Appendix A  Description of Extended Simulations and Table of All Simulation Results

I conducted several fluid and heat transport simulations for Parkfield considering additional permeability cases for the Franciscan Assemblage and Great Valley Sequence (Figure A1 and Table A1), in addition to those described in Figure 5 and Table 2 in the main text of this thesis. The purpose of these simulations was to evaluate the effects of advection on simulated surface heat flow for a greater number of realistic permeabilities, which lie between the upper and lower bounds described in the main text. Because simulated surface heat flow in the Coast Ranges section of the Parkfield model is not significantly different for the lowest and highest permeabilities assigned to the Salinian block, Franciscan Assemblage, and Great Valley Sequence, simulated surface heat flow for the additional permeability functions within these upper and lower bounds are not discussed in the main body of the text, but are described here.

Franciscan Assemblage:

In addition to the equation from *Manning and Ingebritsen* [1999] derived for crustal rocks undergoing metamorphic dehydration (equation A1, “High”), I also assigned permeability in the Franciscan Assemblage using the same function but one order of magnitude lower, referred to as “Medium” (equation A2).

\[ k = -3.2 \times \log(depth(km)) - 14; \quad k \leq 10^{-14} \text{ m}^2 \quad \text{“High”} \quad (A1) \]
\[ k = -3.2 \times \log(depth(km)) - 15; \quad k \leq 10^{-15} \text{ m}^2 \quad \text{“Medium”} \quad (A2) \]

Thus, the three permeability cases I considered for the Franciscan Assemblage were “High,” “Medium,” and “Low” (Table A1, Figure A1).
Great Valley Sequence:

In addition to the “High” and “Low” cases for the permeability of the Great Valley Sequence used in the main text of this thesis (Table A1, Figure A1), I also considered four more cases (Figure A1). First, I characterized the permeability of the Great Valley Sequence using a permeability-effective pressure relationship described by equation A3, and specified the values of the parameters in the equation based on reported values of permeability for the Great Valley Sequence and laboratory-derived permeability results for low porosity sandstone (Table A1).

\[
k = k_0 \times e^{(-y(P_{\text{eff}}-P_0))}
\]

where \(k\) is the permeability at effective pressure \(P_{\text{eff}}\), \(k_0\) is the permeability at atmospheric pressure \(P_0\), and \(y\) is a sensitivity coefficient describing the compaction-induced decrease in permeability with increasing effective pressure [David et al., 1994]. I assumed an increase in effective pressure of 16 MPa per km, corresponding to hydrostatic pore pressure conditions and a rock density of 2.6 g/cm³. In the simulations, I used \(10^{-15}\) m² for \(k_0\), based on permeability values of Great Valley Sequence sandstone [McLean, 1981]. I used \(3.8 \times 10^{-2}\) MPa⁻¹ for the sensitivity coefficient \(y\), corresponding to low porosity sandstones [Yale, 1984] and refer to this permeability scheme as “Medium 1” when referring to the permeability of the Great Valley Sequence (Figure A1, Table A1). Second, I used \(6.3 \times 10^{-2}\) MPa⁻¹ for the sensitivity coefficient \(y\) [Yale, 1984], and refer to this permeability scheme as “Medium 2” when referring to the permeability of the Great Valley Sequence (Figure A1, Table A1). The final two cases I considered for the permeability of the Great Valley Sequence were homogeneous values of \(10^{-16}\) and \(10^{-17}\) m².
San Andreas Fault Damage Zone:

I considered an additional permeability scenario, termed “Low,” in which I assigned the permeability of the entire damage zone at $10^{-20}$ m$^2$. Note this is not compatible with the definition of a damage zone, but I included this permeability scenario in several simulations.

**Figure A1.** All permeability cases assigned to the Salinian block, Franciscan Assemblage, and Great Valley Sequence in the Parkfield simulations. Refer to Table A1 for sources.
Table A1. Permeability and Porosity of Hydrologic Units in the Parkfield Area

<table>
<thead>
<tr>
<th>Unit / Formation</th>
<th>Permeability (m²)</th>
<th>Porosity</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Tertiary Sediments</strong></td>
<td><strong>OBSERVED:</strong> $k = 4 \times 10^{-13}$ (Santa Margarita), up to $8 \times 10^{-12}$ (Etcheogoin), average $6 \times 10^{-16}$ (Montery), $2 \times 10^{-14}$ to $4 \times 10^{-12}$ (Temblor)</td>
<td><strong>OBSERVED:</strong> 30-35%</td>
<td>1-11</td>
</tr>
<tr>
<td><strong>MODELED:</strong></td>
<td>$k = 10^{-15}$ to $10^{-20}$ (isotropic)</td>
<td><strong>MODELED:</strong> 35%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\kappa_{x} = 10^{-14}$ - $10^{-17}$, $\kappa_{y} = 10^{-15}$ - $10^{-17}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2. Salinian Granite</strong></td>
<td><strong>OBSERVED:</strong> $k = 10^{-15}$ - $10^{-18}$, 0 ≤ $z$ ≤ 2 km below land surface (bls); $k$ decreases by ~ 1 order of magnitude per km depth</td>
<td><strong>OBSERVED:</strong> 1-2% (matrix)</td>
<td>12-17</td>
</tr>
<tr>
<td><strong>MODELED:</strong></td>
<td>Low: $k = 10^{-20}$</td>
<td><strong>MODELED:</strong> 5%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High: $k = 10^{-15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$z$ ≤ 1 km bls;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$k = 10^{-15}$ - $(z - 1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 ≤ $z$ ≤ 6 km bls;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$k = 10^{-20}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$z$ ≥ 6 km bls</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3. Franciscan Assemblage</strong></td>
<td><strong>OBSERVED:</strong> not available</td>
<td><strong>OBSERVED:</strong> max 6.2% typically &lt; 2%</td>
<td>18-19</td>
</tr>
<tr>
<td><strong>MODELED:</strong></td>
<td>Low: $k = 10^{-30}$</td>
<td><strong>MODELED:</strong> 5%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High: $k = -3.2 \times \log(z \text{ (km)}) - 14$, $k ≤ 10^{-14}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Medium: $k = -3.2 \times \log(z \text{ (km)}) - 15$, $k ≤ 10^{-15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4. Great Valley Sequence</strong></td>
<td><strong>OBSERVED:</strong> $k ≤ 10^{-15}$ from surface exposures</td>
<td><strong>OBSERVED:</strong> average 10%</td>
<td>20-26</td>
</tr>
<tr>
<td><strong>MODELED:</strong></td>
<td>Low: $k = 10^{-20}$</td>
<td><strong>MODELED:</strong> 10%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High: $k = 10^{-15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Medium 1: $y = 3.8 \times 10^{-2} \text{ MPa}^{-1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Medium 2: $y = 6.3 \times 10^{-7} \text{ MPa}^{-1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$k = k_{o} \times \exp \left[ [y \times (P_{eff} - P_{o}) \right]$, where $k$ is permeability at effective pressure $P_{eff}$, $k_{o}$ is permeability at atmospheric pressure $P_{o}$, and $y$ is a sensitivity coefficient describing a compaction-induced decrease in permeability with increasing effective pressure $k = 10^{-16}$, $k = 10^{-17}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5. San Andreas Fault Damage Zone</strong></td>
<td><strong>OBSERVED:</strong> not available</td>
<td><strong>OBSERVED:</strong> 2-15%</td>
<td>16-17, 27-29</td>
</tr>
<tr>
<td><strong>MODELED:</strong></td>
<td>10x: $k = (k \text{ _ country _ rock}) \times 10$</td>
<td><strong>MODELED:</strong> 10%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100x: $k = (k \text{ _ country _ rock}) \times 100$, $k ≤ 10^{-14}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Low: $k = 10^{-20}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>6. San Andreas Fault Core</strong></td>
<td><strong>OBSERVED:</strong> not available</td>
<td><strong>OBSERVED:</strong> not available</td>
<td>30</td>
</tr>
<tr>
<td><strong>MODELED:</strong></td>
<td>$k = 10^{-20}$</td>
<td><strong>MODELED:</strong> 10%</td>
<td></td>
</tr>
</tbody>
</table>

Tables A2-A4 list the standard deviation in simulated surface heat flow 150-300 m below the land surface, and the simulated mean required recharge in the Coast Ranges section of the Parkfield model (0-64 km along the topographic profile) and in the western Mojave Desert section of the Mojave model (61 km to the end of the topographic profile) for all simulations in this study. The simulation numbers in this table are different than the simulation numbers in Table 3 in the main text.

