Model Reduction

Lab-2

Emperical Interpolation for non-linear problems

By

Sreekanth Reddy

BAKKA CHENNAIAH GARI

Under the guidance of

Kiran Sagar KOLLEPARA

Post-Doctorant, GeM

Faculty of Engineering

Ecole Centrale de Nantes

October, 2023

CONTENTS

OBJECTIVE

To solve a non-linear 2D transient heat diffusion problem by using the Empirical Interpolation Method (EIM).

INTRODUCTION

In the lab-1, The Proper Orthogonal Technique is demonstrated.

POD is often used when dealing with high-dimensional data or complex mathematical models that are computationally expensive to solve. It helps in reducing the dimensionality of the data or the model while preserving the essential information.

Data Collection: POD begins with collecting snapshots of the system's behavior. These snapshots can be obtained from simulations, experiments, or measurements.

Snapshot Matrix: The collected snapshots are arranged in a matrix called the snapshot matrix, where each column represents a snapshot in time or space, and each row corresponds to a particular spatial or temporal point.

Singular Value Decomposition (SVD): The next step is to perform Singular Value Decomposition on the snapshot matrix. SVD factorizes the snapshot matrix into three matrices: U, Σ (Sigma), and V. U contains the spatial modes, Σ contains singular values (representing the importance of each mode), and V contains the temporal modes.

Mode Selection: To reduce the dimensionality of the system, you can select a subset of the modes (spatial and temporal) that capture the most significant variation in the data. Typically, modes with the largest singular values are retained, while the rest are truncated.

Reduced-Order Model: With the selected modes, a reduced-order model (ROM) can be constructed. This reduced model captures the essential dynamics of the system using fewer variables, making it computationally more efficient while maintaining reasonable accuracy.

The problem encountered is linear, which is independent of the dependent variable. In the current lab, A non-linear model is solved using POD and and EIM methods.

THEORY

The governing equation:

$$\rho \cdot Cp \cdot \frac{\partial u}{\partial t} = \nabla \cdot (k(u)\nabla u) + s(x, y, t) \tag{1}$$

Now, Convert the strong form to weak form to allow for the inclusion of boundary conditions, to simplify the treatment of derivatives, and offer flexibility in choosing elements and integration schemes. Multiply with the test function v and integrate over the domain:

$$\int_\Omega \left(\rho \cdot Cp \cdot \frac{\partial u}{\partial t} \cdot v \right) d\Omega = \int_\Omega (\nabla \cdot (k(u)\nabla u)) \cdot v \, d\Omega + \int_\Omega s(x, y, t) \cdot v \, d\Omega \tag{2}$$

By fixed point iteration for the treatment of the non-linear term, start by selecting an initial guess for the solution. This guess should be reasonably close to the actual solution for convergence to occur. The closest solution is the value from the previous iteration $k(u_n) = k(u_{n-1})$.

Integrating by parts of the bilinear term:

$$\int_\Omega (\nabla \cdot (k(u)\nabla u)) \cdot v \, d\Omega = -\int_\Omega ((k(u)\nabla u) \cdot (\nabla v)) \, d\Omega + [k(u)\nabla u \cdot v]_{\partial\Omega_{\text{Neumann}}} \tag{3}$$

Substituting this back into the equation:

$$\rho \cdot Cp \cdot \left(\int_\Omega v \, d\Omega \right) \cdot \frac{\partial u}{\partial t} = -\int_\Omega ((k(u)\nabla u) \cdot (\nabla v)) \, d\Omega + [k(u)\nabla u \cdot v]_{\partial\Omega_{\text{Neumann}}} + \int_\Omega s(x, y, t) \cdot v \, d\Omega \tag{4}$$

The domain of interest is 2D.

By applying finite differences: (Euler Implicit Scheme)

$$\frac{\partial u}{\partial t} = \frac{u_n - u_{n-1}}{\Delta t} \tag{5}$$

Let's simplify the equation:

$$\rho \cdot Cp \cdot \left(\int_{\Omega} v \, d\Omega \right) = C \tag{6}$$

$$\left(\int_{\Omega} v \, d\Omega \right) = M \tag{7}$$

$$\int_{\Omega} ((k(u)\nabla u) \cdot (\nabla v)) = K^{l-1} \cdot u_n^l \tag{8}$$

By substituting the simplified equation:

$$(C + \Delta t \cdot K^{l-1}) \cdot u_n^l = C \cdot u_{n-1}^l + (M \cdot s_n + q_n) \cdot \Delta t \tag{9}$$

where superscript l indicates the non-linear iteration and subscript n indicates the step in the time integration scheme.

