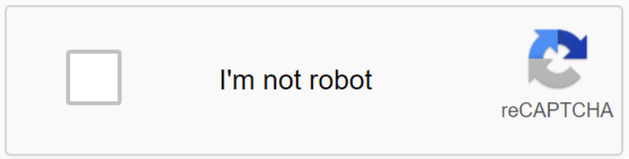

Continue

How to solve coupled partial differential equations in matlab

Commented: Bosnian kingdom on 2 Jun 2019 I try to solve these contctioned partial differential equations. N_u and F_p are the two dependent variables. Once and a space dimensions are there. Are these parabolic types? Should I use pdepe and if so how would the border conditions be given? Commented: darova on February 8th, 2020 I try to solve the following problem: This is a two-point limiting problem for unknown Q and P, which are both 2x2 arrays, and covariance arrays () are given (and are positive determined). Now I try to solve this in MATLAB using BVP4c, which theoretically should work, as this system has a solution (there is a theorem somewhere). To try to implement this, I reshaped both arrays into a large vector from another post that I saw trying to solve an array difference. The problem is the limitations, since they are written in this case. The MATLAB code I use is below: n = 2;m = 2; T = 1; N = 10; Sigma0 = eye (n); SigmaT = 0.25 * eye (n); A = [0 1; 0 0]; B = [0 0; 0 1]; P0 = [-1 -1; -1 0]; Q0 = [1 1; 1 0]; X0 = [P0; Q0]; Xmesh = linspace(0,T,N);solinit = bvpinit(Xmesh, X0(:));bcfun = @(X0,XT) bc(X0,XT,Sigma0,SigmaT);odefun = @(t,X) deriv(X,A,B); sun = bvp4c(odefun,bcfun,solinit); function Xdotvec = deriv(Xvec,A,B) n = 2; X = reshape (Xvec,2 *n,n); Q = X(1:n,1:n); P = X (n + 1: end, 1: end); Qdot = A * Q + Q * A ' - B * B'; Pdot = A * P + P * A ' + B * B'; Xdotvec = [Qdot(:); Pdot(:)];endfunction bcvec = bc(X0,XT,Sigma0,SigmaT) n = 2; X0 = reshape (X0.2 *n, n); Q0 = X0 (1:n,1:n); P0 = X0 (n +1:end, 1:n); Qinv0 = inv(Q0); Pinv0 = inv(P0); bc0 = Qinv0 + Pinv0 - inv(Sigma0); XT = reshape (XT.2 *n, n); QT = XT(1:n,1:n); PT = XT (n +1:end,1:end); QinvT = inv(QT); PinvT = inv(PT); bcT = QinvT + PinvT - inv (SigmaT); bcvec = [bc0(:), bcT(:)]; endThe resulting code does not provide a solution, and I'm not sure why. Maybe it's the first conditions I use for X, but it should only work for any affordable IC. Any thoughts would be appreciated! I have three partial differential equations (PDEs) and an analytical solution for a variable as shown. Using these formulas I will solve for \phi(x,y,t), p(x,y,t), C_{a}(x,y,t) and C_{b}(x,y,t) that is, in terms of space and time. I know there is a function pdepe( ) in Matlab to solve initial limit value problems for parabolic-elliptical PDEs in 1D. I would like to know how this feature or another in Matlab can be used to solve the problem described below which is 2D and written. PROBLEM: The following two equations represent PDEs for two species a and b, respectively: Where D_{h} and q are given as: Here are R_{a}=R_{b}=R, where R is given as: Finally, the last equation is given as: INTRODUCTORY AND LIMIT CONDITIONS: Total domain size is 10 cm x 5 cm and the width of the y-shaped subdomain is 0.5 cm. This subdomain has an initial at 0.50 while in the surrounding surrounding \phi= 0.26. Constant p of 1 Pa and 0 Pa respectively are maintained at 1 and (2, respectively), corresponding to a gradient of approx. 10^-3 m^-1. P on borders (3) and (4) are determined by linear gradients between boundaries (1) and (2). Constant C of C_{a} = 2 mol m^-3 and C_{b} = 0.2302 mol m^-3 is maintained at the limit (3), while concentrations at the limit (4) are set to C_{a} = 1 mol m^-3 and C_{b} = 0.4603 mol m^-3. Concentrations at the limit (1) are determined by constant gradients between borders (3) and (4), while an advective flux limit state of $$(\frac{\partial C}{\partial x} = 0)$$ is set at the expiration date of (2). I want to solve written partial differential equations of the first order, which are of rigid nature. I have encoded in MATLAB to solve this pde's, I have used Method of Line to Convert PDE to ODE, and I have used beam and heating (second order headwind) method to discriminate spatial derivative. The discrete method is total variation-degrading (TVD) to eliminate the oscillation. But rather using TVD and ode15s solver to integrate resulting rigid ode is the resulting plot oscillatory (not smooth). What should I do to eliminate this fluctuation and get correct results. I have attached my MATLAB