### Table A2. Standard Deviation of Simulated Surface Heat Flow, and Simulated Recharge for Parkfield Simulations

<table>
<thead>
<tr>
<th>Parkfield, no heat on San Andreas Fault</th>
<th>PERMEABILITIES OF UNITS IN MODEL (m²)</th>
<th>Salinian Granite</th>
<th>Franciscan Assemblage</th>
<th>Great Valley Sequence</th>
<th>Damage Zone</th>
<th>Fault Core</th>
<th>Standard Deviation of Heat Flow (mW/m²)</th>
<th>Simulated Recharge (cm/yr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number in Appendix</td>
<td>Salinian Tertiary Sediments</td>
<td>granite</td>
<td>franciscan assemblage</td>
<td>great valley sequence</td>
<td>damage zone</td>
<td>fault core</td>
<td>standard deviation of heat flow (mW/m²)</td>
<td>simulated recharge (cm/yr)</td>
</tr>
<tr>
<td>1</td>
<td>10⁻²⁰</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>10⁻¹⁸</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.2</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>5×10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>1.0</td>
<td>0.08</td>
</tr>
<tr>
<td>5</td>
<td>10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>2.1</td>
<td>0.16</td>
</tr>
<tr>
<td>6</td>
<td>2×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>4.1</td>
<td>0.32</td>
</tr>
<tr>
<td>7</td>
<td>3×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>6.1</td>
<td>0.49</td>
</tr>
<tr>
<td>8</td>
<td>4×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>8.0</td>
<td>0.65</td>
</tr>
<tr>
<td>9</td>
<td>5×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>9.9</td>
<td>0.81</td>
</tr>
<tr>
<td>10</td>
<td>7×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>13.5</td>
<td>1.14</td>
</tr>
<tr>
<td>11</td>
<td>8×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>15.1</td>
<td>1.30</td>
</tr>
<tr>
<td>12</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>18.3</td>
<td>1.62</td>
</tr>
<tr>
<td>13</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>absent</td>
<td>absent</td>
<td>18.4</td>
<td>1.61</td>
</tr>
<tr>
<td>14</td>
<td>10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>absent</td>
<td>absent</td>
<td>2.1</td>
<td>.16</td>
</tr>
<tr>
<td>15</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>absent</td>
<td>absent</td>
<td>0.3</td>
<td>0.02</td>
</tr>
<tr>
<td>16</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
<td>10⁻¹⁶</td>
<td>absent</td>
<td>absent</td>
<td>18.3</td>
<td>1.63</td>
</tr>
<tr>
<td>17</td>
<td>10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>10⁻¹⁶</td>
<td>absent</td>
<td>absent</td>
<td>2.1</td>
<td>1.16</td>
</tr>
<tr>
<td>18</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>10⁻¹⁶</td>
<td>absent</td>
<td>absent</td>
<td>0.2</td>
<td>0.02</td>
</tr>
<tr>
<td>19</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
<td>10⁻¹⁷</td>
<td>absent</td>
<td>absent</td>
<td>18.3</td>
<td>1.62</td>
</tr>
<tr>
<td>20</td>
<td>10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>10⁻¹⁷</td>
<td>absent</td>
<td>absent</td>
<td>2.1</td>
<td>0.16</td>
</tr>
<tr>
<td>21</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>10⁻¹⁷</td>
<td>absent</td>
<td>absent</td>
<td>0.2</td>
<td>0.02</td>
</tr>
<tr>
<td>22</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
<td>medium 1</td>
<td>absent</td>
<td>absent</td>
<td>18.4</td>
<td>1.62</td>
</tr>
<tr>
<td>23</td>
<td>10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>medium 1</td>
<td>absent</td>
<td>absent</td>
<td>2.1</td>
<td>1.16</td>
</tr>
<tr>
<td>24</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>medium 1</td>
<td>absent</td>
<td>absent</td>
<td>0.3</td>
<td>0.02</td>
</tr>
<tr>
<td>25</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
<td>medium 2</td>
<td>absent</td>
<td>absent</td>
<td>18.3</td>
<td>1.62</td>
</tr>
<tr>
<td>26</td>
<td>10⁻¹⁶</td>
<td>low</td>
<td>low</td>
<td>medium 2</td>
<td>absent</td>
<td>absent</td>
<td>2.1</td>
<td>1.16</td>
</tr>
<tr>
<td>27</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
<td>medium 2</td>
<td>absent</td>
<td>absent</td>
<td>0.2</td>
<td>0.02</td>
</tr>
</tbody>
</table>

continued on next page)
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>$10^{-15}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>10x</td>
<td>absent</td>
<td>21.3</td>
</tr>
<tr>
<td>29</td>
<td>$10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>10x</td>
<td>absent</td>
<td>3.0</td>
</tr>
<tr>
<td>30</td>
<td>$10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>10x</td>
<td>absent</td>
<td>0.3</td>
</tr>
<tr>
<td>31</td>
<td>$10^{-15}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>100x</td>
<td>absent</td>
<td>20.3</td>
</tr>
<tr>
<td>32</td>
<td>$10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>100x</td>
<td>absent</td>
<td>12.0</td>
</tr>
<tr>
<td>33</td>
<td>$10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>100x</td>
<td>absent</td>
<td>0.5</td>
</tr>
<tr>
<td>34</td>
<td>$10^{-15}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>17.7</td>
</tr>
<tr>
<td>35</td>
<td>$10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>2.0</td>
</tr>
<tr>
<td>36</td>
<td>$10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>0.2</td>
</tr>
<tr>
<td>37</td>
<td>$10^{-15}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>10x</td>
<td>low</td>
<td>18.5</td>
</tr>
<tr>
<td>38</td>
<td>$10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>10x</td>
<td>low</td>
<td>2.1</td>
</tr>
<tr>
<td>39</td>
<td>$10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>10x</td>
<td>low</td>
<td>0.2</td>
</tr>
<tr>
<td>40</td>
<td>$10^{-15}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>100x</td>
<td>low</td>
<td>19.3</td>
</tr>
<tr>
<td>41</td>
<td>$10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>100x</td>
<td>low</td>
<td>5.3</td>
</tr>
<tr>
<td>42</td>
<td>$10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>100x</td>
<td>low</td>
<td>0.0</td>
</tr>
<tr>
<td>43</td>
<td>$10^{-17}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>44</td>
<td>$10^{-16}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>45</td>
<td>$2\times10^{-16}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>46</td>
<td>$4\times10^{-16}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>47</td>
<td>$6\times10^{-16}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>48</td>
<td>$8\times10^{-16}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>49</td>
<td>$10^{-15}$</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>50</td>
<td>$10^{-15}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>51</td>
<td>$10^{-16}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>52</td>
<td>$10^{-17}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>absent</td>
<td>absent</td>
</tr>
<tr>
<td>53</td>
<td>$10^{-15}$</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>21.0</td>
</tr>
<tr>
<td>54</td>
<td>$10^{-16}$</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>3.4</td>
</tr>
<tr>
<td>55</td>
<td>$10^{-17}$</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.5</td>
</tr>
<tr>
<td>56</td>
<td>$10^{-15}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>10x</td>
<td>absent</td>
</tr>
<tr>
<td>57</td>
<td>$10^{-16}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>10x</td>
<td>absent</td>
</tr>
<tr>
<td>58</td>
<td>$10^{-17}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>10x</td>
<td>absent</td>
</tr>
<tr>
<td>59</td>
<td>$10^{-15}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>100x</td>
<td>absent</td>
</tr>
<tr>
<td>60</td>
<td>$10^{-16}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>100x</td>
<td>absent</td>
</tr>
<tr>
<td>61</td>
<td>$10^{-17}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>100x</td>
<td>absent</td>
</tr>
<tr>
<td>62</td>
<td>$10^{-15}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>63</td>
<td>$10^{-16}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>64</td>
<td>$10^{-17}$</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>low</td>
<td>low</td>
</tr>
</tbody>
</table>

(continued on next page)
<table>
<thead>
<tr>
<th></th>
<th>$10^{-15}$</th>
<th>$10^{-16}$</th>
<th>$10^{-17}$</th>
<th>$10^{-15}$</th>
<th>$10^{-16}$</th>
<th>$10^{-17}$</th>
<th>$10^{-15}$</th>
<th>$10^{-16}$</th>
<th>$10^{-17}$</th>
<th>$10^{-15}$</th>
<th>$10^{-16}$</th>
<th>$10^{-17}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>10x</td>
<td>low</td>
<td>22.8</td>
<td>2.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>10x</td>
<td>low</td>
<td>4.2</td>
<td>0.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>10x</td>
<td>low</td>
<td>0.8</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>100x</td>
<td>low</td>
<td>10.9</td>
<td>0.91</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>100x</td>
<td>low</td>
<td>12.4</td>
<td>1.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>2</td>
<td>100x</td>
<td>low</td>
<td>4.2</td>
<td>0.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>$10^{-20}$</td>
<td>$10^{-20}$</td>
<td>$10^{-20}$</td>
<td>absent</td>
<td>absent</td>
<td>0.0</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>$10^{-18}$</td>
<td>absent</td>
<td>absent</td>
<td>0.1</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>$10^{-17}$</td>
<td>absent</td>
<td>absent</td>
<td>0.5</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>$5\times10^{-17}$</td>
<td>$5\times10^{-17}$</td>
<td>$5\times10^{-17}$</td>
<td>absent</td>
<td>absent</td>
<td>2.3</td>
<td>0.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>4.6</td>
<td>0.17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>$2\times10^{-16}$</td>
<td>$2\times10^{-16}$</td>
<td>$2\times10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>9.0</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>$3\times10^{-16}$</td>
<td>$3\times10^{-16}$</td>
<td>$3\times10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>13.3</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>$4\times10^{-16}$</td>
<td>$4\times10^{-16}$</td>
<td>$4\times10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>17.4</td>
<td>0.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>$5\times10^{-16}$</td>
<td>$5\times10^{-16}$</td>
<td>$5\times10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>21.3</td>
<td>0.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>$7\times10^{-16}$</td>
<td>$7\times10^{-16}$</td>
<td>$7\times10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>28.3</td>
<td>1.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>$8\times10^{-16}$</td>
<td>$8\times10^{-16}$</td>
<td>$8\times10^{-16}$</td>
<td>absent</td>
<td>absent</td>
<td>31.6</td>
<td>1.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>$10^{-15}$</td>
<td>absent</td>
<td>absent</td>
<td>37.3</td>
<td>1.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>$k_0 = 10^{-14}$, $k_r = 10^{-15}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>31.4</td>
<td>5.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>$k_0 = 10^{-14}$, $k_r = 10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>4.7</td>
<td>1.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>$k_0 = 10^{-14}$, $k_r = 10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.8</td>
<td>0.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>$k_0 = 10^{-15}$, $k_r = 8\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>16.7</td>
<td>1.46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>$k_0 = 10^{-15}$, $k_r = 5\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>13.6</td>
<td>1.17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>$k_0 = 10^{-15}$, $k_r = 3\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>10.3</td>
<td>0.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>$k_0 = 10^{-15}$, $k_r = 2\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>7.8</td>
<td>0.74</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>$k_0 = 10^{-15}$, $k_r = 10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>4.6</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>$k_0 = 10^{-15}$, $k_r = 10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.5</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>$k_0 = 10^{-15}$, $k_r = 10^{-18}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.1</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>$k_0 = 5\times10^{-16}$, $k_r = 3\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>7.8</td>
<td>0.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>$k_0 = 5\times10^{-16}$, $k_r = 2\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>6.3</td>
<td>0.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>$k_0 = 5\times10^{-16}$, $k_r = 10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>4.0</td>
<td>0.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>$k_0 = 3\times10^{-16}$, $k_r = 2\times10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>5.0</td>
<td>0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>$k_0 = 3\times10^{-16}$, $k_r = 10^{-16}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>3.5</td>
<td>0.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>$k_0 = 3\times10^{-16}$, $k_r = 10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.5</td>
<td>0.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>$k_0 = 10^{-16}$, $k_r = 10^{-17}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.5</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>$k_0 = 10^{-16}$, $k_r = 10^{-18}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.1</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>$k_0 = 10^{-17}$, $k_r = 10^{-18}$</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>absent</td>
<td>absent</td>
<td>0.1</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table A3. Standard Deviation of Simulated Surface Heat Flow, and Simulated Recharge for Parkfield Simulations Incorporating 8.85 mW/m² per km Frictional Heat Generation on the San Andreas Fault