Where, ρ is the mass density, Cp is the specific heat, and $k(u) := k_0 \left(1 + 0.5 \left((u+1)^5 - 1\right)\right)$ is a non-linear material diffusivity.

On the other hand, the heat flux is defined by $q(z,t) := \sin\left(\frac{2\pi}{T} \cdot t\right)$ on the surface of the hole for $0 \leq t \leq \frac{T}{2}$.

The source is defined by $s(x,t) = 3 \cdot \sin\left(\frac{2\pi}{T} \cdot \left(t - \frac{T}{2}\right)\right)$ for every x in the domain.

The provided code demonstrate the finite element formulation of the 2D transient heat diffusion equation, The diffusivity co-efficient in the equation is non-linear which is depending on the temperature, the dependent variable. The fixed-point iteration technique is used for the treatment of the non-linearity. The code is framed in such a way that the FE solution automatically saves to the folder **"FEM_Solution"** which we consider as a true solution to build the reduced basis.

```
time = 1
timeSpanTrain = 0.2
timeSpan = time.max()
maskTrain = (time<timeSpanTrain)
timeTrain = time[maskTrain]
solution = solution[np.ix_(maskDir,maskTrain)]
```

Reduced Basis of the FE solution: From the true solution (FEM_Solution), we consider the solution till the 0.2 seconds to extract the significant modes which captures the behaviour of our interest. By performing the Singular Value Decomposition technique, computing the weights and based on the threshold value, extract the modes. The reduced basis for the solution is stored in an array called **"reduced_basis"**.

```
[leftSingularVectors, singularValues, _] =
np.linalg.svd(solution, full_matrices=False)
maskTruncation = np.zeros(leftSingularVectors.shape[1], dtype=bool)
weights = 1 - singularValues.cumsum()/singularValues.sum()
maskTruncation[weights>=threshold] = True
reducedBasis = np.zeros([maskDir.size, sum(maskTruncation)])
reducedBasis[maskDir,:] = leftSingularVectors[:,maskTruncation]
```

The diffusivity coefficient is a function of the dependent variable. Now that the FE solution is extracted for time [0, 0.2], we find how K is changing with respect to the solution.

```
condfield = condfun(solution[:,maskTrain])
```

Now perform the Singular Value Decomposition and filter the significant modes based on the threshold value. We use weights of the singular values to compare with the threshold value.

```
[leftSingularVectors,singularValues,_] =
np.linalg.svd(condfield, full_matrices=False)
maskTruncation = np.zeros(leftSingularVectors.shape[1], dtype=bool)
weights_c = 1 - singularValues.cumsum()/singularValues.sum()
maskTruncation[weights_c>=threshold] = True
reducedBasis_c = leftSingularVectors[:,maskTruncation]
```

The reduced non-linear basis is stored in **"reduced_basis_c"**.

Now that we have the reduced basis for the solution and non-linear diffusivity.

We approximate the solution u as $u = B\alpha$.

The k which is a function of u is approximated as $k_l = \sum_{k=1}^{d} V^k \Psi_k$.

Substituting the above statements in the weak form equation:

$$(C + \sum_{k=1}^{d} \Delta t V^k \Psi) B\alpha_n^l = CB\alpha_{n-1}^l + (Ms_n + q_n)\Delta t \tag{10}$$

Now we project the equation onto the reduced basis of the solution:

$$B^T(C + \sum_{k=1}^{d} \Delta t V^k \Psi) B\alpha_n^l = B^T CB\alpha_{n-1}^l + B^T(Ms_n + q_n)\Delta t \tag{11}$$

Where:

$$\begin{aligned} B^T CB &= \rho C_p M_{\text{reduced}} \\ B^T V^k B &= K_{\text{reduced}} \\ M &= \text{opMass} \\ Ms_n &= \text{rhs}_{\text{source}} \\ q_n &= \text{rhs}_{\text{neumann}} \\ B^T \text{rhs}_{\text{neumann}} + \text{rhs}_{\text{source}} &= \text{rhs}_{\text{total}} \end{aligned}$$