code.. please see it and suggest some improvement. ∂y(1)/∂t=-0.1 ∂y(1)/∂x + (0.5*e^(15*(y(2)/(1+y(2))))*(1- y(1)) ∂y(2)/∂t=-0.1 ∂y(2)/∂x - (0.4*e^(15*(y(2)/(1+y(2))))*(1- y(1))-0.4 Initial condition: at t = 0 y(1)= y(2)=0 Boundary condition: y(1)= y(2) = 0 at x=0 I have attached my MATLAB code.. please see it and suggest some improvement. function brussode(N) if nargin<1 N = 149; end tspan = [0 10]; m = 0.00035 t = (1:N)/(N+1)*m; y0 = [repmat(0,1,N); repmat(0,1,N)]; p = 0.5 q = 0.4 options = odeset('Vectorized','on','JPattern',jpattern(N)); [t,y] = ode15s (@f, tspan, y0, options); a = size (y,2) u = y (:,1:2:end); x = (1:N)/(N+1); figure; %surf(x,t(end,:),u); plot(x,u(end,:)) xlabel('space'); ylabel('solution'); zlabel('solution u'); %---------- %Nested function -- N is provided by the outer function. % function dydt = f(t,y) %Derivative function dydt = zeros(2*N,size(y,2)); %preallocate dy/dt x = (1:N)/(N+1); % Evaluate the 2 components of the function at one edge of the grid % (with edge conditions). i = 1; %y(1,:) = 0; %y(2,:) = 0; dydt(i,:) = -0.1*(N+1)*(y(i+2,:)-0)+ (0.01/2)*m*((N+1).^3)*(y(i+2,:)-0) + p*exp(15*(0/(1+0)))*(1-0); dydt(i+1,:) = -0.1*(N+1)*(y(i+3,:)-0)+ (0.01/2)*m*((N+1).^3)*(y(i+3,:)-0) - q*exp(15*(0/(1+0)))*(1-0)+0.25; i = 3; %y(1 ,:) = 0; %y(2.:) = 0; virtue(i,:) = -0.1*(N+1)*(i+2,:)-y(i,:)) + (0.01/2)*m*((N+1).^3)*(0.01 /2)*m*((N+1).^3)*(3)*(0.01/2)*m*((N+1).^3)*(0.01/2)*m*((N+1).^3)*(0.0.01/2)*m*((N+1).^3)*(0.01/2)*m 01/2)*m*((N+1).^3)*(0.01/2)*m*((N+1).^3)*(0.01/2)*m*((N+1).^3)(i+3,:)1-y(i,:)) + p*exp(15*(i+1,:)/(1+y(i+1,:)))))*(1-y(i,:)); colored(i+1,:) = + (0,01/2)*m*((N+1).~3)*(y(y(0.01/2)*m*((N+1).~3)*(y(0.01/2)*m*((N+1).~3)*(y(0.01/2)*m*((N+1).~3)*(y(y(0.01/2)*m*((N+1).~3)*(y(0.01/2)*m*(N+). 1).~3)*(y(0.01/2)*i+3,:)-y(i,:)) - q*exp(15*(y(i+1,:)/(1+y(i+1,:))))*(1-y(i,:))+0.25; q*exp(15*(y(i+1,:)/(1+y(i+1,:))))*(1-y(i,:))+0.25; the two components of the function at all internal grid %points. i = 5:2:2*N; %y(1,:) = 0; % y(2,:) = 0; dydt(i,:) = (-0.1/2)*(N+1)*(3*y(i,:)-4*y(i-2,:)+y(i-4,:)) +(0.01/2)*m*((N+1).^3)*(y(i,:)-2*y(i-2,:)+y(i-4,:))+ p*exp(15*(y(i+1,:)/(1+y(i+1,:))))*(1-y(i,:)); dydt(i+1,:) = (-0.1/2)*(N+1)*(3*y(i+1,:)-4*y(i-1,:)+y(i-3,:))+(0.01/2)*m*((N+1).^3)*(y(i+1,:)-2*y(i-1,:)+y(i-3,:)) - q*exp(15*(y(i+1,:)/(1+y(i+1,:))))*(1-y(i,:))+0.25; end %---------- end %brussode %---------- % Subfunction -- the sparsity pattern % function S = jpattern(N) % Jacobian sparsity patter B = ones(2*N,5); B(2:2:2*N,2) = zeros(N,1); B(1:2:2*N-1.4) = zeros(N,1); S = spdiags (B,-2:2,2* N, 2 * N); end %---------- ---------- For the problem, I have two differential equations. These are: dx / dt = -xp + yq + (1-x-y) qdy / dt = -yq + (1-x-y) uFor my problem I have a given set of historical data and my goal is to find the best values for p, q and you to give the best fit to this historical data. I have a column of historical data associated with each formula above. I also need to find the best starting conditions for each equation. So far I have tried to use For Loops for each of p, q& u to find the best values compared to my data. My problem is that I'm not so sure how to code this correctly. If anyone could help me make a start that would be great. I understand the general idea, but am unsure how to perform this in Matlab In a partial differential equation (PDE), the function that is solved depends on several variables, and the differential equation can include partial derivatives taken with respect to each of the variables. Partial differential equations are useful for modeling waves, heat flow, fluid dispersing and other phenomena of spatial behavior that change over time. MATLAB® PDE solver pdepe solves initial limit value problems for systems of PDEs in a spatial variable x and time t. You can think of these as ODEs of a variable that also change with respect to time.pdepe using an informal classification for the 1D equations it solves: Equations with a time derivative are parabolic. An example is the heat equation ∂u∂t=∂2u∂x2. Equations without time derivative are elliptical. An example is the Laplace equation ∂2u∂x2 = 0. pdepe requires at least one parabolic equation in the system. In other words, at least one equation in the