<table>
<thead>
<tr>
<th>Number in Appendix</th>
<th>PERMEABILITIES OF UNITS IN MODEL (m²)</th>
<th>Standard Deviation of Heat Flow (mW/m²)</th>
<th>Simulated Recharge (cm/yr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tertiary Sediments</td>
<td>Salianian Granite</td>
<td>Franciscan Assemblage</td>
</tr>
<tr>
<td>110</td>
<td>10⁻¹⁵</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>111</td>
<td>3×10⁻¹⁶</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>112</td>
<td>10⁻¹⁷</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>113</td>
<td>10⁻¹⁵</td>
<td>10⁻¹⁵</td>
<td>10⁻¹⁵</td>
</tr>
<tr>
<td>114</td>
<td>3×10⁻¹⁶</td>
<td>3×10⁻¹⁶</td>
<td>3×10⁻¹⁶</td>
</tr>
<tr>
<td>115</td>
<td>10⁻¹⁷</td>
<td>10⁻¹⁷</td>
<td>10⁻¹⁷</td>
</tr>
</tbody>
</table>
**Table A4.** Standard Deviation of Simulated Surface Heat Flow, and Simulated Recharge for Mojave Simulations

<table>
<thead>
<tr>
<th>Number in Appendix</th>
<th>PERMEABILITY OF UNITS IN MODEL (m²)</th>
<th>Crystalline Basement</th>
<th>Standard Deviation (mW/m²)</th>
<th>Recharge (cm/yr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10⁻²⁰</td>
<td>10⁻²⁰</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>10⁻¹⁷</td>
<td>10⁻²⁰</td>
<td>0.0</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>10⁻¹⁵</td>
<td>10⁻²⁰</td>
<td>0.1</td>
<td>0.09</td>
</tr>
<tr>
<td>4</td>
<td>10⁻¹⁴</td>
<td>10⁻²⁰</td>
<td>1.1</td>
<td>0.87</td>
</tr>
<tr>
<td>5</td>
<td>10⁻¹³</td>
<td>10⁻²⁰</td>
<td>9.7</td>
<td>1.02</td>
</tr>
<tr>
<td>6</td>
<td>3×10⁻¹⁵</td>
<td>10⁻²⁰</td>
<td>0.3</td>
<td>0.05</td>
</tr>
<tr>
<td>7</td>
<td>6×10⁻¹⁵</td>
<td>10⁻²⁰</td>
<td>0.6</td>
<td>0.11</td>
</tr>
<tr>
<td>8</td>
<td>2×10⁻¹⁴</td>
<td>10⁻²⁰</td>
<td>2.2</td>
<td>0.35</td>
</tr>
<tr>
<td>9</td>
<td>4×10⁻¹⁴</td>
<td>10⁻²⁰</td>
<td>4.5</td>
<td>0.71</td>
</tr>
<tr>
<td>10</td>
<td>6×10⁻¹⁴</td>
<td>10⁻²⁰</td>
<td>6.5</td>
<td>1.06</td>
</tr>
</tbody>
</table>
Appendix B  Simulated Surface Heat Flow for All Parkfield Simulations

The numbers referencing the Parkfield simulations in the titles on the following plots correspond to the numbers in Tables A2 and A3, under “Number in Appendix.”
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 1-3

- Sediment $k = 10^{-20} \text{ m}^2$
- Sediment $k = 10^{-18} \text{ m}^2$
- Sediment $k = 10^{-17} \text{ m}^2$

Simulated Surface Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 4-6

- Sediment $k = 5 \times 10^{-17} \text{ m}^2$
- Sediment $k = 10^{-16} \text{ m}^2$
- Sediment $k = 2 \times 10^{-16} \text{ m}^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 7-9

- sediment $k = 3 \times 10^{-16}$ m$^2$
- sediment $k = 4 \times 10^{-16}$ m$^2$
- sediment $k = 5 \times 10^{-16}$ m$^2$

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 10-12

- sediment $k = 7 \times 10^{-16}$ m$^2$
- sediment $k = 8 \times 10^{-16}$ m$^2$
- sediment $k = 10^{-15}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 13-15

- sediment $k = 10^{-15} \text{ m}^2$
- sediment $k = 10^{-16} \text{ m}^2$
- sediment $k = 10^{-17} \text{ m}^2$

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 16-18

- sediment $k = 10^{-15} \text{ m}^2$
- sediment $k = 10^{-16} \text{ m}^2$
- sediment $k = 10^{-17} \text{ m}^2$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 19-21

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 22-24
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 25-27

Heat Flow (mW/m²)

Distance Along Topographic Profile (km)

- sediment k = $10^{-15}$ m²
- sediment k = $10^{-16}$ m²
- sediment k = $10^{-17}$ m²
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 28-30

Distance Along Topographic Profile (km) vs Heat Flow (mW/m²)

- sediment $k = 10^{-15}$ m²
- sediment $k = 10^{-16}$ m²
- sediment $k = 10^{-17}$ m²

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 31-33

Distance Along Topographic Profile (km) vs Heat Flow (mW/m²)

- sediment $k = 10^{-15}$ m²
- sediment $k = 10^{-16}$ m²
- sediment $k = 10^{-17}$ m²
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 34-36

Distance Along Topographic Profile (km)

Heat Flow (mW/m²)

- sediment $k = 10^{-15} \text{ m}^2$
- sediment $k = 10^{-16} \text{ m}^2$
- sediment $k = 10^{-17} \text{ m}^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 37-39

- **sediment k = 10^{-15} m^2**
- **sediment k = 10^{-16} m^2**
- **sediment k = 10^{-17} m^2**

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 40-42

- **sediment k = 10^{-15} m^2**
- **sediment k = 10^{-16} m^2**
- **sediment k = 10^{-17} m^2**
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 43-45

- sediment $k = 10^{-17}$ m$^2$
- sediment $k = 10^{-16}$ m$^2$
- sediment $k = 2 \times 10^{-16}$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 46-49

- sediment $k = 4 \times 10^{-16}$ m$^2$
- sediment $k = 6 \times 10^{-16}$ m$^2$
- sediment $k = 8 \times 10^{-16}$ m$^2$
- sediment $k = 10^{-15}$ m$^2$

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 50-52

- sediment $k = 10^{-15}$ m$^2$
- sediment $k = 10^{-16}$ m$^2$
- sediment $k = 10^{-17}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 53-55

- solid line: sediment $k = 10^{-15}$ m$^2$
- dashed line: sediment $k = 10^{-16}$ m$^2$
- dotted line: sediment $k = 10^{-17}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 62-64

- sediment $k = 10^{-15}$ m$^2$
- sediment $k = 10^{-16}$ m$^2$
- sediment $k = 10^{-17}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 65-67

- sediment k = 10^{-15} m^2
- sediment k = 10^{-16} m^2
- sediment k = 10^{-17} m^2

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 68-70

- sediment k = 10^{-15} m^2
- sediment k = 10^{-16} m^2
- sediment k = 10^{-17} m^2
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Homogeneous Permeability Models

- $k = 10^{-20} \text{ m}^2$
- $k = 10^{-18} \text{ m}^2$
- $k = 10^{-17} \text{ m}^2$

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 71-73

- $k = 5 \times 10^{-17} \text{ m}^2$
- $k = 10^{-16} \text{ m}^2$
- $k = 2 \times 10^{-16} \text{ m}^2$
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 77-79

- $k = 3 \times 10^{16} \text{ m}^2$
- $k = 4 \times 10^{16} \text{ m}^2$
- $k = 5 \times 10^{16} \text{ m}^2$

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 80-82

- $k = 7 \times 10^{16} \text{ m}^2$
- $k = 8 \times 10^{16} \text{ m}^2$
- $k = 10^{15} \text{ m}^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 83-85

- sediment $k_h = 10^{-14}$ m$^2$, $k_v = 10^{-15}$ m$^2$
- sediment $k_h = 10^{-14}$ m$^2$, $k_v = 10^{-16}$ m$^2$
- sediment $k_h = 10^{-14}$ m$^2$, $k_v = 10^{-17}$ m$^2$

---

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 86-88

- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 8 \times 10^{-16}$ m$^2$
- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 5 \times 10^{-16}$ m$^2$
- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 3 \times 10^{-16}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 89-92

- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 2 \times 10^{-16}$ m$^2$
- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 10^{-16}$ m$^2$
- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 10^{-17}$ m$^2$
- sediment $k_h = 10^{-15}$ m$^2$, $k_v = 10^{-18}$ m$^2$