The right-hand side (RHS) is computed as:

$$RHS = \Delta t(\text{rhs}_{\text{total}}) + (\rho C_p M_r u_{l-1}) \tag{12}$$

Now let us look into the left-hand side (LHS):

Let's say LHS = A (initialization):

$$A = \rho C_p M_{\text{red}} \tag{13}$$

The leftover term is:

$$\sum_{k=1}^{d} \Psi \Delta t(K_{\text{red}}) \tag{14}$$

How to compute Ψ:

We use Empirical Interpolation Method to find Ψ:

$$k_l = \sum_{k=1}^{d} V^k \Psi_k \tag{15}$$

$$P^T K_l = P^T V \Psi_l \tag{16}$$

$$\Psi_l = (P^T V)^{-1} (P^T K_l) \tag{17}$$

Where P is the extractor or pivot matrix, and $(P^T K_l) = f(P^T B \alpha_l)$.

So we are evaluating the non-linear term at only $V.column\,size$ points.

ALGORITHM: EIM (EMPIRICAL INTERPOLATION METHOD)

Input: A matrix V with columns of non-linear reduced basis.

Output: An array intPoints containing selected indices of basis functions

1. Initialize intPoints as an array of size V.columns.
2. Set intPoints[0] to the index of the row where the first column of V has the maximum absolute value.
3. For i from 1 to the number of columns in V (V.shape[1]):
   a. Solve a linear system to compute coefficients c: c = solve(V[intPoints[0:i], 0:i], V[intPoints[0:i], i])
   b. Compute the residual r: r = V[:, i] - V[:, 0:i] @ c
   c. Find the index of the row with the maximum absolute value of the residual: indNew = argmax(abs(r))
   d. Set intPoints[i] to indNew.
4. Return intPoints as the array of selected indices.

V**:** A matrix with columns of non-linear reduced basis.

```
def eim(basis):
'''compute EIM points for the given basis'''
intPoints = -np.ones(basis.shape[1], dtype=int)
intPoints[0] = abs(basis[:, 0]).argmax()
for i in range(1, basis.shape[1]):
    c = np.linalg.solve(basis[intPoints[0:i], 0:i], basis[intPoints[0:i], i])
    r = basis[:,i] - basis[:,0:i] @ c
    indNew = abs(r).argmax()
    intPoints[i] = indNew
return intPoints
```

Description:

Step 1: In the first column of the basis, find the node where the value is maximum using `argmax`, and add the index of the row to `intPoints`. `intPoints` is an array of size V.columns.

Step 2: Iteratively, solve a linear system of equations to compute coefficients c. This is done by solving a linear system formed by the previously selected EIM points and the current basis function. Compute the residual r for the current basis function. The residual is the difference between the current basis function and the projection of that function onto the subspace spanned by the previously selected EIM points using the coefficients c.

For example, let's take $i = 4$ and `intPoints` $= [12, 158, 280, 54]$. Now we are at $V[:, 4]$. Find c_0 such that $V[:, 4] - V[:, 0] \cdot c_0 = 0$ at $V[12, 4]$. Find c_1 such that $V[:, 4] - V[:, 1] \cdot c_0 = 0$ at $V[158, 4]$. Find c_2 such that $V[:, 4] - V[:, 2] \cdot c_0 = 0$ at $V[280, 4]$. Find c_3 such that $V[:, 4] - V[:, 3] \cdot c_0 = 0$ at $V[54, 4]$.

Now in $V[:, 4]$, find the `argmax`, and the index of the row, which has the maximum residual, is added to `intPoints`.

This loop continues until the last column of V.