system must include a time derivative.pdepe also solves certain 2D and 3D problems that reduce to 1D problems due to angular symmetry (see argument description for symmetry constant m for more information). Partial differential formula toolbox™ extends this functionality to generalized issues in 2D and 3D with Dirichlet and Neumann border conditions. A 1D-PDE u(x,t) which depends on time t and a spatial variable x. MATLAB PDE solver pdepe solves systems with 1D parabolic and elliptical PDEs in the form The equation has the properties: the PDEs hold for t0 ≤ t ≤ tf and a ≤ x ≤ b.Spatial interval [a, b] must be limited.m can be 0, 1 or 2, respectively, corresponding slice, cylindrical or spherical metric. If m > 0, ≥ 0 must also hold. The coefficient f(x,t,u,∂u∂x) is a fluxbetermin and s(x,t,u,∂u∂x) is a source limit. The flux beetmine must depend on the partially derived ∂u/∂x. The coupling of the partial derivatives with respect to time is limited to the multiplication of a diagonal matrix c (x, t, u, ∂u∂x). The diagonal elements of this matrix are either zero or positive. An element that is zero corresponds to an elliptical equation, and any other element corresponds to a parabolic equation. There must be at least one parabolic equation. An element of c corresponding to a parabolic equation can disappear by isolated values of x if they are mesh points (points where the solution is evaluated). Discontinuities in c and s due to material interfaces are allowed provided that a mask point is located on each interface. To solve PDEs with pdepe, you need to define the equation coefficients for c, f and s, the first conditions, the behavior of the solution on the boundaries, and a web of points to evaluate the solution. The function calls sun = pdepe (m, pdefun, icfun, bcfun, xmesh, tspan) uses this information to calculate a solution on the specified net: m is the symmetry constant.pdefun defines the equations that are solved.icfun defines the first conditions.bcfun defines the limit conditions.xmesh is a vector of spatial values for x.tspan is a vector of time values for t. Together, the xmesh and tspan vectors form a 2D grid on which pdepe evaluates the solution. You must express the PDEs in the default form expected by pdepe. Written in this form, you can read the values of the coefficients c, f, and s.In MATLAB you can encode the equations with a function of the form function [c, f,s] = pdefun (x,t,u,dudx) c = 1; f = dudx; s = 0; endIn this case, pdefun defines ∂u∂t=∂2u∂x2. If there are multiple equations, c, f, and s are vectors with each element corresponding to one formula. At the first time t = t0, for all x, the solution components satisfy the original conditions of the form In MATLAB you can encode the original conditions with a function of the form function u0 = icfun (x) u0 = 1; endIn this case u0 = 1 defines an initial condition for u0 (x,t0) = 1. If there are multiple equations, u0 is a vector with each element that defines the original condition of a formula. At the limit x = a or x = b, for all t, the solution components satisfy the limit conditions in the form q (x,t) is a diagonal array of elements that are either zero or never zero. Please note that boundary ratio is expressed in the form of flux f, rather than the partial derivative of you with respect to x. Of the two coefficients p(x,t,u) and q(x,t), only p can depend on u.In MATLAB, you can also encode the boundary conditions with a function of the shape function [pL, qL, pR, qR] = bcfun (xL, uL, xR, uR, t) pL = uL; qL = 0; pR = uR - 1; qR = 0; end pL and qL are the coefficients of the left boundary, while pR and qR are the coefficients of the right boundary. In this case, bcfun defines the limit conditionsIf there are multiple equations, the output is pL, qL, pR, and qR vectors with each element that defines the boundary constraint for one formula. The default integration properties in the MATLAB PDE solver are selected to address common issues. In some cases, you can improve the problem solver's performance by overriding these default values. To do this, use the odeset to create an option structure. Then pass the structure of the pdepe as the last input