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 93-95

- sediment $k_h = 5 \times 10^{-16}$ m$^2$, $k_v = 3 \times 10^{-16}$ m$^2$
- sediment $k_h = 5 \times 10^{-16}$ m$^2$, $k_v = 2 \times 10^{-16}$ m$^2$
- sediment $k_h = 5 \times 10^{-16}$ m$^2$, $k_v = 10^{-16}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 96-98

- sediment $k_h = 3 \times 10^{-16}$ m$^2$, $k_v = 2 \times 10^{-16}$ m$^2$
- sediment $k_h = 3 \times 10^{-16}$ m$^2$, $k_v = 10^{-16}$ m$^2$
- sediment $k_h = 3 \times 10^{-16}$ m$^2$, $k_v = 10^{-17}$ m$^2$

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 99-101

- sediment $k_h = 10^{-16}$ m$^2$, $k_v = 10^{-17}$ m$^2$
- sediment $k_h = 10^{-16}$ m$^2$, $k_v = 10^{-18}$ m$^2$
- sediment $k_h = 10^{-17}$ m$^2$, $k_v = 10^{-18}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 102-104

- sediment $k = 10^{-17}$ m$^2$
- sediment $k = 5 \times 10^{-17}$ m$^2$
- sediment $k = 10^{-16}$ m$^2$

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 105-107

- sediment $k = 2 \times 10^{-16}$ m$^2$
- sediment $k = 3 \times 10^{-16}$ m$^2$
- sediment $k = 5 \times 10^{-16}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 108-109

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 110-112
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 113-115

- homogeneous permeability $10^{-15}$ m$^2$
- homogeneous permeability $3 \times 10^{-16}$ m$^2$
- homogeneous permeability $10^{-17}$ m$^2$
Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 116-118

- sediment $k_h = 5 \times 10^{-15}$, $k_v = 10^{-15}$
- sediment $k_h = 5 \times 10^{-15}$, $k_v = 5 \times 10^{-16}$
- sediment $k_h = 3 \times 10^{-15}$, $k_v = 10^{-15}$

Simulated Heat Flow 150-300 m Below Land Surface;
Parkfield Simulations 119-121

- sediment $k_h = 3 \times 10^{-15}$ m$^2$, $k_v = 3 \times 10^{-16}$ m$^2$
- sediment $k_h = 2.5 \times 10^{-15}$ m$^2$, $k_v = 5 \times 10^{-16}$ m$^2$
- sediment $k_h = 2 \times 10^{-15}$ m$^2$, $k_v = 10^{-15}$

Distance Along Topographic Profile (km)

Heat Flow (mW/m$^2$)
Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 122-124

- sediment $k_h = 1.5 \times 10^{-15} \text{ m}^2$, $k_v = 5 \times 10^{-16} \text{ m}^2$
- sediment $k_h = 1.5 \times 10^{-15} \text{ m}^2$, $k_v = 3 \times 10^{-16} \text{ m}^2$
- sediment $k_h = 9 \times 10^{-16} \text{ m}^2$, $k_v = 3 \times 10^{-16} \text{ m}^2$

Simulated Heat Flow 150-300 m Below Land Surface; Parkfield Simulations 125-126

- sediment $k_h = 6 \times 10^{-16} \text{ m}^2$, $k_v = 3 \times 10^{-16} \text{ m}^2$
- sediment $k_h = 2 \times 10^{-16} \text{ m}^2$, $k_v = 10^{-16} \text{ m}^2$
Appendix D  Simulated Surface Heat Flow for All Mojave Simulations

The numbers referencing the Mojave simulations in the titles on the following plots correspond to the numbers in Table A4, under “Number in Appendix.”

For simulations 4-10, the mesh was not refined enough to adequately model the high spatial gradients in temperature accompanying high flow rates in the sediments between 0 and 25 km along the topographic profile, resulting in negative simulated surface heat flow, which is inaccurate. In my simulations, I did not attempt to further refine the mesh in this section of the model because I didn’t use simulated surface heat flow from this section in the data analysis. Additionally, the low-permeability crystalline rock between the Sierra Madre and San Andreas faults that extends to the base of the model acts as a regional groundwater divide, so advection through the section of the model between 0 and 25 km didn’t impact advection between 61 km and the end of the topographic profile, the section where simulated surface heat flow was used in the data analysis. Additionally, the amount of thermal energy exiting the model at the topographic surface between 61 km and the end of the topographic profile is equivalent to the amount of thermal energy entering the model along this section of the topographic profile at the base. This, along with the insignificant differences between simulated and specified pressure and temperature at the topographic surface, for sediment permeability as high as 10^{-13} \text{ m}^2, demonstrates that the section of the model between 61 km and the end of the topographic surface is stable and accurate.

For simulation 5, where the permeability of the sediments is 10^{-13} \text{ m}^2, a permeability of 10^{-20} \text{ m}^2 was assigned to the sediments southwest of 61 km along the topographic profile.
Simulated Heat Flow 150-300 m Below Land Surface;
Mojave Simulations 6-8

Simulated Heat Flow 150-300 m Below Land Surface;
Mojave Simulations 9-10
Appendix E  Simulated Steady-State Temperature Fields for All Mojave Simulations

The numbers referencing the Mojave simulations in the titles on the following plots correspond to the numbers in Table A4, under “Number in Appendix.”

The steady-state simulated temperature fields for each simulation are not in color in the electronic version of this thesis. Each temperature contour plot has 50 contours, with the warmest temperatures at the base of each plot and the coldest temperatures at the top of each plot. The topographic surface is outlined by the uppermost heavy line in each plot.

In the color version of this appendix, two green, single lines near the top of each plot that cut across the temperature contours denote the depths of 150 and 300 m beneath land surface. These depths correspond to the upper and lower bounds of the temperature-depth profile used to compute simulated surface heat flow, and may be slightly visible in the black and white version of this appendix. A color version of this appendix may be available from the author upon request.
Steady-State Temperature Field;
Mojave Simulation 5
Steady-State Temperature Field; Mojave Simulation 6

Steady-State Temperature Field; Mojave Simulation 7
Appendix F  Sensitivity of Calculated Surface Heat Flow to Thermal Gradient Calculation

Simulated surface heat flow varies with depth in the Parkfield and Mojave models, and also depends on the depth “interval” over which I extract the thermal gradient from the simulated temperature field. In linearly interpolate temperature from the model output and compute a thermal gradient over a chosen interval. For example, if I assume a constant thermal gradient over “intervals” of 100 m, then I would assume a constant thermal gradient over depth spans of 0-100 m, 100-200 m, 200-300 m, etc., and calculate one value of heat flow for each depth span. If I assume a constant thermal gradient over “intervals” of 50 m, then I would assume a constant thermal gradient over depth spans of 0-50 m, 50-100 m, 100-150 m, and so on.

Simulated surface heat flow in the Parkfield simulations is not sensitive to the interval of 150 m used to define thermal gradient. To demonstrate this, I calculate thermal gradient as a function of depth at a recharge zone (x = 20 km along the profile in the Parkfield model), a discharge zone (x = 30 km along the profile in the Parkfield model), and at a zone between recharge and discharge zones (x = 28 km along the profile in the Parkfield model) for Parkfield simulation number 12 (Table A2) assuming constant thermal gradients in intervals of 500 m, 300 m, 150 m, and 50 m, which I then use to calculate surface heat flow (Figure F1). Because the element size in the upper 6 km of the Parkfield model is 150 m, the intervals of 500 m, 300 m, and 150 m contain more than one node, whereas the 50 m interval may contain only one node, and therefore this sensitivity analysis also tests the validity of a linear interpolation of the simulated temperature field.

The similarity of surface heat flow calculated over these various intervals suggests that the linear interpolation of the simulated steady-state temperature field is appropriate, and the 150 m interval used to calculate surface heat flow does not affect the results. The maximum difference in simulated surface heat flow at the recharge zone, discharge zone, and intermediate zone for these various intervals is 3 mW/m², 6.2 mW/m², and 4.3 mW/m², respectively. These potential errors in calculated surface heat flow are less than the error bars associated with the Parkfield heat flow data [Fulton et al., 2004].
The results are also not sensitive to the depths at which temperature was extracted to calculate surface heat flow. For example, for Parkfield simulation number 12, the *maximum* variability in surface heat flow in the upper 500 m of the model is $\sim 20$ mW/m$^2$ (Figure F1), but is generally less than 20 mW/m$^2$. Simulated surface heat flow in the Mojave model is close to the conductive case, and therefore variations in thermal gradient and in heat flow with depth are insignificant.

**Figure F1**
Heat flow as a function of depth at a recharge zone ($x = 20$ km), discharge zone ($x = 30$ km), and zone between recharge and discharge ($x = 28$ km) calculated using thermal gradients assumed constant over intervals of 50 m, 150 m, 300 m, and 500 m for Parkfield simulation number 12.
Appendix G  Matlab Codes

Codename: read_sutra:  This code extracts simulation information from the input and output files (e.g., number and location of nodes, simulated temperature field), and takes information specified in the code by the user (e.g., thermal conductivity, desired type of interpolation) in order to calculate surface heat flow along the topographic surface of the model, calculate recharge and discharge rates, format pressure and temperature for creating contour plots, and calculate how closely simulated temperature and pressure obey the boundary conditions.

Format Requirements

1. Input and output files for simulations must be saved as text files (in SUTRA version 2.1, these files are called SUTRA Main Input and SUTRA Main Output)
2. The topographic correction file needs to be in the same directory as the input and output text files for each simulation, and must be saved as a matlab file (not a text file)
3. Comments and directions are in bold and need to remain commented out when this code is run.