Now that we have interpolation points (intPoints), extract the rows of index intPoints from the reduced basis of solution and non-linear term:

$$\text{Int}_t = \text{reducedBasis}[\text{intPoints}, :] \tag{18}$$

$$\text{Int}_c = \text{reducedBasis}_c[\text{intPoints}, :] \tag{19}$$

We now substitute the above into the reduced non-linear equation:

$$K_l = V \Psi_l \tag{20}$$

where, Int_t is representing the K_l Int_c is representing the V

Now,

```
T_eim = Int_t @ solution[:,i-1]
cond_eim = condfun(T_eim)
beta_cond = Int_c.T @ cond_eim
```

$$\Psi = \text{solve}(\text{Int}_c, \text{Int}_t) \tag{21}$$

Ψ is a matrix with columns of coefficients.

```
for i in range(1,numOfFrames):
    rhs = density*specificHeat*M_r @ solution[:,i-1] +
                timeInc*rhsTotalFree[:,i]
    A = density*specificHeat*M_r
    for j in range(len(K_r)):
        A += timeInc* beta_cond[j] * K_r[j]
    solution[:,i] = np.linalg.solve(A,rhs)
```

$$\left[(\rho \cdot C_p \cdot M_{\text{red}}) + \sum_{k=1}^{d} \Psi \cdot \Delta t \cdot (K_{\text{red}})\right] \cdot \alpha_n^l = \left[\Delta t \cdot (rhs_{\text{total}}) + (\rho \cdot C_p \cdot M_r \cdot \alpha_{n-1}^l\right] \tag{22}$$

Now that we know Ψ, we can compute α_n^l by initializing $\alpha_{n-1}^0 = 0$.

The outer loop is w.r.t time and the inner loop is due to non-linearity in diffusivity constant.

Non-linear POD using EIM

1. **Data Collection:** Begin by collecting snapshots of the non-linear system's behavior. These snapshots can be obtained from numerical simulations, experiments, or measurements.

2. **Snapshot Matrix:** Arrange the collected snapshots in a matrix called the snapshot matrix, where each column represents a snapshot in time, and each row corresponds to a particular spatial point (node).

3. **Singular Value Decomposition (SVD):** Perform Singular Value Decomposition on the reduced basis. SVD factorizes the reduced basis matrix into three matrices: U, (Sigma), and V. U contains the spatial modes, contains singular values (representing the importance of each mode), and V contains the temporal modes.

4. **Mode Selection:** To further reduce dimensionality, select a subset of the modes (spatial and temporal) based on the importance indicated by the singular values. Typically, modes with the largest singular values are retained by computing weights and comparing to the threshold, while the rest are truncated.

5. **Reduced Basis:** The Reduced basis of the solution is computed while the non-linear term depends on the solution, Compute the values of the non-linear term using the solution found from the snapshot matrix, perform the SVD and filter the significant modes, leading to the reduced basis for a non-linear term.

6. **EIM Points Selection:** Use the EIM algorithm to select a set of interpolation points (*intPoints*) from the snapshot matrix. The EIM algorithm identifies the most influential snapshots for the non-linear behavior.

7. **Reduced-Order Model (ROM):** Construct a reduced-order model (ROM) using the selected modes. This ROM captures the essential non-linear dynamics of the system using fewer variables, making it computationally efficient while maintaining reasonable accuracy.

