have the "interpolation property," which states that $N_i(x_j) = \delta_{ij}$. The governing equation reduces

$$\ddot{u}_j = \ddot{u}(x_j, t) = \sum_i u_i \frac{\partial^2 N_i}{\partial x^2}(x_j) + F(x_j, t)$$

This looks awfully close to the finite difference method laid out earlier. However, we have to deal with the stress term, which involves second derivatives of the hat shape functions. This is a tricky thing because the second derivative of these piecewise linear shape functions is not well-defined. We can make a simplification: though at this point, we do not know how to compute second derivatives of these shape functions, we can say that

$$\sum_i u_i \frac{\partial^2 N_i}{\partial x^2}(x_j) = u_{j-1} \frac{\partial^2 N_{j-1}}{\partial x^2}(x_j) + u_j \frac{\partial^2 N_j}{\partial x^2}(x_j) + u_{j+1} \frac{\partial^2 N_{j+1}}{\partial x^2}(x_j)$$

The shape functions to the left and right of node x_j do not contribute to the displacement at x_j, but we do expect them to contribute to the curvature of the solution at this point. That more distant shape functions do not contribute is an approximation. Based on this assumption, we will approximate the second derivatives of these three shape functions using only their values at nodes x_{j-1}, x_j, and x_{j+1}. This involves using a forward scheme for x_{j-1} (looking at the displacement two nodes ahead), a centered scheme for x_j (looking to the left and right), and a backwards scheme for x_{j+1} (looking two nodes behind). We assume that the curvature is constant over the domain of each hat function. Mathematically, this is written as

$$\frac{\partial^2 N_{i-1}}{\partial x^2}(x_j) \approx \frac{1}{\Delta x^2}\Big(N_{i-1}(x_{i-1}) - 2N_{i-1}(x_i) + N_{i-1}(x_{i+1})\Big) = \frac{1}{\Delta x^2}(1-0+0) = \frac{1}{\Delta x^2}$$

$$\frac{\partial^2 N_{i}}{\partial x^2}(x_j) \approx \frac{1}{\Delta x^2}\Big(N_{i}(x_{i-1}) - 2N_{i}(x_i) + N_{i}(x_{i+1})\Big) = \frac{1}{\Delta x^2}(0-2+0) = -\frac{2}{\Delta x^2}$$

$$\frac{\partial^2 N_{i+1}}{\partial x^2}(x_j) \approx \frac{1}{\Delta x^2}\Big(N_{i+1}(x_{i-1}) - 2N_{i+1}(x_i) + N_{i+1}(x_{i+1})\Big) = \frac{1}{\Delta x^2}(0-0+1) = \frac{1}{\Delta x^2}$$

If these expressions look confusing, return to Figure 2 and think about how curvature of each shape function is being computed. We need three points to compute curvature, and the $j-1$, j, and $j+1$ grid points are used to approximate the curvature of each of three shape functions. By using these three points and assuming curvature is constant over the non-zero region of each hat function, we are implicitly fitting a quadratic polynomial to the three nodal points. When we plug this back into the weak form, we obtain

$$\ddot{u}_j(t) = \frac{1}{\Delta x^2}\Big(u_{j-1} - 2u_j + u_{j+1}\Big) + F(x_j,t)$$

This is exactly the finite difference scheme laid out in the beginning. We have shown that a piecewise linear discretization of the displacement tested against delta functions at the nodes gives rise to FDM when curvature of the shape functions is approximated in a particular way. In other words, finite difference methods are intimately related to Petrov-Galerkin methods. This bridges a gap between FDM and weak solution techniques for partial differential equations.

We can go even further. Consider a statics problem for which the inertial term disappears. Using the finite difference scheme we have constructed, the governing equation is

$$\frac{1}{\Delta x^2}\Big(-u_{j-1} + 2u_j - u_{j+1}\Big) = F(x_j,t)$$

This can be written as a matrix system of the following form:

$$\frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & \dots \\ -1 & 2 & -1 & 0 & 0 & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ 0 & 0 & -1 & 2 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}\begin{bmatrix} u(x_1) \\ u(x_2) \\ u(x_3) \\ u(x_4) \\ \vdots \end{bmatrix} = \begin{bmatrix} F(x_1) \\ F(x_2) \\ F(x_3) \\ F(x_4) \\ \vdots \end{bmatrix}$$

This is equivalent to the matrix system we arrive at with a finite element method bar discretized with piecewise linear shape functions and zero displacement boundary conditions. To see this note, note that the force vector from finite element methods is computed with

$$F_i = \int_0^L F(x)N_i dx \approx F(x_i)\int_0^L N_i dx = \frac{1}{\Delta x}F(x_i)$$

Here we assume a Bubnov-Galerkin method where the test functions are the same as those used in the discretization of the displacement. We approximate the distributed force as constant over the element, evaluating it at node x_i. The integral of the hat shape function can be computed from basic geometry. The stiffness matrix is

$$K_{ij} = \int_0^L \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} dx$$

When $i = j$, it is easy to see that

$$K_{ii} \text{ (no sum) } = \int_0^L \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} dx = 2\Delta x \left(\frac{1}{\Delta x}\right)^2 = \frac{2}{\Delta x}$$

The factor of 2 arises because the hat shape function associated with node i spans two elements. When $|i-j| > 1$, $K_{ij} = 0$ because the shape functions do not overlap at all. For $|i-j| = 1$, we have that

$$K_{ij} = \int_0^L \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} dx = -\Delta x \left(\frac{1}{\Delta x}\right)^2 = -\frac{1}{\Delta x}$$

This can be seen by noting that shape functions of adjacent indices overlap only in one element, and have slopes of opposite sign. This proves that the finite element problem is

$$\frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & \dots \\ -1 & 2 & -1 & 0 & 0 & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ 0 & 0 & -1 & 2 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}\begin{bmatrix} u(x_1) \\ u(x_2) \\ u(x_3) \\ u(x_4) \\ \vdots \end{bmatrix} = \Delta x \begin{bmatrix} F(x_1) \\ F(x_2) \\ F(x_3) \\ F(x_4) \\ \vdots \end{bmatrix}$$

which is equivalent to the matrix system obtained by the finite difference method when we divide both sides by Δx. It is likely the case that the equivalence between the finite element method and finite difference method holds for two dimensions when the domain is rectangular.