Must Change Before Code Is Run:

1. Names of input and output files (italicized and underlined, remove formatting before code is run)
2. Name of topographic correction file (italicized and underlined, remove formatting before code is run)
3. When calculating surface heat flow, specify which thermal conductivity to use at each location along the topographic profile for the depth range of the computation (150-300 m below land surface). Thus, the surface heat flow calculation is broken up into a series of calculations using indexing to tell matlab at which horizontal positions to use certain bulk thermal conductivities. The indexing will depend on the horizontal scale of the grid – in this code, temperatures are spaced every 100 m in the horizontal direction, so for example 64000 m corresponds to an index number of 640.
4. Column (horizontal position along topographic profile) and row (top of basement) in temperatures_new matrix to calculate thermal gradient corrected for topographic refraction (italicized and underlined, remove formatting before code is run)
5. Intervals used to compute thermal gradient corrected for topographic refraction (italicized and underlined, remove formatting before code is run)
The following are embedded in the code to be specific to the Parkfield and Mojave models and were not changed for post-processing individual simulations.

1. Indexing for Coast Ranges, to find recharge in Coast Ranges
2. Temperature and pressure fields interpolated to 100 m horizontal spacing and 5 m vertical spacing
3. Rows 31-61 in temperatures_new matrix used to compute surface heat flow
4. Bulk thermal conductivities Temperature lapse rate and constant pressure functions used for topographic surface in code, used for computing temperature and pressure boundary errors
5. Elevation datum used for computing head
6. Fluid density function used in SUTRA, used for calculating hydraulic head
7. Desired heat flow (0.078 W/m², for Parkfield) in calculating thermal gradients corrected for topographic refraction

Codename: mapview_vary_distanceapart: For each value of heat flow in the Parkfield heat flow dataset, this code calculates the difference in heat flow between that data point and all other data points, and records the great-circle distance between them. This code can be modified for any dataset in which a parameter varies in map view.

1. Data input matrix must be formatted according to the directions given on the first page of this code. The data input matrix must be loaded into the matlab workspace before this code is run.

Codename: vary_distanceapart: For each value of simulated surface heat flow at a location in the Parkfield model, this code calculates the difference in heat flow between that location and all other locations, and records the distance between them as measured along the topographic profile. This code can be modified for any dataset in which a parameter varies in one-dimension, but locations where there are values of the parameter must be equally spaced throughout the entire dataset.

1. The “data” vector in this code is a simulated surface heat flow vector, and must be loaded as data in the matlab workspace before this code is run.
   Example: data = surface_heat_flow_allsed15
2. Lag is calculated by position – for example, if there are 100 values in the surface heat flow vector, the maximum lag is 99. This lag defined in terms of position must be converted to a distance by using the spacing between values in the surface heat flow vector. For example, if each value in the heat flow vector represents evenly spaced distances of 100 m, then the maximum lag of 99 would
correspond to a distance of 9900 meters. This is not implemented in the code and must be done separately after the code is run.

3. This code only works for evenly spaced data in one-dimension.

---

CODE: read_sutra

```matlab
% read_sutra    Written by Margaret Popek

% --------------- Open Input File ---------------------

cd('location of input file, output file, and topographic correction file')

load topographic_correction topographic_correction  % this is model-specific

model_name = 'name of input file';
filename=[model_name,'.txt'];
fid = fopen(filename);

C = textscan(fid, '%s');  % copies entire input file into a 1 by 1 cell array
fclose(fid);

C_1 = C{1}(1:end,1);  % Transforms cell array into a new cell array consisting of one column. Each row is a word, value, symbol etc. in the input text file separated by a space from the word, value, symbol etc. in the adjacent rows.

% ---------------- Extract Information About Nodes and Elements ---------------------

% Looks for the appearance of the words 'NODE', 'ELEMENT', and 'INCIDENCE' in the input cell array and finds the rows where they are located. Then, uses indexing to extract the number of nodes and elements in the model, node ID numbers and corresponding x-coordinates, y-coordinates, and porosities of all the nodes in the model, as well as element ID numbers and the corresponding node numbers of
```
% each node that makes up each element

ind_node = strfind(C_1,'NODE');
index_node = cellfun(@length,ind_node);
C_1_index_node = find(index_node ~= 0);

ind_element = strfind(C_1,'ELEMENT');
index_element = cellfun(@length, ind_element);
C_1_index_element = find(index_element ~= 0);

C_2_nodes = C_1(C_1_index_node+9:C_1_index_element-2);
number_of_nodes = C_2_nodes(end-8);
number_of_nodes = str2num(number_of_nodes{1}(1,1:end)); % number of nodes in model
C_2_nodes = reshape(C_2_nodes, 9, number_of_nodes);

nodes = C_2_nodes(1,:)';
x_coords = C_2_nodes(3,:)';
y_coords = C_2_nodes(4,:)';
porosity = C_2_nodes(6,:)';

for n = 1:length(nodes);
    nodes(n) = str2num(nodes{n});
    x_coords{n} = str2num(x_coords{n});
    y_coords{n} = str2num(y_coords{n});
    porosity{n} = str2num(porosity{n});
end

nodes = cell2mat(nodes); % ID numbers of all nodes in model
x_coords = cell2mat(x_coords); % x-coordinates of all nodes in model
y_coords = cell2mat(y_coords); % y-coordinates of all nodes in model
porosity = cell2mat(porosity); % porosities of all nodes in model

ind_incidence = strfind(C_1, 'INCIDENCE');
index_incidence = cellfun(@length, ind_incidence);
C_1_index_incidence = find(index_incidence ~= 0);

element_data = C_1(C_1_index_incidence+1:length(C_1));
number_of_elements = str2num(cell2mat(element_data(end-8))); % number of elements in model
element_data = reshape(element_data, 9, number_of_elements);

elements = element_data(1,:)';
node1 = element_data(2,:)';
node2 = element_data(3,:)';
node3 = element_data(4,:)';
node4 = element_data(5,:)';

for n = 1:length(elements);
    elements{n} = str2num(elements{n});
end
node1{n} = str2num(node1{n});
node2{n} = str2num(node2{n});
node3{n} = str2num(node3{n});
node4{n} = str2num(node4{n});
end

elements = cell2mat(elements); % element ID numbers
node1 = cell2mat(node1); % node numbers that make up each element
node2 = cell2mat(node2);
node3 = cell2mat(node3);
node4 = cell2mat(node4);

clear C C_1 ind_node index_node C_1_index_node ind_element
index_element C_1_index_element ans n model_name fid filename

% ----------------- Opens Output File -------------------
model_name = 'name of output file';
filename = ([model_name, '.txt']);
fid = fopen(filename);
C = textscan(fid, '%s'); % copies entire output file into a 1 by 1
cell array
fclose(fid);
C_1 = C{1}(1:end,1); % Transforms cell array into a new cell array
consisting of one column. Each row is a word, value, symbol etc. in
the output text file separated by a space from the word, value,
symbol etc. in the adjacent rows.

% ------ Extract Information About Temperatures and Pressures ------
% Looks for the appearance of the words 'RESULTS' in the output cell
array % and uses it to extract the steady-state temperatures and pressures
% corresponding to each node in the model. Also performs a check to
% that all temperatures and pressures were properly extracted.
ind_results = strfind(C_1, 'RESULTS');
index_results = cellfun(@length, ind_results);
C_1_index_results = find(index_results ~= 0);
C_2_temps =
C_1(C_1_index_results(end)+43:(C_1_index_results(end)+43+(number_of_nodes*2)-1));
C_2_temps = reshape(C_2_temps, 2, number_of_nodes);

temperatures = C_2_temps(2,:)';
nodes_temp = C_2_temps(1,:)';

for n = 1:length(nodes);
    temperatures{n} = str2num(temperatures{n});
    nodes_temp{n} = str2num(nodes_temp{n});
end

temperatures = cell2mat(temperatures); % steady-state temperatures at each node
nodes_temp = cell2mat(nodes_temp); % the node number corresponding to each extracted temperature (appear side by side in the output text file)

check_error = abs(nodes-nodes_temp); % subtracts the node ID number associated with each extracted temperature from the node ID number extracted from the input file
if max(check_error) ~= 0; % if the subtraction does not yield a result of zero, this either means that not all of the temperatures were extracted, or the temperatures were not extracted in order of increasing node ID number
    'CHECK INDEXING' % produces an error message if the subtraction does not yield a result of zero
end

C_2_pressures =
C_1(C_1_index_results(3)+40:(C_1_index_results(3)+40+(number_of_nodes*2)-1));
C_2_pressures = reshape(C_2_pressures, 2, number_of_nodes);
pressures = C_2_pressures(2,:)';

for n = 1:length(nodes);
    pressures{n} = str2num(pressures{n});
end

pressures = cell2mat(pressures); % steady-state pressures for each node in the model

clear C C_2_pressures ind_results C_1_index_results C_2 ans n model_name fid filename nodes_temp check_error

% ---------- Extracts the Energy Flux at the Model Boundary ----------
% Looks for the appearance of the word 'TEMPERATURES' in the output cell
% array and uses indexing to extract the node ID numbers across the top
% boundary and corresponding energy flux and temperature at each boundary node

ind_boundary = strfind(C_1, 'TEMPERATURES');
index_boundary = cellfun(@length, ind_boundary);
C_1_index_boundary = find(index_boundary ~= 0);

C_2_watts = C_1(C_1_index_boundary(end)+4:end-28);
C_2_watts = reshape(C_2_watts, 2,length(C_2_watts)/2);

boundary_nodes = C_2_watts(1,:);'
boundary_watts = C_2_watts(2,:');

for n = 1:length(boundary_nodes);
    boundary_nodes{n} = str2num(boundary_nodes{n});
    boundary_watts{n} = str2num(boundary_watts{n});
end

boundary_nodes = cell2mat(boundary_nodes); % node ID numbers of boundary nodes
boundary_watts = cell2mat(boundary_watts); % energy flux (Watts) at each boundary node
temps_boundary = temperatures(boundary_nodes); % model temperature at each boundary node
porosity = porosity(boundary_nodes); % redefines the porosity vector as the porosity at each boundary node, no longer the porosity at each node in the entire model