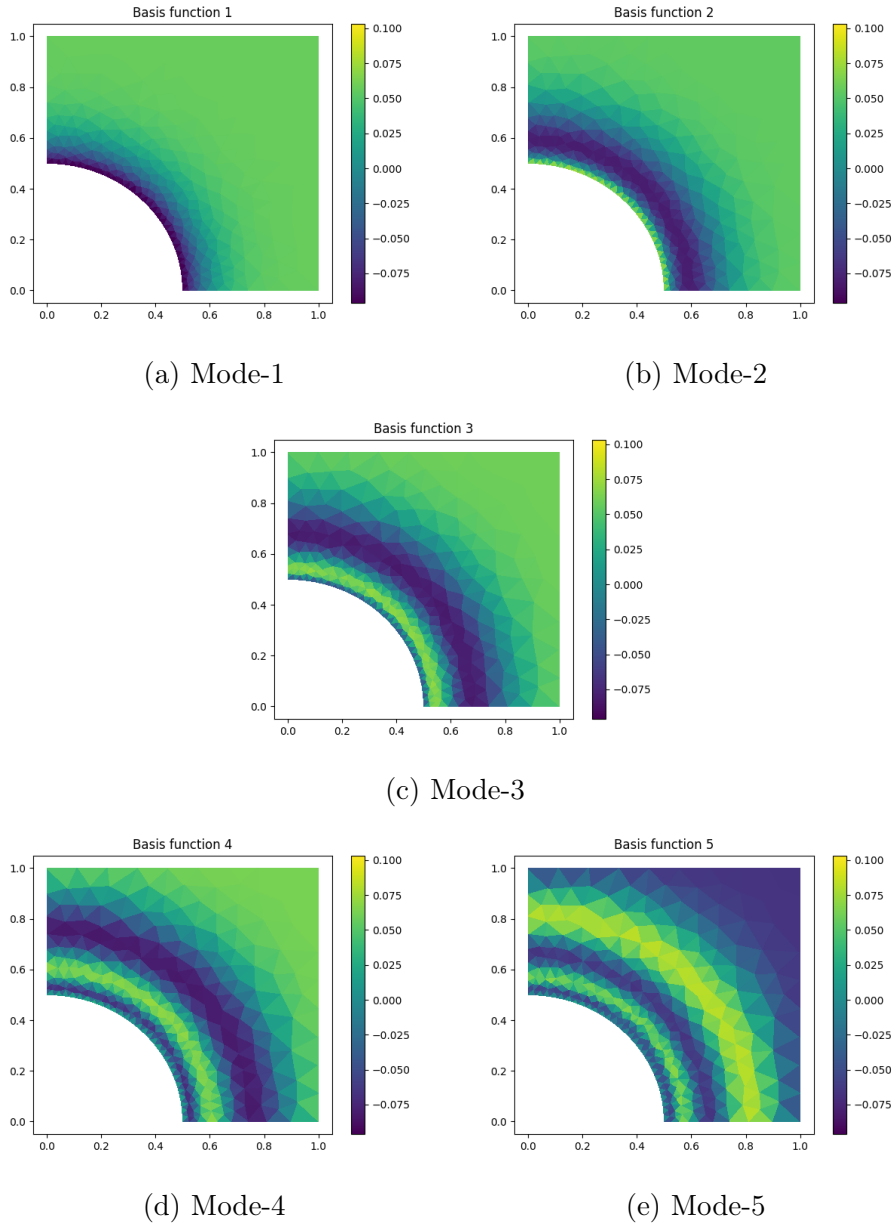
8. **Non-linear Term Approximation:** Approximate the non-linear term in the reduced model using the EIM-selected interpolation points and their corresponding coefficients.

9. **Solve Reduced Model:** Solve the reduced-order model with the approximated non-linear term to obtain the time evolution of the system.

10. **Analysis and Validation:** Analyze the reduced model's accuracy and validate it against the full non-linear system to ensure that it captures the essential features of interest.

RESULTS AND DISCUSSIONS

Exercise-1:



(a) Mode-1 (b) Mode-2

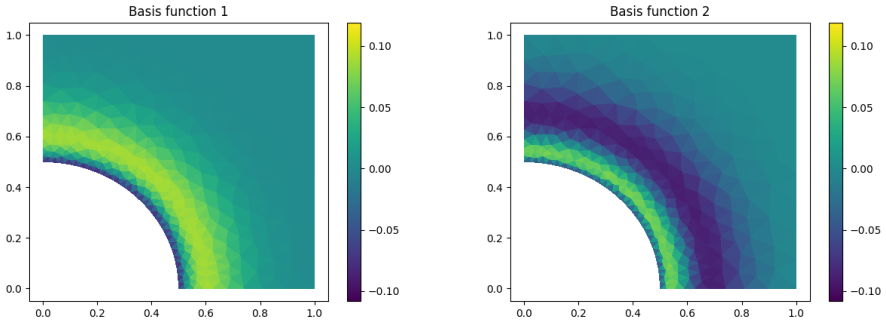
(c) Mode-3

(d) Mode-4 (e) Mode-5

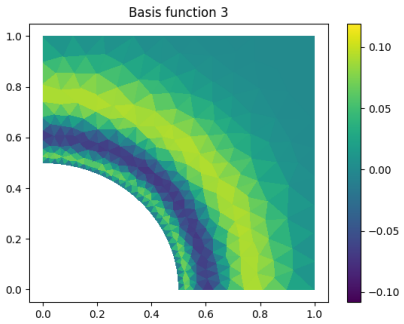
Figure 1: Reduced Non-linear modes

The non-linear term in many physical systems can be high-dimensional and computationally expensive to simulate. The reduced modes represent a more compact representation of this high-dimensional non-linear behavior. They indicate that you can describe the non-linear effects in the system using a smaller number of variables. The selected reduced modes typically capture the most dominant and important non-linear features