argument:sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,options) Of the options for the underlying ODE solver ode15s, only those shown in the table below are available for pdepe. After you resolve an equation with pdepe, MATLAB returns the solution as a 3D arraysol, where the sun(i,j,k) contains the kth component of the solution evaluated by t(i) and x(j). In general, you can extract the kth solution component with the command u = sun(:,:,k). The time network you specify is used exclusively for output purposes, and does not affect the internal time steps that the solver takes. However, the spatial web you specify can affect the quality and speed of the solution. After solving an equation, you can use pdeval to evaluate the solution structure returned by pdepe with another spatial mesh. An example of a parabolic PDE is the heat equation in one dimension: This equation describes the dispersion of heat for 0≤x≤L and t≥0. The goal is to solve for the temperature u (x, t). The temperature is basically a nonzero constant, so the original state is Also the temperature is zero on the left limit, and nonzero at the right limit, so the limit conditions are To solve this equation in MATLAB, you need to encode the equation, initial conditions and boundary conditions, and then select an appropriate solution network before calling the solver pdepe. You can either include the required functions as local functions at the end of a file (as in this example), or save them as separate named files in a directory on the MATLAB path. Code EquationBefore you can encode the equation, make sure that it is in the form that the pdepe solver expects: c(x,t,u,∂u∂x)∂u∂t= x-m∂∂x (xmf(x,t,u,∂u∂x))+s(x,t,u,∂u∂x). In this form, the heat equation is So the values of the coefficients are as follows: The value of m is sent as an argument to pdepe, while the other coefficients are encoded in a function of the equation, ice function [c,f,s] = heatpde (x,t,u,dudx) c = 1; f = dudx; s = 0; end (Note: All features are included as local functions at the end of the example.) Code Initial conditionThe first condition function for the heat equation allocates a constant value for u0. This function must accept an input for x, even if it is unused.function u0 = heatic(x) u0 = 0.5; final code limit conditionsThe standard for the limit conditions expected by the pdepe solver is p(x,t,u)+q(x,t)f(x,t,u,∂u∂x)=0.Written in this form, the limit conditions for this problem are so the values for p and q arepL = uL, qL = 0.pR = uR-1, qR = 0.The corresponding function is dafunction [pl, ql, pr, qr] = heatbc(xl, ul, xr, your, t) pl = ul; ql = 0; pr = your - 1; qr = 0; end Select Solution NetworkUse a spatial network with 20 points and a 30-point time network. Since the solution quickly reaches a smooth state, the times close to t = 0 are more tightly distributed together to capture this behavior in the output. L = 1; x = linspace(0,L,20); t = [linspace(0,0.05,20), linspace(0.5,5,10)]; Solve EquationFinally, solve the equation using symmetry m, PDE equation, the original state, the limit conditions, and the nets for x and t.m = 0; sun = pdepe(m,@heatpde,@heatic,@heatbc,x,t); Plot SolutionUse imagesc to visualize the solution matrix. colormap hot imagesc(x,t,sol) colorbar xlabel('Distance x','interpret','latex') ylabel('Time t','interpret','latex') title('Heating for $0 \le x \le 1$ and $0 \le t \le 5$','interpret','latex')Local functions function [c,f,s] = heatpde(x,t,u,dudx) c = 1; f = dudx; s = 0; end function u0 = hematically(x) u0 = 0.5; end function [pl, ql, pr, qr] = heatbc (xl, ul, xr, your, t) pl = ul; ql = 0; pr = your - 1; qr = 0; endSeveral available sample files serve as good starting point for the most common 1D PDE problems. To explore and run examples, use the Differential Formulas app examples. To run this app, type To open a single file for editing, typeTo run an example, this table contains a list of the available sample PDE files. [1] Skeel, R. D. and M. Berzins, A Method of Spatial Subtlety of Parabolic Equations in a Space Variable, SIAM Journal on Scientific and Statistical Computing, Vol. 11, 1990, p. 1–32. See Alsobvp4c | ode45 | odeset | 10.00:00 pdeval related topicsSolve Single PDESolve System of PDEs PDEs