% ------- Extracts the Fluid Mass Flux Across the Top Boundary -------

% Looks for the appearance of the word 'PRESSURES' in the output cell array
% and uses it to extract the fluid mass flux across the top boundary

ind_outflows = strfind(C_1, 'PRESSURES');
index_outflows = cellfun(@length, ind_outflows);
C_1_index_outflows = find(index_outflows ~= 0);

water_in = C_1(C_1_index_outflows(end)+4 : (C_1_index_outflows(end)+4)+(length(boundary_nodes)*2)-1);
water_in = reshape(water_in, 2, length(boundary_nodes));
water_in = water_in(2,:');

for n = 1:length(water_in);
water_in{n} = str2num(water_in{n});
end
water_in = cell2mat(water_in);
recharge = water_in; \% meters cubed per second

clear C_1_index_boundary index_boundary ind_boundary C_2_watts n

%%%%%%%%%%%%%%%% Organizes Information About Boundary Nodes %%%%%%%%%%%%%%%

boundary_coords = [x_coords(boundary_nodes),y_coords(boundary_nodes),boundary_nodes,boundary_watts, temps_boundary, porosity, recharge];
boundary_coords = sortrows(boundary_coords,1); \% sorts x-coordinates, y-coordinates, boundary node ID number, energy flux at boundary, porosity of boundary nodes, and fluid mass flux at boundary according to increasing x-coordinate

% Boundary node information is now in order of increasing x-coordinate,
% useful for plotting and making calculations that are location-dependent
x_coords_boundary = boundary_coords(:,1);
y_coords_boundary = boundary_coords(:,2);
boundary_nodes = boundary_coords(:,3);
boundary_watts = -boundary_coords(:,4);
temps_boundary = boundary_coords(:,5);
porosity = boundary_coords(:,6);
recharge = boundary_coords(:,7);

% - Converts Fluid Mass Flux at Boundary into Recharge and Discharge Rates

boundary_elements_x_distance = zeros(length(boundary_nodes)-1,1);
boundary_elements_y_distance = zeros(length(boundary_nodes)-1,1);
boundary_elements_hypotenuse = zeros(length(boundary_nodes)-1,1);

% Computes the horizontal and vertical distances between boundary nodes and
% uses it to calculate the actual distance along the slope between each
% boundary node
for n = 1:length(boundary_nodes)-1;
    boundary_elements_x_distance(n) = abs(x_coords_boundary(n+1)-x_coords_boundary(n));
end
boundary_elements_y_distance(n) = abs(y_coords_boundary(n+1)-y_coords_boundary(n));
boundary_elements_hypotenuse(n) =
sqrt(((boundary_elements_x_distance(n))^2)+((boundary_elements_y_distance(n))^2));
end

% Uses the actual distance between boundary node to calculate the
% area associated with each energy and fluid mass flux value at each
% boundary node by adding together half the distance of each adjacent
% boundary element
area_for_boundary_nodes(1) = 0.5*boundary_elements_hypotenuse(1);
area_for_boundary_nodes(end) = 0.5*boundary_elements_hypotenuse(end);

for n = 2:length(boundary_nodes)-1;
    area_for_boundary_nodes(n) = 0.5*boundary_elements_hypotenuse(n-1)+(0.5*boundary_elements_hypotenuse(n));
end

setup_recharge = zeros(length(boundary_nodes),1);
volumetric_fluxes = zeros(length(boundary_nodes),1);

setup_recharge(1) = recharge(1)*(1/1000)*(1/(area_for_boundary_nodes(1)*0.5)); % meters per second
setup_recharge(end) = recharge(end)*(1/1000)*(1/(area_for_boundary_nodes(end)*0.5)); % meters per second

for n = 2:length(setup_recharge)-1;
    setup_recharge(n) = recharge(n)*(1/1000)*(1/(area_for_boundary_nodes(n)));
end

all_recharge = (setup_recharge).*((3.15*10^7)*(100)); % all recharge and discharge for model, in cm/yr

ind = find(all_recharge<0); % finds values of the fluid flow rate that are negative, corresponding to discharge values
discharge = all_recharge(ind); % discharge, in cm/yr, for entire model
discharge = abs(discharge);
setup_recharge = all_recharge;  % redefines recharge and discharge for entire model in terms of cm/yr
setup_recharge(ind) = [];  % deletes negative values to obtain recharge, in cm/yr, for entire model

clear ind

all_recharge_CoastRanges = all_recharge(1:640);  % recharge and discharge, cm/yr, for 0-64 km x-coordinates in the model. Indexing is 1:640 because boundary nodes are spaced 100 m apart, according to the topography in the domain input file
ind = find(all_recharge_CoastRanges < 0);
discharge_CoastRanges = all_recharge_CoastRanges(ind);  % discharge for 0-64 km x-coordinates in the model, cm/yr
recharge_CoastRanges = all_recharge_CoastRanges;
recharge_CoastRanges(ind) = [];  % recharge for 0-64 km x-coordinates in the model, in cm/yr

% Extracts Fluid Velocities and Directions for Each Element in the Model

% Looks for the appearance of the word 'CENTROID' in the output cell array
% and uses it to extract fluid velocities and degree of each flow vector
% from the x-axis, with positive being the clockwise direction.
% Then,
% computes the x and y coordinates of the center of each element in the
% model and the corresponding horizontal and vertical fluid velocities

ind_centroid = strfind(C_1,'CENTROID');
index_centroid = cellfun(@length, ind_centroid);
C_1_index_centroid = find(index_centroid ~= 0);
C_2_velocity = C_1(C_1_index_centroid(end-1)+8:(C_1_index_centroid(end-1)+8+(number_of_elements*2)-1));
C_2_velocity = reshape(C_2_velocity, 2, length(C_2_velocity)/2);
fluid_velocities = C_2_velocity(2,:)';

C_2_angles = C_1(C_1_index_centroid(end)+8:
(C_1_index_centroid(end)+8+(number_of_elements*2)-1));
C_2_angles = reshape(C_2_angles, 2, length(C_2_angles)/2);
fluid_degrees_from_x_axis = C_2_angles(2,:)';

for n = 1:number_of_elements;
   fluid_velocities{n} = str2num(fluid_velocities{n});
fluid_degrees_from_x_axis{n} = str2num(fluid_degrees_from_x_axis{n});
end

fluid_velocities = cell2mat(fluid_velocities); % flow velocity in center of each element
fluid_degrees_from_x_axis = cell2mat(fluid_degrees_from_x_axis); % degree of flow vector in center of each element from x-axis

% Computes the x and y coordinates of the centers of each element in the model

element_node1_xcoordinate = zeros(number_of_elements,1);
element_node2_xcoordinate = zeros(number_of_elements,1);
element_node3_xcoordinate = zeros(number_of_elements,1);
element_node4_xcoordinate = zeros(number_of_elements,1);

element_node1_ycoordinate = zeros(number_of_elements,1);
element_node2_ycoordinate = zeros(number_of_elements,1);
element_node3_ycoordinate = zeros(number_of_elements,1);
element_node4_ycoordinate = zeros(number_of_elements,1);

% Finds x and y coordinates for each of the four nodes that make up each element in the model
for n = 1:length(element_node1_xcoordinate);
    element_node1_xcoordinate(n) = x_coords(node1(n));
    element_node1_ycoordinate(n) = y_coords(node1(n));

    element_node2_xcoordinate(n) = x_coords(node2(n));
    element_node2_ycoordinate(n) = y_coords(node2(n));

    element_node3_xcoordinate(n) = x_coords(node3(n));
    element_node3_ycoordinate(n) = y_coords(node3(n));

    element_node4_xcoordinate(n) = x_coords(node4(n));
    element_node4_ycoordinate(n) = y_coords(node4(n));
end

center_of_element_x = zeros(number_of_elements,1);
center_of_element_y = zeros(number_of_elements,1);

% Computes x and y coordinates of the center of each element in the model by determining the average of the x-coordinates of each of the four nodes that make up each element, and the average of the y-coordinates for each of the four nodes that make up each element
for n = 1:number_of_elements;
    center_of_element_x(n) = mean([element_node1_xcoordinate(n),element_node2_xcoordinate(n),element_node3_xcoordinate(n),element_node4_xcoordinate(n)]);
    center_of_element_y(n) = mean([element_node1_ycoordinate(n),element_node2_ycoordinate(n),element_node3_ycoordinate(n),element_node4_ycoordinate(n)]);
end

x_component_flow = zeros(number_of_elements,1);
y_component_flow = zeros(number_of_elements,1);

% Computes the horizontal and vertical velocities in the center of each element in the model
for n = 1:number_of_elements;
    y_component_flow(n) = (sin((fluid_degrees_from_x_axis(n)*(pi/180))))*(fluid_velocities(n)) ;
    x_component_flow(n) = (cos((fluid_degrees_from_x_axis(n)*(pi/180))))*(fluid_velocities(n)) ;
end

clear C_2_velocity C_2_angles

% ---- Interpolates model steady-state temperature and pressure outputs onto a regular grid -------

new_topo = interp1(x_coords_boundary, y_coords_boundary, min(x_coords):100:max(x_coords));
% interpolates the x and y coordinates of the top boundary into a vector with 100 m horizontal spacing

new_temps_boundary = interp1(x_coords_boundary, temps_boundary, min(x_coords):100:max(x_coords));
% interpolates the boundary temperatures into a vector with 100 m horizontal spacing

x = min(x_coords):100:max(x_coords); % defines 100 m spacing for x-coordinates for interpolating the 2D temperature and pressure fields
y = min(y_coords):5:max(y_coords); % defines 5 m spacing for y-coordinates for interpolating the 2D temperature and pressure fields

[XCOORDS, YCOORDS] = meshgrid(x,y); % creates a regular grid with the horizontal and vertical spacings defined above

% Linearly interpolates model steady-state temperature and pressure fields onto
% regular grid defined above
interpolated_temperatures = griddata(x_coords, y_coords, temperatures, XCOORDS, YCOORDS, 'linear');
interpolated_pressures = griddata(x_coords, y_coords, pressures, XCOORDS, YCOORDS, 'linear');