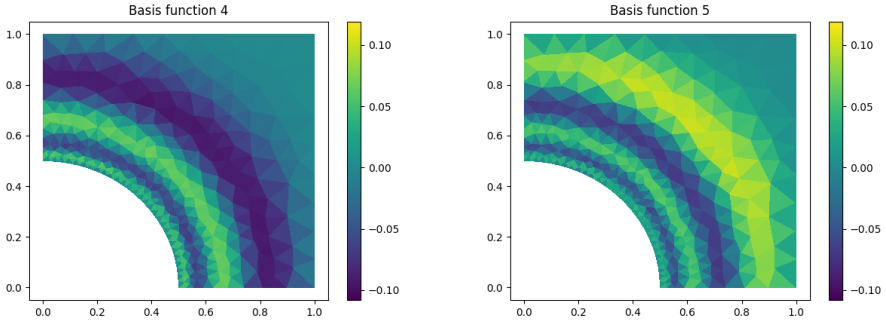
of the system. They indicate which aspects of the non-linearity are essential for accurate modeling and which can be safely neglected or approximated. the reduced modes of the non-linear term indicate a trade-off between computational efficiency and accuracy. They are a way to approximate and represent the non-linear behavior of a high-dimensional system using a smaller set of basis functions, which is crucial for solving complex engineering and scientific problems efficiently.



(a) Mode-1 (b) Mode-2



(c) Mode-3



(d) Mode-4 (e) Mode-5

Figure 2: Reduced Solution modes

The selected reduced modes typically capture the most dominant spatial patterns in the solution. Working with reduced modes enables computationally efficient simulations.

Instead of solving the governing equations for the full high-dimensional solution field, you can use the reduced modes to approximate the solution. This is especially valuable for real-time simulations.

Exercise-2: The spectrum of the FEM solution is full; FEM captures all the spatial and temporal variations of the problem. The condition number of FEM matrices can be large, especially in ill-conditioned problems. A high condition number can indicate that the system is sensitive to small changes in input data, potentially leading to numerical instability. In FEM simulations, the convergence and accuracy of the numerical solution are influenced by the spectral properties of the matrices. If the matrix is well-conditioned, this should lead to accurate convergence.

In the Reduced Order Model, we consider only the dominant eigenvalues based on a threshold value. POD or EIM reduces the spectrum of the system. We are eliminating the less significant modes, leading to the matrix being well-conditioned, thereby leading to accurate convergence.

The accuracy of the solution depends on the spectrum that we consider. The spectrum of a matrix is the set of all eigenvalues of that matrix.

Exercise-3:

For a time span of 0.2 seconds, the plot of Error vs. Time is shown below:

The code snippet below calculates the pointwise absolute error between the two sets of solutions and stores it in an array "variable":

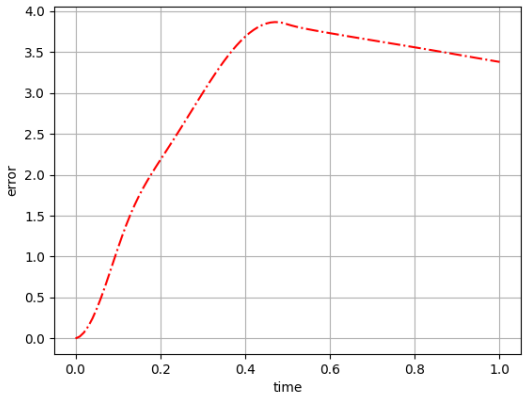


Figure 3: Error vs Time

```
variable = abs(fem_solution - rom_solution)
error = np.linalg.norm(variable, axis=0)
```

It calculates the 2-norm of the error at each time step by taking the square root of the sum of the squares of its elements along the columns.