% The above interpolation results in pressure and temperature fields that
% are upside down, so they must be flipped so that row 1 in each matrix
% corresponds to the highest elevations
interpolated_temperatures = flipud(interpolated_temperatures);
interpolated_pressures = flipud(interpolated_pressures);
YCOORDS = flipud(YCOORDS);

% The interpolations above interpolate temperatures and pressures into x
% and y space that are not part of the topographic surface, so these values
% need to be removed. The following loop places a value of 1000000 at any
% position in the interpolated pressure and temperature fields that has a
% y-coordinate higher than the value of the y-coordinate at the topographic
% surface for each x-coordinate so that these values can be recognized.
for n = 1:length(x);
    for m = 1:length(y);
        if YCOORDS(m,n)>new_topo(n);
            YCOORDS(m,n) = 1000000;
            interpolated_temperatures(m,n) = 1000000;
            interpolated_pressures(m,n) = 1000000;
        end
    end
end

topo_index = zeros(1,length(x));
YCOORDS_new = zeros(length(y),length(x));
temperatures_new = zeros(length(y),length(x));
pressures_new = zeros(length(y),length(x));
% Finds the first y-coordinate for each x-coordinate in the interpolated temperature and pressure fields that is not equal to 1000000, which corresponds to the topographic surface since 1000000 is much higher than the pressures and temperature assigned to the topographic surface. The row corresponding to the topographic surface for each x-coordinate (column) is recorded and the interpolated pressures and temperatures for the topographic surface and land beneath are recorded in new matrices. Not that row 1 in the temperatures_new and pressures_new matrices now corresponds to the topographic surface for each column, and not a specific elevation as in the interpolated_presures and interpolated_temperatures matrices.

for n = 1:length(topo_index);
    topo_index(n) = min(find(Y_COORDS(:,n)~=1000000));
    Y_COORDS_new(1:(length(y)-topo_index(n)+1),n)=Y_COORDS(topo_index(n):end,n);
    temperatures_new(1:(length(y)-topo_index(n)+1),n) = interpolated_temperatures(topo_index(n):end,n);
    pressures_new(1:(length(y)-topo_index(n)+1),n) = interpolated_pressures(topo_index(n):end,n);
end

interpolated_x = X_COORDS(1,:); % same as vector x, just a new name
extracted_topo_surface = Y_COORDS_new(1,:); % y-coordinates corresponding to the topographic surface on the regular grid
extracted_temps_boundary = temperatures_new(1,:); % boundary temperatures on the regular grid
extracted_pressures_boundary = pressures_new(1,:); % boundary pressures on the regular grid

% Computes Heat Flow at A Specified Vertical Interval Beneath Land Surface

% Keeping in mind that the regular grid has 5 m vertical spacing and that row 1 is the topographic surface, define the vertical interval to be used to calculate surface heat flow. Rows 31 to 61 correspond to an interval 150-300 m beneath the topographic surface.
first_row_for_surface_heat_flow = 31;
last_row_for_surface_heat_flow = 61;
% Define the bulk saturated thermal conductivity of the geologic units in
% the model. This section is model-specific, depending on which units are
% present in the model.
thermal_conductivity_sediments = (2.9*0.65)+(0.6*0.35);
thermal_conductivity_basement = (2.9*0.95)+(0.6*0.05);
thermal_conductivity_damagezone = (2.9*0.90)+(0.6*0.10);
thermal_conductivity_GVS = (2.9*0.9) + (0.6*0.10);

linear_thermal_gradient = zeros(length(interpolated_x),2);
thermal_gradient = zeros(length(interpolated_x),1);
surface_heat_flow = zeros(length(interpolated_x),1);

% Performs a linear regression on the temperature-depth data for the
% vertical interval above for every horizontal 100 m (corresponding to the
% horizontal spacing on the regular grid) and stores the slope values as
% the thermal gradient
for n = 1:length(interpolated_x);
    linear_thermal_gradient(n,1:2) =
        polyfit(Y_COORDS_new(first_row_for_surface_heat_flow:last_row_for_surface_heat_flow,n),temperatures_new(first_row_for_surface_heat_flow:last_row_for_surface_heat_flow,n),1);
    thermal_gradient(n) = -linear_thermal_gradient(n,1);
end

% Calculates heat flow in the chosen vertical interval by multiplying the
% thermal gradient at location by the bulk saturated thermal conductivity
% of the geologic unit that is present in that interval. This section is
% model-specific and must be changed each time the geometries of the
% geologic units in the models change because bulk saturated thermal
% conductivity is dependent upon porosity. Additionally, care must be
% taken that potentially uncertain values of calculated heat flow
% near porosity changes are
% documented and not necessarily interpreted as real
% For models incorporating the porosity domains of sediments, GVS, and basement:
for n = 1:640;
surface_heat_flow(n) =
thermal_gradient(n).*thermal_conductivity_sediments;
end

surface_heat_flow(641) =
thermal_gradient(641).*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_GVS));
for n = 642:749;
    surface_heat_flow(n) =
thermal_gradient(n).*thermal_conductivity_GVS;
end

surface_heat_flow(750) =
thermal_gradient(750).*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_GVS));
for n = 751:length(interpolated_x);
    surface_heat_flow(n) =
thermal_gradient(n).*thermal_conductivity_sediments;
end

% % For models incorporating the porosity domains of sediments,
% damage zone,
% % GVS, and basement:
% %

% for n = 1:524;
%     surface_heat_flow(n) =
thermal_gradient(n).*thermal_conductivity_sediments;
% end
%
% surface_heat_flow(525) =
thermal_gradient(525)*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_damagezone));
% for n = 526:550;
%     surface_heat_flow(n) =
thermal_gradient(n).*thermal_conductivity_damagezone;
% end
%
% surface_heat_flow(551) =
thermal_gradient(552)*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_damagezone));
% for n = 552:640;
%     surface_heat_flow(n) =
thermal_gradient(n).*thermal_conductivity_sediments;
% end

%
% surface_heat_flow(641) = thermal_gradient(641).*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_GVS));
% for n = 642:749;
%     surface_heat_flow(n) = thermal_gradient(n).*thermal_conductivity_GVS;
% end
%
% surface_heat_flow(750) = thermal_gradient(750).*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_GVS));
% for n = 751:length(interpolated_x);
%     surface_heat_flow(n) = thermal_gradient(n).*thermal_conductivity_sediments;
% end

%% For Mojave:
% for n = 1:257;
%     surface_heat_flow(n) = thermal_gradient(n).*thermal_conductivity_sediments;
% end
%
% surface_heat_flow(258:264) = thermal_gradient(261).*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_basement));
% for n = 265:606;
%     surface_heat_flow(n) = thermal_gradient(n).*thermal_conductivity_basement;
% end
%
% surface_heat_flow(607:614) = thermal_gradient(611).*((0.5*thermal_conductivity_sediments)+(0.5*thermal_conductivity_basement));
% for n = 615:846;
%     surface_heat_flow(n) = thermal_gradient(n).*thermal_conductivity_sediments;
% end
%
% surface_heat_flow(847:860) = thermal_gradient(860).*((0.25*thermal_conductivity_sediments)+(0.75*thermal_conductivity_basement));
% for n = 861:length(interpolated_x);
surface_heat_flow(n) = thermal_gradient(n).*thermal_conductivity_basement;
% end

surface_heat_flow = surface_heat_flow + topographic_correction; % Adds the topographic correction factor to the computed surface heat flow values

% -Computes Potential Errors with Interpolations

  topo_error = abs(new_topo - extracted_topo_surface); % calculates how closely the topographic surface on the regular grid resembles the interpolated topographic surface
  boundary_temps_error = abs(new_temps_boundary - extracted_temps_boundary); % calculates how closely the boundary temperatures on the regular grid resemble the interpolated boundary temperatures.
  % If any of the above numbers are large, it may mean that a value that was not well interpolated was included

  temperatures_obey_boundary = temperatures(boundary_nodes); % same as temps_boundary vector, the actual, non-interpolated boundary temperatures in the model
  pressures_obey_boundary = pressures(boundary_nodes); % the actual, non-interpolated boundary pressures in the model

% -Computes Potential Errors with Numerical Model Solution

  % Computes the boundary temperatures at each boundary node in the actual
  % model. These should be the same values in the model if the same equation
  % was used.
  lapse_rate_temps = zeros(length(boundary_nodes),1);
  for n = 1:length(boundary_nodes);
    lapse_rate_temps(n) = 10 - ((y_coords_boundary(n)./1000)*6.9);
  end

  max_boundary_temp_error = max(abs(lapse_rate_temps - temperatures_obey_boundary))
  % Computes the difference between the boundary temperatures assigned to the
% model and the steady-state temperatures at the boundary to
calculate how
% well the numerical simulation obeyed the temperature boundary
condition.
% This value is displayed on the matlab screen after running this
code.

max_boundary_pressures_error = max(abs(100000-
pressures_obey_boundary))
% Computes the difference between the boundary pressures assigned to
the
% model and the steady-state pressures at the boundary to calculate
how
% well the numerical simulation obeyed the pressure boundary
condition.
% This value is displayed on the matlab screen after running this
code.

% Displays parameters of interest on the matlab screen after running
this
% code

standard_deviation_HF = std(surface_heat_flow)
mean_recharge = mean(setup_recharge)
mean_CoastRanges_recharge = mean(recharge_CoastRanges)
mean_discharge = mean(discharge)
mean_CoastRanges_discharge = mean(discharge_CoastRanges)

% Produces contour plots of head, fluid density, pressures, and
% temperatures

% Computes equivalent freshwater heads for model

elevation_head = Y_COORDS_new+10000; % model extends for 10 km below
sea level and y-coordinates of elevations are with respect to sea
level, so having the datum as 10 km produces an elevation head equal
to elevation above base of model

fluid_density = zeros(length(y),length(x));

fluid density = 1000 - (0.375*(temperatures_new-20)); % corresponds
to temperature-density function used in the model

head = elevation_head + (((pressures_new)-
100000)./(fluid_density.*9.8066)).*(fluid_density./1000));

% Finds locations in pressures_new matrix that correspond to land.
% Remember that since row 1 in this matrix is the topographic surface, the
% numbers of rows in each column that correspond to actual values vary.
% Rows in each column out of the lower bounds of the model have a value of
% 0, left over from the initialization of the pressures_new matrix.

zeros_index = zeros(length(x),1);
for n = 1:length(x);
    ind = find(pressures_new(:,n)~=0);
    zeros_index(n) = max(ind);
end

% Initializes matrices for contouring with values slightly greater than the
% values of each parameter at the topographic surface, since a matrix to be
% contoured must have a value at each location in the grid
head_to_contour = zeros(max(zeros_index),length(x))+11000;
temperatures_to_contour = zeros(max(zeros_index),length(x))+0;
pressures_to_contour = zeros(max(zeros_index),length(x))+100000;
fluid_density_to_contour = zeros(max(zeros_index),length(x))+1020;

% Records each parameters in the matrices to be contoured such that the
% last row of the matrix is the base of the model
for n = 1:length(x);
    head_to_contour(length(y)-zeros_index(n)+1:end,n)=head(1:zeros_index(n),n);
temperatures_to_contour(length(y)-zeros_index(n)+1:end,n)=temperatures_new(1:zeros_index(n),n);
pressures_to_contour(length(y)-zeros_index(n)+1:end,n)=pressures_new(1:zeros_index(n),n);
fluid_density_to_contour(length(y)-zeros_index(n)+1:end,n)=fluid_density(1:zeros_index(n),n);
end

% Matrix must be flipped upside down before contouring
contour(flipud(head_to_contour));

% --- This section of the code computes the thermal gradient corrected for
% topographic refraction for specific locations in the model, and only in
% the sediments above the basement ---------
% The following only need to be performed once and must be commented out
% afterwards.

conductive_temperatures = temperatures_new; % interpolated temperatures in pure conduction model
temps_without_refraction = zeros(length(y),length(x))-1000000;

for n = 1:length(x);
    for m = 1:length(y);
        if conductive_temperatures(m,n) > 0; % remember some of this matrix is filled with zeros
            temps_without_refraction(m,n) = (abs(Y_COORDS_new(m,n) - Y_COORDS_new(1,n))*(0.078/(thermal_conductivity_sediments)))+temperatures_new(1,n);
            % uses boundary temperatures and what the thermal gradient should be
            % with no topographic refraction to calculate what the temperatures
            % should be. Does this for entire column using the thermal conductivity of the sediments, but will cut off at basement
            % in the following steps
            else
                temps_without_refraction(m,n) = 0;
            end
        end
    end
end

% Find temperatures due to refraction

refraction_effect = temps_without_refraction - conductive_temperatures; % Finds value of topographic refraction effect

temperatures_corrected_refraction = temperatures_new + refraction_effect; % Defines topographic correction factor for temperatures. Although there
% is a value for every position in the model, only the values above the
% basement are valid.

% --- end of one-time calculation -----
% Find corrected thermal gradient for specific column in model

pure_conduction_200 = zeros(row,1);

work = temperatures_new(1:row, column) -
conductive_temperatures(1:row, column);
% (current model with topographic refraction) - (pure conduction
model with topographic
% refraction) equals the difference in temperature between the
current
% model and the pure conduction case if no topographic refraction
existed

for n = 1:row;
pure_conduction_200(n) =
temperatures_new(1, column)+((abs(Y_COORDS_new(n, column) -
Y_COORDS_new(1, column)))*0.0372);
% uses boundary temperature and desired thermal gradient / thermal
% conductivity of sediments (which equals 0.0372) to calculate
% temperatures without thermal refraction down to basement
end

new_grad = pure_conduction_200 + work;
% adds the temperature changes due to advection to the temperatures
without
% refraction, resulting in the thermal gradient for the current
model
% without topographic refraction

% Calculates thermal gradient and heat flow using the corrected
thermal gradient.  Heat flow
% calculated using the corrected thermal gradient should equal heat
flow
% calculated by adding the topographic refraction effect obtained
from the
% steady state temperatures of the conduction model.  Also
calculates thermal gradient by varying the interval used to compute
the thermal gradient.

gradient_profile = zeros(row,1)-10;

for n = column;

temporary_column = new_grad; % assigns corrected temperatures a new
name
for m = 1:10:length(temporary_column);
    if m+9 <= length(temporary_column);
        if (temporary_column(m+9)>0); % here, thermal gradient is computed using the temperatures in intervals of 10 rows, or 50 m
            x_poly = polyfit(Y_COORDS_new((m:m+9),1),temporary_column(m:m+9),1);
            gradient_profile(m:m+9) = abs(x_poly(1));
        else
            else
                end
            end
        end
    else
        end
    end
end

new_heat_flow = zeros(row,1);
for n = 1:row;
    new_heat_flow(n) = (gradient_profile(n).*thermal_conductivity_sediments);
end
% mapview_vary_distanceapart
% Calculates change in heat flow versus distance apart from data points scattered in map view

% ------ Prepare Data Beforehand ---------
% Make a matrix called data_matrix with the following columns:
% Column 1: Data Point Numerical ID
% Column 2: Latitude, in seconds
% Column 3: Longitude, in seconds
% Column 4: Heat Flow
% Column 5: Half the Value of Each Errorbar
% Column 6: Heat Flow - Half Value of Errorbar
% Column 7: Heat Flow + Half Value of Errorbar
% Column 8: Elevation

% ---- Extracts Information from data matrix -----

data_points = data_matrix(:,1);
latitude = data_matrix(:,2);
longitude = data_matrix(:,3);
heat_flow = data_matrix(:,4);
error = data_matrix(:,5);
min_value = data_matrix(:,6);
max_value = data_matrix(:,7);

output = zeros(length(data_points)*200,4) -1; % output vector, with more spaces available than needed, to make sure pairs don't get overwritten.

for n = 1:length(data_points); % the number of data points we have
    for p = 1:length(data_points)-1; % the number of pairs per data point
        if (n+p) <= length(data_points); %
            % Calculates great circle distance between each data points in each pair
            try_x = ((abs((90*3600)-latitude(n+p)))/3600)*pi/180;
            try_y = ((abs((90*3600)-latitude(n)))/3600)*pi/180;
            try_z = ((abs(longitude(n)-longitude(n+p)))/3600)*pi/180;
            output(((n*100)+p),1) = (acos(((cos(try_x)*cos(try_y)) + ((sin(try_x)*sin(try_y))*cos(try_z))))*180/pi)*111;
        % distance between data points in a pair

212
output(((n*100)+p),2) = heat_flow(n);
% heat flow at one point in the pair
output(((n*100)+p),3) = heat_flow(n+p);
% heat flow at the other point in the pair
output(((n*100)+p),4) = abs(((heat_flow(n) - heat_flow(n+p))^1));
% difference in heat flow among the points in a pair
end
end
end

ind = find(output(:,1)==-1);  % finds blank rows and deletes them
output(ind,:) = []; 

clear ind

final_matrix = zeros(length(output(:,1)),2) -1;
for n = 1:length(output(:,1));
    final_matrix(n,1) = output(n,1);  % distance apart
    final_matrix(n,2) = output(n,4);  % delta Q
end
CODE: vary_distanceapart

% vary_distanceapart                Written by Margaret Popek

% variability code FOR EVENLY SPACED DATA in one dimension (example
% cross-section)

vary = zeros(length(data)*1500,4)-1; % max. possible number of pairs is
the length of the data.
% give a crazy value to distinguish from real pairs in the end

for j = 1:length(data)-1; % each value is a value of lag. so if there
are 10 data points, the max. lag is 9

    for n = 1:length(data);

        if (n+j <= length(data)); % if data point exists with that
lag

            vary((n*1500)-1499+j,1) = (abs((data(n) - data(n+j))^1));
            vary((n*1500)-1499+j,2) = j;
            vary((n*1500)-1499+j,3) = n; % data point 1
            vary((n*1500)-1499+j,4) = n+j; % data point 2

        end
    end
end

ind = find(vary(:,1)==-1);
vary(ind,:)=[];

% Can plot variability in heat flow as a function of lag using columns
2 % (lag value) and 1 (difference in heat flow) in vary matrix, after
getting
% rid of the blank rows. However, this creates a very large file and
is
% time-consuming for plotting. The following computes the maximum
value of
% variability for each lag value.

vary = sortrows(vary,2); % sorts rows according to increasing lag
distance
test = vary(1,2);  \% initializes value for first run of for loop
old_ind = 1;  \% initializes value for first run of for loop

max_lag = max(vary(:,2));
variability_data = zeros(max_lag*max_lag, 2);  \% check to make sure this is more rows than values of lag for your data

while test<max_lag;
    ind = min(find(vary(:,2)>test));  \% finds first row not equal to value of test lag
    variability_data(old_ind,1) = vary(old_ind,2);  \% puts value of test lag into matrix
    variability_data(old_ind,2) = max(vary(old_ind:ind-1,1));  \% finds maximum variability for test matrix and records
    old_ind = ind;  \% defines row to start next for loop on, this is row corresponding to the next value of lag
    test = vary(ind,2);  \% defines next value of lag
end

\% does this once more for maximum value of lag
ind = min(find(vary(:,2)>test));
variability_data(old_ind,1) = vary(old_ind,2);
variability_data(old_ind,2) = max(vary(old_ind:end,1));

clear ind

ind = find(variability_data(:,1)==0);  \% find the filler rows with no lag values in them
variability_data(ind,:)=[];  \% gets rid of filler rows

\% Variability data column 1 contains the lag values, and variability column
\% 2 contains the corresponding maximum variability values