Comment on Execution Time:

In the FE Solution, we are iterating with respect to time and non-linearity. The fixed iterative scheme is implemented for the convergence of the non-linearity. While in the reduced system, the only task during the Offline stage is where we come up with a reduced model, which is cost-efficient for complex problems. During the online stage, we get the solutions in real-time. In the case of non-linearity, we do iterate for the non-linearity but in the non-linear reduced basis, which captures the significant modes in non-linearity. This makes the analysis easier. During the online stage, the reduced basis of the solution and reduced basis of non-linearity, which have very few modes compared to the FE solution but are very significant, capture the important modes and give us the solution in real-time.

The execution time of the FE Solution is 141.4249 seconds, while the Reduced Model Solution is 0.0341 seconds with a time span of 0.2 seconds. The execution time with respect to the time span can be visualized in the figure below:

Comment on Varying Time Span:

The time taken for the execution is shown in the figure. We can observe that the

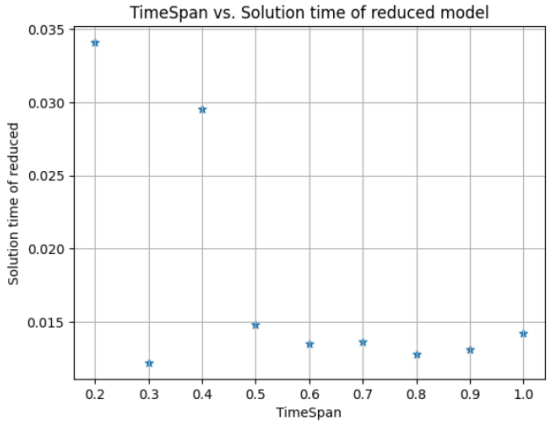


Figure 4: Execution Time vs Time Span

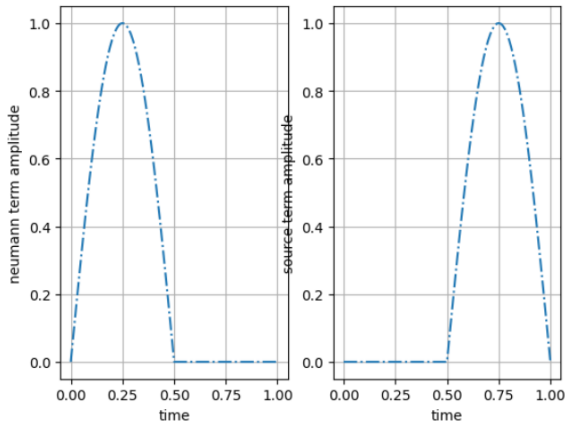


Figure 5: Flux vs Time and Source vs Time

time taken decreases with an increase in time span. As we increase the time span, the number of significant modes below the threshold increases, thereby increasing the size of the reduced matrix for solution and non-linearity. However, the time taken decreases because, by capturing significant modes in non-linearity, the iterations to converge are very few, leading to less time in execution.

The Error vs. Time at a time span of 0.2 seconds is shown in the figure. 3. The error plots do not vary significantly by changing the time span. The error is less in the time span zone compared to the predicted time zone. The maximum error of approximately magnitude 4 is seen at the predicted time of 0.5 seconds. The error is mainly due to the discontinuity in the boundary conditions, where the heat flux is applied until 0.5 seconds and source from 0.5 seconds. Due to the change in Neumann boundary conditions and source term with respect to time and due to non-linearity, the error prevails at any time span.

CONCLUSION:

1. The significance of the spectral properties are discussed, and it is evident that the spectral properties controls the accuracy and convergence.

2. Witnessed how significantly the execution time varies from FE solution but at the cost of accuracy as we consider only the significant modes and truncate the other modes.

3. For reducing the order of the non-linear problem, Emperical Interpolation Method is implemented, where we evaluate the non-linear term only at dim(V) points. The non-linear reduced basis captures only the important modes effecting the non-linearity at the cost of accuracy but time efficient, which can be visualized from the explanations provided.

4. In summary, the reduced modes of the non-linear term indicate a trade-off between computational efficiency and accuracy. To represent the non-linear behavior of a high-dimensional system using a smaller set of basis functions, which is crucial for solving complex engineering and scientific problems efficiently.